JPL-Pub-87-13-Vol-3

```
(NASA-CR-184687)  PROCEEDINGS OF THE                  N89-26541
WORKSHOP ON SPACE TELEROBOTICS, VOLUME 3              --THRU--
(Jet Propulsion Lab.)  412 p        CSCL 09B          N89-26577
                                                      Unclas
                                              G3/63   0188169
```

This publication was prepared by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

ABSTRACT

These proceedings report the results of a workshop on space telerobotics, which was held at the Jet Propulsion Laboratory, January 20-22, 1987. Sponsored by the NASA Office of Aeronautics and Space Technology (OAST), the workshop reflected NASA's interest in developing new telerobotics technology for automating the space systems planned for the 1990s and beyond. The workshop provided a window into NASA telerobotics research, allowing leading researchers in telerobotics to exchange ideas on manipulation, control, system architectures, artificial intelligence, and machine sensing. One of the objectives was to identify important unsolved problems of current interest. The workshop consisted of surveys, tutorials, and contributed papers of both theoretical and pratical interest. Several sessions were held with the themes of sensing and perception, control execution, operator interface, planning and reasoning, and system architecture. Discussion periods were also held in each of these major topics.

ACKNOWLEDGMENT

Thanks are due to D.R. Meldrum, R.S. Doshi, K. Kreutz, H. Seraji, M.H. Milman, and S.K. Owen for significant contributions to the organization of the workshop.

CONTENTS

Volume I

PLENARY SESSION 1

Program Objectives and Technology Outreach
P.S. Schenker 3

Building Intelligent Systems: Artificial Intelligence Research at NASA Ames Research Center
P. Friedland and H. Lum 19

SYSTEM CONCEPTS 27

Space Telerobotic Systems: Applications and Concepts
L. Jenkins 29

Laboratory Testing of Candidate Robotic Applications for Space
R.B. Purves 35

Telerobotic Assembly of Space Station Truss Structure
G. Fischer 45

Robotic Mobile Servicing Platform for Space Station
S.H. Lowenthal and L. Van Erden 55

System Engineering Techniques for Establishing Balanced Design and Performance Guidelines for the Advanced Telerobotic Testbed
W.F. Zimmerman and J.R. Matijevic 67

Dedicated Robotic Servicing for the Space Station
R.F. Thompson, G. Arnold, and D. Gutow 75

Shuttle Bay Telerobotics Demonstration
W. Chun and P. Cogeos 83

Telerobotics: Research Needs for Evolving Space Stations
L. Stark 91

Robot Design for a Vacuum Environment
S. Belinski, W. Trento, R. Imani-Shikhabadi, and S. Hackwood . . . 95

Multi-Limbed Locomotion Systems for Space Construction and Maintenance
K.J. Waldron and C.A. Klein 105

Commercial Users Panel 113

SYSTEM ARCHITECTURE 125

Task Allocation Among Multiple Intelligent Robots
L. Gasser and G. Bekey 127

Execution Environment for Intelligent Real-Time Control Systems
J. Sztipanovits 131

Distributed Intelligence for Supervisory Control
W.J. Wolfe and S.D. Raney 139

An Architecture for Heuristic Control of Real-Time Processes
P. Raulefs and P.W. Thorndyke 149

Hierarchical Control of Intelligent Machines
Applied to Space Station Telerobots
J.S. Albus, R. Lumia, and H. McCain 155

Programming Methodology for a General Purpose Automation Controller
M.C. Sturzenbecker, J.U. Korein, and R.H. Taylor 167

Real-Time Hierarchically Distributed Processing Network
Interaction Simulation
W.F. Zimmerman and C. Wu 173

Metalevel Programming in Robotics: Some Issues
A. Kumar N. and N. Parameswaran 181

An Approach to Distributed Execution of ADA Programs
R.A. Volz, P. Krishnan, and R. Theriault 187

Computational Structures for Robotic Computations
C.S.G. Lee and P.R. Chang 199

A Run-Time Control Architecture for the JPL Telerobot
J. Balaram, A. Lokshin, K. Kreutz, and J. Beahan 211

Distributed Control Architecture for Real-Time Telerobotic Operation
H.L. Martin, P.E. Satterlee, Jr., and R.F. Spille 223

High Performance Architecture for Robot Control
E. Byler and J. Peterson 233

Software and Hardware for Intelligent Robots
G.N. Saridis and K.P. Valavanis 241

Nested Hierarchical Controller With Partial Autonomy
A. Meystel 251

AUTONOMOUS SYSTEMS 271

3-D World Modeling With Updating Capability Based on Combinatorial Geometry
M. Goldstein, F.G. Pin, G. de Saussure, and C.R. Weisbin 273

On Autonomous Terrain Model Acquisition by a Mobile Robot
N.S.V. Rao, S.S. Iyengar, and C.R. Weisbin 283

SOLON: An Autonomous Vehicle Mission Planner
M.J. Dudziak 289

Mission Planning for Autonomous Systems
G. Pearson 303

Certainty Grids for Mobile Robots
H.P. Moravec 307

Environmental Modeling and Recognition for an Autonomous Land Vehicle
D.T. Lawton, T.S. Levitt, C.C. McConnell, and P.C. Nelson 313

GEOMETRIC REASONING 337

Telerobot Task Planning and Reasoning: Introduction to JPL Artificial Intelligence Research
D.J. Atkinson 339

Geometric Reasoning
R.F. Woodbury and I.J. Oppenheim 351

A Qualitative Approach for Recovering Relative Depths in Dynamic Scenes
S.M. Haynes and R. Jain 361

A Framework for Qualitative Reasoning About Solid Objects
E. Davis 369

Volume II

SENSING AND PERCEPTION 1

The Sensing and Perception Subsystem of the NASA Research Telerobot
B. Wilcox, D.B. Gennery, B. Bon, and T. Litwin 3

Knowledge-Based Vision for Space Station Object Motion Detection, Recognition, and Tracking
P. Symosek, D. Panda, S. Yalamanchili, and W. Wehner III 9

Sensor Systems Testbed for Telerobotic Navigation
A.W. Thiele, D.E. Gjellum, R.H. Rattner, and D. Manouchehri 19

Monovision Techniques for Telerobots
P.W. Goode and K. Cornils 23

Tracking 3-D Body Motion for Docking and Robot Control
M. Donath, B. Sorensen, G.B. Yang, and R. Starr 31

Object Apprehension Using Vision and Touch
R. Bajcsy and S.A. Stansfield 45

Sensory Substitution for Space Gloves and for Space Robots
P. Bach-y-Rita, J.G. Webster, W.J. Tompkins, and T. Crabb 51

Electronic Prototyping
J. Hopcroft 59

Multiple Degree of Freedom Optical Pattern Recognition
D. Casasent 65

Maximum Likelihood Estimation of Parameterized 3-D Surfaces
Using a Moving Camera
Y. Hung, B. Cernuschi-Frias, and D.B. Cooper 71

The Architecture of a Video Image Processor for the Space Station
S. Yalamanchili, D. Lee, K. Fritze, T. Carpenter, K. Hoyme,
and N. Murray 85

An Optical Processor for Object Recognition and Tracking
J. Sloan and S. Udomkesmalee 95

Improved Shape-Signature and Matching Methods for
Model-Based Robotic Vision
J.T. Schwartz and M.J. Wolfson 103

A Technique for 3-D Robot Vision for Space Applications
V. Markandey, H. Tagare, and R.J.P. deFigueiredo 111

Differential Surface Models for Tactile Perception of Shape
and On-Line Tracking of Features
H. Hemami 125

Constraint-Based Stereo Matching
D.T. Kuan 131

A Database/Knowledge Structure for a Robotics Vision System
D.W. Dearholt and N.N. Gonzales 135

Sensing and Perception: Connectionist Approaches to
"Subcognitive" Computing
J. Skrzypek 145

Parallel Processing for Digital Picture Comparison
H.D. Cheng and L.T. Kou 155

ROBOT SIMULATION AND CONTROL 163

Inverse Kinematic-Based Robot Control
W.A. Wolovich and K.F. Flueckiger 165

Spatially Random Models, Estimation Theory, and Robot Arm Dynamics
G. Rodriguez 177

Task Oriented Nonlinear Control Laws for Telerobotic
Assembly Operations
R.A. Walker, L.S. Ward, and C.F. Elia 185

A Universal Six-Joint Robot Controller
D.G. Bihn and T.C. Hsia 197

Real-Time Graphic Simulation for Space Telerobotics Applications
E.W. Baumann 207

Manipulator Control and Mechanization: A Telerobot Subsystem
S. Hayati and B. Wilcox 219

Chaos Motion in Robot Manipulators
A. Lokshin and M. Zak 229

Effect of Control Sampling Rates on Model-Based Manipulator
Control Schemes
P.K. Khosla 235

Kinematically Redundant Robot Manipulators
J. Baillieul, R. Brockett, J. Hollerbach, D. Martin, R. Percy,
and R. Thomas 245

Control Strategy for a Dual-Arm Maneuverable Space Robot
P.K.C. Wang 257

A Space Systems Perspective of Graphics Simulation Integration
R. Brown, C. Gott, G. Sabionski, and D. Bochsler. 267

A Learning Controller for Nonrepetitive Robotic Operation
W.T. Miller, III. 273

TELEROBOTS 283

A Flexible Telerobotic System for Space Operations
N.O. Sliwa and R.W. Will. 285

Systems Simulations Supporting NASA Telerobotics
F.W. Harrison, Jr. and J.E. Pennington 293

Coordination of Multiple Robot Arms
L.K. Barker and D. Soloway 301

Development of a Semi-Autonomous Service Robot With Telerobotic Capabilities
J.E. Jones and D.R. White 307

A Task-Based Metric for Telerobotic Performance Assessment
J.F. Barnes 317

Motion Coordination and Programmable Teleoperation Between Two Industrial Robots
J.Y.S. Luh and Y.F. Zheng 325

Use of Control Umbilicals as a Deployment Mode for Free Flying Telerobotic Work Systems
J.S. Kuehn and E.D. Selle 335

The Design of a Nine-String Six-Degree-of-Freedom Force-Feedback Joystick for Telemanipulation
M.L. Agronin 341

MAN MACHINE INTERFACE 349

Issues in Human/Computer Control of Dexterous Remote Hands
K. Salisbury 351

Man-Machine Interface Issues in Space Telerobotics: A JPL Research and Development Program
A.K. Bejczy 361

Task-Level Testing of the JPL-OMV Smart End Effector
B. Hannaford 371

A Natural-Language Interface to a Mobile Robot
S. Michalowski, C. Crangle, and L. Liang 381

Types of Verbal Interaction With Instructable Robots
C. Crangle, P. Suppes, and S. Michalowski 393

MIT Research in Telerobotics
T.B. Sheridan and Staff 403

Telerobotics: A Simulation Facility for University Research
L. Stark et al. 413

Mixed-Initiative Control of Intelligent Systems
G.C. Borchardt 423

Report on the Stanford/Ames Direct-Link Space Suit Prehensor
J.W. Jameson 433

Volume III

PLANNING AND SCHEDULING: BASIC RESEARCH AND TOOLS . . . . 1 OMIT

Planning and Scheduling: Is there a Difference?
K.G. Kempf . . . . 3 omit

The Mechanisms of Temporal Inference
B.R. Fox and S.R. Green 9 S1

Contingent Plan Structures for Spacecraft
M. Drummond, K. Currie, and A. Tate 17 S2

Reasoning and Planning in Dynamic Domains:
An Experiment With a Mobile Robot
M.P. Georgeff, A.L. Lansky, and M.J. Schoppers 27 S3

Route Planning in a Four-Dimensional Environment
M.G. Slack and D.P. Miller 41 S4

Prediction and Causal Reasoning in Planning
T. Dean and M. Boddy 49 S5

The Generic Task Toolset: High Level Languages for the Construction of Planning and Problem Solving Systems
B. Chandrasekaran, J. Josephson, and D. Herman 59 S6

Monitoring Robot Actions for Error Detection and Recovery
M. Gini and R. Smith 67 S7

Recovering From Execution Errors in SIPE
D.E. Wilkins 79 S8

Incremental Planning to Control a Blackboard-Based Problem Solver
E.H. Durfee and V.R. Lesser 91 S9

Knowledge Representation System for Assembly Using Robots
A. Jain and M. Donath 101 S10

Real Artificial Intelligence for Real Problems
R.S. Doshi . . . . 115 OMIT

TRAJECTORY PLANNING FOR MANIPULATORS . . . . 117 OMIT

The Use of 3-D Sensing Techniques for On-Line Collision-Free Path Planning
V. Hayward, S. Aubry, and Z. Jasiukajc 119 S11

Collision-Free Trajectory Planning Algorithm for Manipulators
F. Pourboghrat and J.Y. Han 125 S12

Task Planning and Control Synthesis for Robotic Manipulation in Space Applications
A.C. Sanderson, M.A. Peshkin, and L.S. Homem-de-Mello 129 S13

Using Automatic Robot Programming for Space Telerobotics
E. Mazer, J. Jones, A. Lanusse, T. Lozano-Perez, P. O'Donnell, and P. Tournassoud 139 S14

MANIPULATOR CONTROL . . . . 151 omit

Manipulator Control: An Overview
H. Seraji . . . . 153 omit

Adaptive Force-Position Control for Teleoperated Manipulators
A.J. Koivo 155 S15

Adaptive Control of Dual-Arm Robots
H. Seraji 159 S16

Design of a Reconfigurable Modular Manipulator System
D. Schmitz and T. Kanade 171 S17

Nonlinear Feedback Control of Multiple Robot Arms
T.J. Tarn, X. Yun, and A.K. Bejczy 179 S18

Dynamics and Control of Coordinated Multiple Manipulators
S.A. Hayati 193 S19

A Survey of Adaptive Control Technology in Robotics
S. Tosunoglu and D. Tesar 205 S20

Simple Robust Control Laws for Robot Manipulators, Part I: Non-adaptive Case
J.T. Wen and D.S. Bayard 215 S21

Simple Robust Control Laws for Robot Manipulators, Part II: Adaptive Case
D.S. Bayard and J.T. Wen 231 S22

Algorithms for Adaptive Control of Two-Arm Flexible Manipulators Under Uncertainty
J.M. Skowronski 245 S23

Problems and Research Issues Associated With the Hybrid Control of Force and Displacement
R.P. Paul 255 S24

Adaptive Hybrid Control of Manipulators
H. Seraji 261 S25

Adaptive Hybrid Position/Force Control of Robotic Manipulators
F. Pourboghrat 273 S26

Flexible Manipulator Control Experiments and Analysis
S. Yurkovich, Ü. Özgüner, A. Tzes, and P.T. Kotnik 279 S26

Dual Arm Robotic System With Sensory Input
Ü. Özgüner 289 S27

Geometric Foundations of the Theory of Feedback Equivalence
R. Hermann 299 S28

Reducing Model Uncertainty Effects in Flexible Manipulators
Through the Addition of Passive Damping
T.E. Alberts 307 S29

Parallel Processing Architecture for Computing Inverse Differential
Kinematic Equations of the PUMA Arm
T.C. Hsia, G.Z. Lu, and W.H. Han 317 S30

Manipulator Control by Exact Linearization
K. Kreutz 325 S31

A Virtual Manipulator Model for Space Robotic Systems
S. Dubowsky and Z. Vafa 335 S32

Model Reduction for Discrete Bilinear Systems
A.M. King and R.E. Skelton 345 S33

High Gain Feedback and Telerobotic Tracking
D.E. Koditschek 355 S34

Concept Development of a Tendon Arm Manipulator
and Anthropomorphic Robotic Hand
C.T. Tolman 365 S35

Future Research Directions
Moderator: G.A. Bekey

Panel Members: R. Bajcsy, G. Saridis,
J.Y.S. Luh, T.B. Sheridan,
A. Sanderson, C. Weisbin, ... 373

APPENDIX A

Program Schedule 391

APPENDIX B

Index by Author 405

APPENDIX C

Attendees/Participants 409

omit

PLANNING AND SCHEDULING: BASIC RESEARCH AND TOOLS

OMIT TO
P.-17

Planning and Scheduling: Is there a Difference?

K.G. Kempf
FMC Corporation
Santa Clara, CA 95052

1. Abstract

It is proposed that the ability to classify problems by type and solution techniques by applicability is advantageous for any problem solver. Yet no classification schemes are available in the important areas of robotic planning and scheduling. A standard approach to developing schemes based on work reported in the literature is outlined, but is found to be rather unsatisfying. An alternative approach is sketched based upon the uncertainty inherent in dealing with the real world, and is found to capture more of the intuitive differences between planning and scheduling. A trial explanation for the distinctions between planning and scheduling is then offered. The work of three research groups on robotic assembly is given as support for the distinctions drawn.

2. Introduction

Arguments are introduced in this section in the context of providing perspective and motivation for the discussion which follows. The topics include our approaches to the study of artificial intelligence and computer science as well as robotics.

In some fields of study, it is possible, usually after the expenditure of great effort, to synthesize "laws" which describe various phenomena of interest. These laws can be used to explain the results of past experiments or to predict the outcome of future trials. Use to explain and predict can further confirm the accuracy and generality of the laws, or highlight necessary revision and refinement. Eventually the laws can be utilized in the construction of entities to display specific properties in the domain of interest. Examples of this progression can be drawn from the natural sciences (such as physics) and their application disciplines (such as mechanical engineering).

Some researchers subscribe to the belief that inquiries into the nature of intelligence will lead to the discovery of a (small) set of accurate, general laws of intelligence. They believe that such laws, sufficiently refined and understood, might enable the construction of entities which display intelligence and use humans as an existence proof that such entities are possible. Other researchers are not comfortable with the assumption that a set of laws underlies intelligent behavior in a directly analogous fashion to the physical laws which form the foundation of physics. They argue instead that intelligence is inherently based on learning and using a large and diverse bag of highly specialized tricks. Constructing entities which display intelligence can be accomplished by discovering a few of these tricks, encoding them in a computer program, and using them to solve problems. Intelligence can be incrementally improved by adding more tricks.

Which, if either, of these extreme viewpoints is correct remains to be seen. It is proposed, however, that there is a mechanism that could be used in the realization of an intelligent entity which operates anywhere in the spectrum bounded by these extreme views. In computational terms, it consists (conceptually) of two data structures and three processes. The first data structure contains a description of a space of problem types in terms of the feature vocabulary which the system uses to encode problems about which it knows. The second data structure contains a description of the space of solution techniques with which the system is familiar, also encoded in an appropriate vocabulary. The first process maps any problem which the intelligent entity encounters into the problem space for purposes of classification. The second process maps the problem classification intc the solution technique space for the purpose of selecting the representation and reasoning methods to be used in solving the problem. The third process manages the instantiation and execution of the indicated methods to provide a solution.

At the trick-based extreme, the problem type and solution technique descriptions would be very specific, while at the law-based extreme they would be very general. However, the processes at either extreme perform the same tasks, even though they might be quite different in operational detail. In any case, this model suggests that the intelligence of an entity resides in being able to categorize problems (the first process and the first data structure), select the appropriate solution approach (the second process and the second data structure), and produce a solution (the third process). Furthermore, the model provides for various forms of learning by expansion, refinement, and abstraction of the data structures as well as improvement of the mapping processes.

The object of raising this argument is to point out that there seems to be an asymmetric distribution of work reported in the literature on planning and scheduling in the context of the suggested model of intelligent entities. There has been much work aimed at synthesizing powerful general purpose planning mechanisms, and some work done to build special purpose planners and schedulers which solve specific problems. This work corresponds to the second data structure and the third process in the model, respectively. As might be expected from the model, the specific planners are very often specializations of the general planners. Unfortunately, it is seldom (if ever) the case that a clear statement of the characteristics of the problems for which any of the planners or schedulers have been designed is explicitly offered. This is a reflection of the fact that little if any work has been carried out concerning the vocabulary of problem types. Obviously, without the vocabulary, it is also very difficult to make progress on the processes of classifying problem types based on problem features or selecting solution methods based on problem classification. In the context of the model, much work remains on the development of a problem type vocabulary to produce an intelligent entity.

Regardless of arguments based on a speculative model of intelligent entities, there are other justifications for the exploration of problem type and solution technique description vocabularies. There is reason to believe that any problem solver, natural or artificial, human or robotic, benefits from being able to classify problems which it encounters and to evaluate solution approaches to those problems. An outstanding example of the power which can be generated by explicitly analyzing problem and solution spaces can be found in two of the most important concepts of computer science - computability and complexity. The first aids us in the categorization of problems into those which are in principle computable and those which are not computable. The second supplies the metric for deciding whether a solution approach to a computable problem will be relatively efficient (polynomial run time) or relatively inefficient (exponential run time).

It is suggested that this type of analysis, and more, is indicated for the problem type and the solution technique spaces in planning and scheduling. It would be useful for researchers to have a problem description vocabulary though which they could discuss the similarities and differences among problems. Furthermore, to have a map of the known portions of the solution approach space would help developers assess the results of existing research and would help focus new research efforts on the unknown regions. In addition, these two spaces are necessary for the construction of intelligent entities under the model described earlier.

3. A Standard Approach

In an effort to construct an initial description of these spaces for planning types of problems, a partial literature review has been attempted. Planning problems are usually considered to take as input state descriptions, action descriptions, and path constraints, and to produce as output some ordered set of states and actions which obey the path constraints. One form of problem and solution categorization scheme can be based on the existing literature in which this input/output view of planning is taken.

The state descriptions taken as input consist of an initial state and a goal state. A problem categorization scheme can use as one measure the size and complexity of the initial and goal states given. The methodology used to encode the distinguished initial and goal states is also used by the planner to construct trial intermediate states during plan synthesis and to partially describe the plan which the system outputs. It is often the case in robotic problems that state descriptions concern, at some level of abstraction, objects, their properties, and their relationships to other objects. A solution categorization scheme could use as one measure the technique used by the planner to represent states.

The action descriptions taken as input for robotic problems include a catalog of agents and facilities involved along with their capabilities in terms of manipulation and perception. One useful measure for problem categorization is the number and nature of the agents described. In problems involving robots, the laws of physics must always be considered as supplying actions such as gravity which all agents must recognize but which no agent can control. Beyond "mother nature", there can be one agent or many agents. If there are many agents, they must be defined in terms of their relationships (cooperative, neutral, competitive), communications (none, occasional, continuous), and shared information (none, partial, complete), to name but a few of the relevant characteristics. Early planners usually considered single agent planning situations [1, 2, 3]. More recently, many researchers have focused on multiagent planning problems and the closely related area of distributed planning [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17].

The action description methodology used by the planner can also be used in a solution categorization scheme. In robot problems, actions often change the properties of objects or the relations among objects. One useful measure for solution categorization is whether or not action duration is explicitly represented. The early planners did not explicitly consider duration [1, 2, 3], but more recently the representation and use of action duration has been a fertile area of research [18, 19, 20, 21, 22]. Interesting questions involve the use of time in an absolute or relative way and the representation of time as points or intervals.

The path constraints taken as input are used to define the general type of solution path desired and often have the form of restrictions on properties of the initial and goal states. Four categories are easily distinguished as being useful in problem classification. Problems with properties which are present in the initial state and which must also be present in the goal state are preservation problems. Complementary prevention problems have properties which are absent in the initial state and which must also be absent in the goal state. Problems with properties which are present in the initial state but must be absent in the goal state are destruction problems. The complementary construction problems, which have been most studied in the literature, have properties which are absent in the initial state but must be present in the goal state.

Further measurements for the construction of the solution technique space can be found by considering the way in which the planner goes about synthesizing its output [23]. Early planners used weak methods such as means/ends analysis and simple back-chaining to solve rather simple problems. As researchers began to attempt more and more complex problems, more powerful techniques were used. The idea of hierarchical abstraction was used to focus the planner on the hardest parts of the problem first [24, 25]. Next to be considered were conjunct ordering, subgoal interaction, and the nonlinear nature of plans [26, 27, 28, 29, 30, 31, 32, 33, 34]. Later, pre-formed plan segments were built into planners to provide them with a mechanism of avoiding repeatedly planning typical tasks from scratch [35, 36]. Finally, planners have been given meta-level control so that they can plan about planning as well as planning about solving the problem at hand [37, 38]. Related, but alternative views of planning also have been taken. Opposed to the view that the planner should strive to make perfect plans exclusively, one planner proposed making an imperfect plan and debugging it [39]. Another planner took the view that an attempt should first be made to disprove that goals can be reached before attempting to reach those goals [40].

Pursuit of this type of "standard" approach will probably produce a useful classification scheme for planning problem types and solution techniques. Much more work is required, and even then it remains to be seen whether practically useful representations or data structures can be constructed in the context of the intelligent entity model which stimulated this investigation. It is suggested that a similar effort on scheduling problems, that is one based on input/output properties and on representation/reasoning characteristics, would yield similar results. Unfortunately, this gives a set of structures for planning and another set of structures for scheduling, either avoiding the question of the difference between planning and scheduling altogether, or answering that there is a difference based on the narrow input/output definitions used to build the structures. Given that the terms "planning" and "scheduling" cover a very broad set of problems, such a narrow answer is not very satisfying. It seems intuitively obvious that a workable classification scheme should differentiate a set of problems which are purely planning problems, a set which are clearly scheduling problems, and a set which have characteristics of both, or it should show that there is a spectrum of problems none of which are purely planning problems or purely scheduling problems.

It is suggested that two other observations about planning and scheduling problems can help to rectify this sense of dissatisfaction. One concerns the nature of the uncertainties which are encountered in robotic tasks. The other concerns ways in which to manage the uncertainty. Taken together, these two ideas provide a new perspective on the difference between planning and scheduling.

4. A New Perspective - Observation 1

An interesting exercise is to consider what is known in the earliest stages of a plan/schedule/execute cycle and to contrast this with what is known at the completion of such a cycle. In the early stages, not much is known. In robotic tasks, it may be as little as an abstract goal (the need to repair a broken subsystem on a space station), some idea of the resources which might be called into play (any tools and spares on or available to the station), the objects which may be involved (the tools and all parts in and around the subsystem with the unknown fault), and the manipulatory and perceptual capabilities of the robot(s) in the area as well as its currently assigned tasks. In the final stages, there is a plethora of information. Every important intermediate state can be known to the resolution of the system sensors including the specific objects involved, the values of all of their relevant properties, and the details of all relationships between objects. Every important action can be known within sensor resolution including start times, durations, end times, facilities utilized, even robot spatial and force trajectories. Uncertainty remains, to be sure, but hindsight is greatly superior to foresight in robotic planning, scheduling, and execution.

From this perspective, the important question is what information is required to resolve various uncertainties and when can this information be known with sufficient accuracy that commitments can be made. Planning, scheduling, and execution can be viewed as a continuous series of refinements in which the initial abstract concept is converted into the final concrete realization. An interesting characteristic of robotic tasks is that they are executed in the real world which is the domain of Murphy's laws as well as Newton's laws. Something will probably go wrong, but it cannot be determined in advance what it will be or when it will happen. Furthermore, actions executed in the real world are not easily retracted. In other words, the penalty for commitment made too early will be extra expense, because, once executed, the wrong action will require effort to reverse, if it is reversible at all.

The uncertainty which is encountered during planning, scheduling, and execution has many sources. A major source stems from the fact that the robot (and any supporting computational device) holds an incomplete, inexact digital model of an analog world. The incompleteness is due to the inherent finiteness of the computer relatively to the practically infinite richness of detail in the world. The inexactness is due to the imperfect nature of the input and output channels which the robot (and any supporting computational device) has to its worldly environment. Sensors as input channels and mechanisms as output channels are only accurate within bounds. Any form of intelligent entity dealing with robot tasks must recognize and deal with these facts even though it is helpless to change them. This uncertainty is reflected in the structure of representations of states and actions in systems which must deal with the real world. State descriptions include tolerances on the values associated with object properties. Action descriptions include accuracies on the values achievable by their application. Recently, mechanisms have been reported which can be used to plan about such uncertainty [41].

Since all robotic systems must handle uncertainty at the modeling, sensor, and mechanism level, this type of uncertainty is obviously not of much use for classification (other than realizing that the level of uncertainty in the system must be matched to the level of uncertainty in the task). Another source of uncertainty which occurs at a higher level is useful, however. States and state descriptions involve a degree of uncertainty. In micro-scale, it is not always clear when a particular object which is required will be available, or, if it is available at some point, whether it will still be suitable when needed. In macro-scale, it is certainly possible to imagine that the specifications of initial and goal states can change with time. Similar comments can be made about actions. At the micro-scale, the availability of facilities changes with time, sometime unexpectedly as a higher priority task encounters difficulties or the facility itself requires adjustment or repair. At the macro-scale, it is even possible to imagine that the capability of a facility changes or that facilities are added to or deleted from the pool of resources. These examples each point out that everything in the environment (except the laws of physics) is susceptible to change over time, sometimes gradually, sometimes dramatically. Of fundamental concern is the uncertainty about the availability and suitability of agents, objects, and facilities, and when and to what degree this uncertainty can be resolved.

It is certainly well understood in the robotics community that plans and schedules for robots are fragile in terms of executional integrity [42, 43]. The earliest planning systems included execution monitors [44], and more recent work has been aimed at knowledge-based systems for error recovery [45, 46]. Furthermore, methods have been proposed for incompletely specified or dynamic environments [47, 48, 49]. However, it is not clear that an uncertainty model is a popular one for the classification of planning and scheduling problems. None the less, it is proposed that it is precisely the view of planning, scheduling, and execution as a step by step resolution of uncertainty which begins to formalize the way in which we intuitively view the difference between planning and scheduling problems.

5. A New Perspective - Observation 2

The second observation complements this way of looking at problems by providing a perspective for looking at solution techniques to manage the uncertainties involved in robotic tasks. The key here is the concept of least commitment. In the face of uncertainty, various options can be taken. At one extreme, a guess can be made. If the guess can be made in such a way that it can be easily retracted if it is wrong, then this is a useful strategy [37]. At the other extreme, the decision can be deferred until enough information can be gathered so that the uncertainty can be absolutely resolved. Of course, it is seldom the case in robotic tasks that either of these extremes is obtained. It is not often that so little information is available that a blind guess must be made, or that so much information is available that no doubt remains. It is left to the implementors of planners and schedulers to provide sufficient knowledge to their systems to be able to decide what degree of commitment is warranted by a given level of information. The point is that it is a difficult decision.

Another point is that it is the ordering of actions, the assignment of facilities to carry out actions, and the association of starting and stopping times with actions which are being deferred. These ideas are also not new to the AI or robotics communities, but have been used in different ways than to classify planning and scheduling solution techniques. As has been pointed out, one of the main jobs of a planning system is to order states and actions, so all systems have a basic ordering capability [23]. In addition, the concept of abstraction contains the idea of ordering tasks by importance to provide a focus for the reasoning system [23]. Furthermore, advanced planners are based on the principle of least commitment in the ordering of plan fragments to avoid adverse subgoal interactions [23, 30, 31]. But in keeping with the spirit of this document, it has been suggested only a few times that the output of a planner should at times be limited to a partially ordered set of actions, the final order to be resolved by the execution environment [42, 50]. In the extreme, systems have been implemented that plan only incrementally, either based on the immediate need to do something other than an obvious atomic action [51], or on the opportunity to easily achieve one of a list of goals [52], both driven by the environment.

6. Is There a Difference ?

On one hand, planning and scheduling problems need to be classified by the form and content of the input accepted and the output delivered. Planning and scheduling techniques need to be classified by the representation and reasoning mechanisms which transform the inputs into the outputs. On the other hand, planning and scheduling types of problems can be classified by the time at which sufficient information is available to define the initial and goal states, the facilities catalog, the identity and relationship among agents, the desired actions, the ordering of the actions, the assignment of agents and facilities to the actions, and the association of start and stop times to the actions. Planning and scheduling solution techniques can be classified by the manner in which each of these decisions is delayed until sufficient information is available to justify commitment.

The planner is mainly concerned with what should be done and is invo:.ed with using the initial and goal states along with the capabilities of the facilities in the catalog to select the desired actions. The scheduler is mainly interested in when the desired actions should be accomplished and uses the availabilities of the facilities in the catalog to assign facilities and times to actions. Sometimes the planner must order the actions based on precondition and postcondition arguments, and sometimes the scheduler must order actions based on environmental conditions. Sometimes the environment is very stable and enough information is available that the planner and scheduler can work together to make most decisions prior to the start of execution. Other times the environment is very volatile and so little information is available that the planner and the scheduler must delay almost every decision until an instant before execution.

Confusion between planning and scheduling often occurs because the planner over-commits, taking on the task of ordering actions when there is no justification to prefer one order over another, assigning facilities before the details of availability can be known, and even associating start and stop times before environmental conditions warrant such decisions. Such overcommitment by the planner prematurely removes all decision processes from the scheduler and makes the execution environment unnecessarily susceptible to failures. It is the responsibility of the planner to know when it is absolutely necessary to make these decisions because of unavoidable requirements based on physical laws or facility capabilities, and when it is possible, or even desirable, to leave these decisions for the scheduler.

To underscore these concepts, it should be noted that three different research groups have become interested (apparently independently and simultaneously) in what can be considered a paradigmatic problem in the division of labor between a "planner" and a "scheduler". The domain is robotic assembly, especially in the cases where the assembly can be accomplished by a number of different pathways. A particular problem might occur when a robot is assembling from a complete but random stack of parts in an environment in which all the required facilities are always available. The planner can commit to one particular assembly sequence, usurping the job of the scheduler, but only at the cost of requiring the execution system to always reorder the parts so that they are available in a prescribed sequence. If instead the planner is willing to supply the required set of actions to the scheduler with only the minimum set of ordering constraints, the scheduler can efficiently react to the ordering uncertainty in the availability of parts to guarantee efficient execution.

Notice that the same issues arise if all the parts are available all of the time (for example, bins of each individual part are available in the work cell), but the specialized tools to perform the assembly are only available at random times (for example, because they are used in a number of cells). An even more complex case is encountered if availability of both the parts and the tools is randomized. In all of these circumstances, it is clear that the planner should produce a minimally ordered plan so that the scheduler can capitalize on the uncertainty in the environment rather than being hampered by it.

The groups involved have been interested in the assembly problem based on its own merit and have not been overly concerned with the impact of their work on the categorization of planning and scheduling problem types or solution techniques. They have been more interested in the question of devising the best way of representing scheduling options gained from the planner and reasoning over these options during executions. The representations used have included sets of partial orders [53, 54, 55, 56, 57], AND/OR graphs [58], and Petri nets [59, 60]. It is proposed that the robot assembly problem as formulated by these researchers is an outstanding example of a problem which clearly distinguishes between planning and scheduling, and which clearly indicates the penalties incurred for ignoring the distinctions. It deserves further study in the effort to develop classification schemes for planning and scheduling problems. It is further proposed that the three methods which the researchers have developed deserve comparative study in the effort to develop classification schemes for solution techniques for planning and scheduling problems.

7. Conclusions

It is concluded that there is a distinct difference between planning and scheduling. Until an adequate classification scheme can be developed for planning and scheduling problem types, the case of robotic assembly under various types of uncertainty about availability is offered as a paradigmatic example. It is hoped that this example will stimulate work on the classification of problem types as well as solution techniques.

8. References

1. G. W. Ernst and A. Newell, GPS : A Case Study in Generality and Problem Solving, Academic Press, 1969.

2. R. E. Fikes and N. J. Nilsson, "STRIPS : A New Approach to the Application of Theorum Proving to Problem Solving", Art. Int. 2(3-4), 189-208, 1971.

3. R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and Executing Generalized Robot Plans", Art. Int. 3(4), 251-288, 1972.

4. D. D. Corkill, "Hierarchical Planning in a Distributed Environment", Proc. 6th Inter. Joint Conf. Art. Int. (Tokyo), pp. 168-175, 1979.

5. R. G. Smith, "A Framework for Distributed Problem Solving", Proc. 6th Inter. Joint Conf. Art. Int. (Tokyo), pp. 836-841, 1979.

6. D. E. Appelt, "A Planner for Reasoning about Knowledge and Action", Proc. Nat. Conf. Art. Int. (Stanford), pp. 131-133, 1980.

7. K. Konolige and N. J. Nilsson, "Multiple-Agent Planning Systems", Proc. Nat. Conf. Art. Int. (Stanford), pp. 138-141, 1980.

8. D. McArthur, R. Steeb, and S. Cammarata, "A Framework for Distributed Problem Solving", Proc. Nat. Conf. Art. Int. (Pittsburgh), pp. 181-184, 1982.

9. J. S. Rosenschein, "Synchronization of Multi-Agent Plans", Proc. Nat. Conf. Art. Int. (Pittsburgh), pp. 115-119, 1982.

10. D. D. Corkill and V. R. Lesser, "The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network", Proc. 8th Inter. Joint Conf. Art. Int. (Karlsruhe), pp. 748-756, 1983.

11. S. Cammarata, D, McArthur, and R. Steeb, "Strategies of Cooperation in Distributed Problem Solving", Proc. 8th Inter. Joint Conf. Art. Int. (Karlsruhe), pp. 767-770, 1983.

12. M. Georgeff, "Communication and Interaction in Multiagent Planning", Proc. Nat. Conf. Art. Int. (Washington, D.C.), pp. 125-129, 1983.

13. M. Georgeff, "A Theory of Action for MultiAgent Planning", Proc. Nat. Conf. Art. Int. (Austin), pp. 121-125, 1984.

14. E. H. Durfee, V. R. Lesser, and D. D. Corkill, "Increasing Coherence in a Distributed Problem Solving Network", Proc. 9th Inter. Joint Conf. Art. Int. (Los Angeles), pp. 1025-1030, 1985.

15. C. Stuart, "An Implementation of a Multi-agent Plan Synchronizer", Proc. 9th Inter. Joint Conf. Art. Int. (Los Angeles), pp. 1031-1033, 1985.

16. M. Genesereth, M. Ginsberg, and J. Rosenschein, "Cooperation without Communication", Proc. Nat. Conf. Art. Int. (Philadelphia), pp. 51-57, 1986.

17. M. Georgeff, "The Representation of Events in Multi-agent Domains", Proc. Nat. Conf. Art. Int. (Philadelphia), pp. 70-75, 1986.

18. J. F. Allen and J. A. Koomen, "Planning Using a Temporal World Model", Proc. 8th Inter. Joint Conf. Art. Int. (Karlsruhe), pp. 741-747, 1983.

19. P. Cheeseman, "A Representation of Time for Automatic Planning", Proc. IEEE Inter. Conf. Rob. Auto. (Atlanta), pp. 513-518, 1984.

20. S. Vere, "Planning in Time : Windows and Durations for Activities and Goals", IEEE Trans. PAMI 5(3), pp. 246-267, 1983.

21. S. F. Smith and P. S. Ow, "The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks", Proc. 9th Inter. Joint Conf. Art. Int. (Los Angeles), pp. 1013-1015, 1985.

22. S. Vere, "Temporal Scope of Assertions and Window Cutoff", Proc. 9th Inter. Joint Conf. Art. Int. (Los Angeles), pp. 1055-1059, 1985.

23. E. D. Sacerdoti, "Problem Solving Tactics", Proc. 6th Inter. Joint Conf. Art. Int. (Tokyo), pp. 1077-1085, 1979.

24. L. Siklossy and J. Dreussi, "An Efficient Robot Planner Which Generates its Own Procedures", Proc. 3rd Inter. Joint Conf. Art. Int. (Stanford), pp. 423-430, 1973.

25. E. D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces", Art. Int. 5(2), 115-135, 1974.

26. D. H. D. Warren, "WARPLAN : A System for Generating Plans", Memo #76, Dept. Comp. Logic, Univ. of Edinburgh, 1974.

27. E. D. Sacerdoti, "The Non-Linear Nature of Plans", Proc. 4th Inter. Joint Conf. Art. Int. (Tbiiisi), pp. 206-214, 1975.

28. A. Tate, "Interacting Goals and Their Uses", Proc. 4th Inter. Joint Conf. Art. Int. (Tbilisi), pp. 215-218, 1975.

29. C. Reiger and P. London, "Subgoal Protection and Unravelling During Plan Synthesis", Proc. 5th Inter. Joint Conf. Art. Int. (MIT), pp. 487-493, 1977.

30. E. D. Sacerdoti, A Structure for Plans and Behavior, Elsevier North-Holland, 1977.

31. A. Tate, "Generating Project Networks", Proc. 5th Inter. Joint Conf. Art. Int. (MIT), pp. 888-893, 1977.

32. R. Waldinger, "Achieving Several Goals Simultaneously", Machine Intelligence 8, E. W. Elcock and D. Michie (eds.), Ellis Horwood, pp. 94-136, 1977.

33. M. Stefik, "Planning with Constraints", Art. Int. 16, 111-140, 1980.

34. D. Chapman, "Nonlinear Planning : A Rigorous Reconstruction", Proc. 9th Inter. Joint Conf. Art. Int. (Los Angeles), pp. 1022-1024, 1985.

35. P. R. Davis, "Using and Re-using Partial Plans", Proc. 5th Inter. Joint Conf. Art. Int. (MIT), pp. 494, 1977.

36. P. Friedland and Y. Iwasaki, "The Concept and Implementation of Skeletal Plans", J. Auto. Reasoning 1, 161-208, 1985.

37. M. Stefik, "Planning and Meta-Planning", Art. Int. 16, 141-170, 1980.

38. R. Wilensky, "Meta-planning", Proc. Nat. Conf. Art. Int. (Stanford), pp. 334-336, 1980.

39. G. J. Sussman, A Computer Model of Skill Acquisition, American Elsevier, 1975.

40. L. Siklossy and J. Roach, "Collaborative Problem-Solving between Optimistic and Pessimistic Problem Solvers", Proc. IFIP Congress, pp. 814-817, 1974.

41. R. A. Brooks, "Symbolic Error Analysis and Robot Planning", Inter. J. Robotics Research 1(4), 29-68, 1982.

42. J. H. Munson, "Robot Planning, Execution, and Monitoring in an Uncertain Environment", Proc. 2nd Inter. Joint Conf. Art. Int. (London), pp. 338-349, 1971.

43. P. J. Hayes, "A Representation for Robot Plans", Proc. 4th Inter. Joint Conf. Art. Int. (Tbilisi), pp. 181-188, 1975.

44. R. E. Fikes, "Monitoring Execution of Robot Plans Produced by STRIPS", Proc. IFIP Congress (Ljubljana), pp. 189-194, 1971.

45. M. Gini and G. Gini, "Towards Automatic Error Recovery in Robot Programs", Proc. 8th Inter. Joint Conf. Art. Int. (Karlsruhe), pp. 821-823, 1983.

46. M. H. Lee, D. P. Barnes, and N. W. Hardy, "Knowledge Based Error Recovery in Industrial Robots", Proc. 8th Inter. Joint Conf. Art. Int. (Karlsruhe), pp. 824-826, 1983.

47. R. T. Chien and S. Weissman, "Planning and Execution in Incompletely Specified Environments", Proc. 4th Inter. Joint Conf. Art. Int. (Tbilisi), pp. 169-174, 1975.

48. G. McCalla, L. Ried, and P. F. Schneider, "Plan Creation, Plan Execution, and Knowledge Acquisition in a Dynamic Microworld", Inter. J. Man-Machine Studies 16, 89-112, 1982.

49. B. Ward and G. McCalla, "Error Detection and Recovery in a Dynamic Planning Environment", Proc. Nat. Conf. Art. Int. (Pittsburgh), pp. 172-175, 1982.

50. R. S. Wall and E. L. Rissland, "Scenarios as an Aid to Planning", Proc. Nat. Conf. Art. Int. (Pittsburgh), pp. 176-180, 1982.

51. D. McDermott, "Planning and Acting", Cog. Sci. 2, 71-109, 1978.

52. B. Hayes-Roth, F. Hayes-Roth, S. Rosenschein, and S. Cammarata, "Modeling Planning as an Incremental, Opportunistic Process", Proc. 6th Inter. Joint Conf. Art. Int. (Tokyo), pp. 375-383, 1979.

53. B. R. Fox and K. G. Kempf, "Opportunistic Scheduling for Robotic Assembly," Proc. IEEE Inter. Conf. Rob. Auto. (St. Louis, MO), pp. 880-889, 1985.

54. B. R. Fox and K. G. Kempf, "Complexity, Uncertainty, and Opportunistic Scheduling", Proc. IEEE Conf. AI Appl. (Miami Beach, FL), pp. 487-492, 1985.

55. B. R. Fox and K. G. Kempf, "Planning, Scheduling, and Uncertainty in the Sequence of Future Events", Proc. 2nd Workshop on Uncertainty in AI (Philadelphia), pp. 77-84, 1986.

56. B. R. Fox and K. G. Kempf, "A Representation for Opportunistic Scheduling," Preprints 3rd Inter. Symp. Rob. Res. (Gouvieux, France, 1985), pp. 111-117, 1985.

57. B. R. Fox and K. G. Kempf, "Reasoning about Opportunistic Schedules", Proc. IEEE Inter. Conf. Rob. Auto. (Raleigh, N.C.), to appear, 1987.

58. L. S. Homem de Mello and A. C. Sanderson, "AND/OR Graph Reresentation of Assembly Plans", Proc. Nat. Conf. Art. Int. (Philadelphia), pp. 1113-1121, 1986.

59. C. A. Malcolm, "Petri Nets for Representing Assembly Plans", Working Paper 187, Dept. of A.I., Univ. of Edinburgh, 1986.

60. C. A. Malcolm and A. P. Ambler, "Some Architectural Implications of the Use of Sensors", Research Paper 291, Dept. of A.I., Univ. of Edinburgh, 1986.

S1-61
7P.
189170

The Mechanisms of Temporal Inference*

MP 628000

B.R. Fox
McDonnell Douglas Research Laboratories
St. Louis, MO 63166

S.R. Green
McDonnell Douglas Astronautics Company
Kennedy Space Center, FL 32185

MP 532830

Abstract: The properties of a temporal language are determined by its constituent elements: the temporal objects which it can represent, the attributes of those objects, the relationships between them, the axioms which define the default relationships, and the rules which define the statements that can be formulated. The methods of inference which can be applied to a temporal language are derived in part from a small number of axioms which define the meaning of equality and order and how those relationships can be propagated. More complex inferences involve detailed analysis of the stated relationships. Perhaps the most challenging area of temporal inference is reasoning over disjunctive temporal constraints. Simple forms of disjunction do not sufficiently increase the expressive power of a language while unrestricted use of disjunction makes the analysis NP-hard. In many cases a set of disjunctive constraints can be converted to disjunctive normal form and familiar methods of inference can be applied to the conjunctive sub-expressions. This process itself is NP-hard but it is made more tractable by careful expansion of a tree-structured search space.

1. Introduction

An intelligent autonomous system operating in a remote, unstructured environment must have three capabilities. First, it must be able to create a plan or course of action according to an initial state of the world, a goal state of the world, and some knowledge of its own abilities. Second, it must be able to determine a sequence of actions, according to the constraints on the steps of the plan and the evolving state of the world. Finally, it must be able to produce the desired effect of those actions according to its abilities and the present state of the world.

The performance of the planner, the sequencer, and the executor components of such a system can be very much affected by the language used to represent plans. The concepts of action must be suitable for the planner, which must reason about goals and effects, but at the same time be tractable for the executor, which must produce the desired effects. The concepts of order must be sufficient for the planner, which must control undesired interactions between operations, but at the same time they must not impose unnecessary constraint on the sequencer, which must adapt the sequence of actions to the dynamically changing state of the world. The methods of formulation must enable the planner to produce the most general plans possible, yet at the same time it must be feasible for the sequencer to derive a sequence of actions from those plans. The language must be terse. The size of the plan must be proportional only to its complexity.

*Research conducted under the McDonnell Douglas Independent Research and Development Program.

It has been repeatedly suggested that reasoning over time is an essential element of planning and specific temporal representations have been proposed to facilitate the planning process, including a linear programming model of Malik and Binford[1], the space-time maps of Miller[2], the interval algebra of Allen[3], the point algebra of Vilain and Kautz[4], and the end-point representation of Cheeseman[5]. Although not concerned with planning but instead with the problems of sequencing the activities of robots, Fox and Kempf[6] propose a language of temporal constraints as the target representation for planners.

With this abundance of temporal languages for planning and sequencing, it is important to establish the properties of the proposed languages and to understand the inference methods which can be applied to them. Although a complete survey of temporal reasoning is beyond the scope of this paper, an examination of the basic elements of temporal representation and the methods of temporal inference will establish the primary criteria for comparing these languages.

2. Elements of Temporal Representation

The properties of a language are embodied in its syntactic form and its semantic interpretation. The concern here is with the semantic elements of a language rather than with its syntactic details. Nevertheless, in order to discuss the variety of possible temporal languages, it is necessary to introduce some simple syntactic structures which represent abstract semantic entities.

Temporal languages are concerned primarily with temporal objects: instants and intervals of time. Some authors maintain that instants of time present some semantic difficulties and therefore propose that time intervals should be the primitive element of temporal reasoning[3]. Others maintain that intervals of time can be defined by their endpoints and propose that instants should be treated as the basic element of temporal reasoning. Some sequencing problems involve activities, which in reality occur over some interval of time, but for purposes of analysis can be treated as atomic and indivisible. In the following discussion, instants of time will be treated as primitive objects, denoted by alphanumeric symbols such as **X**, **Y**, and **nine-o'clock**. Likewise, intervals will be denoted by alphanumeric symbols, such as **Z**, **W**, **install-clip**, and **drill-hole**, but when useful or necessary, the initial and final endpoints of an interval will be denoted by a suffix letter **i** or **f** attached to the interval name, such as **Zi** and **Zf**.

Temporal languages are concerned with the attributes of temporal objects. Some languages may allow the specification of the absolute time of some instant or it may be possible to specify the duration of an interval. Specialized systems may associate the properties of physical processes with intervals, such as rates, loads, or volumes. Planning systems may associate propositional variables and their values with temporal objects. Ultimately, each temporal object is associated with some event, activity, or proposition. For instance, it is possible to refer to the instant which begins an occultation, or the interval of time when the action **install-clip** is performed, or the interval of time over which the proposition **channel-is-available** is **true**.

Temporal languages are concerned with the relationships between temporal objects. The most primitive involve the relationships between instants of time. Two instants may be equal, denoted by the operator =, they may be inequal, denoted by the operator <>, or they may be ordered, as denoted by the operator <. In order to avoid any syntactic ambiguity, such relationships are written in fully parenthesized infix notation, as in the expression **(X < Y)**. The relationships between two intervals of time, as defined by Allen, are shown schematically in Figure 1. The relationships between instants and intervals of time can be defined in a similar fashion. All of these relationships can be specified by their respective endpoint relationships as indicated in the right hand column of Figure 1.

In addition to the facilities for explicitly stating the relationships between temporal objects, a temporal language must include some axioms which define the relationships between objects that are not otherwise constrained. Commonly, it is assumed that, in the absence of other explicit constraints, two instants of time, **X** and **Y**, are ordered as either **(X < Y)** or **(Y < X)**, or they are equal, **(X = Y)**. Likewise, unless otherwise constrained, two intervals can be related in any of the 13 possible ways shown in Figure 1. In some applications involving the serial execution of a set of operations, there is no opportunity for any of the operations to be done concurrently. In such cases an axiom which defines the default relationship between intervals states that, unless otherwise constrained, two intervals, **X** and **Y** are disjoint and ordered as either **(X before Y)** or **(Y before X)**. Such axioms play a significant role in the treatment of negation and the processes of inference.

Temporal languages are subject to certain rules of formulation. The simplest rule is to assume that a given set of primitive constraints is to be treated as a conjunction and that they must all be satisfied simultaneously. In contrast, the language defined by Allen allows a restricted form of disjunction. The relationship between a given pair of intervals can be specified as a disjunction of any of the 13 possible primitive relationships. This makes it possible to circumscribe indefinite relationships or to prescribe some relationships which cannot be expressed as one of the 13. For instance, suppose that two intervals, X and Y, must begin at the same time but that there is no constraint on their termination. It would artificially constrain the intervals to state that (X begins Y) because this primitive relationship requires that X terminate before Y. Likewise it would be an artificial constraint to require that (Y begins X). Using this vocabulary of 13 primitive relationships, the relationship between X and Y can only be stated as a disjunction, ((X begins Y) or (Y begins X)). In Allens's language, disjunction is restricted to phrases that define the relationship between a single pair of intervals and cannot be used to pose constraints such as, ((X before Y) or (Z before W)).

The properties of a temporal language are determined by the combination of these elements: the objects, attributes, relationships, default axioms, and rules of formulation. Together these elements determine the set of problems that can be represented. For instance, the language of strict partial orders is composed of symbols which denote instants of time, the primitive ordering relationship <, the default axiom that states for all X and Y either (X < Y) or (Y < X), and a rule of formulation that allows only conjunctions of primitive ordering constraints. A given constraint expression in this language defines a set of admissible total orderings over a set of instants of time. For example, The conjunction ((X < Z) and (Y < Z)) defines 2 admissible orderings of X, Y, and Z: [X,Y,Z] and [Y,X,Z]. The limited rule of formulation in the language of strict partial orders makes it impossible to state the constraints for a problem which admits the 4 linear orderings [X,Y,Z], [Y,X,Z], [Y,Z,X], and [Z,Y,X]. There is no conjunction of primitive ordering constraints which defines exactly this set of linear orderings! The limited forms of disjunction included in Allen's interval algebra or the point algebra defined by Vilain and Kautz encompass some sense of indefiniteness in the relationship between temporal objects but these forms of disjunction are not sufficient to represent the full range of possible ordering problems.

A number of common temporal representations can be quickly distinguished by their constituent elements. For instance, the language of equivalence classes is composed of symbols which denote atomic temporal objects, the equality and inequality relationships, = and <>, an axiom which states that for all X and Y, (X = Y) or (X <> Y), and a rule of formulation which allows only conjunctions of equality constraints. In contrast, the language of graph coloring problems has a similar structure but the rule of formulation allows only conjunctions of inequality constraints. The language of temporal constraints proposed by Fox and Kempf[6] is composed of symbols which denote atomic temporal objects, the ordering relation- ship <, an axiom of serial processes which states that for all X and Y, (X < Y) or (Y < X), and a rule of formulation which allows arbitrary use of conjunction, disjunction, and negation. This axiom limits the scope of this language to problems that involve activities that must be done one at a time, such as a robot performing an assembly task. However, the unrestricted use of disjunction guarantees that this language can represent any problem within that domain. Portrait, a temporal language under development by Fox and Green allows arbitrary use of equality, ordering, conjunction, disjunction, and negation.

3. Methods of Temporal Inference

Temporal reasoning is a process of deriving the properties of temporal objects and the relationships between temporal objects that are implied but may not be explicitly stated in a given set of temporal constraints. The most familiar form of temporal reasoning is constraint propogation. In the language of equivalence classes constraint propogation is based upon two axioms. The first defines the symmetry of equivalence: for all X and Y, (X = Y) implies that (Y = X). The second defines the method for propogating equivalence: for all X, Y, and Z, (X = Y) and (Y = Z) implies that (X = Z). There is no symmetry in the language of strict partial orders, only an axiom which defines the method for propagating order: for all X, Y, and Z, (X < Y) and (Y < Z) implies that (X < Z). The language of partially ordered sets includes an axiom which defines how a disjunction of order and equality can be propogated: for all X, Y, and Z, (X <= Y) and (Y <= Z) implies that (X <= Z). Coupled with this is an axiom which defines how constraints over a given pair of temporal objects can be resolved: for all X and Y, (X <= Y) and (Y <= X) implies that (X = Y). If, after the complete propogation of constraints, only one of the constraints (X <= Y) or (Y <= X) has been imposed then it can be assumed that the two objects are not equal. Vilain and Kautz define a language over instants of time which, for a given pair of instants, allows an arbitrary disjunction of the 3 possible relationships between.that pair. In this context, the propogation of constraints can be best defined by a matrix as shown in Figure 2. Conjunctions of constraints over a single pair of instants can be resolved by a rule of intersection as shown in the matrix of Figure 4. Vilain has demonstrated that constraint propogation within this language is both complete and correct. Allen's interval algebra relies upon similar, tabular rules of inference, but because of the added complexity of this language, constraint propagation is not guaranteed to be complete.

In most circumstances these constraint propagation axioms can be applied in reverse in order to identify the essential constraints in a problem and to eliminate any implied constraints. Given the complete set of implied and essential constraints it is a simple matter to identify the equivalence class of some temporal object along with all of its predecessors, direct predecessors, siblings, successors, and direct successors. For instance, the axiom which defines the predecessors of a temporal object Z states that X is a predecessor of Z if (X < Z). The direct predecessors of Z include all those temporal objects X such that (X < Z) but there does not exist Y such that (X < Y) and (Y < Z). These can be easily identified by scanning the set of essential constraints.

Reasoning about the admissible ordering of temporal objects is directly related to an analysis of precedessors and successors. For instance, in the language of strict partial orders, the controlling axiom specifies that a temporal object X can occur only after all of the predecessors of X. Of course, those objects which have no predecessors can occur at any time. This axiom can be used to incrementally build sequences of activities. At each step of the process simply choose one of those activities which can occur next.

In most sequencing problems the combined ordering constraints limit the admissible sequences of activities but do not remove every sequencing option. In most problems there are many admissible sequences. The number of admissible sequences can serve as a useful indicator of the available sequencing options. In some problems this may provide an estimate of the effort required to find the best sequence of activities. In other problems it may provide an estimate of the inherent flexibility that can be exploited in sequencing those activities. The naive approach to computing this number would be to explicitly enumerate all of the feasible sequences by exhaustive application of the sequencing axiom or other more sophisticated algorithms[7]. Unfortunately, the simplest of problems will prove the most intractable. Consider a serial task of 15 steps with no sequencing constraints. There exists 15! = 1,307,674,368,000 sequences. Even if one sequence could be generated each microsecond it would still requie 15 days to enumerate the entire set. Fortunately, general methods are available which can determine the number of feasible sequences over a strict partial order without explicit enumeration. These methods are first reported in a textbook by Wells[8] but several refinements of these methods were developed at MDRL by the authors. Generally, this computation can be accomplished by recursive application of 3 simple rules:

(1) if a set of activities can be divided into two subsets such that all of the activities in the first set must precede all of the activities in the second set, then the total number of feasible sequences equals the number of feasible sequences for performing the activities in the first set times the number of feasible sequences for performing the activities in the second set.

(2) if a set of activities can be divided into two subsets such that all of the activities in the first set can be performed independently of the acitvities in the second set, then the total number of feasible sequences equals the total number of feasible sequences for performing the activites in the first set times the number of feasible sequences for performing the activities in the second set times the number of ways that one sequence from the first set can be interleaved with one sequence from the second set.

3) if a set of activites cannot be divided into two subsets according to rules (1) or (2) then that set of activites can be partitioned into two strategies for performing those activies which have no feasible sequences in common, and the the total number of feasible sequences will be the number of feasible sequences under the first strategy plus the number of feasible sequences under the second strategy. The partition is generated by identify a pair of unconstrained activities, X and Y. The first strategy is defined by the orginal set of constraints plus the constraint that X must precede Y, (X < Y), and the second strategy adds the constraint that Y must precede X,

(Y < X). (Repeated application of these 3 rules is guaranteed to work regardless of the X and Y chosen when using rule 3, but the number of partitions generated is significantly affected by the choice. By carefully selecting the steps X and Y, it is possible to control the number of partitions ultimately generated.)

By recursive application of these rules it is possible to determine the number of feasible sequences for performing a set of activiites from start to finish, or it can be used to determine the number of ways of completing the task from any given state. In most circumstances the number of feasible sequences corresponds closely to the degree of flexibility inherent in the sequencing of the activites and it can be used as a valuable metric for comparing different plans or strategies. As a side-effect, application of the 3 rules stated above results in the decomposition of a given task into sets of dependent activities, sets of independent activities, and into disjoint sub-strategies. This decomposition can be used by human analysts to better understand the structure of the tasks that they must plan and coordinate.

Unfortunately, inference over a disjunctive language, such as that developed by Fox and Kempf, is much more difficult. One way of resolving the constraints in a disjunctive constraint expression is to convert a given set constraints into disjunctive normal form, i.e. a disjunction of conjunctions of the primitive ordering constraints, keeping only the satisfiable and non-redundant subexpressions. In that form, the methods of inference sketched above can be applied separately to each conjunction of constraints and the results combined under an appropriate interpretation of disjunction. The production of this reduced disjunctive normal form is very difficult, in fact it is NP-hard, but it is an essential part of more general temporal reasoning

For instance, the constraint expression shown in Figure 4 is typical of the constraints imposed on small assembly problems. Production of the disjunction normal form of that constraint expression, using the distributive law of boolean algebra, (X and (Y or Z)) --> ((X and Y) or (X and Z)), results in a set of 1024 conjunctions. In general, the size of the disjunctive normal form grows exponentially with the number of applications of the distributive law. Some of the resulting conjunctions are inconsistent and should never be considered, others are specific cases of more general sub-expressions in the result and can safely be removed. Other simple methods for producing the disjunctive normal form have the same result. However, all of the admissible sequences for performing the task defined by these constraints are embodied in only 22 conjunctions.

An efficient method for deriving that set of 22 conjunctions is closely related to methods for determining the satisfiabliity of boolean expression and is based on the expansion of a tree structured search space. Each node in the search space consists of 2 parts. The first is a partially formed conjunction, and the second is a constraint expression which remains to be satisfied. The root node consists of an empty conjunction coupled with the initial constraint expression. Successor nodes are formed by propagating primitive constraints from the constraint expression into the conjunction being constructed. The target leaf nodes consist of a completed conjunction which satisfies the original constraint expression and an empty set of constraints remaining to be satisfied. Specific heuristics have been developed which make it possible to prune redundant or inconsistent solutions early in the tree expansion. Using these methods the constraint expression shown in Figure 4 produced 28 consistent conjunctions, 6 of which were subsequently identifed as redundant. Subtree expansion was terminated 58 times because inconsistencies were detected and 12 times because redundancies were detected. This is considerably more efficient than producing 1024 conjunctions and then attempting to prune the inconconsistent and redundant sub-expressions.

4. Conclusion

The properties of a temporal language are determined by its constituent elements: the temporal objects which it can represent, the attributes of those objects, the relationships between those objects, the axioms which define the default relationships, and the rules which define the statements that can be formulated. The methods of inference which can be applied to a temporal language are derived in part from a small number of axioms which define the meaning of equality and order and how those relationships can be propagated. More complex inferences involve detailed analysis of the stated relationships. Perhaps the most challenging area of temporal inference is reasoning over disjunctive temporal constraints. Simple forms of disjunction do not sufficiently increase the expressive power of a language while unrestricted use of disjunction makes the analysis NP-hard. In many cases a set of disjunctive constraints can be converted to disjunctive normal form and familiar methods of inference can be applied to the conjunctive sub-expressions. This process itself is NP-hard but it is made more tractable by careful expansion of a tree-structured search space.

References

[1]Malik, J. and Binford, T.O., Reasoning in time and space, Proceedings Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983.
[2]Miller, D., Scheduling heuristics for problem solvers, Research Report 264, Yale University Computer Science Department, New Haven, Conn., 1983.
[3]Allen, J.F., and Koomen, J.A., Planning using a temporal world model, Proceedings Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983.
[4]Vilain, M. and Kautz, H., Constraint propogation algorithms for temporal reasoning, Proceedings Fifth National Conference on Artificial Intelligence, Philadelphia, Penn., 1986.
[5]Cheeseman, P., A representation of time for automatic planning, Proceedings Second IEEE international Conference on Robotics and Automation, Atlanta, Georgia, 1983.
[6]Fox, B.R. and Kempf, K.G., A representation for opportunistic scheduling, Proceedings Third International Symposium on Robotics Research, Paris, France, 1985.
[7]Kalvin, A.D. and Varol, Y.L., On the generation of all topological sortings, Journal of Algorithms, v4, pp. 150-162, 1983.
[8]Wells, M.G., Elements of Combinatorial Computing, Pergamon Press, Elmsford, New York, 1971.

```
X before Y, Y after X            +-----x-----+                  Xi < Xf < Yi < Yf
                                              +-----y-----+

X meets Y, Y met-by X            +-----x-----+                  Xi < Xf = Yi < Yf
                                            +-----y-----+

X overlaps Y, Y overlapped-by X  +-----x-----+                  Xi < Yi < Xf < Yf
                                       +-----y-----+

X starts Y, Y started-by X       +--x--+                        Xi = Yi < Xf < Yf
                                 +-----y-----+

X ends Y, Y ended-by X                 +--x--+                  Yi < Xi < Xf = Yf
                                 +-----y-----+

X contains Y, Y contained-by X   +-----x-----+                  Xi < Yi < Yf < Xf
                                    +--y--+

X equals Y                       +-----x-----+                  Xi = Yi < Xf = Yf
                                 +-----y-----+
```

Figure 1.
Thirteen possible interval relationships.

	yRz	<	=	>	<=	>=	<>	<=>
xRy	xRz							
<		<	<	<=>	<	<=>	<=>	<=>
=		<	=	>	<=	>=	<>	<=>
>		<=>	>	>	<=>	>	<=>	<=>
<=		<	<=	<=>	<=	<=>	<=>	<=>
>=		<=>	>=	>	<=>	>=	<=>	<=>
<>		<=>	<>	<=>	<=>	<=>	<=>	<=>
<=>		<=>	<=>	<=>	<=>	<=>	<=>	<=>

Figure 2.
Matrix of constraint propagation in the point algebra.

	xRy	<	=	>	<=	>=	<>	<=>
xRy	xRy							
<		<	x	x	<	x	<	<
=		x	=	x	=	=	x	=
>		x	x	>	x	>	>	>
<=		<	=	x	<=	=	<	<=
>=		x	=	>	=	>=	>=	>=
<>		<	x	>	<	>	<>	<>
<=>		<	=	>	<=	>=	<>	<=>

Figure 3.
Matrix of constraint resolution in the point algebra.

```
 (co before cl)                    and
 (ba before cl)                    and
((co before st) or (co before dr)) and
((dr before co) or (dr before ba)) and
((ba before dr) or (ba before ca)) and
((ra before co) or (ra before ba)) and
((mi before ra) or (mi before ms)) and
((mi before co) or (mi before ba)) and
((sm before mi) or (sm before ba)) and
((sm before co) or (sm before ba)) and
((ri before co) or (ri before ba))
```

Figure 4.
Typical disjunctive constraints on the steps of an assembly problem.

S2-18
9P
139171

Contingent Plan Structures for Spacecraft

M. Drummond, K. Currie, and A. Tate
University of Edinburgh
Edinburgh EH1 1HN, United Kingdom

EE 872905
ACKN.

1. Abstract.

Most current AI planners build partially ordered plan structures which delay decisions on action ordering. Such structures cannot easily represent contingent actions. ~~This paper presents~~ a representation which can. The representation has some other useful features: it provides a good account of the causal structure of a plan, can be used to describe disjunctive actions, and it offers a planner the opportunity of even *less* commitment than the classical partial order on actions. The use of this representation is demonstrated in an on-board spacecraft activity sequencing problem. Contingent plan execution in a spacecraft context highlights the requirements for a fully disjunctive representation, since communication delays often prohibit extensive ground-based accounting for remotely sensed information and replanning on execution failure.

[illegible]

2. Introduction.

Plan generation isn't problem solving. Planning problems are physical realities which require physical solutions. Planning can only be construed as problem solving when it's part of a larger system which also addresses plan execution; only execution can realize the solution a plan specifies. We use this theme of plan execution to bring together some important issues in AI planning. We consider least commitment plan construction, the representation of teleological information, disjunctive plans, and contingent plan execution in realistically complex domains.

We begin in the next section by briefly discussing the way that most AI planners operate. Commonly used techniques include least commitment action ordering and object selection; we discuss both. Following this, in section 4, we describe an actual planner called O-Plan [1] which uses these techniques to good effect. We cover the essentials of O-Plan's search for an acceptable plan, leaving aside low level details. This discussion is used to show how O-Plan relegates the responsibility for reasoning about disjunctive actions to its search space management component. We argue that what a planner needs is a plan structure which is able to describe the disjunction of action implied by the choices encountered during plan construction. In section 5 we present a solution to the problem. A representation is given which has the properties we seek: it can be used to do least commitment plan construction; it explicitly represents teleological information; *and* it can describe disjunctive actions. Together these abilities allow our plans to be used for plan execution in realistically complex domains. To motivate this, section 6 places the ideas in the context of a spacecraft activity sequencing problem: planetary observation. This example causes us to reflect on the basic principle of least commitment problem solving in general, since it supports a form of least commitment reasoning which commits *even less* than current techniques.

The primary result of this paper is a representation we call *C-Plans*. We claim that the representation is suitable for use in sequencing the activities of automated spacecraft. Further applications-oriented research is required to substantiate this claim.

3. Current AI planners: least commitment plan construction.

An AI planner is given the responsibility of constructing a plan of action. Such a planner is given an initial state description, a set of goals, and a set of action schemas. The schemas are parameterized plans, suitable for solving limited problems. A plan produced by the system is an artifact built from individual operators, appropriately instantiated and ordered. This plan must be sanctioned by the system as a feasible means of achieving the given goals. In this section we examine briefly two of the main operations required to produce this plan: action ordering and variable instantiation.

3.1. Action ordering.

Early planners built totally ordered structures: a plan was a *sequence* of actions. This not only applied to the final plans produced by the system, but also to the partial plans that were built during search. With Sacerdoti's NOAH [3] system this all changed. NOAH built plans as *partial* orders on actions. This meant that it was possible for any two given actions to be unordered with respect to each other. The intuition behind this idea is that a partial order on actions characterizes very many total orders. Today such plans are often called *nonlinear*. It might seem that a system which builds nonlinear plans would be exponentially more efficient than one which builds linear, or totally ordered plans. Unfortunately this has never been proven. Only the intuition exists that nonlinear is better than linear; but this intuition is better than nothing. See Chapman [4] and Drummond [5] for more on this.

3.2. Object selection.

There is another sort of least commitment found in some planners which relates to the way that the objects referred to in plans are selected. Variable instantiation is the process of selecting constants to bind to variables. Each variable can be bound to a specific constant, or unbound, meaning that no constant has yet been selected as appropriate. But it is possible to operate in a more sophisticated way: we can *post constraints* on the permissible constants for any given variable. In this way, we constrain the possible bindings for a variable, rather than select one as correct outright. Information can be gathered during the process of plan construction which leads to the deletion of particular constants from the set of possibilities. The hope is that eventually the set of possibilities will be narrowed to one alternative, or reduced to the empty set, indicating that the constraints posted on the variable are so strict that there is no satisfactory object. This method of associating constants with variables is known as *least commitment object selection*. Its genesis was in Molgen [2].

4. A framework for doing this: O-Plan.

O-Plan is a modern planning system which owes many of its ideas to NonLin [6]. NonLin derives from NOAH, and extends it in many ways. For our purposes the essential contribution of NonLin is its completion of the search space of partial plans: NonLin could find plans that NOAH could not. This is because NonLin had plan modification operations available to it which defined its search space of partial plans so as to include plans that NOAH would never consider. O-Plan inherits its definition of the search space from NonLin. In the next section we consider the mechanisms used by O-Plan to search the space of possibilities. We explain how it keeps track of alternatives, and how it searches through the space of partial plans. We then go on to consider how this mechanism can be extended through a more flexible representation for plans.

4.1. Agenda-based partial plan search.

The planning components in the O-Plan framework employ various techniques to lessen the amount of potential search in any particular application. The techniques include least commitment variable binding, constraint cut-off, temporal coherence and various heuristic functions. O-Plan searches through a space of partial plan states, guided by these techniques, where each partial plan state is derived from the application of a plan modification operator to some current partial plan. This is essentially a search space of plan modifications, or operations whose application results in a new (partial) plan state.

An O-Plan Plan State is a structure of some detail and it includes the partial order of activities that is currently being built (essentially the plan so far), a log of effects and conditions asserted or required in the plan, the teleological information used during plan generation (and available thereafter), variables used during planning and, finally, information on outstanding tasks generated during the planning process. These pending tasks are collected together into agenda lists from which each task can subsequently be scheduled in some opportunistic fashion. This mechanism provides for a dynamic approach similar to that provided by "blackboard" based systems.

In practice there are two main agenda list types, one for task specifications which are fully instantiated, and one for task specifications where certain information has yet to be determined. A third "alternatives" agenda list is currently employed which should eventually disappear, but which has been used in the absence of a complete method for dependency-based plan repair.

Task selection is done under the control of a scheduler, which provides the opportunism for the overall process. This scheduler can be regarded as a "plug-in" module in the O-Plan system and therefore it can reflect various scheduling strategies. The scheduling of a task from the agendas causes a handler (or knowledge source) to process that particular task. The relevant handler is invoked by the type of the task scheduled hence the system is data driven by the tasks themselves. As well as changes to the current Plan State, the processing of a task generally results in the creation of new tasks or the amendment of existing tasks on the agendas.

When choices are made, the task handlers have the option to either post dependency information in the Plan State, or to simply spawn alternative Plan States via the alternatives agenda mentioned earlier. The former method has the advantage that it offers the potential for proper plan repair where only the affected parts of the current (partial) plan are stripped off after a failure, while useful parts are saved and the work done in producing them protected. We are researching how this can be done by using partial plans augmented with teleology information, although there are many outstanding problems.

Processing proceeds in cycles and finishes when all tasks have been processed or when there is reason for a particular task to terminate planning. In theory the handlers are independent of one another but they do have the ability to "poison" the current Plan State if they detect inconsistency or constraint violations. This is the time to backtrack, plan repair or simply give up. Search through the space of partial plans is therefore controlled by the scheduler which chooses the next best thing to do, using information provided by the tasks generated during planning. More detail of the O-Plan control structure can be found in [1].

4.2. The requirement for truly disjunctive structures.

O-Plan searches through a space of partial plans. When there's a choice that cannot be delayed, the current O-Plan task scheduler pursues *one* of the available options by incorporating it into the current Plan State. On failure, O-Plan may reconsider all previous Plan States on the alternatives agenda and pursue a previously ignored plan modification operation. In this way it follows a "one-then-best" search strategy as in NonLin.

An alternative approach is demonstrated by the following scenario. Consider that at some point during its search for an acceptable plan, the system identifies an outstanding goal, *G*. Assume that there are two action schemas which after analysis appear suitable for achieving *G*. The traditional approach says that this choice induces a bifurcation in the search space, each path considering one of the two possible actions. However if our developing plan is able to represent disjunction, such a bifurcation is unnecessary. *Both* possible actions (resulting from instantiating the schemas' variables) can be installed in the plan. The only requirement is that the plan record the fact that these two actions stand in a disjunctive relationship.

By the above discussion we aren't suggesting that a planner consider all possible options at each point in its search; such behavior is doomed to failure, since the number of options open will inevitably be huge. Much of the information r.eeded for later planning also becomes uncertain in a plan with too much disjunction. However if the plan representation is able to describe disjunction, then the system will have the *option* of including action disjunction as appropriate.

Contingent plans are also necessary for doing realistic plan execution monitoring. When a plan is generated, it's unlikely that the generation component can guarantee what the world will be like when plan execution begins. To properly handle this we need disjunctive plans. The planner can produce plans which contain actions to deal with whatever contingencies it deems worth considering. Such a continge"' plan must specify the conditions under which each of the planned actions is appropriate, to allow the execution component to correctly select which action to execute.

So: we would like to formalize a plan structure able to represent disjunction of action. But in doing this there's a trap to avoid. We could easily over-simplify the data structures used by a system such as O-Plan. It would appear possible to formalize a nonlinear plan as a partially ordered set. Mathematically all one requires is a set of actions and an ordering relation over that set. (See [4] for an example.) The ordering relation is required to be irreflexive and transitive, therefore asymmetric. The problem with such a simple formalization is that is fails to capture much of the information that O-Plan exploits during plan generation. In particular, it does not capture the *goal structure* of a plan [7]; that is, the causal structure that exists among the planned actions.

There are other requirements on the formalization that we won't consider in this paper. In particular, we won't address formalizing least commitment object selection. Data structures to support such operations are simple to formalize, but for ease of exposition, we won't do it here. It is straightforward to add this to the formalism we present.

5. Formalizing contingent plans.

We can borrow some notions and notation from Net Theory [8]. Not all the constructions that we need are part of net theory, so we'll have to add a few bits onto the basic framework. We won't motivate our additions; for a brief discussion, see [5], and for more extensive motivation [9]. Essentially, we use Condition/Event systems augmented with event occurrence preference orderings; we also identify the conditions and events of the system with predicates of a simple language. In this section, we'll proceed by informally defining the constructs of our plan language, building up the overall structure we require. The eventual goal is to define *C-plans*, or Contingent Plans, following on the arguments above. It is possible to be quite formal in defining these C-plans, but this paper simply explains and motivates them.

5.1. Basic C-plan structure.

A *proposition* is a functor applied to arguments. A *functor* is written in lower case, followed by its arguments in parentheses. *Arguments* are variables or constants; we allow infinitely many of each. Variables are written in upper case, constants are written in lower case. For example, both *on(a,X)*, and *skew-platform(15,left)* are propositions.

Propositions are identified with what we call *b-elements* and *e-elements*. A b-element is intended to denote a condition in the world, and can be true or false. For instance, the b-element *clear(c)* under a blocks world interpretation is true if and only if the block denoted by *c* has nothing on its upper surface. Propositions are also identified with e-elements. An e-element is intended to denote an action, the occurrence of which changes the holding of certain conditions.[1] For instance the e-element *move(a,b,c)* in a blocks world context might denote the action of moving the block denoted by *a* from the block denoted by *b* to the block denoted by *c*. Certain conditions must hold if this action is to occur; furthermore, when the action does occur, certain conditions in the world will no longer hold, and certain others which did not hold will begin to do so. For example, in the case of the block movement we might expect that *a* can only be moved from *b* to *c* if *a* is initially on *b*. Following the movement, *a* will be on *c*. We need to capture these condition-action relationships in our plan representation.

To do this we introduce the notion of a *flow relation*. A flow relation is a set of ordered pairs, each pair in the set ordering either a b-element and e-element, or e-element and b-element. The ordering of a b-element and e-element is interpreted as an *enable* relation. Thus, the holding of certain conditions is understood to enable the occurrence of certain actions. The ordering of an e-element and b-element is interpreted as a *cause* relation: actions can cause the holding of certain conditions. The flow relation describes the relationship between any given event and that event's enabling conditions and effects. It captures what O-Plan and NonLin call *Goal Structure*; the best dictionary word for this concept is probably *teleology*. We use the word to refer to the *reasons* for some event or condition being included in a plan. The flow relation of a net allows a formal analysis of which actions can be used to enable which other actions; this is essentially the reasoning that O-Plan performs to generate a plan. Other modern planners, such as SIPE [10] also include such information in their plan data structures.

We will refer to the b-elements which are ordered immediately before an e-element as that e-element's *preconditions*; similarly, we will refer to the b-elements ordered immediately after it as its *postconditions*.

Graphically we present b-elements as circles and e-elements as squares. Each circle is labeled with the proposition which is the b-element, and each square is labeled with the proposition which is the e-element. The flow relation is drawn as arcs from circles to squares and from squares to circles. If an arrow is to go from a circle to a square, and another from the same square to the same circle, we draw only one line, and use an arrow-head on each end of the line to indicate the two arcs.

One other ordering relation is needed to complete the basic C-plan structure. This is the *before* relation, used to constrain the way that a net can execute. Intuitively, the before order is a specification of which events must occur before which other events if a plan is to run to its intended completion. We often refer to the *before* relation as *execution advice*. Consider: the cause and enable orderings in a C-plan's flow relation describe what is causally possible. But in planning we are often interested in only one of generally many causally permitted execution sequences. Causal orderings will not always uniquely constrain a set of actions to describe just those behaviors

[1] We use the terms "action" and "event" interchangeably, as convenient.

which achieve a planner's overall goals.

A classic example of this occurs in blocks-world tower construction problems. For example: given the problem of creating a tower with block C on the bottom, block B in the middle, and block A on top, the plan construction reasoning must order the two required stack actions to reflect its overall goals. To see this, assume that all blocks are initially clear and on the table. If a plan calls for stacking A on B, and B on C, then *both* stack actions are enabled in the initial state. It is not an ordering enforced by *causation* that requires the stacking of B on C before A on B. Rather, it is the agent's intention regarding overall plan execution outcome that directs the sequencing of the two actions.

So a C-plan is defined by specifying a set of b-elements (which denote the conditions of interest in the domain being modelled), a set of e-elements (which denote the relevant actions), and an ordering relation on the members of these two sets (technically, the relation is bipartite, since it orders members of two different sets). The C-plan is augmented by giving some execution advice for causally underconstrained actions. This advice takes the form of an ordering relation on C-plan e-elements. To keep the graphical presentation of C-plans simple, we do not draw arcs between e-elements ordered in the execution advice. Instead, the ordered pairs are simply listed beside the net.

A simple blocks world plan basically compatible with what we have defined here can be found in [11].

5.2. C-plan projection.

We now have to say something about the *projection* of a C-plan. A projection is a structure which supports reasoning about the behaviors that a C-plan describes. First we must say something about the conditions under which events can occur and what changes they realize by occurring. Second we must build up the projection structure which describes the overall behavior of a C-plan, using the definition of individual event occurrence as a building block. E-element occurrence can be used as a "state generatoı" to create a state-space account of the behaviors permitted by a plan.

We will call an arbitrary set of b-element propositions a *case*. We interpret such a set of propositions as a partial description of a state of the world. If a proposition is in a case, then it is true; if it is not in the case, then it is false. Graphically, we present cases only in terms of C-plans -- when doing so, we place a dot (a *token*) inside each and only those circles labelled with propositions in the plan which are also in the given case.

We can use this idea of a case as a partial world state description to define when an individual e-element is enabled; that is, when the action it denotes is allowed to occur. To model this, we can say that an e-element is *enabled* in a case if and only if its preconditions are a subset of the case, i.e., if the enabling conditions of the event are true. We also require that none of the e-element's postconditions are already in the case, unless they are also preconditions. Further, we can specify how the world is changed under the occurrence of the action, by defining how an e-element's enabling case is modified to gain a successor. We can generate a new case through the *occurrence* of an e-element: the new case is defined to be the old one, minus all the e-element's preconditions, plus all the e-element's postconditions. The effects of an event are made true in the successor case, and the enabling conditions are made false. If a precondition is not made false by the occurrence of an action, one need only make the relevant b-element a postcondition of the e-element as well.

This definition of e-element occurrence can be used to build up a state-space graph structure which tells a story about the possible behaviors of a C-plan. Given an initial case and a C-plan, we can build up a *projection graph* as follows. The initial case is used as the starting node of the projection graph. E-elements of the given C-plan are repeatedly applied in non-terminal projection graph cases until there are no more cases in which any of the C-plan's e-elements have concession. Arcs leading from node to node in this graph are labelled with e-elements. An arc directed from one node α to another node β indicates that the e-element labelling the arc has concession in the case contained in α and under occurrence, produces the case contained in β.

The idea is that the graph structure defined in this way contains a given initial case as its starting node, and that each node in the graph contains a case reachable under e-element occurrence. With the interpretation of a case as a partial description of the world, the projection gives us a prediction of what a C-plan can do in terms of the possible world states it might give rise to. The initial case describes the "current" state of the world, and cases in the graph reachable from the initial case describe future possible world states. The arcs in the graph denote transitions from one world state to another, and these transitions can be realized through the actual execution of the actions that correspond to the e-elements labelling the arcs.

So the nodes of our projection graph contain cases, and the arcs are labelled with steps. We can map this structure onto the classical AI picture of planning as follows. The first node of our projection is the initial state given in the problem specification. In order to represent a solution to the problem, the plan's projection must give rise to a node which contains the required goals. We can say that a C-plan is a *potential solution to a planning problem* if it is applicable in the initial case of the problem, and under projection gives rise to a case which contains the given goals. Also, a particular case reachable under e-element occurrence in a partially developed C-Plan can be used for "Question Answering" operations in the planner during plan generation.

Using the idea of projection we can now say something more precise about a C-plan's execution advice. Recall the basic idea. Execution advice must contain the information required to remove harmful residual non-determinism. The advice should not restrict legitimately causally independent actions from occurring concurrently, but it should prevent planned actions from occurring in an order permitted by the causal structure of the plan but unintended by the planner. We can explain the meaning of a plan's execution advice by interpreting it as a guide to navigation through the projection structure. Basically, we say that a C-plan's execution advice is *sound* (with respect to a given problem specification) if and only if for all choice points in the projection, if there is *any* hope for success at the choice point, then either all choices lead to success, or for each choice point that could lead to failure, there is advice about another possible alternative, such that the suggested alternative *can* lead to success. In essence, when there is still hope for success the advice prevents the wrong sequencing choice from being made. To achieve this the advice must prescribe an order on e-elements which prevents certain paths through the projection from being considered at execution time.

It is possible to generalize the projection we have defined to deal with least commitment reasoning about action ordering. To do this, one need only say when a set of e-elements are causally independent, and use this definition to specify when sets of e-elements can be applied to a case, in bulk, to derive a successor. If this is done the arcs of the projection graph are labelled with *sets* of e-elements which describe the parallel occurrence of the denoted actions. This means that if some events are causally independent the e-elements which describe them can be applied as a set, and reasoning can continue from the resulting case.

6. A spacecraft activity sequencing example.

This section presents an example problem and its representation using C-plans. This problem would be difficult if not impossible to represent using the classic partially ordered structures found in systems like NOAH and NonLin.

The basic scenario for the example is as follows. While on a deep space mission, a spacecraft is to pass very close to the planet *Jinx*. Earth-based observation has determined that two weather systems obtain on Jinx: crystal clear skies and turbulent sand storms. While it isn't known exactly what conditions will hold when the spacecraft arrives, it is certain to be one of these two. So useful observations can be made regardless of the atmospheric conditions. If the atmosphere is unclouded, then visible light pictures should be taken. If a sand storm is in progress, then infrared pictures will be most effective.

The camera used for visible light and infrared pictures is the same, so it is impossible to take a visible light and infrared picture in parallel. An initialization step is required in order to prepare the camera for visible light or infrared work. Regardless of the sort of picture taken, a digital image is stored in a frame buffer on board. The frame buffer is only large enough to store one picture. Each time a picture is written to the frame buffer by the camera, a transfer operation must free the buffer by copying the information to an on-board tape storage medium. For this simple example, we do not address the problem of transferring the stored images back to Earth.

It would be nice to avoid specifying an observation programme rigidly in advance. Since Jinx is too far from Earth to permit the up-loading of an appropriate command sequence (using information gathered closer to the encounter) it is preferable to be opportunistic, and exploit the atmospheric conditions which obtain when the spacecraft arrives. During the period of contact, conditions may change, and the pictures being taken should reflect current opportunity.

From an AI planning perspective, the problem is to have a plan which represents the disjunctive observational requirement simply and economically. Notice that it is not a problem to have an on-board computer which runs a contingent program during the Jinx encounter phase. In principle, the program could be written in any language whatever, compiled, and up-loaded to the spacecraft well in advance. But for an AI planner the problem is one of representing the disjunction in a way that permits reasoning about a plan, since the plan will form part of a lager scenario with unexpected events and changing requirements. We give a C-plan which does this. It specifies what each of the individual observation operations are and the conditions under which they are to be carried out.

The plan of figure 1 is projected in figure 2. The projection describes the behaviors that are possible for the plan. Each arc in the projection is labelled with an integer as used for each event in figure 1. Notice that for this example no execution advice is required. See [11] for an example of how this ordering relation is used.

The plan describes the following behaviors. While it doesn't matter what conditions obtain when the spacecraft arrives at Jinx, assume for the sake of argument that: the spacecraft camera is initialized for infrared work; that the weather on Jinx is clear; and that the frame buffer is empty. A case which describes these conditions is contained in the projection node *S*1. Two events are possible, as described by the C-plan's e-elements *clouding* (2) and *setup(vis)* (4). *Clouding* (2) denotes the event of the atmosphcre becoming clouded by a storm. The *setup(vis)* (4) e-element denotes the action of configuring the camera to take visible light pictures. Similarly, the e-element *setup(ir)* (3) denotes the action of configuring the camera to take infrared pictures.

There are two tight cycles in the projection, one between *S*2 and *S*3, and one between *S*4 and *S*5. These cycles model the normal behavior of the plan during a period when the atmosphere is in a stable state. A transition from *S*2 to *S*3 models the action of taking a visible light picture, and a transition from *S*3 to *S*2 models the action of transferring the picture information from the frame buffer to tape. Likewise, a transition from *S*5 to *S*4 models the action of taking an infrared picture, and a transition from *S*4 to *S*5 models clearing the frame buffer to tape. All other transitions in the projection can be easily read as setup actions in response to changes in the planet's atmosphere.

7. Conclusions.

There is a relationship between the choices of action schema to achieve goals at plan generation time, and contingent plans which support flexible plan execution. Plan generation is reasoning about goals and the means to achieve them. Plan execution is about actually realizing these promised goals. If fast, efficient, and flexible AI planning systems are to ever exist, they must strike a balance between reasoning about disjunction in advance, and

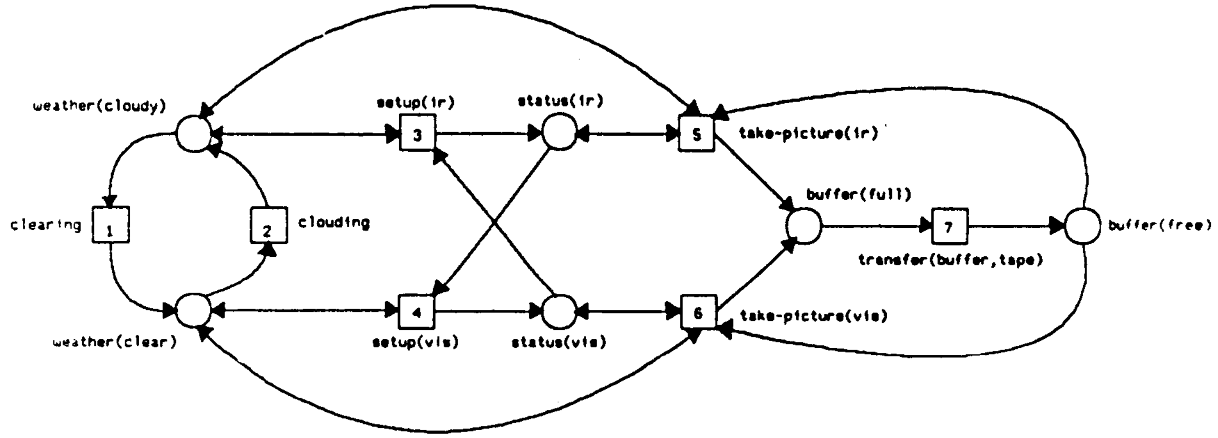


Figure 1: A contingent plan for taking visible light or infrared pictures.

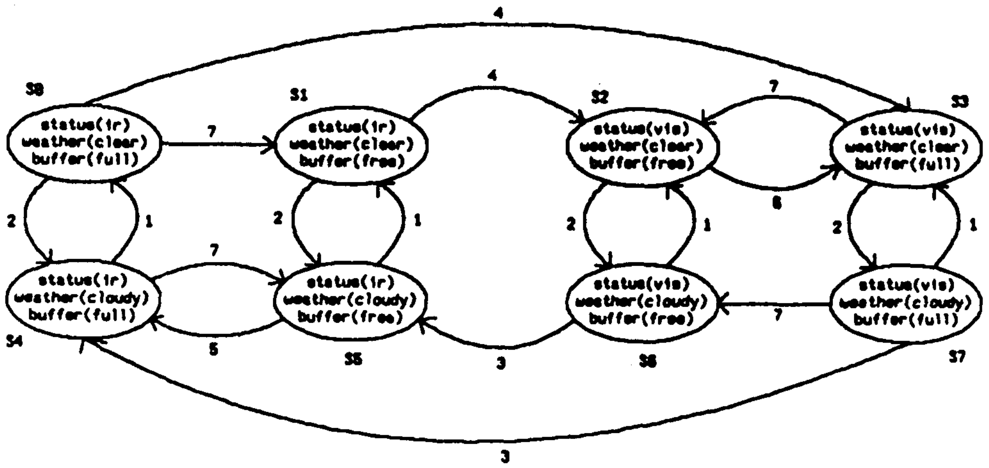


Figure 2: The projection of the picture-taki.g plan.

reasoning about it only when necessitated by plan execution failures. This paper goes some way towards the construction of such a planner by defining a flexible and expressive plan representation which has the ability to represent disjunctive plans. It does this without losing information such as Goal Structure, used by systems like NonLin [6], O-Plan [1] and SIPE [10].

It is important to realize that what we have defined is a representation able to describe contingent actions which is useful from an AI perspective. It is not hard to write contingent computer programs. But the eventual goal is to automate spacecraft command generation. It is likely that AI techniques will be used to perform this task. A start at this has been made with the Deviser planner for the Voyager spacecraft [12]. What this means is that AI representations must be used, and where inadequate, must be improved. Since disjunctive situations will often arise, any planner automatically generating spacecraft commands must be able to reason about disjunction.

We are now working on adapting O-Plan to generate C-plans. Simple disjunctive plans can already be generated; more interesting examples will require more complex generation algorithms. We are currently working on an algorithm to achieve a specified marking in a Petri Net to help produce a robust and efficient C-plan generation algorithm.

8. Acknowledgements.

This work was supported by the U.K. Alvey programme and the Science and Engineering Research Council on Grants GR/D/58987 (An Architecture for Knowledge Based Planning and Control) and GR/E/05421 (T-SAT: AI Applied to a Spacecraft). The support of System Designers plc for the work of the AIAI Knowledge-Based Planning Group is gratefully acknowledged.

9. References.

[1] Currie, K., & Tate, A. 1985. O-Plan: Control in the open planning architecture. In *The Proceedings of the BCS Expert Systems '85 Conference*. Warwick (December), Cambridge University Press.

[2] Stefik, M. 1981. Planning with constraints (Molgen: Part I), *Artificial Intelligence*. Vol. 16, pp. 111-140.

[3] Sacerdoti, E.D. 1975. The non-linear nature of plans. In *The Proceedings of IJCAI-75*. Tbilisi U.S.S.R.

[4] Chapman, D. 1985. Nonlinear planning: a rigorous reconstruction. In *The Proceedings of IJCAI-85*. pp. 1022-1024.

[5] Drummond, M.E. 1986. A representation of action and belief for automatic planning systems. In the proceedings of the *CSLI/AAAI workshop on Planning & Action*. Oregon, U.S.A. Morgan Kauffman. (Also Artificial Intelligence Applications Institute technical report AIAI-TR-16.)

[6] Tate, A. 1977. Generating Project Networks. In *The Proceedings of IJCAI-5*. Cambridge, Mass., U.S.A. pp. 888-893.

[7] Tate, A. 1984. Goal structure -- capturing the intent of plans. In *The Proceedings of ECAI-84*. Pisa, Italy. (September).

[8] Reisig, W. 1985. *Petri nets: an introduction*. Springer-Verlag, EATCS Monographs on theoretical computer science, vol. 4.

[9] Drummond, M.E. 1986. Plan Nets: a formal representation of action and belief for automatic planning systems. Ph.D. Dissertation, Department of Artificial Intelligence, University of Edinburgh.

[10] Wilkins, D.E. 1984. Domain independent planning: representation and plan generation. *Artificial Intelligence*, No. 22.

[11] M.E. Drummond. (August) 1985. Refining and extending the procedural net. In *The Proceedings of IJCAI-85*. Los Angeles, Calif. pp. 1010-1012.

[12] S.A. Vere. (May) 1983. Planning in Time: Windows and Durations for Activities and Goals. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. PAMI-5, No. 3, pp. 246-267.

S3-63
138
188172

Reasoning and Planning in Dynamic Domains: An Experiment With a Mobile Robot

M.P. Georgeff,* A.L. Lansky,* and M.J. Schoppers
Stanford Research Institute International
Menlo Park, CA 94025

36 | 3[illegible]

Abstract

~~We describe~~ progress made toward having an [illegible] autonomous mobile robot reason and plan complex tasks in real-world environments is described. To cope with the dynamic and uncertain nature of the world ~~we~~ use a highly reactive system to which is attributed attitudes of *belief, desire,* and *intention*. Because these attitudes are explicitly represented, they can be manipulated and reasoned about, resulting in complex goal-directed and reflective behaviors. Unlike most planning systems, the plans or intentions formed by the system need only be partly elaborated before it decides to act. This allows the system to avoid overly strong expectations about the environment, overly constrained plans of action, and other forms of over-commitment common to previous planners. In addition, the system is continuously *reactive* and has the ability to change its goals and intentions as situations warrant. Thus, while the system architecture allows for reasoning about means and ends in much the same way as traditional planners, it also possesses the reactivity required for survival in complex *real-world domains*.

~~We have tested~~ the system was tested using SRI's autonomous robot (Flakey) in a scenario involving navigation and the performance of an emergency task in a space station scenario.

1 Introduction

One feature that is critical to the survival of all living creatures is their ability to act appropriately in dynamic environments. For lower life forms, it seems that sufficient capability is provided by stimulus-response and feedback mechanisms. Higher life forms, however, require more complex abilities, such as reasoning about the future and forming plans of action to achieve their goals. The design of reasoning and planning systems that are *situated* in environments with real time constraints can thus be seen as fundamental to the development of intelligent autonomous machines.

In this paper, we describe a system for reasoning about and performing complex tasks in dynamic environments and demonstrate how the system can be applied to the control of an autonomous mobile robot. The system, called a procedural reasoning system (PRS), is endowed with the psychological attitudes of belief, desire, and intention. At any instant, the actions that the system considers performing depend not only on the current desires or goals of the system, but also on its beliefs and previously formed intentions. The system also has the ability to reason about its own internal state – that is, to reflect on its own beliefs, desires, and intentions and to modify these as it chooses. This architecture allows the system to reason about means and ends in much the same way as traditional planners, but provides the reactivity required for survival in complex real-world domains.

As the task domain, we envisage a robot in a space station acting as an astronaut's assistant. When asked to get a wrench, for example, the robot works out where the wrench is kept, plans a route to get it, and goes there. If the wrench is not where expected, the robot may reason further about how to obtain information on its whereabouts. It then finally returns to the astronaut with the wrench or explains why it could not be retrieved.

In another scenario, the robot may be midway through the task of retrieving the wrench when it notices a malfunction light for one of the jets in the reactant control system of the space station. It reasons that handling this malfunction is of higher priority than retrieving the wrench and sets about diagnosing the fault and correcting it. Having done this, it continues with its original task, finally telling the astronaut what has happened.

To accomplish these tasks, the robot must not only be able to create and execute plans, but must be willing to interrupt or abandon a plan when circumstances demand it. Moreover, because the world in which the robot is situated is continuously changing and because other agents and processes can issue demands at arbitrary times, performance of these tasks requires an architecture that is highly reactive as well as goal-directed.

We have used PRS with the new SRI robot, Flakey, to accomplish much of the two scenarios described above, including both the navigation and malfunction-handling tasks. In the next section, we discuss some of the problems with traditional planning systems. The architecture and operation of PRS is then described, and Flakey's primitive capabilities are delineated. We then give a more detailed analysis of the problems posed by this application and our progress to date. We concentrate on the navigation task; the knowledge base used for the jet malfunction handling is described elsewhere [15,17].

2 Previous approaches

Most architectures for intelligent autonomous systems consist of a plan constructor and a plan executor. Typically, the plan constructor plans an entire course of action before commencing execution of the plan [1,11,25,28,30,32,33,34]. The plan itself is usually composed of primitive actions – that is, actions

[*] Also affiliated with the Center for the Study of Language and Information, Stanford University, Stanford, California.

This research has been made possible in part by a gift from the System Development Foundation, the Office of Naval Research under contract no. N00014-85-C-0251 and FMC under contract no. FMC-147466.

that are directly performable by the system. The motivation for this approach, of course, is to ensure that the planned sequence of actions actually achieves the prescribed goal. As the plan is executed, the system performs the primitive actions in the plan by calling various low-level routines. Usually, execution is monitored to ensure that these routines achieve the desired effects; if they do not, the system may return control to the plan constructor to modify the existing plan appropriately. There are, however, a number of serious drawbacks with this architecture as the basis for the design of autonomous agents.

First, this kind of planning is very time consuming, requiring exponential search through potentially enormous problem spaces. It is thus usual for classical AI planners to spend considerable time thinking before performing any effector actions. While this may be acceptable in some situations, it is not suited to domains where replanning is frequently necessary and where system viability depends on readiness to act. In real-world domains, unanticipated events are the norm rather than the exception, necessitating frequent replanning. Furthermore, the real-time constraints of the domain often require almost immediate reaction to changed circumstances, allowing insufficient time for this kind of planning.

Second, in real-world domains, much of the information about how best to achieve a given goal is acquired during plan execution. For example, in planning to get from home to the airport, the particular sequence of actions performed depends on information acquired on the way – such as which turnoff to take, which lane to get into, when to slow down and speed up, and so on. Traditional planners can only cope with this uncertainty in two ways: (1) by building highly conditional plans, most of whose branches will never be used, or (2) by leaving low-level tasks to be accomplished by fixed primitive operators that are themselves highly conditional (e.g., the intermediate level actions (ILAs) used by SHAKEY [23]). The former case is combinatorially explosive or simply cannot be done – the world around us is simply too dynamic to anticipate all circumstances. The latter, as usually implemented, seriously restricts flexibility and reasoning capabilities. Of course, in situations where we can paint ourselves into a corner, some preplanning is necessary. But even this need not involve expanding plans down to the level of primitive operators; indeed, we may do the planning in quite a different abstraction space than that used to guide our actions in the real world (see, for example, the representations in the missionaries and cannibals problem discussed by Amarel [3]).

A third drawback of traditional planning systems is that they usually provide no mechanisms for reacting to new situations or goals during plan execution, let alone during plan formation. For example, many robots (e.g., SHAKEY [23]) effectively shut off their abilities to react to new situations and goals while moving from one location to another. Only low-level feedback mechanisms and emergency sensors such as collision detectors remain enabled. Such disregard for sensory input is particularly undesirable in realistic environments in which unpredictable events may occur or other agents may be active – because of innaccurate information about the actual state of the world, actions may be chosen that are inappropriate to achieving the goals of the system. By remaining continuously aware of the environment, an agent can modify its actions and goals as the situation warrants.

Indeed, the very survival of an autonomous system may depend on its ability to react quickly to new situations and to modify its goals and intentions accordingly. For example, in the scenario described above, the robot must be capable of deferring the task of fetching a wrench when it notices something more critical that needs attention (such as a jet failure). The robot thus needs to be able to reason about its current intentions, changing and modifying these in the light of its possibly changing beliefs and goals. While many existing planners have replanning capabilities, none have accomodated modifications to the system's underlying set of goal priorities.

Finally, current planners are overcommitted to the planning strategy itself – no matter what the situation, or how urgent the need for action, these systems *always* spend as much time as necessary to plan and reason about achieving a given goal before performing any external actions whatsoever. They do not have the ability to decide when to stop planning, nor to reason about the trade-offs between further planning and longer available execution time. Furthermore, they are committed to one particular planning strategy, and cannot opt for different methods in different situations. This clearly mitigates against survival in the real world.

In sum, the central problem with traditional planning systems may be viewed as one of overcommitment. These systems have strong expectations about the behavior of the environment and make strong assumptions about the future success of their own actions. They are strongly committed to their goals and intentions and, except in certain simple ways, cannot modify them as circumstances demand. This would be fine if it were possible to build plans that accommodate all the complexities to which an agent must be responsive; unfortunately, in most real-world domains, the construction of such plans is infeasible.

Of course, we are not suggesting that preplanning, followed by later replanning, can be completely avoided: because of unanticipated changes in the environment, an agent will often have to reconsider its goals or its intended means of achieving these. This is a property of the environment that an agent can do little about. If the agent did not make *some* assumptions about the behavior of the environment, there is little chance it would ever be able to act. On the other hand an agent should not make too many assumptions about the environment – to the extent possible, decisions should be deferred until they *have* to be made. The reason for deferring decisions is that an agent can only acquire *more* information as time passes; thus, the quality of its decisions can only be expected to improve. Of course, there are limitations resulting from the need to coordinate activities in advance and the difficulty of manipulating excessive amounts of information, but some degree of deferred decision-making is clearly desirable.

There has been some work on developing planning systems that interleave plan formation and execution [10,21,29]. While these systems can cope far better with uncertain worlds than traditional planners, they are still strongly committed to achieving the goals that were initially set them. They have no mechanisms for changing focus, adopting different goals, or reacting to sudden and unexpected changes in their environment. The reactive systems used in robotics also handle changes in situation better than traditional planning systems [2,7,18]. Even SHAKEY [23] utilized reactive procedures (ILAs) to realize the primitive actions of the high-level planner (STRIPS), and this idea is pursued further in some recent work by Nilsson [24]. However, there is no indication of how these systems could reason rationally about their future behaviors, such as to weigh the pros and cons of taking one course of action over another.

3 Knowledge Representation

The system we used for controlling and carrying out the high-level reasoning of the robot is called a *Procedural Reasoning System* (PRS) [15].[1] The system consists of a *data base* containing current *beliefs* or facts about the world, a set of current *goals* or *desires* to be realized, a set of *procedures* (which, for historical reasons, are called *knowledge areas* or KAs) describing how certain sequences of actions and tests may be performed to achieve given goals or to react to particular situations, and an *interpreter* (or *inference mechanism*) for manipulating these components. At any moment, the system will also have a *process stack* (containing all currently active KAs) which can be viewed as the system's current *intentions* for achieving its goals or reacting to some observed situation.

The basic structure of PRS is shown in Figure 1. A brief description of each component and its usage is given below.[2] Later sections will give examples of PRS use in the the robot scenario.

[1] Flakey is being used in a variety of experiments at SRI, and PRS is just one of various systems being employed for controlling Flakey.

[2] A more formal description of PRS may be found in [17].

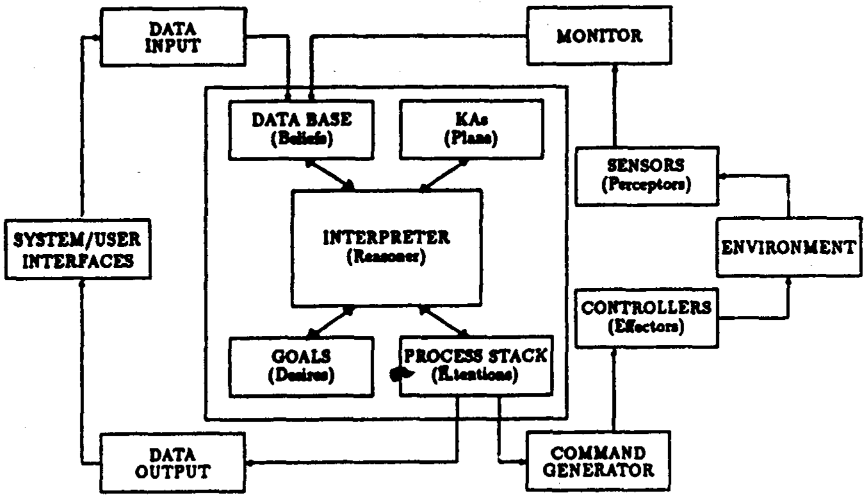


Figure 1: System Structure

3.1 The System Data Base

The contents of the PRS data base may be viewed as representing the current beliefs of the system. Some of these beliefs may be provided initially by the system user. Typically, these will include facts about static properties of the application domain — for example, the structure of some subsystem, or the physical laws that some mechanical components must obey. Other beliefs are derived by PRS itself as it executes its KAs. These will typically be current observations about the world or conclusions derived by the system from these observations. Consequently, at some times the system may believe that it is in a particular hallway, and at other times, in another. Updates to the data base therefore necessitate the use of consistency maintenance techniques.

The data base itself consists of a set of *state descriptions* describing what is [believed to be] true at the current instant of time. We use first-order predicate calculus for the state description language. Free variables, represented by symbols prefixed with $, are assumed to be universally quantified. The statement

`(∨ (¬ (on $x table)) (red (color $x)))`

for example, represents states of the world in which every object on the table is red. Data base queries are done using unification over the set of data base predicates.

State descriptions that describe *internal* system states are called *metalevel* expressions. The basic metalevel predicates and functions are predefined by the system. For example, the metalevel expression `(goal g)` is true if g is a current goal of the system.

3.2 Goals

Goals appear both on the system goal stack and in the representation of KAs. Unlike most AI planning systems, PRS goals represent desired *behaviors* of the system, rather than static world states that are to be (eventually) achieved. Hence goals are expressed as conditions on some interval of time (i.e., some sequence of world states).

Goal behaviors may be described in two ways. One is to apply a *temporal predicate* to an n-tuple of terms. Each temporal predicate denotes an *action type* or a *set* of state sequences. That is, an expression like "`(walk a b)`" can be considered to denote the set of state sequences which embody walking actions from point a to b.

A behavior description can also be formed by applying a temporal operator to a state description. Three temporal operators are currently used. The *expression* (!*p*), *where p is some state description* (possibly involving logical connectives), is true of a sequence of states if *p* is true of the last state in the sequence; that is, it denotes those behaviors that *achieve p*. Thus we might use the behavior description `(!(walked a b))` rather than `(walk a b)`. Similarly, (?*p*) is true if *p* is true of the first state in the sequence – that is, it can be considered to denote those behaviors that result from a successful *test for p*. Finally, (#*p*) *is true if p is preserved* (maintained invariant) throughout the sequence.

Behavior descriptions can be combined using the logical operators ∧ and ∨. These denote, respectively, the intersection and union of the composite behaviors.

As with state descriptions, behavior descriptions are not restricted to describing the external environment, but can also be used to describe the internal behavior of the system. Such behavior specifications are called metalevel behavior specifications. One important metalevel behavior is described by an expression of the form (=> *p*). This specifies a behavior that places the state description *p* in the system data base. Another way of describing this behavior might be (!(*belief p*)).

3.3 Knowledge Areas

Knowledge about how to accomplish given goals or react to certain situations is represented in PRS by declarative procedure specifications called *knowledge areas* (KAs). Each KA consists of a *body*, describing the steps of the procedure, and an *invocation condition* that specifies under what *situations* the KA is useful.

The body of a KA is represented as a graphical network and can be viewed as a plan or plan schema. However, it differs in a very important way from the plans produced by most AI planners: it does not consist of possible sequences of primitive actions, but, rather, of possible sequences of *subgoals* to be achieved. Thus, the bodies of KAs are much more like the high-level "operators" used in planning systems such as NOAH [28] and SIPE [34]. They differ in that (1) the subgoals appearing in the body can be described by complex temporal expressions (i.e., the goal expressions described in the preceding section), and (2) the allowed control constructs are much richer, and include conditionals, loops, and recursion. One important advantage of using abstract subgoals rather than fixed calls to actions is that the knowledge expressed in any given KA is largely independent of other KAs, thereby providing a very high degree of modularity. It is thus possible to build domain knowledge incrementally, with each component KA having a well-defined and easily understood semantics.

The invocation part of a KA contains an arbitrarily complex logical expression describing under what conditions the KA is useful. Usually, this consists of some conditions on current system goals (in which case, the KA is invoked in a goal-directed fashion) or current system beliefs (resulting in data-directed or *reactive* invocation), and may involve both. Together, the invocation condition and body of a KA express a declarative fact about the effects of performing certain sequences of actions under certain conditions.

The set of KAs in a PRS application system consists not only of procedural knowledge about a specific domain, but also includes *metalevel* KAs — that is, information about the manipulation of the beliefs, desires, and intentions of PRS itself. For example, a typical metalevel KA would supply a method for choosing between multiple relevant KAs, or how to achieve a conjunction of goals, or how much further planning or reasoning can be undertaken given the real-time constraints of the problem domain. These metalevel KAs may, of course, utilize domain-specific knowledge as well. In addition to user-supplied KAs, each PRS application contains a set of system-defined default KAs. These are typically domain-independent metalevel KAs.

3.4 The System Interpreter

The system interpreter runs the entire system. From a conceptual viewpoint, it operates in a relatively simple way. At any particular time, certain goals are active in the system, and certain beliefs are held in the system data base. Given these extant goals and beliefs, a subset of KAs in the system will be relevant (applicable). One of these KAs will then be chosen for execution.

In the course of traversing the body of the chosen KA, new subgoals will be posted and new beliefs will be derived. When new goals are pushed onto the goal stack, the interpreter checks to see if any new KAs are relevant, and executes them. Likewise, whenever a new belief is added to the data base, the interpreter will perform appropriate consistency-maintenance procedures and possibly activate new applicable KAs. During this process, various metalevel KAs may also be called to make choices between alternative paths of execution, to choose between multiple applicable KAs, to decompose composite goals into achievable components, and to make other decisions.

This results in an interleaving of plan selection, formation, and execution. In essence, the system forms a partial overall plan (chooses a KA), figures out near term means (tries to find out how to achieve the first subgoal), executes them, further expands the near-term plan of action, executes further, and so on. At any time, the plans the system is intending to execute (i.e., the selected KAs) are both *partial* and *hierarchical* — that is, while certain general goals have been decided upon, specific questions about the means to achieve these ends are left open to future deliberation.

This approach has many advantages. First, systems generally lack sufficient knowledge to expand a plan of action to the lowest levels of detail - at least if the plan is expected to operate effectively in a real-world situation. The world around us is simply too dynamic to anticipate all circumstances. By finding and executing relevant procedures only when needed and only when sufficient information is available to make wise decisions, the system stands a better chance of achieving its goals under real-time constraints.

Because the system is repeatedly assessing its current set of goals, beliefs, and the applicability of KAs, the system also exhibits a very reactive form of behavior, rather than being merely goal-driven. By reactive, we mean more than a capability of modifying current plans in order to accomplish given goals; a reactive system should also be able to *completely change its focus* and pursue new goals when the situation warrants it. This is essential for domains in which emergencies can occur and is an integral component of human practical reasoning.

Because PRS expands plans dynamically and incrementally and also allows for new reactive KAs to respond when they are relevant, there are frequent opportunities for it to react to new situations and to change goals. The system is therefore able to modify its intentions rapidly on the basis of what it currently perceives as well as upon what it already believes, intends, and desires. It can even change its intentions regarding its own reasoning processes - for example, the system may decide that, given the current situation, it has no time for further reasoning and must act immediately.

3.5 Multiple Asynchronous PRSs

In some applications, it is necessary to monitor and process many sources of information at the same time. PRS was therefore designed to allow several instantiations of the basic system to run in parallel. Each PRS instantiation has its own data base, goals, and KAs, and operates asynchronously with other PRS instantiations, communicating with them by sending messages. The messages are written into the data base of the receiving PRS, which must then decide what to do with the new information, if anything.

Typically, this decision is made by a fact-invoked KA (in the receiving PRS), which responds upon receipt of the external message. On the basis of such factors as the reliability of the sender, the type of the message, and the beliefs, goals, and current intentions of the receiver, it is determined what to do about the message - for example, to acquire a new belief, establish a new goal, or modify intentions.

We have found the ability to perform multiple activities concurrently to be crucial in the robot domain. Although some systems do generate plans, portions of which can be executed in parallel (e.g., NOAH [28] and SIPE [34]), our motivations for parallelism are quite different. In our case, the parallelism is essential for processing the constant stream of sensory information and for controlling devices continuously. That is, parallelism is required for the system's proper operation. In NOAH and SIPE, however, the parallelism is simply fortuitous and does not result from any demands on processing speed or distributed functionality.

4 Flakey the Robot

Flakey was designed and built within SRI's Artificial Intelligence Center, and is being used by several research teams to test software-organization ideas. It contains two onboard computers, a SUN II workstation (with 42Mb disk) and a Z80 microprocessor. The Z80 is the low-level controller, receiving instructions from, and returning current information to, the SUN. The SUN, in turn, can be connected to an ethernet cable, allowing the robot to operate in either stand-alone or remote-control modes. The SUN can also be accessed from a small console on Flakey itself.

The Z80 manages 12 sonars, 16 bumper contacts, and 2 stepper motors for left and right wheels. Voice output and video input are managed by the SUN. A robot arm will be added in the future. The application described here uses only the sonars, voice, and wheels.

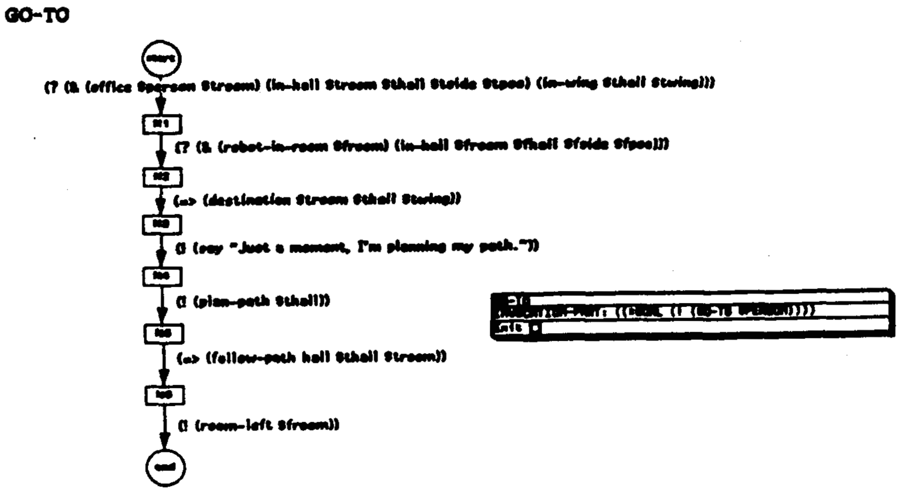

GO-TO
start
(? (& (office $person $troom) (in-hall $troom $thall $tside $tpos) (in-wing $thall $twing)))
N1
(? (& (robot-in-room $froom) (in-hall $froom $fhall $fside $fpos)))
N2
(=> (destination $troom $thall $twing))
N3
(! (say "Just a moment, I'm planning my path."))
N4
(! (plan-path $thall))
N5
(=> (follow-path hall $thall $troom))
N6
(! (room-left $froom))
end
INFO:
Info

Figure 2: Top-level Strategy

The 12 sonars are located approximately 5 inches off the ground, 4 facing forward, 4 backward, and 2 on each side. To obtain a sonar reading, the SUN must issue a request to the Z80 and then wait until the result has been returned. While waiting, the SUN can continue with other processing. At present, the SUN can obtain no more than a few sonar readings per second.

The motors for the left and right wheels can be controlled independently, again by having the SUN send a request to the Z80. For each wheel one can specify a desired distance, a maximum forward speed, and a desired acceleration. The Z80 uses the given acceleration to achieve the maximum speed compatible with the desired distance.

Changing direction is done by requesting different speeds for the two wheels. When the robot is stationary, this can be reduced to a simple rotation; when the robot is moving, more complex algorithms are required. Direction changes are much more difficult when they must be negotiated during a forward acceleration.

As well as receiving the desired values of distance, speed, and acceleration from the SUN, the Z80 transmits current actual values to the SUN. This is done using interrupts that occur at a rate of approximately fifty times per second. The Z80 also runs a position integrator, thus making available the robot's position and orientation relative to particular reference axes. In line with our wish to avoid reliance on dead reckoning, however, we did not use the position integrator for the top-level navigation task; it was used, however, for such low-level tasks as estimating the robot's alignment within a hallway.

There is significant noise in every measurement available to the SUN. The sonars, while generally accurate to about 5 millimeters, can occasionally return invalid readings and can also fail to see an object if the angle of incidence is high enough. Furthermore, Flakey's sonars sense the closest object within a 30-degree cone, so that open doorways are not seen until the sonars are well past the doorpost. Similarly, Flakey will stop within about 5 millimeters of the requested distance and will travel at speeds which fluctuate up to 10 millimeters/second above and below the requested maximum speed.

5 The Domain Knowledge

The scenario described in the introduction includes problems of route planning, navigation to keep on route, and various general tasks such as malfunction handling and requests for information. In this paper, we will concentrate on the route planning and navigation tasks. However, it is important to realize that the knowledge representation provided by PRS is used for reasoning about all tasks that the system performs.

The way the robot (that is, Flakey under the control of PRS) solves the tasks of the space station scenario is roughly as follows. To reach a particular destination, it knows that it must first plan a route and then navigate that route to the desired location (see the KA depicted in Figure 2). In planning the route, the robot uses knowledge of the topology of the station to work out a route to the target location, as is typically done in navigation tasks for autonomous robots [6,7,22]. The topological knowledge is of a very high-level form, stating which rooms are in which corridors and how corridors are connected. A plan formed by the robot is also high-level, typically having the following kind of form: "Travel to the end of the corridor, turn right, then go to the third room on the left." The robot's knowledge of the topology of the problem domain is stored in its data base, and its knowledge of how to plan a route is represented in various route-planning KAs (see Figures 4, 5, and 6). This is quite different from the approach adopted by traditional AI planners, which would find a route by symbolically executing the actual operators specifying possible movements through rooms and down hallways.

A different set of KAs is used for navigating the route mapped out by the route-planning KAs (see Figures 7, 8, and 9). The navigation KAs perform such tasks as sensing the environment, determining when to turn, adjusting bearings where necessary, and counting doors and other openings.

Yet other KAs perform the various other tasks required of the robot. Many of these are described by us elsewhere [17]. Metalevel KAs choose between different means to realize any given goal, and determine the priority of tasks when mutually inconsistent goals (such as diagnosing a jet failure and fetching a wrench) arise. If the robot's route plan fails, the route-planning KAs can again take over and replan a route to the target destination. In the implementation described herein, however, we have not provided any KAs for reestablishing location once the robot has left its room of departure, and so it does not currently exhibit any replanning capability.

5.1 The Planning Space

As stated above, the robot's route planning is done in a very abstract space containing only topological information about how the rooms and hallways connect. It is, in fact, the kind of map found in street or building directories, stripped of precise distances and angles. This is quite natural: when one thinks of going home from the office, one considers primarily the topology of the hallways, footpaths, and roads to be followed, not precisely how long each is, nor the consequences of drifting from side to side — that is too low a level of detail to be considered before setting out along the chosen route.

The three primary KAs used to plan paths are shown in Figures 4, 5, and 6. Given knowledge of the start- and end-points, they first select some intermediate point. They then repeat the process for the two resulting subpaths until all paths are reduced to straight-line trajectories along single hallways. Although it is not the planner we would advocate for more general route planning, it is quite sufficient for our purposes. Indeed, the top-level route planning is probably the simplest aspect of the navigation task.

The topological information needed for route planning is stored in the system data base as a set of facts (beliefs) about how wings, hallways, and rooms are connected. These include facts of the form `(conn j1 k1 j4 direct)` (hallway j1 is connected to hallway k1 DIRECTLY via hallway j4 rather than indirectly via yet further connections), `(in-wing j1 jwing)` (hallway j1 is in wing jwing), and `(in-hall ej225 j1 east 14)` (room ej225 is in hall j1 on the east side of the hall, fourteen rooms from the end). A typical plan constructed by the path-planning KAs is shown in Figure 3. This plan was formed to satisfy the goal of reaching a target room (ej270 in wing j2) from the robot's present location (ej233 in wing j1) and was produced in less than a second. No further predictive planning is required for the robot to negotiate the path.

```
(follow-path hall ej233 j1)
(follow-path hall j1 j4)
(follow-path hall j4 j2)
(follow-path hall j2 ej270)
```

Figure 3: Route from ej233 to ej270

It is important to emphasize that, even during this relatively short planning stage, the robot remains continuously reactive. Thus, for example, should the robot notice indication of a jet failure on the space station, it may well decide to interrupt its route planning and attend instead to the task of remedying the jet problem.

5.2 Reactive, Goal-Directed Behavior

The KAs used to navigate the route fall into three classes: those that interpret the path plan and establish intermediate target locations, those that are used to follow the path, and those that handle critical tasks such as obstacle avoidance and *reacting to emergencies. Each KA manifests a self-contained* behavior, possibly including both sensory and effector components. Moreover, the set of KAs is naturally partitioned according to level of functionality (cf. [7]): low-level functions (emergency reactions, obstacle avoidance, etc.), middle-level functions (following already established paths and trajectories), and higher-level functions (figuring out how to execute a topological route). All of these KAs are simultaneously active, performing their function whenever they may be applicable. Thus, while trying to follow a path down a hallway, an obstacle avoidance procedure may simultaneously cause the robot to veer slightly from its original path.

Once a plan is formed by the route-planning KAs, that plan must be converted into some useable form. Ideally, the plan shown in Figure 3 should be represented as a procedural KA containing the goals "leave room ej233 and go into hall j1," "go to the j1-j4 junction," etc. Since it is not currently possible for KAs to create or modify other KAs, we have, instead, defined a group of KAs that react to the presence of a plan (in the data base) by translating it into the appropriate sequence of subgoals. Each leg of the original plan generates subgoals such as turning a corner, travelling the hallway, and updating the data base to indicate progress. The second group of navigation KAs reacts to these goals by actually doing the work of reading the sonars, interpreting the readings, counting doorways, aligning the robot within the hallway, and watching for obstacles ahead.

For example, consider the KAs in Figures 7 and 8. After having planned out a path as directed by the KA in Figure 2, the robot is given a goal of the form `(! (room-left $froom))` (the variable `$froom` will be bound to some particular constant representing the room that the robot is trying to leave). The KA in Figure 8 will respond and actually perform steps for leaving the given room. The last step in this KA will insert a fact into the system database of the form `(origin $froom $fhall)` (again, the variables will be bound to specific constants). This fact alerts a path interpretation KA (depicted in Figure 7) that the robot is now ready to execute a leg of its path, and supplies the KA with the robot's starting position (i.e., the room adjacent to the robot, `$froom`, and the hall in which it stands, `$fhall`). Assuming that the facts describing a path have been placed in the database (for example, *the set of facts* in Figure 3), the fact-invoked `FIND-NEXT` KA in Figure 7 will respond and begin to interpret the path. It will then proceed and travel down the hallway as instructed. This will in turn establish a new origin position, thereby allowing for the next step of the path to be executed.

A third group of KAs reacts to contingencies observed by the robot as it interprets and executes its path. For example, these will include KAs that respond to the presence of an obstacle ahead (see Figure 9) or the fact that an emergency light has been seen. These reactive KAs are invoked solely on the basis of certain facts becoming known to the robot. Implicit in their invocation, however, is an underlying goal to "avoid obstacles" or "remain safe."

Since a fact-invoked KA can be executed as soon as its triggering facts are known, the KAs invoked by these contingencies can interrupt whatever else is happening. Of course, this may not always be desirable. Ideally, domain-specific metalevel KAs should determine whether and when preemption is desirable, but, at this stage of the project, we have not used metalevel KAs besides those provided as PRS defaults (which give immediate preemption). An alternative to preemption is to send a contingency message to another PRS instantiation that can process that message in parallel.

5.3 Parallelism and Mediation

Because of the real-time constraints and the need for performing several tasks concurrently, it is desirable to use multiple instances of PRS running in parallel. In particular, parallelism can be used for handling contingencies without interrupting other ongoing tasks. Multiple PRS instantiations can also be used as information filters to protect other instantiations from a barrage of uninteresting sensory information. (The need for such filters arises in many *problem domains – for example, in monitoring sensors* on the space shuttle [4].) The strongest reasons, however, have to do with the inherently parallel and largely independent nature of the various computations that must be performed in dynamic environments.

For example, as the robot rolls down a hallway, it fires its sonars to determine how far it is from the walls, and also to count doors. Suppose it decides that the walls are too close and a change in course is warranted. Because speed changes cannot be accomplished instantaneously, changing course may take as long as two seconds. This is long enough for the robot to roll past a doorway. If the procedure that monitors sonar readings is interrupted to effect the

PLAN-PATH

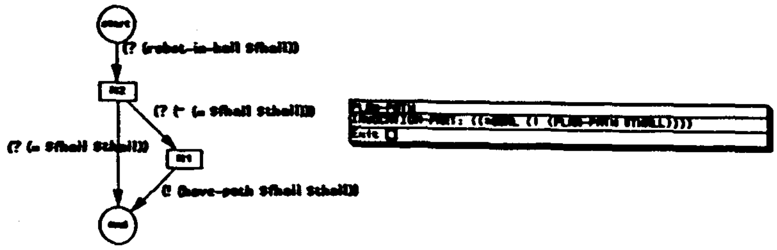

start
(? (robot-in-hall $fhall))
N2
(? (~ (= $fhall $thall)))
(? (= $fhall $thall))
N1
(! (have-path $fhall $thall))
end
PLAN-PATH
INVOCATION-PART: ((GOAL (! (PLAN-PATH $THALL))))
Exit □

Figure 4: Path Planning KA

HAVE-PATH

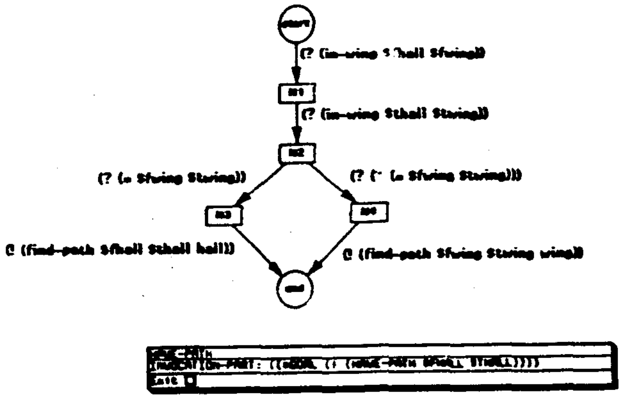

start
(? (in-wing $fhall $fwing))
N1
(? (in-wing $thall $twing))
N2
(? (= $fwing $twing))
(? (~ (= $fwing $twing)))
N3
N4
(! (find-path $fhall $thall hall))
(! (find-path $fwing $twing wing))
end
HAVE-PATH
INVOCATION-PART: ((GOAL (! (HAVE-PATH $FHALL $THALL))))
Exit □

Figure 5: Path Planning KA

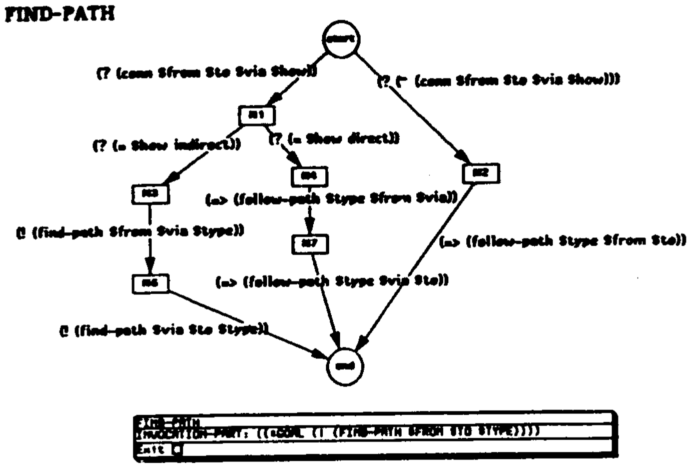

FIND-PATH
(? (conn $from $to $via $how))
(? (~ (conn $from $to $via $how)))
(? (= $how indirect))
(? (= $how direct))
(=> (follow-path $type $from $via))
(! (find-path $from $via $type))
(=> (follow-path $type $from $to))
(=> (follow-path $type $via $to))
(! (find-path $via $to $type))
FIND-PATH
INVOCATION-PART: ((*GOAL (! (FIND-PATH $FROM $TO $TYPE))))
Exit □

Figure 6: Path Planning KA

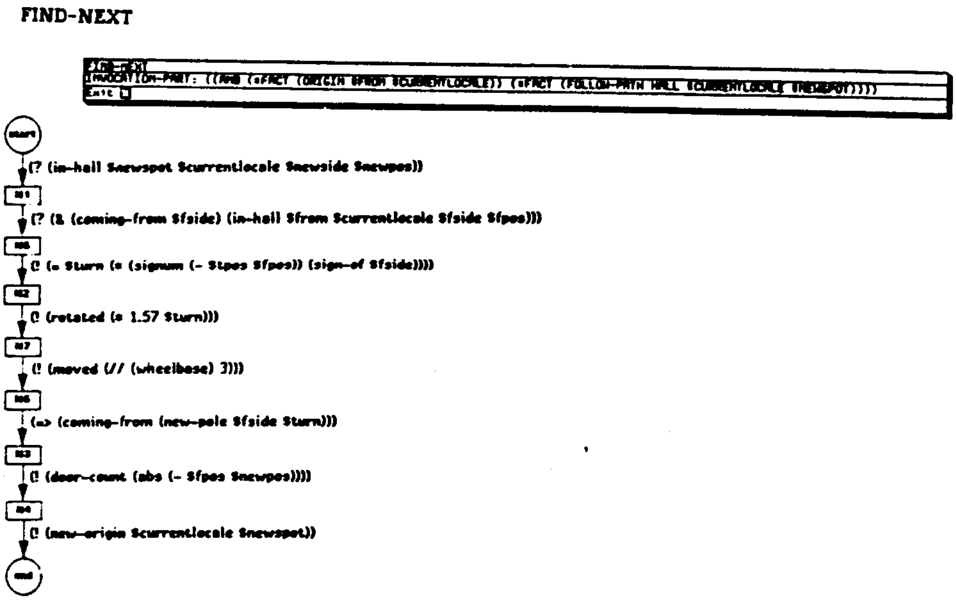

FIND-NEXT
FIND-NEXT
INVOCATION-PART: ((AND (*FACT (ORIGIN $FROM $CURRENTLOCALE)) (*FACT (FOLLOW-PATH HALL $CURRENTLOCALE $NEWSPOT))))
Exit □
(? (in-hall $newspot $currentlocale $newside $newpos))
(? (& (coming-from $fside) (in-hall $from $currentlocale $fside $fpos)))
(! (= $turn (* (signum (- $tpos $fpos)) (sign-of $fside))))
(! (rotated (* 1.57 $turn)))
(! (moved (// (wheelbase) 3)))
(=> (coming-from (new-pole $fside $turn)))
(! (door-count (abs (- $fpos $newpos))))
(! (new-origin $currentlocale $newspot))

Figure 7: Plan Interpretation KA

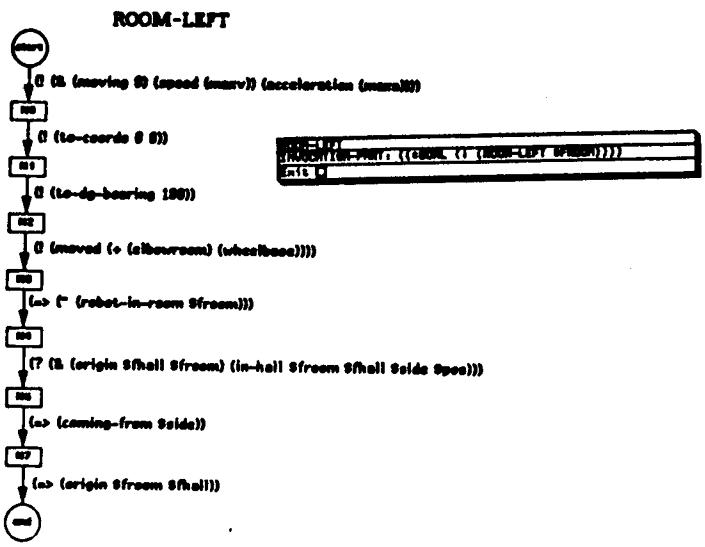

ROOM-LEFT
start
(! (& (moving 0) (speed (maxv)) (acceleration (maxa))))
N0
(! (to-coords 0 0))
N1
(! (to-dg-bearing 180))
N2
(! (moved (+ (elbowroom) (wheelbase))))
N3
(=> (~ (robot-in-room $froom)))
N4
(? (& (origin $fhall $froom) (in-hall $froom $fhall $side $pos)))
N6
(=> (coming-from $side))
N7
(=> (origin $froom $fhall))
end
ROOM-LEFT
INVOCATION-PART: ((GOAL (! (ROOM-LEFT $FROOM))))
Exit □

Figure 8: Route Navigation KA

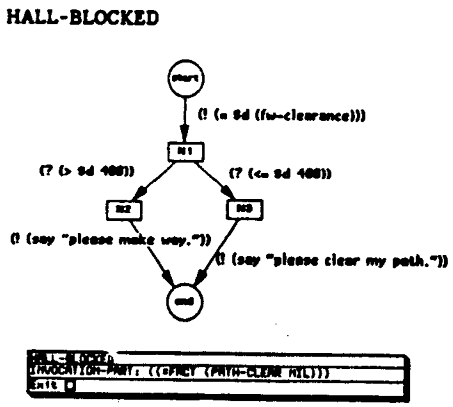

HALL-BLOCKED
start
(! (= $d (fw-clearance)))
N1
(? (> $d 400))
(? (<= $d 400))
N2
N3
(! (say "please make way."))
(! (say "please clear my path."))
end
HALL-BLOCKED
INVOCATION-PART: ((#FACT (PATH-CLEAR NIL)))
Exit □

Figure 9: Reactive KA

course change, the robot might completely miss a door reading. Conversely, delaying course changes for the sake of sonar monitoring could make a collision with a wall inevitable. Of course, travelling at lower speeds would solve the problem, but would also render the robot too slow to be useful.

The most effective way to handle this problem is to allow multiple PRS instantiations to execute concurrently. Running several instantiations asynchronously has its own problems, however. For example, it is desirable to have one PRS instantiation devoted to the task of keeping the robot in the center of the hallway, with another driving the robot to the target location and adjusting speed appropriately (e.g., slowing down when approaching the target). Changes in course are effected by changing the relative velocities of the two wheels, depending on their current velocity, and changes in speed by changing the accelerations of the wheels. The problem is that, if both tasks need to be performed at once, the required wheel operations may interfere with one another. This is an interesting example of a situation in which domain-independent decomposition operators will not work – because of the real-time constraints of the problem domain, it is not suitable to achieve one goal (say, a change in direction) and subsequently achieve the other (change in speed); neither can each goal be achieved independently, as the means for accomplishing these goals interact with one another.

To mediate between interacting goals, we chose to implement a third PRS capable of accepting both speed and direction change requests asynchronously. This PRS could be viewed as a virtual controller. Because the virtual controller is in complete control of the wheels, it can issue instructions that achieve both kinds of requests at once. In this respect, it serves as a special-purpose solution to a particular kind of conjunctive goal; goals to change both speed and direction are decomposed into independent goals to change the left and right wheel speeds.

Related to the problem of interacting goals is that of goal conflict: just as one may have possibly conflicting beliefs about a situation that need to be resolved (the problem of situation assessment), one may also have conflicting goals (or desires) that need mediation [18]. For example, the virtual controller discussed above often gets conflicting speed requests from KAs: the hallway traversal KA might request that a certain velocity be maintained, the KA that detects approach of the target location may request a decrease in velocity, and the KA that detects obstacles could request that the robot stop altogether. At the same time, other KAs might request changes in direction to stay in the center of the hall or to pass around small obstacles.

To resolve these conflicting goals, the virtual controller has to be able to reason about their urgency and criticality. This, in turn, may involve further communication with the systems requesting these goals. Our present solution is to define domain-dependent mediators where necessary, but, at present, no general approach to this problem has been attempted.

5.4 Coping with Reality

Our initial implementation of the robot application used multiple PRS instances interacting with a robot simulator, all running on the Symbolics 3600 This worked well, and demonstrated the suitability of the system for controlling complex autonomous devices. That done, we began work on driving the real robot. This transfer took considerably longer than estimated. Two major problems caused this divergence between expectations and reality.

First, because PRS was implemented on a Lisp machine, interaction with Flakey was confined to occur via an ethernet cable. Software for remote procedure calls over the ethernet limited communication to 15 function calls per second – too slow for timely response to sensor input. Consequently, we were forced to transfer much of the functionality of PRS to Flakey's SUN. This required translating the functionality of the lower-level KAs into C code, as well as explicit coding for message and clock-signal handling. Unfortunately Flakey's operating system also did not support interprocess communication at the bandwidth and efficiency we wanted. This forced us to implement communication through shared memory, with all the concommitant synchronization code needed. After these efforts, the information flowing over the ethernet was at the level of "move N doors" (PRS to Flakey) and "I'm stopping for an obstacle" (Flakey to PRS). Obviously, the translated system is no longer solely constructed from instances of PRS. As a result, our final implementation is considerably more constrained than the simulation version in its ability to reason about its low-level actions and to react appropriately to changing goals.

The second obstacle to translating from our simulated application to the one that could function in the physical world is the nature of the real world itself. A realistic environment is simply not controlled enough to foster efficient debugging. It is hard to repeat experiments (and get the same bugs), time delays become critical, and the behaviors of real sensors and effectors can differ significantly from simulated ones.

The configuration of our current application system is shown is Figure 10. Three machines are involved, a Symbolics 3600, a SUN, and a Z80, running six application processes. The wheels and sonars are also depicted, and may be regarded as physical processes. The rectangular box represents the SUN's shared memory area; arrows represent interprocess communication.

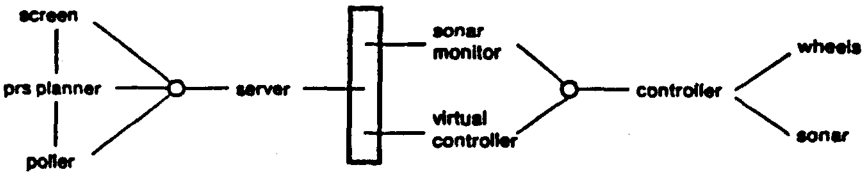


Figure 10: Processes Used in the Implementation

6 Discussion

The primary purpose of this research was to show that the BDI architecture of PRS, the partial hierarchical planning strategy it supports, and its metalevel (reflective) capabilities could be effective in real-world dynamic domains. Furthermore, the design of PRS meets some of the more important desiderata for autonomous systems: modularity, awareness, and robustness [18]. In this section, we will briefly compare our approach to other work in the areas of planning and robotics.

The partial hierarchical planning strategy and the reflective reasoning capabilities used by PRS allow many of the difficulties associated with traditional planning systems to be avoided, without denying the ability to plan ahead when necessary. By finding and executing relevant procedures only when sufficient information is available to make wise decisions, the system stands a better chance of achieving its goals under real-time constraints.

For example, the speed and direction of the robot is determined during plan execution, and depends on such things as proximity of obstacles and the actual course of the robot. Even the method for determining this course depends dynamically on the situation, such as whether the robot is between two hallway walls, adjacent to an open door, at a T-intersection, or passing an unknown obstacle. Similarly, the choice of how to normalize fuel or oxidant tank pressure while handling a jet failure depends on observations made during the diagnostic process.

Because PRS expands plans dynamically and incrementally, there are also frequent opportunities to react to new situations and changing goals. The system is therefore able to modify its intentions (plans of action) rapidly on the basis of what it currently perceives as well as upon what it already believes, intends, and desires. For example, when the system notices a jet-fail alarm while it is attempting to fetch a wrench, it has the ability to reason about the priorities of these tasks, and, if so decided, suspend the wrench-fetching task while it attends to the jet failure. Indeed, the system even continues to monitor the world while it is route planning (in contrast to most robot systems), and this activity too can be interrupted if the situation so demands.

The powerful control constructs used in PRS procedure bodies (such as conditionals, loops, and recursion) are also advantageous. As a result, the robot can display behaviors of the form "do X until B becomes true." When X is "maintain speed at 400mm/sec" and B is "N doorways have been observed" we see why we could dispense with coordinate grids and dead reckoning: we could *define* the robot's behaviors in terms of conditions that changed over time. In contrast, classical planning systems often have difficulty in reasoning about such behavior and are thus restricted to using unchanging features such as fixed locations or distances.

PRS is also very robust in that there may be many different KAs available for achieving some given goal. Each may vary in its ability to accomplish the goal, and in its applicability and appropriateness in particular situations. Thus, if there is insufficient information about the current situation to allow one KA to be used, some other - perhaps less reliable - KA may be available instead. For example, if a topological map of an area is unavailable for planning purposes, the robot need not be rendered ineffective - there may, for example, be some other KA that sets the robot off in the general direction of the target. Parallelism and reactivity also help in providing robustness. For example, if one PRS instantiation is busy planning a route, lower-level instantiations can remain active, monitoring changes to the environment, keeping the robot in a stable configuration, and avoiding dangers.

The system we propcse also meets many of the criteria of rational agency advanced in the philosophical literature on practical reasoning (e.g., see the work of Bratman [5]). Driven by the demands of explaining resource-boundedness and inter- and iutra-agent coordination, recent work in the philosophy of action has moved beyond belief-desire architectures for rational agents and has provided insights into the nature of plans and intentions, and especially the nature of intention formation.

In particular, plans are viewed as being subject to two kinds of constraints: *consistency constraints* and requirements of *means-ends coherence*. That is, an agent's plans need to be both internally consistent and consistent with its beliefs and goals. It should be possible for an agent's plans to be successfully executed (that is, to achieve the more important goals of the system) in a world in which its beliefs are true. Secondly, plans, though partial, need to be filled in to a certain extent as time goes by, with sub-plans concerning means, preliminary steps, and relatively specific courses of action. These subplans must be at least as extensive as the agent believes is required to successfully execute the plan; otherwise they will suffer for means-ends incoherence.

These constraints on the beliefs, desires (goals), and intentions of an agent are realized by the system proposed herein, and as such it can be viewed as an implementation of a rational agent. In addition, the notion of intention we use meets the major requirements put forward by Bratman [5], who considers intentions to have the following properties:

- They lead to action,
- They are parts of larger plans,
- They involve commitment,
- They constrain the adoption of other intentions,
- They are adopted relative to the beliefs, goals, and other intentions of the system.

Of course, our system is far from manifesting the behavioral complexity of real rational agents; however, it is a step in the direction of a better understanding of rational action.

In contrast to most AI planning work, research in robotics *has* been very concerned with reactivity and feedback [2,18,23]. However, most of this work has not been concerned with general problem solving and commonsense reasoning - the work is almost exclusively devoted to problems of navigation and execution of low-level actions. Furthermore, the reactivity is not of the general form we advocated above; although the systems can adjust the *means* for achieving given goals depending on incoming sensory information, they do not exhibit the ability to completely change goal priorities, to modify, defer, or abandon current plans, or to reason about what is best to do in light of the current situation.

Recently, Brooks [7] has advanced some intriguing ideas concerning the structure of autonomous systems. Rather than the horizontal structure typical of most robot systems (where lower levels are restricted to performing basic sensory and effector processing, and the higher levels to planning and reasoning) Brooks advocates a vertical decomposition in which distinct *behaviors* of the robot are separately realized, each making use of the robot's sensory, effector, and reasoning capabilities as needed.

Similarly, PRS provides a vertical, rather than horizontal, decomposition of the robot task domain. Each KA defines a particular behavior of the system, and can involve both processing of sensory information and the execution of effector actions. For example, there is a KA that manifests a behavior to remain clear of obstacles, another KA whose behavior corresponds to keeping a straight course in a corridor, and yet another that chooses and traverses roufes from one room to another. All these KAs use both sensors and effectors to greater or lesser degrees - there is no single subsystem that preprocesses the sensory data before sending it to the reasoning system, and there is no post-processing of plan information that determines actual effector actions.

In this sense, our system is very similar in structure to that proposed by Brooks - indeed, it can claim the same positive benefits [8]:

- There are many parallel paths of control through the system [many different procedures can be used in a given situation] Thus the performance of the system in a given situation is not dependent on the performance of the weakest link in that situation. Rather, it is dependent on the strongest relevant behavior for the situation
- Often more than one behavior is appropriate in a given situation. The fact that the behaviors are [can be] generated by parallel systems [multiple PRS instances] provides redundancy and robustness to the overall system. There is no central bottleneck [through which all the processing or reasoning must occur].

- With some discipline in structuring the decomposition, the individual task-achieving behaviors can run on separate pieces of hardware. Thus [the design] leads to a natural mapping of the complete intelligent system onto a parallel machine. The benefits are threefold: (1) redundancy again; (2) speedup; and (3) a *naturally extensible system*.

The main difference between our system and that advanced by Brooks is that we employ a much more general mechanism for selecting between appropriate behaviors than he does: whereas Brooks uses inhibitory and excitatory links to integrate the set of behaviors defined by each of the system's functional components, we use general metalevel procedures and communication protocols to perform the selection and integration. Of course, such generality will likely preclude meeting some of the real-time constraints of the environment, in which case the metalevel procedures might need to be compiled into a form closer to that envisaged by Brooks. Similarly, while our system naturally maps onto *coarse-grain parallel machines*, sophisticated compilation techniques would be required to map the lower-level functions onto highly parallel architectures.

Currently, PRS does not reason about other subsystems (i.e., other PRS instantiations) in any but the simplest ways. However, the message-passing mechanisms we have employed should allow us to integrate more complex reasoning about interprocess communication, such as described by Cohen and Levesque [9]. Reasoning about process interference and synchronisation is also important where concurrency is involved. The mechanisms developed by us for reasoning about these problems [12,13,14,19,20,31] could also potentially be integrated within PRS. Our future research plans include both work on communication and synchronisation within the PRS framework.

Finally, in giving a description of the PRS architecture, it is important to note that the actual *implementation* of PRS is not of primary concern. That is, while we believe that attributing beliefs, desires, and intentions to autonomous systems can aid in specifying complex behaviors of these systems, and can assist in communicating and interacting with them, we are not demanding that such systems actually be structured with distinct data structures that explicitly represent these psychological attitudes (although, indeed, that is the way we have chosen to implement our system). We can instead view the specification of the PRS system, together with the various metalevel and object-level KAs, simply as a *description* of the desired behavior of the robot. This description, suitably formalised,[3] could be realised in (or compiled into) any suitable implementation we choose. In particular, the beliefs, desires, and intentions of the robot may no longer be explicitly represented within the system. Some interesting work on this problem is being carried out currently by Rosenschein and Kaelbling [26].

7 Limitations

The primary thrust of this work has been to evaluate an architecture for autonomous systems that provides a means of achieving goal-directed, yet reactive, behavior. We have made enough progress to show that this approach works. However, the research is only in its initial stages and there are a number of limitations that still need to be addressed.

First, there is a class of facts our current system must be told; for instance, the robot's starting location. If the robot is initialised in some unknown position on the topological map, the planner will abort. It would be straightforward to solve these problems by including KAs that ask for the missing information, but a completely autonomous recovery would be a much more challenging problem. Possible approaches might involve exploration of the terrain (including movement around the neighboring area) and pattern matching onto known topological landmarks.

Second, there are many assumptions behind the procedures (KAs) used. For example, we have assumed that hallways are straight and corners rectangular; that all hallways are the same width and have that width for their entire length (except for doorways and intersecting halls); that there is only one layer of obstacles in front of any wall (nowhere is there a garbage can in front of a cupboard); that all doors are open and unobstructed; and that other objects move much slower than the robot.

We have also made assumptions that limit the robot's reactivity. For example, we assume that the robot does, in fact, arrive at the junction it planned to reach. If the robot miscounts doorways, it will stop in the wrong place, turn, and start the next leg of its journey without realizing its mistake. The result is generally that the robot will be found begging a wall to "please make way." If the robot realized it was in the wrong position, it could replan to achieve its goals. However, because we assume that the door count is always right, the route planner is never reinvoked.

In addition, some goals are not made as explicit as one would like, but are implicit in the KAs used by the robot. For example, the robot is designed to move as fast as possible without miscounting doorways and to travel along the center of the hallway while accepting the fact that this ideal will rarely be achieved. Such goals are not represented explicitly within KAs. Handling the first kind of goal ("move as fast as possible") would be relatively straightforward, requiring simply that axioms relating robot speed and perceptive capabilities be provided to the system. However, it is not obvious how to explicitly represent the second kind of goal, in which the system attempts to maintain a particular condition but expects at best only to approximate it.

Finally, increased parallelism would have been preferable, allowing the robot to perform more tasks concurrently. For example, we could have included many more low-level procedures for, say, avoiding dangers and exploring the surroundings. This would have provided a much more severe test of the system's capability to coordinate various plans of action, to modify intentions appropriately, and to change its focus of attention.

8 Acknowledgments

Joshua Singer and Mabry Tyson debugged early PRS code and extended the implementation as needed. Stan Reifel and Sandy Wells designed Flakey and its interfaces, and assisted with the implementation described herein. We have also benefited from our participation and interactions with members of CSLI's Rational Agency Group (RATAG), particularly Michael Bratman, Phil Cohen, Kurt Konolige, David Israel, and Martha Pollack. Leslie Pack-Kaelbling and Stan Rosenschein also provided helpful advice and interesting comments.

References

[1] J. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, 26 832-843, 1982.

[2] J. Albus, *Brains, behavior, and robotics*. Byte Books, 1985.

[3] S. Amarel, "On Representations of Problems of Reasoning About Actions," in *Machine Intelligence 3*, ed. D. Michie, Edinburgh University Press, Edinburgh, 1968.

[4] G. Boy, "HORSES: A Human-Orbital Refueling System Expert System," *Report 2, NASA Ames Research Center*, Aero-Space Human Factors Research Division, Moffett Field, California, 1985.

[3] While the semantics of the KAs is formally defined [16,17], we have yet to formally specify the operation of the interpreter underlying PRS. If this were done, we would then have a completely formal specification of the desired behavior of the robot.

[5] M. Bratman, "Intention, Plans, and Practical Reason" Harvard University Press, Cambridge, Massachusetts, forthcoming.

[6] R.A. Brooks, "Visual Map Making for a Mobile Robot," *Proceedings of IEEE Conference on Robotics and Automation*, RA-1, March 1985, pp.31-41.

[7] R. Brooks, "A Robust Layered Control System for a Mobile Robot," AI Memo 864, AI Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1985.

[8] R. Brooks, "Achieving Artificial Intelligence Through Building Robots," AI Memo 899, AI Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1986.

[9] P. R. Cohen and H.J. Levesque, "Speech Acts and Rationality," *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, Illinois, pp.49-60, 1985.

[10] P.R. Davis and R.T. Chien, "Using and Reusing Partial Plans," *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Cambridge, Massachussets, pp. 494, 1977.

[11] R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, 2:189–208, 1971.

[12] M.P. Georgeff, "Communication and Interaction in Multi-Agent Planning," *Proceedings of AAAI-83, National Conference on Artificial Intelligence*, Washington, D.C., 1983.

[13] M. P. Georgeff, "A Theory of Action for Multiagent Planning," in *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, Texas, 1984.

[14] M.P. Georgeff, "A Theory of Process," *Proceedings of the Workshop on Distributed Artificial Intelligence*, Sea Ranch, California (1985)

[15] M. P. Georgeff and A. L. Lansky, "A System for Reasoning in Dynamic Domains: Fault Diagnosis on the Space Shuttle." Technical Note 375, Artificial Intelligence Center, SRI International, Menlo Park, California, 1985.

[16] M. P. Georgeff, A. L. Lansky, and P. Bessiere, "Procedural Logic," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, 1985.

[17] M. P. Georgeff and A. L. Lansky, "Procedural Knowledge," to appear in *Proceedings of IEEE, Special Issue on Knowledge Representation*, October 1986.

[18] L. P. Kaelbling, "An Architecture for Intelligent Reactive Systems," 1986 Workshop on Planning and Reasoning about Action, Timberline, Oregon, July 1986.

[19] A. L. Lansky, "Behavioral Specification and Planning for Multiagent Domains," Technical Note 360, Artificial Intelligence Center, SRI International, Menlo Park, California, 1985.

[20] A. L. Lansky, "A Representation of Parallel Activity Based on Events, Structure, and Causality," in *Reasoning about Actions and Plans*, Proceedings of the 1986 Workshop on Planning and Reasoning about Action, Timberline, Oregon, Morgan Kaufmann Publishers, 1987.

[21] D. McDermott, "Flexibility and Efficiency in a Computer Program for Designing Circuits," Technical Report AI-TR-402, MIT AI LAb, Cambridge, Massachusetts, 1977.

[22] H P. Moravec, "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE*, Vol 71, pp. 872-884, 1983.

[23] N. J. Nilsson, "Flakey the Robot," Technical Note 323, Artificial Intelligence Center, SRI International, Menlo Park, California, 1984.

[24] N.J. Nilsson, "Triangle Tables: A Proposal for a Robot Programming Language," Technical Note 347, Artificial Intelligence Center, SRI International, Menlo Park, California, 1985.

[25] S J. Rosenschein, "Plan Synthesis: A Logical Perspective," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 331-337, Vancouver, British Columbia, 1981.

[26] S. J. Rosenschein and L. P. Kaelbling, "The Synthesis of Digital Machines With Provable Epistemic Properties." *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge*, Asilomar, California, pp. 83-98.

[27] E.D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence*, Vol 5, pp. 115-135, 1974.

[28] E. D. Sacerdoti, *A Structure for Plans And Behavior*, Elsevier, North Holland, New York, 1977.

[29] S. M. Schieber, "Solving Problems in an Uncertain World," Undergraduate thesis, Applied Mathematics Department, Harvard College, Cambridge, Massachusetts, 1981.

[30] M. Stefik, "Planning With Constraints," *Artificial Intelligence*, 16:111-140, 1981.

[31] C.J. Stuart, "Synchronization of Multiagent Plans Using a Temporal Logic Theorem Prover," Technical Note 350, Artificial Intelligence Center, SRI International, Menlo Park, California, 1985.

[32] A. Tate, "Goal Structure — Capturing the Intent of Plans," *Proceedings of the Sixth European Conference on Artificial Intelligence*, pp. 273-276, 1984.

[33] S. Vere, "Planning in Time: Windows and Durations for Activities and Goals," Technical Report, Jet Propulsion Laboratory, Pasadena, California.
(Editor's Note: AIAA Paper 83A43951)

[34] D. E. Wilkins, "Domain Independent Planning: Representation and Plan Generation," *Artificial Intelligence*, 22:269-301, 1984.

Route Planning in a Four-Dimensional Environment

M.G. Slack and D.P. Miller
Virginia Polytechnic Institute
Blacksburg, VA 24061

~~ABSTRACT~~

Robots must be able to function in the real world. The real world involves processes and agents that move independently of the actions of the robot, sometimes in an unpredictable manner. ~~This paper presents~~ a real-time integrated route planning and spatial representation system for planning routes through dynamic domains is presented. The system will find the safest most efficient route through space-time as described by a set of user defined evaluation functions. Because the route planning algorithm is highly parallel and can run on an SIMD machine in O(p) time (p is the length of a path), the system will find real-time paths through unpredictable domains when used in an incremental mode. ~~This paper will discuss~~: Spatial representation, an SIMD algorithm for route planning in a dynamic domain, and results from an implementation on a traditional computer architecture are discussed.

1 - Introduction: Route Planning in Dynamic Domains

The ability to represent and plan movements through space is necessary for any autonomous mobile robot. Mechanical error and uncertainty make it impractical to maneuver a robot through a series of complex tasks strictly by dead-reckoning. If dead-reckoning is of limited use, then some navigation capabilities must be brought into play; navigation depends on having some knowledge of the world outside of the robot. Towards this end, a wide variety of spatial representation systems have been developed in recent years.

A variety of techniques have been used to attack different aspects of the spatial representation problem. Topological graphs [Laumond83], [Chatila85] have been used for guiding route planning through a loosely connected set of convex polygons representing free-space areas in an indoor environment. Regions mapped with traversable conduits [McDermott84] have been used successfully for large scale navigation in uncertain environments. Representation of the exteriors of obstacles as the edges of a highly connected graph was used by Davis, allowing detailed knowledge of the environment and its accompanying uncertainty to be captured [Davis84].

Other representations have been used for capturing movement or navigational details necessary for a robot to plan its activites. Configuration space [Lozano-Perez83] provides a computationally tractable approach to calculating the practical steps for moving a robot from one position to another. Using Voronoi diagrams and representations of free space, movements in three-dimensions have been calculated to maintain a robot the maximal possible distance from any obstacle [Brooks82]. Similar methods, when combined with an analysis of the robot's sensors, can calculate a path that is both relatively safe and easy to navigate [Miller85].

Despite the variety of the techniques mentioned above, all of the systems discussed share some basic limitations. None of the systems takes into account the quality of the surface upon which the robot travels, relying on the surface being either traversable or not. Such a restricted view is continually contradicted by the way people move about. People stray off the sidewalk or jay-walk across a street whenever it is convenient and safe, hence a realistic robot should be able to behave similarly. A further limitation is that all the aforementioned systems are designed to operate under static conditions, where the only aspect of the world that changes is the position of the robot. This is an unrealistic and unacceptable limitation for almost all applications.

In addition to being able to function in a dynamic world, a robot should be able to *reason* about dynamic processes and how they may affect it. For example, if a robot knows the local train schedule and needs to get to the other side of the train tracks, then it should use that information when planning to get to the other side. If the robot has information predicting that a long freight train will be coming just before it can reach the tracks, then given the choice between a short path that involves crossing the train tracks, and a slightly longer plan to go under the tracks, the robot should choose the latter plan. Similarly, if the robot's task is to rob a train, then the ability to plot a path that will allow the robot to jump onto the moving train is necessary.

Unpredictable dynamic processes must also be taken into account during route planning. A cavalry robot that "fears" an attack by a tribe of Indian robots would be better off planning to get to the fort across the open plain, rather than passing through the narrow passageway of *Ambush Canyon*. The primary reason for this is an attack in the canyon would more effectively block the robot from its destination, thereby mandating backtracking.

The single property that most distinguishes this work from previous systems is that it models not only space, but time as well. Rather than making a calculation about whether the robot can *traverse* a particular area independent of time, this system models the ability of the robot to traverse that area at different times. We have accounted for temporal as well as spatial changes in the environment. The message passing technique used allows time to be considered while allowing the system designer to model qualities of the domain, such as the cost of moving from one position to another and the ease of traversing a particular area of space. The remainder of this paper describes a representation and route planning system for use in unpredictable dynamic domains.

2 - The Algorithm

This section will describe an algorithm that finds the best path through a predictable n-dimensional space using user-defined evaluation functions. The algorithm provides for an effective representation of space-time and the ability to functionally define predictable static and dynamic objects that map into a subset of a given space-time.

2.1 - Spatial Representation

A path-finding algorithm that is of any use must provide for: 1) an effective representation of space, 2) the relationship between the elements making up the space, and 3) the ability to define the quality of that space with respect to a robot using the generated plans.

To effectivly represent space, this model uses uniformly shaped, n-dimensional hyper-cubes called "nodes". Each node represents a small chunk of space. *Arbitrarily shaped n-dimensional* areas can be defined through the spatial concatenation of nodes along common (n-1 dimensional) surfaces. The collective area occupied by the nodes is called "space", while the remainder of existence is referred to as "void". For example, Figure 1 shows an arbitrary two-dimensional space constructed from the spatial concatenation of square shaped nodes. In general, the size of a node will be of at least sufficient size to subsume the size of the robot.

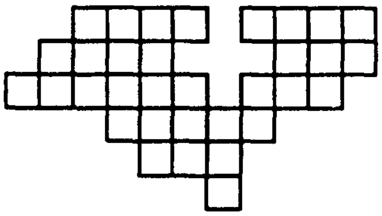
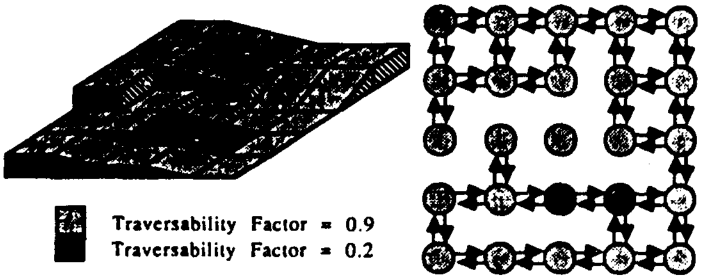


Figure 1

The relationship between adjoining nodes (such as the ability to move from one node to another) is represented by unidirectional links between each of the nodes (see Figure 2b). The reflexive relationship a node has with itself is represented as a link that points back to the node and represents the ability to remain at that node. Associated with each of the links is a cost. The cost represents such things as: whether the surface between the two nodes is continuous, a downgrade, in three-dimensions directly below, or if a node represents a safe place to stop. The function of the links is to provide a communication path over which messages can be sent. A node can have up to $2n$ communication links with its neighboring nodes and one reflexive communication link. Nodes that lie on the edge between space and the void will have fewer communication connections.

The topological features of the space are represented by the cost on links between nodes, whereas the actual traversability of a particular region of space is specified by the node's traversability constant. The traversability constant represents the relative ability of the robot to move over a given node. For example, for a robot with wheels, a concrete floor would have a traversability constant near 1.0 while that of quicksand might be on the order of 0.001.

Consider the spatial representation of the two-dimensional space shown in Figure 2. Part a of the Figure shows the physical layout of the example space. Part b shows how the nodes that define the space are interconnected with one another. The reason for the missing links between some of the nodes is that the cost associated with the link is infinite (the cost on other links is not indicated). As a result, no paths are generated that make those transitions. While the surface is three-dimensional, only a two-dimensional representation is necessary (for this particular example) because the costs on the links between nodes allows for the representation of hills, absorbing the three-dimensional aspect of the space. For example, a link going down hill could have a lower cost than one going up hill. If the features described above are taken together, a robust representation of the salient features of space can be created.



part a

part b

Figure 2

2.2 - Representing predictable objects

For a route planning system to be of value, not only does it have to represent the salient features of space, but it must also provide an effective means for representing predictable objects. Predictable objects are those objects, both static and dynamic, that have fully predictable behavior in both time and space. An object is represented as a function having the defined space and time in its domain, and some subset of the nodes that make up space in its range. The set of nodes generated on any application of the function consists of those nodes in the given space that are occupied (fully or partially) by the object during the given time. For example, consider a model of a simple two-dimensional revolving door. This can be represented by defining a function that has four nodes forming a square in its range. Then, by setting the function to map onto two of the nodes that are diagonal during odd time units and onto the other two nodes during even times, a simple, predictable, revolving door can be created. Such functions can be encoded into each of the nodes at setup time, thus reducing the amount of outside communication during operation.

2.3 - An SIMD Algorithm for Route Planning

The representation of the spatial features and predictable objects thus far described provides a basis for an algorithm that can be directly implemented on an SIMD machine, such as that in [Hillis85]. This is accomplished by assigning each node to a processor. Each processor has message communication links to other processors that transcribe directly from the node links. A message represents the value of a possible transition from one node to another and the quality of the entire path leading up to that transition. To perform the task of reclaiming the generated paths from the processors, each processor must maintain a stack. The stack represents a storage place for logging the history of the activity at the processor. For now, the simplifying assumption will be made that each of the moves made by the robot being simulated will take one unit of time. For example, the time required for the robot to move from one node to an adjacent node, takes one unit of time regardless of the robots previous state. The removal of this assumption will be discussed later (see additional features section 2.5).

Using a synchronous, step-wise process of passing messages from processor to processor, all possible paths that the robot could take through time and space in attaining the desired, static destination location can be considered. The process has two phases and a terminating condition.

The first phase sets up the initial message set. Using the node in space that represents the current location of the robot, a set of messages is created, one message for each communication link associated with the node. Each message has associated with it an energy value that reflects the cost of moving the robot to the space represented by the adjoining node. The particular value of a message's energy is determined by a user-defined evaluation function. The evaluation function considers such things as: the current state of the robot, the cost on the link that the message is to be sent over, the traversability of the node currently being occupied by the robot, etc. The set of messages is then sent to its respective destinations along the communications links of the node.

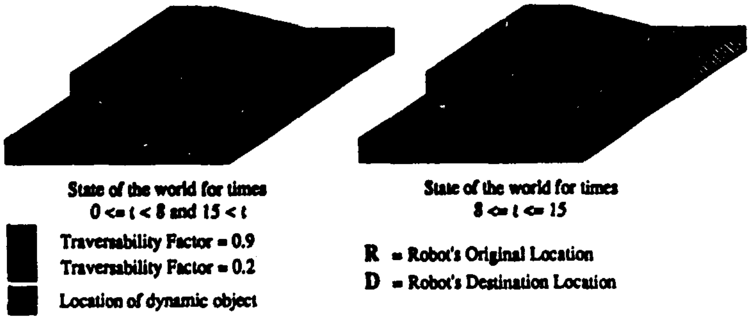
The second phase is the operational phase. It is defined by having each node in space that is not occupied by an object during the current simulation time perform the following process in a synchronous manner: form the base message by setting it to the incoming message with the minimum energy. All other messages can be thrown out because they represent more costly paths that attain the same location in space-time. In a manner similar to that described in the first phase, a new set of messages is created. Each message in the new set is assigned an energy that is a function of the base message and the link over which the message is intended to travel. The base message is then tagged with the time and a pointer indicating the node that created it. The base message is then added to the node's stack. Finally, the node sends the newly created list of messages out along their respective communication link. This process is repeated, until the termination condition is met, each repetition representing a subsequent time unit.

The terminating condition is defined as the state of the system when the energy associated with each of the messages currently being processed in the system is greater than the global bound. The global bound is the minimum energy for all the messages that have reached the destination node (similar to zorch decay in [Charniak86]).

After the ending condition is met, the path through space-time that has the lowest energy associated with it can be retrieved from the destination node. This is done by locating the message on the destination node's stack with the lowest energy value. Once this is done, the path can be obtained by following the pointers back through space (other processors) and time (the stacks associated with the processors) until the robot's original location in space-time is encountered. The stack allows interprocessor communications to be kept to a minimum.

2.4 - A Predictable Example

This section shows how predictable dynamic objects are represented and anticipated. Shown below in Figure 3 is a two-dimensional space made of ramps that are to be navigated by the robot. A dynamic object enters the world at time t = 8 and leaves the world after t=15, indicated by the blackened square in the second of the two Figufes. While the object is present, the node it occupies cannot be traversed by the robot. The nodes over which the robot must travel are, for the most part, easily traversable and thus have a high traversability constant of 0.9. Two of the nodes, however, are made of loose sand and have the low traversability constant of 0.2.



State of the world for times
0 <= t < 8 and 15 < t

State of the world for times
8 <= t <= 15

■ Traversability Factor = 0.9
■ Traversability Factor = 0.2
■ Location of dynamic object
R = Robot's Original Location
D = Robot's Destination Location

Figure 3

The route planning objective is, starting at t=0, to move the robot from its current location (indicated by the R) to the static destination location (indicated by the D) along the space-time path of minimum energy without going over a cliff. By counting the number of node transitions that must be made in traveling from R to D, one can determine that the shortest path the robot could take would be 13 moves long and would use 13 time units. But, when the 8th move is being made, the dynamic object enters the space and disables the indicated node, keeping it from receiving or sending messages. The key is that after seven time units, the message passing process has not propagated to the node with the dynamic object in it. This causes all message activity to be confined to the back portion of the defined space until after t = 15. The blockage is due to the fact that all paths from R to D must pass through the node that is occupied by the object. After t=15 units, the occupied node is freed and resumes processing and passing messages, eventually allowing the system to meet its terminating condition.

A number of points can be made here that are based on the choice of the evaluation function used to determine the message energies. One is that a proper evaluation function will allow the best path to avoid the nodes that are made of sand by giving a high energy value to any message that represents a transition into such a node. A more important point is that the evaluation function determines how and where the robot spends its time while it waits for the object to leave. For example, the robot could remain at R, charging its batteries, wander along the path, or hurry to the object and wait there until the dynamic object goes away. So, depending on the requirements and the situation (e.g., maximize charge time), an evaluation function can be written to determine how and where the robot waits (e.g., add a cost for stopping and starting, or time spent not charging).

2.5 - Additional Features

Some of the system's most powerful features have been omitted thus far for clarity. Among these features are: The ability to describe the destination as a function of both space and time, the ability to consider the openness of a node with respect to its spatial location, and the ability to accurately consider the movement capabilities of the robot using the generated plans.

The ability to describe the destination as a function of both time and space allows the system to solve problems involving alternative planning (e.g. if you can't get to the post office by five, go to the drug store for the stamps) and problems involving coordinating actions with dynamic objects (e.g. jumping on a moving train). This ability is incorporated into the system, by modifying the termination condition of the algorithm to consider a time ordered set of possible destination nodes.

The ability to consider the openness of a node as a spatial relation between it and the nodes surrounding it can be used to generate paths that avoid narrow passages, if possible. Generated plans avoid moving the robot through paths that would become blocked if an unpredictable object were to be encountered during the execution of the plan.

The following gives an example of how an openness function might be defined iteratively for a two-dimensional space. First, assign each node in space a value of one if it is not occupied and a value of zero if it is occupied. Second, have each node send its value along all of its communication links. Third, each node creates a temporary value by summing the values of all incoming information and then integer-dividing them by 1 the first iteration and 2,4,8,16, ... in each subsequent iteration. Lastly, if the temporary value is greater than zero, a new node value is set by multiplying the original node value by the temporary value. Using this scheme, the openness value associated with the nodes will eventually converge to a stable node value pattern. The pattern will be such that the nodes that are in the biggest, most spacious areas will have the highest values, and the nodes in corners or alcoves will have the lowest values. The addition of openness to nodes allows evaluation functions to be written that considers the trade-offs between short path length and increased chance of backtracking due to the chosen route becoming blocked by an unpredictable object.

The ability to effectively represent the time required by a robot to make simulated moves allows plans to be generated that take full advantage of the robot's abilities. For example, moving from rest to another node should take longer than moving from one node to another when the robot is already moving in the desired direction. This is significantly different from the scheme used up to this point, where all moves were considered to take one unit of time. The ability to consider the capabilities of the robot in the generated paths has been incorporated into the model by setting the model to operate in a more asynchronous manner. This asynchrony is accomplished by associating a real-time with each message. The time value of created messages is set by adding to the time in the incoming message the amount of time that is required for the robot to make the move represented by each of the new messages. The ability to effectively predict the performance of the robot is bounded by the precision with which the real-time actions of the robot moving through space-time can be modeled.

3 - Dynamic vs. Unpredictable

Thus far, only the generation of plans that involve predictable objects has been considered. To move autonomous robots in the real world, a route planning system must be able to handle the unpredictability that the real world has to offer as in the case of a robot that must walk across a busy street. The process of incremental route planning has been identified to handle this problem.

Incremental route planning can be viewed as the repeated use of a route planner that executes in a predictable dynamic environment. After each step, the state of the world is tested and updated with any new information, for identification of any unpredictable objects. Incremental route planning is effectively handled by this system because it is structured to operate most efficiently in the incremental form. Unpredictability is handled by the system's ability to rapidly calculate the next best step after every primitive move the robot makes.

By making a simple modification, an incremental version of the algorithm has been constructed from the framework of the previously defined algorithm for predictable domains. The stack is eliminated from each of the processors by making an addition to the messages being passed around the system. The modification involves the addition of an initial direction header. This change is made because all that is needed is the next best move and not the entire path. The headers, of the messages created in phase one of the algorithm, are set to a value representing the link along which that particular message is to be sent. The header, of the messages created during phase two, is copied from the header of the incoming message. To identify the next best move, the terminating condition is modified to keep track of the message representing the current global message energy bound. When the system halts, the header of the global message energy bound indicates the direction of the next best move. This represents a significant simplification of the system, as it eliminates concerns involving the potentially unbounded growth of the node stacks.

4 - Implementation

The algorithm, when fully implemented on an SIMD machine, operates in O(p) time, where p is the length of the longest path through space-time that is bounded by the global message energy bound.

A simulation version of the algorithm, written in NISP [McDermott83], is currently up and running on a VAX 11-785. It functions on the examples given plus others involving unpredictable dynamic environments. The implementation includes software for simulating the SIMD architecture.

5 - Further Research

There are several possible extensions to this model that would increase its representational power. Among the most useful are:

- The granularity of the nodes: It becomes difficult to ensure that there is an adequate path for the robot to traverse, when the granularity of the nodes used to represent space is smaller than the size of the robot or the exact size of the nodes is undetermined.
- Uncertainty in dynamic times: Special concern should be given to route planning where objects entering the space are predictable in their behavior but have uncertain arrival times. For example, the subway train that is running a few minutes behind schedule.
- Achieving maximal efficiency over a set of destinations: This is similar to the traveling salesman problem.
- Modeling unpredictable processes: The power of an incremental route planner can be increased for a particular domain with some model of the typical behavior of the unpredictable objects in that domain. For example, the route planner could give more useful advice to a robot crossing a street if the system had a model of the speed, maneuverability, and direction of travel for the autos traveling the road.
- Representation and coordination of multiple robots moving through space-time: For example, getting Huey, Duey and Luey to meet in the garden on the east end of the space ship at 3pm.

This list represents some of the extensions to our model that are currently under investigation. Extensions into more abstract domains, such as general problem solving using state transition graphs, are also under consideration.

6 - Summary and Conclusions

Planning robot movement in dynamic environments demands that the dynamic aspects of the environment be modeled in at least as much detail as the movements of the robot. We have created a representation system that allows dynamic aspects of the environment and performance aspects of the robot to be easily modeled. It also integrates this model with a high-performance route-planning algorithm. This system has been extended into an incremental route planner which can be used for real-time tactical planning in unpredictable domains. This system has been implemented in an SIMD simulator running on a VAX.

7 - Bibliography

[Brooks82] Brooks, R. A., Solving the find path problem by a good representation of free space, in *Proceedings of AAAI 82*, AAAI, pp. 381-386, 1982.

[Charniak86] Charniak, E., A neat theory of marker passing, in *Proceedings of AAAI 86*, AAAI, pp. 584-588, 1986.

[Chatila85] Chatila, R., Position referencing and consistent world modeling for mobile robots, in *Proceedings of the International Conference on Robotics and Automation*, IEEE, pp. 138-145, 1985.

[Davis84] Davis, E., *Representing and acquiring geographic knowledge*, Technical report 292, Yale University Computer Science Department, 1984.

[Hillis85] Hillis, W. D., *The connection machine*, MIT press, 1985.

[Laumond83] Laumond, J. P., model structuring and concept recognition: Two aspects of learning for a mobile robot, in *Proceedings of the 8th IJCAI*, IJCAI, pp. 839-841, 1983.

[Lozano-Perez83] Lozano-Perez, T., Spatial planning: a configuration space approach, *IEEE transactions on computing*, c'32, pp. 681-698, 1983.

[McDermott84] McDermott, D. V., Davis, E., Planning routes through uncertain territory, *Artificial intelligence*, v22, pp. 107-156, 1984.

[McDermott83] McDermott, D. V., *The nisp manual*, technical report 274, Yale University Computer Science Department, 1983.

[Miller85] Miller, D. P., A spatial representation system for mobile robots, in *Proceedings of the International Conference on Robotics and Automation*, IEEE, pp. 122-127, 1985.

55-63
108
188174

Prediction and Causal Reasoning in Planning

T. Dean and M. Boddy
Brown University
Providence, RI 02912

B 172031 4

Abstract

[illegible]

Nonlinear planners ~~(e.g., NOAH [10])~~ are often touted as having an efficiency advantage over linear planners ~~(e.g., STRIPS [8])~~. The reason usually given is that nonlinear planners, unlike their linear counterparts, are not forced to make arbitrary commitments to the order in which actions are to be performed. This ability to delay commitment enables nonlinear planners to solve certain problems with far less effort than would be required of linear planners. ~~In this paper, we argue~~ that this advantage is bought with a significant reduction in the ability of a nonlinear planner to accurately predict the consequences of actions. Unfortunately, the general problem of predicting the consequences of a partially ordered set of actions is intractable. In gaining the predictive power of linear planners, nonlinear planners sacrifice their efficiency advantage. There are, however, other advantages to nonlinear planning (e.g., the ability to reason about partial orders and incomplete information) that make it well worth the effort needed to extend nonlinear methods. I~~n this paper, we supply~~ a framework for causal inference that supports reasoning about partially ordered events and actions whose effects depend upon the context in which they are executed. As an alternative to a complete but potentially exponential-time algorithm, ~~we~~ provide a provably sound polynomial-time algorithm for predicting the consequences of partially ordered events. If the events turn out to be totally ordered, then the algorithm is complete as well as sound.

~~Keywords: causal reasoning, planning, predicti~~on, nonmonotonic inference, data base management.

supplied [illegible]

1 Introduction

In this paper, we are concerned with the process of incrementally constructing *nonlinear* plans (i.e., plans represented as sets of actions whose order is only partially specified). Nonlinear planning [2] has long been considered to have distinct advantages over linear planning systems such as STRIPS [8] and its descendents. One supposed advantage [10] has to do with the idea that, by delaying commitment to the order in which independent actions are to be performed, a planner can avoid unnecessary backtracking. Linear planners are often forced to make arbitrary commitments regarding the order in which actions are to be carried out. Such arbitrary orderings often fail to lead to a solution and have to be reversed. By ordering only actions known to interact with one another (i.e., actions whose outcomes depend upon the order in which they are executed) the expectation was that nonlinear planners would avoid a lot of unnecessary work.

The problem in getting delayed-commitment planning to work is that it is often difficult to determine if two actions actually are independent. In order to determine whether or not two actions are independent, it is necessary to determine what the effects of those actions are. Unfortunately, in order to determine the effects of an action a it is necessary to determine what is true prior to a being executed, and this in turn requires that we know the effects of those actions that precede a. In general there is no way to determine whether or not two actions are independent without actually considering all of the possible total orderings involving those two actions.

Planning depends upon the ability to predict the consequences of acting. Past planning systems capable of reasoning about partial orders (i.e., nonlinear planners) have either employed weak (and often unsound) methods for performing predictive inference or they have sought to delay prediction until the conditions immediately preceding an action are known with certainty. Delaying predictive inference can serve to avoid inconsistency, but it can also result in extensive backtracking in those very situations that nonlinear planners were designed to handle efficiently.

It our contention that the initial success of Sacerdoti's NOAH [10] program and the promise of NOAH's style of least-commitment planning has caused researchers to ignore important issues in reasoning with incomplete information. The idea of least-commitment planning is not the only reason for building planners capable of reasoning about partially ordered events. Most events will not be under a planner's control and more often than not it will impossible to determine the order of all events with absolute certainty. Planning systems for realistic applications will have to reason about partially ordered events.

In this paper, we consider the problem of reasoning about the effects of partially ordered actions. A theory for reasoning about the effects of actions (or, more generally, the consequences of events) is referred to as a *causal theory*. We will describe a language for constructing causal theories that is capable of representing indirect effects and the effects of actions that depend upon the situation in which they are applied. We will describe a series of algorithms for reasoning about such causal theories. All of these algorithms are polynomial-time, incremental, and insensitive to the order in which facts are added to or deleted from the data base. We show that a particular algorithm is complete for causal theories in which the events are totally ordered, but is potentially inconsistent in cases where the events are not totally ordered. In [6] we show that the general problem of reasoning about conditional actions is *NP-complete*, and, in this paper, we provide a partial decision procedure that, while not complete, is provably sound. What this means for a planner constructing a plan is that the procedure is guaranteed not to misslead the planner into committing to a plan that is provably impossible given what is currently known. If the decision procedure answers yes, then the conditions are guaranteed to hold in every totally ordered extension of the current partial order; if the decision procedure answers no, there is a chance that the conditions hold in every total order, but to determine this with certainty might require an exponential amount of time or space.

2 Temporal Data Base Management

In this section, we consider a particular type of inference system, referred to as a *temporal data base management system* (or TDBMS) [4], that is used to keep track of what is known about the order, duration, and time of occurrence of a set of events and their consequences. The user of a TDBMS is allowed to add two sorts of information: that which is unconditionally believed and that which is believed just in case certain conditions can be shown to hold. The former includes information concerning events that have been observed or are assumed inevitable and information in the form of general rules that are believed to govern the physics of a particular domain.

In specifying conditional beliefs, the user explicitly states what the conditions are, and the TDBMS ensures that those beliefs (and their consequences) are present in the data base just in case the conditions are met. This is achieved through the use of *data dependencies* [7]. In a TDBMS, the primary forms of data dependency (in addition to those common in static situations) are concerned with some fact being true at a point in time or throughout an interval. In addition, there is a nonmonotonic form of temporal data dependency concerned with it being *consistent* to believe that a fact is not true at a point in time or during any part of an interval. These forms of temporal data dependency are handled in the TDBMS using the mechanism of *temporal reason maintenance* [4]. Language constructs are supplied in the TDBMS that allow an application program to query the data base in order to establish certain antecedent conditions (including temporal conditions) and then, on the basis of these conditions, to assert consequent predictions. These predictions remain valid just in case the antecedent conditions continue to hold.

Perhaps one of the most important and most overlooked characteristics of a temporal reasoning system is the ability to handle incomplete information of the sort one invariably encounters in realistic applications. For example, we seldom know the exact duration or time of occurrence of most events. Moreover, for those durations and offsets we do know, they are seldom with respect to a global frame of reference such as a clock or calendar. In the TDBMS, every point is a frame of reference, and it is possible to constrain the distance between any two points simply by specifying bounds on the distance in time separating the two points. By allowing bounds to be both numeric and symbolic, the same framework supports both qualitative (i.e. ordering) and quantitative (distance) relationships.

Another important aspect of reasoning with incomplete information has to do with the default character of temporal inference. In general, it is difficult to predict in advance how long a fact made true will persist. It would be convenient to leave it up to the system to decide how long facts persist based upon the simple default rule [9] that a fact made true continues to be so until something serves to make it false. This is exactly what the TDBMS does. The term *persistence* is used to refer to an interval corresponding to a particular (type of) fact becoming true and remaining so for some length of time. A fact is determined to be true throughout an interval I just in case there is a persistence that begins before the beginning of I and it can't be shown that the persistence ends before the end of I.

The TDBMS permits the specification of partial orders, but it imposes orderings in situations leading to potential incoherency. If the TDBMS encounters two persistence intervals of contradictory types that are not ordered with respect to one another, it prompts the calling program to resolve the possible contradiction by either imposing an order or explicitly introducing a disjunction. By introducing a disjunction, the calling program effectively splits the data base, producing two time lines. The answers returned by queries to the TDBMS indicate the disjuncts that must be true for a query to succeed (i.e., the particular time line that satisfies the query). There are also language constructs that allow a calling program to eliminate disjuncts (and hence time lines) that have been ruled out. Unfortunately, as we will see in Section 5, eliminating explicit contradictions is not sufficient to ensure consistency in a system capable of making conditional predictions from a set of partially ordered events [1]. Before we continue our discussion it will help to introduce some notation.

Relations. Let Π be the set of points corresponding to the begin and end of events in a particular temporal data base. We define a function DIST to denote the best known bounds on the distance in time separating two points. Given $\pi_1, \pi_2 \in \Pi$ such that $\text{DIST}(\pi_1, \pi_2) = \langle low, high \rangle$, we have:

- $\pi_1 \prec \pi_2 \Leftrightarrow low \geq \epsilon$[1] – π_1 precedes π_2
- $\pi_1 \equiv \pi_2 \Leftrightarrow \langle low, high \rangle = \langle 0, 0 \rangle$ – π_1 is coincident with π_2
- $\pi_1 \preceq \pi_2 \Leftrightarrow (\pi_1 \prec \pi_2) \vee (\pi_1 \equiv \pi_2)$ – π_1 precedes or is coincident with π_2
- $\pi_1 \prec_M \pi_2 \Leftrightarrow high \geq \epsilon$ – π_1 possibly precedes π_2
- $\pi_1 \preceq_M \pi_2 \Leftrightarrow high \geq 0$ – π_1 possibly precedes or is coincident with π_2

Tokens. We denote a set of *time tokens* $\text{T} = \{t_0, t_1, \ldots t_n\}$ for referring to intervals of time during which certain events occur or certain facts are known to become true and remain so for some period of time. The latter correspond to what we have be^en calling persistences. For a given token t:

- BEGIN(t), END(t) $\in \Pi$.
- STATUS(t) $\in$ {IN,OUT}, determined by whether the token is warranted (IN) or not (OUT) by the current premises and causal theory.
- TYPE(t) $= P$ where P is an atomic predicate calculus formula with no variables
- DURATION(t) = DIST(BEGIN(t), END(t))

Types. As defined above, the type of an individual token is an atomic formula with no variables (e.g., (on block14 table42)). In general, any atomic formula, including those containing variables, can be used to specify a type. In describing the user interface, universally quantified variables are notated ?variable-name, the scope of the variable being the entire formula in which it is contained (e.g., (on ?x ?y)). In describing the behavior of the inference system, we will use variables of the form t_P to quantify over tokens of type P (e.g., $\forall t_P \in \text{T}\ \text{TYPE}(t_P) = P$).

3 Reasoning about Causality

In the TDBMS, a causal theory is simply a collection of rules, called *projection rules*, that are used to specify the behavior of processes. In the following rule, $P_1 \ldots P_n$, $Q_1 \ldots Q_m$, E, and R designate types, and *delay* and *duration* designate constraints (e.g., $\langle \epsilon, \infty \rangle$). In:

$$\begin{array}{l} \text{(project (and } P_1 \ldots P_n \\ \qquad\qquad \text{(M (not (and } Q_1 \ldots Q_m\text{))))} \\ \qquad E \;\; delay \;\; R \;\; duration\text{)} \end{array} \tag{1}$$

$P_1 \ldots P_n$ and $Q_1 \ldots Q_m$ are referred to as *antecedent conditions*, E is the type of the *triggering event*, and R refers to the type of the *consequent prediction*. The above projection rule states that, if an event

[1] The symbol ϵ is meant to denote an infinitesimal: a number greater than 0 and smaller than any positive number.

R1: (project (and $P_1 \ldots P_n$
(M (not (and $Q_1 \ldots Q_m$))))
E R)

R2: (project (and $P_1 \ldots P_n$)
E R)

R3: (disable (and $Q_1 \ldots Q_m$)
(ab R2))

R4: (disable (and $R_1 \ldots R_o$)
(ab R3))

Figure 1: Hierarchically arranged projection and disabling rules

of type E occurs corresponding to the token t_E and $P_1 \ldots P_n$ are believed to be true at the outset[2] of t_E and it is consistent to believe that the conjunction of $Q_1 \ldots Q_m$ is not true at the outset of t_E, then, after an interval of time following the end of t_E determined by *delay*, R will become true and remain so for a period of time constrained by *duration* (if *delay* and *duration* are not specified, they default to $\langle 0, 0 \rangle$ and $\langle \epsilon, \infty \rangle$, respectively). In the following, we will be considering a restricted form of causal theory, called a *type 1* theory, such that the *delay* always specifies a positive offset (causes always precede their effects).

We also allow the user to specify rules that serve to *disable* other rules [11]. Figure 1 shows a standard projection rule R1 and a pair of projection and disabling rules R2 and R3 that replace R1. The rule R3 is further conditioned by the rule R4. Assuming just the rules R2, R3, and R4, any application of R2 with respect to a particular token t of type E is said to be abnormal with regard to t just in case $Q_1 \ldots Q_m$ hold at the outset of t and it is consistent to believe that R3 is not abnormal with regard to t. The nonmonotonic behavior of type 1 causal theories is specified entirely in terms of disabling rules and the default rule of persistence (see Section 2). In addition to their usefulness for handling various forms of incomplete information, disabling rules make it possible to reason about the consequences of simultaneous actions. The reader interested in a more detailed treatment of causal theories may refer to one of [4], [5], or [11]. In parts of this paper, we will ignore disabling rules and speak of causal theories consisting solely of *simple projection rules* of the form (project (and $P_1 \ldots P_n$) E R).

One of the most problematic aspects of designing a temporal inference system involves defining precisely what it *means* for a fact to be true at a point or throughout an interval (i.e., the conditions under which a query of the form $TT(P, \pi_1, \pi_2)$ will succeed). As a first approximation, we offer the following definition, which we will refer to as *weak true throughout*:

[2] An alternative formulation described in [3] states that the antecedent conditions of a projection rule must be true *throughout* the trigger event rather than true just at the outset. Both formulations are supported in the TDBMS, though we will only be discussing the true-at-the-outset formulation in this paper.

$$\begin{array}{l}
\forall t_E \;\in\; \mathrm{T} \\
\quad ((\mathrm{STATUS}(t_E) = \mathrm{IN}) \wedge \\
\quad\; (\exists t_{P_1} \ldots t_{P_n} \;\in\; \mathrm{T} \\
\quad\;\; (\forall\, 1 \leq i \leq n \; (\mathrm{STATUS}(t_{P_i}) = \mathrm{IN}) \wedge \\
\qquad\qquad\qquad (\mathrm{BEGIN}(t_{P_i}) \preceq \mathrm{BEGIN}(t_E)) \wedge \\
\qquad\qquad\qquad (\mathrm{BEGIN}(t_E) \preceq_M \mathrm{END}(t_{P_i}))\;)) \wedge \\
\quad\;\; \neg(\exists t_{Q_1} \ldots t_{Q_m} \;\in\; \mathrm{T} \\
\quad\;\;\; (\forall\, 1 \leq j \leq m \; (\mathrm{STATUS}(t_{Q_j}) = \mathrm{IN}) \wedge \\
\qquad\qquad\qquad\; (\mathrm{BEGIN}(t_{Q_j}) \preceq \mathrm{BEGIN}(t_E)) \wedge \\
\qquad\qquad\qquad\; (\mathrm{BEGIN}(t_E) \preceq_M \mathrm{END}(t_{Q_j}))\;))) \\
\quad\;\; \Rightarrow\; \exists t_R \;\in \mathrm{T} \\
\qquad\quad (\mathrm{STATUS}(t_R) = \mathrm{IN}) \;\wedge \\
\qquad\quad (\mathrm{DIST}(\mathrm{END}(t_E), \mathrm{BEGIN}(t_R)) \subseteq \mathit{delay}) \;\wedge \\
\qquad\quad (\mathrm{DIST}(\mathrm{BEGIN}(t_R), \mathrm{END}(t_R)) \subseteq \mathit{duration})
\end{array}$$

Figure 2: Weak projection

$$\begin{array}{l}
\forall \pi_1 \pi_2 \;\in\; \Pi \\
\quad \exists t_P \;\in\; \mathrm{T} \\
\qquad (\mathrm{STATUS}(t_P) = \mathrm{IN}) \wedge \\
\qquad (\mathrm{BEGIN}(t_P) \;\preceq\; \pi_1) \wedge \\
\qquad (\pi_2 \;\preceq_M\; \mathrm{END}(t_P)) \\
\quad \Leftrightarrow\; TT(P, \pi_1, \pi_2)
\end{array} \tag{2}$$

In order to specify the behavior of a temporal inference system such as the TDBMS, we also need to define the criterion for inferring consequent effects from antecedent conditions via causal rules. Our first such criterion will be referred to as *weak projection* (Figure 2) and is defined with respect to the general form of a projection rule (1). Weak true throughout and weak projection correspond to the assumption that "what you don't know won't hurt you." Only those events that can be shown to be ordered with respect to a particular point will have any effect at that point. As we will see in Section 5, there are some problems with this formulation.

4 Transactions on the Data Base: the User Interface

Every inference system provides some means for the user to specify rules (referred to collectively as a *causal theory*) for inferring additional consequences of the data (referred to here as a set of *basic facts*). An application program interacts with an inference system by adding and removing items from the set of basic facts, which in the TDBMS corresponds to a set of tokens and a set of constraints. The state of a temporal data base is completely defined by a temporal constraint graph (TCG), consisting of the begin and end points of tokens and constraints between them, and a causal dependency graph (CDG), consisting of dependency structures corresponding to the application of causal rules in deriving new tokens. Each transaction performed on the temporal data base results in changes to these two data structures. The TDBMS is responsible for maintaining the temporal data base so that it captures exactly those consequences that follow from the causal theory and the current set of basic facts.

Generally, the causal theory remains fixed for a particular application, and interaction with the TDBMS consists of a series of transactions and queries. A transaction consists of either adding or removing some token or constraint from the set of basic facts. A query consists of a predicate calculus formula corresponding to a question of the form "Could some fact P be true over a particular interval I?" An affirmative answer returned by the TDBMS in response to such a query will include a set of assumptions necessary for concluding that the fact is indeed true. Any assertions made on the basis of the answer to such a query are made to depend upon these assumptions. There is also a mechanism that enables the TDBMS to detect and, with the assistance of the calling program, resolve inconsistencies in the set of constraints.

5 Completeness and Consistency

The primary source of ambiguity in the TDBMS arises from the fact that the set of constraints seldom determines a total ordering of the tokens in T. Given that most inferences depend only upon what is true during intervals defined by points corresponding to the begin and end of tokens in T, all that we are really interested in is what facts are true in what intervals in the different total orderings of time points consistent with the initial set of constraints. For each total ordering we can identify a unique set of tokens that intuitively should be IN given a particular causal theory.

As far as we are concerned, an *inference procedure* is fully specified by a criterion for inferring consequent effects from antecedent causes via causal rules (e.g. weak projection), a method for actually applying that criterion (an update algorithm), and a criterion for determining if a fact is true throughout some interval (e.g., weak true throughout). We will say that a particular inference procedure is *complete* for a class of causal theories, if for any set of basic facts and causal theory in the class, the statements of the form $TT(P, \pi_1, \pi_2)$ warranted by the inference procedure are exactly those that are true in all total orders consistent with the constraints in the TCG. Similarly, we will say that an inference procedure is *sound* for a class of causal theories, if for any set of basic facts and causal theory in the class, each statement $TT(P, \pi_1, \pi_2)$ warranted by the inference procedure is true for any total order consistent with the set of constraints. Given the preceding definitions, it is easy to show that the TDBMS is complete and sound for type 1 causal theories, assuming that the tokens in T are totally ordered [6].

In situations where the set of basic facts does not determine a total order, it's easy to show that the TDBMS can end up in a state with IN tokens that allow one to conclude statements of the form $TT(P, \pi_1, \pi_2)$ that are not true in *any* totally ordered extension. One thing we might do to improve the chances of the TDBMS warranting only valid statements of the form $TT(P, \pi_1, \pi_2)$ is strengthen the criterion for belief in a given token. We can determine a class of tokens that are said to be *strongly protected*, using the axioms in Figure 3. In these axioms and the rest of the paper, we use T_B to denote the tokens in the set of basic facts. If the set of constraints determines a total ordering, then the set of strongly protected tokens is identical to the set of tokens that are IN, but generally the former is a subset of the latter. Using this notion of strongly protected, we can define a stronger true throughout criterion that we will refer to as *strong true throughout*:

$$
\begin{array}{l}
\forall t \;\in\; \mathrm{T_B} \\
\quad \mathit{STRONGLY\text{-}PROTECTED}(t)
\end{array}
$$

$$
\begin{array}{l}
\forall t_B \;\in\; \mathrm{T} \\
\quad (\mathit{STRONGLY\text{-}PROTECTED}(t_B) \wedge \\
\quad\; (\exists t_{P_1} \ldots t_{P_n} \;\in\; \mathrm{T} \\
\quad\quad (\forall\; 1 \le i \le n\; \mathit{STRONGLY\text{-}PROTECTED}(t_{P_i}) \wedge \\
\qquad\qquad\qquad (\mathrm{BEGIN}(t_{P_i}) \preceq \mathrm{BEGIN}(t_B)) \wedge \\
\qquad\qquad\qquad (\mathrm{BEGIN}(t_B) \preceq_M \mathrm{END}(t_{P_i})) \wedge \\
\qquad\qquad\qquad (\forall t_{\neg P_i} \;\in\; \mathrm{T} \\
\qquad\qquad\qquad\qquad (\mathrm{STATUS}(t_{\neg P_i}) = \mathrm{OUT}) \vee \\
\qquad\qquad\qquad\qquad (\mathrm{BEGIN}(t_{\neg P_i}) \prec \mathrm{BEGIN}(t_{P_i})) \vee \\
\qquad\qquad\qquad\qquad (\mathrm{END}(t_B) \prec \mathrm{BEGIN}(t_{\neg P_i}))\;)))) \\
\quad \Rightarrow\; \exists t_R \;\in\; \mathrm{T} \\
\qquad \mathit{STRONGLY\text{-}PROTECTED}(t_R) \;\wedge \\
\qquad (\mathrm{DIST}(\mathrm{END}(t_B), \mathrm{BEGIN}(t_R)) \subseteq \mathit{delay}) \wedge \\
\qquad (\mathrm{DIST}(\mathrm{BEGIN}(t_R), \mathrm{END}(t_R)) \subseteq \mathit{duration})
\end{array}
$$

Figure 3: Strong protection defined for simple projection rules

$$
\begin{array}{l}
\forall t_B \;\in\; \mathrm{T} \\
\quad ((\mathrm{STATUS}(t_B) = \mathrm{IN}) \wedge \\
\quad\; (\exists t_{P_1} \ldots t_{P_n} \;\in\; \mathrm{T} \\
\quad\quad (\forall\; 1 \le i \le n\; (\mathrm{STATUS}(t_{P_i}) = \mathrm{IN}) \wedge \\
\qquad\qquad\qquad (\mathrm{BEGIN}(t_{P_i}) \preceq_M \mathrm{BEGIN}(t_B)) \wedge \\
\qquad\qquad\qquad (\mathrm{BEGIN}(t_B) \preceq_M \mathrm{END}(t_{P_i})) \wedge \\
\qquad\qquad\qquad (\forall t_{\neg P_i} \;\in\; \mathrm{T} \\
\qquad\qquad\qquad\qquad \neg\mathit{STRONGLY\text{-}PROTECTED}(t_{\neg P_i}) \vee \\
\qquad\qquad\qquad\qquad (\mathrm{BEGIN}(t_{\neg P_i}) \prec_M \mathrm{BEGIN}(t_{P_i})) \vee \\
\qquad\qquad\qquad\qquad (\mathrm{BEGIN}(t_B) \prec_M \mathrm{BEGIN}(t_{\neg P_i}))\;)))) \\
\quad \Rightarrow\; \exists t_R \;\in\; \mathrm{T} \\
\qquad (\mathrm{STATUS}(t_R) = \mathrm{IN}) \;\wedge \\
\qquad (\mathrm{DIST}(\mathrm{END}(t_B), \mathrm{BEGIN}(t_R)) \subseteq \mathit{delay}) \wedge \\
\qquad (\mathrm{DIST}(\mathrm{BEGIN}(t_R), \mathrm{END}(t_R)) \subseteq \mathit{duration})
\end{array}
$$

Figure 4: Improbably weak projection defined for simple projection rules

$$
\begin{array}{l}
\forall \pi_1 \pi_2 \;\in\; \Pi \\
\quad \exists t_P \;\in\; \mathrm{T} \\
\qquad \mathit{STRONGLY\text{-}PROTECTED}(t_P) \wedge \\
\qquad (\mathrm{BEGIN}(t_P) \;\preceq\; \pi_1) \wedge \\
\qquad (\pi_2 \;\preceq_M\; \mathrm{END}(t_P)) \\
\quad \Leftrightarrow\; \mathit{TT}(P, \pi_1, \pi_2)
\end{array}
\tag{3}
$$

As it turns out, weak projection and strong true throughout still do not constitute a sound inference procedure. We can show that, even when we restrict ourselves to strongly protected tokens, most interesting decision problems are intractable. In fact, we can prove that the problem of determining if $TT(P, \pi_1, \pi_2)$ is

true for a type 1 causal theory, with or without disabling rules, is *NP-complete* [6]. Although completeness is computationally infeasible, it is possible to devise an inference procedure that is both sound and capable of performing useful prediction. First, we provide a criterion for generating consequent predictions that takes into account every consequence that might be true in any total order, called *improbably weak projection* (Figure 4). Second, we provide a criterion for true throughout that succeeds only if the corresponding formula will be true in all total orders consistent with the current set of constraints. We define *improbably strong true throughout*:

$$
\begin{array}{l}
\forall \pi_1 \pi_2 \in \Pi \\
\quad \exists\, t_P \in \mathsf{T} \\
\qquad \mathit{STRONGLY\text{-}PROTECTED}(t_P) \wedge \\
\qquad (\mathrm{BEGIN}(t_P) \preceq \pi_1) \wedge \\
\qquad (\forall t_{\neg P} \in \mathsf{T} \\
\qquad\quad (\mathrm{STATUS}(t_{\neg P}) = \mathtt{OUT}) \vee \\
\qquad\quad (\mathrm{BEGIN}(t_{\neg P}) \prec \mathrm{BEGIN}(t_P)) \vee \\
\qquad\quad (\pi_2 \prec \mathrm{BEGIN}(t_{\neg P}))\ ) \\
\quad \Leftrightarrow\ \mathit{TT}(P, \pi_1, \pi_2)
\end{array}
\tag{4}
$$

There is a simple decision procedure for generating all consequences and computing the set of strongly protected tokens. Let $\mathsf{T}_1 = \mathsf{T}_{\mathsf{B}}$, and initially assume that no tokens are strongly protected. Let $i = 1$. To compute the consequences of T_i, set $\mathsf{T}_{i+1} = \mathsf{T}_i$, compute the consequent tokens of each token in T_i using the criterion of improbably weak projection, and add any new tokens to T_{i+1}. Continue to compute new consequent tokens in this manner incrementing i as needed until $\mathsf{T}_i = \mathsf{T}_{i+1}$. Set $\mathsf{T} = \mathsf{T}_i$. At this point, perform a sweep forward in time (relative to the current partial order) determining for each token in T whether or not it is strongly protected and the status, IN or OUT, of each its consequents. In [6], we prove that this decision procedure is sound for a partially ordered set of tokens, and sound and complete for a totally ordered set.

In the same paper, we give two incremental update algorithms with polynomial-time worst case behavior, one for weak projection and weak true throughout, and one for improbably weak projection and improbably strong true throughout. The latter algorithm does not model the decision procedure given above—there is a more complicated procedure with the same behavior that is more efficient by a constant factor. We prove that these algorithms support exactly the conclusions licensed by their respective inference methods. Proving that the algorithms terminate is in one sense impossible. Using the TDBMS and a type 1 causal theory with arithmetic functions (e.g., `(project (contents ?register ?n) (increment ?register) (contents ?register (+ 1 ?n)))`), we can easily simulate a Turing machine. There are a number of methods for either restricting what the user can encode in a causal theory or limiting the scope of the inferences computed by the TDBMS in such a way that we can guarantee that the update terminates. By limiting the scope of the inferences computed by the TDBMS, we potentially sacrifice completeness, but we have shown that to ensure completeness may require an exponential amount for time for other reasons.

6 Conclusions

This paper is concerned with computational approaches to reasoning about time and causality, particularly in domains involving partial orders and incomplete information. We have described a class of causal theories

involving a carefully restricted use of nonmonotonic inference, capable of representing conditional effects and the effects of simultaneous actions. We showed in [6] that the decision problem for nontrivial inference systems involving conditional effects and partially ordered events is *NP-complete*. As an alternative to a complete but potentially exponential-time inference procedure, we have described a decision procedure, for which there is an incremental polynomial-time algorithm, that generates a useful subset of those inferences that will be true in all total orders consistent with the initially specified partial order. The decision procedure is provably sound and the resulting conclusions are guaranteed consistent if the underlying causal theory is consistent. If the events turn out to be totally ordered, the procedure is complete as well as sound.

References

1. Chapman, David, *Planning for Conjunctive Goals,* Technical Report AI-TR-802, MIT AI Laboratory, 1985.
2. Charniak, Eugene and McDermott, Drew V., *Introduction to Artificial Intelligence* (Addison-Wesley Publishing Co., 1985).
3. Dean, Thomas, *Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving,* Technical Report 433, Yale University Computer Science Department, 1985.
4. Dean, Thomas, and McDermott, Drew V., Temporal Data Base Management, to appear in *Artificial Intelligence.*
5. Dean, Thomas, An Approach to Reasoning About the Effects of Actions for Automated Planning Systems, to appear in the 1987 volume of the *Annals of Operations Research* entitled "Approaches to Intelligent Decision Support".
6. Dean, Thomas, and Boddy, Mark, *Incremental Causal Reasoning,* Technical Report CS-87-1, Brown University Department of Computer Science, 1987.
7. Doyle, Jon, A truth maintenance system, *Artificial Intelligence* 12 (1979) 231–272.
8. Fikes, Richard and Nilsson, Nils J., STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence* 2 (1971) 189–208.
9. Reiter, Raymond, A Logic for Default Reasoning, *Artificial Intelligence* 13 (1980).
10. Sacerdoti, Earl, *A Structure for Plans and Behavior* (American Elsevier Publishing Company, Inc., 1977).
11. Shoham, Yoav, *Chronological Ignorance: Time, Knowledge, Nonmonotonicity and Causation,* in Georgeff, Michael P. and Lansky, Amy L. (Eds.), *The 1986 Workshop on Reasoning about Actions and Plans,* (Morgan-Kaufman 1987).

56-63
88
188175

A description

The Generic Task Toolset: High Level Languages for the Construction of Planning and Problem Solving Systems

B. Chandrasekaran, J. Josephson, and D. Herman
Ohio State University
Columbus, OH 43210

is advanced

is criticized, and

ΦM 593268

Abstract

We ~~begin by criticising~~ the current generation of languages for the construction of knowledge-based systems as being at too low a level of abstraction, ~~and argue for~~ the need for higher level languages for building problem solving systems. We ~~introduce our~~ A notion of generic information processing tasks in knowledge-based problem solving, and ~~describe~~ a toolset which can be used to build expert systems in a way that enhances intelligibility and productivity in knowledge acquisition and system construction. ~~We illustrate~~ the power of these ideas by paying special attention to a high level language called DSPL. and ~~describe~~ how it was used in the construction of a system called MPA, which assists with planning in the domain of offensive counter air missions.

is introduced.

is presented

Scope and Intent

This paper is intended to be an introduction to the generic tasks approach to analyzing knowledge systems. descriptions of some of the particular generic tasks that have been identified, and a description of the software tools that have been build as a result. The approach is illustrated by discussing a particular mission-planning system, MPA, which was built using the DSPL language (or tool). The intent of the paper is primarily tutorial, much of the material is summary or repetition of material already presented elsewhere.[1, 2]

is illustrated

Level of Abstraction Problem in Characterizing Knowledge-Based Systems

Much of AI can be said to be a search for the "Holy Grail" of a level of abstraction at which intelligence behavior qua intelligent behavior emerges. Above this level. presumably, are particular pieces of knowledge of the task domain, and below this level are specific details of how the intelligent level is implemented. For example, in a rule-based system such as Mycin, everyone would agree that rules of medicine are not per se matters of AI research. That is, the content of the knowledge base itself is made up of domain particulars. At the same time the language in which the rule system is written, e.g., Lisp, is typically thought of as an implementation-level detail, of no particular interest as AI.

The AI interest of a given expert system is often a matter of the level at which it is viewed. Clearly, taking Mycin as an example, all the following points of view are correct: (i) it suggests therapy for certain kinds of infectious diseases, (ii) it is an embodiment of a diagnostic and therapeutic strategy, and (iii) it is a decision-maker which uses backward chaining to navigate a knowledge base of rules, connecting pieces of evidence to conclusions, in order to arrive at a reliable conclusion from a given set of data. Point of view (i) is of limited interest to AI, and point of view (iii) has been the level at which the system as an AI system has been generally presented; this is the level at which the claim for generality of the underlying approach is made. All the excitement surrounding the "knowledge base/inference engine" paradigm, and the idea that knowledge acquisition and explanation can be keyed to phenomena at the rule level, emphasizes that the level corresponding (iii) has been the level of abstraction at which AI interest has been expressed, and claims of progress have been made.

What of the level corresponding to (ii)? We have for several years at Ohio State worked on the hypothesis that this is indeed an important level of abstraction for AI; and that knowledge representation, control regimes for problems solving, knowledge acquisition, and explanation all can be significantly advanced by looking at phenomena at this level. B. Chandrasekaran has put forth this view in[1, 3, 4]. In this paper we give a critical analysis of an important part of AI, viz., knowledge-based reasoning, and propose that a point of view based on a particular level of abstraction corresponding to generic information processing tasks has a number of advantages both for clarity of analysis and for system design. This is not pure theoretical speculation; researchers in our laboratory have built many systems and tools based on this framework. We wish to point to this level of abstraction as a productive level for concentration, and to indicate the conceptual advantages of it. Knowledge acquisition, system design, control of problem solving and generation of explanation all are facilitated *at the same time*, indicating that there is a naturalness to looking at phenomena at this level.

[1] Research supported by Air Force Office of Scientific Research grant 82 0255, National Science Foundation grant MCS-8305032, and Defense Advanced Research Projects Agency, RADC Contract F30602 85 C 0010. Toolset development is also supported by grants of equipment from Xerox Corporation, IBM, and Texas Instruments

Intuitively one thinks that there must be some commonality of reasoning processes that characterize "diagnosis" as a generic activity, even across domains as different as medicine and mechanical systems. There *should* be *control strategies* and ways of using knowledge that are common to diagnostic reasoning as such, or at least typical of diagnostic reasoning. Similarly there should be common types of knowledge structures and control strategies for, say, design as a kind of reasoning activity. Further, we expect that the structures and control regimes for diagnostic reasoning will be generally different from those for design reasoning. However, when one looks at the formalisms (or equivalently the languages or shells), that are commonly used in expert system design, the knowledge representation and control regimes do not typically capture these distinctions. For example, in diagnostic reasoning, one might generically wish to speak in terms of malfunction hierarchies, rule-out strategies, setting up a differential, etc., while for design, the generic terms might be device/component hierarchies, design plans, ordering of design subtasks, etc. Ideally one would like to represent diagnostic knowledge in a domain by using the vocabulary that is *appropriate for the task*. But typically the languages in which the expert systems have been implemented have sought uniformity across tasks, and thus have lost clarity of representation at the task level. The computational universality of representation languages such as Emycin or OPS5 -- i.e., the fact that any computer program can in principle be written in these languages -- often confuses the issue, since after the system is finally built it is often unclear which portions of the system represent domain expertise, and which are programming devices. In addition, the control regimes that these languages come with (such as forward or backward chaining) do not explicitly indicate the real control structure of the system at the task level. For example, the fact that R1 [5] performs a linear sequence of subtasks -- an atypically simple strategy for design problem solving -- is not explicitly encoded; the system designer so to speak "encrypted" this control in the pattern-matching control of OPS5.

These comments need not be restricted to the rule-based framework. One could represent knowledge as sentences in a logical calculus and use logical inference mechanisms to solve problems; or one could represent it in a frame hierarchy with procedural attachments in the slots. (It is a *relatively* straightforward thing, e.g, to rewrite MYCIN [6] in this manner, see[7].) In the former, the control issues would deal with choice of predicates and clauses, and in the latter, they will be at the level of which links to pursue for inheritance, etc. None of these have any natural connection with the control issues specific to the task.

The other side of the coin, so to speak, regarding control is the following: because of the relatively low level of abstraction relative to the information processing task, there are control issues that are artifacts of the representation, but often in our opinion misinterpreted as issues at the "knowledge-level." E.g., rule-based approaches often concern themselves with conflict resolution strategies. If the knowledge were viewed at the level of abstraction appropriate to the task, often there will be *organizational* elements which would result in only a small set of highly relevant pieces of knowledge or rules to being brought up for consideration, without any conflict resolution strategies being needed. Of course, these organizational constructs could be "programmed" in the rule language, but because of the status assigned to the rules and their control as knowledge-level phenomena (as opposed to the implementation level phenomena, which they often are), knowledge acquisition is often directed towards (typically syntactic) strategies for conflict resolution, whereas the really operational expert knowledge is at the organizational level.

This level problem with control structures is mirrored in the relative poverty of knowledge-level primitives for representation. For example the epistemology of rule systems is exhausted by data patterns (antecedents or subgoals) and partial decisions (consequents or goals), that of logic is similarly exhausted by predicates, functions, inference rules, and related primitives. If one wishes to talk about types of goals or predicates in such a way that control behavior can be indexed over this topology, such a behavior can often be programmed into these systems, but no explicit rendering of them is possible. E.g., Clancey [8] found in his work using *Mycin to teach students* that for explanation he needed to attach to each rule in the Mycin knowledge base encodings of types of goals so that explanation of its behavior can be couched in terms of this encoding, rather than only in terms of "Because .. · was a subgoal of <..>." This is not to argue that rule representations and backward or forward chaining controls are not "natural" for some situations. If all that a problem solver has for knowledge in a domain is in the form of a large collection of unorganized associative patterns, then data-directed or goal-directed associations may be the best that the agent can do. But that is *precisely* the occasion for weak methods such as hypothesize and match (of which the above associations are variants), and, typically, successful solutions cannot be expected in complex problems without combinatorial searches. Generally, however, expertise can be expected to consists of much more organized collections of knowledge, with control behavior *indexed by the kinds of organizations and forms* of knowledge in them.

Thus, there is a need for understanding the generic information processing tasks that underlie knowledge-based reasoning. Knowledge ought to be directly encoded at the appropriate level by using primitives that naturally describe the domain knowledge for a given generic task. Problem solving behavior for the task ought to be controlled by regimes that are appropriate for the task. If done correctly, this would simultaneously facilitate knowledge representation, problem solving, and explanation.

At this point it will be useful to make further distinctions. Typically many tasks that we intuitively think of as generic tasks are really **complex generic tasks**. I. e., they are further decomposable into components which are more elementary in the sense that each of them has a more homogeneous control regime and knowledge structures. For example, what one thinks of as the diagnostic task, while it may be generic in the sense that the task may be quite similar across domains, it is not a unitary task structure. Diagnosis may involve classificatory reasoning at a certain point, reasoning from one datum to another datum at another point, and abductive assembly of multiple diagnostic hypotheses at another point. Hierarchical classification has a different form of knowledge and control behavior from those for data-to-data reasoning, which in turn is dissimilar in these dimensions from assembling hypotheses. Our research focuses on tasks at both these levels, but the latter are viewed as somewhat "atomic" tasks into which more complex, but still generic, tasks such as diagnosis and design can often be decomposed.

To summarize the view presented so far: There is a need for understanding the generic information processing tasks that underlie knowledge-based reasoning. Knowledge ought to be directly encoded at the appropriate level by using primitives that naturally describe the domain knowledge for a given generic task. Problem solving behavior for the task ought to be controlled by regimes that are appropriate for the task. If done correctly, this would simultaneously facilitate knowledge representation, problem solving, and explanation.

Over the years, we have identified, and built systems using, six such generic tasks. Our work on MDX[9, 10], e.g., identified **hierarchical classification**, **knowledge-directed information passing**, and **hypothesis matching** as three generic tasks, and showed how certain classes of diagnostic problems can be implemented as an integration of these generic tasks. Since then we have identified several others: **object synthesis by plan selection and refinement**[11], **state abstraction**[4], and **abductive assembly of hypotheses**[12]. There is no claim that these six are exhaustive; in fact, our ongoing research objective is to identify other useful generic tasks and understand their knowledge representations and strategies for control of problem solving.

Some Generic Tasks

Characterization of Generic Tasks

Each generic task is characterized by: a **task specification** in the form of generic types of **input** and **output** information; specific **forms of knowledge** needed for the task, and specific **organization of knowledge** particular to the task; a family of **control regimes** that are appropriate for the task.

A task-specific control regime comes with certain characteristic types of **strategic goals**. These goal types will play a role in providing explanations of its problem solving behavior.

When a complex task is decomposed into a set of generic tasks, it will in general be necessary to provide for communication between the different structures specializing in these different types of problem solving. Also there is not necessarily a unique decomposition. Depending upon the availability of particular pieces of knowledge, different architectures of generic tasks will typically be possible for a given complex task.

We will now give brief characterizations of the generic tasks that we have identified.

Taxonomic Classification

Task specification: Classify a (possibly complex) description of a situation as an element, as specific as possible, in a *classification hierarchy*. E.g. classify a medical case description as an element of a disease hierarchy.

Forms of knowledge: one main form is <partial situation description> ---> evidence/belief about confirmation or disconfirmation of classificatory hypotheses. E.g., in medicine, a piece of classificatory knowledge may be: certain pattern in X-ray & bilirubin in blood ---> high evidence for cholestasis.

Organization of knowledge: knowledge of the form above *distributed* among concepts in a classificatory concept hierarchy. Each conceptual "specialist" ideally contains knowledge that helps it determine whether it (the concept it stands for) can be *established* or *rejected*.

Control Regime: Problem solving is top down, each concept when called upon tries to establish itself. If it succeeds, it lists the reasons for its success, and calls its successors, which repeat the process. If a specialist fails in its attempt to establish itself, it rejects itself, and all its successors are also automatically rejected. This control strategy can be called ***Establish-Refine***, and results in a specific classification of the case. (The account is a simplified one. The reader is referred to[10] for details and elaborations.)

Goal types: E.g., Establish <concept>, Refine (subclassify) <concept>

Example Use: Medical diagnosis can often be viewed as a classification problem. In planning, it is often useful to classify a situation as of a certain type, which then might suggest an appropriate plan.

Object Synthesis by Plan Selection and Refinement

Task Specification: Design an object satisfying specifications (object in an abstract sense: they can be plans, programs, etc.).

Forms of knowledge: Object structure is known at some level of abstraction, and pre-compiled plans are available which can make choices of components, and have lists of concepts to call upon for *refining* the design at that level of abstraction.

Organization of Knowledge: Concepts corresponding to components organized in a hierarchy mirroring the object structure. Each concept has plans which can be used to make commitments for various "dimensions" of the component.

Control Regime: Top down in general. The following is done recursively until a complete design is worked out: A specialist corresponding to a component of the object is called, the specialist chooses a plan based on the specifications and problem state, instantiates and executes the plan which suggests further specialists to call to set details of the subcomponents. Plan failures are passed up until appropriate changes are made by higher level specialists.

Goal Types: E.g., Choose plan, execute <plan element>, refine <plan>, redesign (modify) <partial design> to respond to failure of <subplan>, select alternative plan, etc.

Example: Expert design tasks, routine synthesis of plans of action.

We will characterize the other generic tasks more succinctly. The reader is referred to[1] for more details.

Knowledge-Directed Information Passing

Task: It is desired to obtain attributes of some datum. by deriving from some conceptually related datum. Some forms of knowledge are: <attribute> of <datum> is inherited from <attribute>~~in~~of parent of <datum>, <attribute> of <datum> is related as <relation> to <attribute> of <concept>. Organization: concepts are organized as a frame hierarchy, with IS-A and PART-OF links. Each frame is a specialist in knowledge-directed data inference for the concept. This is basically a hierarchical information-passing control regime.

Example uses: knowledge-based data retrieval tasks in wide variety of situations, as an intelligent data base in support of problem solvers of other types.

Abductive Assembly of Explanatory Hypotheses

Task Specification: Given a situation (described by a set of data items) to be explained by the best explanatory account, and given a number of hypotheses, each associated with a degree of belief, and each of which offers to explain a portion of the data (possibly overlapping with data to be accounted for by other hypotheses), construct the best composite explanatory hypothesis. Some forms of knowledge are: causal or other relations (e.g. special case of, incompatibility, suggestiveness) between the hypotheses, relative significance of data items, and ways to group data items to be explained. Organization: one main, or a hierarchical community of active abducers, each specializing in explaining a certain portion of the data by composing and criticizing hypotheses. *Control Regime:* (See [13] for a fuller discussion.) A specialized means-ends regime is in control, driven by the goals of explaining all the significant findings, with an economical hypothesis. which is consistent, and has been criticized for certain strengths and weaknesses. Some goal types are: account-for <datum>; check-superfluousness-of <hypothesis>. choose the most significant unexplained finding. The Internist system[11] and the Dendral system[15] perform abductive assembly as part of their problem solving.

State Abstraction

Task Specification: Given a change in some state of a system, provide an account of the changes that can be expected in the functioning of the system. (Useful for reasoning about consequences of changes on complex systems.) One knowledge form is <change in state of subsystem> ---> <change in functionality of subsystem = change in state of the immediately larger system>. The knowledge is organized into conceptual specialists corresponding to systems and subsystems connected in a way mirroring the way the systems and subsystems are put together. The control is basically bottom up, following the architecture of the system/subsystem relationship. The changes in states are followed through, interpreted as changes in functionalities of subsystems, until the changes in the functionalities at the level of abstraction desired are obtained. This form of reasoning is useful for answering questions like: "What system functionalities will be compromised if this valve fails closed?".

Hypothesis Matching

Given a concept and a set of data that describe the problem state, decide if the concept matches the situation. The idea here is to encode the routine knowledge for verification and refutation that the concept applies to the situation. One way this can be done is by using a hierarchical representation of evidence abstractions, where the top node determines the overall degree of matching of the hypothesis to the data, and lower-level nodes represent components or features of evidence for the evidence abstraction at higher levels. Form of knowledge are such as to enable evaluation of strength for each evidence abstraction, and to support mapping degrees of belief in each of these evidence abstractions, to degree of belief in the higher abstractions. Strength for an evidence abstraction can be determined Each evidence abstraction can be determined by matching against prototypical patterns which have evidential significance. Samuel's ***signature tables*** can be thought of as performing this task.

How Existing Expert Systems can be Analyzed in This Framework

Separating the implementation language and the intrinsic nature of the tasks has been argued as being salutary for a number of reasons. Let us look at some of the better known expert systems from the perspective of the framework developed so far in this paper.

MYCIN's task is to (i) *classify* a number of observations describing a patient's infection as due to one or another organism, and (ii) once that is done, to instantiate a plan with parameters appropriate for the particular patient situation. We have shown in[16] how the diagnostic portion of MYCIN can be recast as a classification problem solver, with a more direct encoding of domain knowledge and a control structure that is directly appropriate to this form of problem solving.

Prospector[17] *classifies* a geological description as one of a pre-enumerated set of formations. Internist[14] generates candidate hypotheses by a form of enumeration (plausibility scoring and keeping only the top few) and uses a form of *abductive assembly* to build a composite hypothesis that accounts for all of the data. Assembly and hypothesis enumeration alternate. Dendral[15] generates candidate hypotheses by a form of *hypothesis matching* and uses a form of *abductive assembly* which puts together the best molecular hypothesis from the fragments produced by the matching process.

Note that in these analyses we have not mentioned rules (Mycin), networks (Prospector), graphs (Dendral), etc., which are the *means* of encoding and carrying out the tasks. This separation is an essential aspect of what we mean by the "right" level of abstraction in analysis.

Encoding Knowledge at the Level of the Task: The Generic Task Toolset

For each generic task, the form and organization of the knowledge directly suggests the appropriate representation in terms of which domain knowledge for that task can be encoded. Since there is a control regime associated with each task, the problem solver can be implicit in the representation language. That is, a shell for each generic task can be constructed such that, as soon as knowledge is represented in the shell, a problem solver which uses the control regime on the knowledge can be created by the interpreter. This is similar to what representation systems such as EMYCIN do, but note that we are

deliberately trading generality at a lower level to gain specificity, clarity, and richness of ontology and control at a higher level.

We have designed and implemented representation languages for versions of each of the six generic tasks we described. Here is a list of the generic task tools, each with a brief description of the task for which it is designed.

- **HYPER** for matching concept to situation to determine confidence or appropriateness.
- **CSRL** for taxonomic classification, typically a major component of diagnostic reasoning.[18]
- **DSPL** for object synthesis by plan selection and refinement, captures knowledge for certain routine design and planning tasks.[19]
- **IDABLE** for knowledge-directed information passing for intelligent data retrieval.
- **PEIRCE** for assembly and criticism of composite explanatory hypotheses, a form of abduction or best-explanation reasoning.[20]
- **WWHI** (What Would Happen If) for prediction by abstracting state changes.

We have described how this approach directly helps in providing intelligible explanations of problem solving in expert systems.[21] The approach has a number of other implications. E.g.. *uncertainty handling* in problem solving is usefully viewed as consisting of different types for each kind of problem solving, rather than as a uniform general method.[22]

In principle the tools can be used together to build composite problem solvers that integrate the different types of reasoning associated with the generic tasks. Systems have been built integrating more than one type of reasoning (the Red[23] system for example relies on four of the types) but these systems predate the availability of the tools. At present the actual toolset consists of separately implemented tools in a variety of languages; each tool having an incarnation in Interlisp. LAIR has under development an integrated version of the toolset in Common Lisp.

The computational architecture of a problem-solving system (or system component) built with any of the tools is based on functionally distinct, highly modular elements, tightly organized. A generic task problem solver is a community of agents, where each agent is of a specific type, each has its own embedded knowledge. The agents are organized so that they have specific lines of communication with each other; and, depending on the generic task, they pass control around in a well-defined way in order to cooperate and solve the problem. A HYPER-built system is made up of knowledge groups, hierarchically organized; a CSRL-built system consists of a hierarchical community of classification specialist, each specializing in a single classificatory concept. This sort of system architecture, besides making implementation in an object-oriented programming framework fairly easy, makes for systems that are distributable and have predictable concurrencies. The high degree of modularity--modules having clear functions, and clear interactions with other modules--makes for good software engineering at the knowledge level, "structured programming of knowledge bases" if you will.

DSPL for building a Mission Planning Assistant

We will describe the use of DSPL (Design Specialists and Plans Language) for the design and implementation of an expert system in the domain of tactical air mission planning. After investigating KNOBS system from MITRE Corporation, an existing mission planning system, we developed the Mission Planning Assistant (MPA) using DSPL and our generic task approach to building expert systems. KNOBS was the primary source of domain knowledge for the MPA system. Our project had two main goals. First, we wanted to explore the use of DSPL for routine planning tasks. Initially, DSPL was developed as a result of studying a routine mechanical design task.[11] It seemed to us, however, that routine planning shares many of the characteristics of routine design, suggesting that they might require some of the same kinds of problem-solving activities. Secondly, we wanted to investigate the explanation facilities that are necessary in planning systems. We wanted to demonstrate that our generic task architectures provide very natural and comprehensive frameworks for explanation.

Tactical mission planning in the Air Force essentially involves the assignment of resources to various tasks. The resources are primarily aircraft and their stores located at airbases across the theater of operations. The tasks are specified by an "apportionment" order issued by the Joint Task Force Commander to a Tactical Air Control Center (TACC). This order describes the overall military objectives as determined by the Task Force Commander. The TACC is responsible for assigning aircraft and personnel from specific military units to meet the objectives of the apportionment order. The result of these assignments is an "Air Tasking Order" (ATO) which summarizes the responsibilities of each unit with respect to the day's missions. Each mission planned requires attention to such details as the selection of aircraft type appropriate to the mission, selection of a base from which to fly the mission, and coordination with other missions.

The MPA system we developed currently addresses only a single type of mission, the Offensive Counter-Air (OCA) mission. An OCA mission is an air strike directed specifically against an enemy's airbase. Our selection of the OCA mission arose primarily because of the availability of the KNOBS system and its knowledge base of relevant domain facts. Our approach to tactical mission planning treats the Air Tasking Order (ATO) as an abstract device to be designed. The planning of the missions of the completed ATO involves a process similar to the process a designer undergoes when faced with a complex mechanical device to design. A view of design problem solving should illuminate this idea. For a more comprehensive description of design see[11].

Routine Design and DSPL

The general domain of design is vast; it involves creativity, many different problem-solving techniques, and many kinds of knowledge. Goals are often poorly specified, and may change during the course of problem solving. A spectrum of classes of design problems can be discerned, varying in complexity from those problems requiring significant amounts of "creativity", to the most routine design problems requiring no creativity at all. The complexity of a design problem will depend on what pieces of knowledge are available to the problem solver prior to the start of design, that is, the right pieces of knowledge can remove the need for creativity and turn a complex design task into a routine one.

What we have called "Class 3 Design" characterizes a form of routine design activity which postulates that several distinct types of knowledge are available prior to problem solving. First we assume that complete knowledge of the component breakdown of the to-be-designed device is available to the problem solver, including knowledge of what component attributes need to be specified in order to specify a design. The final design will consist only of components known in advance, and no novel components need to be synthesized. Secondly, we assume that knowledge is available in the form of plan fragments describing the actions required to design each component. A plan for designing a component will typically include the designing of subcomponents as steps in the plan. Thirdly, we assume that recognition knowledge is available that will allow the problem solver to select between the alternative plans for designing a component, depending on the design requirements and the state of the problem solving. The problem solving proceeds by following a top-down process of plan selection and refinement, with localized back up and selection of alternative plans upon failure of a design plan at any level. While the choices at each point may be simple, the design process overall may be quite complex, and objects of significant complexity can be designed. It appears that a significant portion of the everyday activity of practicing designers can be analyzed as class 3 design.

In DSPL, a design problem solver consists of a hierarchy of cooperating, conceptual specialists, with each specialist responsible for a particular portion of the design. Specialists higher up in the hierarchy deal with the more general aspects of the device being designed, while specialists lower in the hierarchy design specific subportions of the device, or address other design subtasks. Any specialist may access a design data-base (mediated by an intelligent data-base assistant). The organization of the specialists and the specific content of each one is intended to precisely capture the *human designer's* expertise in the problem domain. Each specialist in the design hierarchy contains *locally* the design knowledge necessary to accomplish that portion of the design for which it is responsible. There are several types of knowledge represented in each specialist, three of which are described here. First, explicit design plans in each specialist encode sequences of possible actions to successfully complete the specialist's task. Different design plans within a specialist may encode alternative action sequences, but all of the plans within a particular specialist are always aimed at achieving the specific design goals of that specialist. A second type of knowledge encoded within specialists is encoded in *design plan sponsors*. Each design plan has an associated sponsor to determine the appropriateness of the plan in the run-time context. The third type of planning knowledge in a specialist is encoded in design plan selectors. The function of the selector knowledge is to examine the run-time judgments of the design plan sponsors and determine which of the design plans within the specialist is most appropriate to the current problem context.

Control in a DSPL system proceeds downwards from the top-most specialist in the design hierarchy. Beginning with the top-most specialist, each specialist selects a design plan appropriate to the requirements of the problem and the current state of the solution. The selected plan is executed by performing the design actions specified by the plan. This may include computing and assigning specific values to attributes of the device, checking constraints to test the progress of the design, or invoking sub-specialists to accomplish sub-portions of the design. Thus a design plan which refers to a sub-specialist is refined by passing *control* to that sub-specialist in its turn. DSPL also includes facilities for the handling of various types of plan failures, and for controlling redesign suggested by such failures.[24]

Mission Planning as Routine Design

We view tactical mission planning as predominantly a routine design task. The problem can be decomposed into the design of subcomponents of the mission plan, where each component can be designed in a fairly independent fashion. The Air Tasking Order is decomposed into various missions or groups of missions of known types. Each mission or group of missions can be planned relatively independently of the others, modulo resource contention considerations. In both the mission planning and the mechanical design domains, each of the solutions to the subproblems must be appropriately combined into the solution for the problem which they decompose. Due to the well known limitations of human problem solving capacities, it is apparent that a *human problem solver* can be successful in such a situation only to the extent that she can decompose the problem into a manageable number of somewhat independent sub-problems, which can be solved separately and combined into a final solution. The MPA system uses DSPL as a natural mechanism for representing the necessary knowledge.

The MPA System

The MPA system contains six specialists. The topmost specialist, OCA, accepts the mission requirements and ultimately produces the final mission plan. The OCA specialist divides its work between two subspecialists, base and aircraft. The base specialist is responsible for selecting an appropriate base, while the aircraft specialist selects an aircraft type. The aircraft specialist has three subspecialists, one for each of three aircraft types known to the MPA system. As needed, one of these specialists will select an appropriate configuration for its aircraft type.

Problem solving begins when the OCA specialist is requested to plan a mission. Currently, the OCA specialist contains only a single design plan which first requests the base specialist to determine a base, and then requests the aircraft specialist to determine (and configure) an appropriate aircraft for the mission. The current base specialist simply selects a base from a list of candidate bases geographically near the target. The aircraft specialist uses considerations of threat types and weather conditions at the target to select an appropriate aircraft type and number for the mission. The aircraft specialist and its three configuration subspecialists represent the most elaborated domain knowledge in the MPA system.

Suppose the mission requirements call for a night raid. The plan sponsors for both the A-10 and F-4 would rule out the possibility of using these aircraft, since (in our domain model) neither of these aircraft have night flying capability. The F-111 plan sponsor, since it is an all-weather fighter with night capabilities, would not be excluded. The plan sponsor for the F-111, based on this and other considerations (range, ability to carry appropriate ordinance, target characteristics, etc.) would find the F-111 suitable for the mission. The plan selector in the aircraft specialist, finding that two design plans have ruled out, would select the "suitable" F-111 design plan. and return this information to the specialist. The specialist executes the F-111 design plan, which includes setting the aircraft type in the mission template to "F-111", and invoking the F-111 configuration specialist which in turn decides an acceptable ordinance load for the F-111 for this mission. Once the configuration of the aircraft is known, the single aircraft probability of destruction in the mission context can be computed. Finally. knowing the mission capabilities of each aircraft, the required number of aircraft can be determined in order to achieve the required probability of destruction, and the necessary number of aircraft can be reserved from the proper unit.

Summary

We have argued the need for high-level task-specific tools for constructing knowledge-based problem-solving systems. We described our approach, based on the notion of generic information processing tasks, and described a toolset which can be used to efficiently build expert systems. Expert systems built according to this methodology have all of the advantages of control strategies and knowledge representations that are especially suited to their information processing tasks. Advantages include knowing what kinds of knowledge to collect, and where to put it away for use in the problem solver, efficient processing at run time, and explanations of system behavior in terms of strategies and problem solving goals keyed to the type of reasoning. We have illustrated the power of these ideas by paying special attention to a high level language called DSPL, and have shown how it was used in the construction of a system called MPA, for assisting with mission planning in the domain of offensive counter air missions.

References

1. B. Chandrasekaran, "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design". *Invited paper appearing in IEEE Expert*

2. David Herman, John Josephson and Ron Hartung. "Use of DSPL for the Design of a Mission Planning Assistant", *The Proceedings of the IEEE Expert Systems in Government Symposium*, IEEE Computer Society, October 1986, pp. (??).

3. Chandrasekaran, B., "Expert Systems: Matching Techniques to Tasks". Paper presented at NYU symposium on Applications of AI in Business. Appears in *Artificial Intelligence Applications for Business*, edited by W. Reitman, Ablex Corp., publishers, 1984.

4. Chandrasekaran, B., "Towards a Taxonomy of Problem-Solving Types", *AI Magazine*,Vol. 4, No. 1. Winter/Spring 1983, pp. 9-17.

5. McDermott, J., "R1: A Rule-Based Configurer of Computer Systems", *Artificial Intelligence*,Vol. 19. Issue 1,, 1982, pp. 39-88.

6. Shortliffe, E.H., *Computer-based Medical Consultations: MYCIN*, Elsevier/North-Holland Inc., 1976.

7. Szolovits, P./ Pauker, S. G., "Categorical and Probabilistic Reasoning in Medical Diagnosis", *Artificial Intelligence*, , 1978, pp. 115-144.

8. Clancey, William J., "The Epistemology of a Rule-Based Expert System--a Framework for Explanation", *Artificial Intelligence*,Vol. 20, No. 3. May 1983, pp. 215-251.

9. Chandrasekaran, B. / Mittal, S. / Gomez, F. / Smith M.D., J., "An Approach to Medical Diagnosis Based on Conceptual Structures", *Proceedings of the 6th International Joint Conference on Artificial Intelligence*,August 1979, pp. 134-142, IJCAI 1979

10. Chandrasekaran, B. / Mittal, S., "Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems", in *Advances in Computers*, M. Yovits, ed., Academic Press. Vol. 22, 1983, pp 217-293.

11. David C. Brown and B. Chandrasekaran. "Knowledge and Control for A Mechanical Design Expert System", *Computer*,Vol. 19, No. 7. July, 1986, pp. 92-100.

12. Josephson, J. R.; Chandrasekaran, B.; Smith, J.W., "Assembling the Best Explanation", *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, IEEE Computer Society, Denver. Colorado, December 3-4 1984, pp. 185-190, A revised version by the same title is now available as a technical report.

13. Josephson J. R., Chandrasekaran B., Smith J., Tanner M., *Abduction by Classification and Assembly*, Philosophy of Science Association, East Lansing. Michigan, 1986, pp. 458-470, ch. VII.

14. Pople, H. W., "Heuristic Methods for Imposing Structure on Ill-Structured Problems", in *Artificial Intelligence in Medicine*, P. Szolovits, ed., Westview Press. 1982 , pp. 119-190.

15. Buchanan, B. Sutherland, G. / Feigenbaum, E.A., "Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry". in *Machine Intelligence 4*, American Elsevier, New York, 1969.

16. Sticklen, Jon / Chandrasekaran, B. Smith, J.W. Svirbely, John, "MDX-MYCIN: The MDX Paradigm Applied To The MYCIN Domain", *International Jour. Computers And Mathematics with Applications*,Vol. 11, No. 5, 1985, pp. 527-539.

17. Duda, Richard O./ Gaschnig, John G./ Hart, Peter E., "Model Design in the Prospector Concultant System for Mineral Exploration", in *Expert Systems in the Microelectronic Age*, Michie, D., ed., Edinburgh University Press, 1980, pp. 153-167.

18. T. Bylander and S. Mittal, "CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling", *AI Magazine*,Vol. 7, No. 3, August 1986, pp. 66-77.

19. D. C. Brown, "Expert Systems for Design Problem-Solving Using Design Refinement with Plan Selection and Redesign". PhD dissertation

20. W. F. Punch III, M. C. Tanner and J. R. Josephson, "Design Considerations for PIERCE, a High-Level Language for Hypothesis Assembly", *Expert Systems in Government Symposium*,October 1986, pp. 279-281.

21. Chandrasekaran, B., "Generic Tasks in Expert System Design and Their Role in Explanation of Problem Solving". Invited paper presented at the National Academy of Sciences/ Office of Naval Research Workshop on Distributed Problem Solving, May 16-17, 1985, Washington, D.C., appears in the *Proc. of the Workshop* to be published by the National Academy of Sciences.

22. Chandrasekaran, B. / Mittal, S. / Smith, J. W., "Reasoning with Uncertain Knowledge: The MDX Approach", *Proc. 1st Ann. Joint Conf. of the American Medical Informatics Association*,May 1982.

23. Smith, Jack W.; Svirbely, John R.; Evans, Charles A.; Strohm, Pat; Josephson, John R.; Tanner, Mike, "RED: A Red-Cell Antibody Identification Expert Module", *Journal of Medical Systems*,Vol. 9, No. 3, 1985, pp. 121-138.

24. D. C. Brown, *Failure Handling in a Design Expert System*, Butterworth & Co.. London, 1985.

Monitoring Robot Actions for Error Detection and Recovery

M. Gini and R. Smith
University of Minnesota
Minneapolis, MN 55455

ABSTRACT

Reliability is a serious problem in computer controlled robot systems. Although robots serve successfully in relatively simple applications such as painting and spot welding, their potential in areas such as automated assembly is hampered by programming problems. A program for assembling parts may be logically correct, execute correctly on a simulator, and even execute correctly on a robot most of the time, yet still fail unexpectedly in the face of real world uncertainties. Recovery from such errors is far more complicated than recovery from simple controller errors, since even expected errors can often manifest themselves in unexpected ways.

Here, ~~In this paper we present~~ a novel approach is presented for improving robot reliability. Instead of anticipating errors, ~~we~~ use knowledge-based programming techniques so that the robot can autonomously exploit knowledge about its task and environment to detect and recover from failures. ~~We~~ They describe preliminary experiment of a system that ~~we have~~ they designed and constructed in our Robotics Lab.

1. INTRODUCTION

We want to make robots more dependable so that they can be trusted when left unattended. This paper describes the design and development of a robot system that continues to operate satisfactorily even after it encounters a serious error [Gin83b], [Gin85c].

Failures in achieving a task are the result of errors, but not every error produces an immediately detectable failure. Errors can occur at many levels, at the mechanical level (a joint becomes locked), at the hardware level (a sensor does not function properly so that the robot is driven to exceed its joint limits), at the controller level, in the computer controlling the robot (either at the hardware or the software level), and in the environment. We are mostly interested in errors in the environment because they tend to be more unpredictable and difficult to characterize with mathematical models. We are interested in errors in the component parts used for the assembly, and in errors in the work cell (loaders, feeders, conveyor belts, tools). Our goal is to automatically detect problems caused by collisions, jammed parts, gripper slip, misorientation, alignment errors, and missing parts.

In practice, robot systems that can recover from errors without human intervention do not exist today because robot control programs cannot handle the vast range of possible error conditions. It takes uncommon skill and experience to develop such a program, and the resulting program will then only apply to the specific robot task at hand. Moreover, the program may have to be largely rewritten to handle even a minor change in the robot's task [Car85], [Loz83].

A difficult problem in automatic error detection and recovery is detecting that something significant has occurred. Many events are usually reported to the robot controller but not all of them are significant. The same event may be important in some circumstances and almost irrelevant in others. Deciding when something is important is the first step in the error detection process. The second step involves detecting the cause of the error and its effect on the robot environment. Errors might appear a long time after what caused them happened making it more difficult to detect them. Some errors do not affect the execution of the task so they could be left unrecovered. Only after the cause for the error has been identified or, at least, after alternative plausible causes have been found the recovery activity can start.

The robot system discussed here is geared towards industrial assembly tasks. The assembly to be performed is described in a robotics language. If an error occurs while the robot is performing the task, the robot detects the error, and dynamically plans the steps it must take to recover. To do this, the robot applies general knowledge about robots, and assembly tasks, plus specific information about the robot and the program in question.

Our approach simplifies robot programming by putting the burden for general error recovery on the robot system itself. The programmer can concentrate on the task at hand and minimize later maintenance if the robot recovers from most errors itself. This saves engineering time as well as robot downtime.

The system described here works in conjunction with an existing robot programming language. We show later in the paper how we have used it with an implementation of the AML language for the IBM 7565 robot. In addition to developing the testbed with the IBM robot we have developed prototype components of a simulated system. The simulated version is currently more complete than the testbed version since it is much easier to control a simulated robot than a real one.

2. INTELLIGENT ROBOT ERROR RECOVERY

Our system operates in two phases: offline, and online. Figure 1 illustrates its structure. Each box represents a separate program that may run as an independent process. The Preprocessor prepares the assembly task for execution by the robot by generating an Augmented Program (AP). The AP Executive interprets the robot program and monitors the robot's operation. If a serious error occurs, it activates the Recoverer. The Recoverer examines information from the event trace and from the task knowledge base and devises a recovery plan. More details on the component of the system are provided later in the paper.

Since the Preprocessor and the Recoverer both rely on symbolic computational techniques and do not require real-time performance they both reside on the same processor, referred to here as the Manager. Robot control operations require real-time response and reside on other processors.

Design Features of the Testbed

The design of the error recovery testbed incorporates four features of particular interest. First, the automated reasoning of the Manager and real-time functions of the AP Executive operate independently and execute on separate processors. Second, sensors are activated and monitored selectively according to the robot's current action. Third, sensor data are evaluated and assigned qualitative meanings at several levels

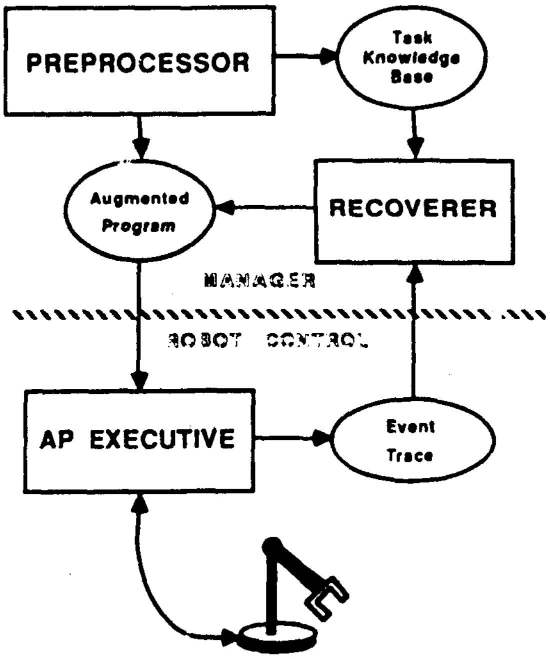


Figure 1: Error Recovery System Components

of the system. Fourth, if the robot's task fails repeatedly, the AP Executive will handle successive restarts and recoveries without ill effect.

The present configuration of the error recovery testbed manages an IBM 7565 manufacturing manipulator. The components of the error recovery Manager exist on an LMI Lambda processor. Motion control and sensor filtering for the IBM 7565 is implemented on an IBM Series 1 minicomputer using the AML programming system [Tay82]. The AP Executive resides on an MC68000 based personal computer system, an Apple Macintosh.

The IBM 7565 is well suited for these experiments. It is a cartesian robot moving on linear tracks over a rectangular work cell. The gripper has six degrees of freedom and its jaw contains six strain gauges. The 7565 is provided with the AML robot programming system which can be used to develop relatively sophisticated programs. AML provides facilities for monitoring the robot sensors and for performing robot motion subject to the presence of appropriate sensor readings.

Separation of reasoning and real-time

The testbed utilizes separate processors for executing the reactive, or real-time, software components and for executing the reflective, or symbolic reasoning, components of the system. Providing separate processors for the real-time and the automated reasoning components of the system prevents time-critical software components from having to compete for computation time. The choice also allows us to choose a managing processor for its symbolic computation capabilities rather than its real-time capabilities. Since the AP Executive is the only component that interacts with the robot continuously, a large scale system could probably share a single Manager among several independent work cells, each with its own AP Executive.

The Manager initiates a task by transmitting the appropriate Augmented Program (AP) to that cell's AP Executive. In return, the AP Executive transmits the event trace of the task's execution. The physical separation of functions makes this information exchange particularly important. The AP must contain all information the AP Executive requires to operate the robot in the work cell. The event trace must contain all information relating to the task's progress necessary to reconstruct activities that took place in the work cell. Whenever feasible and appropriate, the event trace contains specific numerical sensor readings from the work cell.

3. THE PREPROCESSOR

Our system uses a manipulator level robot programming language to specify the task the robot is to execute. This description is given in the AL robot programming language [Gin85b], [Muj79], though any other manipulator level language should work as well. Even though AL is not used in any commercial robot, it has many of features many languages have adopted. We have expanded AL to handle descriptions of objects so that it can be used to drive our graphic simulation system. We have chosen to use a "robot level language" to describe the task rather than a "task level language" because robots in real use are programmed with robot level languages. Starting from an existing and accepted level of language will allow us to grow to more sophisticated languages and yet to keep our ability to experiment with existing commercial robots.

We assume that the task description is accurate and correct in the sense that a robot simulator would execute it reliably. Thus the only errors the system should expect will be introduced by real world uncertainties.

We can't use the original AL program for online monitoring; we rely on an augmented version of the AL program to direct the job of monitoring the robot's activities. We call this program Augmented Program (or AP). The AP is structured as a finite state machine. The machine is represented with a directed graph in which the nodes are arbitrary states. Activities performed by the procedure are specified on the arcs connecting the states; you derive the sequence of actions in the procedure by traversing a series of arcs in the diagram. Each arc has 'events' or 'preconditions' attached to its activities. If a particular arc is leaving the state the system currently is in, then the activities on that arc are performed when the preconditions are satisfied. If two or more arcs leave a given state, then the AP Executive chooses the arc whose preconditions are satisfied first. This structure is especially useful in systems where several asynchronous events (sensor inputs) select and trigger subsequent actions.

This approach lends itself readily to the representation of AL programs. The sequencing of instructions in the original AL program is replaced by transitions in the AP. The AP representation also relates explicit sensor data to what the robot is supposed to be doing. Crucial sensor readings preceding some action in the AL program will correspond to the preconditions of the corresponding state transition in the AP. The preconditions on arcs leading out of a given state will correspond to the set of sensor readings to be monitored by the sensor handler. Thus, the set of significant sensor readings will change automatically as the robot proceeds through its task.

Extracting high-level intentions from an AL program

Functioning of the AP Executive is highly dependent upon the information derived from the off-line portion of the system. In order for the AP Executive to interpret sensory data, the off-line component must provide the intent of the AL instructions. For example, if the intent of an instruction "move.." is to transport an object to a given location it is important to check for slippage by monitoring the touch sensors. If the intent is merely to move the arm, touch sensor data may be irrelevant. The off-line component must be able to extract the semantics of the program.

In our system methods from extracting intents of programs are based on syntactic matching and heuristics. By syntactic matching we mean identifying sequences of instructions that suggest operations such as grasping an object or releasing it. We have explored techniques of this type with excellent results. We can already handle difficult cases, such as, for instance, identifying when an object is grasped from inside a hole. Additional details can be found in [Gin85a], [Gin85c].

Figure 2 shows a state transition diagram used in conjunction with syntactic matching to identify intentions of AL instructions. Arcs shown in gray indicate transitions that have no clear interpretation and that require user intervention.

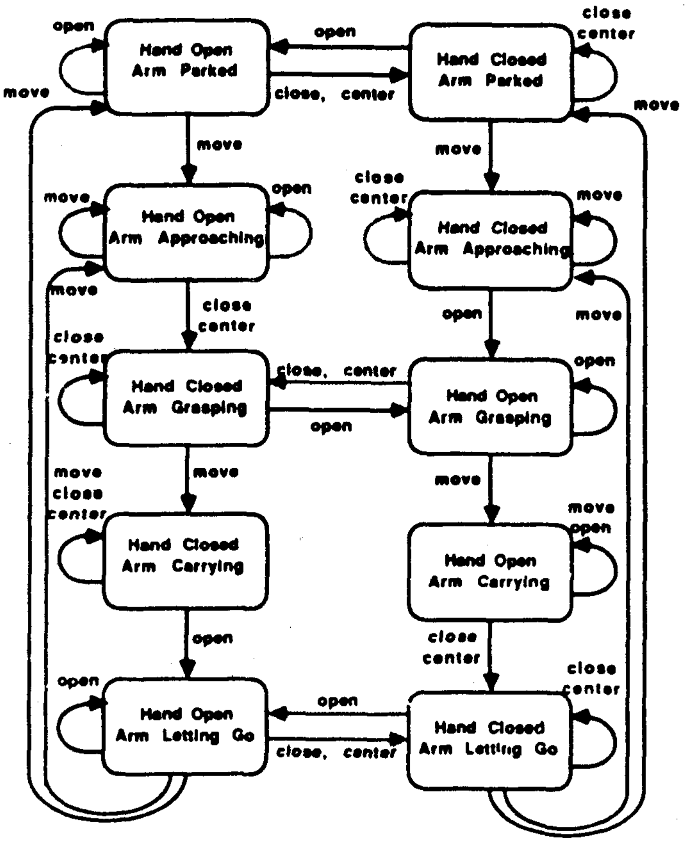


Figure 2: State Transition Diagram

Constructing the Expected World Model

The world model is another critical piece of information for the AP Executive. Suppose that a 'move object' instruction is in execution when the proximity sensors of the hand are activated. Knowledge about the placement of objects in the environment would help determine the cause of the impending collision. In addition to placement of all objects in the environment, geometric and non-geometric properties of objects will be important. If an object being transported is slippery, the chance of dropping it is increased. If it is quite large, chances of a collision may increase, and so on.

The Preprocessor simulates the execution of the AL program to build an Expected World Model. In our current implementation the Expected World Model contains properties of the objects in the environment such as their position after the execution of each instruction, when they are moved, wich robot moves them, etc.

A classical problem with creating a world model offline is caused by branches in the program that create alternative models. Our world model has a tree structure in which branching nodes are labelled by the condition that controls the branch. The existence of the Expected World Model reduces the amount of reasoning during the recovery because we can compare the current with the expected situation to get clues about problems.

Translating an AL Program into AP.

The Preprocessor module has to provide information on how errors can be detected and what errors are likely for each instruction. For example, a 'move object' instruction might lead to a slippage error with high probability. This indicates that finger separation or touch sensors should be checked during the movement. Using the intent of the AL instructions, the Preprocessor generates in the AP program the appropriate sensory conditions to be checked to guarantee the correct execution of the instruction.

We need also to know what sensor values are relevant to detect unexpected events. Some of this knowledge is obtained from the robot program (such as the position of the robot, or the opening of the fingers), some could be obtained from the CAD data base (such as the maximum pressure to exert when grasping an object). Some values cannot be known before executing the program. Sometimes the position of an object is identified only by sensors. Hence it is important to know that no value is known in advance and that sensor data will be used.

A simple AL program and the corresponding AP are illustrated in Figure 3a and 3b. The task described in the AL program is a simple pickup operation. In the AL program WOKH indicates the robot hand and WOKA the robot arm. The example shows that we have modified AL to allow for different names of robots and different

```
OPEN WOKH TO 1.5;
MOVE WOKA TO FRAME (ROT (ROLL,-45),
 VECTOR (-7.77, -14.41, 4.4));
CENTER WOKH;
MOVE WOKA TO FRAME (ROT (ROLL,-45),
 VECTOR (-7.77, -14.41, 7));
```

Figure 3a: Example AL Program

```
((1 ((robot-do open wok 1.5))
    ((open wok) 2)
    ((hand-error wok) 12))
 (2 ((robot-do move wok (-7.77 -14.41 4.4 -45 0 0)))
    ((reach wok) 3)
    ((hit wok) 12)
    ((joint-error wok) 12))
 (3 ((imply create obj-block (-7.77 -14.41 4.4 -45 0 0))
     (expect grasp wok obj-block)
     (robot-do center wok))
    ((center wok) 4)
    ((crush wok) 12)
    ((hand-error wok) 12))
 (4 ((imply grasp wok obj-block)
     (expect carry wok obj-block)
     (robot-do move wok (-7.77 -14.41 7 -45 0 0)))
    ((reach wok) 5)
    ((hit wok) 12)
    ((untouch wok) 12)
    ((joint-error wok) 12))

          (other states in the program ... )

 (12 ((imply error) (robot-do notice operator))))
```

Figure 3b: Augmented Program derived from Figure 3a

ways of expressing rotations. In the corresponding AP states are numbered. Each state contains a collection of entries. The first of them describes the action the robot has to do ("robot-do") and the meaning of the instruction in the physical world. In particular, "expect" shows what is expected to happen at the end of the state if all goes well, "imply" shows logical deductions about objects or the robot that can be made from the intentions extracted during the preprocessing phase. Each other entry specifies a condition to be checked using sensor data and the next state if the condition becomes true. Since the Preprocessor generates the conditions using knowledge about the intention of each robot action the same AL statement usually generates different conditions. As an example we can look at the states 2 and 4 in Figure 3b.

We should note that there is only one error state in the AP. When an error is detected a transition to the error state is generated. We do not use specific error states because we need to check causes of the error to find the recovery procedure. We consider the sensor data obtained as symptoms of the error not as its diagnosis. Often software that handles failures does not make the distinction between symptoms and errors or it assumes that there is a deterministic mapping between symptoms and errors.

4. THE AP EXECUTIVE

The AP Executive is responsible for maintaining an accurate picture of what the robot does. The Recoverer needs to know the robot's situation but the potential complexity of the robot's activities make it hard to derive the necessary details from the program's state. It is easiest for the AP Executive to keep track of what the robot is doing and what objects it is manipulating. The AP Executive can then provide the robot's recent history and a catalog of objects in the workspace when an error occurs.

The AP Executive does more than simply observe and report on the robot's actions. It takes responsibility for issuing commands to move the robot. When the AP says that a robot action is to occur, the AP executive sends the command to the robot. The AP Executive tracks the robot's activities by monitoring data from the robot's sensors. The sequence of sensor data yields an event trace from which we get the robot's recent history.

The Workspace Model

To recover from an error, the system needs to know what objects are in the workspace and where the objects are. At the time of error it should be easy to find out where the AL program failed and what the values of the program's variables are. Unfortunately, we can't deduce the state of the workspace from the state of the program. The program just doesn't keep the right kind of information. But it is possible to deduce when and how the AL program manipulates objects. To monitor objects in the workspace the AP Executive has to be told when an object is acquired, grasped, moved, and discarded. Typically the robot 'acquires' an object from a part dispenser, moves it somewhere, and maybe 'discards' a part by placing it on a conveyor. This is sufficient to keep track of what objects are in the workspace and where they are. The workspace model update process can then follow objects by monitoring such activities in the event trace.

The minimum workspace model is a catalog of objects and their locations. Along with the robot's most recent activities, this model gives enough information to determine what was going on at the time of an error. The Recoverer uses this information to key into more information about the object stored in the offline world model. More details about the Workspace modeller can be found in [Smi86a].

Sensor management through filtration

In the error recovery testbed, sensor information is filtered several ways. Initially, the task's AP identifies specific sensor information that is significant to the execution of that task. This information is given in terms of sensory events specified symbolically that can cause state transitions in the AP. The sensory information is used both to identify potential state transition events and to filter sensory information for the event trace. This information is also passed to "sensor filter" tasks that activate appropriate sensors and map sensor values into events significant to the progress of the robot's task.

Augmented Programs are instruction sequences structured as finite automatons. AP transitions are caused by discrete events, so a robot's progress at its task depends on the occurrence of events that cause appropriate transitions. Significant sensory readings must be mapped into events that cause state transitions. This mapping provides one form of sensory filtration: sensory readings are reported only when the value is significant to the progress of the robot's task. In some cases the identity of the event is the only specific sensory information returned and in other cases numerical data is included in the event trace as well.

Each AP state contains event predicates identifying sensor readings that would be significant to the successful execution of that state. The AP Executive passes the information in the event predicates to the appropriate sensor filters before initiating robot motion. The sensor filters activate
app
ropriate procedures so that necessary sensor readings will take place. The AP Executive omits all sensor information from the event trace that is not identified in the AP as being significant.

Grip sensor filtering on the IBM 7565 is implemented using "monitor" facility of the AML language [Tay82]. Monitors are used to define ranges of sensor values that can activate user-defined procedures or terminate robot motions. When initializing the IBM 7565, the AP Executive defines a set of monitors for classifying gripping forces and associates each monitor with an AP event type. The numerical values used for classifying gripping force depend on the objects being used in the robot's task and the actions performed on them, so these values may be adjusted during initialization.

When the AP Executive gives the IBM 7565 a motion command, it also specifies a set of monitors to activate. The AML system collects the appropriate sensor readings for each active monitor and "trips" the appropriate monitor if its sensor enters the monitor's defined range. This terminates the motion in progress and generates a message to the AP Executive identifying the qualitative value of the sensor reading, as determined from the monitor that was tripped. If nc monitor terminates the active motion, a similar message indicating uninterrupted completion is sent instead. fhe AP Executive then generates a sensor event and, if necessary, updates the event trace and performs an AP state transition.

Since the IBM 7565 is normally programmed in the AML language, the AP Executive translates AP commands and sensor specifications into an AML compatible form. Since AML is a manipulator level language, the mapping of robot actions from the AP form is straightforward. Mapping sensor operations is more complex since APs specify symbolic sensor readings. Figure 4 shows an example of the transformation of a "move" operation in an AP state into the corresponding AML commands executed by the robot. The desired destination and the desired AML monitoring sets to be activated (E_HIT and E_UNTOUCH) are passed to the APM procedure. This procedure, written in AML and executing on the IBM series 1, performs the MOVE operation and the related filtering for the 7565's sensors. The procedure activates the appropriate monitors and performs the motion subject to the selected monitors.

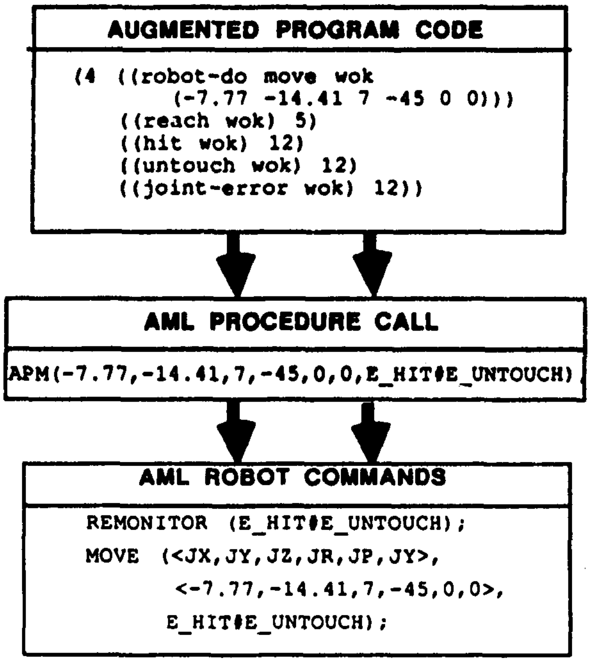
Qualitative sensor interpretation

Although the event trace often provides numerical sensor data, such information is not of primary importance when reasoning about the robot's activities. To meet this need, the error recovery system assigns symbolic meanings to numerical sensor values in a number of ways. Spatial locations and critical dimensions are assigned symbolic names. Gripping forces are assigned qualitative values according to the range in which a force value falls.

Qualitative classification of sensor data often serves a second purpose as well. When executing an AP, the AP Executive responds to events in terms of symbolic classifications. Upon successful completion of a motion command the AP responds to a "reach" sensor event instead of examining and matching the robot's reported destination. If the gripper drops an object and the gripping force drops to a small value, the AP responds to an "untouch" sensor event instead of testing the specific force value. The classification of sensor values into different types of AP events is performed by a sensor filter procedure that operates on the behalf of the AP Executive, as described above.

Ideally, qualitative classification of sensor data should be performed by a component of the Manager and exploit its increased computational capabilities and knowledge bases. Symbolic classification of spatial information is an example of this. The robot's sensor filter identifies whether successful completion of a robot motion caused the robot to reach its sensed position, but the filter does not try to identify the location in terms of the robot task's overall goals. Identification of the particular location is handled by the Manager's work cell modeller.

When an error occurs the Manager produces a model of the robot work cell in terms of its probable state and its intended state. Locations visited by the robot or by objects in the work cell are assigned symbolic identifiers, and a history is produced of visits for each location, object, and robot gripper in the work cell. Error recovery planning consists of producing a sequence of robot actions to change the work cell from its erroneous state to its intended state.



```
(4 ((robot-do move wok
      (-7.77 -14.41 7 -45 0 0)))
   ((reach wok) 5)
   ((hit wok) 12)
   ((untouch wok) 12)
   ((joint-error wok) 12))
```

```
APM(-7.77,-14.41,7,-45,0,0,E_HIT#E_UNTOUCH)
```

```
REMONITOR (E_HIT#E_UNTOUCH);
MOVE (<JX,JY,JZ,JR,JP,JY>,
      <-7.77,-14.41,7,-45,0,0>,
      E_HIT#E_UNTOUCH);
```

Figure 4: Converting from AP statements to AML statements

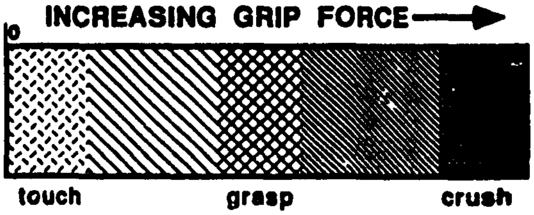


Figure 5: Qualitative interpretation of grasping forces

Qualitative interpretation of gripper forces, on the other hand, must be performed by a sensor filter. Gripper forces, when they are significant, determine whether the gripper is touching an object and holding with an adequate force. Identification of appropriate touching and grasping forces must be communicated to the AP Executive so that appropriate AP state transitions occur depending on the gripping forces encountered. The sensor filter classifies gripping forces into specific ranges according to the robot's current action. Each range corresponds to a type of sensing event that can be produced by the gripping force sensor. Figure 5 shows an example of that.

5. THE RECOVERER

We are interested in discovering causes for errors and errors might appear a long time after what caused them happened [Smi86b]. Error interpretation becomes more difficult as the complexity of the task increases. For example, consider a task where a robot moves cubes from a feeder to a shipping pallet, twelve at a time. What might happen if the the cube falls from the gripper and lands on the pallet, knocking another object off? Most of the failure reason models available only apply to the objects and situations directly related to the sensor reading indicating the error. The robot thus only associates an error with a part if it uses its sensors on the part and finds an error. The lost part won't be missed until someone down the line tries to unload the pallet and finds it one part short.

The symbolic model of the work cell constructed before execution of the task and the trace of events are used here. When something unexpected happens we can trace the error back until we find critical measurements or assumptions made that were not supported by sensor data. For instance, in general we assume that if something is left in a stable configuration it will remain there until a fact appears that shows the configuration has changed. If we discover later that the object moved it means that something happened to move it that was not explicitly noted.

We have developed methods for symbolic tracking of objects from event traces. Common sense heuristics are used for symbolic tracking. For instance, if the robot is holding an object and the robot moves, then we know that the object moves to the same place. Tracking an object in real time with sensors is too expensive, unless we know where to look for it, and how it looks like. Symbolic tracking is less expensive from a computational point of view, and helps in reducing the number of possible causes for errors. After that number has been reduced we can get additional sensor data to guarantee that the correct cause for the error has been identified.

It is curious to observe that most of the AI work in planning has concentrated on checking preconditions before executing every action to guarantee that they are satisfied in the current state of the world. Postconditions are not checked, but are used solely to update the world model. So an error can go unchecked for a while and can be detected only when it affects the preconditions of another action. We check selected conditions after the execution of each instruction to guarantee that failures are identified as soon as they appear. This still does not solve the problem of errors caused by the robot during the movement that require different sensors to be checked. For instance, if the arm bumps into objects during a transfer motion without losing the part it is carrying no error will be reported. This requires failure reason analysis when an error is found.

Once an error has been detected, the recovery process can start using a trace of relevant events and whatever information is available about the task to determine the causes and effects of the error. It is only after the cause for the error has been identified or, at least, after alternative plausible causes has been found the recovery process can start.

To be more specific, if an AP state transition leads to an error state, a message to that effect is appended to the event trace and the trace is passed to the Recoverer component of the Manager. The Recoverer generates a model of the current work cell's state and of its desired state. This model is used to produce a recovery plan in the form ot AP states to be appended to the task's existing AP. The Manager passes these additional states back to the AP Executive where they are executed. If the new states each execute successfully, they will lead the task back to a state in the original AP.

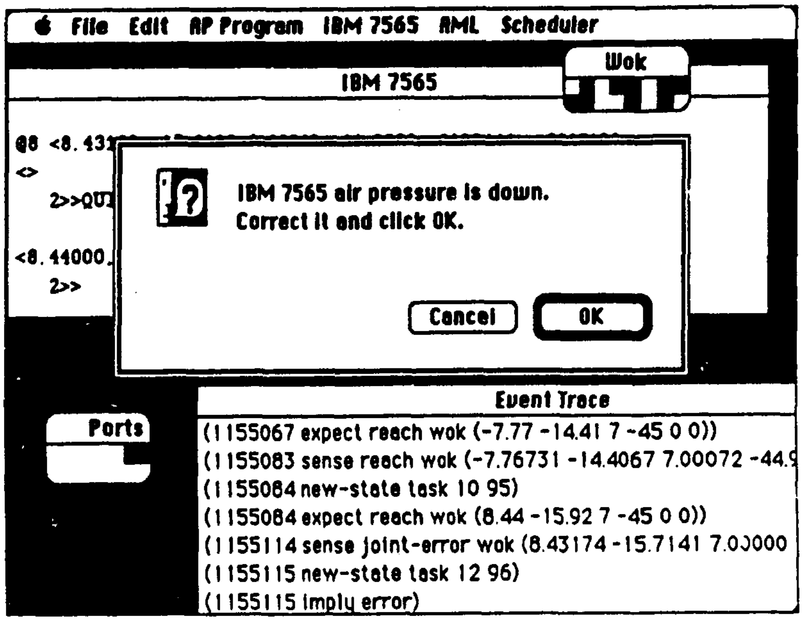
To successfully effect recoveries in this manner, the Recoverer requires a copy of the task's AP and the information in the event trace. The Recoverer can also exploit knowledge about the robot's task, the parts involved, and the work cell to produce the recovery plan. To simplify experiments with error recovery as well as for improved performance in industrial situations, the testbed's AP Executive can handle repeated failures and subsequent recoveries by a robot task. The AP Executive can also display messages on the AP Executive's display screen for explaining error diagnoses or for instructing the robot's operator.

During normal execution, the AP Executive contains a copy of the robot task's AP. If an error recovery occurs, the Recoverer passes additional AP states to the AP Executive. These additional states do not replace existing states in the AP; they are appended to them. To recover, the AP Executive resumes task execution with the first of recovery states passed to it. Once the recovery execution begins, the AP Executive treats the recovery states identically to the states in the original AP. If another failure occurs, whether during the recovery or after completing the recovery, the AP Executive again reports the failure to the Recoverer and resumes executiun when it receives a set of recovery states.

For example, if the robot loses a part, it can attempt a recovery by opening the gripper, moving to the work cell surface, and trying to grab the part. If the part is there, the recovery can proceed. If the grasp fails, the AP Executive simply informs the Recoverer which can then produce another recovery plan and try again.

The ability to do multiple recoveries allows the Recoverer to profit from mistakes in a recovery plan. When faced with multiple recovery choices, the Recoverer can choose the one that is most likely to reduce uncertainty about the state of the work cell. The Recoverer can also produce recovery plans with the sole purpose of taking sensor readings in the robot work cell. If the Recoverer needs to probe a specific spatial location it can produce a recovery plan that performs the desired sensor reading and then immediately fails. The resulting event traces will increase the amount of information in the work cell model and the unsuccessful recovery will not prevent a subsequent recovery from being attempted.

Another useful feature during error recovery is the AP Executive's ability to display messages for the robot's operator. These messages are produced by statements in the AP and thus may be generated by the Recoverer. This facility allows the Recoverer to request specific operator intervention when necessary. This permits experiments with failures that tax the available sensory or reasoning facilities. In the Testbed it also allows experiments with primitive Recoverer software that simply diagnoses the problem and asks for operator intervention. This capability may also have worthwhile industrial applications: the Recoverer could produce messages to guide the robot's operator in manually correcting problems in a complex and unfamiliar assembly. An example is shown in Figure 6.



```
@8 <8.43
<>
   2>>QU

<8.44000
   2>>
```

IBM 7565 air pressure is down.
Correct it and click OK.

Event Trace

```
(1155067 expect reach wok (-7.77 -14.41 7 -45 0 0))
(1155083 sense reach wok (-7.76731 -14.4067 7.00072 -44.9
(1155084 new-state task 10 95)
(1155084 expect reach wok (8.44 -15.92 7 -45 0 0))
(1155114 sense joint-error wok (8.43174 -15.7141 7.00000
(1155115 new-state task 12 96)
(1155115 imply error)
```

Figure 6: Operator display by AP Executive

6. RELATED WORK AND SUMMARY

Our approach differs from other approaches in significant aspects. The current state of the art in industrial robots is that either the robot executes its task regardless of its success or it quits every time it encounters something unexpected [Luh83]. A better approach is to handle the situation by preprogramming error checking and error recovery procedures for every probable error [Bon82], [Gin83a], [Gin85b], [Tay82] This is an expensive method both in engineering and in robot computational resources. It is also easy to forget some important checks.

Since it is difficult to consider all possible errors, many of which might never happen, another method is to generate from the task level description a program that is guaranteed to be correctly executed ever in the presence of uncertainties in the environment. This requires models of robot kinematics and dynamics, and models of physical properties of objects such as friction. This approach has been applied only to fine motions for specific tasks such as insertion operations [Loz84]. Modeling uncertainties [Bro82] and taking into account errors in the model [Don86] helps but the real world is so complex that it might not be worth developing sophisticated models of it.

Much previous research in Artificial Intelligence has centered on detection and correction of errors in simulated robot systems [Wil84]. These studies all make a number of assumptions: knowledge about events is correct, each action produces precisely defined postconditions, there are no uncertain data, correct predicates are generated from sensor data every time they are needed, and sufficient knowledge is provided to take into account all the possible states of the environment. These assumptions are too strict to be realistic.

A few exceptions exist. The most notable is STRIPS [Fik71] the system used to control the mobile robot Shakey. Srinivas [Sri76] [Fri77] has designed a system for analyzing the causes of failures in robot programs and for replanning the robot activity. More recently, work has been done on monitoring the execution of programs with real robots [Lee83], [Lop86]. There is a growing interest in modeling sensors [Fox83], [Hen84], and planning for their use [Doy86] that will provide needed background for work on error detection and recovery.

Our approach is more similar to the way people handle errors and unexpected events. By relating events to general knowledge human beings can identify unexpected situations and by applying common sense and domain specific knowledge they can find solutions to situations never seen before. The key to human performance is in the knowledge about the environment and about the specific task at hand. We want to do something similar for assembly robots. Since the domain is limited and reasonably constrained the amount of knowledge needed can be managed by using present technology [CAR84].

ACKNOWLEDGEMENTS

Several individuals have contributed to the research presented in this paper. We would like to acknowledge the contribution of Rajkumar Doshi, Sharon Garber, Marc Gluch, Shideh Hojat, and Imran Zualkeman.

This work was funded in part by NSF grant DMC-8518735, by the Microelectronic and Information Sciences Center and by the Productivity Center of the University of Minnesota.

REFERENCES

[Bon82] Bonner, S., and Shin, K., "A comparative study of robot languages," Computer Magaz., December 1982, pp 82-96.

[Bro82] Brooks, R. A., "Symbolic Error Analysis and Robot Planning," International Journal of Robotics Research, Vol 1, N 4, Winter 1982, pp 29-68.

[CAR84] Committee on Army Robotics and Artificial Intelligence, et al, "Applications of Robotics and Artificial Intelligence to reduce risk and improve effectiveness," Robotics and Computer-Integrated Manufacturing, Vol 1, N 2, pp 191-222, 1984.

[Car85] Carlisle, B., "Key Issues of Robotics Research", in Hanafusa, H., and Inoue, H. (eds) Robotics Research, The Second International Symposium, The MIT Press, Cambridge, Mass, 1985, pp 501-503.

[Don86] Donald, B., "Robot Motion Planning with Uncertainty in the Geometric Models of the Robot and Environment: a Formal Framework for Error Detection and Recovery", Proc 1986 IEEE Conference on Robotics and Automation, pp 1588-1593, San Francisco, April 1986.

[Doy86] Doyle, R.J., Atkinson, D.J., and Doshi, R.S., "Generating Perception Requests and Expectations to Verify the Execution of Plans", Proc. AAAI-86, Philadelphia, August 1986, pp 81-87.

[Fik71] R. E. Fikes, and N. J. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving", Artificial Intelligence 2, pp 189-208, 1971.

[Fox83] Fox, M.S., et al., "Techniques for Sensor-Based Diagnosis", Proc IJCAI 83, pp 158-163.

[Fri77] Friedman, L., "Robot learning and error correction," Proc. 5th International Joint Conference on Artificial Intelligence, pp 736, 1977.

[Gin83a] Gini, G., and Gini, M., "Explicit programming languages in industrial robots", Journal of Manufacturing Systems, Vol 2, N 1, 1983.

[Gin83b] Gini, M., Gini, G., "Towards automatic error recovery in robot programs." Proc. 8th International Joint Conference on Artificial Intelligence, August 1983, pp 821-823.

[Gin85a] M. Gini, R. Doshi, M. Gluch, R. Smith, I. Zualkernan, "The role of knowledge in the Architecture of a Robust Robot Control," Proc. 1985 IEEE International Joint Conference on Robotics and Automation, pp 561-567, March 1985, St. Louis, Missouri.

[Gin85b] Gini,G. , Gini, M., "Dealing with world model based languages," ACM Trans on Programming Languages, Vol 7, N 2, April 1985, pp 334-347.

[Gin85c] M. Gini, R. Doshi, S. Garber, M. Gluch, R. Smith, I. Zualkernan, "Symbolic Reasoning as a Basis for Automatic Error Recovery in Robotics," Technical Report TR 85-24, Computer Science Department, University of Minnesota, August 1985.

[Hen84] T. Henderson, E. Shilčrat, "Logical Sensor Systems", Journal of Robotics, 1, 2, pp 169-193, 1984.

[Lee83] M. H. Lee, D. P. Barnes, N. W. Hardy, "Knowledge based error recovery in industrial robots", Proc. 8th International Joint Conference on Artificial Intelligence, August 1983, pp 824-826.

[Lop86] E. Lopez-Mellado, R. Alami, "An execution monitoring system for a flexible assembly workcell", Proc. 16th ISIR, September 1986, pp 955-962.

[Loz83] Lozano-Perez, T. "Robot Programming," Proc. of the IEEE, Vol 71, N. 7, July 1983, pp 821-841.

[Loz84] Lozano-Perez, T., Mason, M.T., Taylor, R.H, "Automatic Synthesis of Fine-Motion Strategies for Robots," The International Journal of Robotics Research, Vol 3, N 1, Spring 1984, pp 3-24.

[Luh83] Luh, J. Y. S., "An anatomy of industrial robots and their controls," IEEE Trans. on Automatic Control, Vol AC-28, N. 2, 1983.

[Muj79] Mujtaba, M.S. and Goldman, A., "AL users' Manual", Stanford Artificial Intelligence Laboratory Memo AIM-323, Stanford, Ca, January 1979.

[Smi86a] R. Smith, M. Gini, "Robot Tracking and Control Issues in an Intelligent Recovery System" Proc 1986 IEEE Conference on Robotics and Automation, pp 1070-1075, San Francisco, April 1986.

[Smi86b] R. Smith, M. Gini, "Reliable Real-time Robot Operation Employing Intelligent Forward Recovery," Journal of Robotic Systems, Summer 1986, pp 286-301.

[Sri76] Srinivas, S., "Error recovery in a robot system," PhD Thesis, CIT, 1976.

[Tay82] Taylor, R.H., Summers, P. D., Meyer, J. M., "AML: a manufacturing language," International Journal of Robotics Research, Vol 1, N. 3, 1982.

[Wil84] Wilkins, D., "Monitoring the execution of plans in SIPE", Computational Intelligence, Vol 1, 1985, pp 33 45.

S8-63

128

197177

Recovering From Execution Errors in SIPE

D.E. Wilkins
Stanford Research Institute International
Menlo Park, CA 94025

S0132

Note: *T'ᵢˢ paper originally appeared in Computational Intelligence, Vol. 1, No. 1, pp. 33-45, February 1985.*

Abstract

In real-world domains (e.g., a mobile robot environment), things do not always proceed as planned, so it is important to develop better execution-monitoring techniques and replanning capabilities. ~~This paper describes~~ these capabilities in the SIPE planning system. The motivation behind SIPE is to place enough limitations on the representation so that planning can be done efficiently, while retaining sufficient power to still be useful. This work assumes that new information given to the execution monitor is in the form of predicates, thus avoiding the difficult problem of how to generate these predicates from information provided by sensors.

The replanning module presented here takes advantage of the rich structure of SIPE plans and is intimately connected with the planner, which can be called as a subroutine. This allows the use of SIPE's capabilities to determine efficiently how unexpected events affect the plan being executed and, in many cases, to retain most of the original plan by making changes in it to avoid problems caused by these unexpected events. SIPE is also capable of shortening the original plan when serendipitous events occur. A general set of replanning actions is presented along with a general replanning capability that has been implemented by using these actions.

1 Introduction

A principal goal of our research in planning and plan execution is the development of a domain-independent, heuristic system that can plan an activity and then monitor the execution of that plan. Over the last two years we have designed and implemented such a system, called SIPE (System for Interactive Planning and Execution Monitoring).[1] The basic approach to planning is to work within the hierarchical-planning paradigm, representing plans in procedural networks – as has been done in NOAH [6] and other systems. Several extensions of previous planning systems have been implemented, including the development of a perspicuous formalism for describing operators and objects, the use of constraints for the partial description of objects, the creation of mechanisms that permit concurrent exploration of alternative plans, the incorporation of heuristics for reasoning about resources, and the creation of mechanisms that make it possible to perform deductions.

Given a description of the world and a set of operators that it can apply, SIPE can generate a plan to achieve a goal in the given world. (Operators are the system's description of actions that it may perform.) However, in real-world domains, things do not always proceed as planned. Therefore, it is desirable to develop better execution-monitoring techniques and better capabilities to replan when things do not go as expected. In complex domains it becomes increasingly important to use as much as possible of the old plan, rather than to start all over when things go wrong.

This paper describes the execution-monitoring and replanning abilities that have recently been incorporated into the SIPE system. The particular advantages that can be obtained by using the rich structure in our plan representation are shown, as well as more general problems. The environment of a mobile robot has been used as a motivating domain in the development of some of the abilities here, though the implementation has been carried out in a general, domain-independent manner. This document does not describe resources, constraints, plan generation, and oth-r features of SIPE, nor does it attempt to justify the basic assumptions underlying the system. The interested reader is referred to [10] for this.

The problem we are addressing is the following: given a plan, a world description, and some appropriate description of an unanticipated situation that occurs during execution of the plan, our task is to transform the plan, retaining as much of the old plan as is reasonable, into one that will still accomplish the original goal from the current situation. This process can be divided into four steps: (1) discovering or inputing information about the current situation; (2) determining the problems this causes in the plan, if any, (similarly, determining shortcuts that could be taken in the plan after unexpected but helpful events); (3) creating "fixes" that change the old plan, possibly by deleting part of it and inserting some newly created subplan; and (4) determining whether any changes effected by such fixes will conflict with remaining parts of the old plan. Steps 2 and 4, and possibly 3 as well, involve determining which aspects of a situation later parts of the plan depend upon. Part of this problem is an instance of the standard truth maintenance problem, and SIPE's solution is described in Section 4. In SIPE, Step 4 becomes part of Step 3, as only those fixes that are guaranteed to work are produced. In addition, serendipitous effects are used to shorten the original plan in certain cases.

The major contributions of the replanning module in SIPE result from taking advantage of the system's rich plan representation and from imbedding it within the planning system itself, rather than implementing it as an independent module. This provides a number of benefits, of which the most important follow: (1) the replanning module can exploit the efficient frame reasoning mechanisms in SIPE to discover problems and potential fixes quickly; (2) the deductive capabilities of SIPE are used to provide a reasonable solution to the truth maintenance problem described above; and (3) the planner can be called as a subroutine to solve problems after the replanning module has inserted new goals into the plan.

Another important contribution is the development of a general replanning capability (see Section 6) that has been implemented by using a general set of replanning actions. In general, recovery from an arbitrary error poses a difficult problem. Often very little of the existing plan can be reused. One can always fall back on solving the original problem in the new situation, ignoring the plan that was being executed. The re; anning part of SIPE, however,

[1] The research reported here is supported by Air Force Office of Scientific Research Contract F49620-79-C-0188.

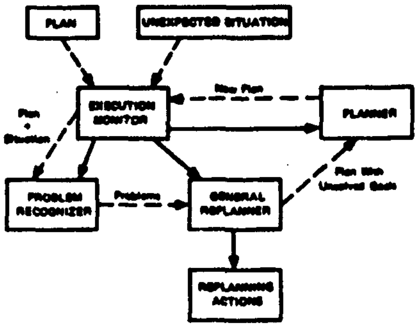


Figure 1: Control and Data Flow in SIPE Modules

tries to change the old plan, while retaining as much of it as possible. Since the problem is so difficult, one would not expect very impressive performanc from a general replanner such as SIPE's.

Better performance requires domain-specific information for dealing with errors. In many domains, the types of errors that are commonly encountered ca be predicted (e.g., the robot arm dropping something it was holding, or missing something it was trying to grasp). For this reason, the general replanne is based on a number of general replanning actions (i.e., actions that modify a plan in ways that are useful for handling unexpected situations) that can b referred to in a language for providing domain-specific error recovery instructions. Section 6 gives the outline of such a language.

1.1 Assumptions

SIPE assumes that information provided about unexpected events is correct and, to a certain extent, complete. This assumption avoids many of the hardest problems involved in getting a planner such as SIPE to control a mobile robot. The challenging task of determining how to generate correct predicate from information provided by the sensors is not addressed. We expect the translation of the information from the robot's sensors (e.g., the pixels from th camera or the range information from ultrasound) into the higher-level predicates used by the planner to be crucial in applying a SIPE-like planner to mobile robot. *We hope to deal with this problem in the near future.*

In a mobile robot domain, it may often be important to expend considerable effort in checking for things that might have gone wrong besides the unexpected occurrence already noticed. There is a substantial tradeoff involved here, as interpreting the visual input of unanticipated scenes may be expensive. Th research described in this paper does not examine this problem either. It assumes that nothing has gone wrong besides reported errors and effects that ca be deduced from them. The problem of uncertain or unreliable sensors or information is also largely unaddressed, except that some predicates and variable may be specified as unknown. What is discussed here is what to do with new information in the form of predicates (if we assume that such predicates hav somehow been discovered). Replanning appropriately with such information is an essential part of the overall solution.

1.2 Overview

Figure 1 shows the various modules in the SIPE execution-monitoring system. The solid arrows show which modules call which others. The broken arrow show the flow of data and information through the system as it replans for an unexpected situation. These arrows are labeled with a description of th data being passed.

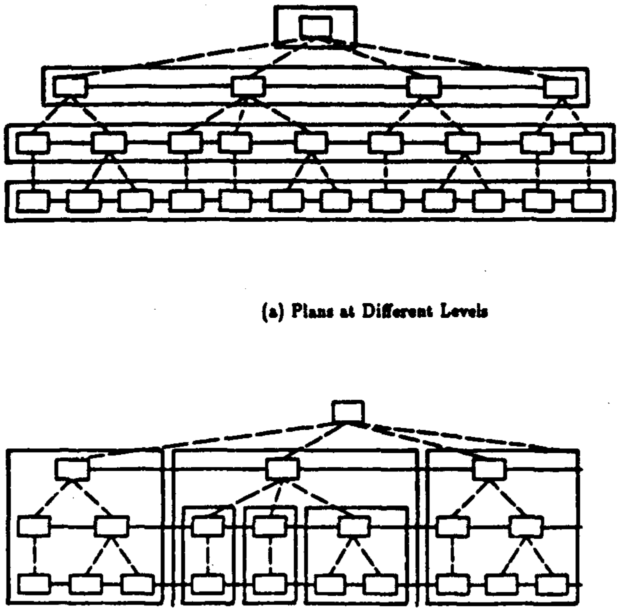
The general replanner is given the list of problems found by the problem recognizer and tries certain replanning actions in various cases, but will not alway find a solution. The general replanner changes the plan so that it will look like an unsolved problem to the standard planner in SIPE (e.g., by insertin new goals). After the replanner has dealt with all the problems that were found, the planner is called on the plan (which now includes unsolved goals). I it produces a new plan, this new plan should solve correctly all the problems that were found.

Section 2 of this paper describes the features of plan representation in SIPE that are relevant to its replanning capabilities. To describe unexpecte situations, a user (at present a human, but eventually this may be a program controlling and interpreting the robot's sensors) can enter arbitrary predicate at any point in the execution or can specify certain things as unknown. Section 3 describes the details of this process. Once the description of the unexpecte situation has been accumulated, the execution monitor calls a problem recognizer described in Section 4, which returns a list of all the problems it detect in the plan. The replanning actions are described in Section 5 and the general replanner in Section 6. Section 7 shows examples of the general replanne in operation.

2 Plans in SIPE

Plans in SIPE are represented as procedural networks [6], with temporal information encoded in the predecessor and successor links between nodes The plan rationale, of primary importance to the execution monitor, is encoded in the network by MAINSTEP links between nodes and by the use o PRECONDITION nodes (described below). MAINSTEP links describe how long each condition that has been achieved must be maintained. A contex must also be given to specify a plan completely, as the network contains choice points from which alternative plans branch. The types of nodes that occu in plans are described below to the extent necessary for understanding the execution-monitoring capabilities.

SPLIT and JOIN nodes provide for parallel actions. SPLITs have multiple successors and JOINs have multiple predecessors so that partially ordered plan can be produced. JOIN nodes have a *parallel-postcondition* slot, which specifies the predicates that must all be true in the situation represented by th JOIN node. If a JOIN node originally has N predecessors, there will be N conjunctions of predicates that must all be true at the JOIN node. (Som



(a) Plans at Different Levels

(b) Wedges Used by the Execution Monitor

Figure 2: SIPE Plan Viewed from Different Perspectives

branches may have been linearised, so there may be fewer than N predecessors after planning.) It is easier to record this at the JOIN node (than by having *previous nodes point to the JOIN as their purpose*), since a failed parallel postcondition can more easily be retried during execution monitoring if there is easy access to all parallel postconditions. The parallel postcondition slot is filled only when the JOIN is first introduced into the plan; it is not updated as more detailed levels of the hierarchy are expanded. As long as the highest level predicates are as desired, it is assumed that the lower-level predicates are irrelevant.

COND, ENDCOND, and CONDPATTERN nodes implement conditional plans. COND and ENDCOND are similar to SPLIT and JOIN, but each successor of the COND begins with a CONDPATTERN node that determines which successor will be executed.

CHOICE nodes denote branching points in the search space. They have multiple successors, but the context selects one of these as being in the current plan. Constraints on variables may be posted relative to this choice point. Thus, if the part of a plan after a CHOICE node is removed, the corresponding choice point in the context should also be removed so that constraints that are no longer valid will be ignored.

GOAL nodes do not occur in final plans, since they represent problems that have not yet been solved. A GOAL node specifies a predicate that must be achieved, but which is not true in the situation represented by its location in the procedural network. Replanning actions will insert GOAL nodes in the plan. Each GOAL node has a MAINSTEP slot, which denotes a point later in the plan that depends on the GOAL. (This describes the rationale for having the GOAL in the plan.) Each goal must be maintained as true until the node which is its MAINSTEP is executed. A MAINSTEP slot can have the atom PURPOSE as its value, denoting that the given predicate *is the main purpose of the plan, not preparation for some later action.*

PHANTOM nodes are similar to GOAL nodes except that they are already true in the situation represented by their location in the procedural network. They are part of the plan because their truth must be monitored as the plan is being executed. They also contain MAINSTEP slots.

PROCESS nodes represent actions to be performed during execution of the plan; they also have MAINSTEP slots, as do PHANTOM and GOAL nodes. In a final plan, all PROCESS nodes will denote primitive actions. (There are also CHOICEPROCESS nodes, which are like PROCESS nodes except that they have a list of actions, one of which must be performed.)

PRECONDITION nodes provide a list of predicates that must be true in the situation represented by their location in the procedural network. Operators may specify preconditions that must obtain in the world state before the operator can be applied. The concept of precondition here differs from its counterpart in some planners, since the system will make no effort to render the precondition true. A false precondition simply means that the operator is not appropriate. Conditions that the planner should make true (and therefore backward-chain on) can be expressed as goal or process nodes.

By distinguishing between PRECONDITIONS, GOALS and PROCESSES, we effectively encode metaknowledge about how to achieve goals. SIPE will use any means to solve a goal node, only the operators listed to solve a process node, but no operators to solve a PRECONDITION node. Thus, a precondition's becoming false does not mean that it should be made into a goal; rather it means that the part of the plan produced by the operator which initially inserted this precondition is invalid. PRECONDITION nodes also help encode the rationale of a plan, since in effect they mean that the part of the plan associated with them (see below) was produced on the assumption that the predicates in the precondition were true.

In addition to the "horizontal" MAINSTEP, predecessor, and successor links within *one level of a plan, there are* "vertical" links between different levels of the hierarchy. Each node that is expanded by the application of an operator has descendant links to each node so produced. The descendant nodes in turn have ancestor links back to the original node one level higher in the hierarchy. Starting with a node that was expanded by an operator application, a *wedge* of the plan is determined by following all its descendant links (in the current context) repeatedly (i.e., including descendants of descendants, and so on) to the lowest level. (This definition of wedges is the same as that used by Sacerdoti [6].) Figure 2 depicts this graphically, with the large boxes in

Part (b) representing wedges. The node originally expanded by an operator application is called the *top* of the wedge. A wedge with its top at a high level in the hierarchy will generally contain many lower-level wedges within itself. The only nodes that can be the tops of wedges are GOAL, PROCESS, and CHOICEPROCESS.

Since PRECONDITION nodes are created only when an operator is applied, the part of a plan associated with a PRECONDITION node can be found by ascending along the ancestor links to the point at which the precondition first became part of the plan (once inserted, PRECONDITION nodes are copied down from level to level). The node that was expanded by an operator to create this precondition is one level higher than where the first PRECONDITION node appears and is the top of the wedge associated with each of the PRECONDITION nodes that are copied from the first one.

3 The Input of Unexpected Situations

During execution of a plan in SIPE, some person or computer system monitoring the execution can specify what actions have been performed and what changes have occurred in the domain being modeled. SIPE changes its original world model permanently so as to show the effects of actions already performed. At any point during execution, the system will accept two types of information about the domain: (1) an arbitrary predicate whose arguments are ground instances, that is now true, false or unknown; and (2) a local variable name that is now unknown. SIPE first checks whether the truth-values for the new predicates differ from its expectations and, if they do, it applies its deductive operators to deduce more changed predicates.

It is important to note that the inputing of predicates does not solve the "pixels to predicates" problem, which is the crucial issue in using a planner such as SIPE to control the actions of a robot. This problem involves translating the input of the robot's sensors (e.g., the pixels from the camera or the range information from ultrasound) into the higher-level predicates used by the planner. The research described here is concerned with what must be done with the predicates once they have been established but does not take up the question of how to determine them automatically. We hope to address this latter task in the near future.

3.1 Unknowns

Unknowns are a new addition to SIPE, as it previously assumed complete knowledge of the world. Having unknown quantities constitutes a fundamental modification because even the method of determining whether a predicate is true must be changed. If the truth-values of critical predicates are unknown, the planner will quickly fail, since none of the operators will be applicable. (Neither a negated nor an unnegated predicate in a precondition will match an unknown one.) Operators can require predicates to be unknown as part of their precondition in case there are appropriate actions to take when things are uncertain. Conditional plans have also been implemented as part of the execution-monitoring package in SIPE; thus, an operator might produce a plan with an action to perceive the unknown value, followed by a conditional plan that specifies the correct course of action for each possible outcome of the perception action. The deductive capabilities have also been enhanced so that operators can deduce that something is unknown.

The ability to specify variables as unknown is simply a tool provided by the system that will presumably be useful in some domains, particularly in a mobile robot domain. The idea behind this tool is that the location of an object may become unknown during execution. Rather than make predicates unknown, which may cause the application of operators to fail, we simply say that the variable representing the location is instantiated to the atom UNKNOWN, rather than to its original location. All predicates with this variable as an argument may then still match as if they were true. Thus, the system can continue planning as if the location were known. The only restriction is that no action can be executed that uses an unknown variable as an argument. When such an action is to be executed (e.g., go to LOCATION1), then the actual instantiation of the variable must be determined before the action is executed (possibly through a perception action). Note that it would be incorrect to continue planning if the truth-values of important predicates depended on the instantiation of the location variable. It is the responsibility of the user not to use the unknown variable if predicates depend on the latter's value.

3.2 Interpreting the Input

The user need not report all predicates that have changed since many of these may be deduced by SIPE's deductive operators. The system's deductive power has been increased recently (see next section) so many effects can be deduced from certain critical predicates. SIPE does not check for additional unexpected predicates. Alternatively, we could decide on some basis (which would have to be provided as part of the domain-specific description) just how much effort to expend on perception actions to find out whether more than the minimum has gone wrong. For example, if we are told that (ON A B) is not true when we expected it to be, we might want to check to see if B is where we thought it was. As it is, SIPE will simply deduce that B is clear (if no other block is on B) and will not try to execute actions to make further checks with regard to the world. This latter procedure could be very expensive for a mobile robot in the absence of good domain-specific knowledge about what was worth checking.

There is a problem with unexpected effects in deciding how they interact with the effects of the action that was currently being executed (e.g., did they happen before, during, or after the expected effects?). Our solution to this problem is to assume that the action took place as expected and to simply insert a "Mother Nature" action after it that is presumed to bring about the unexpected effects (and things deduced from them). The system assumes that any effects of the action being executed that did not actually become true are either provided or can be deduced from the information provided. This solution interfaces cleanly and elegantly with the rest of the planner and avoids having to model the way in which the unexpected effects might interact with their expected counterparts.

4 . Finding Problems in a Plan

Having just inserted a MOTHER-NATURE node (MN node) in a plan being executed, SIPE must now determine how the effects of this node influence the remainder of the plan. There are two aspects to this: the first involves planning decisions that were based on the effects of this node, and the second involves deductions about the state of the world that were based on those effects. Section 4.1 describes the problem recogniser in SIPE, which finds all problems in the remainder of the plan that might be caused by the effects of the MN node. Because of the rich information content in the plan representation (including the plan rationale), there are only six problems that must be checked. As shown in Figure 1, the problems found by the problem recogniser are given to the general replanner. The problem recogniser also notices possible serendipitous effects.

The second aspect mentioned above involves solving the traditional truth maintenance problem. Many effects deduced later in the plan may no longer be true if they depended on predicates that are negated by the MN node. The validity of such deductions must be checked so that the remainder of the plan represents the state of the world accurately. Section 4.2 describes how SIPE solves this problem, correctly updating deductions later in the plan. Deductions that are changed may or may not cause problems that should be recognised by the problem recogniser. If such problems are generated, they will be found by the problem recogniser described in Section 4.1, since the deductions are correctly updated before the problem recogniser is called.

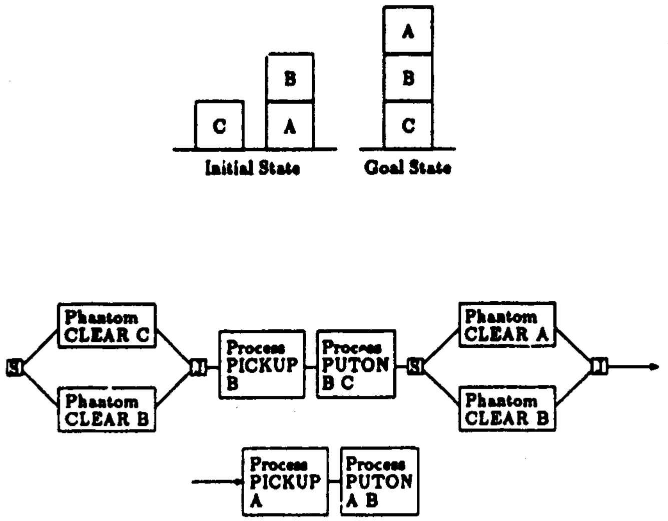


Figure 3: Blocks World Problem and Plan

4.1 Problems found by the problem recognizer

All occurrences of the six problems listed below are found by the problem recogniser. These problems constitute the only things that can go wrong with a plan in SIPE after addition of a MN node at the current execution point. The blocks-world problem in Figure 3 will be used to show an example of each type of problem.

1 - *Purpose not achieved.* If the MN node negates any of the main effects of the action just executed, there is a problem. The main effects must be reachieved. If during execution of the first PUTON node in the plan in Figure 3, either ¬(ON B C) or (ON B D) is given as an unexpected effect, then the MN node inserted after the PUTON node will negate the purpose of the PUTON node – thereby resulting in an instance of this type of problem.

2 - *Previous phantoms not maintained.* SIPE keeps a list of phantom nodes that occur before the current execution point (including those on parallel branches), and whose MAINSTEP slot specifies a point in the plan that has not yet been executed. These are phantoms that must be maintained. If the MN node negates any of these, then there is a problem. The phantoms that are no longer true must be reachieved. Suppose that during execution of the first PICKUP node in the plan in Figure 3, ¬(CLEAR C) is given as an unexpected effect. This type of problem will then occur, since the phantom node (CLEAR C) has a MAINSTEP slot (not shown in the figure) pointing to the first PUTON node, but has been negated by the MN node after the first PICKUP node.

3 - *Process node using unknown variable as argument.* If a variable has been declared as unknown, then the first action using it as an argument must be preceded by a perception action for determining the value of the variable (see Section 3). If the B in the plan were the instantiation of the variable BLOCK1 (instead of being given as part of the problem), and UNKNOWN BLOCK1 were entered during execution of the first PICKUP action, then this type of problem would occur with the immediately following PUTON action, since it would be applied to an UNKNOWN argument.

4 - *Future phantoms no longer true.* A phantom node after the current execution point may no longer be true. It must be changed to a GOAL node so that the planner will try to achieve it. In the sample plan, suppose that (ON D B) were given as an effect during execution of the first PUTON node. This type of problem would then occur with the last (CLEAR B) phantom node in the plan, since it would no longer be true when it is expected to be.

5 - *Future precondition no longer true.* A PRECONDITION node after the current execution point may no longer be true. In this case, we do not want to reachieve it, but rather pop up the hierarchy and perform some alternative action to achieve the goal at that level of the hierarchy. Because the sample plan contains no PRECONDITION nodes, we consider an example of this type in the travel planning domain. Suppose there is an operator for John's taking a taxi to the airport, which has a precondition that John's car is inoperative. If, during execution of the first part of the plan, SIPE is told that John's car is not broken, this type of problem will occur. In this case the reason for taking a taxi to the airport has been invalidated, and the general replanner will pop up the hierarchy and apply a different operator to get John to the airport (presumably driving his car).

6 - *Parallel postcondition not true.* All the parallel postconditions may no longer be true at a JOIN node. (This could be handled by maintaining phantoms, but is more convenient to handle separately.) In this case, we must insert a set of parallel goals after the JOIN, one for each untrue parallel postcondition. The parallel postconditions of the new JOIN will be the same as those on the old JOIN. In the sample plan, the last JOIN node will have both (ON A B) and (ON B C) as parallel postconditions (since they were in parallel originally). Suppose that (ON B TABLE) were given as an effect during the execution of the last PUTON node in the plan. This type of problem would then occur, since the parallel postcondition of (ON B C) would no longer be true.

Because of the way plans are encoded in SIPE, these are the only things that need to be checked when determining whether an MN node affects the remainder of a plan. This illustrates how the rich structure of plans in SIPE helps produce efficient problem detection. It should be noted, however, that processes (actions) are assumed to work whenever their precondition is true and when all phantoms whose MAINSTEP slot points to the process are true. (All such necessary conditions should be encoded as either preconditions or goals, in any case.) There is currently no check for loops caused by the same error happening repeatedly, with the same fix being proposed by the general replanner each time. Various simple checks could easily be added if this were a problem.

Finally, there are two important points to note with regard to the problem recogniser. First, in addition to the above problems, possible serendipitous effects are also noted and included in the list of problems. If the main effect of some action later in the plan is true before the action is executed, then that is noted as a possible place to shorten the plan (this is discussed in more detail in the next section). Second, only the last three problems above interact with the solution to the truth maintenance problem, since only they involve the truth-value of predicates in situations after the current execution point. The problem recogniser takes into account any changed deductions (see Section 4.2) while looking for the latter three problems.

4.2 Solution to the truth maintenance problem

SIPE's solution to the truth maintenance problem is based on the efficiency of its deductive capability. Since it is assumed that processes work as expected whenever their precondition is true and all phantoms whose MAINSTEP slot points to the process are true, only deduced effects need to be checked for their dependence on unexpected effects. (The execution monitor will solve problems having to do with preconditions and phantoms that are not true).

SIPE's deductive capability was designed to find a good balance between expressiveness and efficiency. While providing the power of many useful deductions, it nevertheless keeps deduction under control by severely restricting the deductions that can be made, as well as by having triggers to control the application of deductive operators. All deductions that can be made are performed at the time a node is inserted into the plan. Since deduction is not expensive, the truth maintenance problem is solved simply by redoing the deductions at each node in the plan after an MN node. Even this can be avoided in simple cases, because SIPE carries a list of changed predicates as it goes through the plan and, if they all become true later in the plan (without any deduced effects changing in the interim), then the execution monitor need not look at the remainder of the plan (either for redoing deductions or for finding problems).

5 Replanning Actions

The eight replanning actions described below, REINSTANTIATE, INSERT, INSERT-CONDITIONAL, RETRY, REDO, INSERT-PARALLEL, POP-REDO, and POP-REMOVE have all been implemeuted in SIPE. These actions provide sufficient power to alter plans in a way that often retains much of the original plan. These are domain-independent actions, and they form the basis of the general replanner. They should also prove useful as a basis for domain-specific error recovery operators. Both of these uses are described in more detail in Section 6. The first seven actions can all be used to solve problems found by the problem recognizer, while the last is used to take full advantage of serendipitous effects.

Four of the replanning actions change the plan so that it will contain unsolved problems. The intention (see Figure 1) is that the plan will then later be given to the normal planning module of SIPE (possibly after a number of these replanning actions have changed the plan). The planner will then attempt to find a solution that solves all the problems that have been corrected in the plan. The planner automatically checks to determine whether nodes it splices into the middle of the plan cause problems later, so that any solution found will be correct. (It does this when copying nodes down to the next lower level during planning.) In all actions described below, the context argument merely specifies the context of the current plan.

- *REINSTANTIATE (predicate node context)*

This action attempts to instantiate a variable differently so as to make the given predicate true in the situation specified by the given node. This appears to be a commonly useful replanning action. For example, it might correspond to using a different resource if something has gone wrong with the one originally employed in the plan, or deciding to return to the hopper for another screw rather than trying to find the one that has just been dropped.

An attempted reinstantiation is done by looping through the arguments of the given predicate. For each argument that is a planning variable (as opposed to an actual ground instance), SIPE checks to see if there is another instantiation for it that will make the predicate true. This is cheap and efficient in SIPE, since it merely involves removing the INSTAN constraint on the variable from the current context (and also from all variables constrained to be the same as this one), and then calling the normal matcher (which will return possible instantiations) to determine if the predicate is now true. Note that all other constraints that have been accumulated on this variable are left intact, so only instantiations that meet all relevant requirements are found.

If new instantiations are found, the REINSTANTIATE action checks the remainder of the plan to see if any parts of it might be affected by the new instantiation. This is done by a routine similar to the problem detector described in Section 4 (in fact, the two share much of their code). REINSTANTIATE currently accepts new instantiations only if they cause no new problems (see discussion below on trade-offs). If all new instantiations are rejected, the old INSTAN constraint is simply replaced. Note that replanning may be done later in the plan after the REINSTANTIATE action because of other problems that were found; the only requirement here is that the REINSTANTIATE action itself not introduce new problems later in the plan.

One might use REINSTANTIATE to help with the above mentioned problem of dropping a screw in the following way. Suppose that SCREW1 is a planning variable, while S1 and S2 are particular screws. The plan being executed could have SCREW1 instantiated to S1, a phantom to be maintained with the goal of (KNOWN-LOCATION SCREW1), and a PROCESS node for moving SCREW1 to achieve (AT SCREW1 WORKBENCH). During execution of the latter node, SIPE is told that the finger separation of the arm is zero. From this it could deduce (among other things) ¬(KNOWN-LOCATION SCREW1) and ¬(AT SCREW1 WORKBENCH). The problem of not achieving the purpose of the PROCESS node will result in an (AT SCREW1 WORKBENCH) goal being inserted in the plan. Without REINSTANTIATE, this would involve finding the location of S1 and moving it to the workbench - which may be a very hard problem (as anyone who has ever dropped a screw is aware). The problem of not maintaining the phantom node could trigger REINSTANTIATE on the predicate (KNOWN-LOCATION SCREW1), which would result in SCREW1 being reinstantiated to S2 (whose location is known). This would introduce no new problems and SIPE could proceed to solve the (AT SCREW1 WORKBENCH) goal by getting S2 from the hopper.

To prevent the introduction of a large search space, REINSTANTIATE is limited by the requirement that it not introduce new problems. There are also trade-offs in deciding when to apply REINSTANTIATE as it exists, but these are discussed later in the paper. The implementation described above opts for reinstantiation only when it is likely to be the correct solution. This is consistent with SIPE's running efficiently on the problems it does solve. Alternatively, new instantiations could be accepted even though they caused problems - as long as the latter are less severe than the problems incurred by keeping the old instantiation. Since SIPE has no way of comparing the difficulty of two sets of problems, REINSTANTIATE does not do this. Doing so would introduce another very large search space into the replanning process. However, it would not be difficult to change SIPE to explore this search space if a domain warranted it. There are also ways to partially lift this restriction at the cost of a moderately increased search space (though the tradeoffs involved probably depend on the domain).

One could also expend more effort in finding new instantiations. As implemented, this replanning action will find reinstantiations when only one variable is changed. Some problems could be solved by reinstantiating a whole set of variables, but this would be more expensive and involve a search problem to decide which variables to include in the set. The decision to try only one variable was made because it is efficient while evidently powerful enough to be useful. If the ability to reinstantiate sets of variables appeared useful, implementing it would certainly be tractable.

- *INSERT (node1 node2)*

This action inserts the subplan beginning with node1 (which has been constructed) into the current plan after node2. All links between the new subplan and the old plan are inserted correctly. This is used as a subroutine by many of the actions below.

- *INSERT-CONDITIONAL (variable node context)*

This action is not very interesting, but complements the unknown variable feature, which may be useful. It simply inserts a conditional around the given node that tests whether the given variable is known. If it is, the given node is executed next; otherwise a failure node is executed.

- ***RETRY (node)***

This replanning action is very simple. The given node is assumed to be a phantom node and it is changed to a goal node so that the planner will perceive it as unsolved.

- ***REDO (predicate node context)***

This action creates a GOAL node whose goal is the given predicate. It then calls INSERT to place this new node after the given node in the plan. The planner will see the new node as an unsolved goal.

- ***INSERT-PARALLEL (node predicates context)***

This action essentially does a REDO on each predicate in the list PREDICATES and puts the resulting GOAL nodes in parallel between a newly created SPLIT and JOIN. This subplan is inserted after NODE in the plan. The planner will see these new nodes as unsolved goals. This action is useful for reachieving parallel postconditions.

- ***POP-REDO (node predicates context)***

This and POP-REMOVE are the most complicated of the replanning actions; it is used to remove a hierarchical wedge from the plan and replace it with a node at the lowest level. POP-REDO is used when a PRECONDITION node is no longer true and another action must be applied at a higher level. It could also be used to find higher-level goals from which to replan when there are widespread problems causing the replanning to fail (this is not currently implemented).

When redoing a precondition failure, it is easy to determine the wedge to be removed, since PRECONDITION nodes are copied down from one level to another. The top of the wedge to be removed is the node that was expanded to initially place the given PRECONDITION node (or one of its ancestors that is a PRECONDITION node) in the plan. Actually, only the bottom of the wedge is spliced out of the plan, as planning will continue only from the lowest level. The subplan that is removed at the lowest level is replaced by a copy of the GOAL or CHOICEPROCESS node that was at the top of the wedge. (The INSERT replanning action is used for this.) This is seen as an unsolved goal by the planner, which automatically checks to ascertain whether expansions of this node cause problems later in the plan.

Let us consider the example mentioned earlier of John planning to take a taxi to the airport when his car is broken. The operator for taking the taxi could have a precondition ¬(HAS-CAR JOHN AUTO1) ∨ (BROKEN AUTO1). (This will match John's not having a car or his car being broken.) This operator is applied to solve the GOAL node (AT JOHN AIRPORT) at a high level in the plan, causing a PRECONDITION node for the above precondition to be inserted into the plan and copied down to all lower levels of the plan. Suppose that, during execution, ¬(BROKEN AUTO1) is entered as an unexpected effect during execution of a process before the PRECONDITION node. This node is a future PRECONDITION which becomes false, and the general replanner will apply POP-REDO to the problem. The wedge that is deleted has the GOAL node (AT JOHN AIRPORT) at the top. This may be a very large wedge if its lowest level is as detailed as "find the phone book, look up taxi in the yellow pages, dial a taxi company," etc. At the lowest level, the whole plan of finding a taxi and taking it to the airport is spliced out and replaced by an (AT JOHN AIRPORT) GOAL node. When SIPE's planner is later called on this plan, this GOAL node may be solved by John's driving his car to the airport.

There is one potentially serious complication in the above description of POP-REDO. Namely, various constraints may have been posted on the planning variables because of decisions made in the wedge of the plan that has been effectively removed. Fortunately, because of SIPE's use of alternative contexts, this is easily solved. A context is a list of choice points, and constraints are posted relative to the choice point that forced them to be posted. Therefore, this problem is solved by removing from the current context all the choice points that occurred in the wedge of the plan that was effectively removed. This new context is given as the context argument to future planning actions, and no further action need be taken. This results in ignoring precisely those constraints that should be ignored.

- ***POP-REMOVE (node predicates context)***

SIPE takes advantage of serendipitous effects to shorten a plan by using POP-REMOVE, which removes a wedge but does not insert a node. (It shares much of its code with POP-REDO.) However, in this case it is nontrivial to decide which wedge to remove. There may be various wedges that are candidates and, as with REINSTANTIATE, these candidates may cause problems later in the plan if they are removed. SIPE currently handles this case in the same way it handles REINSTANTIATE. Namely, it removes a wedge, checks to see if this causes any problems, and, if there are any, replaces the wedge. Thus, serendipitous effects are exploited only if doing so does not change the rest of the plan. This is a trade-off like the one discussed previously. SIPE again opts for efficiency, but could easily be changed to explore the additional search space of replanning after the removal of wedges.

SIPE also reduces the search space by generating only one candidate wedge. It gives up taking advantage of the serendipitous effect if this wedge does not work. The candidate wedge is generated by following ancestor links from the node given to POP-REMOVE (which supposedly has a purpose that has become true serendipitously), as long as some main effect of the candidate node is made true by one of the predicates in the list of given predicates (that have unexpectedly become true). The candidate node found in this manner determines the candidate wedge. The wedge is rejected immediately unless all its main effects are true in the given list of predicates.

Figure 4 uses the example of getting John to the airport to help illustrate this selection process. This example depicts a frequently occurring case in which the last action at one level of a wedge achieves the main effect of every level above that. For example, at Level 1 the goal is only to get John to the airport. At Level 2, after the choice has been made to take the taxi, the last node will achieve getting both John and the taxi to the airport. If Level 3 plans the mechanics of leaving the taxi, the last node there might contain all these higher-level effects as well as the thinner state of John's wallet.

The above selection process requires that all goals generated at a higher level and achieved in the candidate wedge be achieved before the wedge becomes a candidate, while goals generated at a lower level than the top of the candidate wedge need not have been achieved serendipitously. Thus, for Wedge 2 to be selected in Figure 4, the serendipitous effects must include (AT JOHN AIRPORT) from the higher level but need say nothing about how much cash John has since that is at a lower level. (It is assumed that, as long as the highest-level goal is achieved, we do not care about the lower-level goals that were necessary to bring this about.) The main effects of higher-level nodes that are achieved within a candidate wedge are easily checked because they are copied down as effects of the node that achieves them. Thus, checking to verify that all main effects of the candidate wedge are true ensures that all important higher-level effects will be true. In the example as shown, Wedge 2 can never be selected by this selection process because Wedge 1 will work whenever Wedge 2 does. However, in another example the effects of Wedge 1 might be achieved at Level 2 before Wedge 2 so that Wedge 2 might then be selected.

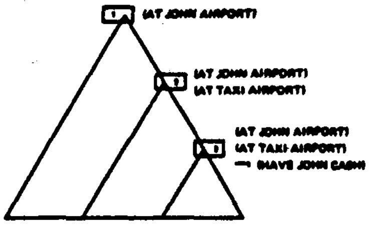


Figure 4: Hierarchical Wedges with a Common Last Action

6 Guiding the Replanning

The replanning actions of the preceding section form the basis for SIPE'S general replanning capability and for a language capable of specifying domain-specific error recovery instructions that has been designed but not implemented. The latter could be thought of as instructions for guiding the search of the general replanner. This section describes the automatic replanner and briefly outlines the error recovery operators.

6.1 The General Replanner

The general replanner takes a list of problems as well as possible serendipitous effects from the problem recogniser, and calls one or more of the replanning actions in an attempt to solve the problem. It first checks that a listed problem is still a problem, since the REINSTANTIATE action may solve many problems at once.

If the problem is a purpose that is not being achieved, the system tries a REDO, which inserts the unachieved purpose as a GOAL node after the MOTHER-NATURE node. If the problem is a previous phantom not being maintained, SIPE first tries REINSTANTIATE and, if that fails, it calls RETRY. The idea is that, if there is another object around with all the desired properties, it would be easier to use that object than to reachieve the desired state with the original object. If a PROCESS node has an unknown variable as an argument, INSERT-CONDITIONAL is called. If a future phantom is no longer true, RETRY is called. As with maintaining phantoms, REINSTANTIATE may be more appropriate, but, in both cases, this depends entirely on the domain; thus the selection here is arbitrary. For preconditions that are not true, the general replanner first calls REINSTANTIATE and, if that fails, calls POP-REDO. If parallel postconditions are not true, the general replanner calls INSERT-PARALLEL with the appropriate parallel goals.

While a general replanning capability is a significant achievement, one cannot expect very impressive performance from a replanner that does not have domain-specific information for dealing with errors. For example, whether or not REINSTANTIATE is likely to succeed will be dependent on the domain. The automatic replanner makes reasonable guesses at what might be a good choice in the domains on which SIPE has been tested. Since it merely chooses a replanning action for each type of problem that is found, it is very simple and could easily be rewritten for different domains.

6.2 Error Recovery Language

We also have designed an extension of the operator description language that enables instructions for handling foreseeable errors to be included in operators. This will allow encoding of domain-specific knowledge for guiding the search of the general replanner (or even avoiding the search altogether). The error recovery operators will have the same syntax as all other SIPE operators, with some new additions made to this language as described below. The plots of these operators will include references to the replanning actions in Section 5. SIPE's ability to specify conditional plans in operators can be used to try a second replanning action if the first one fails.

The error recovery operators will match their argument list to the arguments of the node being executed so that original problem variables can be bound to the variables in the operators. There will be two ways to invoke these operators: one for general operators that solve problems that have been recognised, and one for more specific operators that act directly on unexpected predicates. Both are described below.

The general operators will be applied after a MN node is added and problems have been found by the problem recogniser, but before the general replanner is called (see Figure 1). They will be applied to each of the problems in turn. Like deductive operators, error recovery operators will have a TRIGGER slot [10] to determine when they should be applied. The trigger will be a a combination of keywords and predicates, with the keywords referring to the six types of problems. These triggers will match when their keyword matches the problem being tried and any predicate in the trigger matches the appropriate predicates given in the problem. These operators may also have normal preconditions, which will be matched (in the normal manner) in the situation specified by the MN node. If any general error-recovery operator matches a given problem (i.e., both the trigger and precondition match), then the general replanner simply uses the instructions in the plot of the operator to choose the replanning actions to perform (rather than applying its own actions). Thus domain-specific guidance is supplied to the general replanner. It would be easy for such an operator, for example, to always force or prevent a REINSTANTIATE for a certain type of error under certain conditions.

Specific error-recovery operators are applied directly to predicates as they are inputted to the execution monitor before the problem recogniser is called. This should not be expensive because only operators mentioned in the ERROR slot of the action being executed are tried (see below). This ability seems attractive, since it can save a lot of effort when there is good domain-dependent error-recovery information available. When a specific operator matches an unexpected predicate, it may be possible in certain domains to simply apply the operator and assume that it will solve any problems caused by the unexpected predicate, thus circumventing the normal problem detection mechanism. If this option is chosen, SIPE simply assumes these error recovery operators are correct. Normal operation would involve checking for problems as usual after the application of specific operators.

Nodes in plots of regular operators will be able to specify an ERROR slot that gives names of specific error-recovery operators. When a node with an ERROR slot is being executed, the execution monitor will apply the operators listed in the error slot immediately to any unexpected predicate that is inputted during execution. Matching will be the same as for general operators, except that there are no keywords in the trigger. If one of these operators matches, the replanning actions in its plot will be carried out immediately. There may or may not be an option to preclude further problem detection on a predicate that has been so matched.

As an example of the intended use of a specific error recovery operator, consider the problem of dropping a screw that REINSTANTIATE solved in the general replanner. Suppose it is always desirable to return to the hopper after dropping a screw during the process of transporting it. The node for moving OBJECT1 to LOCATION1 would have an error slot that listed the DROP-SCREW operator. This specific error-recovery operator would match whenever the hand suddenly becomes empty and OBJECT1 is a screw. The *plot* of this operator could then specify a REINSTANTIATE of the variable OBJECT1 (which will still be constrained to be a screw) followed by an INSERT of a GOAL node to be achieved (AT OBJECT1 LOCATION1). Use of this operator could solve all problems of dropping a screw so that use of both the problem recogniser and general replanner could be avoided.

7 Examples

This section presents two simple examples of SIPE monitoring the execution of a simple plan, then replanning when things do not go as expected. SIPE has been tested on larger and more complex problems than those presented here. They involve a standard blocks world with ON and CLEAR predicates and a PUTON operator, as described in more detail elsewhere [10]. The user inputs only what is explicitly mentioned in boldface below; everything else is generated automatically by the system. This first problem was constructed to show the successful use of the REINSTANTIATE replanning action, and the second shows how the system inserts a newly created subplan during the replanning process.

Figure 5 shows the initial world state and the original problem. The problem is to get A on C in parallel with getting any blue block on any red block. In the initial world B1 and B2 are the only blue blocks (they are both on the table) and R1 and R2 are the only red blocks (R1 is on B1 and R2 is on the table and clear). Since A and C are both clear initially, SIPE quickly finds a two-action plan of putting A on C in parallel with putting B2 on R2, as shown in Figure 6.

This plan is then given to the execution monitor module of SIPE, which asks if P197 or P168 is to be executed first. The user types P197 and the system asks for unexpected effects. In this case the user types (ON D R2) followed by NIL to show one unexpected effect, namely D has suddenly appeared on top of R2. This creates a MN node after P197 which also has the following effects deduced by the system: ¬(ON D TABLE) ∧ ¬(CLEAR R2). The problem recogniser is called and it finds only one problem, namely the PHANTOM node P165 in the parallel branch was being maintained but is no longer true. This is given to the general replanner which first tries REINSTANTIATE. This succeeds as the OBJECT1 variable in the PHANTOM node an be rebound to R1 without causing any new problems in the plan. The plan in Figure 7 is passed from the planning module back to the execution monitor (without showing phantom nodes and mainstep slots). P168 is then executed without any unexpected effects and the goal is achieved. Note that the original plan was retained in its entirety and that B2 was placed on R1 instead of R2, thus achieving the original goal of getting A on C and any blue block on any red block.

The second problem is the same as the first, except that the variable REDBLOCK1 is constrained not to be R1 (by specifying IS NOT R1 in the original problem). This will cause REINSTANTIATE to be attempted but fail, since R2 is the only other red block. The original plan produced by SIPE is the same and the unexpected situation input by the user is the same. The problem recogniser again passes the same problem to the general replanner. This time SIPE tries REINSTANTIATE and fails, so it calls RETRY, which causes the (CLEAR R2) phantom in Figure 6 to be made into a goal. The planner solves this by producing a plan that puts D back on the table before B1 is placed on R2. The subplan shown in Figure 8 replaces the (CLEAR R2) phantom node, P165, on the parallel branch before the PUTON B2 R2 node in Figure 6. Without unexpected events, the plan so constructed then executes correctly to achieve the original goal. Alternatively, more unexpected occurrences could be given during execution of the newly constructed plan and SIPE would again go through a similar loop of finding and fixing problems until the original goal is achieved.

8 Comparison with Other Systems

There is very little previous work in this area, since most domain-independent planning systems do not address the problem of replanning. Two that do [3,6] *are discussed below. Tate's NONLIN [8], and Vere's DEVISER [9] do not concern themselves regarding execution.* PLANX10 [1] *lists "plan revision strategies"* as an area for future work, but does not appear to do replanning currently. McCalla and Schneider's ELMER [5] has a module called the "executor" and claims to take an integrated view of planning and execution. The executor adds more detail to the plan by simulating execution. For example, secondary plans are added in parallel with the original plan [4] to provide a demon-like capability for handling certain situations that may arise. This is not replanning, but rather a more detailed level of planning, albeit with complex planning operations. The executor effectively produces complex, conditional plans (with possibly complex parallel interactions) for situations it forsees. It does not accept arbitrary input during execution and then replan by changing the original plan as SIPE does. In fact, the authors mention that "to allow replanning after a plan goes awry" [5] is a future step in their research.

Sussman's HACKER [7] does modify plans (as do most planners that handle parallel actions), but does not deal with unexpected occurrences. HACKER produces plans that are not correct, then simulates them to detect errors. HACKER then solves some of these errors by using a few simple actions, such as *reordering parallel actions or reachieving subgoals that have not been maintained. Thus, the program is actually dealing with expected, not unexpected,* occurrences. SIPE generates correct plans to begin with, then modifies them on the basis of arbitrary unexpected occurrences. What HACKER does with regard to plan modification is analogous to what the critics do in the standard planning module of SIPE. While some of the problems found by such critics are similar to those found by the problem detector in SIPE (e.g., previous phantoms not being protected), they are only a subset. SIPE also provides a richer set of replanning actions for modifying plans.

The PLANEX system at SRI International [2] was used to monitor the execution of STRIPS plans that were represented in triangle tables. PLANEX does not do replanning because it never changes the sequence of actions in the plan. However, it does allow for a weak version of the REINSTANTIATE action in SIPE where a variable can be reinstantiated and the same plan restarted. Without SIPE's ability to post constraints on variables, this is less useful. PLANEX uses the triangle table representation to determine the latest point in the plan where execution could begin in the current situation (unexpected or expected), including both the executed and unexecuted portions of the plan in this calculation. If unexpected occurrences create a situation in which restarting the plan from some point other than the current execution point would solve the problem, PLANEX would do this. (Note this may involve redoing previous actions or skipping actions that had been planned.)

Although PLANEX can restart the original plan at a different point, this should not be construed as replanning. Moreover, it is not likely to be useful in a realistic domain. The world is not so benign as to frequently have unexpected occurrences produce situations in which ones's original plan is still applicable exactly as is from some point. With very high-level examples (as in [2]), this may occasionally happen, but it will happen only rarely with detailed plans. For example, an action such as "pick up block B (wherever it may be)" can simply be repeated when B is accidentally dropped and its new location is unknown. However, if the robot must plan to go to the location of B before picking it up, the original plan will be applicable only in the unlikely event that B is accidentally dropped onto its original location.

Hayes's system [3] and Sacerdoti's NOAH [6] *have* addressed the replanning problem. However, the approaches used in both these systems are considerably simpler and less powerful than that of SIPE. For example, NOAH does not allow the input of arbitrary predicates, so the general replanning problem never

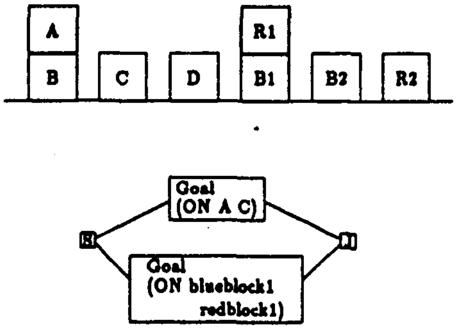


Figure 5: Initial Blocks World and Problem to be Solved

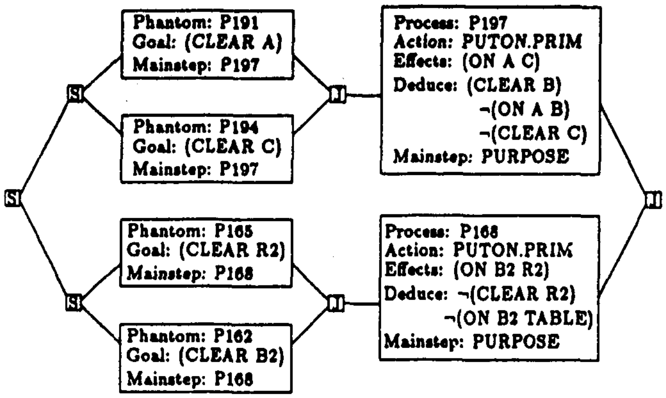


Figure 6: Initial Plan Produced by SIPE

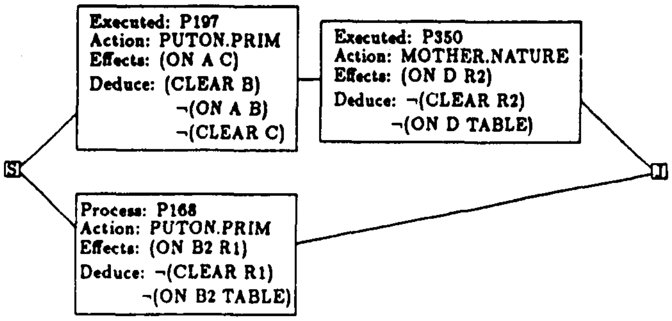


Figure 7: New Plan Produced for Continuing Execution

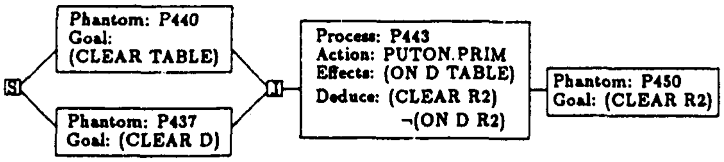


Figure 8: Subplan for Replacing PHANTOM P165

arises. It does permit the user to sperify that whole wedges had been executed at once, and allows a node that has just been executed to be planned for again if it fails. This essentially provides one limited replanning action that is useful only in very specific situations.

Hayes's system does allow the input of some information about unexpected situations. It is not clear what types of information can be provided, but they appear less general than the arbitrary predicates accepted by SIPE. The system's only replanning action is to delete part of the plan. This permits the planner to reachieve higher-level goals, but they must be the same higher-level goals that were already present in the plan. The system deletes everything that depended on *any effect of a decision* that is *no longer valid*. This will in general be wasteful, since much of the plan may be removed unnecessarily. If only one of the many effects of an action has failed, subplans depending on the unfailing effects do not need to be deleted. SIPE would keep such subplans in the plan (and find any problems that may have been generated within them).

SIPE provides a much more powerful replanning capability than either of these systems. It allows the input of arbitrary predicates, computes the extent to which these affect the rest of the plan, only finds complications that are really problematical, and uses a large number of replanning actions (including REINSTANTIATE) to remedy problems in ways that enable much of the original plan to be maintained.

9 Conclusion

9.1 Summary

Given correct information about unexpected events, SIPE is able to determine how this affects the plan being executed. In many cases, it is able to retain most of the original plan by making changes in it to avoid problems caused by these unexpected events. It is also capable of shortening the original plan when serendipitous events occur. It cannot solve difficult problems involving drastic changes to the expected state of the world, but it does handle many types of small errors that may crop up frequently in a mobile robot domain. The execution-monitoring package does this without the necessity of planning in advance to check for such errors.

The major contribution of this work is the development of a general set of replanning actions that are used as the basis of an automatic replanner, as well as the basis of a language for specifying domain-dependent error recovery information. These actions provide sufficient power to alter plans in a way that often retains much of the original plan, (e.g., the REINSTANTIATE action). The general replanner attempts to solve all problems that are found. It is unlikely to be very successful unless it is adapted to particular domains. The design of the language for error recovery operators allows both for operators that will handle very specific situations and for those that will give more general advice to the replanner.

The success of these mechanisms can largely be attributed to taking advantage of the rich structure of SIPE's planner and its plans. Often, the replanner calls the standard planning system as a subroutine. In this way it can take advantage of the efficient frame-reasoning mechanisms in SIPE to discover problems and potential fixes quickly, applying its deductive capabilities to provide a reasonable solution to the truth maintenance problem. The fixes suggested by the replanner need involve only the insertion of new goals into the plan, since calling the planner as a subroutine will solve these goals in a manner that assures there will be no conflicts with the rest of the plan. SIPE's execution-monitoring capabilities make extensive use of the explicit representation of plan rationale. The problem detector makes uses of the information encoded in MAINSTEP slots, phantoms, and preconditions to quickly find all the problems with a plan. Furthermore, it does not remove parts of the original plan unless the parts are actually problematical. The replanning actions make use of constraints, alternative contexts, and wedges in SIPE whenever they consider removing part of the plan.

9.2 Issues and Limitations

From the beginning, the rationale behind SIPE has been to place enough limitations on the representation so that planning can be done efficiently, while retaining enough power to still be useful. This motivation underlies most of the design decisions that have been made in implementing the system, including the design of the replanning module. For example, REINSTANTIATE and POP-REMOVE are limited to prevent the exploration of large search spaces. The use of SIPE's deductive capability to solve the truth maintenance problem also reflects our commitment to this design philosophy. The replanning capabilities have proved useful in two test domains.

The major limitations of this research stem from the assumption of correct information about unexpected events. This avoids many difficult problems, the most important of which is generating the high-level predicates used by SIPE from information provided by the sensors. This appears to be the most critical issue in getting a high-level planner such as SIPE to control a mobile robot. Part of the problem is heuristic adequacy – the robot cannot wait ten minutes for a vision module to turn pixels into predicates while the world is changing. Other questions that have not been discussed here are deciding how much effort to expend checking facts that may be suspect, and modeling uncertain or unreliable sensors. Finding solutions to these problems is of crucial importance to the task of endowing a mobile robot with execution-monitoring capabilities.

Acknowledgments

Many people influenced the ideas expressed in this paper. Special thanks go to Michael Georgeff for many enlightening discussions.

References

[1] Bresina, J., "An Interactive Planner that Creates a Structured, Annotated Trace of its Operation", Tech. Report CBM-TR-123, Department of Computer Science, Rutgers University, December 1981.

[2] Fikes, R., Hart, P., and Nilsson, N., "Learning and Executing Generalised Robot Plans", in *Readings in Artificial Intelligence*, Nilsson and Webber, eds., Tioga Publishing, Palo Alto, California, 1981, pp. 231-249.

[3] Hayes, Philip J., "A Representation for Robot Plans", *Proceedings IJCAI-75*, Tbilisi, USSR, 1975, pp. 181-188.

[4] McCalla, G., and Schneider, P., "The Execution of Plans in an Independent Dynamic Microworld", *Proceedings IJCAI-79*, Tokyo, Japan, 1979, pp. 553-555.

[5] McCalla, G., and Schneider, P., "Planning in a Dynamic Microworld", *Proceedings CSCSI Conference*, Saskatoon, Saskatchewan, 1982, pp. 248-255.

[6] Sacerdoti, E., *A Structure for Plans and Behavior*, Elsevier, North-Holland, New York, 1977.

[7] Sussman, G.J., *A Computer Model of Skill Acquisition*, Elsevier, North-Holland, New York, 1975.

[8] Tate, A., "Generating Project Networks", *Proceedings IJCAI-77*, Cambridge, Massachusetts, 1977, pp. 888-893.

[9] Vere, S., "Planning in Time: Windows and Durations for Activities and Goals", Jet Propulsion Lab, Pasadena, California, November 1981. (Editor's Note: AIAA Paper 83A43951)

[10] Wilkins, D., "Domain-independent Planning: Representation and Plan Generation", *Artificial Intelligence 22*, April 1984, pp. 269-301.

59-63
98

Incremental Planning to Control a Blackboard-Based Problem Solver

E.H. Durfee and V.R. Lesser
University of Massachusetts
Amherst, MA 01003

MK

presented

Abstract

To control problem solving activity, a planner must resolve uncertainty about which specific long-term goals (solutions) to pursue and about which sequences of actions will best achieve those goals. ~~In this paper, we describe~~ a planner that abstracts the problem solving state to recognize possible competing and compatible solutions and to roughly predict the importance and expense of developing these solutions. With this information, the planner plans sequences of problem solving activities that most efficiently resolve its uncertainty about which of the possible solutions to work toward. The planner only details actions for the near future because the results of these actions will influence how (and whether) a plan should be pursued. As problem solving proceeds, the planner adds new details to the plan incrementally, and monitors and repairs the plan to insure it achieves its goals whenever possible. Through experiments, ~~we~~ illustrate how these new mechanisms significantly improve problem solving decisions and reduce overall computation. ~~We~~ briefly discuss ~~our~~ current research directions, including how these mechanisms can improve a problem solver's real-time response and can enhance cooperation in a distributed problem solving network.

1. Introduction

A problem solver's planning component must resolve control uncertainty stemming from two principal sources. As in typical planners, it must resolve uncertainty about which sequence of actions will satisfy its long-term goals. Moreover, whereas most planners are given a (possibly prioritized) set of well-defined long-term goals, a problem solver's planner must often resolve uncertainty about the goals to achieve. For example, an interpretation problem solver that integrates large amounts of data into "good" overall interpretations must use its data to determine what specific long-term goals (overall interpretations) it should pursue. Because the set of possible interpretations may be intractably large, the problem solver uses the data to form promising partial interpretations and then extends these to converge on likely complete interpretations. The blackboard-based problem solving architecture developed in Hearsay-II permits such *data-directed* problem solving [1].

In a purely data-directed problem solver, control decisions can be based only on the desirability of the expected immediate results of each action. The Hearsay-II system developed an algorithm for measuring desirability of actions to better focus problem solving [2]. Extensions to the blackboard architecture unify data-directed and goal-directed control by representing possible extensions and refinements to partial solutions as explicit goals [3]. Through goal processing and subgoals, sequences of related actions can be triggered to achieve important goals. Further modifications separate control knowledge and decisions from problem solving activities, permitting the choice of problem solving actions to be influenced by strategic considerations [4]. However, none of these approaches develop and use a high-level view of the current problem solving situation so that the problem solver can recognize and work toward more specific long-term goals.

In this paper, we introduce new mechanisms that allow a blackboard-based problem solver to form such a high-level view. By abstracting its state, the problem solver can recognize possible competing and compatible interpretations, and can use the abstract view of the data to roughly predict the importance and expense of developing potential partial solutions. These mechanisms are much more flexible and complex than those we previously developed [5] and allow the recognition of relationships between distant as well as nearby areas in the solution space. We also present new mechanisms that use the high-level view to form plans to achieve long-term goals. A plan represents specific actions for the near future and more general actions for the distant future. By forming detailed plans only for the near future, the problem solver does not waste time planning for situations that may never

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8306327, by the National Science Foundation under Support and Maintenance Grant DCR-8318776, by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041.

arise; by sketching out the entire plan, details for the near-term can be based on a long-term view. As problem solving proceeds, the plan must be monitored (and repaired when necessary), and new actions for the near future are added *incrementally*. Thus, plan formation, monitoring, modification, and execution are interleaved [6,7,8,9,10].

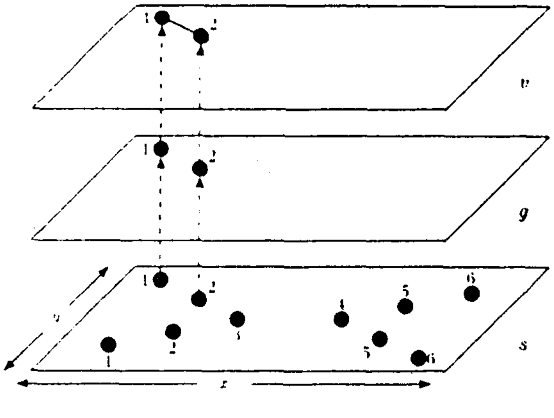
We have implemented and evaluated our new mechanisms in a vehicle monitoring problem solver, where they augment previously developed control mechanisms. In the next section, we briefly describe the vehicle monitoring problem solver. Section 3 provides details about how a high-level view is formed as an abstraction hierarchy. The representation of a plan and the techniques to form and dynamically modify plans are presented in Section 4. In Section 5, experimental results are discussed to illustrate the benefits and the costs of the new mechanisms. Finally, Section 6 recapitulates our approach and describes how the new mechanisms can improve real-time responsiveness and can lead to improved cooperation in a distributed problem solving network.

2. A Vehicle Monitoring Problem Solver

A vehicle monitoring problem solving *node*, as implemented in the Distributed Vehicle Monitoring Testbed (DVMT), applies simplified signal processing knowledge to acoustically sensed data in an attempt to identify, locate, and track patterns of vehicles moving through a two-dimensional space [11]. Each node has a blackboard-based problem solving architecture, with knowledge sources and levels of abstraction appropriate for vehicle monitoring. A *knowledge source* (KS) performs the basic problem solving tasks of extending and refining *hypotheses* (partial solutions). The architecture includes a goal blackboard and goal processing module, and through goal processing a node forms knowledge source instantiations (KSIs) that represent potential KS applications on specific hypotheses to satisfy certain goals. KSIs are prioritized based both on the estimated beliefs of the hypotheses each may produce and on the ratings of the goals each is expected to satisfy. The goal processing component also recognizes interactions between goals and adjusts their ratings appropriately; for example, subgoals of an important goal might have their ratings boosted. Goal processing can therefore alter KSI rankings to help focus the node's problem solving actions on achieving the subgoals of important goals [3].

A hypothesis is characterized by one or more *time-locations* (where the vehicle was at discrete sensed times), by an *event-class* (classifying the frequency or vehicle type), by a *belief* (the confidence in the accuracy of the hypothesis), and by a *blackboard-level* (depending on the amount of processing that has been done on the data). Synthesis KSs take one or more hypotheses at one blackboard-level and use event-class constraints to generate hypotheses at the next higher blackboard-level. Extension KSs take several hypotheses at a given blackboard-level and use constraints about allowable vehicle movements (maximum velocities and accelerations) to form hypotheses at the same blackboard-level that incorporate more time-locations.

For example, in Figure 1 each blackboard-level is represented as a surface with spatial dimensions x and y. At blackboard-level s (signal level) there are 10 hypotheses, each incorporating a single time-location (the time is indicated for each). Two of these hypotheses have been synthesized to blackboard-level g (group level). In turn, these hypotheses have been synthesized to blackboard-level v (vehicle level) where an extension KS has connected them into a single track hypothesis, indicated graphically by connecting the two locations. Problem solving proceeds from this point by having the goal processing component form goals (and subgoals) to extend this track to time 3 and instantiating KSIs to achieve these goals. The highest rated pending KSI is then invoked and triggers the appropriate KS to execute. New hypotheses are posted on the blackboard, causing further goal processing and the cycle repeats until an acceptable track incorporating data at each time is created. One of the potential solutions is indicated at blackboard-level v in Figure 1.



Blackboard-levels are represented as surfaces containing hypotheses (with associated sensed times). Hypotheses at higher blackboard-levels are synthesized from lower level data, and a potential solution is illustrated with a dotted track at blackboard-level v.

Figure 1: An Example Problem Solving State.

3. A High-level View for Planning and Control

Planning about how to solve a problem often requires viewing the problem from a different perspective. For example, a chemist generally develops a plan for deriving a new compound not by entering a laboratory and envisioning possible sequences of actions but by representing the problem with symbols and using these symbols to hypothesize possible derivation paths. By transforming the problem into this representation, the chemist can more easily sketch out possible solutions and spot reactions that lead nowhere, thereby improving the decisions about the actions to take in the laboratory.

A blackboard-based, vehicle monitoring problem solver requires the same capabilities. Transforming the node's problem solving state into a suitable representation for planning requires domain knowledge to recognize relationships—in particular, long-term relationships—in the data. This transformation is accomplished by incrementally clustering data into increasingly abstract groups based on the attributes of the data: the hypotheses can be clustered based on one attribute, the resulting clusters can be further clustered based on another attribute, and so on. The transformed representation is thus a hierarchy of clusters where higher-level clusters abstract the information of lower-level clusters. More or less detailed views of the problem solving situation are found by accessing the appropriate level of this abstraction hierarchy, and clusters at the same level are linked by their relationships (such as having adjacent time frames or blackboard-levels, or corresponding to nearby spatial regions).

We have implemented a set of knowledge-based clustering mechanisms for vehicle monitoring, each of which takes clusters at one level as input and forms output clusters at a new level. Each mechanism uses different domain-dependent relationships, including:

- **temporal relationships:** the output cluster combines any input clusters that represent data in adjacent time frames and that are spatially near enough to satisfy simple constraints about how far a vehicle can travel in one time unit.
- **spatial relationships:** the output cluster combines any input clusters that represent data for the same time frames and that are spatially near enough to represent sensor noise around a single vehicle.
- **blackboard-level relationships:** the output cluster combines any input clusters that represent the same data at different blackboard-levels.
- **event-class relationships:** the output cluster combines any input clusters that represent data corresponding to the same event-class (type of vehicle).
- **belief relationships:** the output cluster combines any input clusters that represent data with similar beliefs.

The abstraction hierarchy is formed by sequentially applying the clustering mechanisms. The order of application depends on the *bias* of the problem solver: since the order of clustering affects which relationships are most emphasized at the highest levels of the abstraction hierarchy, the problem solver should cluster to emphasize the relationships it expects to most significantly influence its control decisions. Issues in representing bias and modifying inappropriate bias are discussed elsewhere [12].

To illustrate clustering, consider the clustering sequence in Figure 2, which has been simplified by ignoring many cluster attributes such as event-classes, beliefs, volume of data, and amount of pending work; only a cluster's blackboard-levels (a cluster can incorporate more than one) and its time-regions (indicating a region rather than a specific location for a certain time) are discussed. Initially, the problem solving state is nearly identical to that in Figure 1, except that for each hypothesis in Figure 1 there are now two hypotheses at the same sensed time and slightly different locations. In Figure 2a, each cluster c_n^l (where l is the level in the abstraction hierarchy) corresponds to a single hypothesis, and the graphical representation of the clusters mirrors a representation of the hypotheses. By clustering based on blackboard-level, a second level of the abstraction hierarchy is formed with 19 clusters (Figure 2b). As is shown graphically, this clustering "collapses" the blackboard by combining clusters at the previous abstraction level that correspond to the same data at different blackboard-levels. In Figure 2c, clustering by spatial relationships forms 9 clusters. Clusters at the second abstraction level whose regions were close spatially for a given sensed time are combined into a single cluster. Finally, clustering by temporal relationships in Figure 2d combines any clusters at the third abstraction level that correspond to adjacent sensed times and whose regions satisfy weak vehicle velocity constraints.

The highest level clusters, as illustrated in Figure 2d, indicate four rough estimates about potential solutions: a vehicle moving through regions $R_1R_2R_3R_4R_5R_6$, through $R_1R_2R_3R_4R'_5R'_6$, through $R'_1R'_2R_3R_4R_5R_6$, or through $R'_1R'_2R_3R_4R'_5R'_6$. The problem solver could use this view to improve its control decisions about what short-term actions to pursue. For example, this view allows the problem solver to recognize that all potential solutions pass through R_3 at sensed time 3 and R_4 at sensed time 4. By boosting the ratings of KSIs in these regions, the problem solver can focus on building high-level results that are most likely to be part of any eventual solution.

In some respects, the formation of the abstraction hierarchy is akin to a rough pass at solving the problem, as indeed it must be if it is to indicate where the possible solutions may lie. However, abstraction differs from problem solving because it ignores many important constraints needed to solve the problem. Forming the abstraction hierarchy is thus much less computationally expensive than problem solving, and results in a representation that is too inexact as a problem solution but is suitable for control. For

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^1	$(1x_1y_1)(2x_2y_2)$	v	-
c_2^1	$(1x_1y_1)$	g	-
c_3^1	$(2x_2y_2)$	g	-
c_4^1	$(1x_1y_1)$	s	-
c_5^1	$(1x_1'y_1')$	s	-
.	.	.	.
.	.	.	.
c_{23}^1	$(6x_6'''y_6''')$	s	

(a)

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^2	$(1x_1y_1)(2x_2y_2)$	s, g, v	$c_1^1, c_2^1, c_3^1, c_4^1, c_8^1$
c_2^2	$(1x_1'y_1')$	s	c_5^1
c_3^2	$(2x_2'y_2')$	s	c_6^1
.	.	.	.
.	.	.	.
c_{19}^2	$(6x_6'''y_6''')$	s	c_{23}^1

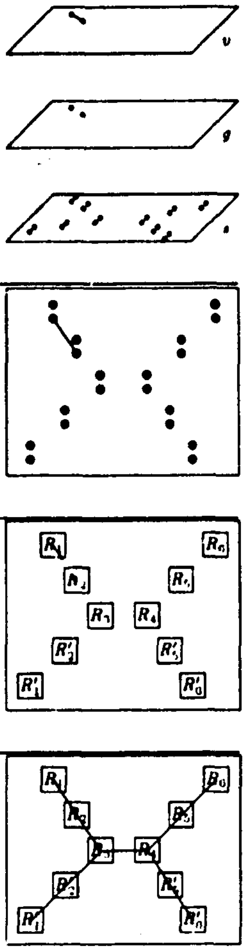
(b)

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^3	$(1R_1)(2R_2)$	s, g, v	c_1^2, c_2^2, c_3^2
c_2^3	$(1R_1')$	s	c_3^2, c_4^2
c_3^3	$(2R_2')$	s	c_6^2, c_7^2
.	.	.	.
.	.	.	.
c_9^3	$(6R_6')$	s	c_{18}^2, c_{19}^2

(c)

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^4	$(1R_1)(2R_2)(3R_3)$ $(4R_4)(5R_5)(6R_6)$	s, g, v	$c_1^3, c_4^3, c_5^3,$ c_6^3, c_8^3
.	.	.	.
.	.	.	.
c_4^4	$(1R_1')(2R_2')(3R_3)$ $(4R_4)(5R_5')(6R_6')$	s	$c_2^3, c_3^3, c_4^3,$ c_5^3, c_7^3, c_9^3

(d)



A sequence of clustering steps are illustrated both with tables (left) and graphically (right). c_i^l represents cluster i at level l of the abstraction hierarchy. In (a), each cluster is a hypothesis. These are clustered by blackboard-level to get (b); note that graphically the levels have been collapsed into one. These clusters are then grouped by spatial relationships to form (c), which in turn is clustered by temporal relationships to form (d).

Figure 2: An Example of Incremental Clustering.

example, although the high-level clusters in Figure 2d indicate that there are four potential solutions, three of these are actually impossible based on the more stringent constraints applied by the KSs. The high-level view afforded by the abstraction hierarchy therefore does not provide answers but only rough indications about the long-term promise of various areas of the solution space, and this additional knowledge can be employed by the problem solver to make better control decisions as it chooses its next task.

4. . Incremental Planning

The *planner* further improves control decisions by intelligently ordering the problem solving actions. Even with the high-level view, uncertainty remains about whether each long-term goal can actually be achieved, about whether an action that might contribute to achieving a long-term goal will actually do so (since long-term goals are inexact), and about how to most economically form a desired result (since the same result can often be derived in different ways). The planner reduces control uncertainty in two ways. First, it orders the intermediate goals for achieving long-term goals so that the results of working on earlier intermediate goals can diminish the uncertainty about how (and whether) to work on later intermediate goals. Second, the planner forms a detailed sequence of steps to achieve the next intermediate goal: it determines the least costly way to form a result to satisfy the goal. The planner thus sketches out long-term intentions as sequences of intermediate goals, and forms detailed plans about :he best way to achieve the next intermediate goal.

A long-term vehicle monitoring goal to generate a track consisting of several time-locations can be reduced into a series of intermediate goals, where each intermediate goal represents a desire to extend the track satisfying the previous intermediate goal into a new time-location.[1] To determine an order for pursuing the possible intermediate goals, the planner currently uses three domain-independent heuristics:

Heuristic-1 ***Prefer common intermediate goals.*** **Some intermediate goals may be common to several long-term goals. If uncertain about which of these long-term goals to pursue, the planner can postpone its decision by working on common intermediate goals and then can use these results to better distinguish between the long-term goals. This heuristic is a variation of least-commitment [13].**

Heuristic-2 ***Prefer less costly intermediate goals.*** Some intermediate goals may be more costly to achieve than others. The planner can quickly estimate the relative costs of developing results in different areas by comparing their corresponding clusters at a high level of the abstraction hierarchy: the number of event-classes and the spatial range of the data in a cluster roughly indicates how many potentially competing hypotheses might have to be produced. This heuristic causes the planner to develop results more quickly. If these results are creditable they provide predictive information, otherwise the planner can abandon the plan after a minimum of effort.

Heuristic-3 *Prefer discriminative intermediate goals.* When the planner must discriminate between possible long-term goals, it should prefer to work on intermediate goals that most effectively indicate the relative promise of each long-term goal. When no common intermediate goals remain, therefore, this heuristic triggers work in the areas where the long-term goals differ most.

These heuristics are interdependent. For example, common intermediate goals may also be more costly, as in one of the experiments described in the next section. The relative influence of each heuristic can be modified parametrically.

Having identified a sequence of intermediate goals to achieve one or more long-term goals, the planner can reduce its uncertainty about how to satisfy these intermediate goals by planning in more detail. If the planner possesses models of the KSs that roughly indicate both the costs of a particular action and the general characteristics of the output of that action (based on the characteristics of the input), then the planner can search for the best of the alternative ways to satisfy an intermediate goal. We have provided the planner for our vehicle monitoring problem solver with coarse KS models that allow it to make reasonable predictions about short sequences of actions to find the sequences that best achieve intermediate goals.[2] To reduce the effort spent on planning, the planner only forms detailed plans for the next intermediate goal: since the results of earlier intermediate goals influence decisions about how and whether to pursue subsequent intermediate goals, the planner avoids expending effort forming detailed plans that may never be used.

Given the abstraction hierarchy in Figure 2, the planner recognizes that achieving each of the four long-term goals (Figure 2d) entails intermediate goals of tracking the vehicle through these regions. Influenced predominantly by Heuristic-1, the planner decides to initially work toward all four long-term goals at the same time by achieving their common intermediate goals. A detailed sequence of actions to drive the data in R_3 at level s to level v is then formulated. The planner creates a plan whose attributes (and their values in this example) are:

- the long-term goals the plan contributes to achieving (in the example, there are four);
- the predicted, underspecified time-regions of the eventual solution (in the example, the time regions are $(1\ \ R_1 or R_1')(2\ \ R_2 or R_2')(3\ \ R_3)$ ...);
- the predicted vehicle type(s) of the eventual solution (in the example, there is only one type of vehicle considered);
- the order of intermediate goals (in the example, begin with sensed time 3, then time 4, and then work both backward to earlier times and forward to later times);
- the blackboard-level for tracking, depending on the available knowledge sources (in the example, this is level v);
- a record of past actions, updated as actions are taken (initially empty);
- a sequence of the specific actions to take in the short-term (in the example, the detailed plan is to drive data in region R_3 at level s to level v);
- a rating based on the number of long-term goals being worked on, the effort already invested in the plan, the average ratings of the KSIs corresponding to the detailed short-term actions, the average belief of the partial solutions previously formed by the plan, and the predicted beliefs of the partial solutions to be formed by the detailed activities.

[1] In general terms, an intermediate goal in any interpretation task is to process a new piece of information and to integrate it into the current partial interpretation.

[2] If the predicted cost of satisfying an intermediate goal deviates substantially from the crude estimate based on the abstract view, the ordering of the intermediate goals may need to be revised.

As each predicted action is consecutively pursued, the record of past actions is updated and the actual results of the action are compared with the general characteristics predicted by the planner. When these agree, the next action in the detailed short-term sequence is performed if there is one, otherwise the planner develops another detailed sequence for the next intermediate goal. In our example, after forming results in R_3 at a high blackboard-level, the planner forms a sequence of actions to do the same in R_4. When the actual and predicted results disagree (since the planner's models of the KSs may be inaccurate), the planner must modify the plan by introducing additional actions that can get the plan back on track. If no such actions exist, the plan is aborted and the next highest rated plan is pursued. If the planner exhausts its plans before forming a complete solution, it reforms the abstraction hierarchy (incorporating new information and/or clustering to stress different problem attributes) and attempts to find new plans. Throughout this paper, we assume for simplicity that no important new information arrives after the abstraction hierarchy is formed; when part of a more dynamic environment, the node will update its abstraction hierarchy and plans whenever such information becomes available.

The planner thus generates, monitors, and revises plans, and interleaves these activities with plan execution. In our example, the common intermediate goals are eventually satisfied and a separate plan must be formed for each of the alternative ways to proceed. After finding a partial track combining data from sensed times 3 and 4, the planner decides to extend this track backward to sensed time 2. The long-term goals indicate that work sho ıld be done in either R_2 or R'_2. A plan is generated for each of the two possibilities, and the more highly rated of these plans is followed. Note, however, that the partial track already developed can provide predictive information that, through goal processing, can increase the rating of work in one of these regions and not the other. In this case, constraints that limit a vehicle's turning rate are used when goal processing (subgoaling) to increase the ratings of KSI's in R'_2, thus making the plan to work there next more highly rated.[3]

The planner and goal processing thus work in tandem to improve problem solving performance. The goal processing uses a detailed view of local interactions between hypotheses, goals, and KSIs to differentiate between alternative actions. Goal processing can be computationally wasteful, however, when it is invoked based on strictly local criteria. Without the knowledge of long-term reasons for building a hypothesis, the problem solver simply forms goals to extend and refine the hypothesis in all possible ways. These goals are further processed (subgoaled) if they are at certain blackboard-levels, again regardless of any long-term justification for doing so. With its long-term view, the planner can drastically reduce the amount of goal processing. As it pursues, monitors, and repairs plans, the planner identifies areas where goals and subgoals could improve its decisions and selectively invokes goal processing to form only those goals that it needs. As the experimental results in the next section indicate, providing the planner with the ability to control goal processing can dramatically reduce control overhead.

In summary, we have developed mechanisms that permit incremental planning of problem solving activities in a blackboard-based problem solver. These mechanisms interleave planning and execution, monitoring plans and replanning when necessary. We base these mechanisms on having a high-level, long-term view of problem solving and on having acceptable models of problem solving actions. Furthermore, note that incremental planning may be inappropriate in domains where details about actions in the distant future can highly constrain the options in the near future. In these domains, constraints must be used to detail an entire plan before acting 13 . However, in unpredictable domains, incremental planning, plan monitoring, and plan repair are crucial to effective control since plans about the near future cannot depend on future states that may never arrive.

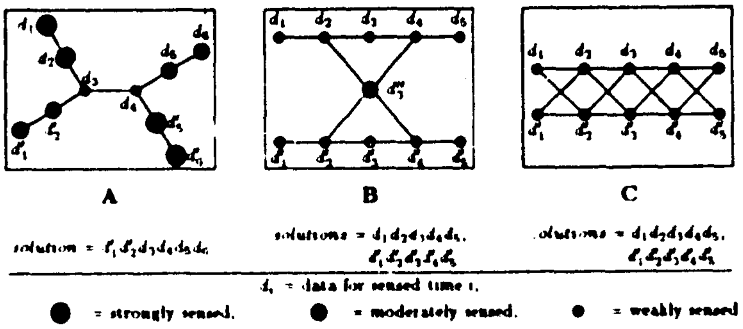
5. Experiments in Incremental Planning

We illustrate the advantages and the costs of our planner in several problem solving situations, shown in Figure 3. Situation A is the same as in Figure 2 except that each region only has one hypothesis. Also note that the data in the common regions is most weakly sensed. In situation B, no areas are common to all possible solutions, and issues in plan monitoring and repair are therefore stressed. Finally, situation C has many potential solutions, where each appears equally likely from a high-level view.

When evaluating the new mechanisms, we consider two important factors: how well do they improve control decisions (reduce the number of incorrect decisions), and how much additional overhead do they introduce to achieve this improvement. Since each control decision causes the invocation of a KSI, the first factor is measured by counting KSIs invoked—the fewer the KSIs, the better the control decisions. The second factor is measured as the actual computation time (runtime) required by a node to solve a problem, representing the combined costs of problem solving and control computation.

The experimental results are summarized in Table 1. To determine the effects of the new mechanisms, each problem situation was solved both with and without them, and for each case the number of KSIs and the computation time were measured. We also measured the number of goals generated during problem solving to illustrate how control overhead can be reduced by having the planner control the goal processing.

[3] In fact the turn to R_2 exceeds these constraints, as does the turn to R'_1, so that the only track that satisfies the constraints is $R'_1 R'_2 R_3 R_4 R_5 R_6$.


A
B
C
solution = $d'_1 d'_2 d_3 d_4 d_5 d_6$
solutions = $d_1 d_2 d_3 d_4 d_5$, $d'_1 d'_2 d'_3 d'_4 d'_5$
solutions = $d_1 d_2 d_3 d_4 d_5$, $d'_1 d'_2 d'_3 d'_4 d'_5$
d_i = data for sensed time i.
● = strongly sensed.
● = moderately sensed.
● = weakly sensed

Three problem solving situations are displayed The possible tracks (found in the abstraction hierarchy) are indicated by connecting the related data points, and the acceptable solution(s) for each situation are given

Figure 3: The Experimental Problem Situations.

Experiments E1 and E2 illustrate how the new mechanisms can dramatically reduce both the number of KSIs invoked and the computation time needed to solve the problem in situation A. Without these mechanisms (E1), the problem solver begins with the most highly sensed data (d_1, d_2, d'_5, and d'_6). This incorrect data actually corresponds to *noise* and may have been formed due to sensor errors or echoes in the sensed area. The problem solver attempts to combine this data through d_3 and d_4 but fails because of turning constraints, and then it uses the results from d_3 and d_4 to eventually work its way back out to the moderately sensed correct data. With the new mechanisms (E2), problem solving begins at d_3 and d_4 and, because the track formed ($d_3 d_4$) triggers goal processing to stimulate work on the moderate data, the solution is found much more quickly (in fact, in *optimal* time 14). The planner controls goal processing to generate and process only those goals that further the plan; if goal processing is done independently of the planner (E3), the overhead of the planner coupled with the only slightly diminished goal processing overhead (the number of goals is only modestly reduced, comparing E3 with E1) nullifies the computation time saved on actual problem solving. Moreover, because earlier, less constrained goals are subgoaled, control decisions deteriorate and more KSIs must be invoked.

The improvements in experiment E2 were due to the initial work done in the common areas d_3 and d_4 triggered by Heuristic-1. Situation A' is identical to situation A except that areas d_3 and d_4 contain numerous competing hypotheses. If the planner initially works in those areas (E5), then many KSIs are required to develop all of these hypotheses—fewer KSIs are invoked without planning at all (E4). However, by estimating the relative costs of the alternative intermediate goals, the planner can determine that d_3 and d_4, although twice as common as the other areas, are likely to be more than twice as costly to work on. Heuristic-2 overrides Heuristic-1, and a plan is formed to develop the other areas first and then use these results to more tightly control processing in d_3 and d_4. The number of KSIs and the computation time are thus reduced (E6).

Expt	Situ	Plan?	KSIs	Rtime	Goals	Comments
E1	A	no	58	17.2	262	-
E2	A	yes	24	8.1	49	-
E3	A	yes	32	19.4	203	independent goal processing and planning
E4	A'	no	58	19.9	284	noise in d_3, d_4
E5	A'	yes	64	17.3	112	noise in d_3, d_4, Heuristic-1 predominant
E6	A'	yes	38	16.5	71	noise in d_3, d_4, Heuristic-2 predominant
E7	B	no	73	21.4	371	-
E8	B	yes	45	11.8	60	-
E9	B	yes	45	20.6	257	independent goal processing and planning
E10	C	no	85	29.8	465	-
E11	C	yes	44	19.3	75	-

Legend

Situ:	The problem situation.
Plan?:	Are the new planning mechanisms used?
KSIs:	Number of KSIs invoked to find solution.
Rtime:	The total runtime (computation time) to find solution (*in minutes*).
Goals:	The number of goals formed and processed.
Comments:	Additional aspects of the experiment.

Table 1: A Summary of the Experimental Results.

In situation B, two solutions must be found, corresponding to two vehicles moving in parallel. Without the planner (E7), problem solving begins with the most strongly sensed data (the noise in the center of the area) and works outward from there. Only after many incorrect decisions to form short tracks that cannot be incorporated into longer solutions does the problem solver generate the two solutions. The high-level view of this situation, as provided by the abstraction hierarchy, allows the planner in experiment E8 to recognize six possible alternative solutions, four of which pass through d_3'' (the most common area). The planner initially forms $plan_1$, $plan_2$, and $plan_3$, beginning in d_3'', d_3, and d_3' respectively (Heuristic-1 triggers the preference for d_3'', and subsequently Heuristic-3 indicates a preference for d_3 and d_3'). Since it covers the most long-term goals, $plan_1$ is pursued first—a reasonable strategy because effort is expended on the solution path if the plan succeeds, and if the plan fails then the largest possible number of candidate solutions are eliminated. After developing d_3'', $plan_1$ is divided into two plans to combine this data with either d_2 or d_2'. One of these equally rated plans is chosen arbitrarily and forms the track d_2d_3'', which then must be combined with d_1. However, because of vehicle turning constraints, only d_1d_2 rather than $d_1d_2d_3''$ is formed. The plan monitor flags an error, an attempt to repair the plan fails, and the plan aborts. Similarly, the plan to form $d_1'd_2'd_3''$ eventually aborts. $Plan_2$ is then invoked, and after developing d_3 it finds that d_2 has already been developed (by the first aborted plan). However, the plan monitor detects that the predicted result, d_2d_3 was not formed, and the plan is repaired by inserting a new action that takes advantage of the previous formation of d_1d_2 to generate $d_1d_2d_3$. The predictions are then more than satisfied, and the plan continues until a solution is formed. The plan to form the other solution is s .. rly successfully completed. Finally, note once again that, if the planner does not control goal processing (E9), unnecessary overhead costs are incurred, although this time the control decisions (KSIs) are not degraded.

Situation C also represents two vehicles moving in parallel, but this time they are closer and the data points are all equally well sensed. Without the new mechanisms (E10), control decisions in this situation have little to go on: from a local perspective, one area looks as good as another. The problem solver thus develops the data points in parallel, then forms all tracks between pairs of points, then combines these into larger tracks, until finally it forms the two solution tracks. The planner uses the possible solutions from the abstraction hierarchy to focus on generating longer tracks sooner, and by monitoring its actions to extend its tracks, the planner more quickly recognizes failed extensions and redirects processing toward more promising extensions. The new mechanisms thus improve control decisions (reduce the KSIs) without adding excessive computational overhead (E11). However, the planner must consider 32 possible solutions in this case and does incur significant overhead. For complex situations, additional control mechanisms may be needed by the planner to more flexibly manage the large numbers of possibilities.

6. The Implications of Abstraction and Planning

We have described and evaluated mechanisms for improving control decisions in a blackboard-based vehicle monitoring problem solver. Our approach is to develop an abstract view of the current problem solving situation and to use this view to better predict both the long-term significance and cost of alternative actions. By recognizing and planning to achieve long-term goals, problem solving is more focused. By using the abstraction hierarchy when making planning decisions, problem solving can be more cost effective. Finally, by interleaving plan generation, monitoring, and repair with plan execution, the mechanisms lead to more versatile planning, where actions to achieve the system's (problem solving) goals and actions to satisfy the planner's needs (resolve its own uncertainty) are integrated into a single plan.

This approach can be generally applied to blackboard-based problem solvers. Abstraction requires exploiting relationships in the data – relationships that are used by the knowledge sources as well—such as allowable combinations of speech sounds [1] or how various errands are related spatially or temporally [4].[4] Planning requires simple models of KSs, recognition of intermediate goals (to extend a phrase in speech, to add another errand to a plan), and heuristics to order the intermediate goals. We believe that many if not all blackboard-based problem solvers (and more generally, problem solvers whose long-term goals depend on their current situation) could incorporate similar abstraction and planning mechanisms to improve their control decisions.

The benefits of this approach extend beyond the examples demonstrated in this paper. For example, goal satisfaction and problem solving termination are important issues in blackboard-based problem solvers. Given its underspecified goals of forming "good" solutions with its input, how does the problem solver recognize when it has found such solutions or when it can improve a solution? The more global view of the problem provided by the abstraction hierarchy helps the problem solver discover areas where improvements are possible and potentially worthwhile. The enumeration of possible solutions, and the success or failure to achieve them, similarly improves the problem solver's ability to determine when a solution is the best of the possible alternatives.

These mechanisms also help a problem solver to make informed decisions about how best to solve a problem under real-time constraints. The KS models provide estimates of the cost (in time) of possible activities so that the amount of time to achieve the next intermediate goal can be predicted. By exploiting the similarities between intermediate goals, moreover, these predictions can be generalized over all intermediate goals (making allowances for more or less costly areas as indicated by the abstraction hierarchy) and the time needs for the entire plan can be predicted. With this prediction, the planner can modify the plan (eliminate actions that unnecessarily increase the belief in a hypothesis, replace expensive actions with actions that inexpensively achieve less exact results) until the predicted time costs satisfy the time constraints.

[4] In fact, the WORD-SEQ knowledge source in the Hearsay-II speech understanding system essentially is a clustering mechanism: by applying weak grammatical constraints about pairwise sequences of words, WORD-SEQ generated approximate word sequences solely to *control* the application of the more expensive PARSE KS that applied full grammatical constraints about sequences of arbitrary length [1]

Finally, planning and prediction are vital to cooperation among problem solvers. A network of such problem solvers that are cooperatively solving a single problem could communicate about their plans, indicating what partial solutions they expect to generate and when. With this information, each problem solver can coordinate its activities with the others to generate and exchange useful results more efficiently, thereby improving network problem solving performance [12,14,5]. In essence, the problem solvers together form a distributed plan. The use of incremental planning, plan monitoring, and plan repair is particularly appropriate in such domains due to the inherent unpredictability of future actions and interactions.

The mechanisms we have outlined in this paper provide the basis for these possibilities. We are currently augmenting the mechanisms with capabilities to perform in more complex, dynamic environments: to modify the abstraction hierarchy when important unexpected situations arise and to model and plan for potential future situations (the arrival of more data to be processed, the actions of other problem solvers). Our new mechanisms, though they address issues previously neglected, should be integrated with other control techniques to be fully flexible, as seen in experiment E11. The combination of our mechanisms and goal processing has proved fruitful, and we believe that our mechanisms could similarly benefit by being integrated with other control approaches such as a blackboard architecture for control [4]. Based on the results we have outlined in this paper, we anticipate that the further development of mechanisms for developing abstract views and for incremental planning to control blackboard-based problem solvers will greatly enhance the performance of these problem solving systems, will lead to improved real-time response and to better coordination in distributed problem solving networks, and will increase our understanding of planning and action in highly uncertain domains.

References

[1] Lee D Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213-253, June 1980.

[2] Frederick Hayes-Roth and Victor R. Lesser Focus of attention in the Hearsay-II speech understanding system. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 27-35, August 1977.

[3] Daniel D Corkill, Victor R. Lesser, and Eva Hudlicka. Unifying data-directed and goal-directed control: an example and experiments. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 143-147, August 1982.

[4] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251-321, 1985.

[5] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Increasing coherence in a distributed problem solving network. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1025–1030, August 1985.

[6] R. T. Chien and S. Weissman. Planning and execution in incompletely specified environments. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 169–174, August 1975.

[7] Randall Davis *A model for planning in a multi-agent environment: steps toward principles of teamwork*. Technical Report MIT AI Working Paper 217, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Massachusetts, June 1981.

[8] Jerome A. Feldman and Robert F Sproull. Decision theory and artificial intelligence II: the hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[9] Gordon I. McCalla, Larry Reid, and Peter F Schneider. Plan creation, plan execution, and knowledge acquisition in a dynamic microworld. *International Journal of Man-Machine Studies*, 16:89–112, 1982.

[10] Earl D Sacerdoti. Problem solving tactics. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 1077–1085, August 1979.

[11] Victor R. Lesser and Daniel D Corkill. The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15–33, Fall 1983.

[12] Edmund H. Durfee *An Approach to Cooperation: Planning and Communication in a Distributed Problem Solving Network*. Technical Report 86-09, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, March 1986.

[13] Mark Stefik. Planning with constraints. *Artificial Intelligence*, 16:111–140, 1981.

[14] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. *Coherent Cooperation Among Communicating Problem Solvers*. Technical Report 85-15. Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, April 1985. Also to appear in *IEEE Transactions on Computers*.

Knowledge Representation System for Assembly Using Robots

A. Jain and M. Donath
University of Minnesota
Minneapolis, MN 55455

1.0 Introduction

Assembly robots combine the benefits of speed and accuracy with the capability of adaptation to changes in the work environment. However, an impediment to the use of robots is the complexity of the man-machine interface. This interface can be improved by providing a means of using apriori knowledge and reasoning capabilities for controlling and monitoring the tasks performed by robots.

Robots ought to be able to perform complex assembly tasks with the help of only supervisory guidance from human operators. For such supervisory guidance, it is important to express the commands in terms of the effects desired, rather than in terms of the motion the robot must undertake in order to achieve these effects.

A suitable knowledge representation can facilitate the conversion of task level descriptions into explicit instructions to the robot. Such a system would use symbolic relationships describing the apriori information about the robot, its environment, and the tasks specified by the operator to generate the commands for the robot.

However, the knowledge representation system should provide a scheme for building assembly models in an environment that maintains the knowledge of the spatial and functional relationships among the engineering parts comprising an assembly. The system then prevents the user from violating these relationships unless specifically asked to do so. Thus, the knowledge representation system has a memory for specified constraints and prevents actions whose side effects cause the violation of these constraints.

The modeling method provides an organizational structure that maps the relationships between the components of an assembly into a set of constraints. These constraints are simplified using a number of rules to determine the available degrees of freedom for a component. When a new relationship is to be established, the required motion is calculated and the feasibility of this motion is determined by checking against the available degrees of freedom.

2.0 Task Level Programming

Task level programming is an attempt to make use of structured programming together with an information base that enables the use of apriori knowledge to interpret the commands issued by the programmer. Examples of task-oriented programming languages include AUTOPASS, RAPT, and LAMA.

The processes are expressed in terms of the desired effects on the objects. Thus, a task is completed not when a manipulator has completed a series of motions, but when the objects have reached the desired configuration. The knowledge representation system converts the task level specifications into manipulator level commands.

AUTOPASS (1) accepts the steps of assembly as input from the user and with the help of a model of the world, converts these into a program that a manipulator can process. However, the system does not recognize changes in relationships occurring due to manipulator actions. The user is responsible for specifying physically realizable operations and also for informing the system of changes in the attachment relationships.

RAPT (2), developed at Edinburgh, has concentrated on specifying the spatial relationships and reasoning about them. The relationships between objects such as "B1 against B2" and "C1 against C2", etc., are converted into algebraic equations. These equations can be expressed in terms of operational parameters, such as joint angles of the manipulator or in terms of the degrees of freedom for the objects. The solution of these simultaneous non-linear equations is quite complicated and time consuming. However, some efforts at recognizing stereotypes and applying standard solutions have been successful. The emphasis here has been on representation of relationships; however, the geometrical representation is at present incomplete. Hence, path planning, collision detection, etc., cannot be implemented using this system alone.

LAMA (3) uses general program plans that are expanded by details of individual assemblies. Polyhedral representation for objects permits trajectory planning and collision detection. The constraints are expressed in terms of parameters that are unspecified elements of partially filled transformation matrices. The constraints are represented as constraint planes, which indicate the boundaries of permissible motion. Ranges of parameters are calculated using volume interactions and constraint plane interactions. The assumption is made that

Cont. on p. 113 →

different constraints on an object are non-interfering. This assumption is not required in the system we propose, as the bodies always move within the specified constraints and inconsistent constraints cannot be established in a constraint enforcing environment.

LM (4) and Feature Descriptor (5) represent other examples of task level programming.

Fahlman (6) proposed a task planner for robot construction tasks in a blocks' world domain. The information contained in the object's motion was not used. The feasibility of the final state is determined from a stability viewpoint, but the motion from the initial to the final state is not considered. A task planner built in a knowledge base that considers the feasibility of object motion can be integrated with a trajectory planning system.

In the proposed knowledge representation system, a constraint enforcing environment is provided at the level of the database. This approach permits supporting a task planner as well as interactive sessions with the programmer. This approach varies from the traditional approach by transferring some supervisory capability to the computer. The feasibility of motion in a constraint enforcing environment is determined by mapping relationships into a constraint set, simplifying the constraint, and interpreting the resulting constraints.

3.0 Geometrical Model

Homogeneous transformations are used to express the position and orientation information for objects. The position and orientation with respect to the world reference frame is stored with each object. Hence, this information is updated only when the object moves. The primitives, e.g., blocks and cylinders, are defined in terms of the centroid position and faces. Loci definitions are used as they are sufficient for the reasoning involved in constraint simplification as discussed later.

An assembly is defined recursively as an object consisting of other objects. By imposing the restriction that these primitives cannot move relative to one another, we can create rigid assemblies. We can represent mechanisms by permitting specified relative degrees of freedom for the primitives that combine to form the assembly.

4.0 Relationships

The objects can be related to one another in a variety of ways with regard to relative motion. These relationships are shown in Figure 1. The contact relationships involve both concave and convex surfaces. A contact relationship between two convex surfaces is called "against". The against relationship can involve a surface, edge, or a point contact between two bodies. A contact relationship between a concave and a convex surface is called a "fit".

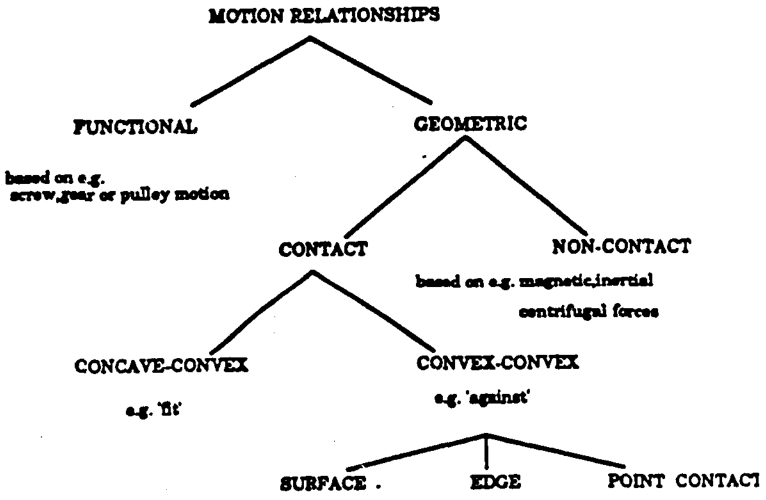


Figure 1: Motion Relationships

4.1 The "Against" Relationship

We will first define "against" in a restrictive sense and then generalize the definition. "Against" is a relationship established between two plane surfaces, such that translation is permitted in the common plane and rotation is permitted about an axis normal to this common plane. An example of against relationship is shown in Figure 2. This "against" relationship exists between face F1 of block B1 and face F2 of Block B2.

This definition of "against" can then be extended to curved convex surfaces, where the contact area reduces from the plane to a line. The object with the curved surface retains any rotation capabilities that it had about

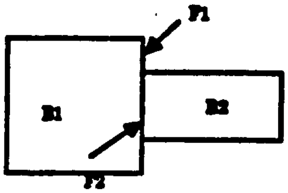


Figure 2: B1 Against B2 (F1, F2)

the body axis normal to the curved cross-section, prior to establishing the "against" relationship. Similar definition applies for spherical surfaces where the contact reduces to a point.

4.2 The "Fit" Relationship

"Fit" is a relationship established between a concave surface such as a hole and a convex surface such as a shaft. "Insert" is the driver function to command the establishment of a "fit" relationship. The hole and shaft can be of various cross-sections. The pre-condition for establishment of a "fit" relationship is that the profiles of the convex and the concave surfaces must match.

In a "fit" relationship, the body containing the hole and the body containing the shaft are restrained to rotate about the hole-shaft axis if the profiles are circular. A translation along this axis is also permitted.

Many special cases of "fit" can then be defined. A restricted "fit" relation might permit only rotational freedom. A rectangular key, on the other hand, is permitted translation but no rotation.

Representation of other relationships such as threaded joints, etc., can be accomplished by using these basic relationships. Permissable translation and rotation would no longer be independent of each other, but would be related by parameters such as the thread pitch.

The contact relationships of "against" and "fit" and other functional relationships in the motion domain map into a set of motion constraints that represent the degrees of freedom for the objects. These constraints are represented in our system in terms of new primitives which are necessary for reasoning about motion. These new constraint relationships represent a general set into which all the relationships that pertain to motion can be mapped. An example of this mapping is shown in Figure 3.

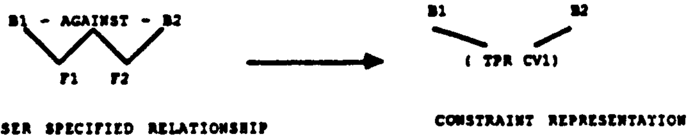


Figure 3: Mapping of Relationships to Constraints

5.0 Drivers

The driver functions that command the motion execution are of two types. The first type is the "move" function. The "move" function is used when some explicit motion is to be implemented by specifying the amount of motion or by specifying the final destination. The second type of drive functions are those which command the establishment of relationships. These functions determine the motion required to establish the relationship, find out if the motion is possible, and, if so, then send out the necessary instructions to the robot to implement the motion. For the two contact relationships, "against" and "fit", the driver functions are respectively called "establish-against" and "insert".

Determination of motion feasibility involves a number of steps. This is shown in Figure 4. The relationships of the body with other bodies, represented as constraints, are simplified to determine the translational and rotational degrees of freedom. If the entire desired motion is not possible with the available degrees of freedom, then an attempt is made to find one or more neighboring bodies such that this set of bodies can together accomplish the remaining motion. This strategy is explained in Section 8.

6.0 Representation of Motion Constraints

We need a representation that is sufficient for the reasoning involved in object manipulation, and is also simple and efficient to manipulate. We have departed from the conventional manner (Popplestone, Taylor, Mazer, etc.) of converting all the constraints into parametric equations and solving them over all the objects under consideration.

Let us define a few symbols first:

T: represents a Translation degree of freedom
R: represents a Rotation degree of freedom
L: stands for a Line. This is used as a mnemonic for the axis of rotation or a line of translation.
P: stands for a Plane.

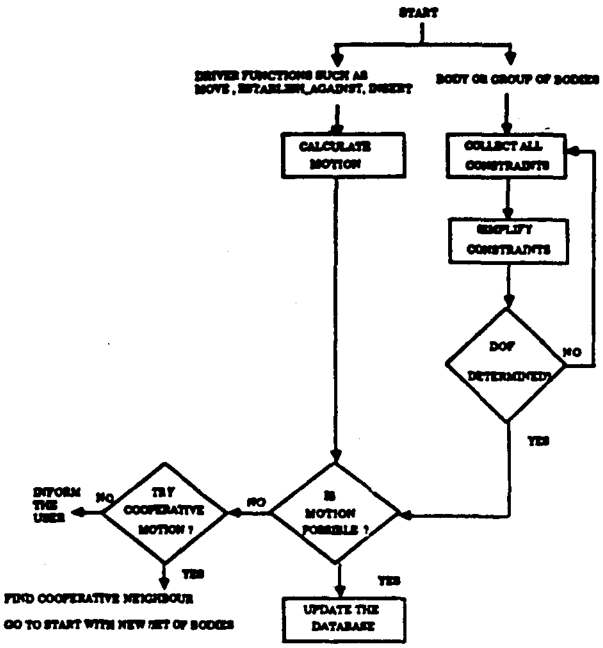


Figure 4: Feasibility of Motion

These mnemonics are combined in the following ways to give the constraints that are established by "establish-against" and "insert" operations:

TPR: This code indicates that Translation is permitted in the Plane and Rotation is permitted about a normal to the Plane. The normal to the Plane must be specified to complete the description of the constraint.

TLR: This code indicates that Translation is permitted along a Line and Rotation is permitted about that Line.

TL: This code indicates that Translation is permitted along a Line and no rotation is permitted.

RL: This codes indicates that Rotation is permitted about the specified axis.

The direction vector that is required with all of these codes is called the constraint vector (CV). Since the magnitude represented by this vector is insignificant, it is stored as a normalized vector. This direction is interpreted differently depending on the presence of P or L in the code.

Besides the above constraint primitives, we must define a few additional constraint classes in order to implement a working set. These include three other constraints--"Stationary," "Attached," and "Fixed"--and the unconstrained condition, "Free". These are clarified as follows:

The "Stationary" constraint indicates that the object has a particular position and orientation, and these cannot change.

Object A is said to be "Attached" to Object B when A and B always move together. In other words, A is rigidly connected to B. Every object must store a list of all other objects that are "Attached" to it. When the move command is to be executed for an object, all the "Attached" objects should move together.

As opposed to the "Stationary" and "Attached" constraints which are specified by the user, "Fixed" is a constraint derived from a set of user specified constraints. "Fixed" represents the condition when a combination of constraints applied to an object results in no freedom of movement for the object when considered as a single entity. However, the implication is that the object could move if it were to move together with one of the constrained objects. This is clarified by the example shown in Figure 5.

Consider the case when the following constraints are established:

B1 AGAINST B2 (F2 F1)

B2 AGAINST B3 (F3 F4)

The features involved are indicated within parentheses.

Under these two constraints, B2 can translate along a line in and out of the paper. Now, suppose we establish a third relation B2 AGAINST B4 (F5 F5) in which B4 is behind B2. The result is that B2 is now "fixed"

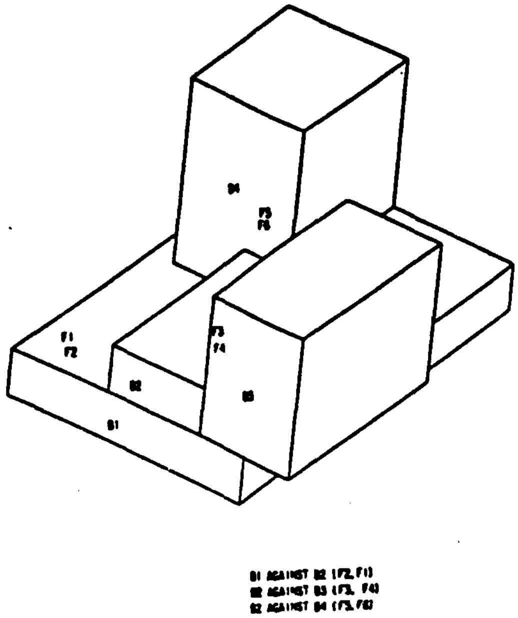


Figure 5: B2 "Fixed" By Constraints

and cannot move by itself. However, any of the combinations (B2,B4), (B2,B3), and (B2,B1) can move along different directions.

The constraints only describe the restrictions that the environment has on the object's motion. The driver functions, the "move" command, and the commands to establish relations use these constraints to determine whether the body can be moved or some other body needs to move along with the body prior to transmitting the command to the robot. By using these constraints, we can build assemblies in a constraint enforcing environment where the user or the task planner is prevented from violating any previously specified constraints, unless the system is specifically asked to do so.

In order to determine the available degrees of freedom, we will first show how to simplify the motion constraints in the next section. Following that, we will develop a method of determining the available degrees of freedom from the specified constraints.

7.0 Simplification of Constraints

Some rules for simplifying constraints are presented here. These apply to the case of rectangular blocks and cylinders. The extension to other bodies involving curved surfaces and to more general profiles is possible.

The rules may not represent a complete set, but are sufficient to demonstrate the feasibility of the approach. Thie rules which follow are for a prototypical object B1 which is being commanded to move by some driver function (→ *signifies is mapped into):*

1. IF (TPR CV1) AND (TPR CV2) AND CV1 X CV2 ≠ 0

   THEN (TL CV) AND CV = CV1 X CV1

Consider the motion for B1 in the following examples (Figure 6).

B2 AGAINST B1 (F2,F1) → (TPR CV1)

B3 AGAINST B4 (F4,F3) → (TPR CV2)

Since CV1 and CV2 are orthogonal to each other and lie in the plane, B1 can translate along a line in and out of the plane.

2. IF (TPR CV1) AND (TPR CV1) AND CV1 X CV2 = 0

   THEN (TPR CV1) OR (TPR CV2)

Consider the motion for B1 in the following example (Figure 7a).

B2 AGAINST B1 (F2,F1) → (TPR CV1)

B3 AGAINST B1 (F4,F3) → (TPR CV2)

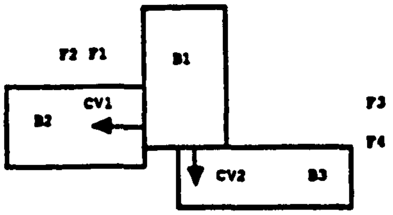


Figure 6: B2 Against B1 (F2, F1)
B3 Against B1 (F4, F3)

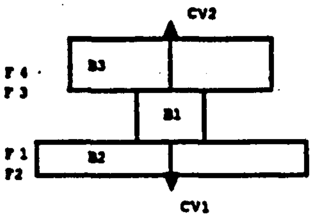


Figure 7a: B2 Against B1 (F2, F1)
B3 Against B1 (F4, F3)

B1 can still translate in the plane perpendicular to CV1 and rotate about CV1. Using CV1 or CV2 does not matter
As shown in Figure 7b, CV1 and CV2 can also be pointing in the same direction.

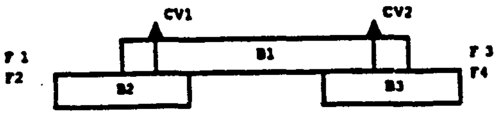


Figure 7b: B2 Against B1 (F2, F1)
B3 Against B1 (F4, F3)

3. IF (TLR CV1) AND (TLR CV2) AND CV1 X CV2 $\neq$ 0

   THEN FIX

Consider the motion for B1 in the following example for a "fit" relationship between a shaft and a hole (Figur 8).

B2 FITS B1 (S2, H1) → (TLR CV1)

B1 FITS B3 (S1, H3) → (TLR CV2)

B1 cannot move all by itself. However, B1 and B2 can move together as can B1 and B3.

4. IF (TLR CV1) AND (TLR CV2) AND CV1 X CV2 = 0

   THEN (TLR CV1) OR (TLR CV2)

Consider the motion for B1 in the following example (Figure 9).

B1 FITS B2 (S1, H2) → (TLR CV1)

B1 FITS B3 (S1, H3) → (TLR CV2)

Until now, we have been considering infinite lines and planes and, hence, no mention has been made regarding th depth of the hole and the length of the shaft. These parameters will be introduced later as factors necessar for determining the possibility of establishing relations required for object manipulation.

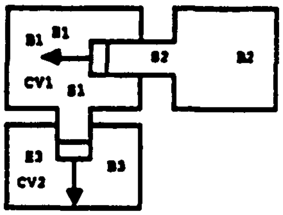


Figure 8: B2 Fits B1 (S2, H1)
B1 Fits B3 (S1, H3)

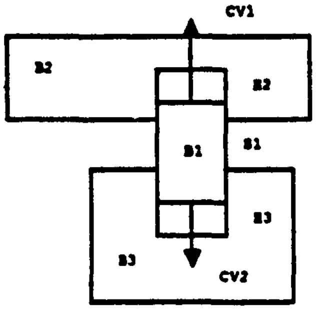


Figure 9: B1 Fits B2 (S1, H2)
B1 Fits B3 (S1, H3)

5. IF (TPR CV1) AND (TLR CV2) AND CV1 X CV2 = 0

THEN (RL CV1) OR (RL CV2)

Consider the motion for B1 in the following example (Figure 10).

B2 AGAINST B1 (F2, F1) → (TPR CV1)

B1 FITS B3 (S1, H3) → (TLR CV2)

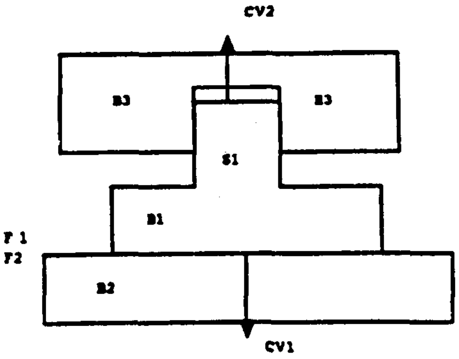


Figure 10: B2 Against B1 (F2, F1)
B1 Fits B3 (S1, H3)

B1 can rotate about CV1. B2 AGAINST B1 permits translation in a plane while B1 FITS B3 permits translation along the normal to the same plane. The net result is that B1 cannot translate at all.

6. IF (TPR CV1) AND (TLR CV2) AND CV1 X CV2 ≠ 0

THEN, IF CV1 - CV2 = 0, THEN (TL CV2), ELSE FIX

Consider the motion for B1 in the following example (Figures 11a and 11b).

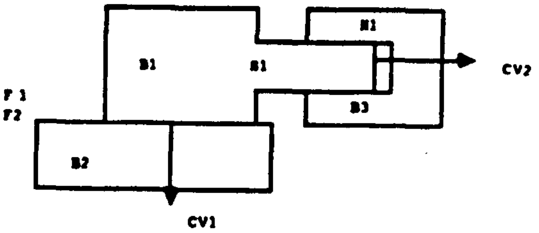


Figure 11a: B1 Against B2 (F1, F2)
B1 Fits B3 (S1, H1)

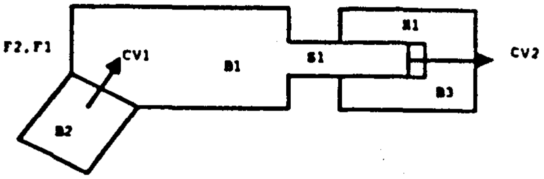


Figure 11b: B1 Against B2 (F1, F2)
B1 Fits B3 (S1, H1)

B1 AGAINST B2 (F1,F2) → (TPR CV1)

B1 FITS B3 (S1,H1) → (TLR CV2)

B1 can translate along CV2 as shown in Figure 11a for CV1 CV2 = 0. However, as shown in Figure 11b, when CV1 CV2 $\neq$ 0, B1 cannot move at all.

7. IF STATIONARY AND ANY ONE CONSTRAINT

THEN STATIONARY

If a body is stationary, it cannot move at all.

8. IF FIX AND ANY ONE CONSTRAINT EXCEPT STATIONARY

THEN FIX

If a body is constrained so that it cannot move by itself, then constraining it further will not change the FIX constraint.

9. IF FREE AND ANY ONE CONSTRAINT C

THEN C

Free refers to an unconstrained state and, hence, C is the only one constraint.

The relationships of a body with other bodies are stored symbolically using the constraint primitives discussed above. When motion feasibility is to be determined, the different relationships are first simplified using these rules. If a group of bodies moves together, then all their relationships with other objects are simplified in order to determine the feasibility of motion of the group as a whole.

The simplification is completed when we have all the possible translational and rotational degrees of freedom for the body or set of bodies. All the constraint vectors are converted into the world reference frame before simplification. This approach is preferred to updating the constraint vector every time the position of the object changes, since the number of updates required is reduced.

The next section discusses the implementation strategy for motion and for the establishment of relations.

8.0 Strategy for Implementing Motion and for Establishment of Relationships

Every primitive object has an arbitrarily selected reference point. The position of this point and the orientation of the body about this point as measured in the world reference frame is considered as the position of the object. The position and orientation of features are described relative to this reference point.

Let us first consider the motion of a primitive object. When the motion of this body to a particular position and orientation is desired, the motion matrix is given by

$$\text{MOTION} = (\text{OLD-POSITION})^{-1} (\text{NEW-POSITION})$$

using the homogeneous transformation notation. This is represented by the diagram in Figure 12.

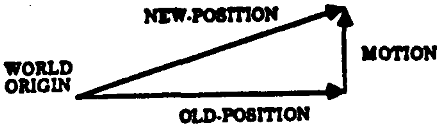


Figure 12: Calculation of Motion Matrix

8.1 Establishing Relations

Driver functions such as "establish-against" and "insert" are used to establish relations between objects. These commands are used to calculate the necessary motion matrix which is used to move the body in order to establish a relationship.

Consider the diagram shown in Figure 13. Let us define the terms used in this diagram:

POSB1: position of object B1 in the world
OLD-POSB2: position of object B2 before moving
NEW-POSB2: position of object B2 after the relation has been established. This is undetermined at this stage.
MOTION: The motion required of Object B2 to establish the relation. This is yet to be determined.
FEAT1, FEAT2: position of features on bodies B1 and B2 that are involved in this relation.
REL-ESTABLISH: a matrix that orients the constraint vectors to establish the relevant relation.

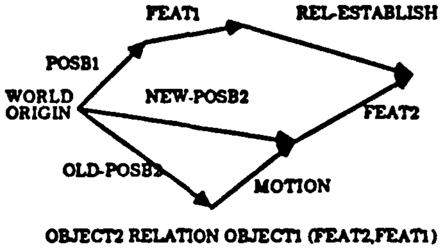


Figure 13: Establishing Relations

In the case of the "against" relationship, the REL-ESTABLISH matrix would define the rotation needed to bring the normal vectors associated with the two faces in line with each other. In the case of the "fit" relationship, it is the matrix that brings the two axial vectors in line with each other.

To determine REL-ESTABLISH, consider the example in Figure 14. Consider two unit normal vectors N1 and N2, associated with the faces F1 and F2 of two objects B1 and B2.

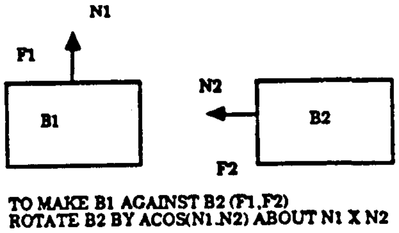

N1
F1
B1
N2
B2
F2
TO MAKE B1 AGAINST B2 (F1,F2)
ROTATE B2 BY ACOS(N1.N2) ABOUT N1 X N2

Figure 14: Definition of REL-ESTABLISH

N1 and N2 are two vectors to be aligned. A rotation of N1 by an angle $= \cos^{-1}$ (N1 N2) about the direction N1 x N2 will bring them into line with each other. This rotation is the one defined by REL-ESTABLISH (Equation 1). An additional translation is required to bring the two faces together. NEW-POSB2 incorporates the combined rotation and translation of this new position for B2 (Equation 2). MOTION is calculated using this new position NEW-POSB2 (Equation 3).

$$\text{REL-ESTABLISH} = \begin{bmatrix} & & & 0 \\ \text{ROT} & \cos^{-1}(\text{N1 N2}), & \text{N1 XN2} & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\left[\text{NEW-POS-B2}\right] = (\text{POSB1})\ (\text{FEAT1})\ (\text{REL-ESTABLISH})\ (\text{FEAT2})^{-1} \tag{2}$$

$$\left[\text{MOTION}\right] = (\text{OLD-POS-B2})^{-1}\ (\text{NEW-POS-B2}) \tag{3}$$

The following "loop equation" can be used at any time to determine if the relation has been established. If the equality is satisfied, then the relation has been established.

$$\left[\text{IDENTITY}\right] = (\text{POSB1})\ (\text{FEAT1})\ (\text{FEAT2})^{-1}\ (\text{POSB2})^{-1} \tag{4}$$

The next task is then to find out if this motion is feasible in the presence of the already established constraints.

However, there is one factor that still needs to be considered. The above-mentioned strategy for establishing relations will work only if object B1 does not move while object B2 is being moved. If object B1 moves in the process of achieving the required motion for object B2, then the relation is not yet established. This is explained by the following example (see Figure 15).

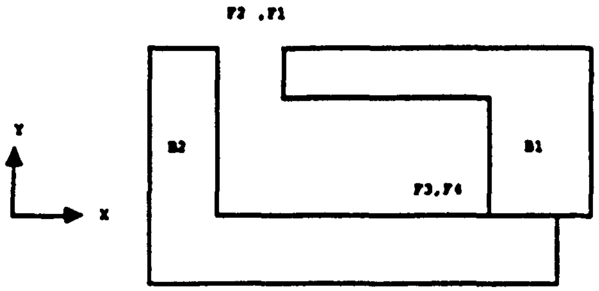


Figure 15: Example of Unachievable Relationship

The given constraint is B1 ATTACHED B2 (F3 F4). We try to establish the relation B2 AGAINST B1 (F1 F1).

The motion needed by B2 to establish this relation is calculated, and in this case, it is simply a translation in the x direction. When this motion is attempted, it is found to be feasible and, hence, is implemented. But since B1 and B2 are "attached", B1 will also move and, hence, the desired relationship will not be established. In fact, for this example, the relation can never be established since the two bodies are attached and, hence, move by the same amounts.

This problem is resolved by using an iterative procedure. Consider the diagram shown for the general case in Figure 16.

In the first attempt, motion M is the required movement of object B2 to establish the relation. However, in the process of planning motion M, motion M1 of object B1 also occurs. At the end of motion M, the "loop equation" (Equation 4) is used to check if the relation has been established. If the relation is established, the new motion matrix will indicate zero translation and rotation. If the relation is not established, then an iteration is performed in which NP2 is redefined as OP2, P1' is redefined as P1, the new rel-establish matrix is given by RE', and the motion matrix is recomputed. This is shown as M2 in Figure 16. By comparing the motion M with motion M2 (as shown later), one can find if any progress has been made towards establishing the relation. If no progress is made as in the example in Figure 15, then we conclude that the new relation cannot be established in the presence of the already established relation.

In the above iterative approach, it is necessary to determine if progress has been achieved in bringing the two objects closer to each other. We assume that the iteration is successful if the new motion prediction is "smaller" than the "initial" motion prediction. A mechanism to compare an initial motion prediction with a new motion prediction is described in more detail in reference (7).

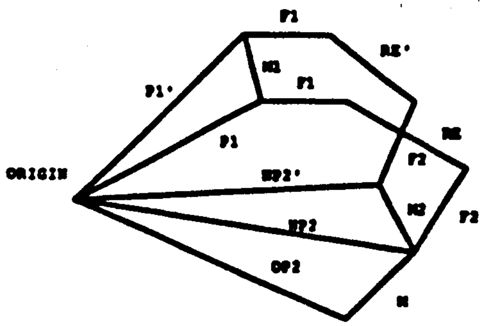


Figure 16:

P1	- old position of body 1
P1'	- new position of body 1
M1	- motion of body 1
F1	- feature of body 1
F2	- feature of body 2
OP2	- old position of body 2
NP2	- new position of body 2
NP2'	- new position desired for body 2
M	- motion of body 2
M2	- motion needed for body 2
RE	- old REL-ESTABLISH
RE'	- new REL-ESTABLISH

8.2 Feasibility of Motion

In order to decide whether a particular body motion is feasible or not, we need to consider the constraints to which the body is subjected. We have shown how to simplify these different constraints in order to determine the possible degrees of freedom. In order to facilitate the process of determining motion feasibility, the translation and rotation are each handled separately.

If, after simplification has been performed, the constraint code includes TL, then a part of the translation needed can be accomplished by moving the body along the direction given by the constraint vector. If the constraint code includes TP, then a part of the motion is achieved by translating in the specified plane. If the entire desired translation is not possible, then one cannot achieve the entire motion by moving this object alone. One must then consider moving a neighboring object along with the body of interest. This strategy is described later.

The rotation part of a given motion matrix is converted into an equivalent angle about an axis located in the world reference frame. The constraint vector associated with the rotation degree of freedom is then compared with this equivalent axis. If the two coincide, then the rotation is possible; otherwise, the rotation is not possible by this body alone.

If the body is found to be constrained from moving when considered alone, then the following strategy for considering cooperative motion with neighboring objects is pursued. A search similar to a breadth first search is used. This is best explained by an example (see Figure 17).

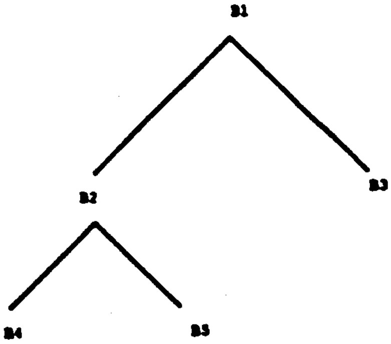


Figure 17: Cooperative Motion

Consider an attempt to move the object B1. Suppose the following relations already exist:

B1 is in contact with B2 and B3.

B2 is in contact with B4 and B5.

The first attempt is to find if B1 can be moved by the required amount all alone. If this is not possible, then an attempt is made to consider the motion of B1 and B2 together. In the constraint simplification that is performed, the constraints between B1 and B2 are not considered in this case as B1 and B2 do not have relative motion.

If B1 and B2 cannot accomplish the entire remaining motion together, then B1 and B3 are considered together. If the entire motion is still not implemented, then B1, B2, and B3 are considered together. The search order used is (B1), (B1,B2), (B1,B3), (B1,B2,B3), (B1,B2,B4), (B1,B2,B5), (B1,B2,B3,B4), (B1,B2,B3,B5), AND (B1,B2,B3,B4,B5).

The set (B1,B2,B3) is considered before (B1,B2,B4) since B2 and B3 are in direct contact with B1 which is the object of interest. The governing rule in this search is to always attempt to move a minimum number of bodies prior to attempting a larger set.

At each stage in the search, motion is carried out as far as possible. If there is still some motion :equired, then the search continues. As a result, the total desired motion for B1 might be achieved partly by B1 alone and partly by B1 and B3 together.

If a constraint "Stationary" is encountered in the search, the search along the path is immediately terminated. Note that all the attached bodies are always considered together in this motion feasibility paradigm.

The structure is described as a tree diagram. However, the generalized version can be a graph (see Figure 18a). This graph can be converted into a tree by repeating appropriate nodes (see Figure 18b).

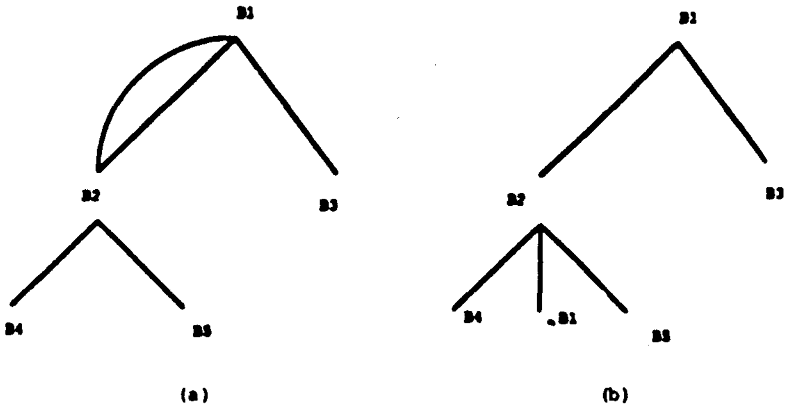


Figure 18: Conversion of Graph to a Tree

8.3 Motion of Assemblies

To determine the motion possibilities for an assembly, we have to treat them as an "attached" group of primitives. We store the position and orientation information for each primitive. When the assembly has to move, all the primitives must move by the same amount.

8.4 Range of Motion

In our discussion so far, we have considered the constraints as basically degrees of freedom. For instance, TPR indicates that translation is allowed in a plane and rotation is permitted about the normal to the plane. However, these motions are not infinite. These motions are limited by the finite dimensions of the bodies involved in the relationship.

9.0 Implementation

The knowledge representation system described here has been implemented using the FLAVOR system on the LMI LISP Machine. A typical data network is shown in Figure 19. In the constraints discussed above, we did not include the range of motion. Geometric algorithms to detect collision of objects and overlapping of faces, overlapping of two holes, etc., are implemented as tests to be conducted before sending any motion command to the robot. This scheme provides for expandability of the system to incorporate checks such as tolerance matching, surface finish matching, checking types of fit, calculating inertia properties, etc. Some examples can be found in reference (7).

The system described thus far is a constraint enforcing system. The breaking or making of relationships as a side effect of motion is not considered. This consideration of dynamic constraint propagation requires a constraint network on a global scale which was indeed designed and implemented (7), but is outside the scope of this paper.

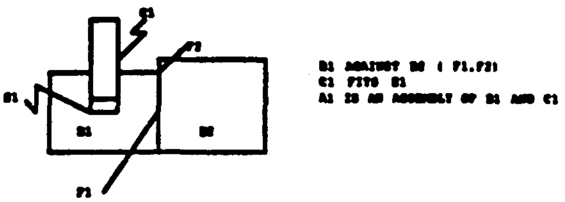
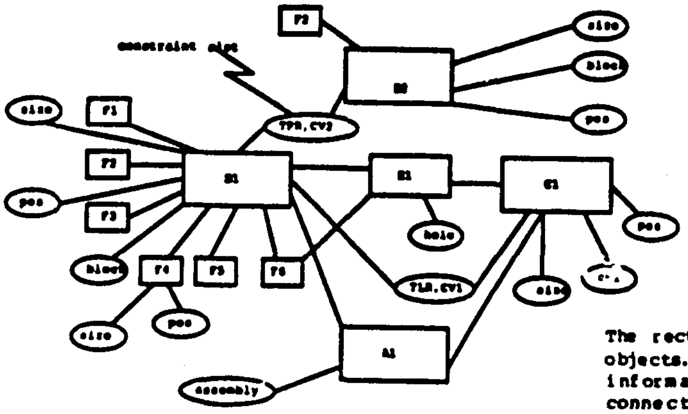


Figure 19a: Assembly Model



The rectangles indicate data objects. The ovals indicate information slots. Ovals connecting two data objects indicate information slots in both data objects.

Figure 19b: Data Network for Model Shown in Figure 19a

Note: This is not a complete network as some repetitive structures are not shown.

10.0 Conclusion

The applications of ~~this~~ the knowledge representation system pres. [illegible] are manifold. The programming of robots using higher level constructs hides the robot specific programming details from the user by submerging them into the domain specific knowledge base. The constraint enforcing environment provides for an on-line debugging facility which operates at the conceptual task level rather than at the program's syntax level. Before sending a command to a robot, the command is simulated in the world model, and this provides for a real time error preventive capability. With the addition of dynamic constraint propagation, the knowledge representation system becomes capable of simulating simple assembly operations. This provides a basis for the development of a task planner.

11.0 Acknowledgment

This work was supported in part by National Science Foundation PYI Grant # DMC8351827.

12.0 References

1. Lieberman, L. L and M. A. Wesley, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly," IBM Journal of Research and Development, July 1977, pp. 321-333.
2. Popplestone, R. J. and A. P. Ambler, "A Language for Specifying Robot Manipulation," Robotic Technology (A. Pugh, ed.), Peter Perigrines Ltd., London, 1983, pp. 125-141.
3. Lozano-Perez, T., "Task Planning," in Robot Motion Planning and Control, edited by M. Brady, et al., MIT Press, 1982.
4. Mazer, E., "LM-GEO, Geometric Programming of Assembly Robots," Laboratoire IMAG, BP N68 38042, Saint Martin D'here, LEDEX.
5. Taskase, H. and N. Nakajima, "A Language for Describing Assembled Machines," International Symposium on Design and Synthesis, 1979.
6. Fahlman, S. E., "A Planning System For Robot Construction Tasks," Artificial Intelligence, Vol. 5, No. 6, 1976.
7. Jain, A. K., "Knowledge Representation For Automatic Assembly Using Robots," M.S. Thesis, Department of Mechanical Engineering, University of Minnesota, 1986.

omit

Real Artificial Intelligence for Real Problems

R.S. Doshi
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

This is a summary of a panel discussion that was held at the conclusion of the sessions on artificial intelligence (AI). ~~Panel members were: P. Cheeseman, NASA Ames; S. Harmon, Robot Intelligence International; V. Hayward, McGill University; K. Kempf, FMC; I. Oppenheim, CMU; A. Sanderson, CMU; and D. Wilkins, SRI.~~

The following questions were discussed by the panel: 1) What other research areas need to he addressed to extend your work? 2) In your research experience, have you found it fruitful to choose application domains from real-world scenarios? 3) The telerobot program needs to merge task planning and spatial reasoning. What do you think needs to be done to achieve this merger? 4) Is the blocks world really simple/trivial? Responses and discussion on these issues are summarized below.

The blocks world domain is definitely not simple. Many AI planning systems have solved many aspects of the toy-blocks world. However, the real-blocks world is largely unsolved. The real-blocks world problem includes the following subproblems: 1) Representing geometries of the workcell, the robot(s), the objects, assemblies of objects; 2) Practical techniques for efficiently reasoning with these representations; 3) Developing mathematical and symbolic models of sensor hardware and interpretation of sensor behavior; 4) Developing mathematical and symbolic models of external agents like friction, dynamics, kinematics and wear and tear. Although these areas are undergoing active research, results are far from complete.

Nowadays, it is imperative to choose domains from real-world scenarios. This helps make for good demonstrations. However, to actually make it realistic, the application must also include one or more (to a certain manageable degree) subproblems of the real-blocks world (discussed above). An example of such a realistic problem is to investigate an AI task planner which also reasons about the (numeric) geometric representations and plans actions based on real, actual, existing robot hardware. Even if the AI task planner is simple, solution of the combined problem adds a lot to the state-of-the-art.

One of the important unsolved problems is that of integration of diverse technologies in the fields of mathematics, computer science, cognitive science, artificial intelligence, mechanical engineering, robotics, electrical engineering, etc.

Integration of AI with non-AI systems is something which is talked about a lot and postponed. Many theoretical suggestions do exist. The general feeling seems to be that the integration will not be a batch-mode integration ("do-this--then-do-that"). It will be a multi-contact, heavily coupled interaction. The important thing is to start integrating and failing. Fail enough, and often enough, so enough can be learned.

PAGE 114 INTENTIONALLY BLANK

omit

TRAJECTORY PLANNING FOR MANIPULATORS

S11-63
58
168180

The Use of 3-D Sensing Techniques for On-Line Collision-Free Path Planning

V. Hayward, S. Aubry, and Z. Jasiukajc
McGill University
Montreal, Quebec, Canada H3A2A7

MP756796

Abstract

~~This paper discusses~~ the state of the art in collision prevention for manipulators with revolute joints, showing that it is a particularly computationally hard problem. Based on the analogy with other hard or undecidable problems such as theorem proving, ~~we propose~~ an extensible multi-resolution architecture for path planning, based on a collection of weak methods. Finally, ~~we examine~~ the role that sensors can play for an on-line use of sensor data.

is discussed.

is proposed

1. Introduction

Automated collision prevention for robot manipulators is an essential feature seldom available, even in its simplest forms, in today's robotic systems. The term "collision prevention" will refer to collision detection and collision-free path planning. This paper presents current issues in collision prevention for manipulator robots with revolute joints in relation to the use of sensors.

Given the difficulty of the problem, only partial solutions have been found. Furthermore, we believe that it is particularly constraining to have to assume that robotic systems will operate in perfectly pre-modeled environments, as do all the currently existing industrial systems. As a consequence of the "perfect model" approach, many largely unsolved issues immediately arise: uncertainty representation and assessment, time-varying environments, computational and storage complexity.

We will rather advocate an "imperfect model" approach that will lead us to the consideration of a multi-level system heavily relying on sensory information and using multi-resolution algorithms. Instead of elaborating a theoretically exact method, and then deriving the computational and storage complexity in order to design a computer system to implement it, we will rather present methods that only partially solve the problem, but whose performance improve as the computing power is increased. Studying the problem of collision path-finding in connection with the available sensing techniques also provides some insight into the solution.

2. Current Methodologies for path planning

Experimental research in path-planning in the context of mobile robots has been fo far more successful because of the reduced complexity of the two-dimensional case. However, many concepts developed in the two-dimensional case do not extend readily to the three-dimensional case. For example, the conceptually attractive "configuration space approach" fails to extend easily to the case of manipulators with revolute joints, whereas it applies very nicely to the case of mobile robots. The reason is that the Cartesian space, in which obstacles—or free-space—are described, maps very awkwardly into the configuration space. In case of a manipulator, the configuration space is equivalent to the joint space. The mapping is highly non-linear and occurs between spaces of different dimensionalities. The computational complexity becomes unmanageable beyond three joints for the problem of mapping the Cartesian space scene into joint space as well as for searching the resulting graph. The approach is limited to three joints (Lozano 1986, Gouzène 1984) even if recursive decomposition schemes are utilized (Faverjon 1985).

Another widely adopted approach (Khatib 1986) is of local nature and consists of the computation of a artificial field of potential increasing near obstacles, and globally decreasing toward a goal position. Pseudo-forces are then included in the low-level motion servo computations of the manipulator. As a result, the manipulator is controlled to move away from obstacles and toward the goal position. Unfortunately, the method breaks down in obstacle configurations that create local minima. Computational complexity also precludes attempts to enlarge the scope of this method. The great attraction of this idea is the possibility to use sensor data directly for the computation of the potential instead of an a priori model. Similar schemes can be formulated in kinematic terms, that is in terms of velocities, instead of forces. The problem of local minima can be largely eliminated by the use of redundant manipulators: the trajectory of the end-effector can be completely specified and the redundant linkages used to avoid obstacles using only local information (Maciejewski 1985). Operation Research has been also considered helpful to attack the complexity problem (Grechanovsky 1983).

The methods of the last category resort to limit the class of tasks being planed. For example, the range of motions can be restricted to pick and place operations (Brooks 1983b). Similarly, the range of obstacles can be restricted, for example to pillars shapes (Luh 1984).

3. Proposed Ideas

We would like to suggest a few new ideas to the problem of collision prevention, in the view of their use in an online control system. We mean by that, trajectory generation techniques that allow the computation of collision-free trajectories in the same amount of time as require by the motion of general purpose robots, using limited computational resources. At this point, we would like to draw an analogy with the problem of theorem proving in computational logic. This problem is undecidable, that is to say, if the proposition submitted to the system is in fact false, the result cannot be obtained in a finite number of steps.

The search space to explore in order to prove a proposition can become arbitrarily large. If the proposition under study in indeed provabie, efficient methods in theorem proving attempt to use powerful heuristics to reduce the search space. These heuristic methods are called "weak methods" because they do not guarantee success, but are likely to converge in most interesting cases. If the proposition is false, this conclusion may not be reachable in a finite number of steps, and one has to resort to cut the search at some arbitrary point and to assume that the proposition probably is false.

In path planning, there are many heuristics available, and we suppose a limited amount of available computations, hence the architecture described below, based on a collection of weak methods.

3.1. Computational Architecture

We require the system to be extensible in the sense defined by Brooks (Brooks 1986). As researchers are devising new methods to calculate collision free trajectories, we would like to be able to integrate these advances while causing a minimum of disturbances to existing and working parts of the system.

The following diagram illustrates the design concept of an extensible architecture. The question of whether each of the methods will reside on one or several processors is of little importance. What is is important is to design them as peers such as they can accept the same input and produce the same output formats. The crucial point is not to attempt to parallelize the computations of one particular method, because we know that many of them require exponential times to execute, but to parallelize the methods between them so that we obtain a natural selection of the most appropriate for the situation at hand.

The computations should be done at a least two levels. The top-level searches for collision-free trajectories, using any of the methods available. Input to the high-level is data obtained from global sensor measurements as discussed in the section "Sensors," or from pre-determined models obtained from data-bases. The lowest level is a local collision detector that also uses either sensor information or pre-determined model information. We will require an efficient collision detector to certify the proposed trajectories, or use on-line proximity sensor mounted on the arm to locally modify the trajectories as they are executed. In the later case, we take the chance that the motion may never terminate.

3.2 A Variety of Heuristics: Archetypical Motions

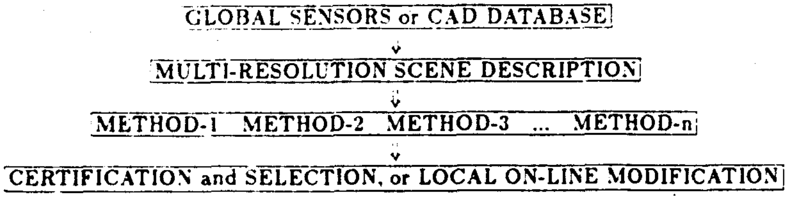
The study of robot motions shows that given an approximate description of a robot's environment, and given the initial and final configurations, robot motions can be classified into classes of archetypical motions. A preliminary analysis shows that collision-free motions bear a strong relationship with the structure of the workspace. This relationships can be exploited to built a system that infers plausible motions. In this framework, the problems under study are:

- Classification of obstacles according to the influence that their shape might have on the motions: small, large (with respect to the robot), compact, elongated, flat, etc... Interesting simple cases are: spheres, infinite cylinders, half spaces, and holes.

- Classification of the relations between these obstacles with respect to the robot elements: proximity, position with respect to the elbow, under, above, on the left, on the right, etc... Inference will occur on criteria of this kind.

- Classification of typical motions: retraction, extension, sweep, wrist re-orientation. The consequences of these motions are explicit: if joint No 1 turns in the positive direction and the arm is stretched, then end-effector sweeps on the left; if the arm retracts and is in the elbow up configuration, then the end-effector moves inward, and the elbow moves upward, etc...

Once the scene and the robot attitude are encoded in terms of facts and rules, motions can be generated by automatic inference.



GLOBAL SENSORS or CAD DATABASE

MULTI-RESOLUTION SCENE DESCRIPTION

METHOD-1 METHOD-2 METHOD-3 ... METHOD-n

CERTIFICATION and SELECTION, or LOCAL ON-LINE MODIFICATION

3.3 Joint Decoupling

Joint decoupling is another way to attack the problem. The observation of certain collision avoiding motions reveals that motion planning can be performed by planning the motions of the joints independently. During such a motion, in the *reference frame* attached to each link of a robot, all the obstacles appear as moving obstacles. The task consists for each link to plan a one-dimensional trajectory in its own coordinate frame with a time-varying environment. We know from (Kant 1984 and 1986) that such a planning is possible, by planning the velocity along a predetermined path. This algorithm finds solutions in a large number of cases, when the priority among the set of joints is adequately determined. *The problem is formaly* equivalent to moving multiple objects as in (Erdman 1986).

3.4 Piece-Wise Trajectory Decomposition

Another heuristic method can be described as follows. If the arm is to move from point A to point B, a trajectory is generated at the first iteration using a very simple scheme: a linear joint interpolated motion between A and B, for example. The trajectory is then verified. In case of collision, an intermediate knot point is generated by the closest non interfering position. The initial segment is then split and the process recursively iterated on the sub-segments.

3.5 The generate-test-refine architecture

We have just listed three powerful heuristics to reduce the search space of the problem. There exist others. We can augment the power of these heuristics by feeding back to a motion planner information provided by the collision detector in case of the failure of a plan, or information provided by a merit estimator, in case of success. The system is left interating during the allocated time period, the last best solution being retained.

3.5 Good Collision Detectors

Of what precedes, we require a good quality collision detector, that is to say, one that does not require exponential nor polynomial times to perform and one that uses multi-resolution algorithms. This problem has been examined in (Hayward 1986). One approach is to perform the modeling the robot in terms of contro' points scattered on its boundary. Collision detection can be then performed by showing that all the control points are in free-space. (note that there is no need to worry about rotations). A multi-resolution system can then be easily obtained.

The quality of the result augments with the allocated running time and the CPU power. Methods for generating multi-resolution control points are indicated in (Bhan 1986). Octree encoding methods provide very naturally for multi-resolution algorithms, however, we have other schemes under consideration because octree make no use of the coherence that might be present in a scene and therefore can lead to great inefficiencies.

4. Sensors

"Model Building Sensing" is used to gather *global* three-dimensional information from the environment. In a robotic context, the sensors perform a "surveying" function, providing information to be used by the path planning module. This is quite different from the on-line uses of sensors in which the local environment is continuously sampled so as to avoid crashes. In particular, model building sensors must operate *over a wider range than their servoing* counterparts.

4.1 Global Sensors versus Proximity Sensors

The chosen sensor must be either a proximity sensor attached to a "roving" arm or it must be capable of acquiring three-dimensional information at a distance. In the first case, the accuracy is limited only by that of the manipulator. However, control problems are likely to crop up for complex environments where concavities abound. Such problems arise because the environment is not known a priori: in fact, the environment is difficult to explore precisely because it is not known! Consequently, such a process is likely to be a slow one.

We contend that such proximity methods are only advisable when the task environment is so intricate that spatial considerations prevent larger apparatus such as those we describe below to conveniently operate. Suppose for example that we wish to model the bottom of a narrow, oblong cavity inside a given object. We can safely assume that our robot arm can indeed penetrate the cavity and orient itself within it, since otherwise there would be little point in modeling it. Using the very manipulator which is to perform the robotic task is then the most direct way to model the task environment.

4.2 Acquiring 3-D Information

Techniques developed for acquiring three-dimensional information at a distance are still the preferred answer to automated model-building in most cases. These techniques can be either photometric or telemetric.

4.2.1 Photometric Techniques

Photometric techniques attempt to infer distance from photographic images. But such images map intensity, an extrinsic characteristic of the three-dimensional world, onto the the two-dimensional plane along the lines of a perspective projection. The task of recovering the correct interpretation for a given image is then a formidable one since it requires that the perspective ambiguity (lines map into points) be resolved from the intensity cue alone; formally, this task consists of inverting an illumination-reflectance operator I which maps the three-dimensional scene to the image plane. The so-called "shape-from" techniques attempt to perform that difficult inversion using a combination of analytical work (Horn 1968; Horn 1975; Ferrie 1986; Levine, O'Handley and Yagi 1973), and of higher-level cognitive processes (Rosenburg, Levine and Zucker 1978; Bajcsy and Lieberman 1976; Shirai 1973; Marr 1976). These methods have been much investigated in part for their similarity to human visual processing, but also because they do not require sophisticated optical hardware.

4.2.2 Telemetric Techniques

In contrast with photometric techniques, telemetric techniques usually require specialized hardware but are much easier to analyze in return and therefore constitute a much preferable means for automatic three-dimensional scene acquisition. The goal here is to build "range images": a range image maps the distance of the closest point in the scene to every node of an orthographic grid the size of that scene. These images are usually constructed by monitoring patterns of points (Hasegawa 1982; Ishii and Nagata 1976), lines (Oshima and Shirai 1979: Sato and Inokuchi 1985), or grids (Potmesil 1979) of light which are successively projected onto the scene and reflected to a sensor located at or near the light emitting device (often a laser). Either positional analysis of the returning rays or time-of-flight discrimination can now be used to infer the range of the closest obstacle. In the first case, simple geometrical relationships relating emitted and returned rays yield the sought distance in a process called *triangulation.* In the second case, the time taken by light rays to travel from and back to the emitting laser source allows us to calculate that same distance. Needless to say, the practicability of the latter method is limited by the very sophisticated electronics that the enormous speed at which light travels requires (Lewis and Johnston 1977; Nitzan, Brain and Duda 1977.)

An alternative time-of-flight method uses sound waves instead of light rays because of their more manageable speed. Although simple in principle, the method suffers from various engineering problems such as the need for frequent recalibration. the difficulty experienced in focusing sound waves, as well as their hard-to-model reflective properties.

In summary, the "safest" and most accurate methods of acquiring distance information seems at present to be triangulation. However, one should not discount ultrasonic time-of-flight methods which are already commercially available. Further, many researchers believe that laser time-of-flight methods will soon present itself as the most viable method since it offers in theory the greatest absolute accuracy. The interested reader should refer to the excellent review by Jarvis (Jarvis 1983) for further reading on range acquisition techniques.

5. Conclusion

In this paper, we have presented an overview of methods related to the collision prevention for manipulators with revolute joints. It has been shown that it is a difficult problem in its generality and we have proposed a computational architecture based on an analogy with an another domain of Artificial Intelligence.

6. Acknowledgements

The ideas in this paper were initially formulated when the first author was developing a Cartesian-based collision detector while at CNRS (France). and by the second author while studying 3D sensing techniques at McGill University. Further contributions are due to conversations with Kamal Kant at McGill University, and to an inspiring lecture delivered by R. A. Brooks.

7. References

Ahuja, N. et al. 1980. Interference detection and collision avoidance among three dimensional objects. *First Ann. Nat. Conf. on Artificial Intelligence*, Stanford, CA.

Ahuja, N., Nash, C. 1984 (May). Octree representation of moving objects. *Computer Vision, Graphics, and Image Processing*, Vol 26, No 2.

Bajcsy, R., Lieberman, L. 1976. Texture gradient as a depth cue. *Computer Graphics & Image Processing*, Vol. 5, 1976, pp. 52-67.

Bham, B. Chih-Cheng, H. 1986 (October). Computer aided geometric design based 3-D models for machine vision. *Proceeding of the Eighth International Conference on Pattern Recognition, IAPR*, Paris, France, pp. 107-110.

Boyse, J. W. 1979. Interference detection among solids and surfaces. *Communications of the ACM*, Vol 22, No 1.

Brooks, R. A. 1983*a* (March). Solving the find-path problem by good representation of free space. *IEEE Trans. on Systems, Man, and Cybernetics*, Vol 13, No 3.

Brooks, R. A. 1983*b* (Winter). Planning collision-free motions for pick-and-place operations. *International Journal of Robotics Research*, Vol 2, No 4.

Brooks, R. A. 1986 (March). A robust Layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14-23.

Cameron, S. 1985(March). A study of the clash detection problem in robotics. *IEEE Second Conference on Robotics and Automation*, St. Louis, Missouri. pp. 488-496.

Canny, J. 1984. Collision detection for moving polyhedra. AI Memo No 806, MIT, October 1984.

Erdman, M., Lozano-Pérez, T. 1986 (May). On Multiple Moving Objects. *Massachusetts Institute of Technology Artificial Intelligence Laboratory*, A. I. Memo 883.

Faverjon, B. 1984 (June). Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator. *IEEE First Conference on Robotics*, Atlanta.

Ferrie. F.P. 1986 (March) Reconstructing and interpreting the 3D shape of moving objects. *Ph.D Thesis, Department of Electrical Engineering, McGill University*, Montréal.

Gouzène, L. 1984 (Winter). Strategies for solving collision-free trajectories problems for mobile and manipulator robots. *International Journal of Robotics Research*, Vol 3, No 4.

Grechanovsky, E., Pinsker, I. Sh. 1983. An algorithm for moving a computer-controlled manipulator while avoiding obstacles. *Proc. Eighth IJCAI.* 807-813. Karlsruhe, W. Germany.

Hasegawa, T. 1982 (May). An interactive system for modeling and monitoring a manipulation environment. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-12. pp. 250-258.

Hayward, V., Paul R. P. 1984 (June). Introduction to RCCL: a robot control "C" library. *IEEE First International Conference on Robotics*, Atlanta.

Hayward, V. 1986 (April). Fast collision detection scheme by recursive decomposition of a manipulator workspace. *IEEE Conference on Robotics and Automation*, San Francisco, CA, pp. 1044–1049.

Horn B.K.P. 1968 (May) Focusing. *M.I.T. Project MAC*, AI Memo 160.

Horn B.K.P. 1975 Obtaining shape from shading information, in *The psychology of computer vision* P.H. Winston (Editor), McGraw-Hill.

Ishii, M. and Nagata, N. 1976. Feature extraction of 3-D objects and visual processing in a hand-eye system using laser tracker, *Pattern Recognition*, Vol. 8 (1976). pp. 229–237.

Jarvis, R. A. 1983 (Mars). A perspective on range finding techniques for computer vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5(2), pp. 122–139.

Kant, K., Zucker, S. W. 1984 (November). Trajectory planning in time-varying environments, I: TPP=PPP+VPP. Technical Report, TR-84-7R, Computer Vision and Robotics Laboratory, Department of Electrical Engineering, McGill University.

Kant, K., Zucker, S. W. 1986. Toward efficient trajectory planning: the path-velocity decomposition. *International Journal of Robotics Research*, Vol 5, No 4, to appear.

Khatib, O. 1986 (Spring). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, Vol 5, No 1.

Knuth, D. E. 1973. *The Art of Computer Programming, Volume 1 / Fundamental Algorithms*, Addison-Wesley.

Levine, M. D., O'Handley, D. A., Yagi, G. M. 1973. Computer determination of depth maps, *Computer Graphics & Image Processing*, Vol. 2, pp. 134–150.

Lewis, R.A. and Johnston, A.R. 1977. A scanning laser range finder for a robotic vehicle, *Proceedings of the fifth International Joint Conference on Artificial Intelligence*, pp. 762–768.

Lozano-Pérez, T. 1986 (June). A simple motion planning algorithm for general robot manipulators. *Massachusetts Institute of Technology Artificial Intelligence Laboratory*, A. I. Memo 896.

Lozano-Pèrez, T. 1984 (November). An approach to automatic robot programming. USA-France Robotics Workshop, University of Pennsylvania.

Lozano-Pèrez, T. 1981. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man and Cybernetics*, Vol SMC11, pp. 681 698.

Lozano-Pèrez, T. 1983 (February). Spatial-Planning: a configuration space approach. *IEEE Transactions on Computers*, Vol C32, No 2.

Luh, J. Y. S. 1984 (Spring). A scheme for collision avoidance with minimum distance traveling for industrial robots. *Journal of Robotic Systems*, Vol 1, No 1.

Maciejewski, A. A., Klein, K. A. 1985 (Fall). Obstacle avoidance for kinematically redundant manipulators. *The International Journal of Robotics Research*, Vol 4, No 3.

Marr D., Poggio, T. 1976 (June). Cooperative computation of stereo disparity, M.I.T., A.I. Lab, Memo No. 364.

Nitzan, D., Brain, A., Duda, A. 1977 (February). The measurement and use of registered reflectance and range data in scene analysis, *Proceedings of the IEEE*, pp. 206–221.

Oshima, M. and Shirai, Y. 1979. A scene description method using three- dimensional information, *Pattern Recognition*, Vol. 11, pp. 9-17.

Potmesil, M. 1979. Generation of 3D surface descriptions from images of pattern illuminated objects, *IEEE Conference on Pattern Recognition and Image Processing*, pp. 553-559.

Rosenburg, D, Levine, M.D. and Zucker, S.W. 1978 (November.) Computing relative depth relationships from occlusion cues, *Proceedings of the Fourth International Joint Conference on Pattern Recognition*, Kyoto. Japan pp. 765-769.

Sato, K., Inokuchi, S. 1985. Three-dimensional surface measurement by space encoding range imaging, *Journal of robotic systems*, 2(1), pp. 27–39.

Udupa, S. M. 1977 (August). Collision detection and collision avoidance in computer controlled manipulators. *Proceedings of the Fifth IJCAI*, MIT, Cambridge Massachusetts, August 1977.

Wong, E. K., Fu, K. S. 1986 (March). A hierarchical orthogonal space approach to three-dimensional path planning. *IEEE Journal of Robotics and Automation*, Voi RA-2, No 1.

Collision-Free Trajectory Planning Algorithm for Manipulators

F. Pourboghrat and J.Y. Han
Southern Illinois University
Carbondale, IL 62901

1. Abstract

Collision-free trajectory planning for robotic manipulators is investigated,~~in this paper.~~ The task of the manipulator is to move its end-effector from one point to another point in an environment with polyhedral obstacles. An on-line algorithm is developed based on finding the required joint angles of the manipulator,according to goals with different priorities. The highest priority is to avoid collisions, the second priority is to plan the shortest path for the end effector, and the lowest priority is to minimize the joint velocity for smooth motion. The pseudo-inverse of the Jacobian matrix is applied for inverse kinematics. When a possible collision is detected, a constrained inverse kinematic problem is solved such that the collision is avoided. This algorithm can also be applied to a time-variant environment.

2. Introduction

Ordinary tasks for a robotic manipulator are to move its end effector from an admissible point to another admissible point in an environment with obstacles. For that, the initial and final configuration of the manipulator are often given for the trajectory planning. Usually, there are infinite paths for the end effector. Even for a specific path of the end effector, there are still infinite trajectories possible for the manipulators. However, some of the trajectories are not feasible becasue of arm geometry, obstacles, and some kinematic or dynamic constraints. Even with the kinematically feasible trajectories, some computational or logic problems in the algorithms may make them impractical.

3. Trajectory Planning

In order to move from one point to another, in the task space, one needs to solve for the angular information from the spatial information, using the tnverse kinematic relationship. Consider a robotic manipulator with n degrees of freedom. Let the kinematic relationship between joint angles and the end-effector position and orientation be given by

$$X = f(q) \tag{1}$$

where X is the m-dimentional vector of the end-effector position and orientation, and q is the n-dimensional vector of joint angles. For a kinematically redundant manipulator, the dimension of q is greater than the dimension of X, (n>m). Differentiating the above relation, we get

$$\dot{X} = J(q)\dot{q} \tag{2}$$

where $J(q) = df/dq$ is the mxn Jacobian matrix, [7]. For a redundant manipulator, the Jacobian matrix will have more columns than rows. Moreover, the inverse of such non-square matrix is not defined in a regular sense. However, useful solutions of equation (2) can be found, by using the generalized-inverses of the Jacobian matrix J, and is given by

$$\dot{q} = J^{\dagger}\dot{X} + (I-J^{\dagger}J)Z \tag{3}$$

where $J^{\dagger} = J^{T}(JJ^{T})^{-1}$ is the pseudo-inverse of J, I is the nxn identity matrix, and Z is an arbitrary n-dimensional vector. When the vector Z is selected to be zero, equation (3) reduces to

$$\dot{q} = J^{\dagger}\dot{X} \tag{4}$$

which gives the best approximate solution to equation (2). This is in the sense that if $\dot{q}_*$ is the solution of (2), given by (4), then $||\dot{q}_*|| < ||\dot{q}||$, where $\dot{q}$ is any other solution of (2) that is given by (3), [5-6]. It should be noted that, such minimum norm, or best approximate solution,is defined when there is no restriction in the task space. This means that, in a restricted environment, the above mentioned best approximate, solution may not be feasible for application and may result in collision.

Let us define collision point to be the point on the manipulator body which has the potential to collide with the obstacle. The collision-free trajectory planning problem here is to develop an algorithm, for the on-line determination of the required joint angle rates, $\dot{q}$, for safe manipulator motion. The approach is to continuously monitor the task space, for detecting possible collisions. If no potential collision is detected, the required joint angle rates are generated, using the best approximate solution, to move the end-effector on a shortest distance. But when a potential collision point is detected, the trajectory is modified in order to avoid collision.

4. Obstacle Avoidance

In order to avoid obstacles, one needs to use the kinematic relationship for the collision points, similar to that of equations (1) and (2). Let the potential collision point, in the task space, be denoted by X_c. Then, similar to equation (2), we can write

$$\dot{X}_c = J_c(q)\dot{q} \tag{5}$$

where J_c is the mxn Jacobian matrix for the collision point. The inverse kinematic solution to the above is similar to (3) and is given by

$$\dot{q} = J_c^{\dagger}\dot{X}_c + (I-J_c^{\dagger}J_c)Z' \tag{6}$$

where $J_c^{\dagger}$ is the pseudo-inverse of J_c, and Z' is an arbitrary n-dimensional vector.

Now, the problem of obstacle avoidance is that, when a potential collision is detected, the highest priority is to avoid the obstacle, and,if needed modify the position of end-effector. In order for the trajectory planning to have minimum norm, we choose $Z = 0$ as in (4). On the other hand we choose $Z'=0$, like in (6), to account for both collision avoidance and trajectory planning. From (3) and (6), a minimum norm solution for Z' is

$$Z' = [J(I-J_c^{\dagger}J_c)]^{\dagger}[\dot{X}-JJ_c^{\dagger}\dot{X}_c].$$

Plugging this back into (6), we get

$$\dot{q} = J_c^{\dagger}\dot{X}_c + (I-J_c^{\dagger}J_c)[J(I-J_c^{\dagger}J_c)]^{\dagger}[\dot{X}-JJ_c^{\dagger}\dot{X}_c].$$

Then, using the following identity, [6]

$$(I-J_c^{\dagger}J_c)[J(I-J_c^{\dagger})]^{\dagger} = [J(I-J_c^{\dagger}J_c)]^{\dagger}$$

we get

$$\dot{q} = J_c^{\dagger}\dot{X}_c + [J(I-J_c^{\dagger}J_c)]^{\dagger}[\dot{X}-JJ_c^{\dagger}\dot{X}_c]. \tag{7}$$

The above relation, generates the joint angle rates q such that the obstacle is avoided and the end effector velocity is modified, for the on-line trajectory planning.

Now the question is how and in what direction the end effector spatial velocity should be changed. For the algorithm to be fast and implementable, a finite search for the minimum norm solution is considered. The value of $\dot{X}_c$ is preselected by the user, and the value of $\dot{X}$ modification is also preselected by a value of ε. The value of $\dot{q}$ may now be found by examining seven different directions for rate modification, e.g., ε-variation in plus or minus X,Y,Z coordinates and also no modification. The smallest norm $||\dot{q}||$ is then chosen for trajectory modification, from seven different possibilities.

The overall algorithm is such that when no potential collision is detected, a minimum norm solution $\dot{q}$ is planned according to (4). But, when a potential collision is detected, the obstacle is avoided and the path in modified according to (7).

In order to develop the algorithm, it is assumed that:

(1) the solution exists
(2) the obstacles are represented by polyhedrals.
(3) the geometrical information about the task area is known, e.g., using sensory systems, the positions of obstacles and manipulator links are known.
(4) the potential collision point on the manipulator links can be detected.
(5) only a single collision may occur, and will be detected, at any given time.

5. The Algorithm

The following summarizes the steps, involved in the proposed algorithm, for the on-line collision-free trajectory planning of the robotic manipulators. The steps of the algorithm are:

(1) Determine a minimum-length path for the end-effector, from the current position to target position.

(2) Check if there is a potential collision point. If there is, go to step (5), otherwise continue.

(3) No potential collision is detected. Make an incremental move according to the joint angle rates vector $\dot{q}$, given by equation (4).

(4) If the end-effector has not reached the target, go to step (2). If it has reached the target, go to step (7).

(5) Potential collision is detected. Make an incremental move according to the joint angle rates vector $\dot{q}$, given by equation (7), such that $||\dot{q}||$ is minimized in a finite search.

(6) Go to step (1).

(7) Stop.

6. Conclusion

On-line, collision-free trajectory planning is discussed. An algorithm, which utilizes sensed information about the configuration of the manipulator and obstacles, is developed based on the task priorities. The order of the task priorities are: to avoid collision, to plan the shortest path for the end effector, and to choose the minimum norm solution. The algorithm is fast and could be implemented on robotic manipulators for on-line trajectory planning.

REFERENCES

[1] Lozano-Perez, T. and Wesley, M.A., "Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," Comm. of ACM, Vol. 22, No. 10, Oct. 1979.

[2] Gouzenes, L., "Strategies for Solving Collision-Free Trajectories Problems for Mobile and Manipulator Robots," Int. J. Robotics Research, Vol. 3, No. 4, 1984.

[3] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," Int. J. Robotics Research, Vol. 5, No. 1, 1986.

[4] Gilbert, E.G. and Johnson, D.W., "Distance Functions and their Application to Robot Path Planning in the Presence of Obstacles," IEEE J. Robotics and Automation, Vol. RA-1, No. 1, March 1985.

[5] Klein, C.A. and Huang, C.H., "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators," IEEE Trans. Syst. Man Cyb., Vol. SMC-13, No. 3, March/April 1983.

[6] Majciejewski, A.A. and Klein, C.A., "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments," Int. J. Robotics Research, Vol. 4, No. 3, 1985.

[7] Lee, C.S.G., et. al., Tutorial on Robotics, IEEE Computer Society, 1983.

Task Planning and Control Synthesis for Robotic Manipulation in Space Applications

A.C. Sanderson, M.A. Peshkin, and L.S. Homem-de-Mello
Carnegie-Mellon University
Pittsburgh, PA 15213

1. Abstract

Space-based robotic systems for diagnosis, repair and assembly of systems will require new techniques of planning and manipulation to accomplish these complex tasks. ~~This paper summarizes~~ results of work in assembly task representation, discrete task planning, and control synthesis which provide a design environment for flexible assembly systems in manufacturing applications, and which extend to planning of manipulation operations in unstructured environments. Assembly planning is carried out using the AND/OR graph representation which encompasses all possible partial orders of operations and may be used to plan assembly sequences. Discrete task planning uses the *configuration map* which facilitates search over a space of discrete operations parameters in sequential operations in order to achieve required goals in the space of bounded configuration sets.

2. Introduction

Space-based robotic systems will be required to perform tasks involving dexterity, perception, and planning. Telerobotic systems integrate human perception and human planning capabilities in order to accomplish tasks. Autonomous systems will increasingly require imbedded task planning systems with accompanying sensory integration, control synthesis, and system architecture to support goal-directed activities in an uncertain environment. Diagnosis, repair, and assembly are tasks which will be essential to the maintenance of space-based systems and require both complex manipulation as well as reasoning about system configuration and system functionality. This paper reviews our recent work on task planning for assembly systems and discusses its implications for the development of robotic systems for assembly, maintenance, repair, and transport tasks.

Both manned and unmanned spacecraft require a variety of maintenance and repair tasks including materials handling, diagnosis of faults, reasoning about the origin of faults, hypothesis formation and testing, planning and executing repair procedures, disassembly, assembly, and replacement of parts. Currently these tasks may be accomplished in a limited way by on-site manual and teleoperated systems. As the number, complexity, cost, and importance of these spacecraft increases, autonomous systems which can provide service and maintenance on a routine basis will become essential.

Diagnosis and repair are problems in reasoning as well as manipulation. Any successful approach to these issues requires a representation of the task and an automated reasoning system which enables a decomposition of the problem into feasible sensing and manipulation procedures. Our work on assembly planning is based on several generations of assembly workcells which we built and demonstrated for manufacturing applications [1]. These flexible workcells incorporated multiple robot arms, vision, tactile and force sensing to accomplish tasks in electronic assembly, wire harness assembly, and assembly of instrument products such as copiers and printers.

Our experience with implementing tasks on these prototype workcells is the basis for current research on the development of tools for efficient design, programming, and implementation of complex systems. Task representation, decomposition, and sequencing [2,3,4], discrete task planning, [8] and adaptive control and learning techniques [9] are principal issues which are currently being addressed. Embedding such adaptation and learning procedures in the control and planning hierarchy is fundamental to successful implementation in uncertain environments. In this paper, we summarize an approach to assembly task representation and sequencing, and describe in more detail the use of the configuration map as a tool in discrete task planning.

The control functions of the system are allocated hierarchically into Strategic, Tactical, Operational, and Device levels. The control synthesis problem is to map the control hierarchy onto the set of feasible assembly plans in order to achieve desired performance. In this procedure, we seek to iteratively adjust the assignment of system resources subject to task precedence and configuration tolerance contraints. This procedure requires the definition of motion strategies and motion primitives which can be employed. We have developed a detailed understanding of sensorless manipulation strategies [5,6,7,8] which facilitate planning of sliding, pushing, and grasping operations. We are studying control structures for vision, tactile, and force feedback [9], and have demonstrated feasibility of adaptive control strategies for visual servoing. This work on sensor-based control is currently being extended to employ learning algorithms at the level of the motion primitive in order to improve performance by local adaptation in the face of uncertainty in the task environment. We have formulated an approach to quantitative description of task uncertainties using entropy methods [10], and have investigated the use of this *parts entropy* approach for planning strategies. We have also developed and demonstrated a new approach to arm signature analysis which improves the identification of kinematic models of manipulator structures and increases the resulting positioning accuracy [11].

Implementation of robotic systems in either a telerobotic or autonomous mode will require many of these planning, control, and manipulation capabilities. Task decomposition and control hierarchy have not been studied sufficiently for the telerobotic case. Development of motion primitives and planning of fine-motion strategies are important topics for research. The addition of adaptive and learning strategies to teleoperator systems is also important. The evolution of autonomous systems from telerobotic systems will require more effective models of human task planning strategies and task representation. The design of the components and tools of the space-based environment will depend on a consistent task representation which evolves to accept autonomous manipulation.

3. Assembly Task Representation

In our approach to assembly system design, [2,3,4], the planning of assembly of one product made up of several parts is viewed as a path search in the state space of all possible configurations of that set of parts. A syntax for the representation of assemblies has been developed based on *contact* and *attachment* relations. A decomposable production system implements the backward search for feasible assembly sequences based on a hierarchy of preconditions: (1) Release of attachments, (2) Stability of subassemblies, (3) Separability of subassemblies, including (a) Local analysis of incremental motion, and (b) Global analysis of feasible trajectories. Because there are many configurations that can be made from the same set of parts, the branching factor from the initial state to the goal state is greater than the branching factor from the goal state to the initial state. The backward search is therefore more efficient and corresponds in this case to the problem of *disassembling* the product using reversible operations. The resulting set of feasible assembly sequences is represented as an AND/OR graph and used as the basis for enumeration of solution trees satisfying system and performance requirements.

Figure 1 shows an example of an AND/OR graph representation of assembly sequences for a simple product with four parts. Each node in the graph corresponds to a subassembly and is described in the representation by a relational structure using the syntax of contacts and attachments. The hyperarcs correspond to the disassembly operations, and the successor nodes to which each hyperarc points correspond to the resulting subassemblies produced by the disassembly operation. For most products, the assembly operations usually mate two subassemblies, and the resulting hyperarcs are typically 2-connectors as in this example.

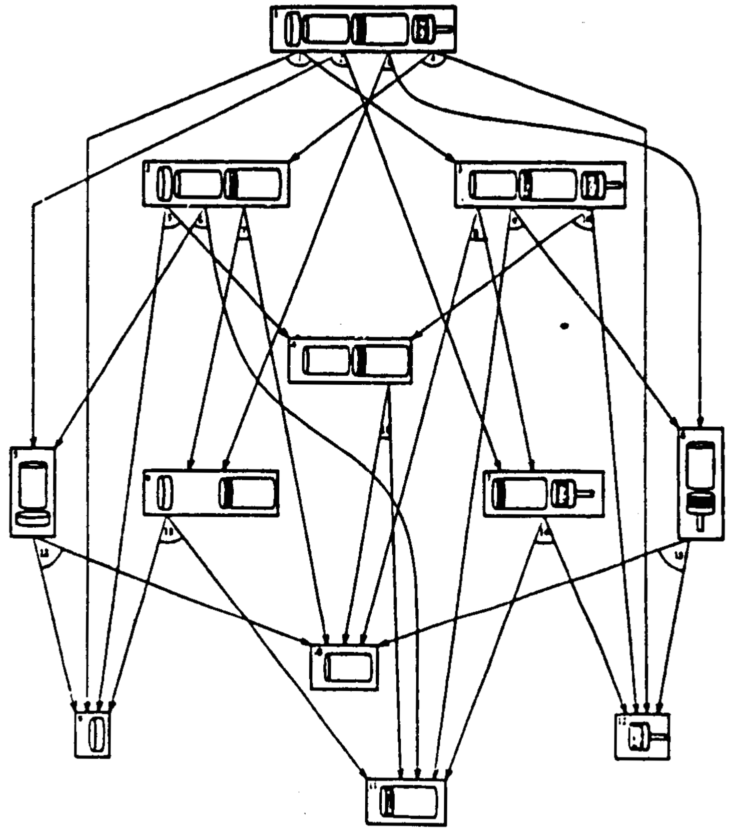


Figure 1. AND/OR graph representation of assembly plans for a simple product.

A solution tree from a node N in an AND/OR graph is a subgraph that may be defined recursively as a subset of branching hyperarcs from the original graph. The AND/OR graph representation therefore encompasses all possible partial orderings of assembly operations. Moveover, each partial order corresponds to a solution tree from the node corresponding to the final (assembled) product. The AND/OR graph representation therefore permits one to explore the space of all possible plans for assembly or disassembly of the product. The problem of selecting the best assembly plan may therefore be viewed as a search problem in the AND/OR graph space, and for some given evaluation function on the graph, generic search algorithms such as AO* [12] may be used. In practice, the development of such an evaluation function is very difficult since it would often depend explicitly on implementation issues such as choice of devices and underlying control strategies. We have explored the assignment of weights to hyperarcs using criteria of (a) operation complexity, and (b) subassembly degrees of freedom, or parts entropy [10]. Such an approach is viewed as a preliminary search procedure which may narrow the search space for later detailed examination using implementation details. In the simple examples studied, the resulting ranking of candidate assembly sequences was consistent with intuitive assessment of complexity.

The representation of assembly plans is particularly important for systems which do online planning or scheduling. Previous studies of online planning problems [13] have used discrete sequence representation or precedence diagrams of operations. In the precedence diagram formalism, typically no single partial order can encompass every possible assembly sequence. The AND/OR graph represents all possible partial orderings of operations, and each partial order corresponds to a solution tree from the node corresponding to the final product. We have illustrated the use of the AND/OR graph for online scheduling of a simple robotic workstation with random presentation of parts [2]. The resulting analysis showed a relative improvement in efficiency (number of operations required) from fixed sequence operation of 6% for precedence diagrams and 18% for the AND/OR graph. The principal advantage in this example was the reduced need for buffering and corresponding retrieval of parts.

The AND/OR graph representation provides a framework for the planning and scheduling of operations sequences. The problems of testing, disassembly, repair, and assembly all benefit from a unified representation which encompasses partial ordering of procedures. Preliminary search of the task space may reduce the candidate subtrees substantially, but the development of final plans typically involves directly the implementation and specification of the underlying devices and motions. In the next section we describe a tool for discrete task planning which facilitates exploration of alternative sequences of operations at the level of parts configurations.

4. Discrete Task Planning

A sequence of assembly or disassembly subtasks is implemented by performing operations on the parts using system resources such as robot hands, fixtures or sensors. The allocation of these resources and the synthesis of control programs to coordinate them must be developed in a second level of planning. In general, such operations require detailed motion planning of individual devices and is extremely difficult. In this section, we describe a definition of discrete operations which lend themselves to planning through manipulation of the configuration map relating input and output configuration states.

Any subtree of the AND/OR graph may be thought of as a *subtask precedence graph*,

and each branch of the subtask precedence graph defines a process in the configuration space of the parts. An assembly operation can then be defined by:

Assembly Operation:

> **Given $c^{o} = (C_i^{o}, C_j^{o}) \in C$,**
> **control manipulation, sensing, and computation**
> **to achieve $c^{f} = (C_i^{f}, C_j^{f}) \in T$, then**
> **execute operation,**

where T = tolerance set,

> **$T \subseteq C = C_i \times C_j$ for entities i, j,**

is the set of configurations (region of configuration space [14]) for which an operation on i,j can be successfully performed.

This definition emphasizes the basic problem in assembly as the control over configuration uncertainty in order to meet tolerance requirements of successive operations. While it is possible to define probability distributions over configurations of parts, in practice, it is very difficult to accurately estimate such distributions, and it is cumbersome to propagate the effect of such distributions through successive operations in a sequence. The configuration map used here provides a tool to compute the effect of operations on bounding sets of configuration points.

A *bounding set* B(v) is defined as

> B(v) = {possible outcomes of v}

where v is a bounded variable. We can define in turn:

> Joint bounding set: $B(v_1, v_2, \ldots, v_n)$
>
> Conditional bounding set: $B(v_1|v_2=\eta) = \{v_1|(v_1,\eta) \in B(v_1,v_2)\}$
>
> Sum of bounding sets: $A + B = \{v|v = a+b \text{ for } a \in A, b \in B\}$
>
> Scalar multiplication: $cA = \{v|v = ca \text{ for } a \in A\}$.

An operation which alters the configuration of a part may be described by a mapping between the initial configuration, Θ_i, and the final configuration Θ_f. An operation with a unique mapping occupies a single point in (C-space x C-space) and completely defines the change in configuration state of the system. In this case, planning of operations reduces to planning of unique trajectories in configuration space. As discussed above, such unique mappings are often of limited use due to the uncertaint y in configurations and the finite tolerance of operations. Then, states of the objects may be described by bounding sets of points in the configuration space.

The *configuration map* $M(A_i, B_f)$ describes a single operation which maps a bounded set of input points to a bounded set of output points:

$$M(S_1, S_2) : \{S_1\} \rightarrow \{S_2\}.$$

The configuration map takes on logical values in (C-space x C-space) where each logical '1' defines a feasible mapping. The configuration map for a rigid part is a function of twelve dimensions, although in many cases these degrees of freedom are not of equal interest.

The usefulness of the configuration map representation of operations lies in the ease of combining sequential operations. An operation $M_1(\Theta_i,\alpha)$ followed by an operation $M_2(\alpha, \Theta_f)$ is defined as:

$$M_{12}(\theta_1, \theta_f) = M_1M_2 = \cup_\alpha\{M_2(\alpha, \theta_f) \cap M_1(\theta_i, \alpha)\}.$$

Sequences of alternative operations may therefore be compared using simple relations.

The configuration map is particularly useful in cases where inputs and outputs may be partitioned into bounded sets. If we identify N subintervals B of the output space and N subintervals of A of the input space, then a symbolic mapping:

$$M' = \cup_j\{A_j \times B_j \mid M(\theta_i, \alpha) > 0 \}.$$

defines bounded regions of the configuration map associated with transformations of bounded sets due to a given operation. A useful instance of the bounded set map occurs when we let:

$$A_j = \cup_{\alpha \in B_j} \{\theta_i \mid M(\theta_i, \alpha) > 0 \}.$$

Then the configuration map

$$M' = \cup_j A_j \times B_j$$

is *rectangular* and the operation is completely defined by the symbolic map and the definition of the underlying sets.

The product of rectangular configuration maps is completely defined by bounding set operations:

$$M_1M_2 = \cup_j {}^2B_j \times \{\cup_{k \in {}^{21}C_j} {}^1A_k\}$$

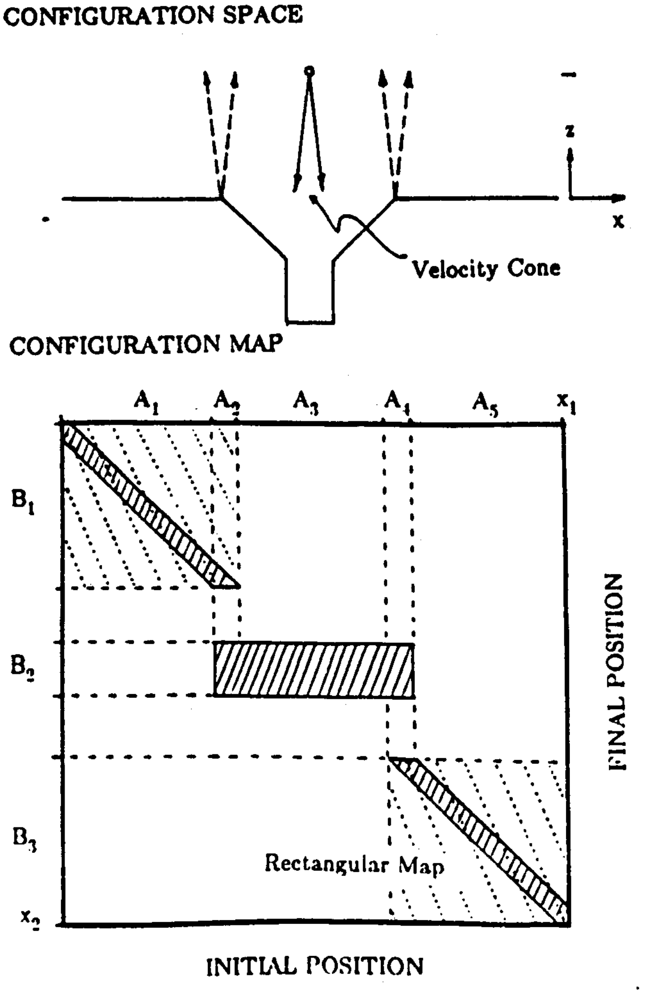
where

$${}^{21}C_j = \{k \mid {}^2A_j \cap {}^1B_k \neq \emptyset \}.$$

is the resulting configuration map product.

Figure 2 shows an example of a peg insertion operation in two dimensions, This type of problem has been studied from the point of view of trajectory planning in configuration space [15]. The configuration map shown in figure 2 is derived from such a trajectory analysis and summarizes the input-output relations in a manner which permits the resulting discrete operation to be integrated into task plan. A different configuration map is developed for each set of discrete operations parameters, and the ability to form configuration map products permits search over the space of operations sequences. In figure 2, the x position of the peg is regarded as the independent variable of the map, and the initial z-position of the peg is fixed for a given configuration map. The operation moves the peg in a -z direction using a compliant move and directional uncertainty represented by the velocity cone [16].

The resulting configuration map in figure 2 has three output bands corresponding to successful insertion, miss-to-the-left, and miss-to-the- right. These three bands occur consistently for different parameter values. Five input bands may then be reconstructed and labelled defining a partitioning of the input configuration space. The resulting map may be 'rectangularized' as shown by the dotted areas, and in that form the symbolic mapping provide a complete description of the operation and a basis for search procedures.


CONFIGURATION SPACE

CONFIGURATION MAP

Figure 2. Configuration map for peg in hole example.

An example of a product of configuration maps is shown for a different set of operations in figure 3. Each of these maps is derived from our analysis of sliding objects [5,6,7] and corresponds to the orientations of a polygonal object being pushed by a two-dimensional fence of finite length. Equivalently, the object may be moving on a conveyor belt past a fixed fence. The independent variable in each map is the object orientation while the operation parameter is the fence angle. The uncertainty represented by the finite width bands in the maps is a result of the unknown support distributions of the objects. In [5,6,7] we derived bounds on the rates of rotation of such objects and have used these to compute the configuration maps for this example. The product of configuration maps therefore defines the bounds on the sets of orientations resulting from successive fence pushing operations, and can be used as a planning tool for designing sequences of fence push operations to achieve required goals.

For discrete tasks, the space of all operations sequences may be represented by a tree. Arcs correspond to operations, and each node represents a set of possible configuration states after execution of all the operations on the path from the root to that node. Figure 4 illustrates one such tree which corresponds to sequences of fence pushing operations for fences of different angles operating on the object shown in figure 3. The possible configurations of a part at a given node are obtained by multiplying the configuration maps for the operations on the path from the root to the node. Traversing the tree in order to search it is facilitated by the ease with which products of multiple configuration maps can be computed using the code sets.

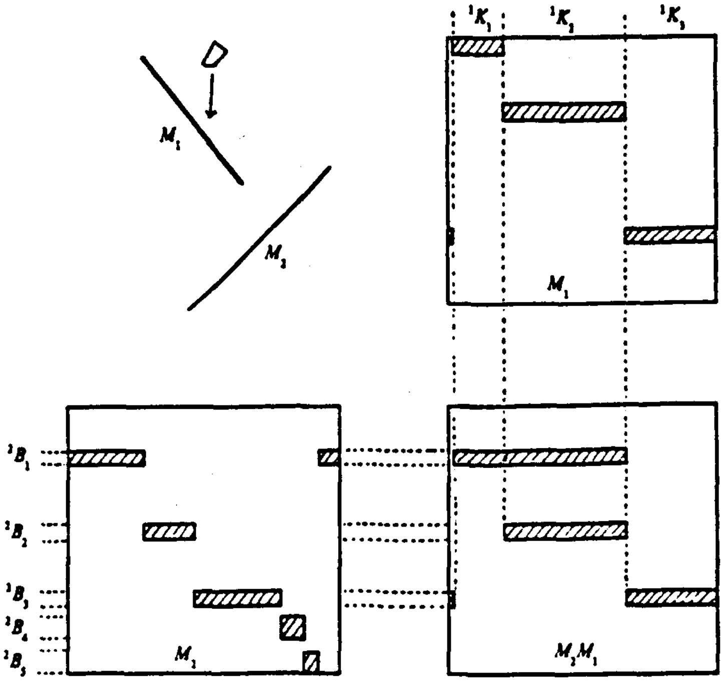


Figure 3. Product of two configuration maps.

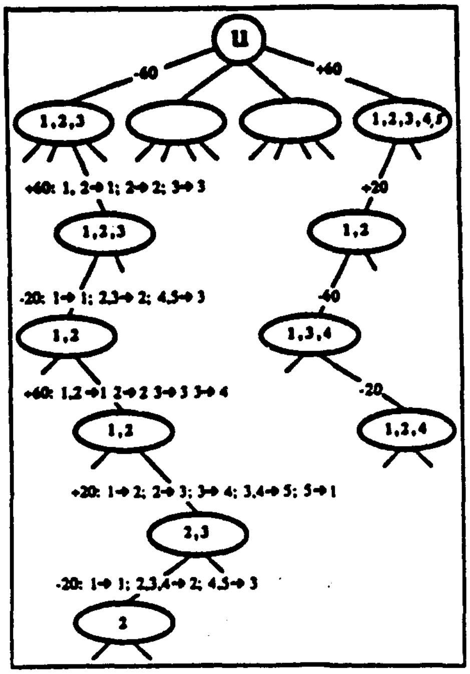


Figure 4. Tree search for operations sequence.

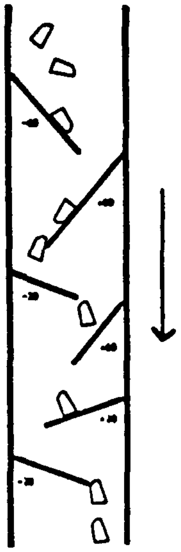


Figure 5. Resulting sequence of fence push operations.

Each node is labelled with the subset of the indices j of B for the bands B for the fence angle of the preceding arc. The goal of this task was to reduce the set of possible configurations to a narrow range of orientation, and a search strategy was implemented to reduce the number of output bands to one using the minimum number of operations.

Searching this tree of discrete operations exhaustively is computationally difficult due to the high branching factor which results from the available set of fence angles at each step. Two techniques have been developed to make this search feasible. First, there are systematic relations among bands for different operations parameters. Since there are only a few distinct code sets for the output arcs, it is often possible to systematically choose the subset of arcs which need to be followed among these outputs. Second, branches of the tree which develop code sets which have occurred previously in a shorter route may be pruned during search.

Implementation of these search techniques permits solution of the fence sequence design problem with the resulting design shown in figure 5. This parts feeder design will align parts with the geometry shown in figure 3 independent of the input orientation. Bounds on the orientation of the resulting single band are also derived from the procedure. The output part is then aligned for acquisition or handling by a robot. Computation for this search problem requires a few seconds of computation.

5. Conclusions

In this paper, we have reviewed several results in assembly representation, discrete task planning, and their relation to underlying control strategies. These methods of planning and manipulation are important for applications which will require autonomous systems to carry out complex tasks in diagnosis, repair, and assembly in space. The development of such analytical tools and their demonstration in prototype systems will be an important part of the evolution of telerobotic and autonomous systems for space applications.

REFERENCES

[1] A. C. Sanderson and G. Perry, "Sensor-based Robotic Assembly Systems: Research and Applications in Electronics Manufacturing," *Proceedings of the IEEE*, Special Issue on Robotics, Vol. 71, No. 7, July, 1983.

[2] L. S. Homem-de-Mello and A. C. Sanderson, "AND/OR Grapa Representation of Assembly Plans," *Proc. 1986 AAAI Conference on Artificial Intelligence*. August, 1986, pp 1113-1119.

[3] A. C. Sanderson and L. S. Homem-de-Mello, "Task Planning and Control Synthesis for Flexible Assembly Systems," *Proc. NATO Int. Advanced Research Workshop on Machine Intelligence and Knowledge Engineering for Robotic Applications*, May, 1986.

[4] B. H. Krogh and A. C. Sanderson, "Modeling and Control of Assembly Tasks and Systems," *CMU Robotics Institute Technical Report*, CMU-RI-TR-86-1,] 1986.

[5] M. A. Peshkin and A. C. Sanderson, "The Motion of a Pushed, Sliding Object, Part 1: Sliding Friction," *CMU Robotics Institute Technical Report*, CMU-RI-TR-85-18, 1985.

[6] M. A. Peshkin and A. C. Sanderson, "The Motion of a Pushed, Sliding Object, Part 2: Contact Friction," *CMU Robotics Institute Technical Report*, CMU-RI-TR-86-7, 1986.

[7] M. A. Peshkin and A. C. Sanderson, "Robotic Manipulation of a Sliding Object," *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, April, 1986,pp 233-239.

[8] M. A. Peshkin and A. C. Sanderson, "Planning Sensorless Robot Manipulation of Sliding Objects," *Proc. 1986 AAAI Conference on Artificiai Intelligence.*, August, 1986, pp. 1107-1112.

[9] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic Sensor-based Control of Robots with Visual Feedback," *IEEE Journal of Robotics and Automation*, in press, 1986.

[10] A. C. Sanderson, "Parts Entropy Methods for Robotic Assembly System Design," *Proc. IEEE Int. Conf. on Robotics and Automation*. March, 1984, pp. 600-608.

[11] H. W. Stone, A. C. Sanderson, and C. P. Neuman, "Arm Signature Identification," *Proc. IEEE Int. Conf. on Robotics and Automation*. April, 1986, pp. 41-48.

[12] N. J. Nilsson, *Principles of Artificial Intelligence*. Springer-Verlag, 1980.

[13] B. R. Fox and K. G. Kempf, "Opportunistic Scheduling for Robotics Assembly." 1985 IEEE International Conference on Robotics and Automation, pp. 880-889, 1985.

[14] T. Lozano-Perez, "Spatial Planning: A Configuration Space Approach." IEEE Transactions on Computers C-32, 2 (February 1983), 108-120.

[15] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots" Int. Journal of Robotics Research 3,1 (Spring, 1984), 3-24.

[16] M. Erdmann, "Using Backprojections for Fine Motion Planning with Uncertainty' Int. Journal of Robotics Research 5,1 (Spring, 1986), 19-45.

S14-63
12P.

188183

Using Automatic Robot Programming for Space Telerobotics

E. Mazer, J. Jones, A. Lanusse, T. Lozano-Perez, P. O'Donnell, and P. Tournassoud
Massachusetts Institute of Technology
Cambridge, MA 02139

MJ 70[illegible]
Ar-N

1 Abstract

is described

~~This paper describes~~ the interpreter of a task level robot programming system called Handey. Handey is a system that can recognize, manipulate and assemble polyhedral parts from given only a specification of the goal. To perform an assembly, Handey makes use of a recognition module, a gross motion planner, a grasp planner, a local approach planner and is capable of planning part re-orientation. The possibility of including these modules ~~in~~ a telerobotics work-station is discussed.

2 Introduction

The projected increase in the use of robots in space will make their increased autonomy essential. Direct teleoperation of robots in complicated, repetitive tasks, such as those found in space, can be very tedious. Robot autonomy would relieve the operators from unnecessary fatigue as well as improving reliability and cost [1].

One step towards improving the autonomy of robots consists of having a system capable of planning simple grasp and assembly operations. This goal seems to be a fairly simple and short term objective and yet, it has only been achieved for very well structured environments. The early research on automatic planning of robot operations [2,3,4] focused exclusively on simple situations involving completely modeled environments.

More recent work has now made it possible to design systems working in much more general environments, including environments with significant uncertainty. These environments resemble the type of environment one can expect to find in the vicinity of a space station. This type of environment can include complex parts and six degree-of-freedom revolute arms, all of it modeled with a CAD system. In this paper we describe Handey, a new system that embodies many of the fruits of this more recent research. While Handey still makes strong assumptions about its environments and tasks, we believe that these assumptions are realistic enough so that Handey can contribute towards improving the tele-operation of manipulators.

3 Handey overview

Handey is a task-level [5] robot programming interpreter, that is, the commands given to the system are not robot motions or gripper operations as in VAL or LM [6,7] but simply by describing a certain desired state of an assembly. For example, a full sequence of robot motions, gripper operations and sensor calls can be replaced in a task-level robot programming system by a single statement: *PLACE PART A on PART B*. The interpreter is responsible for planning and carrying out the detailed motions and other actions which lead to this assembly. In its current stage of development, however, Handey makes use of the "perfect world" hypothesis and so, does not take in account problems related to uncertainty or unexpected events. For example Handey does not

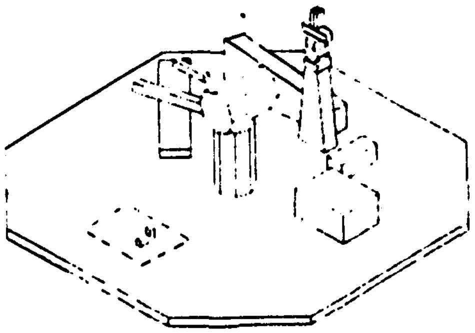


Figure 1: Experimental setup

include a compliant motion planner which would plan assembly strategies in the presence of uncertainty [8,9,10], and does not provide program veiification techniques based on uncertainty propagation to patch a predefined plan [11,12]. This remains as future work.

Figure 1 represents a scene used during the development of the system, the doted line on the table shows the limits of the field of view of the laser range finder (this area is called the V-area in the rest of the paper).

Part A is assumed to be located in the V-area. Nothing is assumed concerning this area: part A can be in any location and it can be partially obscured from the range finder by other objects. Except for part A it is not necessary for parts entirely located in the V-area to be modeled in the CAD system. The location of part B is assumed to be known, as are the locations of obstacles in the workspace outside of the V-area, such as the laser-camera device, which we plan to use in the future to find the exact relative position between the gripper and the part.

The user describes the final assembly with a set of relationships between geometric features of parts A and B, then he activates the interpreter. The following is a typical sequence performed by the interpreter to plan the detailed motions and operations necessary to achieve the assembly.

Determining the Final Location. Based on the specified symbolic geometric relationships between parts A and B, a geometric transform representing the relative location between A and B after the assembly is computed.

Recognizing and Localizing Part A. A model-based vision algorithm is executed to determine the location of part A.

Planning a Grasping operation. The grasp planner first tries to find a grasp which permits placing part A directly on part B. If this is not possible, a regrasping operation will be planned later.

Planning the first gross motion. A collision-free path is planned from the initial position to a point on the boundary of the V-area.

Planning a collision free approach. Since the scene in the V-area is not modeled in the CAD system, it is not possible to find a collision free path by the method used for gross-motion planning. Another planner, using the data provided by the range finder, plans a path for the gripper among the obstacles which are located in the V-area.

Planning re-orientation. In many occasions it is not possible to grasp and to assemble the part keeping the same relative position between the part and the gripper. In this case a regrasp operation is planed in a obstacle free portion of the work space.

Planning the remaining gross motions. The regrasp planner produces a number of intermediary locations to be reached by the robot, during the re-orientation phase, this phase computes all the paths necessary the perform the regrasping and the path to the final destination.

4 Functional description of Handey

Handey is composed of several modules, most of these modules correspond to substantial pieces of code.

4.1 Experimental Environment

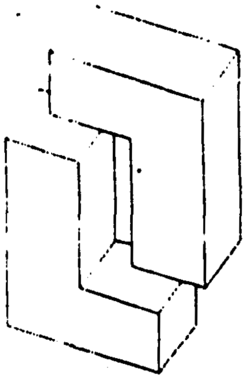
The hardware-dependent software primitives provide a way for the planner to ignore the details needed to operate real-world equipment. It is crucial that these modules be quite good, since they determine the overal system's precision in localizing and moving parts and, as a consequence, will determine the success of the experimentats. Handey makes use of a very limited number of such hardware dependent primitives.

- Range finder calibration: this primitive eliminates non-linearity due to the technology of the laser sensor and scanner hardware. It also determines the scale factors between the sensor and the model.
- Robot Vision calibration: this primitive determines accurately the relative location between the reference frames associated with the robot and the range finder.
- Depth map acquisition: this primitive activates the range finder and returns a depth map in standard units.
- Joint motion: in its current version Handey makes use of one robot motion primitive: "MOVE-JOINT TO (q1,q2,.....q6)" to command a coordinated motion of the robot. The gripper is operated in a binary mode.

4.2 The world modeling system

The world modeling system is used to construct polyhedral models of the parts involved in the assembly including the obstacles (table, ceiling, etc.) and the robot. It is also used to maintain a model of the world during the planning. Once geometric models have been created it is possible to use the following primitives to create, modify or interpret a scene:

- assign a location to a part,
- express the location of a part in a different reference frame,
- affix and unfix parts from the gripper of the robot



- **Face A of Part A is Parallel to Face A of Part B**
- **Face C of Part A is Against Face C of Part B**
- **Face B of part A is Against Face B of Part B**

Figure 2: Describing the final assembly

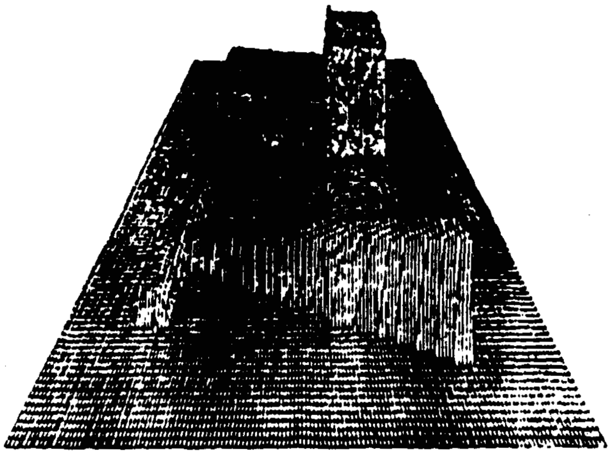


Figure 3: Depth-map produced by the laser range finder

4.3 Computing the final location of part A

The user can describe symbolically the next assembly step by specifying a set of geometric relationships which should hold between geometric features of part A and geometric features of the sub-assembly. Figure 2 represents the set of relationships used to describe the final location of part A.

The set of geometric relationships is then translated into a set of algebraic equations [13]. The system then solvesthe the set of equations to compute the relative position between the part and the sub-assembly. At the present time this feature is not integrated into the on-line version of Handey but works separately.

4.4 Locating part A

The range finder is activated and produces a depth map (figure 3). The map is then processed as if it was an image except that the brightness corresponds directly to the elevation above the work table.

A standard edge operator [15] is run over the image and extended linear segments are identified in the resulting array. Note that this process identifies 3D edge segments,

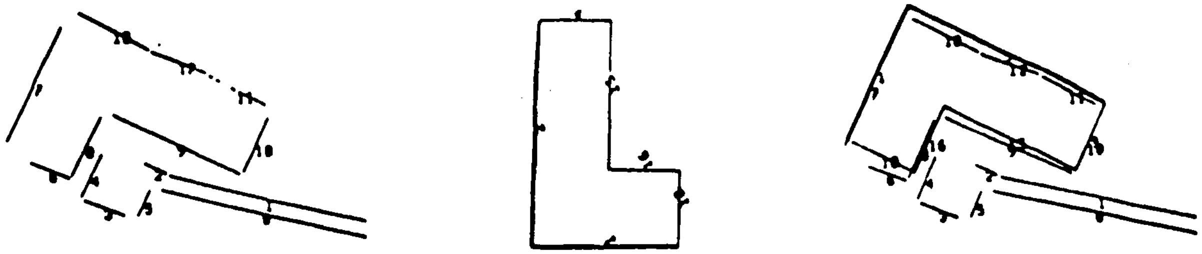


Figure 4. Matching model edges with scene edges

not just their projection in an image. The method used for object localization is a simple hypothesize-verify algorithm based on matching linear segments in the depth map to edges in the polyhedral model of the part. This method is a variation of the method described in Lozano-Pérez and Grimson [16] using edge data instead of face data. Figure 4 represents one matching between edges of the scene and edges of the model.

4.5 Planning collision-free motions

At a number of points in the operation of the system, a collision-free path is required from one specified location to another. Handey uses a simplified version of the path planner described in Lozano-Pérez [17]. This path planner uses the robot's joint space as the configuration space. The version of the path planner used by Handey never computes configuration spaces of dimension greater than three, but it allows motions requiring six degrees of freedom. Essentially, we assume that a path from the start to the goal exists such that the last three joints of the arm retain their starting values until some intermediate point where they change their values at the goal and never change after that. It is easy to construct cases where this assumption will fail, but it works in a large percentage of actual cases.

The actual planning proceeds as follows: An approximate arm model is built in which the last three joints are replaced by a box. This box must be large enough to enclose the last three links, the hand and any object in the hand, not only at their start and goal position but also at the intermediate positions between the two. The three-dimensional configuration space for this model can then be built. We find the closest free point in this configuration space to both start and goal positions, a path is then found between these two free points. Note that the complete robot is guaranteed to be safe along this path, for the whole range of values of the last three joints between the start and the goal. Therefore we can simply interpolate the values of the last three joints between the start and the goal values. Then, we plan a path using the original model of the robot between the free point and the start point itself. We also plan a path from the free point closest to the goal itself. In these two paths the value of the last three joints are fixed. The concatenation of these three paths form the desired path. Figure 5 represents the path found the final motion.

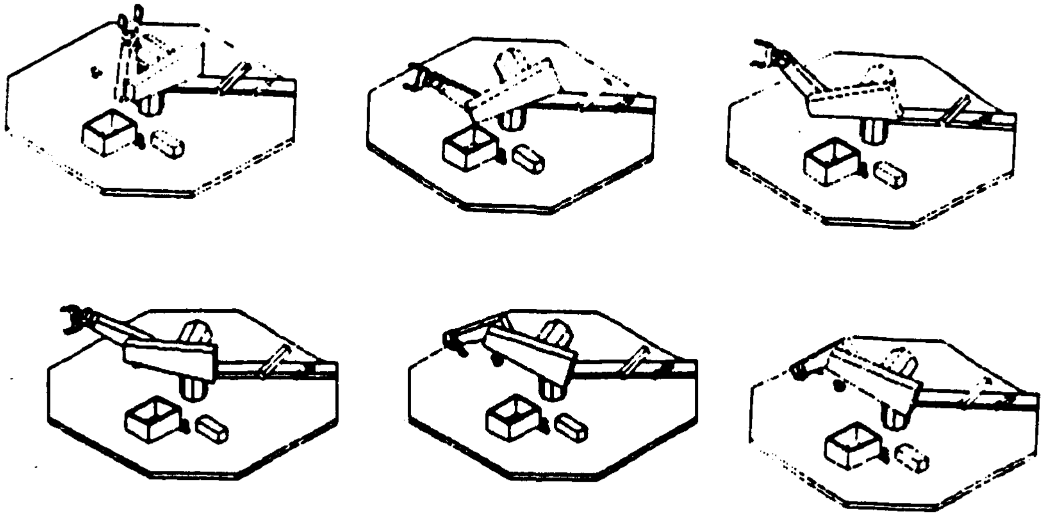


Figure 5 Example of a path generated by the path plann-r

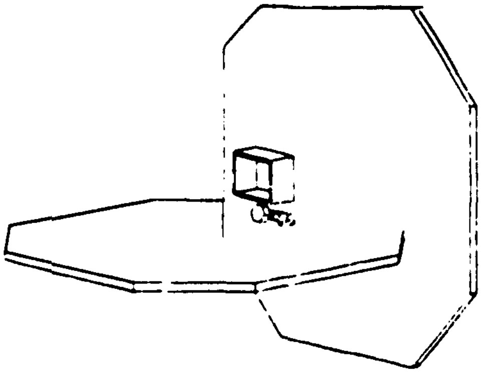


Figure 6 Back-projection of parts

4.6 Grasping

Once the part has been located Handey chooses a grasp. This operation has to take in account several constraints:

- the grasp should be stable,
- a path should exist inside the V-area to reach the grasp,
- the grasp should permit assembling the part once it is in the gripper.

The last constraint can be satisfied by "back projecting" all the obstacles in the V-area. After this operation virtual obstacles exist in the V-area, these obstacles have the same relative location with part A that the real obstacles will have in the final sub-assembly. If one can find a grasp in this environment then it is guaranteed than the grasp will permit assembling the part (figure 6).

4.6.1 Finding a stable grasp

In its current version, Handey uses a grasp planner designed for a gripper equipped with parallel jaws. A future version of Handey will include a more sophisticated planner designed for the three fingers JPL-Stanford-MIT hand [14]. Currently, the planner

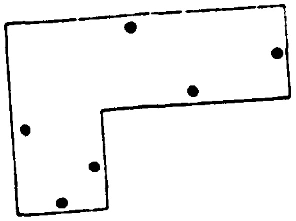


Figure 7 Grasp-points associated with one face

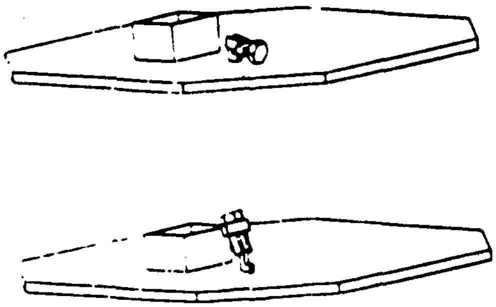


Figure 8 Angular range associated with a grasp

associates two grasps for each locally convex edge of the model. A grasp is defined by one of the face adjacent to the edge and a grasping point. The grasping point is located on the face at a prespecified distance from the edge. Figure 7 represents all the grasping points of one face of part A.

To be valid a grasp should satisfied three conditions:

1. a parallel face should exist and should permit a grasp (mutual visibility [18]) with an allowed distance between the two faces,
2. The gripper should be capable of some rotation around the grasping point (grasping range),
3. an inverse kinematic solution should exist at the grasping point.

The grasping range can be computed using a submodule of the path planner. The grasps are sorted with such grasp permitting the most vertical approach. Figure 8 represents the gripper into two end-points of an angular range.

4.6.2 Planning motions in the V-area

Since no information on obstacles exists in the world-modeling system for the V-area, we must take in account the presence of objects reflected in the depth map. For this purpose we use a planner specialized for planning the motion of the hand in the grasp plane. The grasp plane is a plane parallel and at equal distance from the two faces defining the grasp. When approaching a grasp the fingers remain parallel to the grasp

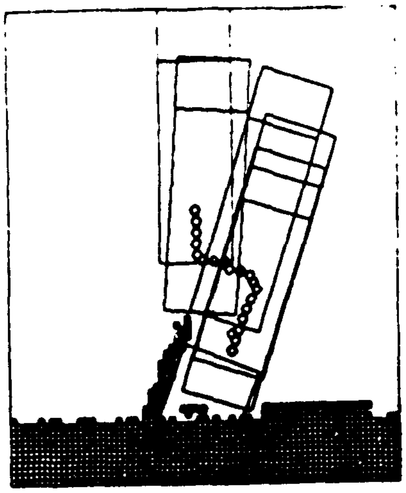


Figure 9: Planning a path in the grasp plane

plane and centered about it but, otherwise, are free to rotate and translate in the plane. Obstacles are projected into this plane to reflect the possibility that they collide either with one of the two fingers, the cross-piece of the hand, or the force sensor.

The planner uses a method loosely modeled on the potential field method for obstacles avoidance [19]. The goal of the grasp planner is to bring a gripping point located between the fingers as close as possible from the g.:asping point without colliding. The grasping point attracts the gripping point of the grippₜr while projected obstacles on the grasping plan repel the boundaries of the projected gripper parts (fingers, cross-piece and force sensor). These pseudo-forces are combined in such a way that the gripper is guided toward the goal in X,Y,Θ on the grasp plane. The initial position and orientation of the gripper is given by the grasping planner. Figure 9 represents the evolution from the initial position toward the goal.

4.7 Regrasping

Back-projected objects are artificially added to the depth map so that the final grasp will also permit the assembly. However this may constrain the problem so much that a feasible solution cannot be found. Handey will backtrack among sorted grasps a limited number of times before giving up and trying to find a solution with regrasping. The V-area path planner is then called without back-projected parts and a regrasping operation is planned.

For each part the grasp planner uses two data structures: placements and grasps.

1. A *placement* is a way of placing the part at a particular location on the work table. This location is chosen in an area known to be free of any obstacle, the regrasping will take place in this area. A parameter φ is associated with each placement P. Changing this parameter corresponds to rotating the part on the table around a vertical axis. All the placements P_i are computed automatically by computing the stable faces of the convex hull of the part.

2. A *grasp* is defined by a parameter θ associated to each grasp G. Changing this parameter corresponds to rotating the gripper along an axis perpendicular to the

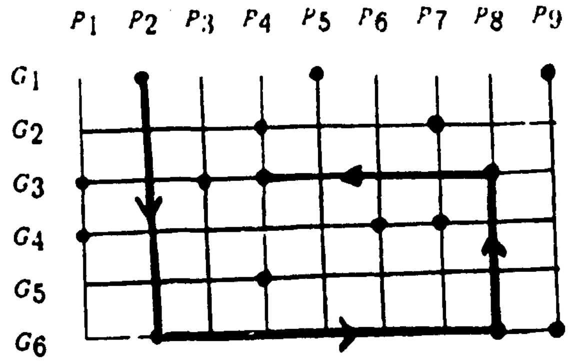


Figure 10: Finding successive placements and grasps

grasping face and containing the grasping point. The set of grasps G_j is also computed automatically.

In order to plan a regrasping operation it is necessary to compute all the vertices of the "regrasping graph". A vertex consists of a data structure defined by a pair $P_i G_j$ having a non-empty $\theta\varphi$ map. A map is built by sampling φ and θ over the interval (0.0,2π). Each φ, θ specifies a single position of the gripper. To be valid, a pair should correspond to a position of the gripper where a solution of the inverse kinematic exists. The map is the set of all the valid $\theta\varphi$ pairs.

There are two operations necessary to perform a re-orientation.

1. Moving from one placement P_i to another P_k. This is possible when a grasp G_j exists, such that the maps associated with $P_i G_j$ and $P_k G_j$ have at least one valid $\theta\varphi$ pair.
2. Changing from one grasp G_j to another grasp G_k. This is possible when a placement P_i exists such that the map associated with $P_i G_j$ and $P_i G_k$ has at least one valid $\theta\varphi$ pair.

The regrasp planner is given an initial position of the part inside the gripper and a final G_f, φ_f grasp which permits the assembly operation. The goal of the regrasp-planner is to find a way through various placements and grasps between the initial and the final grasps. This is represented in figure 10. Horizontal arcs in the figure represent motions of the part from one placement to another and vertical arcs represent motions of the gripper to change the grasp.

5 Applications to Telerobotics

Using telemanipulators in earth orbit has long been recognized as a difficult task. The trend has been to increase the level of commands available to the operator [20]. Prototype telerobotics workstations have been built integrating such high level teleoperation commands. For example, hybrid-control permits the computer to maintain a drill on a given axis while the operator can concentrate controlling the force necessary to perform the drilling operation.

Based on our experience we believe that it is not unrealistic to add some of the capabilities of Handey to such a work-station. As explained in 4.1 the number of primitives used by Handey is fairly limited and should be available in such a workstation anyway.

The most limitating factor being the possibility of including a range finder on the mobile remote servicer (RMS). Once integrated, one can imagine to use a system such Handey in three modes.

- **autonomous mode:** The operator describes the next assembly step. The system computes the sequence of operations and sensor calls to perform the assembly a graphic simulation is presented to the operator before actual execution.
- **partially automatic mode** In this mode the operator asks the system to plan certain portions of the assembly, for example, the system can plan the trajectory to align two axes so that a drilling operation can take place under the control of the operator.
- **monitoring** In this mode the operator first describes what result he expects to achieve.' The system monitors the task and sends an alarm when it detects that the present configuration of the system makes it difficult or impossible to reach the goal. For example grasping a part in a way that the final assembly or an intermediate path is difficult or impossible.

6 Conclusion

Watching Handey performing an assembly is always astonishing and fun, no operator would ever program a task the way our system does. Potentially, we believe that future versions of Handey could be more efficient in performing assembly tasks than typical operators. It could plan paths more effectively in term of time, energy, and safety. It would be less likely to make a mistake such as grasping a part and not being able to move it at a later stage of the assembly because of mechanical stops or collisions. Handey is based on well-establish geometric principles which can make it a robust system. For this reason, it is possible to think of the Handey interpreter as a target system for higher-level planners. The current Handey implementation on a Lisp Machine is still quite slow; it takes approximately 10 min to plan a single pick and place operation. But, we believe that is possible to reduce this time significantly simply by reimplementing it in a machine with fast floating point hardware.

Telerobotics is often presented as feasible alternative to an infeasible autonomous robot. This is certainly true at the present time, but the contrary may be true in the future, that is, the technology necessary to achieve good tele-presence may be more sophisticated than the technology necessary to provide on-board intelligence and dexterity.

7 Acknowledgments

This work was founded primarily by the Office of Naval Research under contract N00014-85-K-0214. Additional support was provided by an NSF Presidential Young Investigator Award (Lozano-Pérez) and the French CNRS (Mazer).

References

[1] D.L. Akin, M.L. Minsky, E.D Thiel and C.R. Kurtzman. *Space Applications of Automation, Robotics and Machine Intelligence System (ARAMIS)- Phase II*. Na-

tional Aeronautics And Space Administration Contractor Report 3734, Volume 1. October 83

[2] L.I. Lieberman, and M.A Wesley. *AUTOPASS: an automatic programming system for computer controlled mechanical assembly*. IBM Journal of Research Development. July 1977.

[3] T.L. Lozano-Pérez. The Design of a Mechanical Assembly system, MIT Artificial Intelligence Laboratory, TR 397, December 1976.

[4] R.J. Popplestone. A.P. Ambler and I.M Bellos. *An Interpreter for a Language for Describing Assemblies*, Artificial Intelligence 14, 1980.

[5] J.C Latombe. *A Survey of Advanced Software for robot manipulators"*, AICA congress, Padoue, October 1982.

[6] Unimation, Inc. *User's guide to VAL, a robot programming and control system*, Unimation Inc. February 1979

[7] J.C. Latombe and E. Mazer. *LM: a High-level programming language for controlling manipulators*, 11th International Symposium on Industrial Robots, Tokyo, October 1981.

[8] M.A. Erdmann. *On Motion Planning with Uncertainty*, MIT, Artificial Intelligence Laboratory, TR 810, 1984.

[9] B.R Donald. *A theory of Error Detection and Recovery: Robot Motion Planning with Uncertainty in the Geometric Models of the robot and Environment*, International Workshop on Geometric Reasoning, Oxford University, England July 1986.

[10] S.J Buckley. *Planning and Teaching Compliant Motion Strategies* PHD Thesis, MIT Artificial Intelligence Laboratory, January 1987.

[11] R.A. Brooks. *Symbolic Error Analysis and Robot Planning*, International Journal of Robotic Research, Vol 1, no. 4 December 1982

[12] J. Pertin and P. Puget. *Controle dans le Système de Programmation Automatique Sharp, Gestion des contraintes d'accessibilité et d'incertitudes* IMAG TR 615, June 1986.

[13] E. Mazer. *Geometric Programming of Assembly Robots LM-GEO*, International Meeting on Advanced Software in Robotics, Liège, May, 1983.

[14] N.G. Van-Duc. *The Synthesis of Stable Force-Closure Grasps*, MIT Artificial Intelligence Laboratory, TR 905, May 1986.

[15] J.F Canny *Finding Edges and Lines In Images*, MIT Artificial Intelligence Laboratory, TR 720 May, 1983.

[16] W.E.L. Grimson and T. Lozano-Pérez. *Recognition and Localization of Overlapping Parts from Sparse Data*, MIT Artificial Intelligence Laboratory, A.I Memo 841 June, 1985.

[17] T. Lozano-Pérez. *Motion Planning For Simple Robot Manipulators*, Third International Symposium on Robotics Research, Paris, October 1985.

[18] J. Pertin. *Modelisation du Raisonnement Géométrique pour la programmation des Robots.* Thèse INPG, July 1986.

[19] O. Khatib *Real-Time Obstacle Avoidance for robot Manipulators and Mobile Robots* The International Journal of robotics Research. Vol5. No. 1 Spring 1986

[20] A.K Bejczy. *Robots as Man-extension Systems in Space,* IJAC 9th Triennial World Congress, Budapest, Hongary, 1984.

Omit

MANIPULATOR CONTROL

OMIT

Manipulator Control: An Overview

H. Seraji
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

The next generation of robot manipulators will exhibit rudimentary intelligence by acquisition and perception of sensory data and operation in partially unknown environments with some degree of autonomy. One of the crucial components for realizing these capabilities is a sophisticated manipulator control system. Consequently, research in advancing the control of manipulators is an essential step towards the development of future robots.

The topic of "Manipulator Control" was chosen as one of the central themes of the NASA Workshop on Space Telerobotics. Six sessions containing 38 papers were devoted to manipulator control. In the course of presentations and subsequent panel discussions, it became evident that the topics under manipulator control may broadly be classified into two categories.

The first category contains topics which are largely resolved by now, such as advanced control methodologies for single-arm robots in free motion. Theoretical research in these areas has been pursued actively during the past decade and has reached a modest level of maturity. Nevertheless, there are very few practical implementations of the advanced control techniques, even at the academic level. It was generally felt that more emphasis should be placed on implementing advanced control methodologies on robot manipulators.

The second category covers those topics which are partially known at the present time. These include force control, coordinated multiple-arm control and control of redundant and flexible arms. Although some partial results have been reported on these topics, considerable research is still much needed. For instance, in the area of force control, when the robot makes contact with the environment, dynamic models which adequately represent this interaction are not yet completely developed. In a similar manner, optimum coordination and task allocation among multiple cooperative arms are not yet resolved. It was generally believed that support of fundamental theoretical research on these topics is currently needed.

In summary, the manipulator control sessions at the Workshop were successful in providing a forum for technical interaction among researchers in robot control. Furthermore, the findings of the Workshop were beneficial to the robotics community and in particular to NASA in enhancing its perception of manipulator control technology and in identifying the directions of future research and development in robot control.

Adaptive Force-Position Control for Teleoperated Manipulators

A.J. Koivo
Purdue University
Lafayette, IN 47907

Abstract

An adaptive controller with self-tuning can be designed for teleoperated robotic manipulators by determining a time-series model for the function of the teleoperator. Specifically, the position and force exerted by the operator are modelled for determining the derived values for the trajectory of the end-effector of the manipulator. Thus, the adaptive controller can be designed by following the steps which have previously been presented for the controller design of the gross motion.

1. Introduction

A teleoperated robotic manipulator refers usually to a system in which an operator equipped with sufficient sensors, effectors and computer intelligence can make the manipulator perform complex tasks either under human supervision or autonomously. The human operator supervises the robotic system which is performing low-level tasks by intermittently monitoring and/or reprogramming the computer. Thus, the teleoperator can increase the level of intelligence of the overall system.

The teleoperator functions in the system as the "master" and the manipulator as the "slave". The teleoperator is mainly interested in the motion of the end-effector, and not so much in the motion of the intervening segments. Although the control of manipulator motion is commonly performed in the joint space, it can also be accomplished directly in the Cartesian coordinate system [5]. If the motion of the master (the teleoperator) is described in the Cartesian world coordinate system, the slave can be made to follow the motion of the master by controlling the motion of the slave (manipulator) in the Cartesian base coordinate system. In the preliminary work described here, we will use this approach to control the force exerted by the end-effector of the manipulator, and its gross motion. We will construct an adaptive self-tuning controller for the control of the force and position of the end-effector.

The overall system is first described briefly, and the problem formulation is given. A time-series model for the motion of the teleoperator is then developed. This model is used to predict the desired motion of the manipulator. An adaptive controller is then designed to make the manipulator follow the desired motion without the supervision of the operator.

2. Teleoperated Robotic Manipulator System and Problem Statement

The overall system consists of a robotic manipulator with an end-effector, computer, and a teleoperator, whose arm and hand are constrained to have the same configuration as the manipulator. The hand is assumed to possess two (jaw-like) fingers. It will be assumed in this preliminary study that the position of the hand as well as the force and moments exerted by the hand can be measured.

The measurements of the position of the hand and the forces (moments) exerted by the hand on the object will be described as a multivariate discrete time-series model. The parameters of this model are estimated recursively by the least squares error method on-line. The resulting model will be used to predict the desired values of the variables for controlling the manipulator, and its end-effector.

The dynamics of the manipulator will also be modelled by means of a multivariate stochastic discrete time-series equation with unknown parameters. This vector difference equation is used as the basis in designing an adaptive self-tuning controller for the manipulator motion.

The problems to be considered in the following consist of constructing (i) a discrete time-series model for the desired values of the position and forces for the end-effector; (ii) a multivariate auto-regressive (ARX) model with external inputs for designing an adaptive self-tuning controller for the dynamics of the manipulator. The difference equation model for the desired values of the position and force are based on the measurements which become available when the teleoperator performs a task (i.e., teach-by-doing). The ARX-model is determined on the basis of the measurements available from the position sensors and the force sensors of the manipulator. We will assume that the forces exerted by the end-effector on the object are "soft", i.e., that they can be modelled by linear springs.

3. Mathematical Model for Teleoperated Manipulator System

In order to make the end-effector follow the path determined by the teleoperator, and exert the force (torque) specified by the same operator, the values of these variables will be measured as a function of time. The future values of these variables can be predicted by using a time-series model constructed on the basis of the measurements. Suppose that the *teleoperator exerts a force* $f^d(k)$ at *time* kT (T=*sampling period*) on an object, while following a trajectory passing through the points $p^d(k)$ expressed relative to a chosen Cartesian world coordinate system. These values can be modelled by time-series models:

$$f^d(k) = A_o^d + \sum_{i=1}^{n} A_i^d\, f^d(k-i) + \xi^d(k) \tag{1}$$

$$p^d(k) = B_o^d + \sum_{i=1}^{n} B_i^d\, p^d(k-i) + \eta^d(k) \tag{2}$$

where the equation error is signified by $\xi^d(k)$, and $\eta^d(k)$; the unknown parameters in the matrices A_j^d, and B_j^d, j = 0,1,...,n are estimated by the least squares error method on the basis of the measurements.

The *one-step ahead predicted values for the force* f^d and p^d are computed by the following equations:

$$f^d(k+1|k) = \hat{A}_o^d + \sum_{i=1}^{n} \hat{A}_i^d\, f^d(k-i+1) \tag{3}$$

$$p^d(k+1|k) = B_o^d + \sum_{i=1}^{n} \hat{B}_i^d\, p^d(k-i+1) \tag{4}$$

where the *terms on the right* are known from the measurements and the calculations of the parameter estimates.

To construct a controller for the manipulator, an ARX-model is used as the basis of the design. If the position of the end-effector relative to the Cartesian base coordinate system p(k) at time kT is measured (e.g., encoder readings), while it exerts a force f(k) on an object, then an ARX-model for the measurements can be written as:

$$p(k) = C_o + \sum_{i=1}^{m} C_i p(k-i) + G_1 u(k-1) + e_1(k) \tag{5}$$

$$f(k) = D_o + \sum_{i=1}^{m} D_i f(k-i) + E_1 u(k-1) + e_2(k) \tag{6}$$

where the modelling errors are denoted by $e_1(k)$ and $e_2(k)$; the unknown parameters in the matrices C_j, and j, j = 0,...,m are estimated recursively on line by the least squares error method on the basis of the available measurements.

The desired values for the position and force are related to the values $p^d(k)$ and $f^d(k)$ determined by equations (1) through (4) for the teleoperator. By applying appropriate coordinate transformation, which relates the Cartesian world coordinate system used in describing $\{p^d(k)\}$ and $\{f^d(k)\}$, to the *Cartesian base coordinate system*, the desired values $\{\bar{p}^d(k)\}$ and $\{\bar{f}^d(k)\}$ expressed in the base coordinate system can be determined.

Having obtained the desired values for the manipulator motion, an adaptive self-tuning controller can next be designed. It is determined by minimizing the following performance criterion:

$$I_k[u] = E\{\,||p(k+1)-\bar{p}^d(k+1|k)||^2_{(I-S)} + ||f(k+1)-\bar{f}^d(k+1|k)||^2_S + ||u(k)||^2_R\,\} \tag{7}$$

where the matrix S represents the selection matrix for determining when the force-servoing or position-servoing will be used. The norm refers to the usual generalized Euclidean norm. The expectation operation is conditioned on the available measurements.

The problem is solved by minimizing $I_k[u]$ in equation (7) subject to the plant equation constraints given by equations (5) and (6).

The minimizing controller $u^*(k)$ is specified by the following equation:

$$\hat{G}_1'(I-S)[\hat{C}_o + \sum_{i=1}^{m}\hat{C}_i p(k+1-i) + \hat{G}_1 u^*(k) - \bar{p}^d(k+1|k)] +$$

$$+ \hat{E}_1' S[\hat{D}_o + \sum_{i=1}^{m}\hat{D}_i f(k-i) + \hat{E}_1 u^*(k) - \bar{f}^d(k+1|k)] + R u^*(k) = 0 \tag{8}$$

Equation (8) can be solved for the control input $u^*(k)$ in the feedback form. It should be observed that the desired trajectory must also be updated according to equations (1) through (4), with the model parameters.

Simulations studies are currently being conducted to demonstrate for the feasibility of the approach.

4. Conclusions

An adaptive self-tuning controller design has been presented for the operation of teleoperated manipulators. The approach is first to model the function of the teleoperator. Then, the controller design is performed on the basis of the desired trajectory determined. Simulation studies are currently being conducted using the approach.

Acknowledgement

This work has been supported in part by National Science Foundation Grant CDR 8500022.

5. References

[1] L. M. Jenkins, "Telerobotic Work System - Space Application," *Proc. 1986 IEEE International Conference on Robotics and Automation,* San Francisco, CA, April 1986, pp. 804-806.

[2] J. E. Pennington, "Space Telerobotics: A Few More Hurdles," ibid, pp. 813-816.

[3] T. B. Sheridan, "Human Supervisory Control of Robot Systems," ibid, pp. 808-812.

[4] H. L. Martin, and W. R. Hamel, "Joining Teleoperation with Robotics for Remote Manipulation in Hostile Environments," *Proc. of Robots 8 Conference,* June 1984.

[5] A. J. Koivo, "Self-tuning Manipulator Control in Cartesian Base Coordinate System," *J. of Dynamical Systems, Measurements, and Control,* December 1985, pp. 316-323.

Adaptive Control of Dual-Arm Robots

H. Seraji
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract: The paper describes three strategies for adaptive control of cooperative dual-arm robots. In the position-position control strategy, the adaptive controllers ensure that the end-effector positions of both arms track desired trajectories in Cartesian space despite unknown time-varying interaction forces exerted through the load. In the position-hybrid control strategy, the adaptive controller of one arm controls end-effector motions in the free directions and applied forces in the constraint directions; while the adaptive controller of the other arm ensures that the end-effector tracks desired position trajectories. In the hybrid-hybrid control strategy, the adaptive controllers ensure that both end-effectors track reference position trajectories while simultaneously applying desired forces on the load. In all three control strategies, the cross-coupling effects between the arms are treated as "disturbances" which are rejected by the adaptive controllers while following desired commands in a common frame of reference. The adaptive controllers do not require the complex mathematical model of the arm dynamics or any knowledge of the arm dynamic parameters or the load parameters such as mass and stiffness. The controllers have simple structures and are computationally fast for on-line implementation with high sampling rates.

1. Introduction

During the past decade, robot manipulators have been utilized in industry for performing simple tasks, and it is foreseen that in the near future anthropomorphic robots will replace human operators in carrying out various complex tasks both in industry and in hazardous environments. Nevertheless, present-day robots can be considered at best as "handicapped" operators due to their single-arm structure. It is evident that multiplicity of robot arms yields greater dexterity and increased efficiency and provides the capability of handling larger loads. Dual-arm robots will therefore have capabilities which may match those of ambidextrous human operators in dexterity and efficiency.

The research on dual-arm robots is at its early stages at the present time, and a few approaches are currently available. Nakano et al. [1] propose a method for control of dual-arm robots in a master/slave manner. Ishida [2] considers parallel and rotational transfer of loads using dual-arm robots. Fujii and Kurono [3] suggest a technique for dual-arm control based on the method of virtual reference. Alford and Belyeu [4] describe a method for coordinated control of two arms. Zheng and Luh [5,6] obtain constrained relations and control laws for two coordinated arms. Tarn et al. [7] employ the exact linearization technique for dual-arm control. Hayati [8] proposes a method for controlling dual-arm robots based on partitioning the load between the arms. Koivo [9] suggests an adaptive control technique for dual-arm robots using the self-tuning approach. Lim and Chyung [10] describe a positional control scheme for two cooperating robot arms.

The control architecture considered in this paper is based on the tri-level hierarchical control of dual-arm robots as shown in Figure 1. In this tri-level control architecture, the high level plans the task to be performed and decomposes the task into appropriate subtasks for the right and left arms. In the intermediate level, each subtask is transformed into a sequence of synchronous desired trajectories of end-effectors motions and applied forces. The low level is concerned with the execution of the desired trajectories and employs feedback from the current status of the arms. In this tri-level hierarchy, the low level "achieves" the desired motion and operates in "millisecond" time-scale, the intermediate level "determines" the motion desired for the subtask in "second" time-scale, and the high level "plans" the subtask sequences in "minute" time-scale.

The present paper describes three control strategies for low-level adaptive control of cooperative dual-arm robots using recent results on adaptive control of single-arm robots [11-13]. In each control strategy, a suitable task-related coordinate frame of reference is chosen for both arms and the desired motions and applied forces for each arm are expressed in this common reference frame. Then each arm will move as though it were carrying out the commanded motion by itself in the reference frame. The adaptive controllers ensure that the controlled variables track the desired reference commands and reject the unwanted disturbances caused by interaction forces and torques exerted through the load.

The paper is structured as follows. In Section 2, the position-position control strategy is discussed. In Section 3, the position-hybrid control strategy is developed. The hybrid-hybrid control strategy is addressed in Section 4. Section 5 discusses the results of the paper and draws some conclusions.

2. Position-Position Control Strategy

In this section, we shall investigate the first control strategy for dual-arm manipulators in which both arms are in pure position control, as shown in Figure 2. In other words, the positions and orientations of both end-effectors are required to track desired trajectories in a common frame of reference. In this situation, uncontrolled forces and torques will be exerted on the common load held by the end-effectors. In this section, we investigate the performance of the position control systems in the face of the interaction forces and torques exerted through the load.

2.1 Position Controller for Right/Left Arm

The dynamic model of each manipulator arm can be represented by a differential equation in Cartesian space as [14]

$$M(X)\ddot{X} + N(X,\dot{X}) + G(X) + H(\dot{X}) \pm f = F \qquad (1)$$

where the above terms are defined as:

$X, \dot{X}, \ddot{X}$ = nx1 vectors of end-effector position, velocity and acceleration in a fixed task-related Cartesian frame of reference

F = nx1 vector of "virtual" Cartesian forces applied to the end-effector as the control input

$M(X)$ = nxn symmetric positive-definite Cartesian mass matrix

$N(X,\dot{X})$ = nx1 Cartesian Coriolis and centrifugal force vector

$G(X)$ = nx1 Cartesian gravity loading vector

$H(\dot{X})$ = nx1 Cartesian friction force vector

f = nx1 vector of forces and torques exerted by the end-effector on the load

The force/torque vector f both imparts motion to and applies force/torque on the load and acts as the coupling element between the two arms. In the following analysis, the force/torque vector f will be considered as a "disturbance input" to the position control system. The function of the control system is to ensure that the end-effector position vector X tracks the nx1 vector of desired trajectory X_d despite the disturbance force f. For each manipulator arm, let us apply the linear adaptive position control law [12]

$$F(t) = d(t) + [K_p(t)\, E(t) + K_v(t)\dot{E}(t)] + [C(t)\, X_d(t) + B(t)\, \dot{X}_d(t) + A(t)\, \ddot{X}_d(t)] \qquad (2)$$

as shown in Figure 3, where $E(t) = X_d(t) - X(t)$ is the nx1 position tracking-error vector. In the control law (2), the nx1 vector $d(t)$ is an auxiliary signal to be synthesized by the adaptation scheme, while $[K_pE + K_v\dot{E}]$ and $[CX_d + B\dot{X}_d + A\ddot{X}_d]$ are the contributions due to the feedback and feedforward controllers respectively. Following the method described in Reference [12], the required auxiliary signal and controller gains are updated according to the following adaptation laws:

$$d(t) = d(0) + \delta_1 \int_0^t r(t)dt + \delta_2\, r(t) \quad (3)$$

$$K_p(t) = K_p(0) + \alpha_1 \int_0^t r(t)\, E'(t)dt + \alpha_2 r(t)\, E'(t) \quad (4)$$

$$K_v(t) = K_v(0) + \beta_1 \int_0^t r(t)\, \dot{E}'(t)dt + \beta_2 r(t)\, \dot{E}'(t) \quad (5)$$

$$C(t) = C(0) + \nu_1 \int_0^t r(t)\, X_d'(t)^{dt} + \nu_2 r(t)\, X_d'(t) \quad (6)$$

$$B(t) = B(0) + \gamma_1 \int_0^t r(t)\, \dot{X}_d'(t)^{dt} + \gamma_2 r(t)\, \dot{X}_d'(t) \quad (7)$$

$$A(t) = A(0) + \lambda_1 \int_0^t r(t)\, \ddot{X}_d'(t)^{dt} + \lambda_2 r(t)\, \ddot{X}_d'(t) \quad (8)$$

where

$$r(t) = W_p\, E(t) + W_v\, \dot{E}(t) \quad (9)$$

is an nx1 vector, $\{\delta_1, \alpha_1, \beta_1, \nu_1, \gamma_1, \lambda_1\}$ *are positive scalars*, $\{\delta_2, \alpha_2, \beta_2, \nu_2, \gamma_2, \lambda_2\}$ are positive or zero scalars, and the prime denotes transposition. In equation (9), W_p and W_v are nxn constant weighting matrices chosen by the *designer to reflect the relative significance of the position and velocity errors* E and $\dot{E}$. It must be noted that since we cannot physically apply the Cartesian control force F to the end-effector, we instead compute the nx1 equivalent joint torque vector T to effectively cause this force. Thus, for __each__ manipulator arm, the control law in joint space is given by

$$T(t) = J'(\theta)\, F(t) = J'(\theta)\, [\, d(t) + K_p(t)E(t) + K_v(t)\dot{E}(t) + C(t)X_d(t) + B(t)\dot{X}_d(t) + A(t)\ddot{X}_d(t)] \quad (10)$$

where θ is the nx1 vector of joint angular positions and $J(\theta)$ is the nxn Jacobian matrix of the manipulator arm.

Because of the simplicity of the adaptation laws (3) - (8), the robot control algorithm can be implemented using high sampling rates (typically 1 KHz). In each sampling period (~ 1msec), the controller gains can change significantly; whereas the terms M, N, G, H, and f in the robot model (1) cannot change noticeably. As a result, in deriving equations (3) - (8), it was assumed that these terms are unknown and "slowly time-varying" relative to the adaptation laws. It is seen that the inclusion of the disturbance force f in the robot model (1) does not affect the controller adaptation laws since the change in f over one sampling period is relatively small.

The above argument suggests that when both manipulator arms are controlled using the two __independent__ adaptive position controllers, we expect the end-effectors to track the desired position trajectories despite the interaction forces and torques exerted through the load. It must be noted that since the *force* on the load is not a controlled variable in this scheme, this strategy can lead to undesirable load forces when the position trajectories are not planned in coordination or are not tracked closely.

3. __Position-Hybrid Control Strategy__

In this section, the position-hybrid control strategy for dual-arm manipulators will be studied in which the left arm is in pure position control and the right arm is in hybrid *position/force* control, as shown in Figure 4. In other words, for the left arm, the end-effector position is required to track a desired trajectory in a frame of reference. For the right arm, in the same *reference frame*, the contact force between the end-effector and the load must be controlled in the directions constrained by the load, while the end-effector position is to be controlled simultaneously in the *free directions*. This control strategy can be applied when one robot arm is confined to operate only in position control mode whereas the other arm can be controlled in hybrid control mode.

3.1 Position Controller for Left Arm

For the left arm, the interaction forces and torques exerted through the load are considered as "disturbances," and the adaptive position control system can ensure tracking of the desired position trajectories despite such disturbances, as outlined in Section 2. The adaptive position control law for the left arm shown in Figure 3 is given by [12]

$$T_{\ell}(t) = J_{\ell}'(\theta_{\ell}) \; [\bar{d}(t) + \bar{K}_p(t)E(t) + \bar{K}_v(t)\dot{E}(t) + \bar{C}(t)X_{\ell d}(t) + \bar{B}(t)\dot{X}_{\ell d}(t) + \bar{A}(t)\ddot{X}_{\ell d}(t)] \qquad (11)$$

where T_{ℓ} is the nx1 joint torque vector, θ_{ℓ} is the nx1 joint angle vector, $J_{\ell}(\theta_{\ell})$ is the nxn Jacobian matrix, $X_{\ell d}(t)$ is the nx1 vector of desired Cartesian position trajectory, $E(t) = X_{\ell d}(t) - X(t)$ is the nx1 position tracking-error vector and the terms in equation (11) are adapted as follows:

$$\bar{d}(t) = \bar{d}(0) + \delta_1 \int_0^t r(t)dt + \delta_2 r(t) \qquad (12)$$

$$\bar{K}_p(t) = \bar{K}_p(0) + \alpha_1 \int_0^t r(t)E'(t)dt + \alpha_2 r(t)E'(t) \qquad (13)$$

$$\bar{K}_v(t) = \bar{K}_v(0) + \beta_1 \int_0^t r(t)\dot{E}'(t)dt + \beta_2 r(t)\dot{E}'(t) \qquad (14)$$

$$\bar{C}(t) = \bar{C}(0) + \nu_1 \int_0^t r(t) \; X_{\ell d}'(t)dt + \nu_2 r(t) \; X_{\ell d}'(t) \qquad (15)$$

$$\bar{B}(t) = \bar{B}(0) + \gamma_1 \int_0^t r(t) \; \dot{X}_{\ell d}'(t)dt + \gamma_2 r(t) \; \dot{X}_{\ell d}'(t) \qquad (16)$$

$$\bar{A}(t) = \bar{A}(0) + \lambda_1 \int_0^t r(t) \; \ddot{X}_{\ell d}'(t)dt + \lambda_2 r(t) \; \ddot{X}_{\ell d}'(t) \qquad (17)$$

where

$$r(t) = W_p E(t) + W_v \; \dot{E}(t) \qquad (18)$$

and the symbols are defined in Section 2.

3.2 Hybrid Controller for Right Arm

We shall now discuss the hybrid position/force controller for the right arm. Consider a task-related "constraint frame" (coordinate system) which is defined by the particular contact situation occurring between the right end-effector and the load. In this frame, the n degrees-of-freedom (or directions) in the Cartesian space {X} can be partitioned into two orthogonal sets; the m constraint directions in subspace {Z} and the ℓ free directions in subspace {Y}, with $n = m + \ell$. In the m constraint directions, the end-effector makes contact with the load and the *contact force* needs to be controlled. In the ℓ free directions, the end-effector is free to move and the *end-effector position* is to be controlled. In the hybrid control architecture [16,17], two separate controllers are employed for simultaneous force and position control. The "force controller" achieves tracking of desired force setpoints in the constraint directions; while the "position controller" accomplishes tracking of desired position trajectories in the free directions.

The dynamic model of the right arm in the constraint directions can be written as [13]

$$A(X,\dot{X}) \; \ddot{P}(t) + B(X,\dot{X}) \; \dot{P}(t) + P(t) + C_p(Y) \pm f_z = F_z(t) \qquad (19)$$

where F_z is the mx1 "virtual" Cartesian force vector applied to the end-effector in the constraint directions, Z is the mx1 vector of end-effector position, the mxm matrices A and B are highly complex nonlinear functions of the end-effector position X, C_p is the cross-coupling from the position loop in the force loop and f_z is the component of the force exerted on the end-effector by the load in the constraint directions. The term f_z represents the cross-coupling that exists between the arms through the load and is considered as a "disturbance" to the hybrid controller.

In a recent paper [13], an adaptive force control scheme is developed within the hybrid control architecture. For the right arm, the linear adaptive force control law in the constraint directions is given by [13]

$$F_z(t) = P_r(t) + d(t) + K_p(t)\, E(t) + K_I(t) \int_0^t E(t)dt - K_v(t)\dot{Z}(t) \tag{20}$$

as shown in Figure 5, where $P_r(t)$ is the desired contact force on the load used as a feedforward term, $d(t)$ ic an auxiliary signal, $E(t) = P_r(t) - P(t)$ is the deviation of the actual force $P(t)$ from the desired value, and $\{K_p(t),\ K_I(t),\ K_v(t)\}$ are adaptive gains of the PID controller. The terms in the force control law (20) are adapted as follows:

$$d(t) = d(0) + \delta_1 \int_0^t q(t)dt + \delta_2 q(t) \tag{21}$$

$$K_I(t) = K_I(0) + \alpha_1 \int_0^t q(t)\, E^{*\prime}(t)dt + \alpha_2 q(t)\, E^{*\prime}(t) \tag{22}$$

$$K_p(t; = K_p(0) + \beta_1 \int_0^t q(t)\, E'(t)dt + \beta_2 q(t)\, E'(t) \tag{23}$$

$$K_v(t) = K_v(0) - \gamma_1 \int_0^t q(t)\, \dot{Z}'(t)dt - \gamma_2 q(t)\, \dot{Z}'(t) \tag{24}$$

where

$$q(t) = W_I\, E^*(t) + W_p E(t) - W_v \dot{Z}(t) \tag{25}$$

In equations (21)-(25), $E^*(t) = \int_0^t E(t)dt$ is the integral error vector, $\{\delta_1,\ \alpha_1,\ \beta_1,\ \gamma_1\}$ are positive scalars, $\{\delta_2,\ \alpha_2,\ \beta_2,\ \gamma_2\}$ are positive or zero scalars, and $\{W_I,\ W_p,\ W_v\}$ are constant weighting matrices chosen by the designer to reflect the relative significance of E^*, E and $\dot{Z}$.

The dynamic model of the right arm in the free directions can be written as [13]

$$A_o(X,\dot{X})\, \ddot{Y}(t) + B_o(X,\dot{X})\, \dot{Y}(t) + C_o(X,\dot{X})Y(t) + C_f(P) \pm f_y = F_y(t) \tag{26}$$

where f_y is the component of the end-effector force in the free directions, C_f is the cross-coupling from the force loop, A_o, B_o, C_o are complex nonlinear matrices, Y is the end-effector position vector and F_y is the "virtual" end-effector control force. For the right arm, the linear adaptive position control law in the free directions is given by

$$F_y(t) = \tilde{d}(t) + \tilde{K}_p(t)\, E_p(t) + \tilde{K}_v(t)\, \dot{E}_p(t) + \tilde{C}(t)\, R(r) + \tilde{B}(t)\, \dot{R}(t) + \tilde{A}(t)\, \ddot{R}(t) \tag{27}$$

as in Section 2, where R is the desired position trajectory, $E_p = R - Y$ is the position tracking-error, and F_y is the "virtual" Cartesian force in the free directions. Thus, in order to implement the force and position controllers (20) and (27) in the hybrid control architecture, the joint space control law for the right arm is given by

$$T_r(t) = J_r'(\theta_r) \begin{pmatrix} F_z(t) \\ F_y(t) \end{pmatrix} \tag{28}$$

as shown in Figure 6, where θ_r is the joint angle vector, T_r is the joint torque vector, and J_r is the Jacobian matrix of the right arm with appropriate reordering of columns of J_r if necessary.

The hybrid controller adaptation laws (3)-(8) and (21)-(24) are extremely simple, and therefore the control algorithm can be implemented using high sampling rates ($\simeq$ 1 KHz); yielding improved performance particularly in force control applications. Since in each sampling period ($\simeq$ 1 msec) the terms in the robot models (19) and (26) cannot change noticeably, it is reasonable to assume that these terms are "slowly time-varying" compared to the adaptation scheme. Thus the inclusion of the disturbance forces f_z and f_y in the robot models (19) and (26) does not affect the controller performance.

We conclude that using the position-hybrid control strategy, the left end-effector will track the desired position trajectory despite the interaction forces through the load. The right end-effector will exert the desired force on the load in certain directions while simultaneously tracking the desired position trajectory in the orthogonal directions. It must be noted that in this control strategy, slight fluctuations may be observed on the load force due to very small vibrations of the left arm under position control.

4. Hybrid-Hybrid Control Strategy

In this section, the hybrid-hybrid control strategy for dual-arm manipulators will be studied in which both arms are in hybrid position/force control, as shown in Figure 7. In other words, in a common frame of reference, for both arms, the forces exerted by the end-effectors on the load in the constraint directions {Z} must be controlled, while simultaneously the end-effectors are required to track desired position trajectories in the free directions {Y}. Any unwanted forces and torques on the load generated by the relative position and orientation of the end-effectors will act as "disturbances," and the adaptive hybrid controllers ensure that the desired position/force trajectories are tracked despite such disturbances.

4.1 Hybrid Controller for Right/Left Arm

Following Section 3, for each manipulator arm the hybrid position/force control law in the joint space can be written as

$$T(t) = J'(\theta) \begin{pmatrix} F_z(t) \\ F_y(t) \end{pmatrix} \qquad (29)$$

where $J(\theta)$ is the Jacobian matrix (with appropriate column reordering if necessary), and $F_z(t)$ and $F_y(t)$ are the "virtual" Cartesian forces applied to the end-effector in the constraint directions {Z} and free directions {Y}, respectively. As shown in Figure 5, the force control law is given by

$$F_z(t) = P_r(t) + d(t) + K_I(t) \int_0^t E_z(t)dt + K_p(t)E_z(t) - K_v(t)Z(t) \qquad (30)$$

where $P_r(t)$ is the desired force setpoint, $E_z(t) = P_r(t) - P_z(t)$ is the *force* tracking-error and the adaptation laws are:

$$d(t) = d(0) + \delta_1 \int_0^t q(t)dt + \delta_2 q(t)$$

$$K_I(t) = K_I(0) + \alpha_1 \int_0^t q(t)\ E_z^{*\prime}(t)dt + \alpha_2 q(t)\ E_z^{*\prime}(t)$$

$$K_p(t) = K_p(0) + \beta_1 \int_0^t q(t)\ E_z'(t)dt + \beta_2 q(t)\ E_z'(t)$$

$$K_v(t) = K_v(0) - \gamma_1 \int_0^t q(t)\ \dot{Z}'(t)dt - \gamma_2 q(t)\ \dot{Z}'(t)$$

where

$$q(t) = W_I E_z^*(t) + W_p E_z(t) - W_v \dot{Z}(t)$$

and $E_z^*(t) = \int_0^t E_z(t)dt$ and $\{W_I, W_p, W_v\}$ are desired weighting matrices.

The position control law is expressed as

$$F_y(t) = \bar{d}(t) + \bar{K}_p(t)E_y(t) + \bar{K}_v(t)\dot{E}_y(t) + \bar{C}(t)R(t) + \bar{B}(t)\dot{R}(t) + \bar{A}(t)\ddot{R}(t) \qquad (31)$$

where $R(t)$ is the desired position trajectory, $E_y(t) = R(t) - Y(t)$ is the *position* tracking-error and the adaptation laws are:

$$\bar{d}(t) = \bar{d}(0) + \bar{\delta}_1 \int_0^t r(t)dt + \bar{\delta}_2 r(t)$$

$$\bar{K}_p(t) = \bar{K}_p(0) + \bar{\nu}_1 \int_0^t r(t)\, E_y'(t)dt + \bar{\nu}_2 r(t) E_y'(t)$$

$$\bar{K}_v(t) = \bar{K}_v(0) + \bar{\eta}_1 \int_0^t r(t)\, \dot{E}_y'(t)dt + \bar{\eta}_2 r(t) \dot{E}_y'(t)$$

$$\bar{C}(t) = \bar{C}(0) + \bar{\mu}_1 \int_0^t r(t)\, E'(t)dt + \bar{\mu}_2 r(t) E'(t)$$

$$\bar{B}(t) = \bar{B}(0) + \bar{\gamma}_1 \int_0^t r(t)\, \dot{E}'(t)dt + \bar{\gamma}_2 r(t) \dot{E}'(t)$$

$$\bar{A}(t) = \bar{A}(0) + \bar{\lambda}_1 \int_0^t r(t)\, \ddot{E}'(t)dt + \bar{\lambda}_2 r(t) \ddot{E}'(t)$$

where

$$r(t) = \bar{W}_p E_y(t) + \bar{W}_v \dot{E}_y(t)$$

and $(\bar{W}_p, \bar{W}_v)$ are desired weighting matrices.

The above controller adaptation laws are extremely simple, and therefore, the hybrid control algorithm can be implemented using high sampling rates (≈ 1 KHz), yielding improved performance. Under the adaptive hybrid controllers, both end-effectors are expected to exert the desired forces on the load while simultaneously moving on desired trajectories. The hybrid-hybrid control strategy is most suitable when simultaneous control of both position and force is required.

5. Discussion and Conclusions

Three adaptive control strategies for cooperative dual-arm robots are described in this paper. In these strategies, each robot arm is considered a subsystem of the total system and is controlled independently using an adaptive controller in the low level of the control hierarchy. Each controller ensures that the controlled variables follow desired commands and reject unwanted cross-coupling effects from other subsystems which are treated as "disturbances." The subsystems are coordinated through trajectory generators in the intermediate level, where synchronous desired trajectories for both arms are specified in a common task-related frame of reference. An important feature of the present approach is that the overall control system for N cooperative arms is reduced to N decentralized independent single-arm controllers. The control schemes do not require communication and data exchange among individual controllers, which is an appealing feature from both computational and reliability points of view. Furthermore, available techniques for single-arm control can be utilized directly in multiple-arm environments.

The control strategies described in this paper do not require the knowledge of the load parameters such as mass and stiffness or the robot dynamic parameters such as link masses and inertias, and can therefore cope with uncertainties or variations in the system parameters. Furthermore, the complex dynamic model of the arms is not used in generating the control actions. The control schemes are very simple and extremely fast for on-line implementation with high sampling rates, yielding improved dynamic performance. The control methodology described in this paper can also be utilized in the coordinated control of N-arm robots when N exceeds two.

6. Acknowledgment

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

7. References

[1] E. Nakano, S. Ozaki, T. Ishida and I. Kato: "Cooperational control of the anthropomorphous manipulator MELARM." Proc. 4th Intern. Conf. on Industrial Robots, pp. 251-260, 1974.

[2] T. Ishida: "Force control in coordination of two arms," Proc. 5th Intern. Conf. on Artificial Intelligence, pp. 717-722, 1977.

[3] S. Fujii and S. Kurono: "Coordinated computer control of a pair of manipulators," Proc. 4th World Congress on Theory of Machines and Mechanisms, pp. 411-417, Newcastle-upon-Tyne (England), 1975.

[4] C.O. Alford and S.M. Belyeu: "Coordinated control of two robot arms," Proc. Intern. Conf. on Robotics, pp. 468-473, Atlanta, GA, 1984.

[5] Y.F. Zheng and J.Y.S. Luh: "Constrained relations between two coordinated industrial robots," Proc. Machine Intelligence Conf., Rochester, NY, 1985.

[6] J.Y.S. Luh and Y.F. Zheng: "Computation of input generalized forces for robots with closed kinematic chain mechanisms," IEEE Journal of Robotics and Automation, pp. 95-103, Vol. RA-1, No. 2, 1985.

[7] T.J. Tarn, A.K. Bejczy and X. Yun: "Coordinated control of two robot arms," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 1193-1202, San Francisco, CA, 1986.

[8] S. Hayati: "Hybrid position/force control of multi-arm cooperating robots," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 82-89, San Francisco, CA, 1986.

[9] A.J. Koivo: "Adaptive position-velocity-force control of two manipulators," Proc. 24th IEEE Conf. on Decision and Control, pp. 1529-1532, Ft. Lauderdale, FL, 1985.

[10] J. Lim and D.H. Chyung: "On a control scheme for two cooperating robot arms," Proc. 24th IEEE Conf. on Decision and Control, pp. 334-337, Ft. Lauderdale, FL, 1985.

[11] H. Seraji: "Adaptive control of robotic manipulators," (JPL Internal Document, Engineering Memorandum 347-182), Jet Propulsion Laboratory, Pasadena, California, January 1986.

[12] H. Seraji: "Direct adaptive control of manipulators in Cartesian space," Journal of Robotic Systems, February 1987 (to appear).

[13] H. Seraji: "Adaptive force and position control of manipulators," (JPL Internal Document, Engineering Memorandum 347-192), Jet Propulsion Laboratory, Pasadena, California, October 1986.

[14] O. Khatib: "Dynamic control of manipulators in Cartesian space," Proc. 6th IFToMM Congress on Theory of Machines and Mechanisms, pp. 1128-1131, New Delhi, India, 1983.

[15] H. Seraji, M. Jamshidi, Y.T. Kim and M. Shahinpoor: "Linear multivariable control of two-link robots," Journal of Robotic Systems, pp. 349-365. Vol. 3, No. 4, 1986.

[16] M.H. Raibert and J.J. Craig: "Hybrid position/force control of manipulators," ASME J. Dyn. Systems, Measurement and Control, Vol. 102, pp. 126-133, 1981.

[17] M.T. Mason: "Compliance and force control for computer controlled manipulators," IEEE Trans. Systems, Man and Cybernetics, SMC11(6), pp. 418-432, 1981.

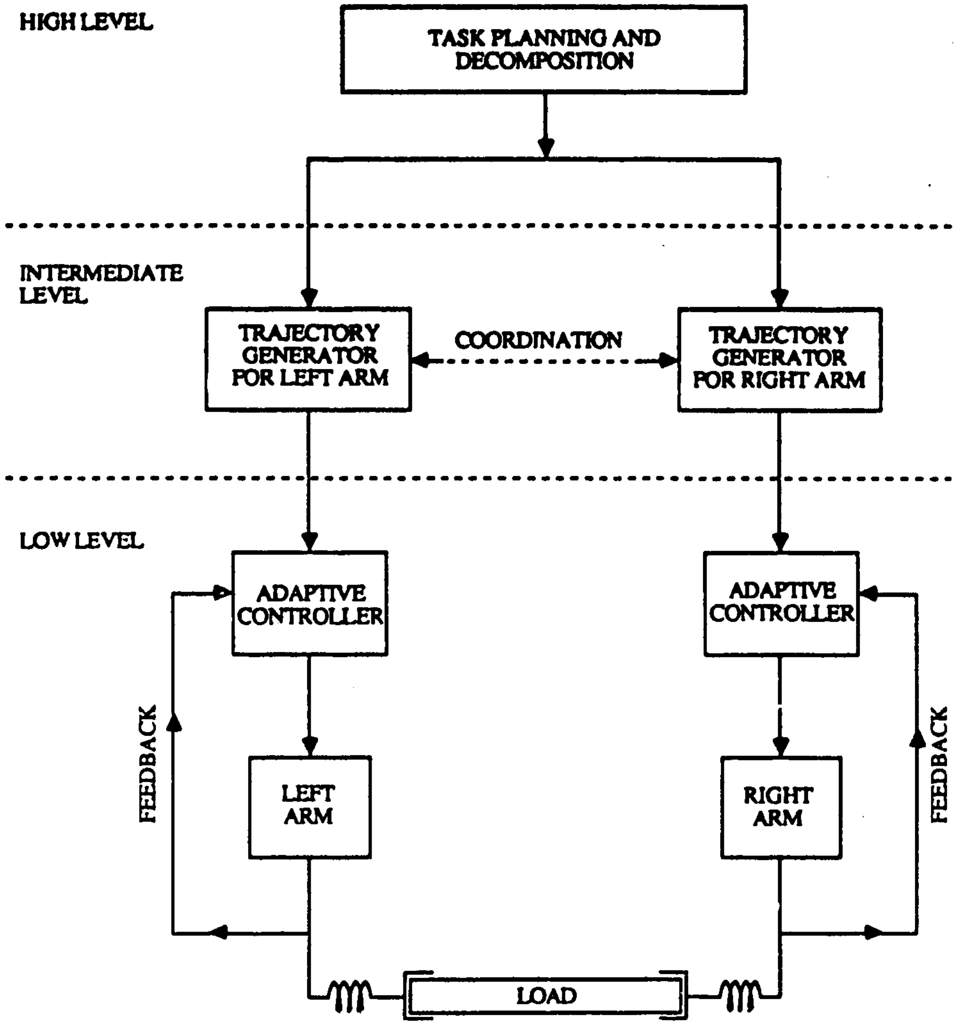


Figure 1. Tri-level Hierarchical Control of Dual-Arm Robot

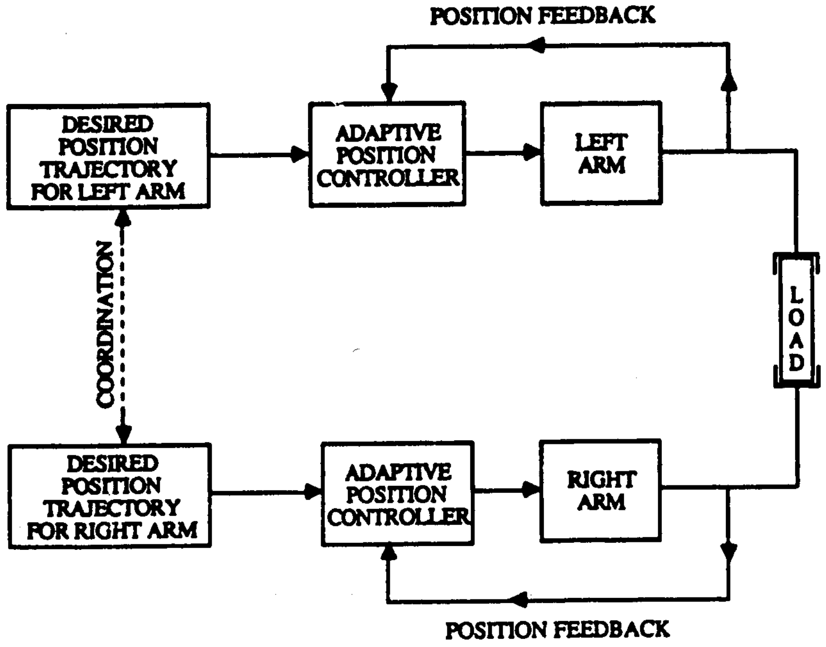

POSITION FEEDBACK
DESIRED POSITION TRAJECTORY FOR LEFT ARM
ADAPTIVE POSITION CONTROLLER
LEFT ARM
COORDINATION
LOAD
DESIRED POSITION TRAJECTORY FOR RIGHT ARM
ADAPTIVE POSITION CONTROLLER
RIGHT ARM
POSITION FEEDBACK

Figure 2. Position-Position Control Strategy

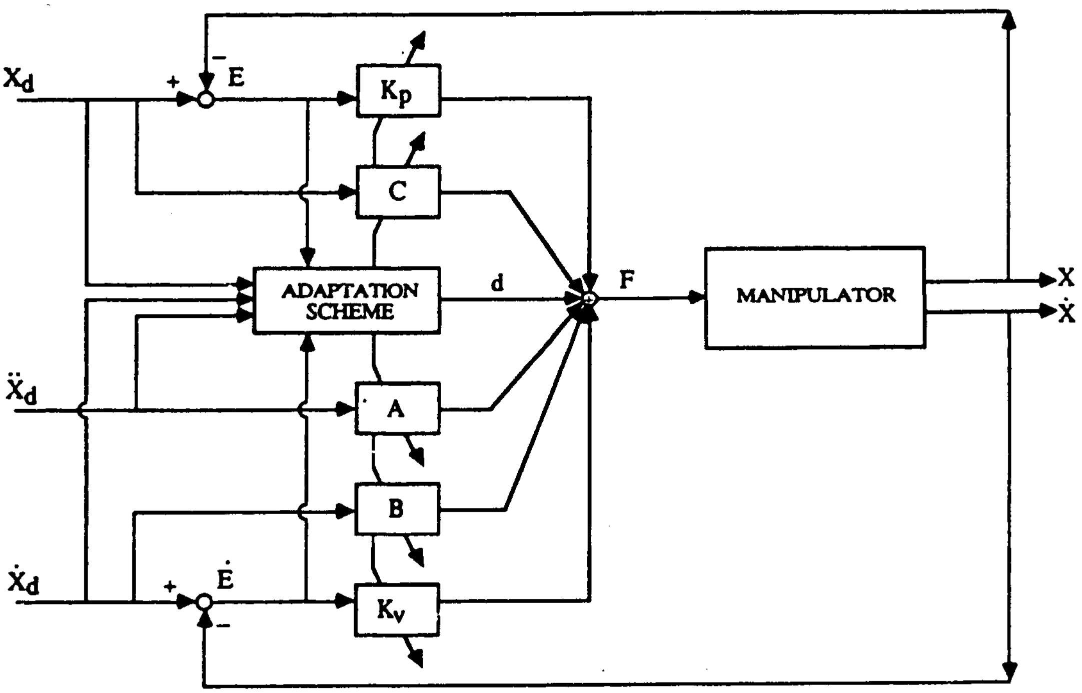


Figure 3. Adaptive Position Control System

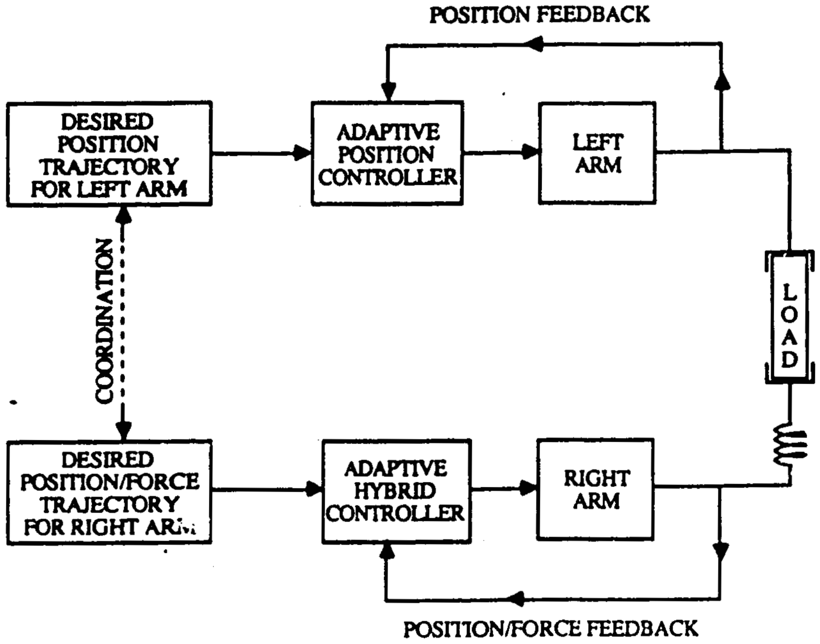

POSITION FEEDBACK
DESIRED POSITION TRAJECTORY FOR LEFT ARM
ADAPTIVE POSITION CONTROLLER
LEFT ARM
COORDINATION
LOAD
DESIRED POSITION/FORCE TRAJECTORY FOR RIGHT ARM
ADAPTIVE HYBRID CONTROLLER
RIGHT ARM
POSITION/FORCE FEEDBACK

Figure 4. Position-Hybrid Control Strategy

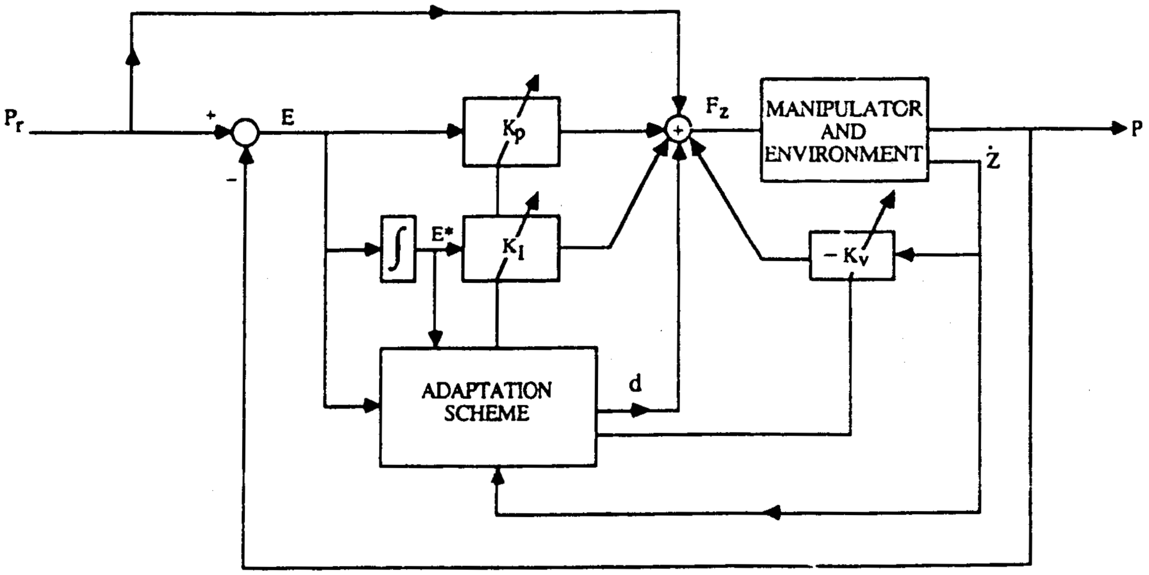


Figure 5. Adaptive Force Control Syst⁻m

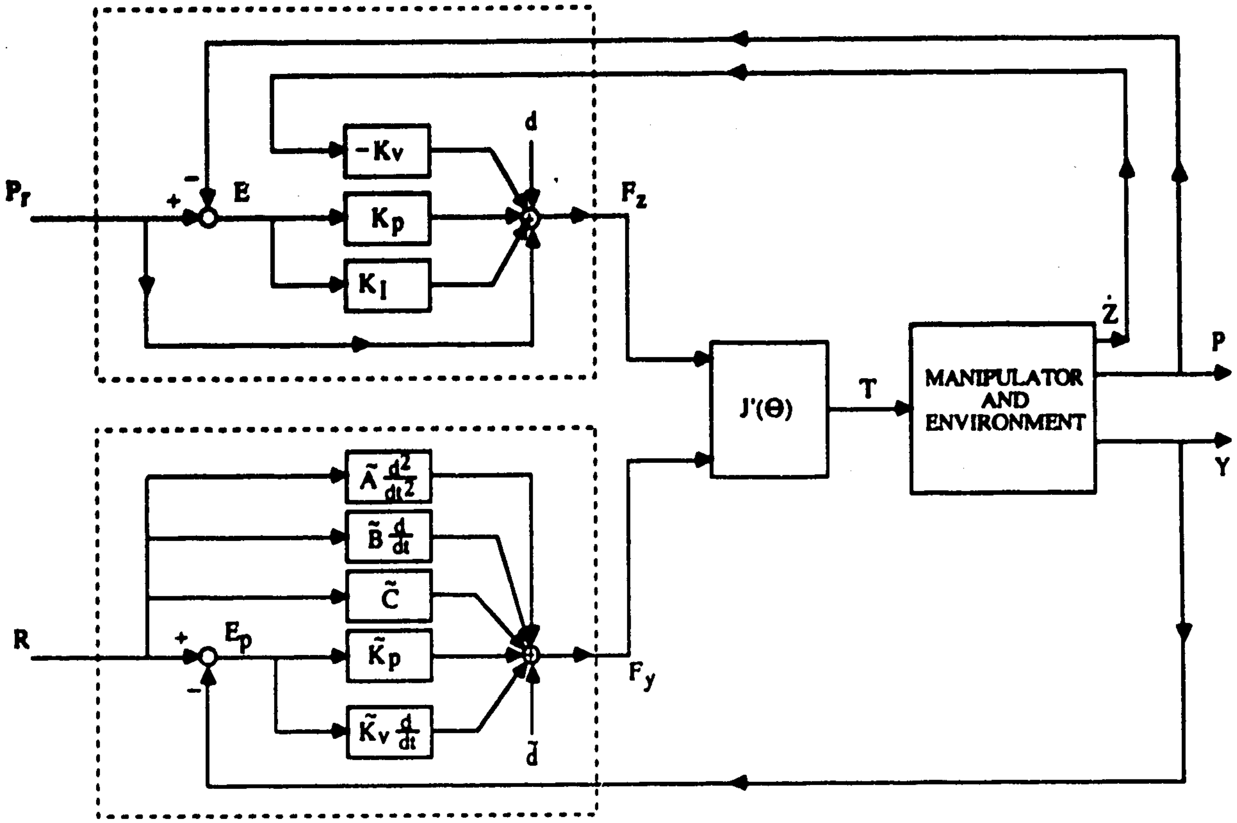


Figure 6. Hybrid Position/Force Control System

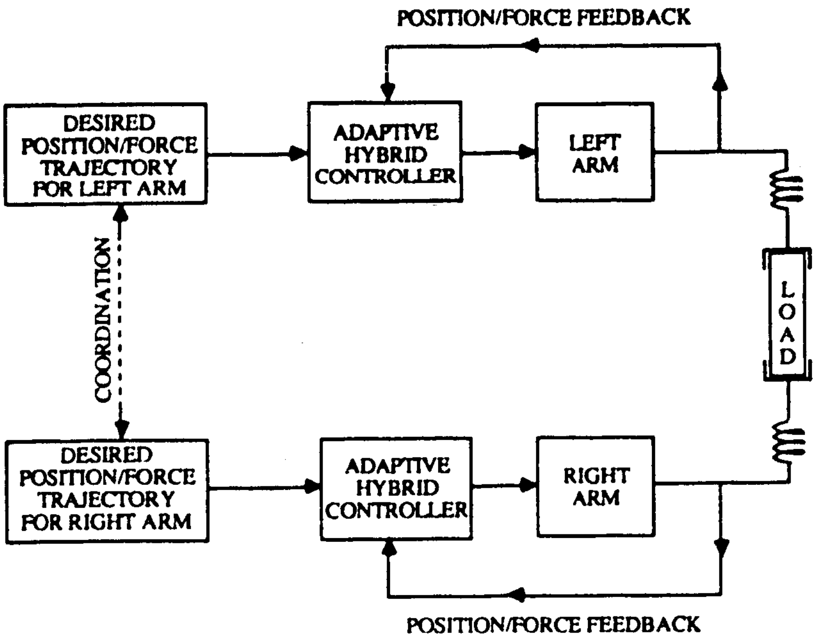

POSITION/FORCE FEEDBACK
DESIRED POSITION/FORCE TRAJECTORY FOR LEFT ARM
ADAPTIVE HYBRID CONTROLLER
LEFT ARM
COORDINATION
LOAD
DESIRED POSITION/FORCE TRAJECTORY FOR RIGHT ARM
ADAPTIVE HYBRID CONTROLLER
RIGHT ARM
POSITION/FORCE FEEDBACK

Figure 7. Hybrid-Hybrid Control Strategy

517-37
88
188126

Design of a Reconfigurable Modular Manipulator System

D. Schmitz and T. Kanade
Carnegie-Mellon University
Pittsburgh, PA 15213

CH 188[illegible]

Abstract

Using manipulators with a fixed configuration for specific tasks is appropriate when the task requirements are known beforehand. However, in less predictable situations, such as an outdoor construction site or aboard a space station, a manipulator system requires a wide range of capabilities, probably beyond the limitations of a single, fixed-configuration manipulator. To fulfill this need, ~~we~~ Researchers have been working on a Reconfigurable Modular Manipulator System (RMMS).

Unlike conventional manipulators with fixed configurations, the RMMS will utilize a stock of interchangeable link and joint modules. Given requirements such as the workspace, dynamic accuracy, and the payload required to accomplish a task, the RMMS will design the most appropriate manipulator configuration, select suitable modules form the inventory, generate an assembly procedure, configure the controller, and finally apply the resultant manipulator to the task. In this way, the RMMS will fill a far wider requirement space than any single manipulator. It is also inherently easy to maintain and transport, since it can be readily assembled and disassembled.

Researchers
~~We~~ have designed and are constructing a prototype RMMS. The prototype currently consists of two joint modules and four link modules. The joints utilize a conventional harmonic drive and torque motor actuator, with a small servo amplifier included in the assembly. A brushless resolver is used to sense the joint position and velocity. For coupling the modules together, we use a standard electrical connector and V-band clamps for mechanical connection, although more sophisticated designs are under way for future versions. The joint design yields an output torque fo 50 ft-lbf at joint speeds up to 1 radian/second. The resolver and associated electronics have resolutions of 0.0001 radians, and absolute accuracies of ±0.001 radians. Manipulators configured from these prototype modules will have maximum reaches in the 0.5 to 2 meter range.

The real-time RMMS controller consists of a Motorola 68020 single-board computer which will perform real time servo control and path planning of the manipulator. This single board computer communicates via shared memory with a SUN3 workstation, which serves as a software development system and robot programming environment. ~~We~~ Researchers have designed a bus communication network to provide multiplexed communication between the joint modules and the computer controller. The bus supports identification of modules, sensing of joint states, and commands to the joint actuator. This network has sufficient bandwidth to allow servo sampling rates in excess of 500 Hz.

1. Introduction

Many applications of robotics in space will be different from the typical applications found in industry. Unlike conventional industrial manipulators which work in a precisely known and controlled environment, a space robot is envisioned as a backup astronaut, performing construction, maintenance, and experimentation. Such a robot must be capable of a wide range of tasks, from small scale, high precision operations such as replacing electronic components in faulty equipment, to immense scale, such as assembling room sized structures into a space station. Many of these tasks will be poorly defined or completely unexpected, particularly maintenance operations.

It is inconceivable to develop a single manipulator which meets even those requirements we can predict as necessary for a space manipulator. The level of dexterity and versatility provided by current manipulator technology is sufficient for only very constrained or well known operations. Nor is it practical, given the expense of transporting material to space, to maintain a large number of different manipulators at potential task sites. Our solution is the development of a manipulator system, which can be readily adapted to meet the individual constraints of each particular application as it occurs.

Designing a manipulator for a single, specific task is conceptually the same process as that employed now for conventional robot applications. However, at present the process of designing and manufacturing a robot system for a particular application is generally too long to be practical for the highly variable and urgent tasks that arise during space missions. To achieve the desired range of robot capabilities, streamlining and automation of the system configuration process are required.

In order to explore tnis approach, we are currently designing a Reconfigurable Modular Manipulator System, or RMMS. The RMMS is a collection of manipulator components or modules (links, joints, actuators, and end effectors), with a wide range of performance (length, strength, torque, speed, resolution, *etc*) utilizing common electrical and mechanical interfaces. This design allows a large number of different manipulators to be assembled, at the task site, from a small inventory of components. In parallel with the development of reconfigurable hardware, a similar software effort is underway to automate the generation of servo-controllor and path planning algorithms, provide a simple means of programming the manipulator task, and actually synthesize a workable manipulator configuration for a given task. Such a manipulator can thus be custom tailored to perform a specific task, and then broken down and re-used in a different configuration when a new task arises.

In a sense, the RMMS concept is an extension of conventional interchangeable manipulator end tools. To date, however, the major efforts in this area have been in the development modular hardware, so that a robot manufacturer can produce various manipulator configurations from standard sub-assemblies [1], or to allow remote maintenance of manipulators in hazardous (eg. radioactive) environments [2]. In contrast, the aim of our RMMS effort is to develop a manipulator system which is modular and reconfigurable by the user at the task site.

2. Design Philosophy and Implementation

An RMMS consists of the same major subsystems as those found in conventional manipulators:

- A pnysical structure made up of joints and links.
- Servo systems for each joint, consisting of actuators, transmissions, and sensors.
- A computer controller and programming environment.

The major difference between an RMMS and a conventional manipulator are the standardized component interfaces. This includes the mechanical mating of manipulator modules, the format of data communication, the communication protocols between hardware and software, and between various levels of software. Although adopting such standards impose inherent restrictions on the design of the actual components, this disadvantage is far offset by the interchangeability of manipulator components and the capability for rapid reconfiguration. In the following subsections, we discuss the conceptual design of each major component and interface in the RMMS we are developing, and the actual implementation in the prototype system.

2.1. Link and Joint Modules

The mechanical modules making up an RMMS are divided into two groups, joints and links. Links are simply structural elements, and joints are servo mechanisms, made up of sensors and actuators. When we represent the kinematics of a manipulator by a series of transformation matrices representing its links and joints, links have fixed transformation matrices, while joints have variable ones (a function of the joint variable). Electrical power is bussed and communication is multiplexed over a small number of conductors permanently installed in each module, allowing for simple assembly without custom cabling.

One implication of this modular joint design is that the *entire* joint actuator must be packaged *within* the joint module. Each joint module must include a motor (or some type of actuator), a transmission mechanism, a position sensor, and the necessary power electronics to control the motor. Although these design constraints limit the power which can be generated by the joint due to the limited size of the motor, transmission, and power amplifier, this is not viewed as a major short comming of the design, particularly for space applications. By properly selecting the transmission reduction ratio, high torques at low speeds can be obtained, which is appropriate for manipulating massive objects in near zero gravity (and also on earth) as long as speed of operation is not critical.

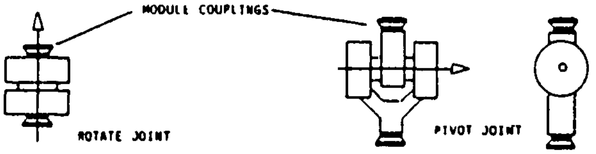


Figure 2-1: Modular Joint Assemblies

For simplicity and convenience, we are considering only the two common types of revolute joint in our RMMS. These two types are rotate, and pivot, and are distinguished by the orientation of the joints link axes with the joint axis. Both types of joint are shown schematically in Figure 2-1. A rotate type joint has link axes which are co-linear with each other and with the joint axis. A pivot has link axes which are both perpendicular to the joint axis.

Our current design for a pivot joint is shown in the photograph in Figure 2-2, and in the section drawing in Figure 2. The joint actuator is a conventional servo motor and linear amplifier driving a harmonic drive with 100:1 reduction ratio. This design yields a maximum output torque of 60 ft·lbf, and maximum axis speed of 1 radian/second. Also integral with the joint assembly is a brushless resolver mounted coaxially with the output shaft, providing position feedback with an accuracy of ±0.001 radian. If lower position resolution is acceptable, lower resolution (and less expensive) sensing electronics can be installed in the joint module. A wire windup allows the resolver (and output shaft) to turn up to 480° before damaging the resolver electrical connections. All of the actuator components are packaged in a sub-assembly of the joint module, allowing a number of different types of module to be based on common parts. The total weight of the joint is 25 lbs. A more compact and lighter version of this joint, as well as several kinematic variations on this basic design are currently under development.

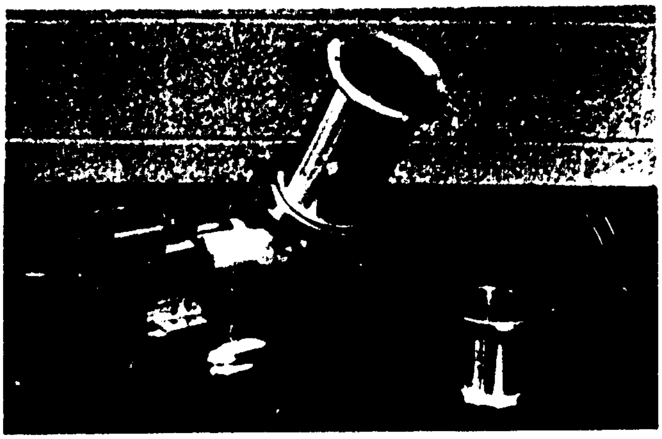


Figure 2-2: CMU RMMS Prototype Pivot Joint

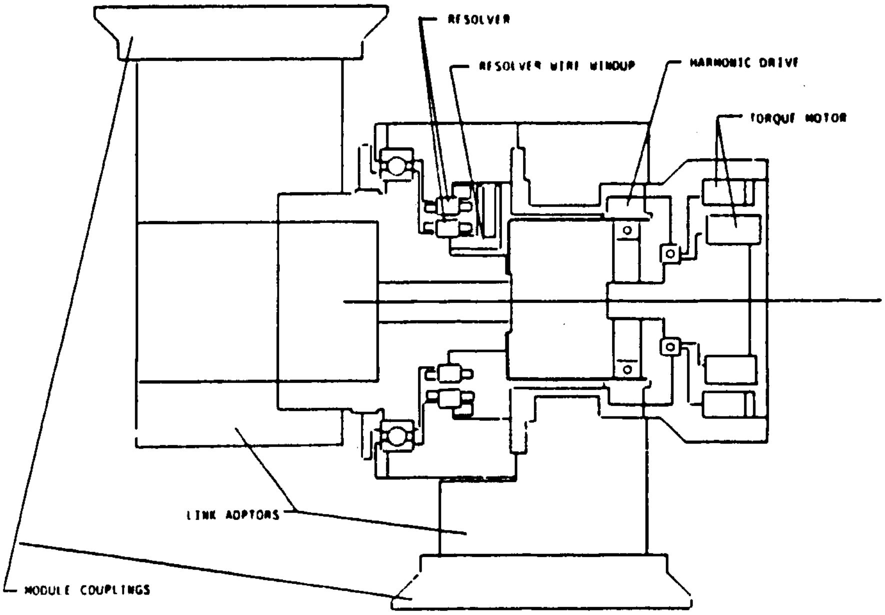


Figure 2-3: Section View of an RMMS Joint

2.2. Joint - Link Interface

In order to assemble the joint and link modules into a manipulator, a method of mechanically coupling the modules is required. This coupling must both align the modules, and lock them together with sufficient strength to transmit the internal forces generated by the movement of the manipulator. In addition to structurally coupling the modules together, this interface must also electrically couple the modules, and be able to sense the coupling orientation of successive modules.

The current interface design is shown in the photograph in Figure 2-4. An arrangement of pins and holes limit the coupling orientation to four, equally spaced positions. An LED in one flange and four phototransistors in the other allow the controller to sense which of the four possible orientations is in use. A commercial V-band clamp couples the two flanges together. As shown in the figure, the same coupling flange is an integral part of the link modules. Although rudimentary, this design provides the necessary functionality for the module interface. However it is not easily operated. Future versions will make use of either quick release V-band clamps, or a more sophisticated design with an automated locking mechanism to allow automatic "peg-in-hole" type coupling.

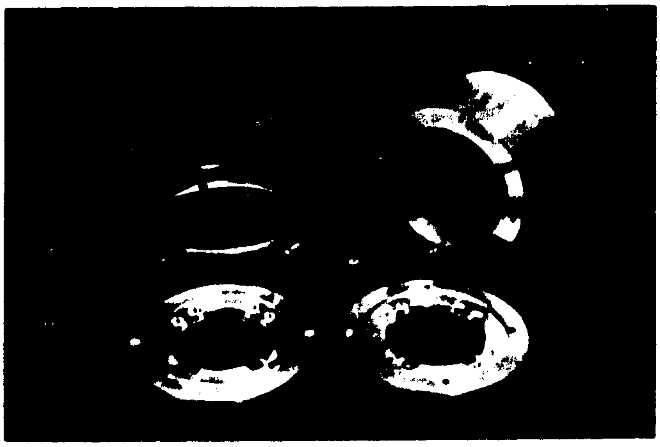


Figure 2-4: Prototype Module Interface

2.3. Communication Interface

As mentioned, each joint will contain the power and sensor electronics for the actuator. In order to control the joint actuators and obtain sensor feedback, a communication link between the joint modules and a computer controller is required. In order to allow standard connectoring between joint modules, this communication link must be implemented using a fixed number of conductors, yet be capable of supporting an arbitrary number of modules. This implies a multiplexed communication link, similar to a computer bus or LAN.

Due to the high overhead associated with existing LANs, our prototype utilizes a bus type implementation. The design is shown schematically in Figure 2-5. The bus design is based on a conventional 8 bit bi-directional data bus, an additional 5 control lines, and a rather unconventional 4 bit daisy chained address bus. The daisy chained address bus provides automatic node address configuration, that is, the first module in the manipulator is node address 1, the second module is node address 2, and so on. This is accomplished by including a "subtract one" circuit in each module which is in the path of the node address lines. Each joint can thus detect "address equals zero" as the node address. Due to the low data rate of the bus (current bus clock is 500 KHz), the propagation delay added by the subtract circuit is negligible.

2.4. Software Controller

In order to augment the capabilities of the RMMS hardware, a reconfigurable control and path planning algorithm are required. These algorithms would allow the control computer to automatically synthesize a control program for a given manipulator configuration and task requirement. This entails automatically deriving the manipulator forward and inverse kinematics, and inverse dynamics given the kinematic and dynamic parameters of the modules. This is one of the major research area of our RMMS project.

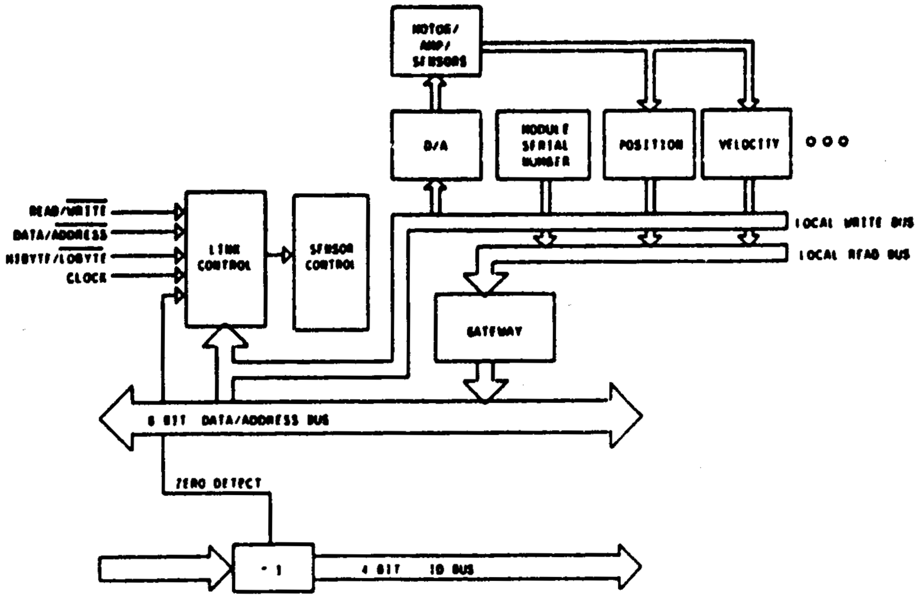


Figure 2-5: Manipulator Communication Bus Logic

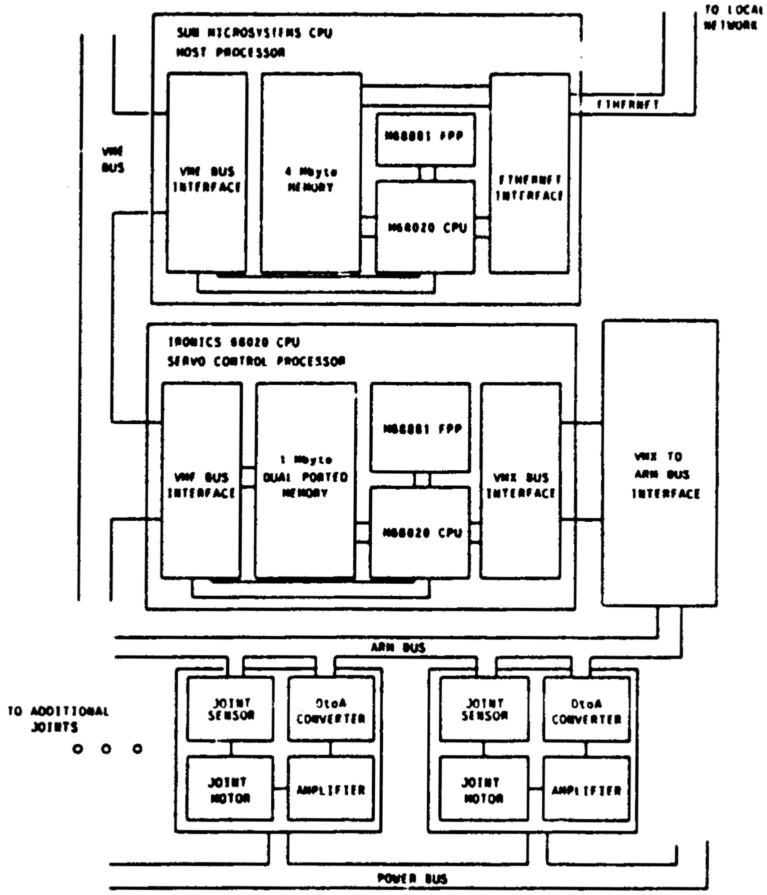


Figure 2-6: Schematic of RMMS Computing Architecture

2.5. RMMS Computing Architecture

The realtime control software in our RMMS runs on a dedicated *controller CPU*, with a hardware interface to the inter-module communication network. This controller CPU will perform the necessary realtime control of the manipulator, and receive commands from a second, *master CPU*. The intent of this architecture is to have the manipulator controller appear as a peripheral in a standard microprocessor system. Communication between the master and controller CPU will be via a number of memory mapped control and status registers maintained in shared memory on the controller CPU. This allows the controller CPU to be readily integrated into any master CPU system sharing the same system bus.

The present implementation of this architecture is shown in Figure 2-6. The controller CPU is an Ironics single-board computer, based on a Motorola 68020 processor and VME bus, with 1 MByte of dual ported RAM. The master CPU is a SUN3 workstation, also based on the Motorola 68020 and VME bus. The similarity between the two machines allows us to use the same editor and compiler for both processors, simplifying software development and inter-processor communication. The interface to the manipulator communication network is via the VMX bus interface included on the Ironics. The VMX bus is a recognized extension to the VME bus, intended as a local IO bus in multiprocessor systems such as this.

3. Span of Possible RMMS Configurations

In order to illustrate the range of capabilities of an RMMS, we have estimated several performance specifications for all of the manipulators possible given a reasonable size inventory of components. This set of specifications is used to classify the possible manipulators into groups with similar characteristics. By noting the number of different configurations within each class, we can gain an appreciation for the versatility provided by an RMMS.

3.1. Module Inventory

Based on our current design effort, let us consider an example RMMS with a module inventory consisting of:

- 25 joints, consisting of 5 sets with maximum joint torques of: 10 Nm, 30 Nm, 60 Nm, 120 Nm and 200 Nm. Each set consists of 3 pivot and 2 rotate joints.
- 12 joint position sensors, consisting of 3 each of the following resolutions. 10 bit, 12 bit, 14 bit and 16 bits.
- 20 links, consisting of 4 each of the following lengths: 0.1 m, 0.2 m, 0.5 m, 1.0 m and 2.0 m.

3.2. Obtainable Configurations and Range of Specifications

To limit the scope of this analysis, let us make the following assumptions:

- The manipulators will be 3 degree-of-freedom robots, made up of revolute joints, and will end in a conventional three axis (rotate-pivot-rotate) wrist. Thus the final joint of the three axis manipulator should not be a rotate joint, as this would be redundant.
- Two types of revolute joint assemblies will be considered: pivot and rotate, which are shown in Figure 2-1.
- Consecutive pivot joints must have either parallel or perpendicular axis orientations (in terms of the Denavit-Hartenburg convention $\alpha = 0^o$ or 90^o).
- The orientation of the manipulator with respect to the world will be ignored.

Within these assumptions, there are seven basic meaningful manipulator configurations, which are shown in Figure 3-1. Within each of the seven configurations, numerous combinations are possible by using links of assorted lengths and assorted joint module designs, resulting in a wide range of manipulator performance.

The manipulator specifications we have considered in comparing the obtainable manipulator configurations are reach, tip force, and tip position accuracy. The specifications were defined as follows:

- Reach: length of the manipulator when extened. We use the sum of the link lengths as an estimate.
- Tip force: the minimum of the three joint torque capacities divided by each joints longest moment arm. Measured in Newtons.
- Accuracy: the root mean square value of the uncertainty in tip position due to the error in measuring the joint positions. Measured in meters.

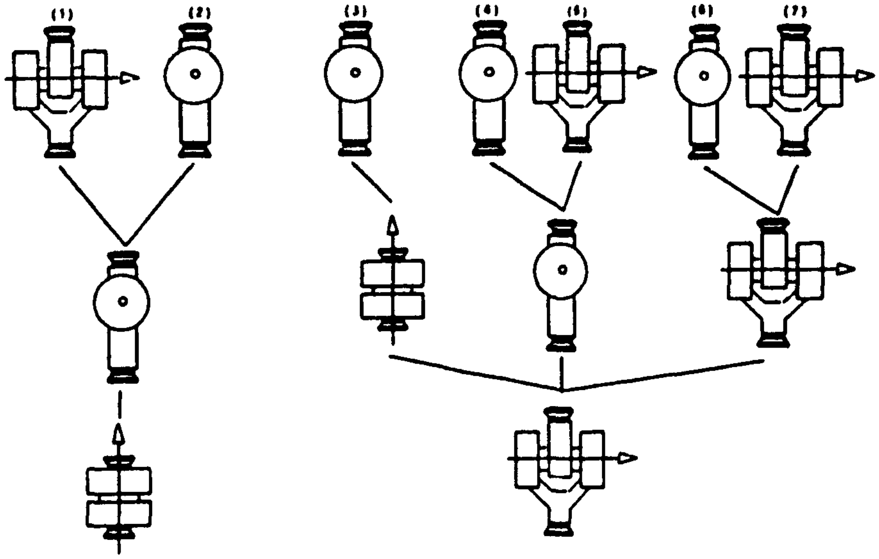


Figure 3-1: Meaningful Manipulator Configurations

We have calculated the number of obtainable arm configurations with various combinations of tip force, accuracy, and reach. The results are given as histograms in Table 3-1, which show how the obtainable configurations are distributed over the range of specifications.

We can interpret the histograms in Table 3-1 by considering the classification of typical robot tasks and performance requirements shown in Table 3-2. Ignoring speed requirements (which are generally imposed by economic thru-put constraints rather than the ability to perform a task) we can see that the obtainable range of manipulators spans each of the four major classes of task.

force											
200 N	0	232	170	10	40	0	0	0	0	0	
	104	1142	567	132	18	0	0	0	0	0	
	0	132	125	6	6	6	0	0	0	0	
	0	328	545	28	44	4	0	0	0	0	
	212	1513	1772	706	220	0	26	0	0	0	
	0	1497	1075	495	567	38	8	0	0	0	
	170	2058	2555	2173	1306	188	202	14	4	0	
	125	3136	4377	1350	3191	600	814	180	53	0	
	0	4773	8994	7376	11028	1800	3589	913	207	0	
20 N	0	0	6595	5873	10643	2614	5861	1518	611	0	reach
	0.5 m									5.0 m	

position error											
0.05 m	0	0	0	0	0	0	0	0	16	0	
	0	0	0	0	0	0	8	20	12	0	
	0	0	0	0	0	0	72	60	44	0	
	0	0	0	0	16	38	355	137	15	0	
	0	0	0	16	303	114	723	120	97	0	
	0	0	0	388	1751	521	360	217	1	0	
	0	0	138	2115	2604	294	934	88	46	0	
	0	4	2537	789	995	345	478	214	71	0	
	0	2616	4617	2844	4715	832	1528	294	98	0	
0.005 m	448	5892	6596	3256	3504	544	918	194	48	0	reach
	0.5 m									5.0 m	

Table 3-1: Histogram of Obtainable Manipulator Configuration Distributions

4. Summary

An RMMS as described in this paper can provide a wide range of manipulator performance from a small amount of hardware. This is very important for performing tasks which are either poorly defined, often one time occurrences, or located in isolated environments. Examples of such tasks include nuclear plant maintenance and practically any type of manipulation required aboard a spacecraft. In addition to the increase in capability afforded by an RMMS, there are various side benefits to such a modular design which make it very attractive for commercial use, most notably the ease of maintaining, modifying, and updating such a system.

We are now building a small RMMS prototype. As of January 1987, two joint modules and four link modules have been machined, and are now being integrated with a computer control system. At the same time, theoretical work is under way to develop algorithms for generating manipulator kinematics, dynamics, and control. A major effort is being made to reduce the size and weight of the current joint design, which is a limiting factor in both manipulator speed and payload. First operation of the system is expected in the second quarter of 1987.

TASK SPECIFICATION	MANIPULATOR PERFORMANCE REQUIREMENTS			
	REACH (meters)	PAYLOAD (kg)	ACCURACY (mm)	SPEED (m/sec)
CLASS I	0.3 - 0.75	0.5 - 1.0	0.05 - 0.5	0.5 - 2.0
CLASS II	0.75 - 1.5	1.0 - 5.0	0.1 - 2.0	0.5 - 2.0
CLASS III	0.75 - 1.5	5.0 - 25.0	2.0 - 15.0	0.2 - 1.0
CLASS IV	1.5 - 3.0	1.0 - 5.0	2.0 - 15.0	0.5 - 2.0

Examples of typical class applications:

Class I - precision assembly, printed circuit component insertion.

Class II - small part handling, pick and place assembly, loading and unloading of machine tools

Class III - large part handling, high contact force operations, surface grinding and deburring

Class IV - seam tracking for welding or sealing, spray painting

Table 3-2: Manipulator Task Classification

Acknowledgments

The authors would particularly like to thank Mark DeLouis for his contributions to the RMMS design, and for machining and assembling the prototype RMMS hardware in time to appear in this presentation. We would also like to thank Regis Hoffman, Laura Kelmar, Pradeep Khosla, and Jim Moody for contributing their experience and ideas to the RMMS design, and a great deal of effort to its continual development and implementation.

References

[1] K. H. Wurst.
The Conception and Construction of a Modular Robot System.
In *Proceeding of the 16th International Symposium on Industrial Robotics*, pages 37-44. ISIR Organizing Committee, 1986.

[2] D. P. Kuban and H. L. Martin.
An Advanced Remotely Maintainable Force-Reflecting Servomanipulator Concept.
In *Proceeding of the 1984 National Topical Meeting on ROBOTICS AND REMOTE HANDLING IN HOSTILE ENVIRONMENTS*, pages 407-415. American Nuclear Society, 1984.

S18-37
138
188167

Nonlinear Feedback Control of Multiple Robot Arms

T.J. Tarn and X. Yun
Washington University
St. Louis, MO 63130

A.K. Bejczy
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

WG

1. Abstract

~~In this paper we model~~ multiple coordinated robot arms are modeled by considering the arms (1) as closed kinematic chains and (2) as a force constrained mechanical system working on the same object simultaneously. In both formulations a new dynamic control method is discussed. It is based on a feedback linearization and simultaneous output decoupling technique. Applying a nonli ear feedback and a nonlinear coordinate transformation, the complicated model of the multiple robot arms in either formulation is converted into a linear and output decoupled system. The linear system control theory and optimal control theory are used to design robust controllers in the task space. The first formulation has the advantage of automatically handling the coordination and load distribution among the robot arms. In the second formulation, by choosing a general output equation we could superimpose the position and velocity error feedback with the force-torque error feedback in the task space simultaneously.

2. Introduction

The notion of "multiple robot arms" originates from two everyday scenarios. The first scenario is an authropomorphic one by noting that humans have two arms and hands and everyday manual work is normally performed by two-handed humans. In fact, manual activities and tasks are normally perceived and designed such that they assume two-handed humans; a one-handed person is a handicapped person from that point of view. Thus, in order to rͻplace humans with robots to perform normal manual activities it seems natural to visualize and design robots with two arms and hands. The second scenario is an industrial one by noting that production lines in industry assume an organized distribution of manipulative activities along the production line that can be carried out by a distributed set of robot arms in a proper arrangement.

Scenarios of multiple robot arms are also assumed and predicted for space applications in a natural way. Space station assembly, maintenance and servicing will require the in-site manual work of EVA astronauts in the initial operational configuration. This manual work also includes the simultaneous activities of two or more EVA astronauts in the handling or assembly of large structural elements in space. Most satellite servicing and maintenance operations also assume two-handed manual work of EVA astronauts. Thus, the objective of decreasing EVA activities in Earth orbit by introducing and increasing robot activities there requires the consideration and the design of the control of multiple robot arms.

The technically interesting and challengine problems in the control of multiple robot arms arise when (i) the work envelopes of two or more robot arms overlap and (ii) two or more robot arms simultaneously work on the same object in a presumably cooperative manner to perform a given task which cannot be performed by one arm only.

The Control problem of two or multiple robot arms has been studied by many investigators [1-12]. Although the control problem of two or multiple arms is complex, some examples of applications, such as a two-arm lathe loader, a two-arm robot press loader/unloader, and two single-arm robots working together to handle stamping press loading and unloading, are given by Chimes [1]. In these applications, the problem is solved specifically. The system design is based on a solid understanding of the problem.

Hemami and Wyman [2] investigated the problem of force control in closed chain dynamic systems. In their work, the dynamic system is linearized about an operating point and linear feedback is used to maintain the forces of constraints. The validity of the method is restricted to a rather small neighborhood of the operating point in which the dynamic system can be linearized. Orin and Oh [3] considered the control of force distribution in robotic mechanisms containing closed kinematic chains. The problem of solving for the input joint torques from a given trajectory is underspecified. The linear programming has been used to obtain a solution which optimizes a weighted combination of energy consumption and load balancing. The dynamic equations of the mechanisms are excluded from the control method. The stability of the control algorithm is in no way ensured. Ishida [4] developed a force control technique which uses a wrist force sensor to measure the interactive force between two arms. The parallel transfer task and the rotational transfer task are considered only. The control algorithm is derived for both master/slave mode and indistinguished mode (the same status mode). Fujii and Kurono [5] proposed the method of virtual reference. This method consists of the identification of the joint control mode required to perform a desired Cartesian motion. The control loop at each joint uses only position feedback and no compensation for the coupling between joints.

Alford and Belyeu [6] have designed a hierarchical computer control structure for two PUMA robot arms operating in a master/slave mode. The proposed coordinated control system has joint position predictors, a coordinate transformation, and a slave command modifier. An explicit control algorithm is derived and tested/implemented for an experimental path: a straight line in the vertical direction. However, the question on how to define the prediction function, the transformation, and the modification function is left open in the paper, and the dynamics of the arms is excluded from the algorithm.

When two robot arms work on an object certain constraints must be satisfied in order to carry out a smooth, coordinated operation. Zheng and Luh [7] have derived a set of holonomic constraints on positions and orientations of the end effectors for two robots in three specific working conditions, namely, handling a rigid-body object, handling a pair of pliers, and handling an object having a spherical joint. The result is extended to the constraints between joint velocities and accelerations of the two robots for the three above mentioned cases [8].

Considering tasks of transferring an object by holding it with two robot arms, Lim and Chyung [9] introduced a position control method using kinematic relations between the object and the two robot arms. By first specifying the trajectory of the object, the differential changes of each robot hand are computed from the differential changes of the planned path. The commands or differential changes of each joint of the two robot arms are generated by applying the inverse Jacobian matrix. The method is simple but applicable only when the involved motion is very slow. Freund and Hoyer [10-12] proposed a hierarchical control method for collision avoidance in multi-robot systems. The method adopts a hierarchical coordinator and is systematic. However, an algorithm is needed to design the couplings among robots. Vukobratovic and Potkonjak [13] described a method which can be used to obtain the closed chain dynamics of two coordinated robot arms. However, the reaction force and reaction moment between the two arms are retained in the final equations. Hayati [20] extended the idea of hybrid position/force control to the multi-arm case. Based on equations of motion for a multi-arm system, which are derived in a constrained coordinate frame located at the grasped object, a controller is designed to cooperate n robot arms such that the load is shared among the arms in a non-conflicting way. A minimization of the magnitude of forces and torques is performed to decide how much each robot arm should contribute. It appears that the existing coordinated control methods fall in lack of either systematic synthesis of the control system or full consideration of robot arm dynamics.

In this paper we concentrate on the application of nonlinear feedback to the control of multiple robot arms. Previously we derived a general algorithm for the control of a single rigid robot arm through nonlinear feedback and state transformation resulting exact system linearization and simultaneous output decoupling [15,16]. Our control design technique elevates the robot arm servo problem from the joint space to the task space with three important consequences. (i) On the joint level our scheme computes and commands drive forces or torques on their actuator-equivalent quantities (current, voltage, pressure). (ii) The robot arm system in the task space is considered as a linear system, and the powerful tools of linear control theory, including optimal control, are applicable to robot arm controller design in the task space. (iii) Our controller can directly respond to task space commands provided that these commands are formulated in form of closed time functions. The question discussed in this paper is: how can our control method be applied to the control of multiple robot arms.

We are discussing two modeling approaches. In the first approach, we model the multiple arm system as a single system, that is, as a closed loop kinematic chain. In the second approach we retain the single arm models, but we introduce task constraints and force-moment measurements in the control scheme. The paper concludes with a brief discussion of computational architectures that are needed to implement our control technique for the control of multiple robot arms.

3. Closed Chain Formulation

As the first approach to coordinated control of multiple robot arms, we consider the multiple robot arms as a single mechanical system consisting of kinematic closed chains. For tasks of lifting a heavy workpiece using robot arms, two or more robots are required if the workpiece is out of loading limit of any available robot arm. Suppose that m robot arms are used in such a task and that they all grasp on the same object (workpiece) in order to lift it, turn it, etc. Our primary concern is to obtain a dynamic model of these robots for the control purpose. Since they grasp on the same object, the dynamic behavior of one robot is not independent of the dynamic behavior of the other robots any more. A unity of mechnical system is rather formed by the robot arms involved and by the grasped object.

We will derive the Lagrange's equations of motion for this mechnical system. Those equations will serve as a model of the system to design control algorithms. For the m robots of consideration, we name them robot 1, robot 2, ..., and robot m, respectively. We assume that robot i has n_i links. We also assume that each robot firmly grasps the object so that there is no movement between its end effector and the object. Closed chains are formed in such a configuration by the m robot arms, the object, and the ground. Notice that the object and the last links of the robot arms become a single link. From the Kutzbach-Grubler criterion [17], the degrees of freedom of a spatial linkage structure connected by joints with each joint possessing one degree of freedom are given as follows

$$p = 6(i-1) - 5j \tag{1}$$

where i is the number of links and j is the number of joints. This formula reflects the fact that each moving link has six degrees of freedom and the fixed link (the ground) has none, and that each joint of one degree of freedom causes a loss of five degrees of freedom for a link. For our case of m robots, the degrees of freedom of this entire mechanical system is then

$$p = 6[\sum_{k=1}^{m} (n_k-1) + 1] - 5 \sum_{k=1}^{m} n_k = \sum_{k=1}^{m} n_k - 6m + 6 \tag{2}$$

where n_k is the number of links of robot k. If three robot arms are involved to perform a task, Table 1 shows 10 different combinations of three robot arms with five, six or seven degrees of freedom.

Before proceeding, let us define some notations that will be used in the rest of this section.

$\theta^i = [\theta^i_1 \ \theta^i_2 \ \dots \ \theta^i_{n_i}]'$: joint variables of robot i

$\theta = [(\theta^1)' \ (\theta^2)' \ \dots \ (\theta^m)']'$: joint variables of the mechanical system

$q = [q_1 \ q_2 \ \dots \ q_p]'$: generalized coordinates

$\tau = [\tau_1 \ \tau_2 \ \dots \ \tau_p]'$: generalized forces corresponding to q

$F^i = [F^i_1 \ F^i_2 \ \dots \ F^i_{n_i}]'$: joint force/torque of robot i

$F = [(F^1)' \ (F^2)' \ \dots \ (F^m)']'$: joint force/torque of the mechanical system

$n = n_1 + n_2 + \dots + n_m$.

The generalized coordinates q can be chosen arbitrarily as long as they are linearly independent of each other. They are functionally related to the joint variables θ. We denote the relation by

$$q = Q(\theta) \ . \tag{3}$$

Knowing the generalized coordinates q, the configuration of the mechanical system, thus the joint variable θ, is uniquely determined. We denote such inverse relation by

$$\theta = \Theta(q) \ . \tag{4}$$

With the above notations, the Lagrange's equations of motion for the mechanical system are described by

$$\left(\frac{\partial\Theta}{\partial q_i} \right)' \left(\frac{\partial^2 L}{\partial \dot{\theta}^2} \frac{\partial\Theta}{\partial q} q + \frac{\partial^2 L}{\partial \dot{\theta}^2} \begin{bmatrix} \dot{q}' \dfrac{\partial^2 \Theta_1}{\partial q^2} \dot{q}' \\ \vdots \\ \dot{q}' \dfrac{\partial^2 \Theta_n}{\partial q^2} \dot{q} \end{bmatrix} + \frac{\partial^2 L}{\partial \theta \partial \dot{\Theta}} \frac{\partial\Theta}{\partial q} \dot{q} - \left(\frac{\partial L}{\partial \theta} \right)' \right) = \left(\frac{\partial\Theta}{\partial q_i} \right)' F$$

$$i = 1, 2, \dots, p \tag{5}$$

where L is the Lagrangian of the whole mechanical system. Equation (5) is a generalization of the equations of motion of two robot arms presented in [14].

We assign a coordinate frame to each link of every robot arm. We locate a world coordinate frame in the common work space of the m robots. In the process of expressing the kinetic and potential energies of the mechanical system, we divide the mass of the object into m parts. Each robot is responsible for one part of the object mass by adding it to the mass of the last link. After carrying out the derivations of the Lagrangian function, we obtain, the dynamic equations of the mechanical system

$$D(q)\ddot{q} + E(q, \dot{q}) + G(q) = J_\theta' F \qquad (6)$$

where

$$D(q) = J_\theta' \tilde{D}\,(\theta(q))\; J_\theta$$

$$J_\theta = \frac{\partial\theta}{\partial q}$$

$$\tilde{D}(\theta) = \begin{bmatrix} D^1 & & & O \\ & D^2 & & \\ & & \ddots & \\ O & & & D^m \end{bmatrix}$$

$D^r = (D^r_{ij})$ is the inertia matrix of robot r

$$D^r_{ij} = \sum_{k=\max(i,j)}^{n_r} \text{Trace}\;\left(\frac{\partial T^r_k}{\partial\theta^r_i}\; I^r_k\; \frac{\partial (T^r_k)'}{\partial\theta^r_j}\right)$$

$$E(q,\dot{q}) = J_\Theta' \tilde{D}\;(\Theta(q)) \begin{bmatrix} \dot{q}' \dfrac{\partial^2\Theta_1}{\partial q^2}\dot{q} \\ \cdot \\ \cdot \\ \cdot \\ \dot{q}' \dfrac{\partial^2\Theta_n}{\partial q^2}\dot{q} \end{bmatrix} + J_\Theta' \begin{bmatrix} \dot{q}' J_\Theta' E_1 \\ \cdot \\ \cdot \\ \cdot \\ \dot{q}' J_\Theta' E_n \end{bmatrix} J_\Theta \dot{q}$$

$$E_i = \begin{bmatrix} E^1_i & & & O \\ & E_i{}^2 & & \\ & & \ddots & \\ O & & & E_i{}^m \end{bmatrix} \quad , \quad '=1, \ldots, n$$

$E^r_i = (E^r_{ijk})$ is the coefficient of centripetal (j=k) or Coriolis (j≠k) force of robot r

$$E^r_{ijk} = \sum_{s=\max(i,j,k)}^{n_r} \text{Trace}\;\left(\frac{\partial T^r_s}{\partial\theta^r_k\partial\theta^r_j}\; I^r_s\; \frac{\partial (T^r_s)'}{\partial\theta^r_i}\right)$$

$$G(q) = - J'_\Theta \begin{bmatrix} G^1 \\ G^2 \\ \cdot \\ \cdot \\ \cdot \\ G^m \end{bmatrix} , \qquad G^r = \begin{bmatrix} G_1^r \\ \cdot \\ \cdot \\ \cdot \\ G_{n_r}^r \end{bmatrix} \text{ is the gravity force of roLot r}$$

$$G_i^r = \sum_{k=i}^{n_r} m_k^r \, g' \frac{\partial T_k^r}{\partial \theta_i^r} \, \bar{r}_k^r \; .$$

In the above definitions, $T_i^r = A_{01}^r A_{12}^r \cdots A_{(i-1)i}^r$, where A_{ij}^r is the Denavit-Hartenberg homogeneous trar.sformation matrix from coordinate frame i to coordinate frame j of robot r; m_i^r is the mass of link i of robot r; $\bar{r}_i^r$ is the mass center of link i of robot r; I_i^r is the pseudoinertia matrix of link i of robot r; g is the acceleration of gravity, defined to be a 4x1 column vector with the last component being equal to zero.

Equation (6) characterizes the dynamic behavior of the whole mechanical system. However, this equation is nonlinear, coupled, and complicated. It poses great difficulty in controller designs. We propose to linearize and output decouple the system (6) using a nonlinear feedback and a nonlinear coordinate transformation. Let us introduce a state space variable x by setting

$$x_i = q_i \, , \quad x_{i+p} = \dot{q}_i \quad , \quad i=1, 2, \ldots, p$$

$$x^1 = [x_1 \; x_2 \; \ldots \; x_p]' \quad , \quad x^2 = [x_{p+1} \; \ldots \; x_{2p}]'$$

$$x = \begin{bmatrix} x^1 \\ x^2 \end{bmatrix}$$

The dynamic equation (6) can be written as

$$\begin{aligned} \dot{x} &= \begin{bmatrix} x^2 \\ -D^{-1}(x^1)[E(x^1,x^2)+G(x^1)] \end{bmatrix} + \begin{bmatrix} 0 \\ D^{-1}(x^1) \; J'_\Theta \end{bmatrix} F \\ &= f(x) + g(x)F \end{aligned} \tag{7}$$

We take the position (orientation) of the object handled as the system output

$$y = h(x^1) = [h_1(x^1) \; h_2(x^1) \; \ldots \; h_p(x^1)]'. \tag{8}$$

For the nonlinear feedback, the so-called decoupling matrix is [15,16]

$$A(x) = J_h(x^1) \; D^{-1}(x^1) \; J'_\Theta$$

where J_h is the Jacobian matrix of h. The nonlinear feedback has the form

$$F = \alpha(x) + \beta(x) \; u$$

where $\alpha(x)$ and $\beta(x)$ are determined from the following two algebraic equations [15,16]

$$A(x) \; \alpha(x) = -L_f^2 h \tag{9}$$

$$A(x) \; \beta(x) = \gamma. \tag{10}$$

In the above equations, L_f^2h is the second order Lie derivative of h along f,

$$\gamma = \begin{bmatrix} \gamma_1 & & & 0 \\ & \gamma_2 & & \\ & & \ddots & \\ 0 & & & \gamma_p \end{bmatrix} ,$$

γ_i = [1 1 ... 1] is a $1xm_i$ new vector with all entries equal to 1 and m_i, i=1, ... p, are chosen such that $m_i > 0$ and $m_1 + m_2 + \ldots + m_p = n$. The index m_i is associated with the fact that a total number of n independent actuators (inputs) are to be divided into p groups to control p outputs. The required nonlinear coordiante transformation is given by [15,16]

$$\phi(x) = [h_1 \; L_f h_1 \; \ldots \; h_p \; L_f h_p]' \; .$$

Since both equations (9) and (10) are underdetermined, there are infinite many solutions for them. Any solution serves the purpose of linearization and decoupling provided that $\beta(x)$ is invertible. A solution to equation (9) is given by [18]

$$\alpha(x) = - A^+(x) \; L_f^2h(x) \qquad (11)$$

where $A^+ = A'(AA')^{-1}$ is the generalized inverse of A(x). The general solution to equation (10) is [18]

$$\beta(x) = A^+(x)\gamma + (I - A^+A) \; H \qquad (12)$$

where H is an arbitrary matrix which is to be chosen to make $\beta(x)$ invertible.

After applying the nonlinear feedback and the nonlinear coordinate transformation, the original system (7) with output (8) is converted into the following linear and decoupled system

$$\dot{z} = Az + Bu \qquad (13a)$$

$$y = Cz \qquad (13b)$$

where

$$A = \begin{bmatrix} A_1 & & 0 \\ & \ddots & \\ 0 & & A_p \end{bmatrix} , \qquad B = \begin{bmatrix} B_1 & & 0 \\ & \ddots & \\ 0 & & B_p \end{bmatrix} , \qquad C = \begin{bmatrix} C_1 & & 0 \\ & \ddots & \\ 0 & & C_p \end{bmatrix}$$

$$A_i = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} , \quad B_i = \begin{bmatrix} 0 \\ \gamma_i \end{bmatrix} , \quad C_i = [1 \; 0] \; , \quad i=1, \ldots, p.$$

Note that the obtained linear system (13) consists of p independent subsystems. The control problem of the whole mechanical system is then simplified to a design problem of individual subsystems. The ith subsystem is defined by

$$\begin{bmatrix} \dot{z}_{2i-1} \\ \dot{z}_{2i} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_{2i-1} \\ z_{2i} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u^i \qquad (14a)$$

$$y_i = [1 \quad 0] \begin{bmatrix} z_{2i-1} \\ z_{2i} \end{bmatrix} , \quad i = 1, \ldots, p \qquad (14b)$$

where u^i is the ith group input with m_i components. To stabilize the subsystem (14), we introduce a constant feedback $u^i = - K^i z^i + v^i$ with

$$K^i = \begin{bmatrix} 0 & 0 \\ k_{i1} & k_{i2} \end{bmatrix}$$

where $z^i = [z_{2i-1} \quad z_{2i}]'$, and v^i is the new reference input. With such a constant feedback, subsystem (14) becomes

$$\begin{bmatrix} \dot z_{2i-1} \\ \dot z_{2i} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{i1} & -k_{i2} \end{bmatrix} \begin{bmatrix} z_{2i-1} \\ z_{2i} \end{bmatrix} + \begin{bmatrix} 0 \\ \gamma_i \end{bmatrix} v^i \tag{15a}$$

$$y_i = [1 \quad 0] \begin{bmatrix} z_{2i-1} \\ z_{2i} \end{bmatrix} , \qquad i = 1, \ldots, p, \tag{15b}$$

or in compact form

$$\dot z^i = \bar A_i z^i + B_i v^i$$

$$y_i = C_i z^i$$

where $\bar A_i$ can be easily identified from equation (15a). For the above system (15), the damping ratio ξ and the natural frequency ω_n are related with the feedback gains by

$$\omega_n^2 = k_{i1} \qquad\qquad 2 \xi \omega_n = k_{i2}.$$

We now consider equation (15) as the new mathematical model of the real system which is exactly linearized, output decoupled and stabilized. The desired (nominal) input to each subsystem can be derived from the following system

$$\begin{bmatrix} \dot z^d_{2i-1} \\ \dot z^d_{2i} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{i1} & -k_{i2} \end{bmatrix} \begin{bmatrix} z^d_{2i-1} \\ z^d_{2i} \end{bmatrix} \begin{bmatrix} 0 \\ \gamma_i \end{bmatrix} (v^i)^d \tag{16a}$$

$$y_i^d = [1 \quad 0] \begin{bmatrix} z^d_{2i-1} \\ z^d_{2i} \end{bmatrix} , \qquad i = 1, \ldots, p \tag{16b}$$

where the superscript "d" indicates "desired" quantities. From equation (16), the desired input can be obtained in terms of the desired task space trajectory.

$$\gamma_i (v^i)^d = y_i^d + k_{i2} \dot y_i^d + k_{i1} y_i^d , \quad i = 1, \ldots, p. \tag{17}$$

It is observed that the left hand side of equation (17) is the sum of m_i inputs in task space computed from the planned trajectory. For a given planned trajectory, at any instant time the right hand side of equation (17) is a given value. Applying the generalized inverse, we obtain [18]

$$(v^i)^d = \gamma_i (\quad \gamma \; \gamma_i')^{-1} (y_i^d + k_{i2} \dot y_i^d + k_{i1} y_i^d). \tag{18}$$

Note that in our control design methodology the actual control vector is the task space command as formulated by equation (17). On the joint level, our methodology computes drive forces or torques for the individual actuators, and the servo design is on the task level.

Let the output error be defined as follows:

$$e_i = \begin{bmatrix} e_{i1} \\ e_{i2} \end{bmatrix} = \begin{bmatrix} y_i - y_i^d \\ \dot{y}_i - \dot{y}_i^d \end{bmatrix}$$

where y_i and $\dot{y}_i$ are the real (measured) values, and y_i^d and $\dot{y}_i^d$ are the desired values. To eliminate the output error e_i, we utilize an optimal error correcting control loop by minimizing the following cost functional

$$J = \int_0^T [(\Delta v^i)' R \ \Delta v^i + e_i(t)' Q e_i(t)] dt + e_i(T)'S e_i(T).$$

The optimal correction is given by

$$\Delta v^i = -R^{-1} B_i' P(t) e_i(t) \tag{19}$$

where P(t) is a positive definite solution of the Riccati equation

$$\dot{P}(t) = -P(t) \bar{A}_i - \bar{A}_i' P(t) + P(t) B_i R^{-1} B_i' P(t) - Q$$
$$P(t) = S,$$

with

$$\bar{A}_i = \begin{bmatrix} 0 & 1 \\ -k_{i1} & -k_{i2} \end{bmatrix}.$$

The overall structure of the controller design is depicted in Figure 1.

4. Force Control Approach

In this approach, we consider the dynamics of each robot separately, but we pose constraints on the dynamic equations by introducing the interactive force and interactive moment among the robot arms.

We have proposed a force control approach to the coordination of two robot arms performing a single task [19]. The coordination between two robot arms is achieved by monitoring the interactive force and moment at the end effectors. Now we extend this method to multi-arm case.

Suppose that m robot arms ($m \geq 2$) are working on an object, e.g., lifting or turning a heavy workpiece. The problem we are dealing with is to find a control algorithm for m robots such that the task is performed in a coordinated fashion. We assume that each robot has a force (torque) sensor installed at its end effector. Using force control approach, the coordination among m robot arms is realized by regulating the force and moment applied to t object by each robot. With the aid of proper task planning, m robot arms are able to move a non-conflicting way.

The dynamic equations of a system of m robot arms are given as follows:

$$D_i(q^i) \ddot{q}^i + E_i(q^i, \dot{q}^i) + J_i'(q^i) F^i = \tau^i \ , \quad i = 1, 2, \ldots, m$$

where q^i is an n_i-dimensional joint variable vector of robot i, n_i is the degrees of freedom of robot i, F^i is an n_i-dimensional vector of the force and moment measurements of robot i, τ^i is an n_i-dimensional joint torque (force) vector of robot i, and J_i is the Jacobian matrix of robot i.

Now we introduce a state variable x by letting

$$x^i = q^i, \qquad x^{m+i} = \dot{q}^i, \qquad i = 1, \ldots, m,$$

i.e.,

$$x^1 = [x_1 \ x_2 \ \ldots \ x_{n_1}]' = [q_1^1 \ q_2^1 \ \ldots \ q_{n_1}^1]' = q^1,$$

$$x^2 = [x_{n_1+1} \cdots x_{n_1+n_2}]' = [q_1^2 \cdots q_{n_2}^2]' = q^2,$$

$$\vdots$$

$$x^m = [x_{n_1+\ldots+n_{m-1}+1} \cdots x_n]' = [q_1^m \cdots q_{n_m}^m]' = q^m,$$

$$x^{m+1} = [x_{n+1} \cdots x_{n+n_1}]' = [\dot{q}_1^1 \cdots \dot{q}_{n_1}^1]' = \dot{q}^1$$

$$x^{m+2} = [x_{n+n_1+1} \cdots x_{n+n_1+n_2}]' = [\dot{q}_1^2 \cdots \dot{q}_{n_2}^2]' = \dot{q}^2$$

$$\vdots$$

$$x^{2m} = [x_{n+n_1+\ldots+n_{m-1}+1} \cdots x_{2n}]' = [\dot{q}_1^m \cdots \dot{q}_{n_m}^m]' = \dot{q}^m,$$

where $n = n_1 + n_2 + \ldots + n_m$. Then x is a 2n-dimensional vector partitioned into 2m blocks

$$x = [x_1 \ \ x_2 \cdots x_n \ x_{n+1} \cdots x_{2n}]' = \begin{bmatrix} x^1 \\ \vdots \\ x^m \\ x^{m+1} \\ \vdots \\ x^{2m} \end{bmatrix},$$

$$\tilde{x} = [x_1 \ \ldots \ x_n]$$

with the first m blocks (corresponding to the first n components $\tilde{x}$) representing the joint positions of m robots and with the last m blocks repressnting the joint valocities of m robots.

The dynamic equations of m robots can now be written in terms of state variable x as follows:

$$\dot{x} = \begin{bmatrix} \dot{x}^1 \\ \vdots \\ \dot{x}^m \\ \dot{x}^{m+1} \\ \vdots \\ \dot{x}^{2m} \end{bmatrix} = \begin{bmatrix} x^{m+1} \\ \vdots \\ x^{2m} \\ -D_1^{-1}(x^1)[E_1(x^1,x^{m+1})+J_1'(x^1)F^1] \\ \vdots \\ -D_m^{-1}(x^m)[E_m(x^m,x^{2m})+J_m'(x^m)F^m] \end{bmatrix} + \begin{bmatrix} \bigcirc \\ D_1^{-1}(x^1) \\ \vdots \\ D_m^{-1}(x^m) \end{bmatrix} \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_m \end{bmatrix}$$

or $$\dot{x} = f(x) + g(x)\tau \qquad (20)$$

where f ang g can be easily identified from the above equation. We take the output equations of the form

$$y = h(x) = \begin{bmatrix} h^1 \\ h^2 \\ \vdots \\ h^m \end{bmatrix} = \begin{bmatrix} W_p^1 P^1 + W_F^1 F^1 \\ W_p^2 P^2 + W_F^2 F^2 \\ \vdots \\ W_p^m P^m + W_F^m F^m \end{bmatrix} \qquad (21)$$

where W_p^i, W_F^i, $i = 1, \ldots, m$, are the weighting matrices, and p^i is the position and orientation vector of robot i in the world coordinate frame. The dimension of output vector y is n.

Equation (20) represents a nonlinear and coupled system with output (21). Using a nonlinear feedback $\tau = \alpha(x) + \beta(x)u$ and a nonlinear coordinate transformation T(x), we are able to linearize and output decouple the system (20). The $\alpha(x)$ and $\beta(x)$ in the nonlinear feedback are given by

$$\alpha(x) = -A^{-1}(x)\, L_f^2\, h \qquad (22)$$

$$\beta(x) = A^{-1}(x) \qquad (23)$$

where

$$A(x) = \frac{\partial h}{\partial x} \begin{bmatrix} D_1^{-1}(x^1) & & \bigcirc \\ & \ddots & \\ \bigcirc & & D_m^{-1}(x^m) \end{bmatrix} .$$

The nonlinear transformation is given by

$$T(x) = \begin{bmatrix} h_1 \\ L_f h_1 \\ \vdots \\ h_n \\ L_f h_n \end{bmatrix} \qquad (24)$$

Application of the nonlinear feedback and the nonlinear coordinate transformation converts the system (20) with the output (21) into the following linear and decoupled system

$$\dot{z} = Az + Bu \qquad (25a)$$

$$y = Cz \qquad (25b)$$

where

$$z = [z_1 \ldots z_{2n}]' \ , \quad u = [u_1 \ldots u_n]' \ , \quad y = [y_1 \ldots y_n]' \ ,$$

$$A = \begin{bmatrix} A_1 & & \bigcirc \\ & \ddots & \\ \bigcirc & & A_n \end{bmatrix} , \quad B = \begin{bmatrix} B_1 & & \bigcirc \\ & \ddots & \\ \bigcirc & & B_n \end{bmatrix} , \quad C = \begin{bmatrix} C_1 & & \bigcirc \\ & \ddots & \\ \bigcirc & & C_n \end{bmatrix}$$

$$A_i = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} , \quad B_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix} , \quad C_i = [1 \ 0] \ , \quad i=1, 2, \ldots, n.$$

Note that system (25) consists of n independent subsystems. Likewise as in the closed chain formulation, for each subsystem we can design a constant feedback to stabilize it and design an optimal error-correcting loop to eliminate the output errors. The overall controller structure is shown in Figure 2.

5. Conclusion

Our approaches to the control problem of multiple robot arms are motivated by the desire of making rigorous use of the dynamics of robot arms involved in the task. The closed chain approach is initiated from the fact that the dynamic behaviors of the robot arms are not independent of each other any more if they grasp on a common object. In this approach, the multiple robot arms are modeled as a single mechanical system by choosing a set of generalized coordinates whose number equals the number of degrees of freedom of the whole system. Figure 1 shows the schematic structure of the controller for the closed chain approach as implemented on computers. From the initial physical task, the task planning of the upper left block in Figure 1 produces a trajectory in the task space expressed as a smooth function of time. The command generator block realizes equation (18) and yields the desired reference input. The lower left block is the implementation of the optimal error correction described by equation (19). It takes the task space error as its input, and produces the optimal correction as its output. The $Q(\theta)$ block to the right of the multiple robot arms establishes the generalized coordinates as well as their time derivatives from the measured joint positions and velocities of the robot arms. The bulk of the controller is the nonlinear feedback block which computes the joint driving torques or forces. Because the dynamic projection functions D^i, E_i, and G^i are derived in terms of the joint variables, it may be convenient to use the joint variables in addition to the generalized coordinates for computing the nonlinear feedback.

Different from the closed chain approach, the force control approach assumes that each robot arm has a force and moment sensor located at the end effector. The force and moment measurements are introduced into the dynamic equations and output (task) equations. This is schematically depicted in Figure 2. The measurements F^1, F^2, ..., F^m are transmitted to the nonlinear feedback block, the output h block, and the coordinate transformation T block. The three blocks to the left of the nonlinear feedback block in Figure 2 are structurally similar to those in Figure 1.

Using the results from differential geometric system theory, we are able to linearize and to decouple the complicated dynamic equations of multiple robot arms including the object held by the arms. Independent of the approach being taken, we eventually deal with a linear and decoupled system. Thus we can have a unified design technique for coordinated control of multiple robot arms.

It should be noted that both methods used in this paper are systematic and are robot arm independent. The most important feature is that the control algorithms are task independent, that is, there is no need to change the structure of the controller or even the parameters of the controller from task to task. As natural as would be, the change of tasks only causes the adjustment of the input command which is conveniently given in the task space rather than in the joint space. The two control methods can be used in slightly different situations. For example, if the robot arms are loosely connected through the object, the force control approach is preferable; if the robot arms are mechanically locked while transferring the object, the closed-chain approach is more likely a solution.

Each control scheme naturally leads itself for computational implementation using distributed computing system, possibly in multi-bus architecture. Figures 1 and 2 provide a high level structure of computational implementation requirements. The details of the implementation require a deeper analysis.

6. ACKNOWLEDGEMENT

The research described in this paper was jointly performed by Washington University, St. Louis, Missouri, and the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, and was jointly sponsored by the National Science Foundation and the National Aeronautics and Space Administration.

7. REFERENCES

[1] P.R. Chimes, "Multiple-Arm Robot Control Systems," Robotics Age, Oct. 1985, pp. 5-10.

[2] H. Hemami and B. Wyman, "Indirect Control of the Forces of Constraint in Dynamic System," J. Dynamic Systems, Measurement and Control, Vol. 101, 1979, pp. 355-360.

[3] D. E. Orin and S. Y. Oh, "Control of Force Distribution in Robotic Mechanisms Containing Closed Kinematic Chains," J. Dynamic Systems, Measurement and Control, Vol. 102, June 1981, pp. 134-141.

[4] T. Ishida, "Force Control in Coordination of Two Arms," Proceedings of the 5th International Joint Conference on Artificial Intelligence, Aug. 1977, pp. 717-722.

[5] S. Fujii and S. Kurono, "Coordinated Computer Control of a Pair of Manipulators," I. Mech. E. (Japanese pub.) 1975, pp. 411-415.

[6] C. O. Alford, S. M. Belyeu, "Coordinated Control of Two Robot Arms," International Conference on Robotics, Atlanta, Georgia, March 13-15, 1984, pp. 468-473.

[7] Y. F. Zheng, J. Y. S. Luh, "Constrained Relations Between Two Coordinated Industrial Robots," Proc. of 1985 Conference of Intelligent System and Machines, Rochester, Michigan, April 23-24, 1985.

[8] Y. F. Zheng, J. Y. S. Luh, "Control of Two Coordinated Robots in Motion," Proceedings of the 24th IEEE Conference on Decision and Control, Fort Lauderdale, Florida, Dec. 11-13, 1985, pp. 1761-1765.

[9] J. Lim, D. H. Chyung, "On a Control Scheme for Two Cooperating Robot Arms," Proc. of 24th Conference on Decision and Control, Fort Lauderdale, Florida, Dec. 11-13, 1985, pp. 334-337.

[10] E. Freund, H. Hoyer, "Collision Avoidance in Multi-Robot Systems," The Second International Symposium of Robotics Research, Kyoto-Kaikan, Kyoto, Japan, Aug. 20-23, 1984, pp. 135-146.

[11] E. Freund, "On the Design of Multi-Robot Systems," International Conference on Robotics, Atlanta, Georgia, March 13-15, 1984, pp. 477-490.

[12] E. Freund, H. Hoyer, "On the On-Line Solution of the Findpath Problem in Multi-Robot Systems," The Third International Symposium of Robotics Research, Gouvieux, France, Oct. 7-11, 1985.

[13] M. Vukobratovic and V. Potkonjak, Applied Dynamics and CAD of Manipulation Robots, Springer-Verlag, Berlin, 1985.

[14] T. J. Tarn, A. K. Bejczy, and X. Yun, "Design of Dynamic Control of Two Cooperating Robot Arms: Closed Chain Formulation," Proceedings of 1987 IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, March, 1987.

[15] Y. Chen, Nonlinear Feedback and Computer Control of Robot Arms, D.Sc. Dissertation, Washington University, St. Louis, Missouri, Dec. 1984.

[16] T. J. Tarn, A. K. Bejczy, A. Isidori and Y. Chen, "Nonlinear Feedback in Robot Arm Control," Proceedings of the 23rd IEEE Conference on Decision and Control, Las Vegas, December 12-14, 1984.

[17] J. E. Shigley, Kinematic Analysis of Mechanics, McGraw-Hill Book Company, 1969.

[18] F. A. Graybill, Introduction to Matrices with Applications in Statistics, Wadsworth Publishing Company, Inc., Belmont, California, 1969.

[19] T. J. Tarn, A. K. Bejczy, and X. Yun, "Dynamic Coordination of Two Robot Arms," Proceedings of 25th IEEE Conference on Decision and Control, Athens, Greece, Dec. 1986.

[20] Samad Hayati, "Hybrid Position/Force Control of Multi-Arm Cooperating Robots," Proceedings of 1986 IEEE International Conference on Robotics and Automation, San Francisco, California, April 1986, pp. 82-89.

Table 1. Degrees of freedom of the closed chains formed by three robot arms

cases	n_1	n_2	n_3	i	j	$n_1+n_2+n_3$	p
1	7	7	7	20	21	21	9
2	6	7	7	19	20	20	8
3	6	6	7	18	19	19	7
4	5	7	7	18	19	19	7
5	5	6	7	17	18	18	6
6	6	6	6	17	18	18	6
7	5	6	6	16	17	17	5
8	5	5	7	16	17	17	5
9	5	5	6	15	16	16	4
10	5	5	5	14	15	15	3

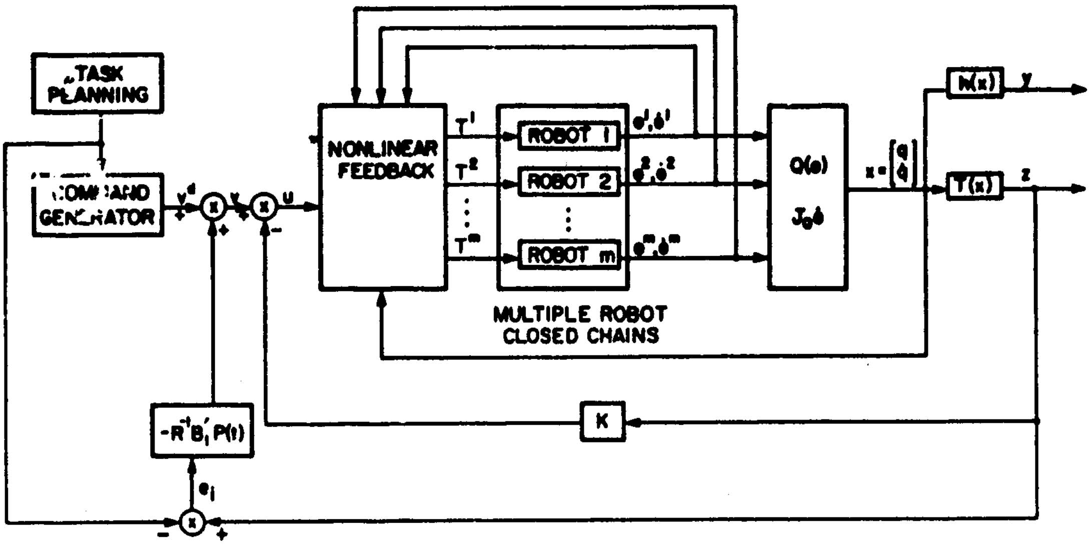


Fig. 1 Schematic Control Structure of the Closed Chain Approach

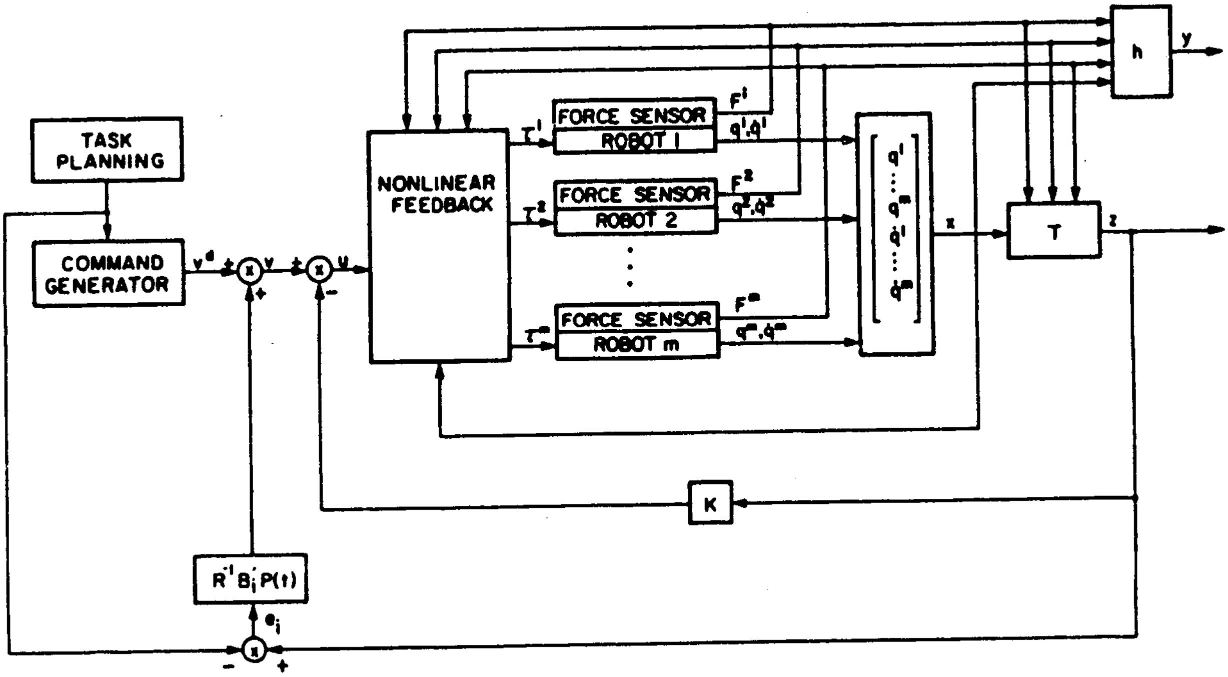


Fig. 2 Schematic Control Structure of the Force Control Approach

Dynamics and Control of Coordinated Multiple Manipulators

S.A. Hayati
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

ABSTRACT

~~In this paper we present~~ a technique is presented for controlling multiple manipulators which are holding a single object and therefore form a closed kinematic chain. The object, which may or may not be in contact with a rigid environment, is assumed to be held rigidly by n robot end-effectors. The derivation is based on setting up constraint equations which reduce the *6xn degrees of freedom of n* manipulators each having six joints. Additional constraint equations are considered when one or more of the degrees of freedom of the object is reduced due to external constraints. Utilizing the operational space dynamics equations, a decoupling controller is designed to control both the position and the interaction *forces* of the object with the environment. Finally, ~~[illegible]~~ simulation results for the control of a pair of two-link manipulators are presented.

1. INTRODUCTION

The topic of multiple robot control is relatively new in robotics research. The extension of robot control techniques to the case of multiple manipulators is necessitated by realities encountered both for manipulating small objects and for handling large workpieces. The manipulation of objects normally requires at least two hands to simultaneously position and reorient the object so that either one or both hands can perform their respective tasks. Here, the employment of an extra arm is required not by limitations on force/torque output of an arm but by the fact that in complex task execution, the workpiece must frequently be reoriented to expose the hard-to-reach areas. In fact, some experts have suggested that robots will not truly be integrated in the future factory environment until multiple robot task planning and control have reached a certain level of maturity. In this paper we will only address the control problem associated with multiple manipulators handling a commonly held object. Other problems, such as several manipulators working in the same workcell without manipulating the same objects simultaneously, or two arms working on the same objects but having *relative motion* with respect to the workpiece, are important research topics not addressed in this paper.

Early work in this area [1, 2] has mostly dealt with master/slave control techniques. This approach is conceptually simple. One manipulator is position servoed while the other manipulator is force servoed. The *force servoed arm is controlled* in compliant mode to follow the master (position servoed) arm. Additional feed forward forces (torques) are added to the force controlled arm to realize the required interaction forces between the arms and the object which is being manipulated. More recently, among others, Luh and Zheng have reported their work in dual arm coordination. We will mention only some of their work here. In reference [3] they discuss the kinematic issues related to the coordination of dual manipulator robots. Their approach is based on position servoing of one of the arms and the use of kinematic constraint equations to modify the trajectory of the second arm to realize cooperation between the arms. In reference [4] they extend their analysis to include the constraint equations for the joint acceleration. Then, using the arm Jacobian, joint torques are computed to drive the arms.

In this paper we derive the equations of motion in the so-called Operational Space (or Cartesian state space) [5]. We assume a general case of n cooperating robots which are holding an object rigidly. This object may also be constrained from motion in one or more dimensions by an external environment. Equations of motion are derived using the Lagrange multiplier technique. References [6] and [7] provide excellent background for this formulation. It is assumed that each manipulator is equipped with a force/torque sensor capable of measuring three orthogonal forces and torques in a given coordinate frame. The aim is to control the position *of the object* and its *interaction forces* with the environment in the sense of hybrid control [8]. This paper extends the previously reported results in [9] by a more compact formulation and by explicitly computing the interaction forces between the robots. Finally, a simulation study for a pair of two-link cooperating manipulators is presented to validate the analysis.

2. DYNAMICS OF MULTIPLE COOPERATING ROBOT MANIPULATORS

In this section, we will derive the dynamics equations for a closed kinematic chain formed by n manipulators rigidly grasping an object. We will assume that each manipulator has six joints and that none of the manipulators experience singularity (i.e., degenerate manipulator Jacobian) as they perform a task. The grasped object may be in contact with a rigid environment.

In deriving the equations of motion, we will assume that there exists a coordinate frame which is attached to the object. In fact, in the next section, we will use this coordinate frame to specify the desired motion/force of the object. Figure 2.1 shows a schematic drawing of this multiple manipulator system.

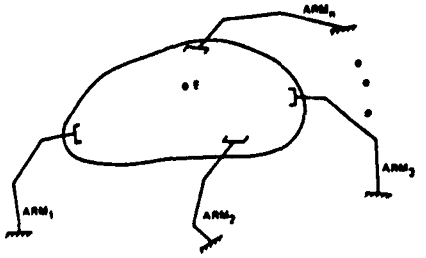


Figure 2.1.

The derivations of the equations of motion are considerably simplified when a) we use the Cartesian state dynamics equations [5,10], and b) we lump the object mass/inertia into that of the sixth link of the arms.

Let us begin with the well known equation of motion for a single multi-link arm [9,10]

$$\underline{M}(\underline{q})\,\underline{\ddot{q}} + \underline{V}(\underline{q},\underline{\dot{q}}) + \underline{G}(\underline{q}) = \underline{\tau} \qquad (2.1)$$

where $\underline{q} \in R^n$ denotes the joint state variable, $\underline{M}(\underline{q}) \in R^{nxn}$ is the inertia matrix which is symmetric and positive definite, $\underline{V}(\underline{q},\underline{\dot{q}}) \in R^n$ is the centrifugal and Coriolis force/torque vector, $\underline{G}(\underline{q}) \in R^n$ is the gravity vector, and $\underline{\tau} \in R^n$ is the joint force/torque vector. In order to simplify the wording of the paper, we will assume that all the joints are revolute and hence use the term "torque" when referring to generalized joint force/torque vectors. The above equation applies only to idealized frictionless rigid arms.

In the first part of the derivation, we will consider an object-fixed coordinate frame relative to which the dynamics equations will be written. We will also assume that the object has been partitioned into n parts. Each of these parts will then be considered as a part of the last link of each arm. Figure 2.2 illustrates one of these parts together with the last link of arm i.

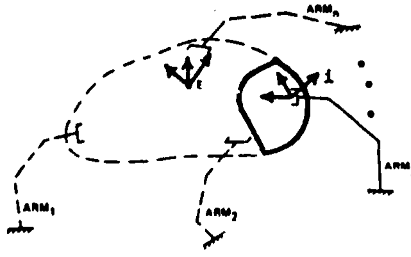


Figure 2.2. Schematic drawing of the partitioned load and its integration with the last link of arm i.

If ${}^iJ_i$ is the pseudo-inertia matrix of the last link represented in the i coordinate frame, its representation in the E frame is obtained from:

$$ {}^EJ_i = \int_{\text{last link}} {}^E\underline{r}_i \, {}^E\underline{r}_i^T \, dm_i \qquad (2.2)$$

Since the arms are considered to be rigidly attached to the load, there is a constant transformation relating the i and E frames, ${}^E_iT$. It then follows that

$$ {}^EJ_i = \int_{\text{last link}} {}^E_iT \; {}^i\underline{r}_i(dm_i) \; ({}^E_iT \; {}^i\underline{r}_i)^T = {}^E_iT \; {}^iJ_i \; {}^i_ET \qquad (2.3)$$

where ${}^EJ_i$ is the pseudo-inertia matrix of the last link of the ith arm expressed in the E frame. Let us now assume that ${}^EJ_{Ei}$ is the pseudo-inertia matrix of the ith partition of the load. This means that

$${}^{E}J_{Ei} = \int_{\text{partition } i} {}^{E}\underline{x}_{E}\, {}^{E}\underline{x}_{E}^{T}\, dm_{E} \qquad (2.4)$$

where partition i is as shown schematically in Figure 2.2. Utilizing equations (2.3) and (2.4), we can now define a new last link for the ith arm which is essentially the combination of the original last link plus a portion of the load. Also note that the coordinate frame of the last link is now E instead of the frame i.

Although in principle one can use joint space dynamics equations for the system of multiple manipulators, the derivation is considerably simplified when the so-called operational space formulation (or Cartesian state space dynamics) is used. The relationship between the joint space and the Cartesian space formulation can be derived simply by utilizing the manipulator Jacobian as [5,10]:

$$\dot{\underline{x}} = J(\underline{q})\, \dot{\underline{q}} \qquad (2.5)$$

where J is the manipulator Jacobian and $\underline{x}$ is the position/orientation of the last link coordinate frame. The Jacobian used here is a general one, meaning that one can consider it to be defined relative to the last frame, the world frame, or in fact any other desired frame. Taking the time derivative of (2.5) results in

$$\ddot{\underline{x}} = \dot{J}(\underline{q})\, \dot{\underline{q}} + J(\underline{q})\, \ddot{\underline{q}} \qquad (2.6)$$

Substituting $\ddot{\underline{q}}$ from (2.6) into (2.1), we obtain

$$M(\underline{q})\;\; J^{-1}(\underline{q})\; [\ddot{\underline{x}} - \dot{J}\dot{\underline{q}}] \;\; + \underline{V}(\underline{q},\dot{\underline{q}}) + \underline{G}(\underline{q}) = \underline{\tau}$$

or

$$M_x(\underline{q})\, \ddot{\underline{x}} + \underline{V}_x(\underline{q},\dot{\underline{q}}) + \underline{G}_x(\underline{q}) = \underline{F} \qquad (2.7)$$

where

$$M_x(\underline{q}) = J^{-T}(\underline{q})\; M(\underline{q})\; J^{-1}(\underline{q}) \qquad (2.8)$$

$$\underline{V}_x(\underline{q},\underline{q}) = J^{-1}[\; \underline{V}(\underline{q},\dot{\underline{q}}) - M(\underline{q})\; J^{-1}(\underline{q})\; \dot{J}(\underline{q})\; \dot{\underline{q}}] \qquad (2.9)$$

$$\underline{G}_x(\underline{q}) = J^{-T}(\underline{q})\; \underline{G}\; (\underline{q}) \qquad (2.10)$$

In equation (2.7), $\underline{F}$ represents an equivalent generalized force vector applied at the end-point (E-frame). Let us denote by $\underline{f}$ the interaction generalized force vector on one of the robot arms at the E-frame. The equations of motion are therefore

$$M_{xi}(\underline{q}_i)\, \ddot{\underline{x}} + \underline{V}_{xi}(\underline{q}_i,\dot{\underline{q}}_i) + \underline{G}_{xi}(\underline{q}_i) = \underline{F}_i + \underline{f}_i \quad , \; i = 1,\ldots,n \qquad (2.11)$$

Since $\underline{f}_i$'s are the internal forces created at point E, their sum is zero.

$$\sum_{i=1}^{n} \underline{f}_i = 0$$

The equation of motion, therefore, may be obtained by simply adding the equations (2.11)

$$[\sum_{i=1}^{n} M_{xi}(\underline{q}_i)]\, \ddot{\underline{x}} + \sum_{i=1}^{n} \underline{V}_{xi}(\underline{q}_i,\dot{\underline{q}}_i) + \sum_{i=1}^{n} \underline{G}_{xi}(\underline{q}_i) = \sum_{i=1}^{n} \underline{F}_i \qquad (2.12)$$

In order to simplify the notation, let us define

$$M_x(\underline{Q}) = \sum_{i=1}^{n} M_{xi}(\underline{q}_i) \qquad (2.13)$$

$$\underline{V}_x(\underline{Q},\underline{Q}) = \sum_{i=1}^{n} \underline{V}_{xi}(\underline{q}_i,\dot{\underline{q}}_i) \qquad (2.14)$$

$$\underline{G}_x(\underline{Q}) = \sum_{i=1}^{n} \underline{G}_{xi}(\underline{q}_i) \qquad (2.15)$$

$$\underline{F} = \sum_{i=1}^{n} \underline{F}_i$$

where

$$\underline{Q} = (\underline{q}_1^T,\, \underline{q}_2^T, \ldots, \underline{q}_n^T)$$

Now, equation (2.12) can be written in the simpler form

$$M_x(\underline{Q})\, \ddot{\underline{x}} + \underline{V}_x(\underline{Q},\dot{\underline{Q}}) + \underline{G}_x(\underline{Q}) = \underline{F} \qquad (2.16)$$

Note that M_x is still a positive definite matrix. The interaction generalized forces, i.e. f_i's, can be computed by solving for $\ddot{X}$ in equation (2.16) and substituting the results in (2.11)

$$f_i = M_{xi}(q_i)\, M_x^{-1}(Q)\ [F - V_x(Q,\dot{Q}) - G_x(Q)] \\ + V_{xi}(q_i,\dot{q}_i) + G_{xi}(q_i) - F_i \quad , \ i = 1,\dots,n \qquad (2.17)$$

In summary, if the individual joint torques are given, one can compute the equivalent F_i's by utilizing the Jacobian matrices. Equation (2.16) can then be used to solve the forward dynamics of the multiple arm system. Finally, the interaction forces can be computed from equation (2.17). Implicit in the above derivation are the constraint equations

$$x = H_1(q_1) = H_2(q_2) \ \dots = H_n(q)$$

where H_i is the forward kinematics relation for arm i.

3. DYNAMICS OF MULTIPLE COOPERATING ROBOT MANIPULATORS WITH EXTERNAL CONSTRAINTS

. In this section we will derive the equations of motion when the object that is being held by n robot manipulators is in contact with a frictionless rigid environment. Let C be a coordinate frame on the object to represent the generalized constraint forces. This coordinate frame, in general, will be distinct from the E-frame. With a slight modification, equation (2.16) can be transformed so that the generalized force vector F can be represented in the C frame. If ${}^E_C T$ is a homogeneous transformation that relates the C frame to the E frame, then ${}^E_C J$ can be obtained to relate the generalized force vectors in these two frames. The Jacobian matrix ${}^E_C J$ is given from

$${}^E_C J = \left[\begin{array}{c|c} {}^E_C R & {}^E P_{cORG} \times {}^E_C R \\ \hline 0 & {}^E_C R \end{array}\right] \qquad (3.1)$$

where

$${}^E_C T = \left[\begin{array}{c|c} {}^E_C R & {}^E P_{cORG} \\ \hline 0 & 1 \end{array}\right] \qquad (3.2)$$

In general, since the C-frame can be time varying, ${}^E_C T$ and consequently ${}^E_C J$ are also time varying matrices. Let x_c denote the position/orientation of the C-frame. Then

$$\dot{x}_c = {}^C_E J\ \dot{x} \qquad (3.3)$$

$$\ddot{x}_c = {}^C_E \dot{J}\ \dot{x} + {}^C_E J\ \ddot{x} \qquad (3.4)$$

and hence equation (2.16) is

$$M_{xc}(Q)\ \ddot{x}_c + V_{xc}(Q,\dot{Q}) + G_{xc}(Q) = F_c \qquad (3.5)$$

where M_{xc}, V_{xc}, G_{xc} are defined by a set of matrix equations (2.8) - (2.10) with J replaced by ${}^C_E J$ and

$$F_c = ({}^E_C J)^T\ F.$$

Let us assume that $\phi(x_c) \in R^m$, $m \leq 6$, represents the constraint function such that

$$\phi(x_c) = 0 \ \forall\ t \geq 0 \qquad (3.6)$$

If we denote by f_c the generalized constraint force vector, then the equation of motion is

$$M_{xc}(Q)\ \ddot{x}_c + V_{xc}(Q,\dot{Q}) + G_{xc}(Q) = F_c + f_c \qquad (3.7)$$

The constraint force f_c can be obtained by noting that the virtual work performed by f_c is equal to zero and by using the Lagrange multiplier method as discussed in [6] and [7].

The statement of virtual work is

$$f_c^T\ \delta x_c = 0 \qquad (3.8)$$

From the constraint equation (3.6) we can write

$$\frac{\partial \phi(x_c)}{\partial x_c} \, \delta x_c = 0 \tag{3.9}$$

We can now use the Lagrange multiplier $\lambda \in R^m$ to relate (3.8) and (3.9) as

$$[f_c^T - \lambda^T \frac{\partial \phi(x_c)}{\partial x_c}] \, \delta x_c = 0$$

Since δx_c is an arbitrary infinitesimal displacement vector, it is not equal to zero. Therefore

$$f_c = D^T(x_c) \, \lambda \tag{3.10}$$

where

$$D(x_c) = \partial \phi(x_c) \, / \, \partial x_c \tag{3.11}$$

To find λ, we differentiate (3.6) with respect to time twice

$$\dot{\phi}(x_c) = D(x_c) \, \dot{x}_c = 0 \tag{3.12}$$

$$\ddot{\phi}(x_c) = \dot{D}(x_c) \, \dot{x}_c + D(x_c) \, \ddot{x}_c = 0 \tag{3.13}$$

and substitute for $\ddot{x}_c$ from (3.7)

$$\dot{D}(x_c) \, \dot{x}_c + D(x_c) \, M_{xc}^{-1}(Q) \, [\, F_c + D^T(x_c) \, \lambda - V_{xc}(Q,\dot{Q}) - G_{xc}(Q) \,] = 0$$

which gives λ as

$$\lambda = [\, D(x_c) \, M_{xc}^{-1}(Q) \, D^T(x_c) \,]^{-1} \; \{ \, D(x_c) \, M_{xc}^{-1}(Q) \, [V_{xc}(Q,\dot{Q}) + G_{xc}(Q) - F_c] - \dot{D}(x_c) \, \dot{x}_c \, \} \tag{3.14}$$

Equation (3.7) together with equation (3.14) completely characterize the equation of motion of the multiple cooperating robot with external motion constraints on the object.

4. CONTROL OF MULTIPLE COOPERATING ROBOTS

In this section we will introduce a control technique which is based on the operational space formulation to control the position of the object and interaction forces of the object with the external environment. The approach presented here is based on distributing the actuation forces among the arms such that each arm contributes to the motion of the object and to its own inertial forces in a predetermined manner. Let us begin by introducing a decoupling generalized force such that

$$F_c = -[f_{cd} + K_{fd}(\dot{f}_{cd} - \dot{f}_c) + K_{fp}(f_{cd} - f_c)] + M_{xc}(Q) \, [\ddot{x}_{cd} + K_{pd}(\dot{x}_{cd} - \dot{x}_c) + K_{pp}(x_{cd} - x_c)]$$
$$+ V_{xc}(Q,\dot{Q}) + G_{xc}(Q) \tag{4.1}$$

where f_{cd} is the desired interaction force vector between the object and the external environment.

We must now show that the above choice for F_c does result in the multiple robotic system following the specified trajectory and applying the desired forces on the constraint surface. In order to simplify the derivation and without loss of generality, let us assume that the position and force subspaces are reordered in the equations such that the first 6-m elements of the position vector correspond to the position controlled subspace and the last m elements belong to the force controlled subspace. This means that, for example,

$$\ddot{x}_{cd} = \left[\begin{array}{c} \ddot{x}'_{cd} \\ \hline 0 \end{array}\right] \begin{array}{l} \}\ 6-m \\ \}\ m \end{array} \qquad (4.2)$$

$$\dot{x}_{cd} = \left[\begin{array}{c} \dot{x}'_{cd} \\ \hline 0 \end{array}\right] \begin{array}{l} \}\ 6-m \\ \}\ m \end{array} \qquad (4.3)$$

and

$$f_{cd} = \left[\begin{array}{c} 0 \\ \hline f'_{cd} \end{array}\right] \begin{array}{l} \}\ 6-m \\ \}\ m \end{array} \qquad (4.4)$$

Let us partition the mass matrix as

$$M_{xc} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad (4.5)$$

and its inverse as

$$M_{xc}^{-1} = \begin{bmatrix} M'_{11} & M'_{12} \\ M'_{21} & M'_{22} \end{bmatrix} \qquad (4.6)$$

where $M_{11} \in R^{6-m \times 6-m}$. With the above decomposition of the force-position subspaces, the D matrix (see equation 3.11) can be written as

$$D(x_c) = [\ 0 \mid I_{mxm}\ ] \qquad (4.7)$$

and

$$\dot{D}(x_c) = 0 \qquad (4.8)$$

With the above choice for $\phi(x_c)$, we can now compute the closed loop dynamics of the multiple manipulator system with F_c as the generalized control force vector given by equation (4.1). Let us partition the actual force, acceleration, and velocity vectors as

$$f_c = \begin{bmatrix} f_{c1} \\ f_{c2} \end{bmatrix}, \quad \ddot{x}_c = \begin{bmatrix} \ddot{x}_{c1} \\ \ddot{x}_{c2} \end{bmatrix}, \quad \dot{x}_c = \begin{bmatrix} \dot{x}_{c1} \\ \dot{x}_{c2} \end{bmatrix} \qquad (4.9)$$

and also select K_{fp}, K_{fd}, K_{pd}, K_{pp} as

$$K_{fd} = \begin{bmatrix} 0 & 0 \\ 0 & k_{fp} \end{bmatrix}, \quad K_{fp} = \begin{bmatrix} 0 & 0 \\ 0 & k_{fd} \end{bmatrix}, \quad K_{pp} = \begin{bmatrix} k_{pp} & 0 \\ 0 & 0 \end{bmatrix}, \quad K_{pd} = \begin{bmatrix} k_{pd} & 0 \\ 0 & 0 \end{bmatrix} \qquad (4.10)$$

where k_{ij}'s are diagonal matrices. From equation (4.1) F_c is given by

$$F_c = \begin{bmatrix} 0 \\ f'_{cd} \end{bmatrix} - \begin{bmatrix} 0 \\ k_{fd}\,\dot{e}_f \end{bmatrix} - \begin{bmatrix} 0 \\ k_{fp}\,e_f \end{bmatrix} + M_{xc}\left\{ \begin{bmatrix} \ddot{x}'_{cd} \\ 0 \end{bmatrix} + \begin{bmatrix} k_{pd}\,\dot{e}_p \\ 0 \end{bmatrix} + \begin{bmatrix} k_{pp}\,e_p \\ 0 \end{bmatrix} \right\} + V_{xc} + G_{xc} \qquad (4.11)$$

where $e_f = f'_{cd} - f_{c2}$ and $\dot{e}_p = \dot{x}'_{cd} - \dot{x}_{c1}$

The reaction force f_c can be computed by substituting from equation (4.11) for F_c in equation (3.10) and (3.14).

After some manipulation of the equations, one obtains

$$\underline{f}_e = \begin{bmatrix} 0 \\ f'_{ed} + k_{fd}\dot{e}_f + k_{fp}\, e_f \end{bmatrix} \qquad (4.12)$$

Substituting from (4.9) for $\underline{f}_e$, we have

$$\underline{f}_{e2} = f'_{ed} + k_{fd}\, \dot{e}_f + k_{fp}\, e_f$$

or

$$k_{fd}\, \dot{e}_f + (k_{fp} + I)\, e_f = 0$$

This result indicates that the generalized reaction force will be equal to the desired force. The control time constant can be designed by selecting the proper k_{fd} and k_{fp} diagonal gain matrices.

Similarly, by substituting from equation (4.1) into equation (3.6) and using equation (4.12), we obtain

$$M_{xc}(Q) \begin{bmatrix} \ddot{e}_p + k_{pd}\, \dot{e}_p + k_{pp}\, e_p \\ 0 \end{bmatrix} = 0 \qquad (4.13)$$

This means that in the position subspace, the multiple manipulator system can be controlled by the proper choice of the k_{pd} and k_{pp} matrices.

To summarize, a generalized force vector as given by equation (4.1) was found such that both the position of the object and its interaction forces with an environment can be controlled. Note that this equation does not specify how the generalized force vector will be realized by a system of redundant actuators of multiple cooperating manipulators. One choice is to let each arm supply enough torques at its joints to provide the necessary actuation for its own links plus a portion of the object's. Similarly, one can determine a priori how much each arm must contribute to the realization of the interaction forces between the object and the environment (for more details, please see Ref. 8). The internal forces at the origin of the E-frame can also be controlled in the position subspace by adding force vectors to $\underline{F}_i$ (see equation 2.17) such that one has

$$F = \sum_{i=1}^{n} (F_i + \underline{F}'_i)$$

or

$$\sum_{i=1}^{n} \underline{F}'_i = 0$$

This means that the total effect of adding the $\underline{F}'_i$ does not affect the motion of the multiple manipulator system while realizing the desired internal forces.

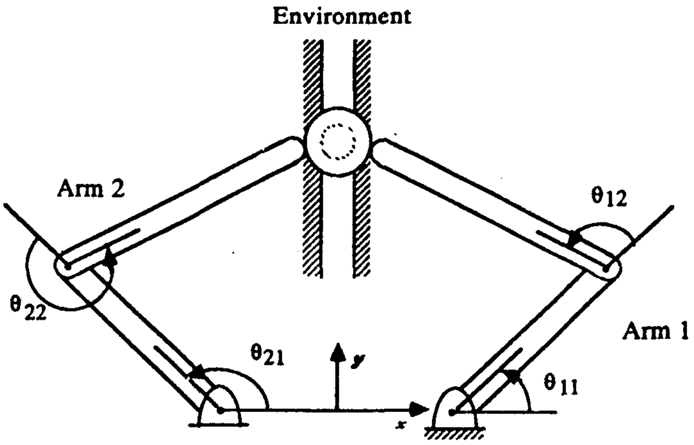


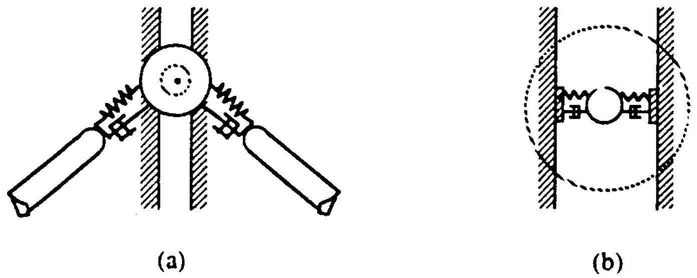
Fig. 5.1 Schematic Drawing of a Pair of Two-Link Cooperating Manipulators

5. EXAMPLE

In order to validate the proposed theory, a simulation study was undertaken. The model consisted of simple planar robotic system composed of a pair of two-link revolute manipulators. The object was assumed to a point mass attached to the upper link of each of the manipulators. nce each manipulator has only t degrees of freedom, the object was assumed to be in contact with the upper links only through interaction forc rather than through forces and torques.

Two cases were considered. In the first case, it was assumed that there were no environmental constrai This case represents a pure transport problem where two manipulators cooperate in moving an object. In t second case, an environment was assumed which restricted the motion of the object in the x direction. In bo cases, the desired motion in the y direction was obtained from a constant acceleration trajectory. The desir motion in the x direction was set equal to the initial x value (i.e., x = 0) in the first case, and the desir interaction force between the object and the environment was set equal to zero in the second case.

The simulation of closed kinematic chains can either be performed by computing the interaction forces bas on the dynamics equations (such as those developed in this paper) or by assuming the existance of stiff sprin (or spring-dashpot) at the contact points. Since in actual experiments, one uses force/torque sensors to obta these interaction forces (torques), the second approach is used in this study. Figure 5.2 illustrates t modeling of the contact mechanism between the upper links and the object and between the object and t environment. Figure 5.3 shows the overall simulation block diagram.


(a)

(b)

Figure 5.2 Detailed modeling of the connections between (a) the arms and the object, and (b) the object and the external constraint

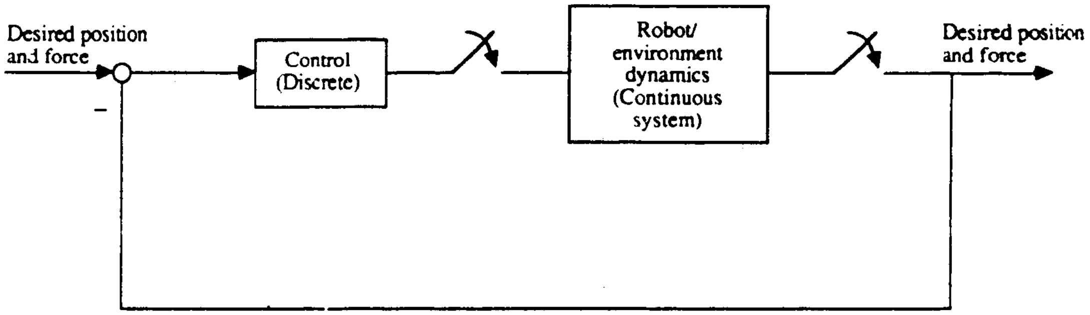

Desired position and force
-
Control (Discrete)
Robot/ environment dynamics (Continuous system)
Desired position and force

Figure 5.3 Simplified block diagram of the simulation study

Although this paper does not attempt to address the digital control aspects of the problem, the simulation stu was made more realistic by separating the continuous and discrete parts as shown in Figure 5.3.

Figures 5.4 through 5.6 show the responses of the system for the following set of parameters and control gains.

link lengths	= .04 [m]	, equal for all lengths
link masses	= 4.0 [Kg] = 2.0 [Kg]	, for lower links , for upper links
object mass	= 2.0 [Kg]	
Initial joint angles:	$\theta_{11} = 30^{\circ}$, $\theta_{21} = 150^{\circ}$,	$\theta_{12} = 120^{\circ}$ $\theta_{22} = -120^{\circ}$
Position loop gains	$k_{pp} = 4900.0$,	$k_{pd} = 98.0$
Force loop gains	$k_{fp} = 1.0$,	$k_{fd} = 0$
Spring constants	500,000 [N/m] for all of the springs	
Damping constants	140.00 [N/(m/sec)]	
Sampling period	1 msec	

Figure 5.4 shows the tracking capability of the object moved by the arms. Since the difference between the position of the object and the tip of each upper link is insignificant, the plots show the desired vs. actual position of the upper links. Figure 5.5 shows the tracking of the object in the second case (motion constrained in the x direction). Figure 5.6 shows the interaction forces between the object and the environment. The simulation study indicates that the control algorithm developed in this paper yields excellent results. It must be understood that several important practical problems such as friction, flexibility of the links and joints, link parameter errors, etc., were not included in this simulation study. Further research and study is necessary to include such effects.

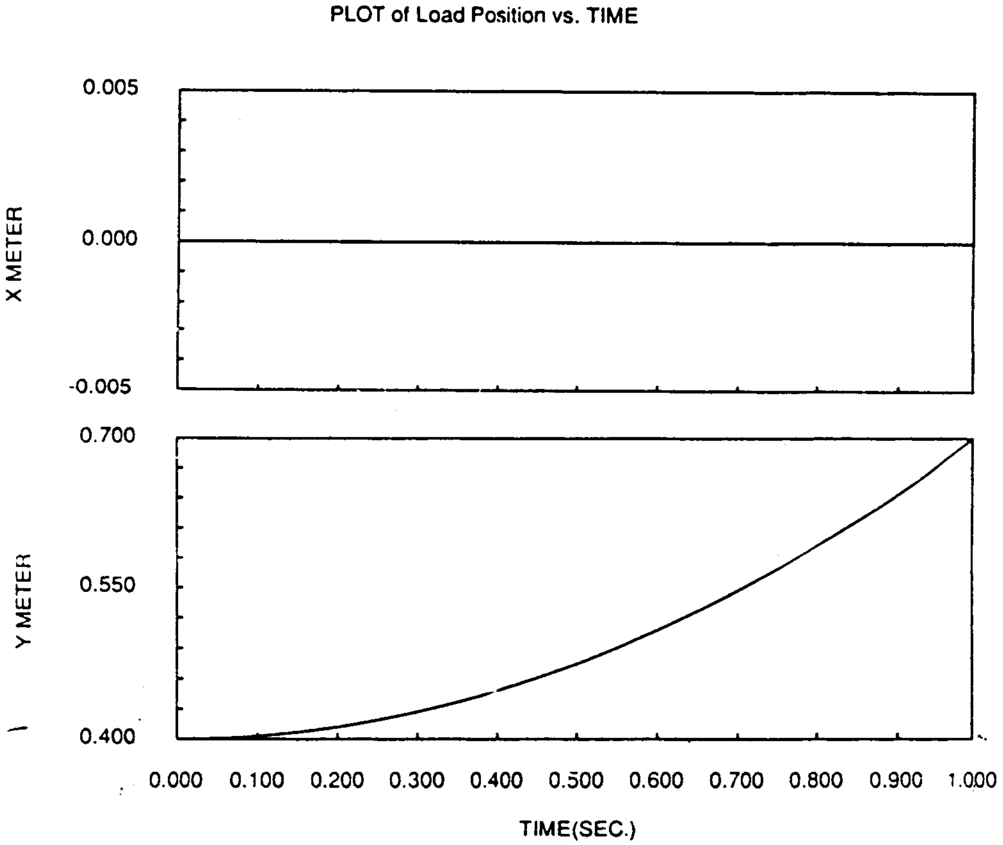


Figure 5.4 Pure Position Control with Cooperating Arms

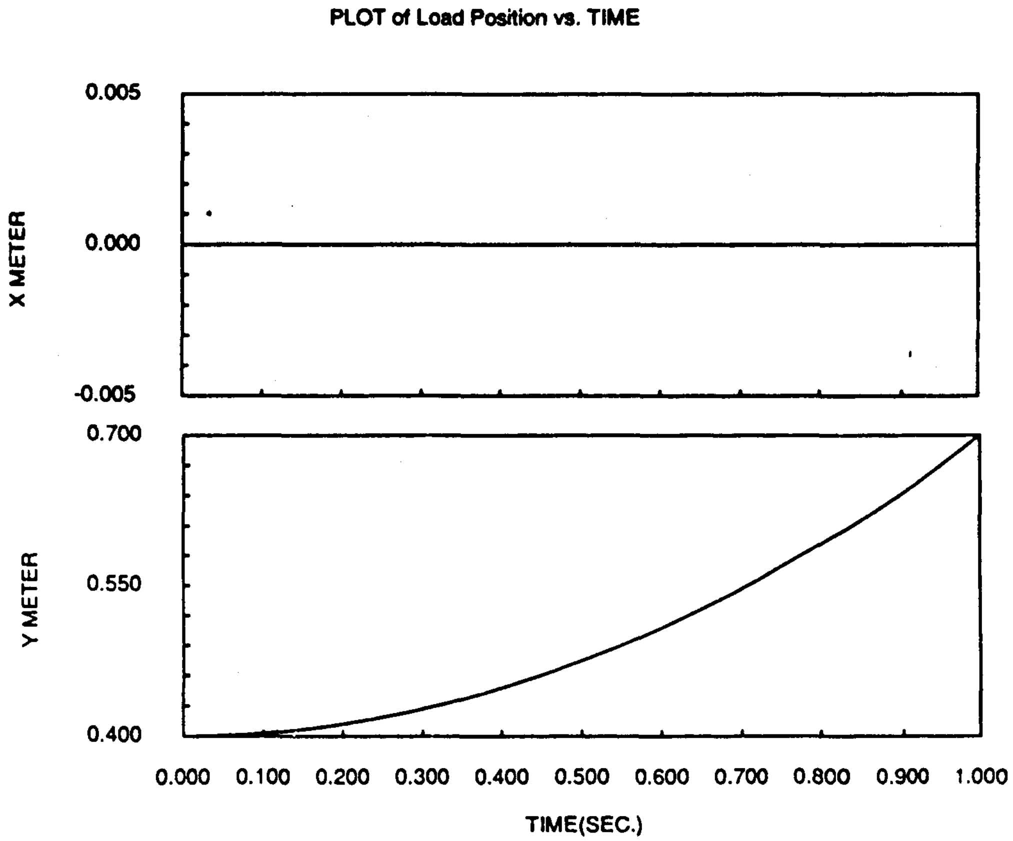


Figure 5.5 Position Tracking in Position/Force Control with Cooperating Arms

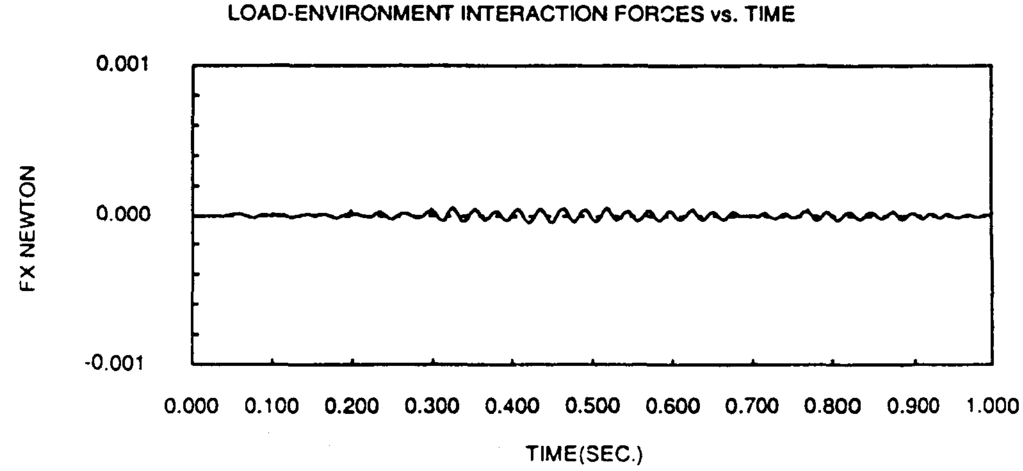


Figure 5.6 Force Tracking in Position/Force Control with Cooperating Arms

6. CONCLUSIONS

This paper presented a theory for the position and force control of multiple manipulators holding an object which is in contact with an environment. The derivation is for n manipulators each having six degrees of freedom. The control is based on the Cartesian formulation of the arm dynamics and extends the single-arm hybrid position/force control concept to the case of multiple arms. Simple but realistic simulation studies confirmed that the developed control concept results in excellent position tracking and force control.

7. ACKNOWLEDGEMENT

The research described in this document was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

8. REFERENCES

[1] Ishida, T., "Force Control in Coordination of Two Arms," Proceedings of the 5th International Conference on Artificial Intelligence, pp. 717-722, August 1977.

[2] Fujii, S. and Kurono, S., "Coordinated Computer Control of a Pair of Manipulators," Fourth World Congress on the Theory of Machines and Mechanisms, University of Newcastle upon Tyne, September 8-12, 1975.

[3] Zheng, Y.F. and Luh, J.Y.S., "Constrained Relations Between Two Coordinated Industrial Robots," Proceedings of Machine Intelligence Conference, Rochester, Michigan, April 23-24, 1985.

[4] Zheng, Y.F. and Luh, J.Y.S., "Joint Torques for Control of Two Coordinated Moving Robots," Proceedings of the 1986 International Conference on Robotics and Automation, San Francisco, California, April 7-10, 1986.

[5] Khatib, O., "The Operational Space Formulation in Robot Manipulator Control," 15th ISIR, Tokyo, Japan, September 11-13, 1985.

[6] McClamroch, N.H., "Dynamics of a Closed Chain Manipulator," Proceedings of the 1985 American Control Conference, Seattle, Washington.

[7] Hemami, H., "Modeling and Control of Constrained Dynamics Systems with Application to Biped Locomotion in the Frontal Plane," IEEE Transactions on Automatic Control, Vol. AC-24, No. 4, August 1979.

[8] Hayati, S., "Hybrid Position/Force Control of Multi-Arm Cooperating Robots, Proceedings of the IEEE Conference on Robotics and Automation, San Francisco, California, April 7-10, 1986.

[9] Bejczy, A.K., "Robot Arm Dynamics and Control," Jet Propulsion Laboratory, California Institute of Technology, TM 33-669, February, 1974.

[10] Craig, J.J., Introduction to Robotics: Mechanisms and Control, Addison Wesley Publishing Company, 1986.

A Survey of Adaptive Control Technology in Robotics

S. Tosunoglu and D. Tesar
University of Texas at Austin
Austin, TX 78712

ABSTRACT

This paper reviews the previous work on the adaptive control of robotic systems. Although the field is relatively new and does not yet represent a mature discipline, considerable attention for the design of sophisticated robot controllers has occurred. In this presentation, adaptive control methods are divided into model reference adaptive systems and self-tuning regulators with further definition of various approaches given in each class. The similarity and distinct features of the designed controllers are delineated and tabulated to enhance comparative review.

1. INTRODUCTION

Control of robotic manipulators is a challenging problem mainly due to tne nonlinear and coupled nature of the system dynamics. A considerable amount of valuable work has been produced in the dynamic formulation and the control of these systems within the last two decades. Since the pioneering works of Uicker [1], Hooker and Margulies [2], and Kahn and Roth [3] on the formulation of dynamics, researchers have concentrated on the efficient computer implementation and numerical construction of the dynamic equations. While the work on the efficient dynamic equation algorithms is still going on, control of manipulators has also received significant attention. Over the years, literature on the manipulator control methods using optimization, linearization, nonlinearity compensation, and recently, adaptive techniques has become quite rich.

This paper reviews the accumulated work in the area of adaptive control as applied to robotics. The reader should note that adaptive control in itself is not yet a mature discipline in systems .heory. Also, since some of the existing tools in adaptive control are strictly for linear and/or time-invariant systems, their application to robotics deserves special attention. The immaturity of adaptive control is best demonstrated by the lack of a definit.on of adaptive control agreed to by the leading researchers [4].

According t. Webster's dictionary, to adapt means "to adjust (oneself) to new circumstances". Adaptive control, then, in essence, is used to mean a sophisticated, flexible control system relative to the conventional fixed feedback system. An adaptive system will assure quality system performance when large and unpredictable variations in the plant dynamics or loading occur. Although our aim is by no means to establish the missing definition, since the robotics community seems to have reached a consensus on what is meant by adaptive control, we will give our definition to illustrate our interpretation of adaptive control.

<u>Definition</u>: A feedback control system is adaptive if the gains are selected with the on-line information of plant outputs and/or plant state variables.

This definition is depicted in block diagram format in Figure 1. The above definition encompasses all the previous work on the adaptive control of manipulators currently available to us.

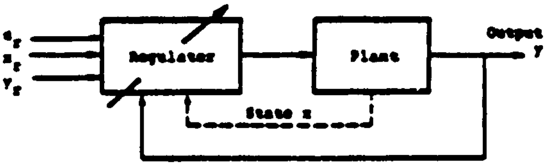


Figure 1. Block Diagram Representation of an Adaptive Control System

Although the early work on adaptive control dates back to the 1950s, its first extensive application to robotics was given by Dubowsky and DesForges in 1979 [5]. Since then a variety of different methods has been developed. So far, the existing adaptive control methods applied to robotics may be categorized under the design of

i. Model Reference Adaptive Systems (MRAS)

ii. Self-tuning Regulators (STR)

The following methods are used in the design of MRAS:

i. Local parametric optimization

ii. Lyapunov's second method

iii. Hyperstability theory

iv. Sliding control theory

The STR design procedure may be divided into three steps:

i. Selection of a parametric structure to represent the robotic system via discete-time modeling

ii. On-line estimation of system parameters using the least squares, extended least squares or maximum likelihood methods

iii. On-line controller design based on the estimated system parameters via extended minimum variance or pole-zero placement techniques

Block diagrams of MRAS and STR are illustrated in Figures 2 and 3. Note that the dotted boxes in these figures may be reduced to the regulator block in Figure 1. After a brief review of system dynamics, the related background work is presented below.

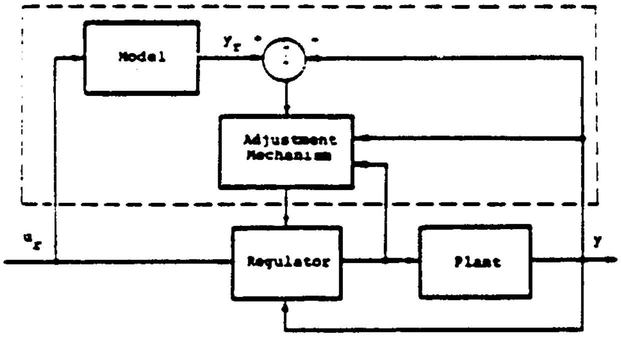


Figure 2. Block Diagram of Model Reference Adaptive System

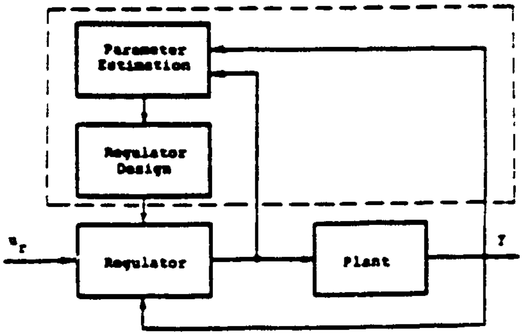


Figure 3. Block Diagram of Self-tuning Regulator

2. SYSTEM DYNAMICS

Dynamic equations of an n-link, n-degree-of-freedom, spatial, serial robot arm with rigid links are given by

$$A(\theta)\ddot{\theta} - F(\theta,\dot{\theta})\dot{\theta} - G(\theta)\theta = u \qquad (1)$$

where $\theta \epsilon R^n$ is the relative joint displacement vector ($x = [\theta^T \quad \dot{\theta}^T]^T \epsilon R^{2n}$), $A(\theta) \epsilon R^{nxn}$ is the generalized inertia matrix, $-f = -F(\theta,\dot{\theta})\dot{\theta} \epsilon R^n$ represents the inertia torques due to centrifugal and Coriollis accelerations, $-g = -G(\theta)\theta \epsilon R^n$ is the gravity loads as seen at the joints where $G(\theta) \epsilon R^{nxn}$ is non-unique, and $u \epsilon R^n$ is the control. In state-space representation, Eq. (1) can be given by

$$\dot{x} = \begin{bmatrix} 0 & I \\ A^{-1}G & A^{-1}F \end{bmatrix} x + \begin{bmatrix} 0 \\ A^{-1} \end{bmatrix} u \qquad (2)$$

Note that functional dependencies are dropped for clarity. If each actuator (D.C. motor) is modeled as a second-order, linear, time-invariant subsystem (neglecting the armature inductance), and is coupled with the manipulator dynamics, the previously defined state vector, x, will be preserved and the control will be the actuator input voltage. In this case, Eq. (2) takes the following form

$$\dot{x} = \begin{bmatrix} 0 & I \\ \Lambda^{-1}(G-E_1) & \Lambda^{-1}(F-E_2) \end{bmatrix} x + \begin{bmatrix} 0 \\ \Lambda^{-1}E_3 \end{bmatrix} u \qquad (3)$$

where $\Lambda = A + J$ is the combined inertia matrix with $J = \text{diag}[J_k]$, J_k is the rotor inertia of the k^{th} actuator referred to output shaft, E_1, E_2 and E_3 are diagonal, positive definite constant matrices and functions of various actuator/gear train parameters.

Although most of the works do not include the actuator dynamics, the above, simplified form may be substituted, since the form of the equations remains the same. Depending on the adaptation algorithm, these constant actuator parameters may either be included in the on-line identification scheme or assumed known. In our presentation, the generic u will represent the suitable control (either the effective input torques or the voltages). The only exception is [6] where third-order actuator dynamics is studied in addition to the above simplified form. The dynamic equations, Eq. (2) or (3), may be given in terms of the robot-hand coordinates expressed in a fixed reference frame (task-oriented coordinates) and adaptive controllers may be designed for this system [6,7,8].

3. MRAS-BASED CONTROLLERS

In MRAS design, usually a second-order, linear, time-invariant, continuous-time reference model is selected for each link of the serial robot. Then, a control law is derived to force the robot to behave like the selected model. As mentioned earlier, local parametric optimization [5,9], Lyapunov's second method [10], hyperstability [11,12,13], or the sliding control theory [14,15,16] is usually employed to achieve the goal.

In 1979, Dubowsky and DesForges [5] implemented the local parametric optimization technique on a robot arm. In their formulation, each servomechanism is modeled as a second-order, single-input, single-output system, neglecting the coupling between system degrees of freedom. Then for each degree-of-freedom, position and velocity feedback gains are calculated by an algorithm which minimizes a positive semi-definite error function utilizing the steepest descent method. Stability is investigated for the uncoupled, linearized system model. This work represents the first implementation of adaptive control to robotics.

The recent works have concentrated on the designs based on the Lyapunov's second method and the hyperstability theory. In the most general case, these control methods yield the following control structure u_p:

$$u_p = \delta_1 f_1(A_p, F_p, G_p, x_p) + \delta_2 f_2(K_i, x_p, x_r, u_r) + \delta_3 f_3(m_j, x_p, x_r, \dot{x}_r) \qquad (4)$$

where subscripts p and r represent the plant and reference model, respectively, δ_i is either 0 or 1, $f_i \epsilon R^n$, $K_i \epsilon R^{nxn}$ nonlinear or constant gain matrix, i=1,2, or 3, $m_j \epsilon R^1$ represents an unknown system parameter like the payload, link mass content, center of gravity location, etc., where a combination of these constant parameters or a nonlinear term is to be estimated in the adaptatión process, and j=1,2,...,k, where k depends on the specific controller design. Although some controllers call for plant joint accelerations, they are not shown in Eq. (4).

The first term f_1 in Eq. (4) describes the nonlinearity compensation. It may represent the complete manipulator dynamics as in [17,18], or only the gravity terms and the Jacobian as in [7]. A controller with $\delta_1 = 1$ and $\delta_2 = \delta_3 = 0$ indicates only a nonlinearity compensation. The second term in u_p represents the feedback portion of the controller. The gain matrices K_i may either be nonlinear or constant. Now $\delta_1 = \delta_3 = 0$ and $\delta_2 = 1$ represent the control structures of [16,19,20,21] among others. The third term in u_p includes the portion of the control where system parameters are estimated [17,18,22]. Slotine [18], for example, includes all the components into his controller, therefore, $\delta_1 = \delta_2 = \delta_3 = 1$. Takegaki and Arimoto's control strategy [7] may be summarized by $\delta_1 = \delta_2 = 1$ and $\delta_3 = 0$, Horowitz and Tomizuka's [22] $\delta_1 = 0$, $\delta_2 = \delta_3 = 1$, etc.

Various MRAS control structures are summarized in Table 1. This table differentiates the methods which require explicit calculation of dynamic equations (Nonlinearity Compensation) from the methods which adaptively estimate the plant parameters on-line (Incorporation of Plant Parameter Estimation). However, further distinction is needed in the latter group, since while one approach explicitly identifies the nonlinear terms (as in A, G, and F with reference to Eq. (2)), and estimates them on-line, the recent methods treat the constant robot parameters as unavailable, estimate and compensate them in their algorithms. Some methods choose nonlinear feedback matrices in their controllers (H, L, S in Table 1) without incorporating the explicit system parameter estimations.

The early works presented in Table 1 have generally avoided the nonlinearity compensation and opted for the assumption that the nonlinear system parameters vary slowly in time. On this basis, a stability analysis is given for the system. This assumption almost certainly is too restrictive, since the nonlinear manipulator system parameters are functions of the joint position and velocities. The faster the robot movement is, the more rapidly the system parameters will vary. The objective on the other hand, for the more sophisticated control methods is to enable fast robot movements with high precision. As a result of revolutionary advances in the microprocessor industry with prices steadily coming down, the possibility of real-time implementation of computation intensive algorithms is steadily improving. Recently, Wander and Tesar [23,24] have implemented the complete dynamic equations [25] of a 6-link, general architecture robot arm in 6.5 milli sec. (150 Hz) in explicit form without using recursion. Their algorithm is able to treat an n-degree-of-freedom (DOF) serial system of completely general parameters (43 milli sec. for 12-DOF). They have implemented the algorithm on an Analogic AP-500 array processor. Further work on the comparative analysis of various computer architectures is underway at the University of Texas at Austin.

Some of the most recent works include nonlinearity compensation along with a feedback portion and parameter identification features [6,17,18]. Once the control has the form $u_p = \tilde{A}_p(\theta_p)u_p'$, where subscript p denotes the plant, $\tilde{A}_p$ is the on-line calculated generalized inertia matrix and u_p' is yet to be selected, generally, global stability of the closed-loop system can be shown provided that $A_p^{-1}\tilde{A}_p = I$, where I is the identity matrix of order n, is maintained [6]. Otherwise, in reference to Table 1, all methods without nonlinearity compensation need to assume system parameters stay constant during the adaptation.

Table 1. Summary of MRAS Controllers in the Literature

	CONTROLLER CHARACTERISTICS			
	NONLINEARITY COMPENSATION [1]	LINEAR / NL FEEDBACK / METHOD	INCORPORATN OF PLANT PARAMETER ESTIMATION [2]	ASSUMED SYS. PARAMETERS STAY CONST. DURING ADAPTATION [3]
Dubowsky and DesForges 1979 [5]	--	O	--	√
Horowitz and Tomizuka 1980 [22]	--	C	N	√
Takegaki and Arimoto 1981 [7]	G	L	--	√
Balestrino.et al. 1983 [19]	--	H	--	√
Stoten 1983 [21]	--	H	--	√
Balestrino.et al. 1984 [16]	--	H,S	--	√
Nicosia and Tomei 1984 [27]	--	H	--	√
Vukobratovic.et al.1985 [30]	--	H	--	√
Landau 1985 [36]	--	C	N	√
Whyte 1985 [28]	--	L,C	--	√
Lim and Eslami 1985 [20]	--	L	--	√
	√	L	--	--
Hsia 1986 [29]	--	L	--	√
	--	H,C	--	√
	--	C	N	√
Craig, et al. 1986 [17]	√	--	√	--
Slotine 1986 [18]	√	L,C,S	√	--
Tosunoglu and Tesar 1986 [6]	A	L,H,C	√	--
	√	L,H,C	√	--

Key to Remarks:
1 : Calculates complete or partial nonlinear dynamics on-line.
2 : Robot link lengths, mass contents, actuator parameters, etc., if not otherwise specified.
3 : If "yes", stability analysis based on this assumption.

G : Gravity load compensated; also requires on-line Jacobian calculation.
A : Requires only the on-line calculation of the inertia matrix.
O : Nonlinear-gain feedback using local parametric optimization.
C : Constant-gain feedback.
L : Nonlinear-gain feedback using Lyapunov's second method.
H : Nonlinear-gain feedback using hyperstability theory.
S : Nonlinear-gain feedback using sliding theory.
N : Structure of nonlinear system parameters (functions of state variables) are explicitly assumed known and are adaptively estimated; stability analysis based on hyperstability theory.
√ : Yes.
– : No.

Balestrino, et al. [19] have developed an adaptive controller which produces discontinuous control signals leading to chattering. Stability analysis is presented using hyperstability theory. In [16], Balestrino, et al. present three methods; the first is based on the theory of variable-structure systems, the second on the hyperstability and the third is a combination of the first two methods. Designed controllers produce high-frequency chatter which is highly undesirable since higher order dynamic modes may be excited. Numerical simulations show an extremely high frequency of sign switches in the plant input, prohibiting its physical realization. Stoten [21] formulates the MRAS problem closely following the procedures in [9] and simulate the algorithm for a 1-link manipulator.

Horowitz and Tomizuka [22] study the adaptive control of a 3-link arm. Gravity effects and the mass and inertia of the first link are neglected. Each nonlinear term in the dynamic equations is identified a priori, treated as unknown, and estimated by an adaptation algorithm. For the modeled system and the designed controller, stability analysis is given by Popov's hyperstability theory. Later, Anex and Hubbard [26] have experimentally implemented this algorithm with some modifications. System response to high speed movements is not tested, but practical problems encountered during the implementation are addressed in detail.

Takegaki and Arimoto [7] propose an adaptive method to track desired trajectories which are described in the task-oriented coordinates. The suggested controller compensates gravity terms, calculates the Jacobian and the variable gains, but does not compensate the manipulator dynamics completely. System stability is assured if the manipulator hand velocity is sufficiently slow, i.e., nonlinear system parameters change slowly.

Nicosia and Tomei [27] derive control laws using the hyperstability theory to follow a linear, time-invariant reference model. The plant (manipulator) parameters and the payload are assumed known and are not identified. Their controller does not produce chattering and is relatively easy to implement. Lim and Eslami [20] propose controller designs based on Lyapunov's second method. The author's objective is to control the linearized dynamic equations with the developed controllers; hence, assuring the stability of the linearized system. Later, nonlinearity compensation is suggested to enhance the system response. Whyte [28] designs an adaptive controller via Lyapunov's second method. The algorithm does not require any knowledge of the manipulator dynamics and selects nonlinear gains in the feedback loop to follow the reference model. System stability is shown, provided that the parameter changes are slow. Hsia [29] reviews the current methods used in adaptive control and gives brief formulations for each method. Vukobratovic, et al. [30], review local parametric optimization and hyperstability-based methods and choose not to include the approaches based on Lyapunov's second method in their book on the non-adaptive and adaptive control of manipulators.

Tosunoglu and Tesar [6] select a generalized nonlinear reference model which represents ideal robot dynamics. The plant, the actual robot whose system parameters may not be exactly known, is then forced to behave like the reference model to follow the desired trajectory. The advantage of the nonlinear reference model is that the adaptation process ceases once the nominal trajectory is recovered. (Such is not the case when linear models are selected.) Error-driven dynamics is derived and the system is augmented to include the integral feedback feature to eliminate the parameter discrepancies between the plant and the reference model, and the disturbances acting on the system. It is shown that the controllers designed in this work via Lyapunov's second method also produce hyperstable systems. Simulations demonstrate successful trajectory tracking on 3- and 6-link, spatial manipulators under unknown payloads and estimated system parameters (link lengths, masses, inertia components, payload, etc.). The authors also provide comparative analyses of the effect of integral feedback and various controller update rates, 60 to 200 Hz.

Craig, Hsu and Sastry [17] take an interesting approach in designing an adaptive controller using the Lyapunov's second method. In this work, the structure of the terms in the dynamic equations is assumed known, but their numerical values remain unknown. They partition the dynamics into known and unknown portions and estimate the unknown parameters along with compensation for the nonlinearities. Global stability is proved by assuming that a matrix function of the plant joint position, velocity and accelerations is bounded. Although all the terms which are functions of positions are bounded, velocity and accelerations may increase without bounds; thus, making the matrix unbounded. However, modifications in the controller structure may alleviate this problem. Their numerical simulations identify link masses and Coulomb friction coefficients for a two-link manipulator with encouraging results.

Slotine and Li [18] derive a control law with full feedforward dynamics compensation, PD feedback and on-line payload and manipulator parameter estimation using Lyapunov's second method. Since this control scheme does not eliminate the steady-state errors, the authors restrict the steady-state position errors to lie on a sliding surface. This modification, in turn, causes the loss of numerical efficiency where, interestingly, the authors make use of the recursive computation feature of the manipulator dynamics. Later, an approximate implementation is suggested to improve the numerical efficiency. Payload parameter identification is simulated on a two-link manipulator.

Once these current methods are refined, application to manipulators with higher degrees of freedom will naturally follow. Determination of the structure of the constant terms (for identification) for manipulators with higher number links may be achieved with symbolic generation of dynamic equations, but the effect of increased number of terms will require further investigation.

4. SELF-TUNING REGULATOR (STR) BASED CONTROLLERS

In this method, typically, nonlinear manipulator dynamics is linearized about a nominal trajectory and then discretized. The discretized model gives the structure of the parametric model whose parameters need to be estimated on-line. The parametric model is given by the following multivariable difference equation

$$y(k) = \theta^T\phi(k-1) + e(k) \tag{5}$$

where $y(k)\epsilon R^n$ is the system output, k is the sampling time counter, $\theta\epsilon R^{nx(2nm+1)}$ contains the parameters to be identified, $\phi\epsilon R^{(2nm+1)}$ represents the combined system input and output vector, $e\epsilon R^n$ is a random, zero-mean Gaussian white noise, and m is the order of the estimation model.

Parameter estimation is based on the system identification techniques using the sampled input-output data. Although such techniques include the least squares, extended least squares and maximum likelihood methods; the recursive least squares method is extensively used because of its lower computational requirements [8,25,29-36]. The recursive least squares estimation is given by

$$\hat{\theta}_i(k) = \hat{\theta}_i(k-1) + P(k)\phi(k-1)[y_i(k) - \hat{\theta}_i^T(k-1)\phi(k-1)] \tag{6}$$

where

$$P(k) = \frac{1}{\lambda}\left[P(k-1) - \frac{P(k-1)\phi(k-1)\phi^T(k-1)P(k-1)}{\lambda + \phi^T(k-1)P(k-1)\phi(k-1)} \right]$$

$\hat{\theta}_i(k)$ represents the estimate of the i^{th} row of θ defined in Eq. (5), P(k) is a square symmetric matrix of order (2nm+1), and $0 < \lambda < 1$ is an exponential forgetting factor.

Once the parameters are identified at each sampling time, regulator parameters are estimated using the extended minimum variance [8,30,32], or pole-zero placement techniques [29,33,35]. The method described above is known as explicit or indirect STR. If the regulator parameters are estimated directly by a reparameterization of the process model, the model is called implicit or direct STR. Usually implicit STR algorithms cancel all process zeros making them suitable only for minimum phase systems.

Koivo and Guo [32] assume an autoregressive model and identify system parameters using the recursive least squares technique. They design an extended minimum variance controller for the estimated model. The method chooses a quadratic performance index in terms of the state error vector and the system control vector and minimizes it relative to admissible controls while satisfying Eq. (5). Their simulations include decoupled and partially coupled parametric model structures. They report that the parameter convergence is faster in the decoupled case, and that no significant improvement in the system response is observed for the coupled model. This is rather interesting, because the amount of on-line calculations is considerably reduced for the decoupled case. Also reported is the fact that the model and the controller parameter identifications may not converge fast enough while the robot motion takes place.

Lee [8] derives the perturbation equations, discretizes them and estimates the system parameters using the recursive least squares method. Then an adaptive controller is designed using the extended minimum variance technique. The parameter identification requires the estimation of $6n^3$ parameters on-line (216 for a 6-link manipulator). Lee provides a detailed breakdown of the computational requirements and concludes that for a 6-link manipulator the control scheme can be updated at about 56 Hz on a PDP-11/45.

Hsia [29] reviews the STR formulation on a decoupled model. Karnik and Sinha [35] develop an STR based on a non-minimum phase model which assigns the system poles while retaining all the zeros. This algorithm is developed for a UNIMATE-2000 robot. Landau [9,36] and Vukobratovic, et al. [30] review various STR designs in detail.

In general, discrete-time formulation is especially suitable for computer control. However, on-line estimation of all system parameters and the control design make STR computationally intensive. Astrom [4] reports that numerical estimation techniques tend to be numerically unstable as the number of parameters increases in the system model. In this case, the complete system is parameterized. However, the papers reviewed in this work do not raise the question with regard to numerical instability although they do indicate the importance of initial parameter selections. In STR methods, convergence of estimated parameters in the adaptation process is guaranteed if the system parameters are constant. Since the actual robot model parameters are nonlinear functions of the state vector, the question of system parameters varying slowly in time again arises in the STR methods.

5. CONCLUSIONS

Adaptive control of robotic systems has received significant attention within the past few years. A class of control laws based on the MRAS design realize the adaptation through signal synthesis with a completely known parameter structure, while some methods select a subclass of the parameters for identification for reduced computational burden. All adaptive controllers via STR design and some MRAS-based methods estimate the complete (nonlinear) system parameters on-line.

Stability analysis usually relies on the condition that the nonlinear system parameters vary slowly. This condition is removed if a nonlinearity compensation component is also incorporated i the controller. The most recent works, which exploit the special structure of manipulator dynamics, seem to favor this feature. The use of state-of-the-art microprocessor technology along with the sophisticated dynamic formulation algorithms strongly indicate that real-time implementation of dynamic compensation is rapidly becoming feasible.

Further research to perfect the existing algorithms and to provide rigorous stability proofs, which will improve the maturity of the adaptive control, is still needed. Although today's industrial robots employ linear controllers to accomplish a number of useful tasks, fast and precise robot movements remain to be implemented. Development of laboratory test beds and implementation of the developed adaptive controllers on robotic manipulators are also crucial, since only after successful demonstrations will technology transfer be possible.

6. ACKNOWLEDGEMENTS

This work is partially funded by the U.S. Department of Energy (Grant No. DE-FG02-86NE37966) and the NASA Johnson Space Center (Grant No. NAG 9-188). The authors gratefully acknowledge the funding agencies.

REFERENCES

1. Uicker, J. J., "On the Dynamic Analysis of Spatial Linkages Using 4 by 4 Matrices," PhD Dissertation, Dept. Mechanical Engineering and Astronautical Sciences, Northwestern University, Massachusetts, 1965.

2. Hooker, W. W., and Margulies, G., "The Dynamical Attitude Equations for an N-Body Satellite," J. Astronautical Sciences, Vol. 12, pp. 123-128, 1965.

3. Kahn, M. E., and Roth, B., "The Near-Minimum Time Control of Open-Loop Articulated Kinematic Chains," Stanford Artificial Intelligence Memo No. 106, December 1969.

4. Astrom, K. J., "Theory and Applications of Adaptive Control--A Survey," Automatica, Vol. 19, pp. 471-486, 1983.

5. Dubowsky, S., and DesForges, D. T., "The Application of Model-Referenced Adaptive Control to Robotic Manipulators," ASME J. Dynamic Systems, Measurement and Control, Vol. 101, pp. 193-200, September 1979.

6. Tosunoglu, S., "Adaptive Control of Robotic Manipulators," PhD Dissertation, Dept. Mechanical Engineering, University of Florida, Gainesville, Florida, May 1986.

7. Takegaki, M., and Arimoto, S., "An Adaptive Trajectory Control of Manipulators," Int. J. Control, Vol. 34, pp. 219-230, 1981.

8. Lee, S.C.G., and Lee, B. H., "Resolved Motion Adaptive Control for Mechanical Manipulators," ASME J. Dynamic Systems, Measurement and Control, Vol. 106, pp. 134-142, June 1984.

9. Landau, Y. D., Adaptive Control: The Model Reference Approach, Marcel Dekker. Inc., New York, 1979.

10. Kalman, R. E., and Bertram, J. E., "Control System Analysis and Design Via the Second Method of Lyapunov," ASME J. Basic Engineering, Series D, Vol. 8(2), pp. 371-393, June 1960.

11. Popov, V. M., "The Solution of a New Stability Problem for Controlled Systems," Automation and Remote Control, Vol. 24, pp. 1-23, January 1963.

12. Landau, I. D., "A Hyperstability Criterion for Model Reference Adaptive Control Systems," IEEE T. Automatic Control, Vol. 14, pp. 552-555, October 1969.

13. Landau, I. D., "A Generalization of the Hyperstability Conditions for Model Reference Adaptive Systems," IEEE T. Automatic Control, Vol. 17, pp. 246-247, April 1972.

14. Slotine, J.-J.E., "On Modeling and Adaptation in Robot Control," Proc. IEEE Conf. on Robotics and Automation, pp. 1387-1392, 1986.

15. Slotine, J.-J.E., "The Robust Control of Robot Manipulators," The Int. J. of Robotics Research, Vol. 4, pp. 49-64, 1985.

16. Balestrino, A., De Maria, G., and Zinober, A.S.I., "Nonlinear Adaptive Model-Following Control," Automatica, Vol. 20, pp. 559-568, 1984.

17. Craig, J. J., Hsu, P., and Sastry, S. S., "Adaptive Control of Mechanical Manipulators," Proc. IEEE Conf. on Robotics and Automation, pp. 190-195, 1986.

18. Slotine, J.-J.E., and Li, W., "On the Adaptive Control of Robot Manipulators," Robotics: Theory and Applications, Winter Annual Meeting of ASME, DSC-Vol. 3, pp. 51-56, December 1986.

19. Balestrino, A., De Maria, G., and Sciavicco, L., "An Adaptive Model Following Control for Robotic Manipulators," ASME J. Dynamic Systems, Measurement and Control, Vol. 105, pp. 143-151, September 1983.

20. Lim, K. Y., and Eslami, M., "New Controller Designs for Robot Manipulator Systems," Proc. American Control Conf., pp. 38-43, 1985.

21. Stoten, D. P., "The Adaptive Control of Manipulator Arms," Mechanism and Machine Theory, Vol. 18, pp. 283-288, 1983.

22. Horowitz, R., and Tomizuka, M., "An Adaptive Control Scheme for Mechanical Manipulators--Compensation of Nonlinearity and Decoupling Control," ASME Paper No. 80-WA/DSC-6, 1980.

23. Wander, J. P., "Real-Time Computation of Influence-Coefficient Based Dynamic Modeling Matrices for Improved Manipulator Control," MS Thesis, Dept. Mechanical Engineering, University of Florida, Gainesville, Florida, 1985.

24. Wander, J. P., and Tesar, D., "Pipelined Computation of Manipulator Modeling Matrices," IEEE J. Robotics and Automation, accepted for publication.

25. Thomas, M., and Tesar, D., "Dynamic Modeling of Serial Manipulator Arms," ASME J. Dynamic Systems, Measurement and Control, Vol. 104, pp. 218-228, September 1982.

26. Anex, R. P., and Hubbard, M., "Modeling and Adaptive Control of a Mechanical Manipulator," ASME J. Dynamic Systems, Measurement and Control, Vol. 106, pp. 211-217, September 1984.

27. Nicosia, S., and Tomei, P., "Model Reference Adaptive Control Algorithms for Industrial Robots," Automatica, Vol. 20, pp. 635-644, 1984.

28. Whyte, H. D., "Practical Adaptive Control of Actuated Spatial Mechanisms," Proc. IEEE Conf. on Robotics and Automation, pp. 650-655, 1985.

29. Hsia, T. C., "Adaptive Control of Robot Manipulators--A Review," Proc. IEEE Conf. on Robotics and Automation, pp. 183-189, 1986.

30. Vukobratovic, M., Stokic, D., and Kircanski, N., Non-Adaptive and Adaptive Control of Manipulation Robots, Springer-Verlag, Berlin, 1985.

31. Wittenmark, B., and Astrom, K. J., "Practical Issues in the Implementation of Self-tuning Control," Automatica, Vol. 20, pp. 595-605, 1984.

32. Koivo, A. J., and Guo, T. H., "Adaptive Linear Controller for Robotic Manipulators," IEEE T. Automatic Control, Vol. 28, pp. 162-171, February 1983.

33. Walters, R. and Bayoumi, M., "Application of a Self-tuning Pole-placement Regulator to an Industrial Manipulator," IEEE Conf. on Decision and Control, pp. 323-329, 1982.

34. Lee, C.S.G., Chung, M. J., and Lee, B. H., "An Approach of Adaptive Control for Robot Manipulators," J. Robotic Systems, Vol. 1, pp. 27-37, 1984.

35. Karnik, A. M., and Sinha, N. K., "Adaptive Control of an Industrial Robot," Robotica, Vol. 4, pp. 243-246, 1986.

36. Landau, I. D., "Adaptive Control Techniques for Robotic Manipulators--The Status of the Art," Preprints 1st IFAC Symp. on Robot Control, pp. 11-19, 1985.

S21-63
150
133190

Simple Robust Control Laws for Robot Manipulators, Part I: Non-adaptive Case

J.T. Wen and D.S. Bayard
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

J J 5 [illegible]

Abstract

A new class of exponentially stabilizing control laws for joint level control of robot arms is introduced. It has been recently recognized that the nonlinear dynamics associated with robotic manipulators have certain inherent passivity properties. More specifically, the derivation of the robotic dynamic equations from the Hamilton's principle gives rise to natural Lyapunov functions for control design based on total energy considerations. Through a slight modification of the energy Lyapunov function and the use of a convenient lemma to handle third order terms in the Lyapunov function derivatives, closed loop exponential stability for both the set point and tracking control problem is demonstrated. The exponential convergence property also leads to robustness with respect to frictions, bounded modeling errors and instrument noise. In one new design, the nonlinear terms are decoupled from real-time measurements which completely removes the requirement for on-line computation of nonlinear terms in the controller implementation. In general, the new class of control laws offers alternatives to the more conventional computed torque method, providing trade offs between robustness, computation and convergence properties. Furthermore, these control laws have the unique feature that they can be adapted in a very simple fashion to achieve asymptotically stable adaptive control.

1. Introduction

The problem of joint level control for the multi-link rigid articulated robot arm is addressed in this paper. Accurate measurements of the joint variables, either angular or displacement, and the joint velocities are assumed available. Traditionally, this problem has been treated by the PID algorithm. Since the justification of using PID control is based on either linearization or some local stability argument [1], its application is limited to small angle maneuvers. Large excursions usually require partitioning a desired trajectory into intermediate points and PID control is used to drive the arm between adjacent points. This approach is less than satisfactory since global stability and adequate performance are not guaranteed. This then motivates the computed torque method [2] which compensates for nonlinear terms in the robot dynamics. Assuming that the robot dynamics are known exactly, the compensated system appears like a decoupled system of double integrators and the closed loop dynamics can be shaped into desirable forms.

A different approach has been advanced in the past few years. It is based on the recognition that robot arms belong to the class of natural systems, which means time invariant, unconstrained and lying in a conservative force field [3]. It is natural to investigate if the structure specific to this class of systems can be exploited in controller design. It has long been known [3,4 and earlier] that negative proportional (generalized position) and derivative (generalized velocity), or equivalently PD, feedback globally asymptotically stabilized natural systems. The stability analysis is based on a Lypaunov function motivated by total energy considerations. Application of this result to robot arms has been relatively recent. In particular, global asymptotic stability under joint level PD feedback with gravity compensation has been shown [5-8]. Application to the tracking problem is more difficult due to the time varying nature of the problem. More specifically, the stability analysis requires a generalization of the invariance principle to time-varying systems; this issue has been partially addressed in [9-10].

In this paper, we will introduce a new class of exponentially stabilizing control laws for both the set point and tracking control problems. The stability proof is achieved by making use of a particular class of energy-like Lyapunov functions in conjunction with a useful lemma for addressing the higher order terms in the Lyapunov function derivatives. In the set point control case, Lyapunov functions based on various artificial potential fields are used to derive control laws possessing desired properties. These include set point controllers having simple PD or PD+bias structures and the ability to handle joint stop constraints. In the tracking control case, this new class of Lyapunov functions avoids the need for a generalized invariance principle, which, as mentioned above, has been the major source of difficulty in existing approaches. This leads to a new class of exponentially stabilizing tracking control laws. In one design among this new class, the nonlinear terms are decoupled from real-time measurements which completely removes the requirement for on-line computation on nonlinear terms in the controller implementation. This result is believed to have no counterpart in the present day literature. In general, the new class of control laws offers alternatives to the more conventional computed torque method, providing tradeoffs between robustness, computation and convergence properties. Furthermore, these control laws have the unique property that they can be adapted in a very simple fashion to achieve asymptotically stable adaptive control. This last property will be elaborated on in the companion paper [13]. The closed loop exponential stability also allows the robustness property to be established with respect to viscous and Coulomb friction, bounded modeling error and instrument noise.

This paper is organized as follows: Some background derivations, identities, notations, lemmas and releva results in the literature are covered in Sec. 2. Several useful set point controllers based on different artificial potential energies are presented in Sec. 3. A new Lyapunov function is also introduced to demonstrate exponential convergence. In Sec. 4, a new family of exponentially stabilizing tracking control laws are derive We will discuss the trade off between the ease of implementation and the strength of assumptions for these controllers. Their robustness properties are also analyzed in this section. Finally, conclusions are drawn in Sec. 5 together with a table summarizing all of the controllers presented in this paper and the conditions for stability. Due to the size limitation of this paper, the detail derivations are given in [2] .

2. Background

2.1 Robot Dynamic Equation

In this section, the dynamic equation of robot manipulator is derived. At the first glance, it appears as a complex, tightly coupled set of nonlinear equations. However, based on derivation from Hamiltonian principle, the nonlinearity actually contains a great deal of structure. As a result, some important identities are developed in the next section on which the rest of the paper is based.

An n-link rigid robot arm belongs to the class of so-called natural systems with the kinetic and potential energies given by

$$T = \frac{1}{2} q_2^T M(q_1) q_2$$

$$U = - q_2^T u + g(q_1) \qquad (2.1)$$

where

T = kinetic energy , U = potential energy , q_1 = joint angle or position vector ϵR^n ,

q_2 = joint velocity vector ϵR^n , $M(q_1)$ = mass inertia matrix ϵR^{nxn} , $g(q_1)$ = gravitational potential energy ,

u = joint torque force vector ϵR^n

Note that since all the analysis is done at joint level, the arm can be redundant (more than 6 joints). To derive the differential form of the robot dynamics, first set up the Lagrangian

$$L = T - U$$

Then apply the Lagrange's Equaticn

$$\frac{d}{dt}\left(\frac{\partial L}{\partial q_2}\right) - \frac{\partial L}{\partial q_1} = 0$$

This gives the dynamic equation of robot motion:

$$\dot{q}_1 = q_2$$

$$M(q_1)\dot{q}_2 = - C(q_1,q_2)q_2 - k(q_1) + u \qquad (2.2)$$

where

$$C(q_1,q_2) = \sum_{i=1}^{n} [(e_i\ q_2^T M_i(q_1))^T - \frac{1}{2} (e_i\ q_2^T M_i(q_1))] \qquad (2.3)$$

e_i = ith unit vector

$$M_i(q_1) = \frac{\partial M(q_1)}{\partial q_{1i}}$$

$$k(q_1) = \frac{\partial g(q_1)}{\partial q_{1i}}$$

q_{1i} = ith component of q_1

The term $C(q_1,q_2)q_2$ represents the coriolis and centrifugal forces and $k(q_1)$ represents the gravity load. Note that $C(q_1, q_2)$ is determined entirely from the mass-inertia matrix. Many desirable properties, for example, inherent passivity, well-posedness of solution (no finite escape under any bounded control), existence of solution to optimal control problem [14] etc. are the consequence of this additional structure. Other important properties of (2.2) include that $M(q_1)$ and $M_i(q_1)$ are symmetric and $M(q_1)$ is positive definite, for all $q_1 \epsilon R^n$. For later use, the matrices $C(q_1,q_2)$ and $M_i(q_1)$ are interpreted as R^{nxn} valued function of two n-vectors (q_1 and q_2) and one n-vector (q_1), respectively.

2.2 Some Useful Identities

Some key identities that will be used throughout this paper and the companion paper [13] are derived in this section. First define some notations:

$$M_D(q_1,Z) = \sum_{i=1}^{n} M_i(q_1) z\, e_i^T \tag{2.4}$$

$$\dot{M}(q_1,q_2) = \frac{d}{dt} M(q_1) \tag{2.5}$$

$$J(q_1,z) = \sum_{i=1}^{n} [(e_i\, z^T M_i(q_1)) - (e_i\, z^T M_i(q_1))^T] \tag{2.6}$$

$$r(q_1,q_2,q_{2d}) = (q_2-q_{2d})^T [\tfrac{1}{2} \dot{M}(q_1,q_2)(q_2-q_{2d}) - C(q_1,q_2)q_2] \tag{2.7}$$

q_{1d}, q_{2d} = desired joint position and joint velocities

$$\begin{aligned} \Delta q_1 &= q_1 - q_{1d} \\ \Delta q_2 &= q_2 - q_{2d} \end{aligned} \tag{2.8}$$

Again, M_D and J are regarded as R^{nxn} valued function of two n-vector arguments. Note that J is a skew symmetric matrix, i.e., $J+J^T = 0$.

Identity 1

$$\dot{M}(q_1,q_2)z = M_D(q_1,z)q_2 \tag{2.9}$$

Identity 2

$$C(q_1,z)z = \frac{1}{2} (M_D(q_1,z) - J(q_1,z))z \tag{2.10}$$

Identity 3

$$J(q_1,z) = M_D^T(q_1,z) - M_D(q_1,z) \tag{2.11}$$

Identity 4

$$M_D^T(q_1,x)y = M_D^T(q_1,y)x \tag{2.12}$$

Identity 5

$$r(q_1,q_2,q_{2d}) = \frac{1}{2}\ \Delta q_2^T(J(q_1,q_2)q_{2d} - M_D(q_1,q_{2d})q_2) \tag{2.13}$$

$$= \frac{1}{2}\ \Delta q_2^T(J(q_1,q_{2d})q_2 - M_D(q_1,q_2)q_{2d}) \tag{2.14}$$

$$= \frac{1}{2}\ \Delta q_2^T(J(q_1,q_{2d})q_{2d} - M_D(q_1,q_2)q_{2d}) \tag{2.15}$$

$$= \frac{1}{2}\ \Delta q_2^T(J(q_1,q_2)q_2 - M_D(q_1,q_{2d})q_2) \tag{2.16}$$

Identity 6

$$M_D(q_1,\Delta q_2)q_2 - C(q_1,q_2)q_2 - \frac{1}{2} (J(q_1,q_2)q_{2d} - M_D(q_1,q_{2d})q_2)$$

$$= M_D(q_1,\Delta q_2)q_2 - C(q_1,q_2)q_2 - \frac{1}{2} (J(q_1,q_{2d})q_2 - M_D(q_1,q_2)q_{2d})$$

$$= \frac{1}{2} (M_D^T(q_1,\Delta q_2)\Delta q_2 + M_D^T(q_1,q_{2d})\Delta q_2 - M_D(q_1,q_{2d})\Delta q_2 + M_D(q_1,\Delta q_2)q_{2d}) \tag{2.17}$$

$$M_D(q_1,\Delta q_2)q_2 - C(q_1,q_2)q_2 - \frac{1}{2} (J(q_1,q_{2d})q_{2d} - M_D(q_1,q_2)q_{2d})$$

$$= (M_D^T(q_1,q_{2d}) - M_D(q_1,q_{2d}))\Delta q_2 + \frac{1}{2} M_D^T(q_1,\Delta q_2)\Delta q_2 + \frac{1}{2} M_D(q_1,\Delta q_2)q_{2d} \tag{2.18}$$

$$M_D(q_1,\Delta q_2)q_2 - C(q_1,q_2)q_2 - \frac{1}{2} (J(q_1,q_2)q_2 - M_D(q_1,q_{2d})q_2)$$

$$= \frac{1}{2} (M_D(q_1,\Delta q_2)\Delta q_2 + M_D(q_1,\Delta q_2)q_{2d}) \tag{2.19}$$

2.3 Important Lemmas

In this section, two important stability lemmas are presented that will play pivotal roles in later sections. The first lemma is essentially a local stability theorem that establishes a region of convergence. The second lemma generalizes the first result to the Lagrange stability case. The proofs of these lemmas are given in [23].

Lemma 2-1

Given a dynamical system

$$\dot{x}_i = f_i(x_1,\dots,x_N,\ t) \quad , \quad x_i \in R^{n_i} \ ; \quad t \geq 0$$

Let f_i's be locally Lipschitz with respect to $x_1,\dots,x_N$ uniformly in t on bounded intervals and continuous in t for $t \geq 0$.

Suppose a function $V: R^{n_1 \times \dots \times n_N} \times R_+ \to R_+$ is given such that

$$V(x_1,\dots,x_N,\ t) = \sum_{i,j=1}^{N} x_i^T P_{ij}(x_1,\dots,x_j,\ t)x_j \ ,$$

V is positive definite in $x_1,\dots,x_N$

$$\dot{V}(x_1,\dots,x_N,\ t) \leq -\sum_{i \in I_1} (\alpha_i - \sum_{j \in I_{2_i}} \gamma_{ij}||x_j(t)||^{k_{ij}})||x_i(t)||^2 \tag{2.20}$$

where $\alpha_i,\ \gamma_{ij},\ k_{ij} > 0,\ \ I_{2_i} \subset I_1 \subset \{1,\dots,N\}$

Let $\xi_i > 0$ be such

$$\xi_i||x_i||^2 \leq V(x_1,\dots,x_N,\ t) \tag{2.21}$$

Let $V_o \overset{\Delta}{=} V(x_1(0),\dots,x_N(0),\ 0)$

If $\forall\ i \in I_1$,

$$\alpha_i > \sum_{j \in I_{2_i}} \gamma_{ij} \left(\frac{V_o}{\xi_j}\right)^{\frac{k_{ij}}{2}} \tag{2.22}$$

then $\forall\ \lambda_i \in (0,\ \alpha_i - \sum_{j \in I_{2_i}} \gamma_{ij} \left(\frac{V_o}{\xi_j}\right)^{\frac{k_{ij}}{2}})$,

$$\dot{V}\ (x_1,\dots,x_N,\ t) \leq -\sum_{i \in I_1} \lambda_i\ ||x_i||^2 \qquad \forall\ t \geq 0 \tag{2.23}$$

In the above lemma, we choose to bound over $\sum_{j \in I_{2_i}} \gamma_{ij}\ ||x_j(t)||^{k_{ij}}$

(Condition (2.20) reflects that choice). This choice is arbitrary; in fact, we can extract any quadratic cross term from

$$\sum_{j \in I_{2_i}} \gamma_{ij}\ ||x_j(t)||^{k_{ij}}\ ||x_i(t)||^2$$

and overbound the rest. After completing square stability condition similar to (2.22) can be stated. We do not pursue this generalization here.

Lemma 2-2

Suppose in Lemma 2-1,

$$\dot{V}(x_1,\dots,x_N,t) \leq -\sum_{i \in I_1} (\alpha_i - \sum_{j \in I_{2_i}} \gamma_{ij}\ ||x_j||^{k_{ij}})\ ||x_i||^2 + \rho \tag{2.24}$$

$$I_1 = \{1,\dots,N\}$$

Let $p = \sup_{x_i \in R^{n_i},\, t \in R+} ||P(x_1,\dots,x_N,t)|| < \infty\ ,\ [P]_{ij} = P_{ij}$

If $\forall\ i \in I_1$

$$\alpha_i > \frac{\rho p}{V_o} + \sum_{j \in I_{2_i}} \gamma_{ij} \left(\frac{V_o}{\xi_j}\right)^{\frac{k_{ij}}{2}} \tag{2.25}$$

then

$$\limsup_{t \to \infty} V\ (x_1(t),\dots,x_N(t),t) \leq \frac{\rho}{\lambda} \tag{2.26}$$

$$\text{where}\quad \lambda = \frac{1}{p} \min_{i \in I_1} (\alpha_i - \sum_{j \in I_{2_i}} \gamma_{ij} \left(\frac{V_o}{\xi_j}\right)^{\frac{k_{ij}}{2}}) \tag{2.27}$$

Furthermore, the convergence to the set $\{(x_1,\dots,x_N): ||x_k||^2 \leq \frac{\rho}{\xi_k \lambda}\ ,\ k \in I_1\}$ is exponential with rate λ.

The utility of this result is mainly in robustness analysis. Basically, given a bounded set of possible initial conditions, excluding a neighborhood of the origin, α_i's must be large enough in the sense of (2.25) for the trajectories to be uniformly bounded. What if $V_o = 0$? We can shift $t = 0$ to some finite time later when $V_t \neq 0$. In practice, a neighborhood around origin can usually be excluded since some robust locally stable control algorithm such as PID takes over.

2.4 Recent Results

Some of the recent results related to Lyapunov analysis of robot systems are reviewed in this section. For the set point control problem, [5-10] has applied the result that linear negative feedback of generalized position and velocities globally asymptotically stabilizes a natural system to robot manipulators. We will restate this result, mention work for the tracking problem in [9,10,18] and state some open issues that will be addressed in the remainder of this paper and in [13].

<u>Theorem 2-1</u>

Consider (2.2) with the control law

$$u = - K_p \ \Delta q_1 - K_v q_2 + k(q_1) \quad , \quad K_p > 0, \ K_v > 0 \tag{2.28}$$

The null state of $(\Delta q_1, q_2)$ - system is a globally asymptotically stable equilibrium.

The main idea of this approach is to shape the potential field in such a way that it is globally convex and attains a global minimum at $\Delta q_1 = 0$. Complete damping (in the terminology of [3]) is added through the derivative feedback to drive the system to the minimum potential energy state which by design is the desired state. To be specific, suppose the desired potential field is $U^*(\Delta q_1)$. The total energy under this potential is

$$V = T + U^* \tag{2.29}$$

Rewrite V as

$$V = T + U^o + U^* - U^o = V^o + U^* - U^o$$

where U^o is the original potential energy <u>without</u> external force fields, and V^o is the corresponding total energy. Let $p = M(q_1)q_2$ be the generalized momentum. From Hamilton's equation,

$$\begin{aligned} \dot{V} &= (\frac{\partial V^o}{\partial p})^T u + q_2{}^T(\frac{\partial U^*}{\partial \Delta q_1} - \frac{\partial U^o}{\partial \Delta q_1}) \\ &= q_2{}^T(u + \frac{\partial U^*}{\partial \Delta q_1} - \frac{\partial U^o}{\partial \Delta q_1}) \end{aligned} \tag{2.30}$$

Hence, to drive the desired total energy to its minimum state, we can select ([5])

$$u = - K_v q_2 - \frac{\partial U^*}{\partial \Delta q_1} + \frac{\partial U^o}{\partial \Delta q_1} \tag{2.31}$$

Then $\dot{V} = - q_2{}^T K_v q_2$. From the fact that $-2\dot{q}_2{}^T K_v q_2$ is uniformly bounded ($\dot{V} \leq 0$), $q_2(t) \to 0$ as $t \to \infty$ [19].

$$\begin{aligned} \dot{p} &= - \frac{\partial V^o}{\partial q_1} + u \\ &= - \frac{\partial T}{\partial q_1} - \frac{\partial U^o}{\partial \Delta q_1} + u \\ &= - \frac{\partial T}{\partial q_1} - K_v q_2 - \frac{\partial U^*}{\partial \Delta q_1} \end{aligned} \tag{2.32}$$

Since $\frac{\partial T}{\partial q_1} \to 0$, $\frac{\partial U^*}{\partial \Delta q_1} \to 0$, also. Hence, if $U^*(\Delta q_1)$ is globally convex with minimum at $\Delta q_1 = 0$, $\Delta q_1(t) \to 0$ as $t \to \infty$. If $U^*(\Delta q_1) = \frac{1}{2} \Delta q_1{}^T k_p \Delta q_1$, Theorem 2-1 is immediately obtained.

Obviously, any other potential field convex in Δq_1 that has global minimum at $\Delta q_1 = 0$ (and no local minima) can be used. We will use this idea in the next section to address the joint stop issue.

This control law is very appealing in its simplicity and obvious robustness with respect to modeling error in mass matrix, and centrifugal and coriolis terms. Generalization to the tracking problem is partially addressed in [9, 10]. A control algorithm is given in [9], but it lacks stability analysis. In [10], Metrosov's Theorem [11] is used. A question remains on the necessity and applicability of the Metrosov's Theorem to the tracking problem. One version of the tracking control law in Section 4 is the same as in [10], but the stability issue is resolved more completely. Nonadaptive version of the tracking control laws in [10, 18] do yield global asymptotic stability. However, the simple structure of (2.28) is lost; even for set point control, full model information is needed.

Based on the above very brief review of the currently available pertinent results, it is evident that the following issues remain open:

1. Can we get away with no gravity information, thus achieving a "universal" (arm independent) set point control law?
2. Computed torque achieves exponential stability. Are schemes based on energy Lyapunov functions inherently inferior (e.g., only asymptotic stability is possible) or have we not been clever enough in choosing the Lyapunov functions?
3. The tracking problem produces a time-varying system. Can the Invariance Principle still be applied?
4. How far can we reduce the on-line computation requirement (thus allow increasing performance) for both set point and tracking problems? What is the price to be paid?
5. Can we ensure any robustness properties with respect to friction, instrument noise, modeling errors?
6. How does one incorporate joint stop constraints?
7. Would these schemes (set point and tracking controls) still work if unknown parameters are adapted?

The rest of this paper will be devoted to answering issues 1-6. The last item is addressed in [13] and [24].

2.5 Computed Torque from Lyapunov Perspective

In Section 2.4, we introduced the total energy Lyapunov function (2.29) to derive a simple set point control law. The computed torque method can also be motivated in the same manner with a different Lyapunov function. For generality, we will consider the tracking case. Let

$$V(\Delta q_1, \Delta q_2) = \frac{1}{2}||\Delta q_2||^2 + \frac{1}{2}\Delta q_1^T K_p \Delta q_1 \qquad (2.33)$$

Calculate derivative along solution,

$$\dot{V} = \Delta q_2^T (M^{-1}(q_1)(-C(q_1,q_2)q_2 - k(q_1) + u) - \dot{q}_{2d} + K_p \Delta q_1)$$

If the computed torque control is used

$$u = k(q_1) + C(q_1,q_2)q_2 + M(q_1)(\dot{q}_{2d} - K_p\Delta q_1 - K_v \Delta q_2) \qquad (2.34)$$

then

$$\dot{V} \leq - \Delta q_2^T K_v \Delta q_2$$

From the same line of reasoning as before, the closed loop system is globally asymptotically stable. However, we know that the closed loop system is linear, therefore it is exponentially stable. This means that we should look for a better Lyapunov function. An obvious choice is to add in a cross term in (2.33). Then

$$V(\Delta q_1, \Delta q_2) = \frac{1}{2}||\Delta q_2||^2 + \frac{1}{2}\Delta q_1^T (K_p + cK_v)\Delta q_1 + c\Delta q_1^T \Delta q_2 \qquad (2.35)$$

where c is small constant so that V is positive definite. Take derivative and apply (2.34),

$$\begin{aligned}\dot{V} &= - \Delta q_2^T K_v \Delta q_2 + c\Delta q_2^T K_v \Delta q_1 + c||\Delta q_2||^2 - c\Delta q_1^T K_p \Delta q_1 - c\Delta q_1^T K_v \Delta q_2 \\ &= -(\sigma_{min}(K_v) - c)\,||\Delta q_2||^2 - c\Delta q_1^T K_p \Delta q_1\end{aligned}$$

which shows closed loop exponential stability.

Note that in (2.34), in contrast to (2.28), even for set point case, full model nonlinearity cancellation is needed. The approach in this paper is to use the energy Lyapunov function instead of (2.33) to generate control laws. We will see in later sections that this affords a much larger class of controls which contains much simpler structure in certain cases (especially for set point control).

3. New Results on PD Set Point Control

3.1 Simple PD Controls

In this section, we will explore using different U* in the controller design. The following has been suggested in [5]:

$$U^*(\Delta q_1) = \frac{1}{2}\Delta q_1^T K_p \Delta q_1 + g(\Delta q_1 + q_{1d}) - g(q_{1d}) - \Delta q_1^T k(q_{1d}) \qquad (3.1)$$

If K_p is sufficiently large, $\Delta q_1 = 0$ is the global minimum of $U(\Delta q_1)$. Hence, a simpler control law can be used:

$$u = - K_p \Delta q_1 - K_v q_2 + k(q_{1d}) \tag{3.2}$$

Suppose each joint is constrained between joint stops:

$$q_{1i}^{(\ell)} \leq q_{1i} \leq q_{1i}^{(h)} \tag{3.3}$$

and the set point is in the interior of the joint inputs:

$$q_{1i}^{(\ell)} < \underline{q}_{1i} \leq q_{1id} \leq \bar{q}_{1i} < q_{1i}^{(h)} \tag{3.4}$$

Let the desired potential function be

$$U^*(\Delta q_1) = \frac{1}{2} \Delta q_1^T K_p \Delta q_1 + \sum_{i=1}^{n} (H_i(\Delta q_{1i} + q_{1id}) + L_i(\Delta q_{1i} + q_{1id})) \tag{3.5}$$

where H_i and L_i are appropriate upper and lower barrier potential functions for joint i [23].

Then, $\Delta q_1 = 0$ is a global minimum of $U^*(\Delta q_1)$ [23] . From (2.31)

$$u = - K_v q_2 - K_p \Delta q_1 - \dot{H}(\Delta q_1) - \dot{L}(\Delta q_1) + k(q_1) \tag{3.6}$$

Similarly, if K_p is sufficiently large $(K_p + \frac{\partial k}{\partial \Delta q_1} (\Delta q_1 + q_{1d}) > 0)$, the following control law also achieves global asymptotic stability:

$$u = - K_v q_2 - K_p \Delta q_1 - \dot{H}(\Delta q_1) - \dot{L}(\Delta q_1) + k(q_{1d}) \tag{3.7}$$

Control laws (3.2), (3.6) and (3.7) still require information on the gravity load. It is interesting to ask if this last piece of model information can be removed. This case corresponds to the desired potential energy

$$U^*(\Delta q_1) = \frac{1}{2} \Delta q_1^T K_p \Delta q_1 + g(\Delta q_1 + q_{1d}) \tag{3.8}$$

The corresponding control law is

$$u = - K_p \Delta q_1 - K_v \Delta q_2 \tag{3.9}$$

From section 2.4,

$$\frac{\partial U^*}{\partial \Delta q_1} (\Delta q_1) = K_p \Delta q_1 + k(\Delta q_1 + q_{1d}) \to 0$$

This implies

$$\limsup_{t \to \infty} ||\Delta q_1(t)|| \leq \sigma_{\min}(K_p) \sup_{q_1 \epsilon R^n} || k(q_1)||$$

If $K_p + \frac{\partial k(q_1)}{\partial q_1} > 0 \ \forall q_1 \epsilon R^n$, $- K_p k(\Delta q_1 + q_{1d})$ is a contraction map in Δq_1. Then $\exists !\, q_1^*$ such that

$$K_p(q_1^* - q_{1d}) + k(q_1^*) = 0 \tag{3.10}$$

$$\lim_{t \to \infty} q_1(t) = q_1^*$$

This result suggests a very simple, robust and practical control scheme. The feedback gain K_p can be chosen large enough to justify the use of PID control [1, ch. 6 of 19] which is locally stable. Typically, $k(q_1)$ and $\frac{\partial k(q_1)}{\partial q_1}$ are composed of trigonometric functions, therefore, they are uniformly bounded.

3.2 PD Control with Exponential Convergence Rate

Use of the Invariance Principle in Section 2.4 only shows asymptotic stability. Some guaranteed rate of convergence is highly desirable not just for performance reasons but for robustness analysis and adaptive control also. In Section 2.5, a Lyapunov function with cross term has been used to show exponential stability. This suggests a similar modification here. The result is summarized below.

Theorem 3-1

Given the control law (2.31)

$$u = -K_v q_2 - \frac{\partial U^*}{\partial q_1} + \frac{\partial U^o}{\partial q_1} \tag{2.31}$$

Suppose $\exists\ \nu > 0$ such that

$$\Delta q_1{}^T \frac{\partial U^*}{\partial \Delta q_1} (\Delta q_1) > \nu \, ||\Delta q_1||^2 \tag{3.11}$$

and $U^*(\Delta q_1)$ has a global minimum at $\Delta q_1 = 0$, then the closed loop $(\Delta q_1, q_2)$ system is exponentially stable.

Proof:

Modify the total energy Lyapunov function (2.29) to

$$V = T + U^* + c\Delta q_1{}^T p + \frac{1}{2} c\Delta q_1{}^T K_v \Delta q_1$$

where c is a small constant so that V is positive definite in p and q_1. Without loss of generality, U* can be considered positive definite in q_1 (by adding an appropriate constant). Then from (2.32),

$$\dot V = q_2{}^T(u + \frac{\partial U^*}{\partial q_1} - \frac{\partial U^o}{\partial q_1}) + c\ q_2{}^T\ p + c\Delta q_1{}^T(-\frac{\partial T}{\partial q_1} - \frac{\partial U^o}{\partial q_1} + u) + c\ q_2{}^T K_v \Delta q_1$$

$$= -q_2{}^T K_v q_2 + c\ q_2{}^T M(q_1) q_2 - c\ \Delta q_1{}^T \frac{\partial T}{\partial q_1} - c\ \Delta q_1{}^T \frac{\partial U^*}{\partial q_1}$$

Let

$$\mu \overset{\Delta}{=} \sup_{q_1 \in R^n} ||M(q_1)|| \tag{3.12}$$

$$\xi_1 = \inf_{||\Delta q_1||=1} [U^*(\Delta q_1) + \frac{1}{2} c\Delta q_1{}^T K_v \Delta q_1 - \frac{1}{2} c\mu\ell^2] > 0 \quad \text{for some constant } \ell. \tag{3.13}$$

From Lemma 2-1, $\forall\ \lambda_2 \in (0, \alpha_2 - \gamma_{21} \ (\frac{V_o}{\xi_1})^{\frac{1}{2}})$,

$$\dot V \le -\lambda_1 ||\Delta q_1||^2 - \lambda_2\ ||\Delta q_2||^2$$

Note

$$\frac{\partial T}{\partial q_1} = \frac{1}{2} \sum_{i=1}^{n} e_i\ q_2{}^T M_i(q_1)\ q_2 = \frac{1}{2} M_D{}^T(q_1, q_2) q_2 \tag{3.14}$$

Then

$$\dot V \le -(\sigma_{min}(K_v) - c\mu)||a_2||^2 - c\ \nu\ ||\Delta q_1||^2 - \frac{c}{2}\ \Delta q_1{}^T\ M_D{}^T(q_1, q_2) q_2$$

Define

$$\eta_1 = \sup_{q_1} \sum_{i=1}^{n} ||M_i(q_1)|| \tag{3.15}$$

Then

$$\dot V \le -\alpha_1 ||\Delta q_1||^2 - \alpha_2 ||\Delta q_2||^2 + \gamma_{21} ||\Delta q_1||\ ||\Delta q_2||^2$$

where

$$\alpha_1 = c\nu$$

$$\alpha_2 = \sigma_{min}(K_v) - c\mu$$

$$\gamma_{21} = \frac{1}{2} c\ \eta_1$$

Choose

$$c < \sigma_{min}\ (K_v)(\mu + \frac{1}{2} \eta_1\ (\frac{V_o}{\xi_1})^{\frac{1}{2}})^{-1} \tag{3.16}$$

where $V_o = V \big|_{t=0}$ and

$$\leq - \lambda V$$

for some $\lambda > 0$. Hence, the closed loop system is exponentially stable. ▮

Given any U* according to (3.11), $K_v > 0$ and initial condition, there always exists c that satisfies (3.16). Even though c is not needed in the implementation, its maximum allowable size affects the convergence rate. The artificial potentials $U^* = 1/2\, \Delta q_1^T K_p \Delta q_1$, (3.1) and (3.5) all satisfy the assumptions of Theorem 3-1. Therefore, the corresponding closed loop systems are exponentially stable. For the potential given by (3.8) and K_p large enough $(K_p + \frac{\partial k}{\partial q_1} > 0)$, we can add a constant to U* so that U* is positive definite in $q_1 - q_1^*$ and (3.11) is satisfied for $\Delta q_1 = q_1 - q_1^*$, where q_1 solves (3.10). Then Theorem 3-1 implies exponential convergence of q_1 to q_1^* which is within $\sigma_{min}(K_p) \sup_{q_1} ||k(q_1)||$ from the true q_{1d}.

4. New Results in Tracking Control

4.1 Exponentially Stable Algorithms

Frequently a robot is required to follow a prespecified path for continuous action at the end effector (e.g., arc welding), tracking of a moving target (e.g. pick and place operation from conveyer belt) or other high level objectives (e.g., time optimality, collision avoidance, arm singularity avoidance, etc.). This can be posed as the problem of tracking the desired positions and velocities (q_{1d}, q_{2d}) by (q_1, q_2). In this section, we will extend the basic ideas put forth in Section 3 to the tracking problem. The error equation is now in the form

$$\begin{aligned} \Delta\dot{q}_1 &= \Delta q_2 \\ M(q_1)\Delta\dot{q}_2 &= - C(q_1,q_2)q_2 - k(q_1) + u - M(q_1)\dot{q}_{2d} \end{aligned} \tag{4.1}$$

We will first state several direct generalizations of Theorem 3-1. An energy type Lyapunov function similar to (2.33) used in Section 2.5 to motivate computed torque is used here.

<u>Theorem 4-1</u>

Consider (4.1) with the control law

$$u = - K_v\Delta q_2 + k(q_1) - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_1)\dot{q}_{2d} - D(q_1,q_2,q_{2d}) \tag{4.2}$$

where D is given by any one of the following expressions

$$D(q_1,q_2,q_{2d}) = \frac{1}{2}(J(q_1,q_2)q_{2d} - M_D(q_1,q_{2d})q_2) \tag{4.2a}$$

$$D(q_1,q_2,q_{2d}) = \frac{1}{2}(J(q_1,q_{2d})q_2 - M_D(q_1,q_2)q_{2d}) \tag{4.2b}$$

$$D(q_1,q_2,q_{2d}) = \frac{1}{2}(J(q_1,q_{2d})q_{2d} - M_D(q_1,q_2)q_{2d}) \tag{4.2c}$$

$$D(q_1,q_2,q_{2d}) = \frac{1}{2}(J(q_1,q_2)q_2 - M_D(q_1,q_{2d})q_2) \tag{4.2d}$$

Assume $\exists \nu > 0$ such that

$$\Delta q_1^T \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) > \nu\, ||q_1||^2 \tag{4.3}$$

and $U^*(\Delta q_1)$ is positive definite in Δq_1. Then the null state of the $(\Delta q_1, \Delta q_2)$ system is a globally exponentially stable equilibrium.

<u>Proof:</u> Use the following Lyapunov function

$$V(\Delta q_1,\Delta q_2) = \frac{1}{2}\Delta q_2^T M(q_1)\Delta q_2 + U^*(\Delta q_1) + c\, \Delta q_1^T M(q_1)\Delta q_2 + \frac{1}{2} c\, \Delta q_1^T K_v \Delta q_1 \tag{4.4}$$

where c is a small constant, such that V is positive definite in Δq_1 and Δq_2. Take derivative along solution:

$$\begin{aligned} \dot{V}(\Delta q_1,\Delta q_2) = {} & \Delta q_2^T(M(q_1)\Delta\dot{q}_2 + \frac{1}{2} M_D(q_1,\Delta q_2)q_2 + \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + c\, M(q_1)\Delta q_2 \\ & + c\, K_v\Delta q_1) + c\, \Delta q_1^T(M(q_1)\Delta\dot{q}_2 + M_D(q_1,\Delta q_2)q_2) \end{aligned}$$

Substitute (4.1) and (4.2) and usr (2.7)

$$\dot{V}(\Delta q_1,\Delta q_2) = \Delta q_2{}^T(-C(q_1,q_2)q_2 - k(q_1) + u - M(q_1)\dot{q}_{2d} + \tfrac{1}{2} M_D(q_1,\Delta q_2)q_2$$

$$+ \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + c\ M(q_1)\Delta q_2 + c\ K_v\Delta q_1)$$

$$+ c\ \Delta q_1{}^T(-C(q_1,q_2)q_2 - k(q_1) + u - M(q_1)\dot{q}_{2d} + M_D(q_1,\Delta q_2)q_2)$$

$$= r(q_1,q_2,q_{2d}) - \Delta q_2{}^T(K_v - cM(q_1))\Delta q_2 - \Delta q_2{}^T D(q_1,q_2,q_{2d})$$

$$+ c\Delta q_1{}^T(M_D(q_1,\Delta q_2)q_2 - C(q_1,q_2)q_2)$$

$$- c\Delta q_1{}^T \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) - c\Delta q_1{}^T D(q_1,q_2,q_{2d})$$

Apply Identity 5, $r - \Delta q_2{}^T D = 0$. Define η_1, η_2 as follows

$$\eta_1 = \sup_{q_1 \in \mathbb{R}^n} \sum_{i=1}^{n} ||M_i(q_1)||$$

$$\eta_2 = \sup_{q_{2d}} ||q_{2d}||\eta_1$$

From Identity 6,

$$|c\Delta q_1{}^T(M_D(q_1,\Delta q_2)q_2 - C(q_1,q_2)q_2 - D(q_1,q_2,q_{2d})|$$

$$\le c||\Delta q_1||(a\eta_2||\Delta q_2|| + \frac{\eta_1}{2}||\Delta q_2||^2)$$

where $a=\frac{3}{2}$ for (4.2a) , $a=\frac{3}{2}$ for (4.2b) , $a=\frac{5}{2}$ for (4.2c) , $a=\frac{1}{2}$ for (4.2d)

Hence,

$$\dot{V}(\Delta q_1,\Delta q_2) \le -(\sigma_{\min}(K_v) - c\mu)\ ||\Delta q_2||^2 - c\ \nu\ ||\Delta q_1||^2$$

$$+ c\ ||\Delta q_1||(a\eta_2||\Delta q_2|| + \frac{\eta_1}{2}||\Delta q_2||^2)$$

Completing square for the cross term,

$$\dot{V}(\Delta q_1,\Delta q_2) \le -\alpha_1||\Delta q_1||^2 - \alpha_2||\Delta q_2||^2 + \gamma_{21}||\Delta q_1||\ ||\Delta q_2||^2 \qquad (4.5)$$

where

$$\alpha_1 = c(\nu - \tfrac{1}{2}\,a\,\eta_2\rho^2)$$

$$\alpha_2 = \sigma_{\min}(K_v) - c(\mu + \frac{1}{2}\frac{a\eta_2}{\rho^2})$$

$$\gamma_{21} = \tfrac{1}{2}\,c\,\eta_1$$

Given ν, choose $\rho^2 < \frac{2\alpha_1}{a\eta_2}$. By Lemma 2-1, for

$$c < \sigma_{\min}(K_v)\ (\mu + \frac{1}{2}\ \frac{a\eta_2}{\rho^2} + \frac{1}{2}\eta_1\ (\frac{V_o}{\xi_1})^{\frac{1}{2}})^{-1}$$

(V_o, ξ_1 are as defined from the proof of Theorem 3-1) and $\forall\ \lambda_2 \in (0,\alpha_1 - \gamma_{21}\ (\frac{V_o}{\xi_1})^{\frac{1}{2}})$

$$\dot{V} \le -\alpha_1||\Delta q_1||^2 - \lambda_2||\Delta q_2||^2$$

$$\le -\lambda\ V$$

for some $\lambda > 0$. Hence, the closed loop system is exponentially stable.

A common Lyapunov function used for tracking problem has been [9, 10]

$$V(\Delta q_1,\Delta q_2) = \tfrac{1}{2}\ \Delta q_2{}^T\ M(q_1)\Delta q_2 + U^*(\Delta q_1)$$

In this case, a generalization of Invariance Principle to time-varying case is required. There are two possibilities. The result in [Theorem A.7.6, 21] appears promising, but we must verify that (4.1) is positive precompact (in the sence defined in [21]). A more direct route is to use [Lemma 1, 22] which states that if $\Delta\dot{q}_2$ and $\Delta\ddot{q}_2$ are both bounded uniformly in t (it follows from $\dot{V} \leq 0$), then $\Delta q_2(t) \to 0$ implies $\Delta\dot{q}_2(t) \to 0$.

Note that $U^*(\Delta q_1)$ does not depend on time explicitly. This restriction eliminates some of the candidates used in set point case. How to generalize to the case of $U^*(\Delta q_1,t)$ and $\frac{\partial U^*}{\partial t}(\Delta q_1,t)$ not necessarily negative semidefinite is currently under investigation.

Control laws (4.2a-d) all have same stability property nominally. When q_2 is a very noise measurement, as is typically the case, (4.2c) which only uses q_2 once may have better robustness.

Note that all the controllers have structures very similar to computer torque; in fact, if all occurrences of q_{2d} are replaced by q_2, then the nonlinear compensation is exactly the same as the case of computer torque. However, in their present forms, (4.2a-d) cannot take advantage of well known recursive algorithms for inverse dynamics computation [11, 12]. Therefore, we next present slightly modified versions that can be implemented with these algorithms.

Corollary 4-1

Consider (4.1) with the control law

$$u = - K_v\Delta q_2 + k(q_1) - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_1)\dot{q}_{2d} + C(q_1,q_{2d})q_{2d} \tag{4.6}$$

where $U^*(\Delta q_1)$ satisfies the same assumptions as in Theorem 4-1.

If

$$\sigma_{\min}(K_v) > \frac{\eta_2}{2}$$

then the null state of the $(\Delta q_1,\Delta q_2)$ system is a globally exponentially stable equilibrium.

Corollary 4-2

Consider (4.1) with the control law

$$u = - K_v\Delta q_2 + k(q_1) - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_1)\dot{q}_{2d} + C(q_1,q_2)q_2 \tag{4.7}$$

where $U^*(\Delta q_1)$ satisfies the assumptions as in Theorem 4-1. Given a set of possible initial conditions, if K_v is sufficiently large, then the closed loop system is exponentially stable with respect to that set.

If $U^*(\Delta q_1) = \frac{1}{2}\Delta q_1^T K_p \Delta q_1$, (4.7) is actually a modification of the computed torque method with K_p, K_v replacing $M(q_1)K_p$, $M(q_1)K_v$.

So far we have generated many control laws that are similar to computed torque. However, the ones just requiring $K_v > 0$, $\nu > 0$ are not easily implementable, and the easily implementable ones need stronger conditions (K_v sufficiently large). The next control law that we shall present is of very appealing structure: the real time update computations are linear and the off-line computation can take advantage of efficient algorithms (e.g., Newton-Euler type). The trade-off is that K_v and ν must both be large enough for a given set of initial conditions.

Theorem 4-2

Consider (4.1) with the control law

$$u = - K_v\Delta q_1 + k(q_{1d}) - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_{1d})\dot{q}_{2d} + C(q_{1d},q_{2d})q_{2d} \tag{4.8}$$

where $U^*(\Delta q_1)$ satisfies the assumptions as in Theorem 4.1. Given a set of possible initial conditions, if K_v and ν, are sufficiently large, then the closed loop system is exponentially stable with respect to that set.

Typically, $q_2(0) = q_{2d}(0) = 0$ and $\Delta q_1(0)$ is always within 2π. Hence V_o is bounded above, and the result is essentially a global one. This scheme requires both ν and K_v large enough. This requirement is made easier by shifting the computational burden to offline thus allowing very high sampling rates which in turn means high gains can be tolerated.

4.2 Robustness

Lyapunov analysis provides a useful approach to study the robustness issue. Robustness is a much abused word in the literature. Here, we use insensitive design to mean preservation of stability (in the sense of Lagrange) under sufficiently small perturbations. Furthermore, the size of the ultimate bound should vary continuously with the size of perturbations. By robust design we mean a controller design that preserves stability under prescribed size of perturbations. In this section, we will examine frictions, both viscous and Coulomb type, bounded modeling error and bounded actuator and sensor noises.

Friction forces can be approximately modeled by Coulomb friction, or dry friction, and viscous friction due to oil lubrication. For joint i, the frictional force is given by

$$f_{fric}^{(i)} = - F_{1i} \operatorname{sgn}(q_{2i}) - F_{2i} q_{2i} \quad , \quad F_{1i}, F_{2i} > 0 \tag{4.9}$$

Equation (4.1) is then

$$M(q_1)\Delta\dot{q}_2 = - C(q_1,q_2)q_2 - k(q_1) + u - M(q_1)\dot{q}_{2d} - F_1 \operatorname{sgn}(q_2) - F_2 q_2$$

where F_1 and F_2 are diagonal matrices with elements F_{1i}, F_{2i}, respectively, $\operatorname{sgn}(q_2)$ represents a vector with elements $\operatorname{sgn}(q_{2i})$.

From [23], the set point controller is both insensitive and robust. The tracking controllers are also insensitive with respect to frictions. For robust design for a given level of friction, $\sigma_{min}(K_v)$ and ν should be chosen large enough.

Next we consider modeling error in controller implementation. Model parameters k, M, M_D can all be written linearly in constant parameters. Assume bounded errors are incurred in these parameters. Control laws (4.2) are in the form

$$u = - K_v\Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + k(q_1) + M(q_1)\dot{q}_{2d} - D(q_1,q_2,q_{2d}) + \Delta_1 + \Delta_2$$

The additional terms in $\dot{V}$ are

$$- (C\Delta q_1 + \Delta q_2)^T (\Delta_1 + \Delta_2)$$

which, aftar overbounding [23], becomes

$$(c||\Delta q_1|| + ||\Delta q_2||)\,(\delta_1 + \delta_2 + \delta_3\,||\Delta q_2|| + \delta_4\,||\Delta q_2||^2)$$

After completing squares, the overbound over the extra terms in $\dot{V}$ become

$$\frac{c}{2}\left(\delta_3 s_3{}^2 + \frac{(\delta_1+\delta_2)}{s_1{}^2}\right) ||\Delta q_1||^2 + \left(\delta_3 + \frac{c\delta_3}{2s_3{}^2} + \frac{\delta_1+\delta_2}{2\,s_2{}^2}\right) ||\Delta q_2||^2$$

$$+ C\delta_4\,||\Delta q_1||\,||\Delta q_2||^2 + \delta_4\,||\Delta q_2||^3 + \frac{1}{2}(\delta_1+\delta_2)\,(cs_1{}^2 + s_2{}^2)$$

For δ_1, δ_5 sufficiently small, $\exists\, \alpha_1, \alpha_2, \rho > 0$ such that

$$\dot{V} \le - \alpha_1||\Delta q_1||^2 - \alpha_2\,||\Delta q_2||^2 + c(\frac{\eta_1}{2} + \delta_4)\,||\Delta q_1||\,||\Delta q_2||^2 + \delta_4\,||\Delta q_2||^3 + \rho$$

By Lemma 2-2, if $V_o > 0$ and δ_1, δ_5 are small, the system remains Lagrange stable and the ultimate bound vanishes as δ_1, $\delta_5 \to 0$. Hence, the design is insensitive with respect to modeling errors. For robust design for a given level of modeling uncertainties, $\sigma_{min}(K_v)$ and ν should be large enough.

Finally, suppose bounded errors ε_1, ε_2 and ε_3 are incurred in q_1, q_2 and u, respectively. Control laws (4.2) are now in the form

$$u = - K_v\,\Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + k(q_1) + M(q_1)\dot{q}_{2d} - D(q_1,q_2,q_{2d}) + \Delta_1 + \Delta_2$$

Follow the same steps as before, we overbound the extra terms in $\dot{V}$ by sums of $||\Delta q_1||^2$, $||\Delta q_2||^2$, $||\Delta q_1||\,||\Delta q_2||^2$ and constant terms. Again use Lemma 2-2 to conclude that the controller is insensitive to bounded instrument noises and robust for a given level of noise if $\sigma_{min}(K_v)$ and ν are sufficiently large.

As an aside, it should be noted that similar results as derived here hold for computed torque techniques also. For the case of set point control under friction, the insensitivity and robustness properties of the design here do not follow directly from the analysis.

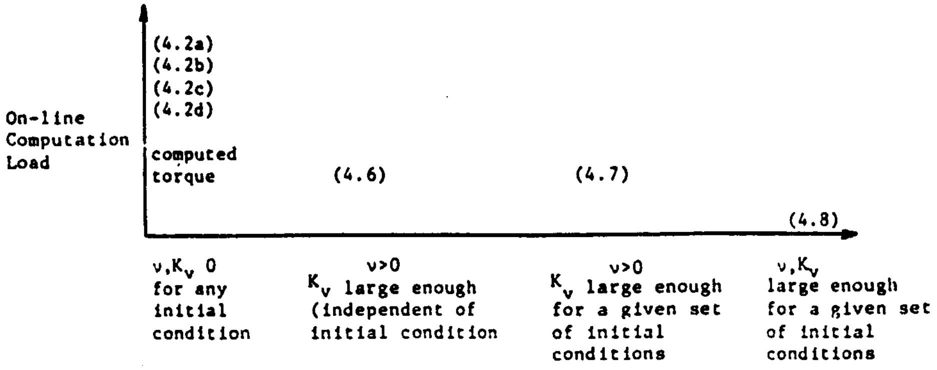
There are obviously many more practical implementation issues not addressed here: sampling effect, actuator saturation, joint and arm flexibilities and instrument dynamics. They are currently under investigation in the same framework.

5. Summary

We have introduced a new class of exponentially stabilizing control laws for the joint level control of robot manipulators (summarized in Table I). The stability result is achieved by making use of a particular class of energy-like Lyapunov functions (of the form (4.4)) in conjunction with a useful lemma (Lemma 2-1) for addressing higher order terms in Lyapunov function derivatives. This approach avoids the need for a generalization of the invariance principle to time-varying systems, which has been the major source of difficulty in the past [9,10].

In the set point control case, by incorporating artificial potential fields in the Lyapunov function, we have derived a class of exponentially stabilizing, PD + potential shaping type of control laws. Several useful potential fields have been examined resulting in simple structures: PD and PD+bias, and the ability to handle joint stop constraints with PD + joint-stop-barrier controller.

In the tracking control case, the modified Lyapunov function leads to a new class of exponentially stable control laws. This class of control laws offers an alternative to the convertional computed torque method and provides trade-offs between on-line computation (which directly relates to performance through maximum sampling rate) and condition for stability. In one new design, (4.8), the nonlinear structure is decoupled from the real-time measurements which completely removes the requirement for on-line nonlinear computation. The chart below illustrates the trade-offs in the various tracking control laws.



The framework of Lyapunov stability analysis also allows robustness issues to be directly addressed. Specifically, insensitivity property (preservation of stability under small perturbation) and conditicn for robust design (preservation of stability under a specified amount of perturbation) for this new class of controllers have been derived with respect to viscous and Coulomb friction, modeling error and bounded instrument noise.

The new stability anslysis and controller design teqhniques presented in this paper open up many promising avenues for future research. In particular, our current research directions include: ways to incorporate time-varying artificial potential fields in the tracking problem and the generalization of the exponentially stabilizing joint-level control laws to the task space.

Acknowledgement

The authors would like to thank Dr. K. Kreutz for many helpful discussions. This research has been performed at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration.

6. References

[1] S. Arimoto and F. Miyazaki, "Stability and Robustness of PID Feedback Control for Robot Manipulators of Sensory Capability," Proc. Int. Symp. Robotics Research, MIT Press, Cambridge, MA, 1983.

[2] A.K. Bejczy, Robot Arm Dynamics and Control, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California TM 33-669, Feb., 1974.

[3] L. Meirovitch, Methods of Analytical Dynamics, McGraw Hill, New York, 1970.

[4] R. Pringle, Jr., "On the Stability of a Body with Connected Moving Parts," AIAA Journal, Vol. 4, No. 8, Aug., 1966.

[5] M. Takegaki and S. Arimoto, " A New Feedback Method for Dynamic Control of Manipulators," ASME J. Dynamic Systems, Measurement and Control, Vol. 102, June, 1981.

[6] D.F. Golla, Linear State Feedback Control of Rigid Link Manipulators, University of Toronto, Institute for Aerospace Studies, May, 1979.

[7] D.E. Koditschek, "Natural Motion for Robot Arms," Proc. IEEE Conf. Decision and Control, Las Vegas, NV, 1984.

[8] F. Miyazaki and S. Arimoto, "Sensory Feedback for Robot Manipulators," J. of Robotic Systems, Vol. 2, 1985.

[9] D.E. Koditschek, "Natural Control of Robot Arms," submitted for publication.

[10] B. Paden and J.-J.E.Slotine, "PD + Compensation Robot Controllers: Tracking and Adaptive Control," IEEE Int'l. Conf. on Robotics and Automation, Raleigh, N.C., 1987.

[11] J.Y.S. Luh, M.W. Walker and R.P.C. Paul, "On-Line Computational Scheme for Mechanical Manipulators," ASME J. Dynamic Systems Measurement, and Control, Vol. 102, 1981.

[12] G. Rodriguez, Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics, JPL Publication 86-48, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 1986.

[13] D.S. Bayard and J.T. Wen, "Simple Robust Control Laws for Robot Manipulators, Part II: Adaptive Case," Proceedings of the Workshop on Space Telerobotics, JPL Publication 87-13, Jet Propulsion Laboratory, Pasadena, California, 1987.

[14] J.T. Wen, "On Minimum Time Control of Robot Manipulators," in Recent Trends in Robotics, Modeling, Control and Education, Ed. by M. Jamshidi, J.Y.S. Luh and M. Shahinpoor, North-Holland, 1986.

[15] D.D. Siljak, Large Dynamic Systems, North-Holland, Amsterdam, The Netherlands, 1978.

[16] J.P. LaSalle, "Some Extensions of Lyapunov's Second Method," IRE Trans. on Circ. Theory, Dec., 1960.

[17] N. Rouche, P. Habets and M. Laloy, Stability Theory by Lyapunov's Direct Method, Springer-Verlag, New York, 1977.

[18] J.-J.E. Slotine and W. Li, "On the Adaptive Control of Robot Manipulators," ASME Winter Meeting, Anaheim, CA, 1986.

[19] J.K. Hale, Ordinary Differential Equations, Wiley Interscience, New York, 1969.

[20] H. Asada and J.-J.E. Slotine, Robot Analysis and Control, John Wiley and Sons, 1986.

[21] Z. Arstein, Limiting Equations and Stability of Nonautonomous Ordinary Differential Equations, in The Stability of Dynamical Systems, by J.P. LaSalle, CBMS Regional Conference Series in Applied Math, 1976.

[22] J.S.C Yuan and W.M. Wonham, "Probing Signals for Model Reference Identification," IEEE Trans. of Aut. Contr., AC-22, No. 4, 1977.

[23] J.T. Wen and D.S. Bayard, Robust Control for Robotic Manipulators-Part I: Non-Adaptive Case, (JPL Internal Document, Engineering Memorandum, EM 347-87-203), Jet Propulsion Laboratory, Pasadena, California, 1987. Pasadena, California, 1987

[24] D.S. Bayard and J.T. Wen, Robust Control for Robotic Manipulators-Part II: Adaptive Case, (JPL Internal Document, Engineering Memorandum, EM 347-87-204), Jet Propulsion Laboratory, Pasadena, California, 1987. Pasadena, California, 1987.

Equation Number	Control Laws	Condition for Exponential Stability	Comments
(2.31)	$u = -K_v q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + \frac{\partial U^0}{\partial q_1}(\Delta q_1)$	$U^*(\Delta q_1)$ has global minimum at $\Delta q_1 = 0$ $\nu > 0$ $\Delta q_1^T \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) > \nu \|\Delta q_1\|^2$ $K_v > 0$	General Set Point Control
(2.28)	$u = -K_v q_2 - K_p \Delta q_1 + k(q_1)$	$K_p > 0$, $K_v > 0$	$U^*(\Delta q_1) = \frac{1}{2} \Delta q_1^T K_p \Delta q_1$
(3.2)	$u = -K_v q_2 - K_p \Delta q_1 + k(q_{1d})$	$K_p > \sup_{q_1} \frac{\partial k(q_1)}{\partial q_1}$, $K_v > 0$	$U^*(\Delta q_1) = \frac{1}{2} \Delta q_1^T K_p \Delta q_1 + g(\Delta q_1 + q_{1d}) - g(q_{1d})$ $- \Delta q_1^T k(q_{1d})$
(3.6)	$u = -k_v q_2 - K_p \Delta q_1 - \dot{H}(\Delta q_1) - \dot{L}(\Delta q_1) + k(q_1)$	$K_p > 0$, $K_v > 0$, $\underline{q}_{1i} \le q_{1id} \le \bar{q}_{1i}$	$U^*(q_1) = \frac{1}{2} \Delta q_1^T K_p \Delta q_1 + \sum_{i=1}^{n} (H_i(\Delta q_{1i} + q_{1id}) + L_i(\Delta q_{1i} + q_{1id})$
(3.7)	$u = -K_v q_2 - K_p \Delta q_1 - \dot{H}(\Delta q_1) - \dot{L}(\Delta q_1) + k(q_{1d})$	$K_p > \sup_{q_1} \frac{\partial k(q_1)}{\partial \Delta q_1}$, $K_v > 0$, $\underline{q}_{1i} \le q_{1id} \le \bar{q}_{1i}$	$U^*(\Delta q_1) = \frac{1}{2} \Delta q_1^T K_p \Delta q_1 + g(\Delta q_1 + q_{1d}) - g(q_{1d})$ $- \Delta q_1^T k(q_{1d}) + \sum_{i=1}^{n} (H_i(\Delta q_{1i} + q_{1id}) + L_i(\Delta q_{1i} + q_{1i})$
(3.9)	$u = -K_v q_2 - K_p \Delta q_1$	$\limsup_{t \to \infty} \|\Delta q_1(t)\| \le \sigma_{\min}(K_p) \sup_{q_1} \|k(q_1)\|$, $K_v > 0$ If $K_p > \sup_{q_1} \frac{\partial k(q_1)}{\partial q_1}$, $\lim_{t \to \infty} q_1(t) = q_1^*$, $K_p(q_1^* - q_{1d}) + k(q_1^*) = 0$	$U^*(\Delta q_1) = \frac{1}{2} \Delta q_1^T K_p \Delta q_1 + g(\Delta q_1 + q_{1d})$ Global Lagrange Stability
(2.34)	$u = M(q_1)(-K_p \Delta q_1 - K_v \Delta q_2 - \dot{q}_{2d}) + k(q_1) + C(q_1, q_2) q_2$	$K_p > 0$, $K_v > 0$	Computed Torque. Newton-Euler Algorithm can be used to update control law.
(4.2a)	$u = -K_v \Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}(J(q_1, q_2) q_{2d} - M_D(q_1, q_{2d}) q_2)$	$K_v > 0$ U^* has global minimum at $\Delta q_1 = 0$ U^* time - invariant $\nu > 0$ $\Delta q_1^T \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) > \nu \|\Delta q_1\|^2$	Convergence rate depends on initial condition. Newton-Euler algorithm not applicable
(4.2b)	$u = -K_v \Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}(J(q_1, q_{2d}) q_2 - M_D(q_1, q_2) q_{2d})$	Same as (4.2a)	Same as (4.2a)
(4.2c)	$u = -K_v \Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}(J(q_1, q_{2d}) q_{2d} - M_D(q_1, q_2) q_{2d})$	Same as (4.2a)	Same as (4.2a)
(4.2d)	$u = -K_v q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}(J(q_1, q_2) q_2 - M_D(q_1, q_{2d}) q_2)$	Same as (4.2a)	Same as (4.2a)
(4.6)	$u = -K_v \Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + k(q_1) + M(q_i) q_{2d} + C(q_1, q_{2d}) q_{2d}$	Same condition on U^* as in (4.2a) ; $K_v > \frac{n_2}{2}$	Newton-Euler algorithm can be applied to $(q_1, q_{2d}, \dot{q}_{2d})$. Convergence rate depends on initial condition.
(4.7)	$u = -K_v \Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + k(q_1) + M(q_1) q_{2d} + C(q_1, q_2) q_2$	$\nu > 0$, K_v sufficiently large with respect to a given set of initial conditions	Modified Computed Torque. Newton-Euler algorithm can be applied to $(q_1, q_2, \dot{q}_{2d})$. Convergence rate depends on initial condition.
(4.8)	$u = -K_v \Delta q_2 - \frac{\partial U^*}{\partial \Delta q_1}(\Delta q_1) + k(q_{1d}) + M(q_{1d}) q_{2d} + C(q_{1d}, q_{2d}) q_{2d}$	ν , K_v sufficiently large with respect to a given set of initial conditions	Nonlinear compensation can be computed off-line by applying Newton-Euler algorithm to $(q_{1d}, q_{2d}, \dot{q}_{2d})$. Convergence rate depends on initial condition.

TABLE I - SUMMARY OF CONTROL LAWS -

Simple Robust Control Laws for Robot Manipulators, Part II: Adaptive Case

D.S. Bayard and J.T. Wen
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

1. Abstract

A new class of asymptotically stable adaptive control laws is introduced for application to the robotic manipulator. Unlike most applications of adaptive control theory to robotic manipulators, this analysis addresses the nonlinear dynamics directly without approximation, linearization, or ad-hoc assumptions, and utilizes a parameterization based on physical (time-invariant) quantities. This approach is made possible by using energy-like Lyapunov functions which retain the nonlinear character and structure of the dynamics, rather than *simple* quadratic forms which are ubiquitous to the adaptive control literature, and which have bound the theory tightly to linear systems with unknown parameters. It is a unique feature of these results that the adaptive forms arise by straightforward certainty equivalence adaptation of their nonadaptive counterparts found in the companion to this paper (i.e., by replacing unknown quantities by their estimates) and that this simple approach leads to asymptotically stable closed-loop adaptive systems. Furthermore, it is emphasized that this approach does not require convergence of the parameter estimates (i.e., via persistent excitation), invertibility of the mass matrix estimate, or measurement of the joint accelerations.

1. Introduction

In past years, many papers have appeared on the application of adaptive control theory to robotic manipulators (cf., [2]-[7], and Hsia [8] for overview). It is a general property of adaptive designs based on Lyapunov's Direct Method, that the Lyapunov function is chosen as a simple quadratic type, well-known and well studied in the standard adaptive control literature [12][13]. However, this particular Lyapunov function was originally motivated for applications to the standard adaptive control problems (i.e., linear systems with unknown parameters), and not for nonlinear dynamical systems. Hence, applications of standard adaptive control techniques to robotic manipulators invariably require the dynamics to be considered as linear. This in turn, requires the use of ad-hoc assumptions and/or analysis techniques including 1) treatment of *position dependent quantities as unknown constants, for which they must be assumed* to vary slowly with time; 2) linearization of the system about some local operating point-valid only for small excursions from nominal; 3) the use of linear decoupled models for the links, which neglects nonlinearities and crosscoupling effects; and 4) neglecting the nonlinear and time-varying dynamics completely by assuming the plant is linear. Hence, stability results based on these assumptions are questionable, and a rigorous proof of stability for adaptive control of robotic manipulators remains unresolved.

A recent exception to the above criticism is due to the work of Craig, Hsu and Sastry [9]. Here, a useful "linear in the parameters" formulation is exploited to simplify the analysis, and to demonstrate global convergence of an adaptive version of the computed-torque control law - without approximation to the nonlinear dynamics. However, the resulting adaptive controller requires the invertibility of the mass matrix estimate (which is not guaranteed a-priori), and measurement of the joint accelerations (which is generally unavailable). It is suggested in [9], that the former can be handled by projecting parameter estimates into known regions of parameter space for which the mass matrix inverse exists, and in which the true parameters are required to lie. However, knowledge and calculation of such regions is not straightforward and appears to be a weakness of the method.

In this paper, the "linear in parameters" formulation of [9] is used in conjunction with a different Lyapunov function. Here, the choice of Lyapunov function is more closely related to the energy of the system, and better retains the nonlinear structure and character of the dynamics. In addition, many problems associated with adapting the computed-torque control law directly are avoided by making use of the new class of exponentially stabilizing controllers introduced in [1]. Although these controllers are very similar in form to the computed torque method, they have many advantages in the nonadaptive case (cf., [1]), and have the unique property that they can be made adaptive by using a straightforward certainty equivalence approach (i.e., by replacing unknown quantities by their on-line estimates). Furthermore, the class of adaptive systems defined in this manner can be shown to be asymptotically stable without approximation to the nonlinear manipulator dynamics. This approach does not require convergence of parameter estimates (i.e., via persistent excitation), invertibility of the mass matrix estimate, or measurement of joint accelerations.

In the most recent literature (i.e., preprints, conference papers, etc.) there appears to be other work currently taking place which combines the linear in parameters formulation with a new Lyapunov function [10], [11]. Although this work is very new and is evolving very rapidly, we will try to contrast our results where possible, and provide an overall perspective.

The format of the paper is as follows. In Sec. 2 the results of [1] are reviewed and summarized as required for treatment of the adaptive control case. In Sec. 3, asymptotic stability is proved for the class of systems arising from certainty equivalence adaptation of the control laws in [1]. Slightly tangential to the main thrust of the paper is the analysis in Sec. 4 of the adaptive computed torque method. Since the computed-torque control law is widely established in the literature, and widely applied in practice, it is useful to apply the techniques developed herein to see to what extent it can be made adaptive and to what extent stability can be guaranteed. In Sec. 5, several remarks are made pertinent to the new adaptive designs, and conclusions are given in Sec. 6.

2. Background and Notation

2.1 Manipulator Dynamics

The well-known Lagrange-Euler equations of motion for the n-joint manipulator is given as follows,

$$\dot{q}_1 = q_2 \tag{2.1}$$

$$M(q_1)\dot{q}_2 = - C(q_1,q_2) - k(q_1) + u \tag{2.2}$$

where

$$C(q_1,q_2) = \sum_{i=1}^{n} [(e_i q_2^T M_i(q_1))^T - \frac{1}{2} (e_i q_2^T M_i(q_i))] \tag{2.3}$$

$$e_i \overset{\Delta}{=} i^{th} \text{ unit vector}$$

$$M_i(q_1) = \frac{\partial M(q_1)}{\partial q_{1i}} \quad ; \quad q_{1i} \overset{\Delta}{=} i^{th} \text{ component of } q_1$$

$$k(q_1) \overset{\Delta}{=} \text{gravity load}$$

Here, $u \varepsilon R^n$ is a generalized torque vector, q_1, q_2, $\dot{q}_2 \varepsilon R^n$ are generalized joint position, velocity and acceleration vector, (e.g., q_1 is an angle or a distance for a revolute or prismatic joint, respectively, $M(q_1) \varepsilon R^{nxn}$ is the symmetric positive definite mass inertia matrix; $C(q_1,q_2) \varepsilon R^n$ is the Coriolis and centrifugal force vector; and $k(q_1) \varepsilon R^n$ is the gravitational load vector.

2.2 Some Useful Identities

Let the following notations be defined,

$$M_D(q_1,z) = \sum_{i=1}^{n} M_i(q_1) z\, e_i^T$$

$$\dot{M}(q_1,q_2) = \frac{d}{dt} M(q_1) = \sum_{i=1}^{n} M_i(q_1)\, e_i^T q_2$$

$$J(q_1,z) = \sum_{i=1}^{n} [(e_i z^T M_i(q_1) - (e_i z^T M_i(q_1))^T]$$

$$\Delta q_1 \overset{\Delta}{=} q_1 - q_{1d} \quad , \quad \Delta q_2 \overset{\Delta}{=} q_2 - q_{2d}$$

$$q_{1d},\ q_{2d} \overset{\Delta}{=} \text{desired joint position and velocities respectively} \quad (q_{2d} = \dot{q}_{1d})$$

$$r(q_1,q_2,q_{2d}) = \Delta q_2^T [\frac{1}{2} M(q_1,q_2) \Delta q_2 - C(q_1,q_2) q_2]$$

Using the above notation, the following identities are quoted from [1] without proof. In these identities, x, y and z are used to denote arbitrary vectors of appropriate dimension,

<u>Identity 1</u>

$$\dot{M}(q_1,q_2) z = M_D(q_1,z) q_2 \qquad \text{where vector } z \text{ is arbitrary}$$

<u>Identity 2</u>

$$C(q_1,z) z = \frac{1}{2} (M_D(q_1,z) - J(q_1,z)) z$$

<u>Identity 3</u>

$$J(q_1,z) = M_D^T(q_1,z) - M_D(q_1,z)$$

2.3 Important Lemma

In this section, a useful lemma is reviewed, quoted directly without proof from [1]. For convenience, this result will be alternatively referred to as the β-Ball Lemma due to the method used to prove it.

Lemma 2-1 (β-Ball Lemma)

Given a dynamical system

$$\dot{x}_i = f_i(x_1,\ldots x_N,t) \quad , \quad x_i \epsilon R^{ni} \quad , \quad t \geq 0$$

Let f_i's be locally Lipschitz with respect to $x_1,\ldots,x_N$ uniformly in t on bounded intervals and continuous in t for $t \geq 0$. Suppose a function $V: R^{n_1 \times \cdots \times n_N} \times R_+ \rightarrow R_+$ is given such that

$$V(x_1,\ldots,x_N,t) = \sum_{i,j=1}^{N} x_i^T P_{ij}(x_1,\ldots,X_j,t)x_j \quad ,$$

V is positive definite in $x_1,\ldots,x_N$

$$\dot{V}(x_1,\ldots,x_N,t) \leq - \sum_{i\epsilon I_1} (\alpha_i - \sum_{j\epsilon I_{2i}} \gamma_{ij}||x_j(t)||^{k_{ij}})||x_i(t)||^2 \tag{2.4}$$

where α_i, γ_{ij}, $k_{ij} > 0$, $I_{2i} \subset I_1 \subset \{1,\ldots,N\}$

Let $\xi_i > 0$ be such that,

$$\xi_i||x_i||^2 \leq V(x_1,\ldots,x_N,t) \tag{2.5}$$

Let $V_o \overset{\Delta}{=} V(x,(0),\ldots,x_N(0),0)$

If $\forall i \epsilon I_1$,

$$\alpha_i > \sum_{j\epsilon I_{2i}} \gamma_{ij}(\frac{V_o}{\xi_j})^{\frac{k_{ij}}{2}} \tag{2.6}$$

then $\forall\ \lambda_i \epsilon (0,\ \alpha_i - \sum_{j\epsilon I_{2i}} \gamma_{ij}(\frac{V_o}{\xi_j})^{\frac{k_{ij}}{2}})$,

$$\dot{V}(x_1,\ldots,x_N,t) \leq - \sum_{i\epsilon I_1} \lambda_i\ ||x_i||^2 \quad \forall\ t \geq 0 \qquad \blacksquare$$

2.4 Exponentially Stabilizing Control Laws

In [1], various new exponentially stabilizing compensators were introduced for both the set-point and tracking control problems. For the purposes of adaptive control, it is of interest to consider the subset of this class summarized in Table I. In addition, the well-known computed torque control has also been included in Table I for comparison purposes. It is noted that the desired potential field $U^*(\Delta q_1)$ used in [1] has been chosen here simply as,

$$U^*(\Delta q_1) = \frac{1}{2}\ \Delta q_1^T\ K_p \Delta q_1, \tag{2.7}$$

so as not to obscure the presentation with additional obstacle avoidance objectives. Nevertheless, many of the adaptive control results presented herein are easily extended to the more general case.

It is useful to observe that all Control Laws 1-7 differ from the computed torque method in that the mass matrix $M(q_1)$ does not premultiply the position and velocity feedback gains K_p and K_v respectively. This property is critical since it renders this entire class of control laws amenable to simple adaptation schemes (i.e., certainty equivalence adaptation) which can be shown to lead to desired asymptotic stability properties. The presence of the mass matrix premultiplier otherwise prevents simple cancellations in the Lyapunov function derivative, hindering most attempts to apply adaptive control directly to the nonlinear dynamic manipulator equations. A recent exception to this can be found in the work of Craig, Hsu and Sastry [9].
However, the resulting adaptation law requires that the estimated mass matrix be invertible for all values of estimated parameters. This in turn requires on-line projections of parameter estimates into prespecified bounded regions of parameter space where $\hat{M}(q_1)$ is not only invertible, but where the true parameters are certain to lie. This approach not only requires tight bounds on parameter uncertainty, but involves a very difficult (al beit off-line) determination of the proper parameter projection domains. This problem is further exacerbated by the fact that the adaptation law is not parameterized by physical parameters and is of the form where the transformation back to physical parameters is neither straightforward or unique. These problems are overcome in this paper by using the exponentially stabilizing control laws of Table I, which do not involve a premultiplying mass matrix on the feedback gains.

TABLE I

COMPUTED TORQUE CONTROL LAW	CONDITIONS FOR STABILITY[†]	CROSS-REFERENCE TO [1][*] AND [20]
$u = -M(q_1)(K_p\Delta q_1 + K_v\Delta q_2) + k(q_1) + M(q_1)\dot{q}_{2d} + C(q_1,q_2)q_2$	None required	(2.34)
NEW EXPONENTIALLY STABLE CONTROL LAWS		
1. $u = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}J(q_1,q_2)q_{2d} + \frac{1}{2}M_D(q_1,q_{2d})q_2$	None required	(4.2a)
2. $u = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}J(q_1,q_{2d})q_2 + \frac{1}{2}M_D(q_1,q_2)q_{2d}$	None Required	(4.2b)
3. $u = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}J(q_1,q_{2d})q_{2d} + \frac{1}{2}M_D(q_1,q_2)q_{2d}$	None Required	(4.2c)
4. $u = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}J(q_1,q_2)q_2 + \frac{1}{2}M_D(q_1,q_{2d})q_2$	None Required	(4.2d)
5. $u = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_1) + M(q_1)\dot{q}_{2d} + C(q_1,q_{2d})q_{2d}$	$\sigma_{min}(K_v) > \frac{\eta_2}{2}$	(4.6)
6. $u = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_1) + M(q_1)\dot{q}_{2d} + C(q_1,q_2)q_2$	$\sigma_{min}(K_v)$ sufficiently large w.r.t. Initial condition	(4.7)
7. $u = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_{1d}) + M(q_{1d})\dot{q}_{2d} + C(q_{1d},q_{2d})q_{2d}$	$\sigma_{min}(K_v)$ sufficiently large w.r.t. Initial condition	(4.8)

[†] General Assumptions $\|q_{1d}\|$, $\|q_{2d}\|$, $\|\dot{q}_{2d}\|$ bounded; $k_v = K_v^T > 0$, $K_p = K_p^T > 0$

[*] Let $U^*(\Delta q_1) \triangleq \frac{1}{2}\Delta q_1^T K_p \Delta q_1$ in [1]

In the nonadaptive case, comparisons between the new control laws of Table I and the computed torque method can be found in [1]. Nevertheless, a brief account is in order here. In particular, Control Laws 1, 2,3,4 are roughly "on par" with the computed torque method in the nonadaptive case, guaranteeing exponential stability with no conditions on K_p or K_v. Unlike the computed torque method, however, they are not in a form suitable for application of the recursive Newton-Euler computation technique. This presently appears to be their major disadvantage. In order to overcome this difficulty, Control Laws 5, 6 and 7 were developed in a form suitable for recursive Newton-Euler computation. Relative to the computed torque method, Control Law 5 utilizes the desired velocity signal q_{2d} in place of the measured velocity q_2 in the nonlinear terms of the controller. This "cleans up" the feedback signal in the sense that nonidealities due to sensor dynamics and measurement noise in q_2 are avoided in the nonlinear feedback terms. Control Law 7 further replaces q_1 in K, M and C by q_{1d}. This decouples the nonlinear terms from real-time measurements, which completely removes the requirement for on-line computation of nonlinear terms in the controller implementation. Control Law 6 is exactly the computed torque method without the premultiplying mass matrix term described earlier. The advantages of these controllers are off-set slightly by the conditions imposed on K_p and K_v for guaranteeing asymptotic stability i.e., that K_v be chosen sufficiently large for Control Laws 1, 2, 3, 4, 5, 6 and that both K_v and K_p be chosen sufficiently large for Control Law 7. It will be seen in the adaptive case that these requirements can be removed by adapting these feedback gains approoriately.

The use of q_{2d} rather than q_2 in many of the new control laws offers additional advantages. In particular, in the set-point control application $q_{2d}=\dot{q}_{2d}=0$. Hence, there is considerable simplification in the control laws relative to the computed torque method, i.e., the nonlinear terms vanish from the control law. This simplification carries over directly to the adaptive case and provides substantial simplification in set-point control relative to the recent adaptive control laws of Slotine and Li [11] and Paden [10].

3. A New Class of Asymptotically Stable Adaptive Control Laws

All of the new exponentially stabilizing control laws summarized in Table I have the unique property that can be adapted in real-time so as to yield asymptotically stable adaptive control systems. Furthermore, the adaptation is done in a certainty equivalence fashion, i.e., by simply replacing unknown quantities in the control laws by their estimates - as generated by an appropriate parameter adaptation algorithm. In this section, asymptotic stability for the various control laws will be proved, and the proper mechanisms for parameter adaptation will be derived.

The simplicity in structure of the adaptive control schemes presented here is largely due to a "linear in the parameters" formulation of the problem. This particular parameterization is becoming increasingly popular in recent literature (cf., [9][10][18][19]) and will be discussed in more detail below.

3.1 Linear in the Parameters Formulation

A useful parameterization of the nonlinear dynamical equations arises by noting the following relations (x, y and z arbitrary vectors),

$$C(x,y,)y = H_C(x,y)\theta_C$$

$$M(x)z = H_M(x,z)\theta_M$$

$$k(x) = H_k(x)\theta_k$$

$$M_D(x,y)y = H_D(x,y)\theta_D$$

where H_C, H_M, H_k and H_D are known matrix valued functions of x, y and z, and where θ_C, θ_M, θ_k, and θ_D are vectors of constant parameters related directly to true physical parameters (masses, inertias, link lengths, center of gravities, etc.). It is emphasized that this parameterization does not contain any hidden "slowly varying" states in the parameter vector definition and does not require any linearization of the dynamical equations of motion.

3.2 Global Asymptotic Stability for Adaptation of Control Laws 1, 2, 3, 4

In this section, global asymptotic stability is proved for adaptation of Control Laws 1, 2, 3 and 4. In order to avoid redundant analysis, the details of the proof will be considered only for Control Law 1, and the extension to the other control laws will follow immediately by taking advantage of the unified treatment of these control laws given in [1].

3.2.1 Asymptotic Stability

Consider Control Law 1,

$$u^o = -K_p\Delta q_1 - K_v\Delta q_2 + k(q_1) + M(q_1)\dot{q}_{2d} - \frac{1}{2}J(q_1,q_2)q_{2d} + \frac{1}{2}M_D(q_1,q_{2d})q_2 \tag{3.1}$$

Here, superscript "o" is used to denote the ideal nonadaptive control law, i.e., the completely "tuned" control law which would be used if the parameters were known exactly. Using the linear in the parameters formulation discussed in Sec. 3.1 there exists a matrix $H_1(q_1, q_2, q_{2d}, \dot{q}_{2d})$ and a vector of parameters θ such that,

$$u^o = -K_p\Delta q_1 - K_v\Delta q_2 + H_1\theta \tag{3.2}$$

where

$$H_1\theta \triangleq M(q_1)\dot{q}_{2d} - \frac{1}{2}J(q_1,q_2)q_{2d} + \frac{1}{2}M_D(q_1,q_{2d})q_2 \tag{3.3}$$

Here, the parameters in θ are constant with time and are related directly to physical link and payload parameters. When these parameters are unknown, the parameter vector θ is replaced by its estimate $\theta(t)$ in real-time to give the following adaptive control law,

$$u = -K_p\Delta q_1 - K_v\Delta q_2 + H_1\hat{\theta} \tag{3.4}$$

Subtracting (3.2) from (3.4) and rearranging gives

$$u = u^o + H_1(\hat{\theta}-\theta) \triangleq u^o + H_1\phi \tag{3.5}$$

This is an important relation since it shows that the adaptive control is equal to the nonadaptive control plus an expression which is linear in the parameter error $\phi \triangleq \hat{\theta}-\theta$.

The proof of stability then follows by choosing the following Lyapunov function,

$$V = V^o + \frac{1}{2}\phi^T\Gamma\phi \qquad \Gamma = \Gamma^T > 0 \tag{3.6a}$$

where V^o is the Lyapunov function for the nonadaptive control law used in [1], and where $\phi^T\Gamma\phi$ is a positive definite function in the parameter error ϕ. For completeness, V^o is rewritten here (cf., [1],(4.4) where $U^*(\Delta q_1) \triangleq \frac{1}{2}\Delta q_1^T K_p \Delta q_1$),

$$V^o = \frac{1}{2}\,\Delta q_2{}^T\, M(q_1)\Delta q_2 + \frac{1}{2}\,\Delta q_1{}^T\,(K_p{+}cK_v)\Delta q_1 + c\Delta q_1{}^T\, M(q_1)\Delta q_2 \qquad (3.6b)$$

Taking the derivative of V along system trajectories and substituting control law (3.5) gives upon rearranging,

$$\dot{V} = \dot{V}^o + (\Delta q_2 + c\Delta q_1)^T\, H_1\phi + \dot{\phi}^T\Gamma\phi \qquad (3.7)$$

where $\dot{V}^o$ is the Lyapunov function derivative for the nonadaptive case, and where the additional terms involving ϕ on the right hand side of (3.7) arise directly from the additional terms involving ϕ in the control law (3.5) and the Lyapunov function (3.6) respectively.

The second and third terms of (3.7) are cancelled exactly by the choice of adaptation law,

$$\dot{\phi} = \dot{\hat{\theta}} = -\,\Gamma^{-1}H_1{}^T(\Delta q_2 + c\Delta q_1) \qquad (3.8)$$

The expression for the remaining term V^o is simply taken from [1] as, (see [1], (4.5) where $\nu \overset{\Delta}{=} \sigma_{min}(K_p)$, also note that Control Law 1 corresponds to case (4.2b) for which $a = \frac{3}{2}$)

$$\dot{V} = \dot{V}^o$$

$$= -\,\alpha_1||\Delta q_1||^2 - \alpha_2||\Delta q_2||^2 + \gamma_{21}\;||\Delta q_1||\;||\Delta q_2||^2 \qquad (3.9)$$

where

$$\alpha_1 = c(\sigma_{min}(K_p) - \frac{3}{4}\,\eta_2\rho^2) \qquad (3.10a)$$

$$\alpha_2 = \sigma_{min}(K_v) - c(\mu + \frac{3}{4}\,\frac{\eta_2}{\rho^2}) \qquad (3.10b)$$

$$\gamma_{21} = \frac{c}{2}\,\eta_1 \qquad (3.10c)$$

$$\eta_2 = \max_{q_{2d}}\; ||q_{2d}||\eta_1 \qquad (3.11a)$$

$$\eta_1 = \max_{q_1}\; (\sum_i\; ||M_i(q_1)||) \qquad (3.11b)$$

$$\mu = \max_{q_1}\; ||M(q_1)|| \qquad (3.11c)$$

$0 < c < \ell^2$ arbitrary

ρ^2 arbitrary

ℓ^2 arbitrary

Applying the β-ball argument of Lemma 2.1 to (3.9) using the values of α_1, α_2, and γ_{21} given in (3.10), it follows that if,

$$\sigma_{min}(K_p) > \frac{3}{4}\,\eta_2\,\rho^2 \qquad (3.12a)$$

$$\sigma_{min}(K_v) > c(\mu + \frac{3}{4}\,\frac{\eta_2}{\rho^2} + \frac{\eta_1}{2}\,(\frac{V_o}{\xi_1})^{\frac{1}{2}}) \qquad (3.12b)$$

Then,

$$\dot{V} \leq -\,\lambda_1||\Delta q_1||^2 - \lambda_2||\Delta q_2||^2 \qquad (3.13)$$

for any λ_1 and λ_2 such that,

$$\lambda_1\; \varepsilon(0,\; c(\sigma_{min}(K_p) - \frac{3}{4}\,\eta_2\,\rho^2)) \qquad (3.14a)$$

$$\lambda_2\; \varepsilon(0,\; \sigma_{min}(K_v) - c(\mu + \frac{3}{4}\,\frac{\eta_2}{\rho^2} + \frac{\eta_1}{2}\,(\frac{V_o}{\xi_1})^{\frac{1}{2}}) \qquad (3.14b)$$

where

$$V_o = V\Big|_{t=0} \qquad (3.15)$$

$$\xi_1 = \frac{1}{2}\,[\sigma_{min}(K_p + cK_v) - \ell^2\, c\, \underline{\sigma}(M)] \qquad (3.16a)$$

$$\xi_2 = \frac{1}{2}\,(1 - \frac{c}{\ell^2})\;\underline{\sigma}\;(M) \qquad (3.16b)$$

$$\underline{\sigma}\;(M) \overset{\Delta}{=} \min_{q_1}\; \sigma_{min}\;(M(q_1))$$

Since ρ^2 is arbitrary, it can be chosen sufficiently small so that (3.12a) is satisfied. With this choice of ρ^2, the value of c in (3.12b) can be chosen sufficiently small so that inequality (3.12b) is satisfied. Hence (3.13) follows. This is essentially the same proof of stability as in the nonadaptive case (c.f., [1] Theorem 4-1) with the following exceptions,

1) The value of V_o in (3.12b) and (3.14b) now includes the initial parameter error $\frac{1}{2}\,\phi^T(0)\Gamma\phi(0)$.

2) The value of c is now required for implementation of the parameter adaptation law (3.8).

3) $\dot{V}$ in (3.13) is now only negative semidefinite in the state since the full state vector in the adaptive case is augmented by ϕ.

It is noted that proporty 3 destroys the simple exponential stability argument used earlier in the nonadaptive case (cf., [1], Theorem 4-1) to insure asymptotic convergence of $||\Delta q_1||$ and $||\Delta q_2||$. In addition since the error system in $(\Delta q_1, \Delta q_2, \phi)$ is nonautonomous (and in general, nonperiodic), standard invariance principles are not applicable. Alternatively, we make use of a lemma due originally to Barbalat, quoted without proof from Popov [14] (pg. 211).

Lemma 3-1 (Barbalat)

If W is a real function of the real variable t, defined and uniformly continuous for $t \geq 0$ and if the limit of the integral

$$\lim_{t\to\infty} \int_0^t W(t')dt' ,$$

exists and is a finite number, then

$$\lim_{t\to\infty} W(t) = 0 . \qquad \blacksquare$$

For our purposes let,

$$W(t) \triangleq \lambda_1||\Delta q_1(t)||^2 + \lambda_2||\Delta q_2(t)||^2$$

so that

$$\dot{V} \leq - W \tag{3.17}$$

Integrating both sides of (3.17) from 0 to t, yields upon rearranging,

$$\int_0^t W\,dt' \leq V_o - V(t) \tag{3.18}$$

Since V_o is bounded, and V(t) is nonincreasing and bounded below, it follows that

$$\lim_{t\to\infty} \int_0^t W\,dt' < \infty$$

Also, since $\dot{W}$ is bounded, W(t) is uniformly continuous. Hence, application of Barbalat's Lemma gives,

$$\lim_{t\to\infty} W = 0 \tag{3.19}$$

or equivalently $||\Delta q_1||\to 0$ and $||\Delta q_2||\to 0$.

This completes the proof of asymptotic stability. The proof, however, is not a global one due to property 2, i.e., the value of c which was not required in the nonadaptive case now appears in the parameter adaptation law (3.8). Hence, one is committed to choosing a particular value of c in the adaptive implementation. Of course, c can always be chosen sufficiently small to satisfy the requirement, however, the position tracking performance determined by the magnitude of λ_1 in (3.14a) must be compromised as a result. Hence in practice, the initial choice of c can be made using whatever bounds on η_1, η_2, μ, $\underline{\sigma}(M)$ and V_o are available a-priori, and the value of c can be improved (increased) on-line as more information becomes available. It is noted that (3.16a) and (3.16b) impose additional constraints on how large c can become, since it is required that $\xi_1 > 0$ and $\xi_2 > 0$ for a positive definite V (these conditions can be shown sufficient).

The asymptotic stability proof presented above for adaptation of Control Law 1, is easily extended to adaptation of Control Laws 2, 3 and 4, since the corresponding nonadaptive Lyapunov function derivatives for these control laws are of exactly the same form as $\dot{V}^o$ in (3.25) (see [1], Theorem 4-1 for details). For convenience, all asymptotically stable adaptive control laws discussed thus far, and their appropriate parameter adaptation laws are summarized in Table II, corresponding to cases 1.a, 2.a, 3.a, and 4.a, respectively.

An alternative to choosing c sufficiently small in the above asymptotic stability argument is to choose K_v sufficiently large. In this case, the condition on c above can be removed completely by adapting K_v on-line. This modification insures global asymptotic stability of the adaptive control system (i.e., choice of c independent of the initial condition V_o) and is discussed in more detail below.

3.2.2 Global Asymptotic Stability-Adapting K_v

Since the velocity gain K_v enters linearly in the control law, it can be adapted as if it were an unknown parameter using the same formulation of Sec. 3.2.1. It will be shown that this approach removes the dependence of the choice of c on the initial condition V_o and this completes the proof of global asymptotic stability for the adaptive case.

Consider Control Law 1 written in adaptive form, where both θ and K_v are adapted in real time i.e.,

$$u = - K_p \Delta q_1 - \hat{K}_v \Delta q_2 + H_1 \hat{\theta} \tag{3.20}$$

Here, $\hat{K}_v$ is a time-varying quantity which remains to be specified, and H_1 is as defined earlier in (3.3). The nonadaptive control law u^o in (3.1) is subtracted from (3.20) to give the following expression,

$$u = u^o - \Delta K_v \Delta q_2 + H_1 \phi \tag{3.21}$$

where $\Delta K_v \overset{\Delta}{=} \hat{K}_v - K_v$ and $\phi = \hat{\theta} - \theta$.

The Lyapunov function for the stability analysis is given as

$$V = V^o + \frac{1}{2} \phi^T \Gamma \phi + \frac{1}{2} \delta \ TR\{\Delta K_v^T \Delta K_v\} \ , \quad \delta > 0 \ , \quad \Gamma = \Gamma^T > 0 \tag{3.22}$$

where a new term has been added relative to (3.6a), quadratic in the error ΔK_v. Taking the derivative of V along system trajectories and substituting control law (3.21) gives upon rearranging

$$\begin{aligned} \dot{V} = \dot{V}^o &+ (\Delta q_2 + c\Delta q_1)^T H_1 \phi + \dot{\phi}^T \Gamma \phi \\ &+ TR\{[\delta \Delta \dot{K}_v^T - \Delta q_2 (\Delta q_2 + c\Delta q_1)^T] \Delta K_v\} \end{aligned} \tag{3.23}$$

The latter terms are cancelled exactly by the choice of parameter adaptation laws,

$$\dot{\phi} = \dot{\hat{\theta}} = - \Gamma^{-1} H_1^T (\Delta q_2 + c\Delta q_1) \tag{3.24a}$$

$$\Delta \dot{K}_v = \dot{\hat{K}}_v = + \frac{1}{\delta} (\Delta q_2 + c\Delta q_1) \Delta q_2^T \tag{3.24b}$$

The choice leaves $\dot{V}$ exactly of the form (3.9) i.e., applying the β-Ball Lemma 2.1,

$$\dot{V} = \dot{V}^o \leq - \lambda_1 ||\Delta q_1||^2 - \lambda_2 ||\Delta q_2||^2 \tag{3.25}$$

if,

$$\sigma_{min}(K_p) > \frac{3}{4} \eta_2 \rho^2 \tag{3.26}$$

$$\sigma_{min}(K_v) > c(\mu + \frac{3}{4} \frac{\eta_2}{\rho^2} + \frac{\eta_1}{2} (\frac{V_o}{\xi_1})^{\frac{1}{2}}) \tag{3.27}$$

In (3.26) and (3.27), all quantities are defined exactly as in (3.12a) and (3.12b) respectively, except for V_o which is presently the initial value of V in (3.22). Furthermore, the values of ξ_1 and ξ_2 are once again given as

$$\xi_1 = \frac{1}{2} [\sigma_{min}(K_p + cK_v) - \ell^2 c \underline{\sigma}(M)] \tag{3.28}$$

$$\xi_2 = \frac{1}{2} (1 - \frac{c}{\ell^2}) \underline{\sigma} (M) \tag{3.29}$$

An important observation is that,

$$V_o \sim O(||K_v||^2) \ , \ ||K_v|| \to \infty \tag{3.30a}$$

$$\xi_1 \sim O(||K_v||) \ , \ ||K_v|| \to \infty \tag{3.30b}$$

Hence, for any choice of $\delta>0$, $c>0$ and $K_p=K_p^T>0$, there exist values of ρ^2, ℓ^2, and $K_v=K_v^T>0$ (with $\sigma_{min}(K_v)$ sufficiently large) such that inequalities (3.26) and (3.27) are satisfied, and $\xi_1>0$, $\xi_2>0$ in (3.28) and (3.29), respectively. Global asymptotic stability of this adaptive control scheme then follows immediately by application of Barbalat's Lemma to the Lyapunov function derivative (3.25), as was done earlier in equations (3.17) through (3.19).

The global asymptotic stability of adaptive controllers based on Control Laws 2, 3 and 4 (where K_v is adapted on-line) follow from an identical argument, since $\dot{V}^o$ corresponding to the nonadaptive Lyapunov function derivatives for these control laws are of exactly the same form as $\dot{V}^o$ in this analysis (see [1], Theorem 4-1 for details). For convenience, these adaptive control laws involving adaptation of K_v are summarized in Table II, corresponding to cases 1.b, 2.b, 3.b, and 4.b, respectively.

3.3 Global Asymptotic Stability for Adaptation of Control Laws 5, 6 and 7

Global asymptotic stability for adaptation of Control Laws 5, 6 and 7 can be proved using exactly the same techniques as applied in Sec. 3.2. The only difference lies in slight variations in the nonadaptive Lyapunov function derivative V^o which arises in each adaptive control analysis

Due to space limitations, these proofs have been omitted, but the results are summarized in Table II corresponding to cases 5.a, 5.b, 6.a, 6.b, and 7.a, 7.b, respectively. Details can be found in [21], to which the equation numbers in Table II are referenced.

4. Adaptive Computed Torque Method

It was mentioned earlier that in the computed torque method (i.e., control law 8) the presence of the $M(q_1)$ term premultiplying the K_p and K_v gains complicates the Lyapunov analysis and hinders most simple attempts to make it adaptive. Nevertheless, the computed torque controller is a well-known control law in the literature and is widely applied in practice. Hence, it is useful to investigate under what conditions it can be made adaptive, and to what extent adaptive stability can be guaranteed. For this purpose, we consider a special case of the computed torque control law which has scalar gains k_p and k_v, i.e.,

$$u^o = -M(q_1)(k_p\Delta q_1 + k_v\Delta q_2) + k(q_1) + M(q_1)\dot{q}_{2d} + C(q_1,q_2)q_2 \tag{4.1}$$

This is written in adaptive form as,

$$u = u^o + H_8\phi \tag{4.2}$$

where $\phi = \hat{\theta} - \theta$, and the linear in the parameters part has been chosen as,

$$H_8\theta \overset{\Delta}{=} -M(q_1)(k_p\Delta q_1 + k_v\Delta q_2) + k(q_1) + M(q_1)\dot{q}_{2d} + C(q_1,q_2)q_2 \tag{4.3}$$

Let a Lyapunov function be defined as,

$$V = V^o + \frac{1}{2}\phi^T\Gamma\phi \quad , \quad \Gamma = \Gamma^T > 0 \tag{4.4a}$$

where

$$V = \frac{1}{2}\Delta q_2{}^T M(q_1)\Delta q_2 + \frac{1}{2}(k_p + ck_v)\Delta q_1{}^T M(q_1)\Delta q_1 + c\Delta q_1{}^T M(q_1)\Delta q_2 \tag{4.4b}$$

Then, the derivative of (4.4) along system trajectories induced by control (4.2) is given by

$$\dot{V} = \Delta q_2{}^T [-k_v M(q_1)\Delta q_2 + \frac{1}{2} M_D(q_1,\Delta q_2)q_2]$$

$$+ c\Delta q_1{}^T [-k_p M(q_1)\Delta q_1 + M_D(q_1,\Delta q_2)q_2 + M_D(q_1,\Delta q_1)q_2] \tag{4.5}$$

$$+ (\Delta q_2 + c\Delta q_1)^T H_8\phi + \dot{\phi}^T\Gamma\phi \tag{4.5}$$

Let,

$$\dot{\phi} = \dot{\hat{\theta}} = -\Gamma^{-1} H_8{}^T(\Delta q_2 + c\Delta q_1) \tag{4.6}$$

Then,

$$\dot{V} \leq -\alpha_1||\Delta q_1||^2 - \alpha_2||\Delta q_2||^2 + \gamma_{12}||\Delta q_2||\ ||\Delta q_1||^2$$

$$+ (\gamma_{21}||\Delta q_1|| + \gamma_{22}\ ||\Delta q_2||)||\Delta q_2||^2 \tag{4.7}$$

where

$$\alpha_1 = c(k_p\,\underline{\sigma}(M) - \eta_2(1 + \rho^2)) \tag{4.8}$$

$$\alpha_2 = k_v\underline{\sigma}(M) - \frac{1}{2}\eta_1 - \frac{c\eta_2}{\rho^2} \tag{4.9}$$

$$\gamma_{12} = c\eta_1 \tag{4.10}$$

$$\gamma_{21} = c\eta_1 \quad ; \quad \gamma_{22} = \frac{1}{2}\eta_1 \tag{4.11}$$

Applying the β-Ball Lemma, it follows that,

$$\dot{V} \leq - \lambda_1 ||\Delta q_1||^2 - \lambda_2 ||\Delta q_2||^2 \tag{4.12}$$

if,

$$ck_p \underline{\sigma}(M) > \eta_2(1 + \rho^2)) + c\ \eta_2 (\frac{V_o}{\xi_1})^{\frac{1}{2}} \tag{4.13}$$

$$k_v \underline{\sigma}(M) > \frac{1}{2} \eta_1 + \frac{c\eta_2}{\rho^2} + \frac{1}{2} \eta_1 \ (\frac{V_o}{\xi_2})^{\frac{1}{2}} + c\ \eta_1 (\frac{V_o}{\xi_1})^{\frac{1}{2}} \tag{4.14}$$

$$\xi_1 = \frac{1}{2} [k_p + ck_v - c\ell^2] \underline{\sigma}(M), \tag{4.15}$$

$$\xi_2 = \frac{1}{2} (1 - \frac{c}{\ell^2}) \underline{\sigma}(M) \quad ; \ell^2 > 0 \text{ arbitrary} \tag{4.16}$$

It is noted that for any $c > 0$, both k_p and k_v can always be chosen sufficiently large so that $\xi_1 > 0$ and $\xi_2 > 0$ (for appropriate choice of $\ell^2 > 0$ in (4.15), (4.16)), and inequalities (4.13) and (4.14) are satisfied. Hence, the adaptive computed torque control law given by (4.2), (4.3) with parameter adaptation (4.6) is asymptotically stable when k_p and k_v are chosen sufficiently large.

Since k_p and k_v must be chosen sufficiently large with respect to the initial condition V_0 (c.f., (4.13), (4.14)) this proof of asymptotic stability is not global (i.e., for fixed k_p and k_v there will always exist some V_0 such that λ_1 and/or λ_2 are not positive). For this particular algorithm, it is presently not clear how to adapt k_p and k_v to insure global asymptotic stability since the control u in (4.1) is not linear in the parameters (θ, k_p, k_v).

5. Summary and Remarks

The adaptive control laws derived herein, along with the sufficient conditions for stability and appropriate parameter adaptation laws are summarized in Table II. Several remarks are in order at this point in the discussion.

Remark 5-1 All adaptive control laws in this paper were derived for the general tracking control law. However, significant simplification occurs in many of these designs for the special case of set-point control (i.e., $q_{2d} = \dot{q}_{2d} = 0$).

Remark 5-2 The adaptive robustness issue remains open. Certainly for parameter adaptation laws of the form given in Table II, there will be sensitivities to noise disturbances and unmodelled dynamics directly analogous to those which arise in the linear adaptive control case. It presently appears that many of the robustness techniques developed in the linear adaptive control literature will carry over to the nonlinear adaptive control application. This conjecture, however, remains to be investigated.

Remark 5-3 In the nonadaptive case, many of the control laws in Table II are in a form appropriate for application of the recursive Newton-Euler computational algorithm. However, the Newton-Euler algorithm requires knowledge of all physical parameters-more parameters than are actually needed to control the system adaptively and more than are actually adapted on-line in the vector θ of Table II. Hence, the transformation from θ back to physical parameters is required in order to salvage use of the Newton-Euler algorithm in the adaptive case. However, the transformation is generally nonlinear and will not lead to a unique solution unless further constraints are imposed. One typical set of constraints arises when only the payload mass is unknown. In the more general adaptive case, it is useful to note that all linear in the parameters expressions can be implemented directly, since representations of the form $H\theta$ are assumed to be available in symbolic form.

Remark 5-4 The control laws of Table I were derived in [1] for the general desired potential energy function. This feature was dropped in the adaptive case in order to simplify the analysis. However, it appears that the adaptive control laws developed herein can be extended to the more general case and this line of research presently under investigation.

Remark 5-5 A brief comparison with the recent results Paden [10] and Slotine and Li [11] is useful. In [10] [11], adaptive control laws are derived by choosing u to cancel various terms in the Lyapunov function derivative, rather than overbounding them (via Lemma 2.1) as was done here. This approach has the advantage of providing global asymptotic convergence without adapting gains K_v and K_p. The control laws, however, are by necessity more complex than those designs considered here, and do not simplify in the set-point control case.

6. Conclusions

A new class of asymptotically stable adaptive control laws is defined by adapting the control laws of [1] in a certainty equivalence fashion. These algorithms are proved to be asymptotically stable without approximations, linearizations or ad-hoc assumptions concerning the nonlinear manipulator dynamics. Furthermore, the asymptotic convergence properties can be made global by appropriate adaptation of feedback gains. On-going research efforts are directed at adaptive robustness, computation, and obstacle avoidance problems.

TABLE II SUMMARY OF ASYMPTOTICALLY STABLE ADAPTIVE CONTROL LAWS

CASE #	ADAPTIVE CONTROL	LINEAR IN PARAMETERS EXPRESSION	PARAMETER ADAPTATION	STABILITY CONDITIONS†	REMARKS
1.a	$u = - K_p\Delta q_1 - K_v\Delta q_2 + H_1\hat{\theta}$	$H_1\theta = k(q_1)+M(q_1)\dot{q}_{2d} - \frac{1}{2} J(q_1,q_2)q_{2d} + \frac{1}{2} M_D(q_1,q_{2d})q_2$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_1^T(\Delta q_2+c\Delta q_2)$	$\sigma_{min}(K\ )$ sufficiently large or c sufficiently small (see (3.12))	*Asymptotic Stability
1.b	$u = - K_p\Delta q_1 - \hat{K}_v\Delta q_2 + H_1\hat{\theta}$	$H_1\theta$ = same as 1.a above	$\dot{\hat{\theta}} = - \Gamma^{-1} H_1^T(\Delta q_2+c\Delta q_1)$ $\dot{\hat{K}}_v = \frac{1}{\delta} (\Delta q_2+c\Delta q_1)\Delta q_2^T$	None rèquired	*Global Asymptotic Stability
2.a	$u = - K_p\Delta q_1 - K_v\Delta q_2 + H_2\hat{\theta}$	$H_2\theta=k(q_1)+M(q_1)\dot{q}_{2d} - \frac{1}{2} J(q_1,q_{2d})q_2 + \frac{1}{2} M_D(q_1,q_2)q_{2d}$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_2^T (\Delta q_2+c\Delta q_1)$	$\sigma_{min}(K_v)$ sufficiently large or c sufficiently small	*Asymptotic Stability
2.b	$u = - K_p\Delta q_1 - \hat{K}_v\Delta q_2 + H_2\hat{\theta}$	$H_2\theta$ = same as 2.a above	$\dot{\hat{\theta}} = - \Gamma^{-1} H_2^T (\Delta q_2+c\Delta q_1)$ $\dot{\hat{K}}_v = \frac{1}{\delta} (\Delta q_2+c\Delta q_1)\Delta q_2^T$	None required	*Global Asymptotic Stability
3.a	$u = - K_p\Delta q_1 - K_v\Delta q_2 + H_3\hat{\theta}$	$H_2\theta=k(q_1)+M(q_1)\dot{q}_{2d} - \frac{1}{2} J(q_1,q_{2d})q_{2d} + \frac{1}{2} M_D(q_1,q_2)q_{2d}$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_3^T (\Delta q_2+c\Delta q_1)$	$\sigma_{min}(K_v)$ sufficiently large or c sufficiently small	*Asymptotic Stability
3.b	$u = - K_p\Delta q_1 - \hat{K}_v\Delta q_2 + H_3\hat{\theta}$	$H_3\theta \hat{=}$ same as 3.a above	$\dot{\hat{\theta}} = - \Gamma^{-1} H_3^T (\Delta q_2+c\Delta q_1)$ $\dot{\hat{K}}_v = \frac{1}{\delta} (\Delta q_2+c\Delta q_1)\Delta q_2^T$	None required	*Global Asymptotic Stability
4.a	$u = - K_p\Delta q_1 - K_v\Delta q_2 + H_4\hat{\theta}$	$H_4\theta=k(q_1)+M(q_1)\dot{q}_{2d} - \frac{1}{2} J(q_1,q_2)q_2 + \frac{1}{2} M_D(q_1,q_{2d})q_2$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_4^T (\Delta q_2+c\Delta q_1)$	$\sigma_{min}(K_v)$ sufficiently large or c sufficiently small	Asymptotic Stability

4.b	$u = - K_p\Delta q_1 - K\,\Delta q_2 + H_4\hat{\theta}$	$H_4\theta \overset{\Delta}{=}$ same as 4.a above	$\dot{\hat{\theta}} = - \Gamma^T H_4^T(\Delta q_2+c\Delta q_1)$ $\dot{\hat{K}}_v = \frac{1}{\delta}(\Delta q_2+c\Delta q_1)\Delta q_2^T$	None required	Global Asymptotic Stability
5.a	$u = - K_p\Delta q_1 - K_v\Delta q_2 + H_5\hat{\theta}$	$H_5 = K(q_1)+M(q_1)\dot{q}_{2d} + C(q_1,q_{2d})q_{2d}$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_5^T(\Delta q_2+c\Delta q_1)$	$\sigma_{min}(K_v)$ sufficiently large (see (3.36), (3.37))	*†Asymptotic Stability
5.b	$u = - K_p\Delta q_1 - \hat{K}_v\Delta q_2 + H_5\hat{\theta}$	$H_5\theta$ - same as 3.a above	$\dot{\hat{\theta}} = - \Gamma^{-1} H_5^T(\Delta q_2+c\Delta q_1)$ $\dot{\hat{K}}_v = \frac{1}{\delta}(\Delta q_2+c\Delta q_1)\Delta q_2^T$	None required	*†Global Asymptotic Stability
6.a	$u = - K_p\Delta q_1 - K_v\Delta q_2 + H_6\hat{\theta}$	$H_6\theta = k(q_1)+M(q_1)\dot{q}_{2d} + C(q_1,q_2)q_2$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_6^T(\Delta q_2+c\Delta q_1)$	$\sigma_{min}(K_v)$ sufficiently large (see (3.45), (3.46))	†Asymptotic Stability
6.b	$u = - K_p\Delta q - \hat{K}_v\Delta q_2 + H_6\hat{\theta}$	$H_6\theta$ = same as 4.a above	$\dot{\hat{\theta}} = - \Gamma^{-1} H_6^T(\Delta q_2+c\Delta q_1)$ $\dot{\hat{K}}_v = \frac{1}{\delta}(\Delta q_2+c\Delta q_1)\Delta q_2^T$	δ sufficiently small (see (3.55), (3.56))	†Global Asymptotic Stability
7.a	$u = - K_p\Delta q_1 - K_v\Delta q_2 + H_7\hat{\theta}$	$H_7\theta = k(q_{1d})+M(q_{1d})\dot{q}_{2d} + C(q_{1d},q_{2d})q_{2d}$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_7^T(\Delta q_2+c\Delta q_1)$	$\sigma_{min}(K_p)$ and $\sigma_{min}(K_v)$ sufficiently large w.r.t. initial cond. (see (3.55), (3.56))	*†Asymptotic Stability
7.b	$u = - \hat{K}_p\Delta q_1 - \hat{K}_v\Delta q_2 + H_7\hat{\theta}$	$H_7\theta$ = same as 5.a above	$\dot{\hat{\theta}} = - \Gamma^{-1} H_7^T(\Delta q_2+c\Delta q_1)$ $\dot{\hat{K}}_v = \frac{1}{\delta_v}(\Delta q_2+c\Delta q_1)\Delta q_2^T,\ \delta_v>0$ $\dot{\hat{K}}_p = \frac{1}{\delta_p}(\Delta q_2+c\Delta q_1)\Delta q_1^T,\ \delta_p>0$	None required	*†Global Asymptotic Stability
8.	$u = H_8\hat{\theta}$	$H_8\theta \overset{\Delta}{=} -M(q_1)(k_p\Delta q_1+k_v\Delta q_2)+k(q_1)+M(q_1)\dot{q}_{2d}+C(q_1,q_2)q_2$ $k_p>0,\ k_v>0$	$\dot{\hat{\theta}} = - \Gamma^{-1} H_8^T(\Delta q_2+c\Delta q_1)$	k_p and K_v sufficiently large w.r.t. Initial Condition (see (4.13), (4.14))	†Computed Torque Method with scalar gains - Asymptotic Stability

ǂ General Assumptions:

$K_v=K_v^T>0,\ K_p=K_p^T>0,\ \Gamma=\Gamma^T>0,\ c>0,\ \delta>0,\ \delta_p>0,\ \delta_v>0$

*Significant simplification for set-point control
$\dot{q}_{2d} = q_{2d} = 0$

†Recursive Newton Euler applicable in nonadaptive case

Acknowledgements

The authors are indebted to Dr. K. Kreutz for many helpful technical discussions. This work was done at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration.

References

[1] J.T. Wen and D.S. Bayard, "Simple Robust Control Laws for Robotic Manipulators - Part I: Nonadaptive Case," this Workshop.

[2] S. Dubowsky and D.T. Desforges, "The Application of Model-Referenced Adaptive Control to Robotic Manipulators," Trans. ASME J. Dynamic Systems, Measurement and Control, Vol. 101, 1979, pp. 193-200.

[3] A.J. Koivo and T.-H. Guo, "Adaptive Linuar Controller for Robotic Manipulators," IEEE Trans. Auto. Contr., Vol. AC-28, No. 2, 1983, pp. 162-171.

[4] C.S.G. Lee and M.J. Chung, "An Adaptive Control Strategy for Mechanical Manipulators," IEEE Trans. Auto. Contr., Vol. AC-29, No. 9, 1984, pp. 837-840.

[5] M. Tomizuka and R. Horowitz, "Model Reference Adaptive Control of Mechanical Manipulators," IFAC Workshop on Adaptive Systems in Control and Signal Processing, San Francisco, CA, 1983.

[6] M. Takegaki and S. Arimoto, "An Adaptive Trajectory Control of Manipulators," Int. J. Contr., Vol. 34, No. 2, 1981, pp. 219-230.

[7] K.Y. Lim and M. Eslami, "Adaptive Controller Designs for Robotic Manipulator Systems Using Lyapunov Direct Method," IEEE Trans. Auto. Contr., Vol.-AC-30, No. 12, 1985, pp. 1229-1233.

[8] T.C. Hsia, "Adaptive Control of Robot Manipulators - A Review," IEEE Int. Conf. Robotics and Automation, San Francisco, CA, 1986.

[9] J. Craig, P. Hsu, and S. Sastry, "Adaptive Control of Mechanical Manipulators," IEEE Conf. Robotics and Automation, San Francisco, CA, April 1986.

[10] B. Paden, "PD+ Robot Controllers: Tracking and Adaptive Control," Private Communication.

[11] J.-J.E. Slotine and W. Li, "On the Adaptive Control of Robot Manipulators," ASME Winter Meeting, Anaheim, CA, 1986.

[12] I.D. Landau, "A Survey of Model Reference Adaptive Techniques - Theory and Applications," Automatica, Vol. 11, 1974, pp. 353-379.

[13] K.S. Narendra and R.V. Monopoli, Eds., Applications of Adaptive Control, Academic Press, New York, 1980.

[14] V.M. Popov, Hyperstability of Control Systems, Springer-Verlag, New York, 1973.

[15] P.A. Ioannou and P.V. Kokotovic, "Robust Redesign of Adaptive Control," IEEE Trans. Auto. Contr., Vol. AC-29, No. 3, 1984, pp. 202-211.

[16] P.A. Ioannou, "Robust Adaptive Controller with Zero Residual Errors," IEEE Trans. Auto. Contr., Vol. AC-31, No. 8, 1986, pp. 773-776.

[17] D.S. Bayard, C.H.C. Ih and S.J. Wang, "Adaptive Control for Flexible Space Structures with Measurement Noise," submitted to American Control Conference, Minneapolis, Minnesota, June 10-12, 1987.

[18] C.G. Atkinson, C.G. An and J.M. Hollerbach, "Estimation of Initial Parameters of Manipulator Loads and Links," International Sympcsium on Robotics Research, 1985.

[19] P. Khosla and T. Kanade, "Parameter Identification of Robot Dynamics," IEEE Conf. Decision and Control, Fort Lauderdale, Florida, 1985.

[20] J.T. Wen and D.S. Bayard, Robust Control for Robotic Manipulators, Part I: Non-Adaptive Case,(JPL Internal Document, Engineering Memorandum, EM 347-87-203). Jet Propulsion Laboratory, Pasadena, California, 1987.

[21] D.S. Bayard and J.T. Wen, Robust Control for Robotic Manipulators, Part II: Adaptive Case,(JPL Internal Document, Engineering Memorandum, EM 347-87-204). Jet Propulsion Laboratory, Pasadena, California, 1987.

Algorithms for Adaptive Control of Two-Arm Flexible Manipulators Under Uncertainty

J.M. Skowronski
University of Queensland
St. Lucia, Australia 4067

1. Abstract

is used

~~The paper uses~~ a nonlinear extension of model reference adaptive control (MRAC) technique to guide a double arm nonlinearizable robot manipulator with flexible links, driven by actuators collocated with joints subject to uncertain payload and inertia. The objective is to track a given simple linear and rigid but compatible dynamical model in real, possibly stipulated time and within stipulated degree of accuracy of convergence while avoiding collision of the arms. The objective is attained by a specified signal adaptive feedback controller and by adaptive laws, both given in closed form. A case of 4 DOF manipulator illustrates the technique.

2. Introduction

The MRAC technique becomes popular proposition for guidance of recent robot manipulators, with demand for precision pointing in difficult conditions, under the action of full scale dynamic forces, and subject to uncertainty in parameters. Such manipulators, particularly these used on spacecraft are highly nonlinear and nonlinearizable structures (geometric nonlinearity of elastic links, large angle articulation, nonlinear coupling of DOF's, nonignorable gyro and Coriolis forces, several equilibria), while classical MRAC is linear and applicable to rigid bodies only. Thus the extension is needed for handling nonlinearity, see [1], and flexible links, see [2]. On the other hand many robotic objectives, again particularly these in difficult space conditions require at least two arm systems. Thus the tracking has to be a double MRAC (mutual reference adaptive control) which secures tracking the same model by two arms while avoiding mutual collision - cf. [3], [4]. If adaptive (self-organizing) control is intended, the tracking relates not to a given path but to a given dynamic target-model with prescribed target-parameters. We take the model simple thus rigid and linear, but locally compatible with the nonlinear arms regarding equilibria. Each arm is represented as an open chain with n DOF, nonlinear characteristics and coupling, elastic links, driven by actuators collocated with joints, under uncertain inertia parameters and uncertain payload. The tracking is done in real possibly stipulated time by a designed signal adaptive feedback controller and integrable adaptive laws in the state space, while avoiding collision between arms of all the joints (and elastic nodes) in Cartesian configuration space.

3. Motion Equations

Lagrange motion equations give the rigid dynamics of the arms in the general format

$$A^j(q^j,s^j)\ddot{q}^j + \Gamma^j(q^j,\dot{q}^j,\lambda^j) + \Pi^j(q^j,\lambda^j,s^j) = B^j(q^j,\dot{q}^j)u^j \ , \quad j = 1,2 \ , \tag{1}$$

where $q^j(t) \in \Delta_q \subset \mathbf{R}^n$, $t \geq t_o = 0$, is the configuration vector of the joint variables $q_1^j,\ldots,q_n^j$ of the j-th arm varying in the known bounded work region Δ_q of the configuration space $\mathbf{R}^n$; $\dot{q}(t)$ is the corresponding vector of joint velocities in the specified bounded subset $\Delta_{\dot{q}}$ of the space tangent to $\mathbf{R}^n$; $u^j(t) \in U \subset \mathbf{R}^n$ are the control vectors in given compact set of constraints U; $\lambda^j(t) \in \Lambda \subset \mathbf{R}^\ell$, $\ell \leq 2n$, are the vectors of adjustable system parameters in bounded bands of values Λ, and $s^j(t) \in S \subset \mathbf{R}^k$ is an uncertainty parameter within the known band S. Moreover $A^j(q^j,s^j)$ are the inertia n×n matrices obtained in the known way from the quadratic form of kinetic energy. The vectors $\Pi^j = (\Pi_1^j,\ldots,\Pi_n^j)^T$ represent potential forces (gravity, spring) while $\Gamma^j = (\Gamma_1^j,\ldots,\Gamma_n^j)^T$ represent the internal nonpotential acting forces (Coriolis, gyro, centrifugal, damping structural or viscous, etc.) and B^j is the actuator transmission (gear) nonsingular n×n matrix. The control vectors $u^j(t)$ are selected for the objectives of tracking and avoidance by adaptive feedback control programs $u^j(t) = p^j(q^1(t),q^2(t),\dot{q}^1(t),\dot{q}^2(t),\lambda^1(t),\lambda^2(t))$ on corresponding products of $\Delta_q \times \Delta_{\dot{q}} \times \Lambda$. For convenience the superscripts "j" will be dropped until they are needed to avoid ambiguity.

Considering the links elastic we introduce the deformation coordinates for the i-th link as shown in Fig. 1, while using the Ritz-Kantorovitch series expansion

$$r_i(y_i,t) = \sum_{\nu=1}^{m} r_i^{\nu}(y_i)r_i^{\nu}(t) = r_i(y_i)r_i(t) \tag{2}$$

and for $v_i(y_i,t)$, $w_i(y_i,t)$ analogously, with the exact solution expected for $m \to \infty$. We take m large

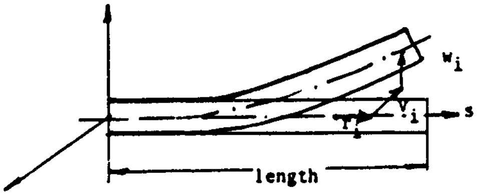


Figure 1. Flexible link

enough so that the Kantorovitch linearization is physically justified. The technical way about it is to stepwise subdividing the links between grid as long as the difference of results for successive m's becomes small. Having specified (2) we form the vector $\eta(t) \overset{\Delta}{=} (\eta_1(t),\dots,\eta_n(t))^T$, where $\eta_i(t) \overset{\Delta}{=} (r_i(t),v_i(t),w_i(t))^T$ and following [5] write the hybrid system as

$$\begin{pmatrix} A & A_c \\ A_c^T & A \end{pmatrix}\begin{pmatrix} \ddot{q} \\ \ddot{\eta} \end{pmatrix} + \begin{pmatrix} 0 & D_c \\ 0 & D \end{pmatrix}\begin{pmatrix} \dot{q} \\ \dot{\eta} \end{pmatrix} + \begin{pmatrix} 0 & P_c \\ 0 & P \end{pmatrix}\begin{pmatrix} q \\ \eta \end{pmatrix} + \begin{pmatrix} \Gamma(q,\dot{q}) \\ \Gamma_\eta(\eta,\dot{\eta}) \end{pmatrix} + \begin{pmatrix} \Pi(q,\lambda,s) \\ \Pi_\eta(\eta,s) \end{pmatrix} = \begin{pmatrix} B(q,\dot{q}) \\ 0 \end{pmatrix}u \tag{3}$$

where $A_\eta(\eta,s)$, $\Gamma_\eta(\eta,\dot{\eta})$, $\Pi_\eta(\eta,s)$ are the elastic correspondents of A, Γ, Π while $A_c(q,\eta)$, $D_c(q,\dot{q},\eta,\dot{\eta})$, $P_c(q,\eta)$ and the internal damping $D(q,q,\eta,\eta)$ as well as the hybrid restoring coefficients $P(q,\eta)$ are matrices coupling the elastic and joint coordinates. These matrices are formed by integrals over the shape functions, see [5]. Letting

$$A(q,\eta,s) = \begin{pmatrix} A & A_c \\ A_c^T & A_\eta \end{pmatrix}$$

to be the hybrid inertia matrix which is nonsingular positive definite,we inertially decouple (3):

$$(\ddot{q},\ddot{\eta})^T + \mathcal{D}(q,\dot{q},\eta,\dot{\eta},\lambda,s) + \mathcal{P}(q,\eta,\lambda,s) = \mathcal{B}(q,\dot{q},s)u \tag{4}$$

where $\mathcal{D} \overset{\Delta}{=} A^{-1}(D_c\dot{\eta}+\Gamma,D\dot{\eta}+\Gamma_\eta)^T$ and $\mathcal{P} \overset{\Delta}{=} A^{-1}(P_c\eta+\Pi,P\eta+\Pi_\eta)^T$ are successively vectors of nonpotential and potential forces and the meaning of the matrix $\mathcal{B}$ is obvious. The vectors q, $\dot{q}$, η, $\dot{\eta}$ form the state vector $x(t) = (x_1(t),\dots,x_N(t))^T \overset{\Delta}{=} (q(t),\eta(t),\dot{q}(t),\dot{\eta}(t))^T \in \Delta_q \times \Delta_\eta \times \Delta_{\dot{q}} \times \Delta_{\dot{\eta}} \overset{\Delta}{=} \Delta \subset \mathbf{R}^N$, $N = 4n$, for each arm. For convenience (4) may be then written in the general state form

$$\dot{x} = f(x,u,\lambda,s) \tag{5}$$

with $f = (f_1,\dots,f_N)$ of the shape specified by (4) in an obvious way. Formally (5) may be written in the contingent form:

$$\dot{x} \in \{f(x,u,\lambda,s) \mid s \in S\} \tag{5'}$$

which for suitable $f(\cdot)$, $p(\cdot)$, $\lambda(\cdot)$ has solutions $x(t) = k(x^0,t)$, $t \geq 0$, absolutely continuous curves through each $x^0 = x(0)$ in Δ . We shall consider the class of such solutions $K(x^0)$ by exhausting all values of $s(t)$ in (5) at each t .

4. The Reference Model

We let the given Cartesian "world" coordinates representation of the reference model in general terms

$$\dot{\xi}_m = F(\lambda_m)\dot{\xi} \tag{6}$$

with 2n DOF, $\xi(t) \in \mathbf{R}^{3.2n}$, and $F(\lambda_m)$ suitable matrix, be off-line recalculated to the joint coordinate format of the rigid linear system

$$\ddot{q}_m + \mathcal{D}_m(\lambda_m)\dot{q}_m + \mathcal{P}_m(\lambda_m)q_m = 0 \tag{7}$$

with the 2n-vectors q_m, $\dot{q}_m$ of joint coordinates and velocities, state $x_m(t) = (q_m(t),\dot{q}_m(t))^T \in \Lambda \subset \mathbf{R}^N$, and

$\mathcal{D}_m$, $\mathcal{P}_m$ suitable matrices, while $\lambda_m = (\lambda_{m1},\ldots,\lambda_{m\ell}) = \text{const} \in \Lambda \subset R^\ell$, $\ell = n$. Moreover

$$\mathcal{P}_m(\lambda_m)(q^e,\eta^e) = 0 \tag{8}$$

with (q^e,η^e) denoting the equilibria of (3) on the surface $\dot{q} = 0$, $\dot{\eta} = 0$. The total energy of the model will be denoted by $E_m(\xi_m,\dot{\xi}_m)$ in the world coordinates and $E_m(q_m,\dot{q}_m)$ in the joint coordinates, obviously equal to one another. Then

$$E_m(q_m,\dot{q}_m) = \tfrac{1}{2}\dot{q}_m^T\dot{q}_m + \int_{q_m^o}^{q_m} \mathcal{P}_m(\lambda_m)\, d\sigma \tag{9}$$

and substituting (7),

$$\dot{E}_m(q_m,\dot{q}_m) = -\mathcal{D}_m(\lambda_m)(\dot{q}_m)^2 . \tag{10}$$

The model is selected such as to allow achieving of a stipulated target behavior in the state space. To focus attention on something specific and yet general enough, let it be stability of the origin, guaranteed by the nonaccummulation of the total energy i.e. non-negative damping

$$\dot{E}_m(q_m,\dot{q}_m) \le 0 , \quad \forall \dot{q}_m \ne 0 \tag{11}$$

while

$$\nabla E_m(q_m,\dot{q}_m) > 0 \tag{12}$$

in-the-large i.e. on same $C\Delta_L = \Delta - \Delta_L$, where Δ_L is the set in R^N enclosing all the equilibria.

5. Objectives

Now we consider both arms $j = 1,2$ and the model together. The block scheme of the system is shown in Fig. 2.

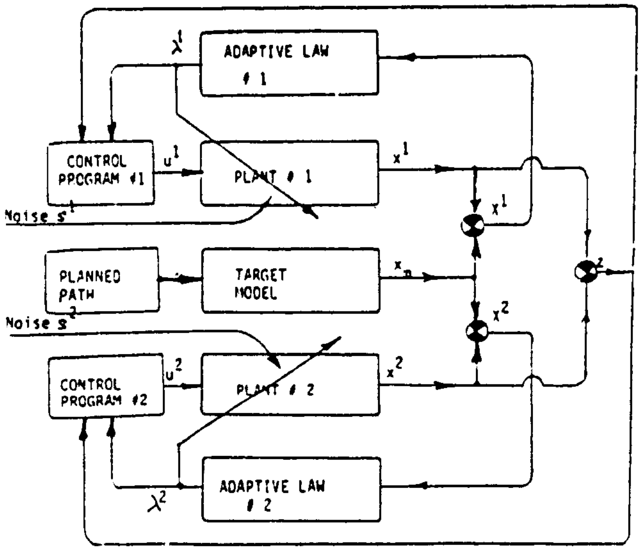


Figure 2. Block scheme of the system

Define two product 2N-vectors $X^j(t) = (x^j(t),x_m(t))^T \in \Delta \times \Delta \triangleq \Delta^2$ and two ℓ-vectors $\alpha^j(t) = \lambda^j(t) - \lambda_m$, which vary in $\Delta^2\times\Lambda$ generating the product trajectories $(X^j(X^{jo},t),\ \alpha^j(\alpha^{jo},t))$, $t \ge 0$, $X^{jo} = X^j(0)$, $\alpha^{jo} = \alpha^j(0)$. Then we define the "diagonal" sets

$$M^j = \{(X^j,\alpha^j) \in \Delta^2\times\Lambda \mid x^j = x_m,\ \alpha^j = 0\} , \quad j = 1,2 ,$$

and given stipulated $\mu^j > 0$, their neighbourhoods

$$M^j = \{(X^j,\alpha^j) \in \Delta^2\times\Lambda \mid |x^j - x_m| < \mu^j,\ |\alpha^j| < \mu^j\} , \quad j = 2 .$$

Moreover we let Δ_o be a desired subset of Δ where we want the tracking to occur, and let t_c be the stipulated time after which the tracking is attained with accuracy μ^j .

First Objective: The manipulator arms (1) are mutually μ-tracking the target (7) on Δ_o if there is a pair of controllers $p^j(\cdot)$, $j = 1,2$ such that for each solution $k^j(x^{jo},t)$, $t \geq 0$ of (4) in $K(x^{jo})$, the set $\Delta_o^2 \times \Lambda$ is positively invariant: $(X^{jo},\alpha^{jo}) \in \Delta_o^2 \times \Lambda \Rightarrow (X^j(t),\alpha^j(t)) \in \Delta_o^2 \times \Lambda$ and given t_c, for each $k^j(\cdot) \in K(x^{jo})$ the product trajectories satisfy

$$(X^j(t),\alpha^j(t)) \in M_\mu^j \ , \ \forall t \geq t_c \ . \qquad (13)$$

The convergence is illustrated in Fig. 3.

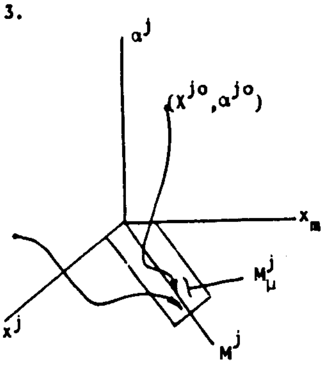


Figure 3. Convergence of product trajectories

Suppose the transformation from joint to world coordinates (forward kinematics) is given by

$$\xi_\sigma^j = \xi_\sigma^j(q^j,\eta^j) \ , \quad \sigma = 1,\ldots,3\cdot 2n \qquad (14)$$

and denote $Z(t) \overset{\Delta}{=} (x^1(t),x^2(t))$. Then we let the set

$$A \overset{\Delta}{=} \{Z \in \Delta^2 \ | \ |\xi_\sigma^1 - \xi_\nu^2| \leq d, \ \forall \sigma,\nu = 1,\ldots,3\cdot 2n\}$$

be the collision set between arms to be avoided. We define $CA \overset{\Delta}{=} \Delta_o^2 - A$, specified by $|\xi_\sigma^1 - \xi_1^2| > d$, and let

$$\Delta_A \overset{\Delta}{=} \{Z \in \Delta^2 \ | \ d < |\xi_\sigma^1 - \xi_\sigma^2| < \varepsilon\}$$

be the "slow down" safety zone, with $\varepsilon > 0$ suitable constant.

Second Objective: The tracking arms (1) avoid collision iff there is Δ_A such that for any $Z^o \in CA$, and any pair $k^j(\cdot) \in K^j(x^{jo})$ the corresponding product trajectory

$$Z(Z^o,t) \in CA \ , \ \ \forall t \geq 0. \qquad (15)$$

6. Sufficient Conditions

We return now to the first objective and specify by $N[\partial(\Delta_o^2 \times \Lambda)]$ a neighborhood of the boundary $\partial(\Delta_o^2 \times \Lambda)$ of the region $\Delta_o^2 \times \Lambda$. Then let $N_\varepsilon \overset{\Delta}{=} [\partial(\Delta_o^2 \times \Lambda) \cap \overline{\Delta_o \times \Lambda}]$, $CM_\mu^j \overset{\Delta}{=} (\Delta_o^2 \times \Lambda) - M_\mu^j$ and introduce open $D^j \supset \overline{CM}_\mu^j$ such that $D^j \cap M^j = \phi$. Further we consider four C^1-functions $V_s^j(\cdot)$: $\bar{N}_\varepsilon \to R$, $V_\mu^j(\cdot)$: $D^j \to R$, $j = 1,2$ with the positive constants

$$\left.\begin{array}{l} v_s^j = V_S^j(X^j,\alpha^j) \ , \ (X^j,\alpha^j) \in \partial(\Delta^2 \times \Lambda) \\ v_\mu^{j-} = \inf V_\mu^j(X^j,\alpha^j) \ | \ (X^j,\alpha^j) \in \partial M_\mu^j \cap \overline{CM}_\mu^j \\ v_\mu^{j+} = \sup V_\mu^j(X^j,\alpha^j) \ | \ (X^j,\alpha^j) \in \partial(\Delta^2 \times \Lambda) \cap CM_\mu^j \end{array}\right\} \qquad (16$$

The first relation obviously requires forming $V_S^j(\cdot)$ from suitable $\partial(\Delta_o \times \Lambda)$ taken as its level, or conversely, forming $\partial\Delta_o$, $\partial\Lambda$ from levels of suitable $V_S(\cdot)$. In the latter case a Δ_o, Λ smaller than these desired will be the secure choice.

THEOREM 1: Objective 1 is attained if, given Δ_o, Λ, μ there are programs $p^j(\cdot)$ and functions $V_S^j(\cdot)$, $V_\mu^j(\cdot)$ such that for all $(X^j,\alpha^j) \in \Delta_o^2 \times \Lambda$,

(i) $V_S^j(X^j,\alpha^j) \le v_s^j$, $\forall (X^j,\alpha^j) \in N_\varepsilon$, $j = 1,2$

(ii) for each $u^j \in p^j(X^1,X^2)$;

$$V_S^j(X^j(t),\alpha^j(t)) < 0 \ , \quad \forall s^j \in S \tag{17}$$

along the product trajectories $(X^j(X^{jo},t)\alpha^j(\alpha^{jo},t))$, $t \ge 0$, $j = 1,2$;

(iii) $0 < V_\mu^j(X^j,\alpha^j) \le v_\mu^{j+}$, $\forall(X^j,\alpha^j) \in \overline{CM}_\mu^j$, $j = 1,2$;

(iv) $V_\mu^j(X^j,\alpha^j) \le v_\mu^{j-}$, $\forall(X^j,\alpha^j) \in D^j \cap M_\mu^j$, $j = 1,2$;

(v) for each $u^j = p^j(X^1,X^2)$ there is a constant $c_j > 0$ such that

$$\dot V_\mu^j(X^j(t),\alpha^j(t)) \le -c_j \ , \quad \forall s^j \in S \tag{18}$$

along the product trajectories $(X^j(X^{jo},t),\alpha^j(\alpha^{jo},t))$, $t \ge 0$, $j = 1,2$.

Remark 1: The Objective 1 holds after a stipulated $t_c < \infty$ if Theorem 1 is satisfied with c_j selected by

$$c_j \overset{\Delta}{=} \frac{v_\mu^{j+}}{t_c} \ , \quad j = 1,2 \ . \tag{19}$$

THEOREM 2: Objective 2 is attained if Theorem 1 holds and given d there is a C^1-function $V_A(\cdot): \Delta_A \to R$ such that for the tracking pair $p^j(\cdot)$, for all $Z \in CA$,

(vi) $V_A(Z) > V_A(z)$, $\forall z \in \partial A$;

(vii) for each $u^j \in p^j(Z)$,

$$\dot V_A(Z(Z^o,t)) \ge 0 \ , \quad Z^o \in \Delta_A \ , \quad \forall s^j \in S \tag{20}$$

along product trajectories $Z(Z^o,t)$, $t \ge 0$.

PROOF. Suppose some $Z(Z^o,t)$, $t \ge 0$, $Z^o \in \Delta_A$ crosses ∂A at $t_1 > 0$. Then by (vi) , $V_A(Z(t_1)) < V_A(Z^o)$ which contradicts (vii).

7. Controllers and Adaptive Laws

Let us set up

$$V_S^j \overset{\Delta}{=} E_m(x^j) + E_m(x_m) + a^j\alpha^j \ ; \tag{21}$$

$$V_\mu^j \overset{\Delta}{=} \begin{cases} |E_m(x^j) - E_m(x_m)| + a^j\alpha^j \ , & (X^j,\alpha^j) \in CM_\mu^j \ , \\ a^j\alpha^j , & (X^j,\alpha^j) \in M_\mu^j \ ; \end{cases} \tag{22}$$

$$V_A = \tfrac{1}{2}|E_m(x^1)-E_m(x^2)| \tag{23}$$

where $a^j = (\text{sign}\ \alpha_1^j,\dots,\text{sign}\ \alpha_n^j)$, $j = 1,2$, and $E_m(x^j)$ is $E_m(\cdot)$ with x_m exchanged for x^j . Choosing N_ε in $C\Delta_L$, the character of $E_m(\cdot)$ specified additionally by (12), satisfies (i), (iii) and (iv).

To see that (vi) holds, observe that $E_m(x^j) = E_m(\xi^j,\dot\xi^j)$ of (6) and that increasing the distance $|\xi_\upsilon-\xi_\upsilon^2| \neq 0$ for at least one υ from its ∂A value increases the value of V_A .

To check upon conditions (ii), (v), (vii) we differentiate (21) - (23) with respect to time

$$\dot V_S^j(t) = \dot E_m(x^j) + \dot E_m(x_m) + a^j\dot\alpha^j \ ; \tag{24}$$

$$\dot V_\mu^j(t) = \begin{cases} \dot E_m(x^j) - \dot E_m(x_m) + a^j\dot\alpha^j \ , & (X^j,\alpha^j) \in C^+M_\mu^j \ , \\ \dot E_m(x_m) - \dot E_m(x^j) + a^j\dot\alpha^j \ , & (X^j,\alpha^j) \in C^-M_\mu^j \ , \\ a^j\dot\alpha^j \ , & (X^j,\alpha^j) \in M_\mu^j \ ; \end{cases} \tag{25}$$

$$\dot V_A(t) = [E_m(x^1)-E_m(x^2)]\cdot[\dot E_m(x^1)-\dot E_m(x^2)] \ , \tag{26}$$

where

$$\dot E_m(x^j) = \nabla E_m(x^j)\cdot f^j(x^j,u^j,w^j) = (Bu-D-P+P_m q_m)(\dot q,\dot\xi) \ . \tag{27}$$

The brackets of the functions $\mathcal{B}$, $\mathcal{D}$, $\mathcal{P}$ dropped for clarity. Moreover $C^{\pm}M^j_\mu$ are subsets of CM^j_μ defined by

$$C^+M^j_\mu: \quad E_m(x^j) \ge E_m(x_m)$$
$$C^-M^j_\mu: \quad E_m(x^j) < E_m(x_m) \quad .$$

With a suitable choice of initial states the following set of conditions imples (ii), (v) and (vii):

(a) $$\min_{u^j}\max_{s^j} \dot E_m(x^j) \le \dot E_m(x_m) \ , \ \forall (X^j,\alpha^i) \in C^+M^j_\mu \ ,$$
$$\max_{u^j}\min_{s^j} \dot E_m(x^j) \ge \dot E_m(x_m) \ , \ \forall (X^j,\alpha^j) \in C^-M^j_\mu \ ;$$

(b) $$\max_{u^1}\min_{s^1} \dot E_m(x^1) > \min_{u^2}\max_{s^2} \dot E_m(x^2) \ , \ \forall Z \in C^+A \ ,$$
$$\min_{u^1}\max_{s^1} \dot E_m(x^1) < \max_{u^2}\min_{s^2} \dot E_m(x^2) \ , \ \forall Z \in C^-A \ ,$$

for $\dot q \ne 0$, $\dot\eta \ne 0$, $j = 1,2$. In the above $C^{\pm}A$ are subsets of CA defined by:

$$C^+A: \quad E_m(x^1) \ge E_m(x^2) \ ,$$
$$C^-A: \quad E_m(x^1) < E_m(x^2) \ .$$

(c) $$a^j\dot\alpha^j = \dot E_m(x_m) - c_j \ , \ \alpha^j \ne 0 \ , \ j = 1,2 \ .$$

Observe that for $\alpha^j = 0$ there is no need for adaptation and that the system (4) crosses the surface $\dot q = 0$, $\dot\eta = 0$ time instantenously (vertically) so there is no need for control in view of the smoothness of trajectories. Conditions (a), (b) are called control conditions helping to design $p^j(\cdot)$, condition (c) is called adaptive, helping to design adaptive laws. Let us check that (a), (b), (c) indeed imply (ii), (v), (vii). Consider first the case $E_m(x^j) \ne E_m(x_m)$. Substituting (c) into (23) in view of (ii) we obtain $\dot V^j_S$ = negative terms + $\dot E_m(x^j)$. Boundedness of the work space necessitates the power; $\dot E_\omega(x^j) \le 0$ thus (ii). Substituting (a), (c), and (ii) into (24) with (18), we satisfy (v) in stipulated time t_c . Note that this holds for any initial states. The case $E_m(x^j) = E_m(x_m)$ is trivial as then $\dot V^j_S = 3\dot E_m(x_m) < 0$, $\dot V^j_\alpha = \dot E_m(x_m) - c_j \le -c_j$. Finally we check (vii). Again first let $E_m(x^1) \ne E_m(x^2)$ and observe that (b) substituted to (26) implies (vii). The case $\dot E_m(x^1) = \dot E_m(x^2)$ is obviously trivial.

Observe that, with (10), (c) is implied by the following adaptive laws

$$\dot\alpha^j_i = -\text{sign}\ \alpha^j_i\left(\mathcal{D}_{mi}\dot q^2_{mi} - \frac{c_j}{n}\right), \tag{28}$$

for $\alpha^j_i \ne 0$, $i = 1,\dots,n$. Physically the solutions $\dot\alpha^j(\alpha^{jo},t)$ represent the model energy flux which become positive or negative depending upon where α^{jo} is located (below or above the surface $\alpha^j = 0$) thus regulating the increment of α^j to zero from anywhere outside the surface $\alpha^j = 0$.

8. Modular Double RP-Manipulator

Our technique is illustrated below on the case study of the four DOF manipulator with two arms shown in Fig. 4.

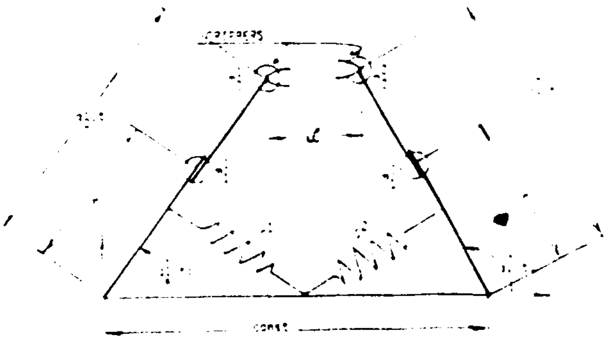


Figure 4. The modular 2-RP manipulator

The Lagrange equations of motion for each arm result in the following motion equations

$$(m_1r^2+m_2q_2^2)\ddot{q}_1 + 2_m q_2\dot{q}_1\dot{q}_2 + \lambda_3|\dot{q}_1|\dot{q}_1 + g(m_1r+m_2q_2)\cos q_1 - m_1gr + \lambda_1 q_1 + {}_2q_1^3 = u_1$$
$$m_2\ddot{q}_2 - m_2q_2\dot{q}_1^2 + \lambda_4\dot{q}_2 + m_2g\sin q_1 = u_2 \,. \qquad (29)$$

Here λ_3, λ_4 are damping coefficients, $\lambda_1,\lambda\lambda_2$ spring coefficients, g-gravity acceleration, the remainder of notations shown in Fig. 4. The superscripts "j", j = 1,2 , are ignored for the time being. We take the possible payload on the grippers as unknown but within known bounds which makes m_2 specified by

$$\underline{m} \le m_2 \le \overline{m} \,,$$

where $\underline{m}$, $\overline{m}$ positive constants. Allowing $\sin q_1 = q_1 - \frac{1}{6}q_1^3$, $\cos q_1 = 1 - \frac{1}{2}q_1^2$, and subdividing the equations (29) by corresponding inertia coefficients we obtain:

$$\ddot{q}_i + \Gamma_i + \Pi_i = {}_iu_i \,, \; i = 1,2 \qquad (30)$$

where

$$\left.\begin{aligned}
\Gamma_1 &= \frac{2m_2q_2\dot{q}_1\dot{q}_2 + \lambda_3|\dot{q}_1|\dot{q}_1}{m_1r^2 + m_2q_2^2} \,, \\
\Gamma_2 &= -q_2\dot{q}_1^2 + 1/m_2\,\lambda_4\dot{q}_2 \,, \\
\Pi_1 &= \frac{\lambda_1q_1 - \frac{1}{2}gm_1rq_1^2 + \lambda_2q_1^3 - \frac{1}{2}gm_2q_2q_1^2 + gm_2q_2}{m_1r^2 + m_2q_2^2} \\
\Pi_2 &= gq_1 - \frac{1}{6}gq_1^3 \,, \\
B_1 &= \frac{1}{m_1r^2 + m_2q_2^2} \,, \quad B_2 = \frac{1}{m_2}
\end{aligned}\right\} \qquad (31)$$

The reference model is taken as

$$\left.\begin{aligned}
&\ddot{q}_{m1} + \lambda_{m3}\dot{q}_{m1} + \lambda_{m1}q_{m1} + gq_{m2} = 0 \,, \\
&\ddot{q}_{m2} + \lambda_{m4}\dot{q}_{m2} + gq_{m1} = 0 \,.
\end{aligned}\right\} \qquad (32)$$

The total energy of the model is

$$E_m(q_m,\dot{q}_m) = \frac{1}{2}(\dot{q}_{m1}^2+\dot{q}_{m2}^2) + \frac{1}{2}\lambda_{m1}q_{m1}^2 + 2gq_{m1}q_{m2} \,. \qquad (33)$$

Differentiating it with respect to time and substituting (32),

$$\dot{E}_m(q_m,\dot{q}_m) = -\lambda_{m3}\dot{q}_{m1}^2 - \lambda_{m4}\dot{q}_{m2}^2 \,.$$

Accordingly,

$$\dot{E}_m(q,\dot{q}) = (B_1u_1-\Gamma_1)\dot{q}_1 + (B_2u_2-\Gamma_2)\dot{q}_2 \,.$$

Choose $(X^{jo}, \dot{x}^{jo}) \in C^*M^j_q$, j = 1,2 and $z^o \in C^*A$. Then the control conditions (a), (b) hold if successively

$$\left.\begin{aligned}
&\min_{u_1^j}\max_{m_2^j}[(B_1^ju_1^j-\Gamma_1^j)\dot{q}_1^j] \le -\lambda_{m3}(\dot{q}_{m1})^2 \,, \; j = 1,2 \\
&\min_{u_2^j}\max_{m_2^j}[(B_2^ju_2^j-\Gamma_2^j)\dot{q}_2^j] \le -\lambda_{m4}(\dot{q}_{m2})^2 \,, \; j = 1,2
\end{aligned}\right\} \qquad (34)$$

and

$$\left.\begin{aligned}&\max_{u_1^1}\min_{m_2^1}[(B_1^1u_1^1-\Gamma_1^1)\dot{q}_1^1] > \min_{u_1^2}\max_{m_2^2}[(B_1^2u_1^2-\Gamma_1^2)\dot{q}_1^2]\\ &\max_{u_2^1}\min_{m_2^1}[(B_2^1u_2^1-\Gamma_2^1)\dot{q}_2^1] > \min_{u_2^2}\max_{m_2^2}[(B_2^2u_2^2-\Gamma_2^2)\dot{q}_2^2] \quad .\end{aligned}\right\} \qquad (35)$$

Thus we choose u_1^1 such that for $\dot{q}_1^1 \neq 0$,

$$\min_{u_1^1}\max_{m_2^1}[(B_1^1u_1^1-\Gamma_1^1)\dot{q}_1^1] \leq -\lambda_{m3}(\dot{q}_{m1})^2$$

and for such u_1^1 , we choose u_1^2 satisfying

$$\min_{u_1^2}\max_{m_1^2}[(B_1^2u_1^2-\Gamma_1^2)\dot{q}_1^2] < \max_{u_1^1}\min_{m_2^1}[(B_1^1u_1^1-\Gamma_1^1)\dot{q}_1^1]$$

The procedure for u_2^1 and u_2^2 is identical utilizing the second inequalities of (34), (35). Assuming symmetry of arms: $m_1^1 = m_1^2 = m_1$, $m_2^1 = m_2^2 = m_2 \in [\underline{m},\overline{m}]$, $r^1 = r^2 = r$, and substituting the expressions for Γ_i^j , Π_i^j , B_i^j , $i,j = 1,2$, we obtain the tracking controllers

$$u_1^1(t) = \begin{cases} -\dfrac{\lambda_{m3}(\dot{q}_{m1})^2}{\dot{q}_1^1}[m_1r^2 + \overline{m}(q_2^1)^2 + 2\overline{m}q_1^1\dot{q}_2^1 + \lambda_3^1|\dot{q}_1|\dot{q}_1 \ , \ \forall\dot{q}_i^j \neq 0 \\ \text{suitable constant}, \ \forall\dot{q}_i^j = 0 \ , \end{cases}$$

$$u_1^2(t) = \begin{cases} \dfrac{-\lambda_{m3}(q_{m1})^2}{q_1^2}[m_1r^2 + \overline{m}(q_2^1)^2] + 2\underline{m}\dot{q}_2\dot{q}_1^2(\dot{q}_2^2) + \lambda_3^2|\dot{q}_1^2|(\dot{q}_1^2)^2 \ , \ \forall\dot{q}_i^j \neq 0 \\ \text{suitable constant}, \ \forall\dot{q}_i^j = 0 \ ; \end{cases}$$

and the collision avoidance controllers

$$u_2^1(t) = \begin{cases} \dfrac{-\lambda_{m4}(\dot{q}_{m2})^2\overline{m}}{\dot{q}_2^1} - \dfrac{\underline{m}}{q_2^1}q_2^1(\dot{q}_1^1)^2 + \lambda_4^1\dot{q}_2^1 \ , \ \forall\dot{q}_2^1 \neq 0 \\ \text{suitable constant}, \ \forall\dot{q}_2^1 = 0 \end{cases}$$

$$u_2^2(t) = \begin{cases} \dfrac{-\lambda_{m4}(\dot{q}_{m2})\underline{m}}{\dot{q}_1^2} - \overline{m}q_2^2\dot{q}_1^2\dot{q}_2^2 - \dfrac{\lambda_4^2(\dot{q}_2^2)^2}{q_1^2} \ , \ \forall\dot{q}_1^j \neq 0 \ , \\ \text{suitable constant}, \ \forall\dot{q}_i^j = 0 \ , \end{cases}$$

which imply the control conditions (a), (b) for our example. The adaptive laws (24) are

$$\begin{aligned}&\dot{\lambda}_1^j \quad 0 \ , \quad \dot{\lambda}_2^j - 0\\ &\dot{\lambda}_3^j = -(\text{sign}\ \lambda_3^j)\lambda_{m3}\dot{q}_{m1}^2 - \tfrac{1}{2}c_j\\ &\dot{\lambda}_4^j = -(\text{sign}\ \lambda_4^j)\lambda_{m4}\dot{q}_{m2}^2 - \tfrac{1}{2}c_j\end{aligned}$$

for $j = 1,2$. The first two laws vanish identically, since by design $\lambda_1^j = \lambda_{m1}$, $\lambda_2^j = \lambda_{m2}$. Numerical simulation of our modular case, with the data $m_1 = 70$kg, $\underline{m} = 30$kg, $\overline{m} = 40$kb, $r = 0.66$m, $\lambda_{m1} = 20$, $\lambda_{m2} = 20$, $\lambda_{m3} = 5$, $\lambda_{m4} = 2$, is shown in Fig. 5, and confirms the convergence-avoidance required.

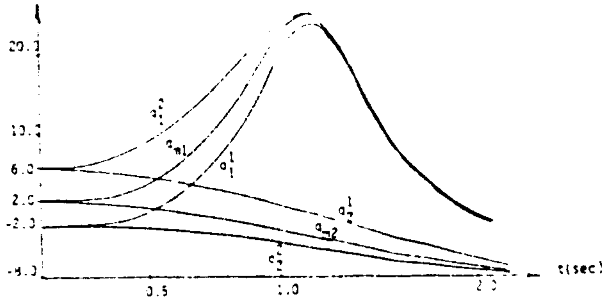


Figure 5. Numerical simulation

9. References

[1] J.M. Skowronski, Control Dynamics of Robot Manipulators, Academic Press, 1986.

[2] J.M. Skowronski, Model Reference Adaptive Control Under Uncertainty of Nonlinear Flexible Manipulators, AIAA Paper #'86 - 1976 - CP, Proc. AIAA Guidance, Navigation and Control Conf., Williamsburg,VA, 1986.

[3] G. Leitmann, J.M. Skowronski, On Avoidance Control, Journal of Optimization Theory and Applications, Dec. 1977.

[4] G. Leitmann, J.M. Skowronski, A Note on Avoidance Control, Optimal Control Applications and Methods, Dec. 1983.

[5] A. Truckenbrodt, Effects of Elasticity on the Performance of Industrial Robots, Proc. 2nd IASTED Danos International Symposium on Robotics, Switzerland, 1982, pp. 52-56.

Problems and Research Issues Associated With the Hybrid Control of Force and Displacement*

R.P. Paul
University of Pennsylvania
Philadelphia, PA 19104-6314

1 Introduction

Robots working in a perfectly structured world do not need sensing: a structured world in which the dimensions of all parts are within tolerance and in which careful planning has taken place to ensure that such parts can be assembled, a world in which everything is precisely located and everything functions as planned, a world in which all necessary jigs[1] have been designed and provided. Such a world is the production engineer's dream and will probably never exist. Even in today's most automated and structured factories there are still operators present to "un-jam" the perfect machines when the structure gets a little out of line. These machines, which we call robots, are of course only *programmable universal transfer devices*, machines which can be programmed to move tirelessly from position to position as perfectly as the parts and machines they work with. Real robots don't belong in factories any more than people do; what is needed in factories is well designed automation tended by operators.

Of course there is a limit to the number of well designed pieces of automation we can have. In the home, for instance, a sewing machine and a food-processor do their jobs much better than humans[1], but the modern kitchen is slowly beginning to fill up with such special purpose devices, which the displaced humans now spend their "leisure" time fixing. Humans are needed to provide the structure required by these devices. The dish-washer functions well in its own environment[1], but who puts the dishes in and takes them out? Further, the automation of many tasks such as dishwashing requires the substitution of massive quantities of energy and natural resources (The Regular Cycle uses 20 gallons of water which must be heated to at least 180°) in place of intelligence ("That plate's o.k., he didn't use it, just brush off the crumbs,").

When any lack of structure occurs, however, we must rely on sensors. If something is misplaced, or if something will not fit, relying on a sensor-less, geometry controlled approach would be a disaster. Sensors have two roles, to monitor task execution and to establish the state of the world. Both these tasks require the use of a world model. Both tasks also require reasoning and planning. Establishing the state of the world requires a sensor strategy and the interpretation of sensor data in terms of the world model. Monitoring task execution also requires sensor strategies and interpretation of sensor readings in terms of the world model. Errors, when they occur, are detected by the interpretation of sensor data, once again in terms of the world model. If this is to be done reliably a number of independent sensors is needed (sensors fail also). Once an error state is determined, appropriate recovery action must be planned. If a part is dropped on the floor, then it may be left, but if it is dropped inside a mechanism where it could prevent functioning, then it must either be retrieved or the mechanism replaced. Error recovery is not simple.

Of all the sensors that a robot might have, *force sensing* is the most fundamental. *Blind people function quite well in the world but people who have no kinesthetic feedback are totally helpless.* Consider the well structured world of manufacturing with a task fully under position control: the detection of any unexpected force is a clear indication that something has gone wrong. *Force sensing can provide this vital information.* In any situation where complete structure is absent, force sensing becomes primary in the sequencing of a task. Consider teleoperation where tasks have some structure[2].

*This material is based on work supported by the National Science Foundation under Grant No. DMC-8411879. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

> The general-purpose manipulator may be used for moving objects, moving levers and knobs, assembling parts, and manipulating wrenches. In all these operations the manipulator must come into physical contact with the object before the desired force and movement can be made on it. A collision occurs when the manipulator makes this contact. General-purpose manipulation consists essentially of a series of collisions with unwanted forces, the application of wanted forces, and the application of desired motions. The collision forces should be low, and any other unwanted forces should also be small.

Goertz identifies three clear states:

1. Motion in free space.
3. The exertion of a force.

This work of Goertz influenced the use of force sequencing in the manipulator control system WAVE[3] and later that of Inoue[4]. Two types of commands were included in a language WAVE describing a sequence of manipulator motions.

STOP terminate the current motion when a force equal to the argument is detected, also known as a "guarded move"[5].

FORCE during the next motion, the force in a given direction, is to be controlled to the value given as an argument. If the force is specified to be zero then the manipulator is "free" to move in the direction specified.

A further command allowed for a force to be applied by the manipulator, of course, the manipulator would have to be free in the direction In the "Force Vector Assembler Concept [6]" commanded manipulator Cartesian velocities cculd be modified by measured end-effector forces and moments

$$\mathbf{v} = \mathbf{v_0} - [\mathbf{M}]\,\mathbf{f} \qquad (1)$$

Off-diagonal elements of the matrix **M** allowed for motion to be specified in directions orthogonal to an applied force. A curious side effect of this produced a switching phenomenon similar to that described above—a continuous control system with two states. *The end effector would trace along an edge until a corner was reached and then proceed to trace along the next edge.* Unfortunately it was not possible to continue in this fashion along the following edge. A similar "switching" phenomenon occurs in a special device for making chamfer-less insertions. *The pin is brought into the hole at an angle, on contact a linkage rotates the pin to align it with the hole axis whereupon insertion occurs.* These phenomena are, however, limited to only two states and do not generalize further. Recently this type of control has formed the basis of more complex insertion strategies [7] in the form of a "generalized damper" in which the force expected is proportional to the velocity error along some direction. Both of these strategies are limited to two-state systems. A task to insert a key into a lock, turn the lock 180 degrees, and then withdraw the key, cannot be characterized by such a continuous system. It is, however, simple to describe such a task in terms of *force/displacement transitions* and control mode switches such as used in WAVE [8].

We may then characterize manipulator control into two basic states and transitions between them:

- When a manipulator is moving in free space it controls displacement and monitors force. The detection of any unpredicted forces indicates a serious error state.
- When a manipulator is constrained by the environment it controls force and monitors displacement. The detection of any unpredicted displacement indicates a serious error.
- On contact, or on breaking contact, the control and monitoring modes switch. As contact is made the reaction forces rise, indicating contact. When the desired contact force is obtained the control mode switches from displacement control to force control and the contact force is maintained at the required value. As contact is broken the reaction forces go to zero and the control mode switches to displacement control with the contact force maintained at zero.

The detection of contact is a problem for a rigid manipulator of finite inertia. When contact is detected the manipulator is brought to rest discontinuously — *it is stopped*. The kinetic energy is dissipated by various mechanisms, some potentially destructive. Given the stiffness of the manipulator and of the environment there is a clearly defined maximum speed at which contact may be safely detected and controlled.

2 Force and Position Controlled Degrees of Freedom

When a manipulator is constrained by the environment, force is controlled. There are, however, six environment constraints, three of translation and three of rotation. For each of these six degrees-of-freedom either force control or position control may be specified[3].

A robot manipulator closing a door by grasping the handle firmly has only one degree of rotational freedom — the rotation of the door about the hinge axis. In this situation force control is required along all three translation axes and force control is required about the two rotation axes perpendicular to the hinge axis. Rotational position control is required about the hinge axis. Note that one doesn't simply push on the door handle to close the door but one controls the angle of closing as a function of time — how fast the door is closed — "Don't slam the door!" All the remaining axes are in a force control mode with a desired force of zero along and about all other axes. Notice also in the above example that the forces and displacement control modes may be simply described in some orthogonal coordinate frame. In the example given, the origin of the coordinate frame would be along the hinge axis with one of the axes aligned with the hinge axis. If the z axis were aligned with the hinge axis then we could specify the compliance needed as shown in Figure 1. Notice the motion request ROTATE and the motion

```
        WITH
          FORCE  X = 0,
          FORCE  Y = 0,
          FORCE  Z = 0,
          TORQUE X = 0,
          TORQUE Y = 0
          ROTATE ABOUT Z
        UNTIL
          TORQUE Z  = 100;
```

Figure 1: A Program to Close a Door

termination specification UNTIL TORQUE Z = 100.

Manipulators are controlled by actuators located at their joints. To provide for the control of the six Cartesian environment variables, position and rotation, six joints are required. If a degree of freedom of the manipulator is constrained then attempting to control all six joints will result in an over-constrained system; large internal forces can result. If one of the joints which contributes to motion in the constrained direction or axis is controlled in force in place of displacement, the overconstraint disappears and the system is controllable. This approach was first used by Inoue in turning a crank[9] and later formed the basis of the compliance used in WAVE[3]. If more than one degree-of-freedom of constraint exists then additional joints must be force controlled to provide for each constraint. In the door closing example given above, five joints of a six-degree-of-freedom manipulator would be force controlled at zero force, and one joint, whose principal motion was in the door closing direction, would be displacement controlled.

If the motion of the joint selected to provide a degree-of-freedom does not correspond completely with the constrained direction then the the position of the manipulator will be modified in the unconstrained directions In turning the crank, the crank would be either slightly ahead or behind its correct position. If this matters it may be compensated for by modifying the commanded Cartesian position[10,11]. The major problem with compliance provided in this manner is in selecting the appropriate joints to provide the compliance. While it is always obvious which joints should be controlled in any given situation, there is as yet no formal algorithm to select these joints automatically. Another drawback is that in certain motions the joint to provide the compliance changes as the motion is made. Consider turning a crank: with the crank at the top of its motion, a joint which controls vertical motion would be appropriate to provide the necessary radial compliance, but as the crank is turned the radial direction requires a joint which controls horizontal motion. Switching between joints can be done[3] but it is difficult.

This form of compliance is very simple to implement in manipulators whose actuators are powered by electric motors as motor current is directly proportional to torque[3]. Joint friction and gearing, however, detract from this simple form of control and various attempts have been made to close a torque control loop around the joint[12]. These methods have met with only moderate success as the control coupled two rigid systems of comparable frequency response[13].

In 1981 Raibert and Craig developed a control method called "Hybrid Position/Force Control[14]," based on the theoretical formulation of the above compliance methods by Mason[15]. In this method not only was the compliance specified in an appropriate Cartesian coordinate frame but the control separation between position and force was also performed in Cartesian coordinates. The observed joint position of the manipulator was converted into Cartesian coordinates and subtracted from the desired Cartesian coordinate position yielding Cartesian position errors. Any position errors in a complying or force control direction were then set to zero and the remaining errors were transformed back into joint coordinates using the Jacobian inverse. These errors were then fed to a PID controller to reduce errors in position controlled directions to zero. Note that no position feedback is applied in any complying direction. Similarly, force errors were compared to the desired force to yield force errors in the Cartesian control frame. Any errors in a non-compliant or position controlled direction were then set to zero before these force errors were transformed into joint torque errors by the Jacobian transpose. Note that no forces were specified in any position controlled direction. In the system implemented by Raibert and Craig[14] a force and torque sensor was mounted at the wrist of the manipulator to provide feedback for the force loop. Stabilization of the force loop was, however, marginal with resort to ad hoc control methods necessary. Once again we have two rigid systems of comparable frequency response, the manipulator and the force sensor, such a system is very difficult to stabilize [13]. A similar system making use of the relationship between motor currents and joint torques has also been implemented[11]. This system, with open-loop torque control, does not suffer from the stability problems but does suffer from frictional and gearing disturbances.

In 1983, Khatib, at Stanford University went one step farther and resolved the manipulator joint inertias into effective Cartesian Inertias seen from the end-effector of the manipulator[16]. Once Cartesian position errors were detected, using the hybrid position/force control scheme, a PID controller was implemented in Cartesian end-effector coordinates to produce corrective accelerations which were then transformed into corrective forces by the effective Cartesian inertias. The resulting forces were transformed into joint torques in order to control the manipulator, again using the Jacobian transpose relationship between forces and joint torques. Unfortunately, as the manipulator configuration changes, the rate of change of effective Cartesian joint inertia varies much more rapidly than does the corresponding joint inertias. This is a considerable computational burden. A second problem is that feedback gains are applied in Cartesian coordinates while the manipulator is actuated in joint coordinates, and while it is possible to set constant high gains in joint coordinates it is not clear if similarly high gains are possible in Cartesian coordinates. No comparison between control methods for the same manipulator has been made.

3 Stability

In the previous Section we described the hybrid control of position and force. This represent the two states described by Goertz[2]. In this Section we will consider the stability of these two modes and the problems of transitions between them. In the position control of manipulators high stiffness is desired so that the manipulator will be unaffected by disturbances. We would like the manipulator to move swiftly from position to position, stopping as quickly as possible with no overshoot. When the manipulator is begin controlled at some position, we would like it to be unaffected by the application or any external forces of moments. It should act like a very stiff damped spring, very hard to deflect and *dead-beat* in its response to external disturbances. This is achieved by the application of feedback. Position feedback is required to provide stiffness, velocity feedback to provide damping, and integral feedback to provide for the removal of any bias forces. Feedback gains are limited by the stiffness of the manipulator itself. The setting of gains and the design of a manipulator for a given stiffness are a difficult engineering problems. The result is a system which has a well behaved basic response with a number of high frequency modes which decay slowly when excited. Such systems behave adequately in position mode but perform poorly in force control. The force sensor and environment are, unfortunately, both systems with natural frequency responses of the same order of magnitude as that of the manipulator. When these are coupled by contact of the manipulator with the environment then the resulting system is very difficult, if not impossible, to stabilize [17,13,18,19,12,14]. Whitney and Eppinger in their papers both indicate that stability may only be obtained when the sensor is stiff and the environment soft or when the sensor is soft and the environment stiff. Unfortunately, a soft sensor completely negates the stiffness required for position control.

The remaining problem is the implementation of the transitions between position and force control. This occurs when the manipulator makes contact with the environment. Contact between a rigid manipulator and a rigid environment is not well defined—the manipulator is moving at some velocity and then it is stopped. Where does the energy go? It is absorbed by the compliances in the system and, hopefully, dissipated. This can be destructive of many mechanical components such as, precision gears, shafts, actuators, etc. To run any commercially available robot into a brick wall would result in considerable damage! The use of any form of force sensing aggravates this problem as the force sensor is typically the least stiff member of the system, the most fragile, and absorbs all the energy. The design problem of Scheinman's "Maltese Cross" wrist force sensor was not the sensor itself but the force overload mechanism needed to protect it from damage. No form of force sensor based feedback changes this problem as the time constant of the interaction is much shorter than that of the regulator. On contact, the force sensor sees a rapidly increasing force and the sensor output goes immediately off-scale. The time scale of this interaction is of the order of a few microseconds. This signal is processed by a regulator which has a well defined minimum time response of the order of milliseconds. Contact is long since over before the regulator can respond and any damage to occur has already occurred. The *contact* problem is unsolved for *rigid* manipulator, *rigid* sensor, *rigid* environment problems.

4 Mechanical Compliance

Based on a careful analysis of a peg-in-hole insertion [20] and the force-vector steering method [6], a mechanical implementation of an insertion algorithm was developed at Draper Laboratories, the "remote center compliance — RCC" [21]. This device provided the necessary compliance to make peg insertions into low clearance holes from a vertical direction. The compliance was provided passively by springs. [1]

In the initial version of the remote center compliance no displacement sensing was provided, making the device very susceptible to damage if the displacement capacity of the device was exceeded. However, a later version, "the Instrumented Remote Center Compliance — IRCC" also provided displacement sensing which could be monitored to prevent damage. Both devices could be locked for position control to provide the two necessary control modes. The device was low inertia with high bandwidth so that contact could be made at high speed by the manipulator with the small energy of contact (due to the low inertia of the RCC) absorbed by the passive compliance. The use of *passive* compliance solves the *contact* problem[2] although the device must be locked to provide for position control and the stiffness k is defined mechanically and may not be programmed.

In order to overcome the locking problem Roberts[23] investigated an instrumented single compliant link. The displacement of the link was used to stiffen the link for position control and to soften the link for force control. In the position control mode any displacement of the end of the terminal link caused the manipulator to move in the opposite direction so as to restore the initial position. In the force control mode any displacement of the terminal link would cause the manipulator to move so as to restore the initial displacement. Contact could be detected by the deflection of the terminal link and the resulting motion, while the manipulator was brought to rest, absorbed by the compliant link as in the IRCC. It was shown that both modes were stable. We are currently working on a six-degree-of-freedom version of the device and hope to show stability and function.

[1] Hanafusa and Asada made use of a spring loaded hand to provide compliance between the workpiece, the manipulator, and the environment but did not directly address the contact problem [22].

5 Conclusions

The hybrid control of force and position is basic to the science of robotics but is only poorly understood. Before much progress can be made in robotics, this problem needs to be solved in a robust manner. However, the use of hybrid control implies the existence of a model of the environment, not an exact model (as the function of hybrid control is to accommodate these errors), but a model appropriate for planning and reasoning. The monitored forces in position control are interpreted in terms of a model of the task as are the monitored displacements in force control. The reaction forces of the task of "writing" are far different from those of "hammering." The programming of actions in such a modeled world becomes more complicated and systems of "task level" programming need to be developed.

Sensor based robotics, of which force sensing is the most basic, implies an entirely new level of technology. Indeed, robot force sensors, no matter how compliant they may be, must be protected from accidental collisions. This implies other sensors to monitor task execution and again the use of a world model. This new level of technology is the"task level," in which task actions are specified, not the actions of individual sensors and manipulators.

6 Research Issues

We may identify the following research issues in position and force control:

- Matching individual joints to Cartesian degrees-of-freedom.
- Control of the force of all the links of a manipulator not simply control of the force exerted at the end-effector.
- The hybrid position/force control of redundant manipulators.
- Robust rigid manipulator, rigid environment force and contact control.
- Contact transitions
- Compliant end-effector control of robot manipulators to provide for both position and force control.
- Compliant manipulator control to provide for both position and force control.
- Task level systems to provide for the protection of sensors.
- Motion modeling.

[2] This was graphically demonstrated by Dan Whitney at a conference in which he marched, arm rigidly outstretched, towards a an unknown wall. Without the compliance of a bent arm (to provide mechanical compliance) he would not have been able to react fast enough (regulator) to avoid hurting himself on contact

References

[1] Daniel E. Whitney. Real robots don't need jigs. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 746–752, April 1986.

[2] Ray C. Goertz. Manipulators used for handling radioactive materials. In E. M. Bennett, editor, *Human Factors in Technology*, chapter 27, McGraw Hill, 1963.

[3] Richard P. Paul. *Modeling, Trajectory Calculation and Servoing of a Computer Controlled Arm.* Technical Report AIM 177, Stanford Artificial Intelligence Laboratory, Stanford University, 1972.

[4] Hirochika Inoue. *Force Feedback in Precise Assembly Tasks.* Technical Report AIM-308, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, August 1974.

[5] Peter M. Will and David D. Grossman. An experimental system for computer controlled mechanical assembly. *IEEE Teans. on Computers*, C-24(9):879–888, 1975.

[6] James L. Nevins and Daniel E. Whitney. The force vector assembler concept. In *Proceedings, First CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators*, Udine, Italy, September 1973.

[7] Tomás Lozano-Pefez, Matthew T. Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. In Michael Brady and Richard Paul, editors, *Robotics Research: The First International Symposium*, pages 65–96, MIT Press, Cambridge, Massachusetts, 1984.

[8] Richard P. Paul. WAVE: a model-based language for manipulator control. *The Industrial Robot*, 4(1):10–17, March 1977.

[9] Hirochika Inoue. Computer controlled bilateral manipulator. *Bulletin of the Japanese Society of Mechanical Engineers*, 14(69):199–207, 1971.

[10] Richard P. Paul. *Robot Manipulators: Mathematics, Programming, and Control. MIT Press Series in Artificial Intelligence*, MIT Press, Cambridge, Massachusetts, 1981.

[11] Hong Zhang and Richard P. Paul. Hybrid control of robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 602–607, March 1985.

[12] Johnson Y-S Luh, William D. Fisher, and Richard P. Paul. Joint torque control by a direct feedback for industrial robots. In *Proceedings of the 20th. IEEE Conference on Decision and Control*, pages 265–271, San Diego, CA, December 1981.

[13] Steven D. Eppinger and Warren P. Seering. On dynamic models of robot force control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 29–34, April 1986.

[14] Marc H. Raibert and John J. Craig. Hybrid position/force control of manipulators. *Trans. ASME Journal of Dynamics, Systems, Measurement, and Control*, 102:126–133, June 1981.

[15] Matthew T. Mason. *Compliance and Force Control for Computer Controlled Manipulators.* Technical Report AIM-515, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, April 1979.

[16] Oussama Khatib. The operational space formulation in the analysis, design, and control of robot manipulators. In Olivier Faugeras and Georges Giralt, editors, *Robotics Research: The Third International Symposium*, pages 263–270, MIT Press, Cambridge, Massachusetts, 1986.

[17] Daniel E. Whitney. Historical perspective and state of the art in robot force control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 262–268, April 1985.

[18] James A. Maples and Joseph J. Becker. Experiments in force control of robotic manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 695–702, April 1986.

[19] Oussama Khatib and Joel Burdick. Motion and force control of robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1381–1386, April 1986.

[20] Daniel E. Whitney. Quasi-static assembly of compliantly supported rigid parts. *Trans. ASME Journal of Dynamics, Systems, Measurement, and Control*, 104:65–77, March 1982.

[21] Samuel H. Drake. *Using Compliance in Lieu of Sensory Feedback for Automatic Assembly.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1977.

[22] Hideo Hanafusa and Haruhiko Asada. A robot hand with elastic fingers and its application to assembly process. In Michael Brady et al., editors, *Robot Motion*, chapter 5, pages 337–360, MIT Press, Cambridge, Massachusetts, 1982.

[23] Randal K. Roberts, Richard P. Paul, and Benjamin M. Hillberry. The effect of wrist force sensor stiffness on the control of robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 269–274, April 1985.

S25-63
118
188194

Adaptive Hybrid Control of Manipulators

H. Seraji
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

JJ570450

Abstract

~~The paper presents~~ simple methods for the design of adaptive force and position controllers for robot manipulators within the hybrid control architecture. The force controller is composed of an adaptive PID feedback controller, an auxiliary signal and a force feedforward term; and it achieves tracking of desired force setpoints in the constraint directions. The position controller consists of adaptive feedback and feedforward controllers and an auxiliary signal; and it accomplishes tracking of desired position trajectories in the free directions. The controllers are capable of compensating for dynamic cross-couplings that exist between the position and force control loops in the hybrid control architecture. The adaptive controllers do not require knowledge of the complex dynamic model or parameter values of the manipulator or the environment. The proposed control schemes are computationally fast and suitable for implementation in on-line control with high sampling rates.

is presented.

1. Introduction

Although control of robot manipulators has been studied extensively in recent years, this study has been focused primarily on position control of manipulators in free motion within an unconstrained environment. In many practical applications, the manipulator is constrained by the environment and certain degrees-of-freedom are lost for motion due to environmental constraints. When the manipulator makes contact with the environment, the contact forces must be controlled in the constraint directions, while the positions are controlled simultaneously in the free directions.

The problem of manipulator control in a constrained environment has been investigated by several researchers [1]. At present, three major conceptual approaches exist for simultaneous position and force control. Paul and Shimano [2] suggest a method which uses certain joints for position control while the remaining joints are used for force control. Salisbury [3] puts forward a technique for controlling the end-effector stiffness characteristics in the Cartesian space. Raibert and Craig [4] propose a conceptual architecture, based on the analysis of Mason [5], for "hybrid control" which allows forces to be controlled in the constraint directions by a force controller, while simultaneously controlling positions in the free directions by a position controller. Raibert and Craig, however, do not prescribe a general and systematic method for the design of position and force controllers. Nevertheless, hybrid control has gained considerable popularity over the other two alternatives for simultaneous position and force control [6-13].

The present paper puts forth systematic methods for the design of adaptive force and position controllers within the hybrid control architecture. The force controller achieves tracking of desired force setpoints, while the position controller accomplishes tracking of desired position trajectories. The force and position controller gains are generated by adaptation laws by means of simple arithmetic operations, and thus the controllers are computationally fast and suitable for on-line implementation with high sampling rates. The adaptive controllers do not require knowledge of the complex dynamic model or parameter values of the manipulator or the environment.

The paper is structured as follows. In Section 2, the hybrid control architecture is outlined and the problem is stated. Section 3 addresses the design of force control system using model reference adaptive control (MRAC) theory. The design of position control system is discussed briefly in Section 4. In Section 5, the force and position controllers are integrated in the hybrid control architecture. Finally, Section 6 discusses the results of the paper and draws some conclusions.

2. Problem Statement

In this section, the hybrid force/position* control architecture is discussed briefly, and the force and position tracking control problems are stated.

Let us consider a robot manipulator which performs a number of different tasks in a Cartesian space {X}. Each task, in general, involves motion of the manipulator end-effector in certain directions and, simultaneously, exertion of force by the end-effector on the environment in the remaining directions [5]. The directions of motion and force depend on the nature of the particular task to be performed and are reflected in the "task matrix" in the hybrid control architecture shown in Figure 1. The task matrix also contains the transformations required to map the measurements $\underline{\theta}$ and $\underline{P}_s$ of joint encoders and force/torque sensors into position and force variables in the constraint frame defined with respect to the task geometry [4]. Note that the desired force and position trajectories are also specified in the constraint frame. For any given task, the n-dimensional Cartesian space {X} can be decomposed into two orthogonal ℓ- and m-dimensional subspaces {Y} and {Z}, where $n = \ell + m$. The "position subspace" {Y} contains the ℓ directions (i.e., degrees-of-freedom) in which the manipulator end-effector is free to move and along which <u>end-effector</u> <u>position</u> is to be controlled. The "force subspace" {Z} contains the remaining m directions in which the manipulator end-effector is constrained by and interacts with the environment and along which the <u>contact</u> <u>force</u> is to be controlled.

In the hybrid force/position control problem addressed in this paper, we consider the "virtual" Cartesian force $\underline{F}$ acting on the end-effector as the manipulated variable and the position or force of the end-effector as the controlled variables [14]. The hybrid control architecture is based on two independent and non-interacting controllers as shown in Figure 1; namely, the position controller which operates in {Y} and the force controller which acts in {Z}. The position controller generates the Cartesian end-effector force $\underline{F}_y$ required to cause the end-effector motion to track a desired position trajectory in {Y}. The force controller produces the Cartesian end-effector force $\underline{F}_z$ needed to ensure that the end-effector force follows a desired force setpoint in {Z}. Since we cannot physically apply Cartesian forces to the end-effector, we instead compute and implement the equivalent joint torques needed to effectively cause these forces. The required joint torques are obtained from the Cartesian forces by means of the Jacobian matrix $J(\underline{\theta})$ of the manipulator, where $\underline{\theta}$ is the joint angle vector.

We shall now address the problems of force and position control separately in Sections 3 and 4 and then integrate the results in Section 5.

3. Design of Force Control System

In this section, a simple dynamic model for force control in the subspace {Z} is described, and an adaptive force control scheme is developed.

3.1 <u>Dynamic Force Model</u>

The full dynamic model of the end-effector plus force/torque sensor in contact with the environment is complex [15]. However, the dynamic behavior of this system can be modelled approximately by a mass-spring-damper in each degree-of-freedom as shown in Figure 2 and described by the differential equation

$$m\,\ddot{z}(t) + d\,\dot{z}(t) + k\,z(t) = f(t) \qquad (1)$$

Generalizing this simple model to the m-dimensional force subspace {Z}, the dynamic behavior of the system in {Z} can be expressed by the differential equation

$$M_o\ddot{\underline{Z}}(t) + D_o\dot{\underline{Z}}(t) + K_e\underline{Z}(t) = \underline{F}_z(t) \qquad (2)$$

where $\underline{Z}(t)$ is the mx1 end-effector position/orientation vector, M_o is the symmetric positive-definite mxm generalized mass matrix, D_o is the mxm generalized damping matrix, K_e is the diagonal mxm generalized stiffness matrix and $\underline{F}_z$ is the mx1 force vector applied to the end-effector in the force subspace {Z}. The elements of K_e are the "equivalent" translational (force) and rotational (torque) coefficients of elasticity (stiffness) of the system in various directions in {Z}. By an appropriate choice of the {Z} subspace origin, the mx1 force/torque vector $\underline{P}(t)$ exerted by the end-effector on the environment is related to $\underline{Z}(t)$ by the generalization of Hooke's law as

$$\underline{P}(t) = K_e\,\underline{Z}(t) \qquad (3)$$

From equations (2) and (3), we obtain

$$A\,\ddot{\underline{P}}(t) + B\,\dot{\underline{P}}(t) + \underline{P}(t) = \underline{F}_z(t) \qquad (4)$$

where $A = M_oK_e^{-1}$ and $B = D_oK_e^{-1}$ are mxm matrices. Equation (4) gives a simple dynamic model of the system in the force subspace {Z}. Since the manipulator dynamics is highly nonlinear, the matrices A and B in equation (4) are dependent on the end-effector Cartesian position and velocity vectors $\underline{X}$ and $\dot{\underline{X}}$ and also on the system parameters such as the equivalent stiffness and the payload mass, which are represented by the parameter vector $\underline{p}$. Furthermore, due to internal cross-coupling of the manipulator dynamics, a "disturbance" term $\underline{C}_p(\underline{Y})$ must be included in equation (4) to represent the dynamic coupling from the position loop into the force loop, where $\underline{Y}$

* In this paper, "position" implies position and orientation and "force" implies force and torque.

is the end-effector position vector in (Y). Thus, a more realistic model for force control is obtained as

$$A(\underline{X},\dot{\underline{X}},\underline{p})\,\ddot{\underline{P}}(t) + B(\underline{X},\dot{\underline{X}},\underline{p})\dot{\underline{P}}(t) + \underline{P}(t) + \underline{C}_p(\underline{Y}) = \underline{P}_s(t) \qquad (5)$$

Equation (5) is a set of highly complex nonlinear and coupled second-order differential equations.

3.2 Force Control Scheme

In order to control the force/torque $\underline{P}(t)$ exerted by the end-effector on the environment, let us employ a PID controller with adaptive gains $\{K_p(t),\ K_I(t),\ K_D(t)\}$ and an auxiliary signal $\underline{d}(t)$ in the force control law

$$\underline{P}_s(t) = \underline{P}_r(t) + K_p(t)\,\underline{E}(t) + K_I(t)\int_0^t \underline{E}(t)dt + K_D(t)\,\dot{\underline{E}}(t) + \underline{d}(t) \qquad (6)$$

where $\underline{P}_r(t)$ is the mx1 vector of desired force trajectory used as a feedforward term, and the mx1 force tracking-error vector $\underline{E}(t) = \underline{P}_r(t) - \underline{P}(t)$ is the deviation of the actual (measured) force from the desired value. Since in practical applications the desired force trajectory is very often a constant setpoint $\underline{P}_r(t) = \underline{P}_r$, the PID control law is particularly suitable for this situation. Furthermore, the auxiliary signal $\underline{d}(t)$ compensates for the cross-coupling term $\underline{C}_p$ and the time and parameter variations of A and B matrices. Note that the feedforward term $\underline{P}_r(t)$ is included in the control law (6) since ideally we want $\underline{P}(t) = \underline{P}_r(t)$. In equation (6), the gains of the PID controller, namely $K_p(t)$, $K_I(t)$ and $K_D(t)$, and also the auxiliary signal $\underline{d}(t)$ are adapted in real-time to accomplish force setpoint tracking in spite of the nonlinear and possibly time-varying behavior of the system model (5).

On applying the linear control law (6) to the system model (5), we obtain

$$A\,\ddot{\underline{P}}(t) + B\,\dot{\underline{P}}(t) + \underline{P}(t) + \underline{C}_p = \underline{P}_r(t) + K_p\underline{E}(t) + K_I\int_0^t \underline{E}(t)dt + K_D\dot{\underline{E}}(t) + \underline{d}(t) \qquad (7)$$

Using $\underline{E}(t) = \underline{P}_r(t) - \underline{P}(t)$ and noting that $\dot{\underline{E}}(t) = -\dot{\underline{P}}(t)$ and $\ddot{\underline{E}}(t) = -\ddot{\underline{P}}(t)$ for a constant desired force, equation (7) can be written as

$$\ddot{\underline{E}}(t) + A^{-1}(B + K_D)\,\dot{\underline{E}}(t) + A^{-1}(I + K_p)\,\underline{E}(t) + A^{-1}K_I\,\underline{E}^*(t) = A^{-1}\,[\underline{C}_p - \underline{d}(t)] \qquad (8)$$

where $\underline{E}^*(t) = \int_0^t \underline{E}(t)dt$ is the mx1 integral error vector. Equation (8) can be expressed in standard state-space form as

$$\frac{d\underline{x}_e(t)}{dt} = \begin{pmatrix} 0 & I_m & 0 \\ 0 & 0 & I_m \\ -A^{-1}K_I & -A^{-1}(I+K_p) & -A^{-1}(B+K_D) \end{pmatrix}\underline{x}_e(t) + \begin{pmatrix} 0 \\ 0 \\ A^{-1}(\underline{C}_p-\underline{d}) \end{pmatrix} \qquad (9)$$

where $\underline{x}_e(t) = \begin{pmatrix} \underline{E}^*(t) \\ \underline{E}(t) \\ \dot{\underline{E}}(t) \end{pmatrix}$

is the 3mx1 augmented error vector. Equation (9) constitutes the "adjustable system" in the MRAC framework.

Now, in the ideal situation, the desired behavior of the force error $\underline{E}_m(t)$ is described by the homogeneous differential equation

$$\ddot{\underline{E}}_m(t) + D_3\dot{\underline{E}}_m(t) + D_2\underline{E}_m(t) + D_1\underline{E}_m^*(t) = \underline{0} \qquad (10)$$

where D_1, D_2 and D_3 are constant mxm matrices which are chosen such that equation (10) is stable and embodies the desired performance of the force control system. By choosing D_1, D_2 and D_3 as diagonal matrices, the force errors will be decoupled; for instance

$$\ddot{E}_{mi}(t) + d_{3i}\dot{E}_{mi}(t) + d_{2i}E_{mi}(t) + d_{1i}E_{mi}^*(t) = 0 \qquad (11)$$

where the coefficients d_{1i}, d_{2i} and d_{3i} are chosen such that the tracking-error $E_i(t) = P_{ri}(t) - P_i(t)$ has a desired behavior and $d_{2i}d_{3i} > d_{1i}$ to ensure stability. Equation (10) can be written as

$$\dot{\underline{x}}_m(t) = \begin{pmatrix} 0 & I_m & 0 \\ 0 & 0 & I_m \\ -D_1 & -D_2 & -D_3 \end{pmatrix}\underline{x}_m(t) = D\,\underline{x}_m(t) \qquad (12)$$

where $\underline{x}_m(t) = \begin{pmatrix} \underline{E}_m^*(t) \\ \underline{E}_m(t) \\ \dot{\underline{E}}_m(t) \end{pmatrix}$ is the 3mx1 desired error vector. Equation (12) constitutes the "reference model" in the context of MRAC theory. Since the initial values of the actual and desired forces are often the same, the initial error $\underline{x}_m(0)$ is equal to zero, and hence from equation (12), $\underline{x}_m(t) = \exp[Dt]\cdot\underline{x}_m(0) \equiv \underline{0}$ for all t.

Now, in order for the adjustable system state $\underline{x}_e(t)$ to tend to the reference model state $\underline{x}_m(t) = \underline{0}$ asymptotically, from Reference [16] we require

$$\begin{pmatrix} 0 \\ 0 \\ A^{-1}\dot{\underline{d}} \end{pmatrix} = Q_o^{-1} M\underline{x}_e + Q_o^* M\dot{\underline{x}}_e \tag{13a}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ A^{-1}\dot{K}_I & A^{-1}\dot{K}_p & A^{-1}\dot{K}_D \end{pmatrix} = Q_1^{-1} M\underline{x}_e\underline{x}_e' \begin{pmatrix} \alpha_1 I_m & 0 & 0 \\ 0 & \beta_1 I_m & 0 \\ 0 & 0 & \gamma_1 L \end{pmatrix}$$

$$+ Q_1^* M \frac{d}{dt}(\underline{x}_e\underline{x}_e') \begin{pmatrix} \alpha_2 I_m & 0 & 0 \\ 0 & \beta_2 I_m & 0 \\ 0 & 0 & \gamma_2 L \end{pmatrix} \tag{13b}$$

where " ' " denotes transposition, $\{\alpha_1, \beta_1, \gamma_1\}$ are positive scalars, $\{\alpha_2, \beta_2, \gamma_2\}$ are zero or positive scalars, and L is an mxm constant matrix to be specified later. In equation (13), Q_o and Q_1 are symmetric positive-definite 3mx3m matrices, Q_o^* and Q_1^* are symmetric positive semi-definite 3mx3m matrices, and the symmetric positive-definite 3mx3m matrix

$$M = \begin{pmatrix} M_1 & M_2 & M_3 \\ M_2 & M_4 & M_5 \\ M_3 & M_5 & M_6 \end{pmatrix}$$

is the solution of the Lyapunov equation for the reference model (12), namely

$$M\ D + D'M = -N \tag{14}$$

where N is a symmetric positive-definite 3mx3m matrix. In deriving equation (13), the matrices A, B, and $\underline{C}_p$ in the robot model (5) are assumed to be unknown and "slowly time-varying" compared with the adaptation algorithm, since these matrices cannot change significantly in each sampling interval which is of the order of a millisecond. Now, in order to make the controller adaptation laws <u>independent</u> of the model matrix A, we choose

$$Q_o = \frac{1}{\delta_1} A^* \quad ; \quad Q_1 = A^* \quad ; \quad Q_o^* = \delta_2[A^*]^{-1} \quad ; \quad Q_1^* = [A^*]^{-1} \tag{15}$$

where $\{\delta_1, \delta_2\}$ are positive and zero or positive scalars, and

$A^* = \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & A \end{pmatrix}$ is a symmetric positive-definite 3mx3m matrix.

Substituting from equation (15) into equation (13), after simplification we obtain the adaptation laws

$$\dot{\underline{d}}(t) = \delta_1\underline{q}(t) + \delta_2\dot{\underline{q}}(t) \tag{16}$$

$$\dot{K}_I(t) = \alpha_1[\underline{q}(t)\ \underline{E}^{*\prime}(t)] + \alpha_2 \frac{d}{dt}[\underline{q}(t)\ \underline{E}^{*\prime}(t)] \tag{17}$$

$$\dot{K}_p(t) = \beta_1[\underline{q}(t)\ \underline{E}'(t)] + \beta_2 \frac{d}{dt}[\underline{q}(t)\ \underline{E}'(t)] \tag{18}$$

$$\dot{K}_D(t) = \gamma_1[\underline{q}(t)\ \dot{\underline{E}}'(t)L] + \gamma_2 \frac{d}{dt}[\underline{q}(t)\ \dot{\underline{E}}'(t)L] \tag{19}$$

where $\underline{q}(t)$ is the mx1 "weighted" force error vector defined as

$$\underline{q}(t) = M_3\ \underline{E}^*(t) + M_5\ \underline{E}(t) + M_6\ \dot{\underline{E}}(t) \tag{20}$$

and M_3, M_5, M_6 are appropriate submatrices of M. Thus, the required auxiliary signal and PID controller gains are obtained as

$$\underline{d}(t) = \underline{d}(0) + \delta_1 \int_0^t \underline{q}(t)dt + \delta_2\underline{q}(t) \tag{21}$$

$$K_I(t) = K_I(0) + \alpha_1 \int_0^t \underline{q}(t)\ \underline{E}^{*\prime}(t)dt + \alpha_2\underline{q}(t)\ \underline{E}^{*\prime}(t) \tag{22}$$

$$K_p(t) = K_p(0) + \beta_1 \int_0^t \underline{q}(t)\ \underline{E}'(t)dt + \beta_2 \underline{q}(t)\ \underline{E}'(t) \qquad (23)$$

$$K_D(t) = K_D(0) + \gamma_1 \int_0^t \underline{q}(t)\ \dot{\underline{E}}'(t)Ldt + \gamma_2 \underline{q}(t)\ \dot{\underline{E}}'(t)L \qquad (24)$$

The force control law is then given by

$$\underline{P}_c(t) = \underline{P}_r(t) + \underline{d}(t) + K_I(t) \int_0^t \underline{E}(t)dt + K_p(t)\underline{E}(t) + K_D(t)\dot{\underline{E}}(t) \qquad (25)$$

It is noted that the auxiliary signal $\underline{d}(t)$ can be generated by a PI^2D controller driven by the force error $\underline{E}(t)$ since, from equations (20)-(21), $\underline{d}(t)$ can be expressed as

$$\underline{d}(t) = \underline{d}(0) + [\delta_2 N_6]\dot{\underline{E}}(t) + [\delta_1 N_6 + \delta_2 N_5]\underline{E}(t)$$

$$+ [\delta_1 N_5 + \delta_2 N_3] \int_0^t \underline{E}(t)dt + [\delta_1 N_3] \int_0^t \left\{ \int_0^t \underline{E}(t)dt \right\} dt$$

In practical implementation of any force control law, differentiation of the noisy force measurement $\underline{P}(t)$ is undesirable and, moreover, differentiation of the constant force setpoint $\underline{P}_r$ produces unwanted impulses. This argument suggests that the derivative $\dot{\underline{E}}(t)$ in equations (20), (24) and (25) must be replaced by $-K_e\dot{\underline{Z}}(t)$ using equation (3). This yields the linear adaptive force control law

$$\underline{P}_c(t) = \underline{P}_r(t) + \underline{d}(t) + K_I(t) \int_0^t \underline{E}(t)dt + K_p(t)\underline{E}(t) - K_v(t)\dot{\underline{Z}}(t) \qquad (26)$$

which is shown in Figure 3, where $K_v(t) = K_D(t)K_e$ is the mxm velocity feedback gain matrix and the term $K_v(t)\dot{\underline{Z}}(t)$ represents velocity damping. Choosing $L = (K_e^{-1})^2$, the adaptation laws now become

$$\underline{d}(t) = \underline{d}(0) + \delta_1 \int_0^t \underline{q}(t)dt + \delta_2 \underline{q}(t) \qquad (27)$$

$$K_I(t) = K_I(0) + \alpha_1 \int_0^t \underline{q}(t)\underline{E}^{*\prime}(t)dt + \alpha_2 \underline{q}(t)\underline{E}^{*\prime}(t) \qquad (28)$$

$$K_p(t) = K_p(0) + \beta_1 \int_0^t \underline{q}(t)\underline{E}'(t)dt + \beta_2 \underline{q}(t)\underline{E}'(t) \qquad (29)$$

$$K_v(t) = K_v(0) - \gamma_1 \int_0^t \underline{q}(t)\dot{\underline{Z}}'(t)dt - \gamma_2 \underline{q}(t)\dot{\underline{Z}}'(t) \qquad (30)$$

where

$$\underline{q}(t) = N_3\underline{E}^*(t) + N_5\underline{E}(t) - N_6^*\dot{\underline{Z}}(t) \qquad (31)$$

and $N_6^* = N_6K_e$. It can be shown that by proper selection of matrix N in the Lyapunov equation (14), the submatrices N_3, N_5 and N_6 in equation (20) can be made equal to the desired values W_I, W_p and $W_D = W_vK_e^{-1}$ respectively, and hence equation (31) becomes

$$\underline{q}(t) = W_I\underline{E}^*(t) + W_p\underline{E}(t) - W_v\dot{\underline{Z}}(t) \qquad (32)$$

where W_I, W_p and W_v are the mxm diagonal weighting matrices chosen by the designer to reflect the relative significance of the integral error $\underline{E}^*$, the force error $\underline{E}$, and the velocity $\dot{\underline{Z}}$, respectively. From equations (27)-(30), the general expression for a typical controller gain K(t) which acts on the signal $\underline{y}(t)$ to generate the term $K(t)\underline{y}(t)$ in the control law (26) can be written as

$$K(t) = K(0) + \mu_1 \int_0^t \underline{q}(t)\underline{y}'(t)dt + \mu_2 \underline{q}(t)\underline{y}'(t) \qquad (33)$$

where μ_1 and μ_2 are scalar gains. This computation can be performed by a simple "adaptation module" shown in the block diagram of Figure 4 (For $\underline{d}(t)$, we set $\underline{y} = 1$). The adaptation module acts on the two input signals $\underline{q}(t)$ and $\underline{y}(t)$ to produce the output signal $K(t)\underline{y}(t)$. The force control law (26) can then be constructed by parallel connection of four such modules.

The force control scheme developed in this section is extremely simple, since the adaptation laws (27)-(30) generate the controller gains by means of simple integration using, for instance, the trapezoidal rule in equation (33) can be implemented as

$$\underline{K}(i) = \underline{K}(i-1) + \mu_1 \cdot \frac{T_s}{2} [\underline{q}(i)\underline{y}'(i) + \underline{q}(i-1)\underline{y}'(i-1)] + \mu_2[\underline{q}(i)\underline{y}'(i)] \tag{34}$$

where the integer i denotes the sampling instant and T_s is the sampling period. As a result, the force control law (26) can be evaluated very rapidly and consequently, the force control scheme can be implemented for real-time control with high sampling rates (typically 1 KHz). High sampling rate is very desirable in force control applications and yields improved dynamic performance. The adaptation laws (27)-(30) do not require the complex nonlinear model of manipulator dynamics (5) or any knowledge of parameters of the manipulator or the environment. This is due to the fact that the adaptive force controller has "learning capabilities" and can rapidly adapt itself to gross changes in the manipulator or the environment parameters.

4. Design of Position Control System

In this section, a dynamic model for position control in the subspace {Y} is described and an adaptive position control scheme is briefly explained.

4.1 Dynamic Position Model

The dynamics of the end-effector in the Cartesian space {X} can be represented by [14]

$$\Lambda(\underline{X})\ddot{\underline{X}} + \underline{\mu}(\underline{X}, \dot{\underline{X}}) + \underline{\psi}(\underline{X}) + \underline{P} = \underline{F} \tag{35}$$

where Λ is the Cartesian mass matrix, $\underline{\mu}$ is the Cartesian centrifugal, Coriolis and friction vector, $\underline{\psi}$ is the Cartesian gravity loading vector, $\underline{P}$ is the force vector exerted by the end-effector on the environment, and $\underline{F}$ is the generalized "virtual" Cartesian force vector applied to the end-effector. In the position subspace {Y}, equation (35) can be written as [17]

$$A(\underline{X}, \dot{\underline{X}}, \underline{p})\, \ddot{\underline{Y}}(t) + B(\underline{X}, \dot{\underline{X}}, \underline{p})\, \dot{\underline{Y}}(t) + C_o(\underline{X}, \dot{\underline{X}}, \underline{p})\, \underline{Y}(t) + \underline{C}_f(\underline{P}) = \underline{F}_y(t) \tag{36}$$

where the $\ell x \ell$ matrices A, B, C_o are highly complex nonlinear functions of $\underline{X}$, $\dot{\underline{X}}$ and the system parameters $\underline{p}$. $\underline{C}_f$ represents the dynamic coupling effect from the force loop into the position loop which is a function of the force vector $\underline{P}$ in {Z}, and $\underline{F}_y$ is the $\ell x1$ force vector applied to the end-effector in the position subspace {Y}. Equation (36) is a set of highly complex nonlinear and coupled second-order differential equations.

4.2 Position Control Scheme

The Cartesian position control scheme is developed fully in Reference [18]. For the sake of completeness, the results are summarized in this section.

The linear adaptive position control law is given by

$$\underline{F}_y(t) = \underline{f}(t) + K_p(t)\underline{E}_p(t) + K_v(t)\dot{\underline{E}}_p(t) + C(t)\underline{R}(t) + B(t)\dot{\underline{R}}(t) + A(t)\ddot{\underline{R}}(t) \tag{37}$$

as shown in Figure 5, where $\underline{R}(t)$ is the $\ell x1$ reference (desired) position trajectory vector, $\underline{E}_p(t) = \underline{R}(t) - \underline{Y}(t)$ is the $\ell x1$ position tracking-error vector, $\underline{f}(t)$ is an auxiliary signal, and $[K_p\underline{E}_p + K_v\dot{\underline{E}}_p]$ and $[C\underline{R} + B\dot{\underline{R}} + A\ddot{\underline{R}}]$ are the contributions due to the feedback and feedforward controllers respectively. The required auxiliary signal and controller gains are adapted according to the following laws:

$$\underline{f}(t) = \underline{f}(0) + \delta_1 \int_0^t \underline{r}(t)dt + \delta_2\underline{r}(t) \tag{38}$$

$$K_p(t) = K_p(0) + \nu_1 \int_0^t \underline{r}(t)\underline{E}_p'(t)dt + \nu_2\underline{r}(t)\underline{E}_p'(t) \tag{39}$$

$$K_v(t) = K_v(0) + \eta_1 \int_0^t \underline{r}(t)\dot{\underline{E}}_p'(t)dt + \eta_2\underline{r}(t)\dot{\underline{E}}_p'(t) \tag{40}$$

$$C(t) = C(0) + \mu_1 \int_0^t \underline{r}(t)\underline{R}'(t)dt + \mu_2\underline{r}(t)\underline{R}'(t) \tag{41}$$

$$B(t) = B(0) + \gamma_1 \int_0^t \underline{r}(t)\dot{\underline{R}}'(t)dt + \gamma_2\underline{r}(t)\dot{\underline{R}}'(t) \tag{42}$$

$$A(t) = A(0) + \lambda_1 \int_0^t \underline{r}(t)\ddot{\underline{R}}(t)dt + \lambda_2\underline{r}(t)\ddot{\underline{R}}(t) \tag{43}$$

where the $\ell x1$ "weighted" position error vector $\underline{r}(t)$ is defined as

$$\underline{r}(t) = W_p\underline{E}_p(t) + W_v\dot{\underline{E}}_p(t) \qquad (44)$$

In equations (38)-(43), $\{\delta_1, \nu_1, \eta_1, \mu_1, \gamma_1, \lambda_1\}$ are positive scalars, $\{\delta_2, \nu_2, \eta_2, \mu_2, \gamma_2, \lambda_2\}$ are positive or zero scalars, and W_p and W_v are weighting matrices specified by the designer to reflect the relative significance of the position and velocity errors $\underline{E}_p$ and $\dot{\underline{E}}_p$. Note that from equations (38)-(43), the controller gains can be computed using the same adaptation "module" as in Section 4 (eqn. 33) shown in Figure 4. It is seen that the position control scheme is extremely simple since the controller gains are obtained from equations (38)-(43) by simple integration (such as trapezoidal rule) and thus the computational time required to evaluate the position control law (37) is extremely short. Thus, the position control scheme can be implemented for on-line control with high sampling rates (~ 1 KHz), resulting in improved dynamic performance.

5. Hybrid Force/Position Control System

In Sections 3 and 4, the Cartesian end-effector forces $\underline{F}_x$ and $\underline{F}_y$ are generated by the force and position controllers to accomplish force and position tracking in {Z} and {Y} respectively. Since Cartesian forces cannot be applied to the end-effector in practice, these end-effector forces must be mapped into the equivalent joint torques. Thus, in order to implement the force and position controllers, the control law in joint space is given by [19]

$$\underline{T}(t) = J'(\underline{\theta}) \begin{pmatrix} \underline{F}_x(t) \\ \underline{F}_y(t) \end{pmatrix} \qquad (45)$$

where $\underline{\theta}$ is the nx1 joint angle vector, $\underline{T}$ is the nx1 joint torque vector, and J is the nxn Jacobian matrix of the manipulator, with appropriate reordering of the columns of J if necessary.

It is important to note that although the force and position controllers are separate in the hybrid control architecture, there exists dynamic cross-coupling from the force control loop into the position control loop and vice versa. This coupling is due to the fact that the end-effector dynamics in the Cartesian space {X} is strongly cross-coupled; i.e. the application of end-effector force in any direction affects the end-effector positions in all directions. The cross-coupling effects are modelled as "disturbance" terms $\underline{C}_p$ and $\underline{C}_f$ in the force and position control loops. The adaptive controllers are capable of compensating for these disturbances and maintaining a good tracking performance. The ability to cope with cross-coupling effects in the hybrid control architecture is an important feature of the adaptive control schemes of Sections 3 and 4.

6. Discussion and Conclusions

Simple adaptive force and position control schemes for manipulators in a hybrid control architecture are described in this paper. The control schemes are computationally fast and do not require the complex dynamic model or parameter values of the manipulator or the environment. The force and position control loops are stable since the design is based on the Lyapunov method which guarantees stability as a by-product of the design.

There are certain differences between the proposed approach and the conventional hybrid control of Raibert and Craig [4]. Firstly, in the present approach, the force and position control problems are formulated in the Cartesian space with the end-effector Cartesian forces as the manipulated variables; whereas in [4], the problems are formulated in the joint space. The proposed formulation results in computational improvement since inverse Jacobians are not needed in the control loops. Secondly, in the proposed approach, the "task matrix" operates on the measured variables so as to produce the position and force variables that need to be controlled; whereas in [4], a selection matrix and its complement are used after formation of tracking-errors. The present approach seems more straightforward and appealing than the conventional approach.

An attractive feature of the adaptive controllers designed in this paper is their abilities to compensate for dynamic cross-couplings that exist between the position and force control loops in the hybrid control architecture. Furthermore, the adaptive force and position controllers have "learning capabilities" to cope with unpredictable changes in the manipulator or environment parameters such as the stiffness This is due to the fact that the controller gains are adapted rapidly on the basis of the manipulator performance. The low computational requirements make the proposed control schemes suitable for implementation in on-line hybrid control with high sampling rates.

7. Acknowledgement

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

8. References

[1] D.E. Whitney: "Historical perspective and state of the art in robot force control," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 262-268, St. Louis, MO, 1985.

[2] R.P. Paul and B. Shimano: "Compliance and control," Proc. Joint Aut. Control Conf., pp. 694-699, San Francisco, CA, 1976.

[3] J.K. Salisbury: "Active stiffness control of a manipulator in Cartesian coordinates," Proc. IEEE Conf. on Decision and Control, pp. 95-100, Albuquerque, NM, 1980.

[4] M.H. Raibert and J.J. Craig: "Hybrid position/force control of manipulators," ASME J. Dyn. Systems, Measurement and Control, Vol. 102, pp. 126-133, 1981.

[5] M.T. Mason: "Compliance and force control for computer controlled manipulators," IEEE Trans. Systems, Man and Cybernetics, SMC11(6), pp. 418-432, 1981.

[6] H. Zhang and R.P. Paul: "Hybrid control of robot manipulators," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 602-607, St. Louis, MO, 1985.

[7] R. West and H. Asada: "A method for the design of hybrid position/force controllers for manipulators constrained by contact with the environment," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 251-259, St. Louis, MO, 1985.

[8] P.G. Backes, G.G. Leininger and C.H. Chung: "Real-time Cartesian coordinate hybrid control of a PUMA 560 manipulator," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 608-613, St. Louis, MO, 1985.

[9] P.G. Backes, G.G. Leininger and C.H. Chung: "Joint self-tuning with Cartesian setpoints," ASME J. Dyn. Systems, Measurement and Control, Vol. 108, pp. 146-150, 1986.

[10] O. Khatib and J. Burdick: "Motion and force control of robot manipulators," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 1381-1386, San Francisco, CA, 1986.

[11] T. Yoshikawa: "Dynamic hybrid position/force control of robot manipulators," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 1393-1398, San Francisco, CA, 1986.

[12] A.J. Koivo: "Force-position-velocity control with self-tuning for robotic manipulators," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 1563-1568, San Francisco, CA, 1986.

[13] C.C. Nguyen, F.J. Pooran and T. Premack: "Control of robot manipulator compliance," Proc. Intern. Symp. on Robot Manipulators, Albuquerque, NM, 1986.

[14] O. Khatib: "Dynamic control of manipulators in operational space," Proc. 6th IFToMM Congress on Theory of Machines and Mechanisms, pp. 1128-1131, New Delhi, India, 1983 (Wiley, New Delhi).

[15] S.D. Eppinger and W.P. Seering: "On dynamic models of robot force control," Proc. IEEE Intern. Conf. on Robotics and Automation, pp. 29-34, San Francisco, CA, 1986.

[16] H. Seraji: "A simple method for model reference adaptive control," submitted for publication to IEEE Trans. Aut. Control, 1986.

[17] A. Balestrino, G. DeMaria and L. Sciavicco: "An adaptive model following control for robotic manipulators," ASME J. Dyn. Systems, Measurement and Control, Vol. 105, pp. 143-151, 1983.

[18] H. Seraji: "Direct adaptive control of manipulators in Cartesian space," Journal of Robotic Systems, Vol. 4, No. 1, pp. 157-178, 1987.

[19] J.J. Craig: "Robotics-Mechanics and Control," Addison-Wesley Publishing Company, Reading, MA, 1986.

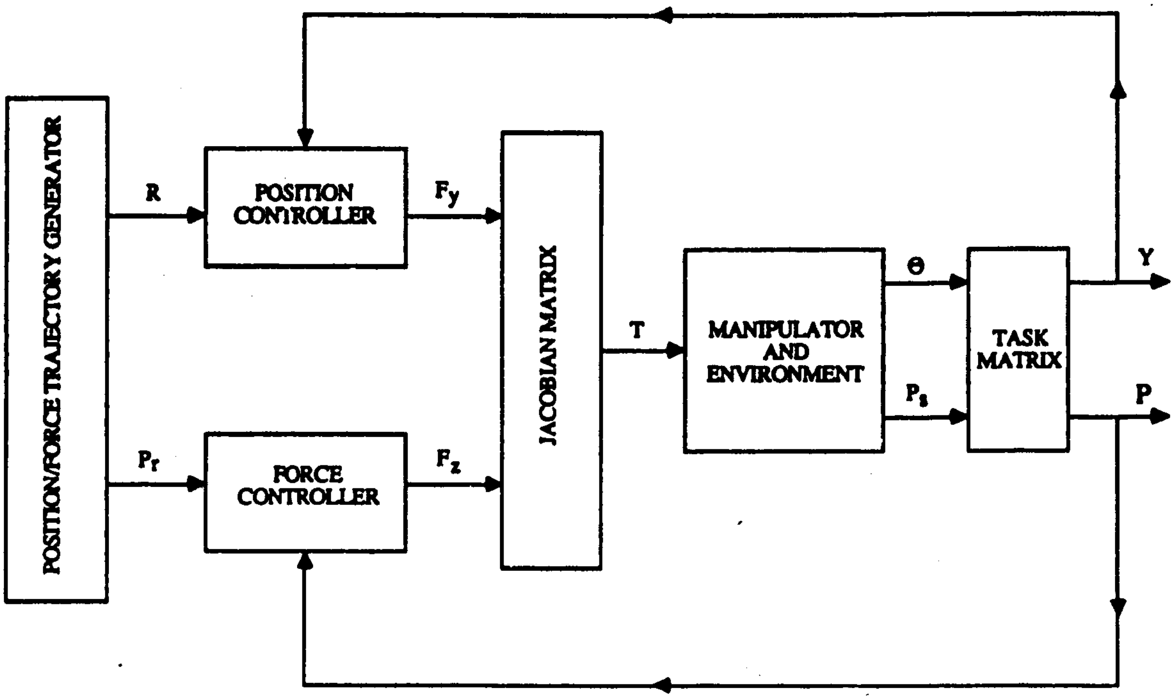


Figure 1. Hybrid Control Architecture

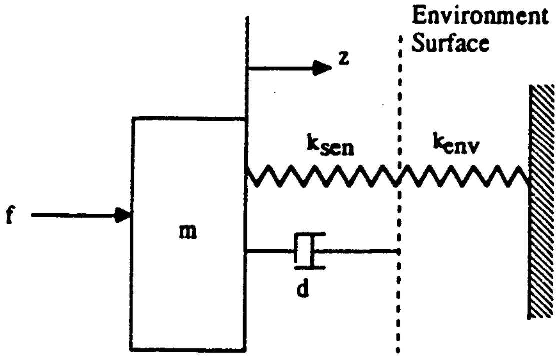


Figure 2. Mass-Spring-Damper Model

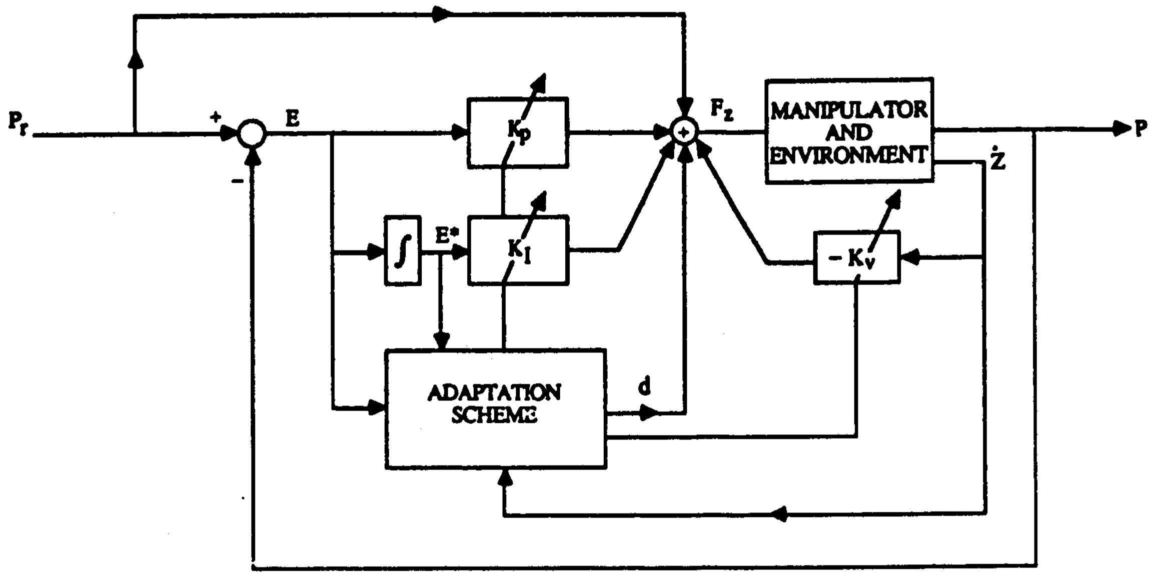


Figure 3. Adaptive Force Control System

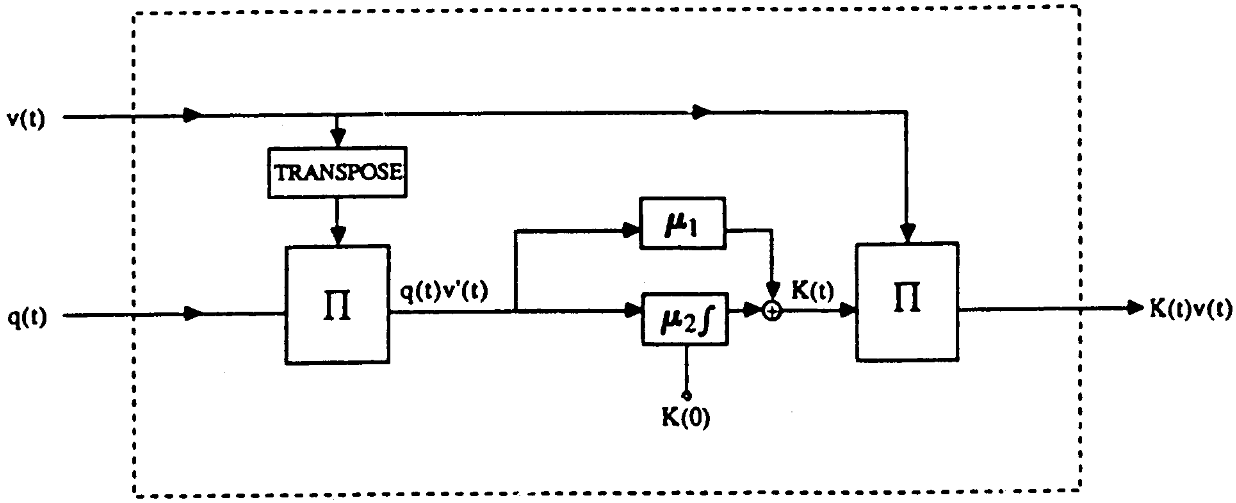


Figure 4. Structure of the Basic Adaptation Module

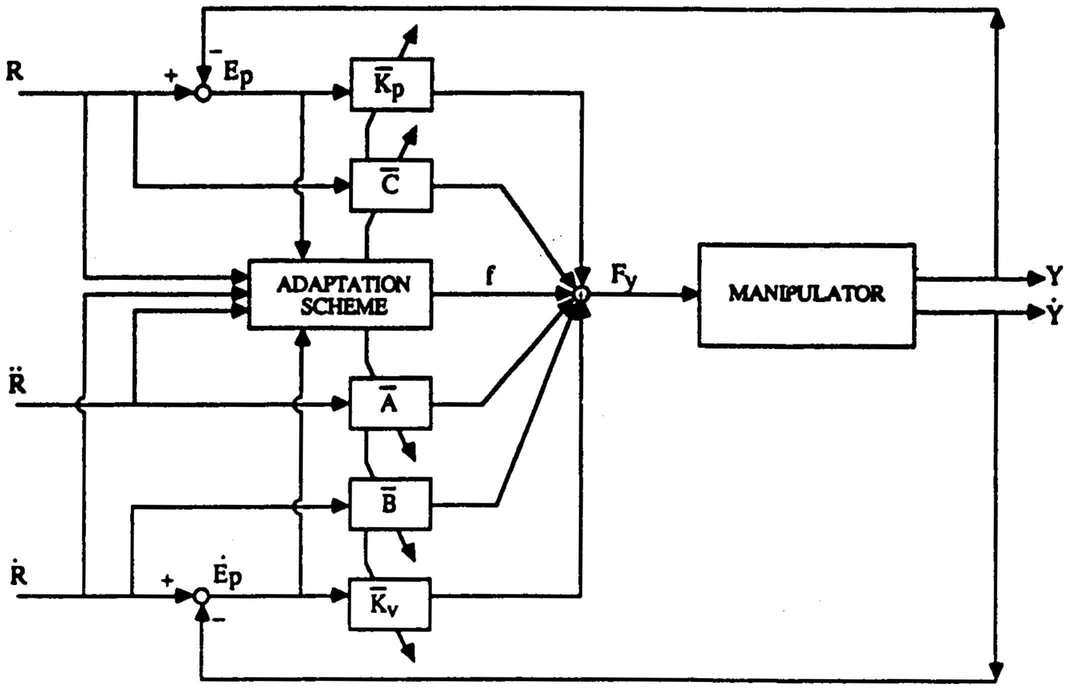


Figure 5. Adaptive Position Control System

S26-37
58
188195

Adaptive Hybrid Position/Force Control of Robotic Manipulators

F. Pourboghrat
Southern Illinois University
Carbondale, IL 62901

U6507000

1. Abstract

~~In this paper~~, the problem of position and force control for the compliant motion of the manipulators is considered. The external force and the position of the end-effector are related by a second order impedance function. The force control problem is then translated into a position control problem. For that, an adaptive controller is designed to achieve the compliant motion. The design uses the Liapunov's direct method to derive the adaptation law. The stability of the process is guaranteed from the Liapunov's stability theory. The controller does not require the knowledge of the system parameters for the implementation, and hence is easy for applications.

2. Introduction

While position control is appropriate when a manipulator is following a trajectory through space, when any contact is made between the end-effector and the manipulator's environment, position control may not suffice. Precise control of manipulators, in the face of uncertainties and variations in their environments, is a prerequisite to feasible application of robot manipulators to complex handling and assembly problems, in industry and space. An important step toward achieving such control may be taken by providing manipulator hands with sensors that provide information about the progress of interactions with the environment. Properly applied force control can reduce the positioning accuracy necessary to perform a given task accurately, and in fact make possible assembly tasks which would be otherwise impossible.

The problem of position/force control has attracted many researchers in the recent past years [1-5]. Among these works one can distinguish two different approaches. The first approach is aimed at providing the user with a means of specifying and controlling forces and positions in a non-conflicting way, [1-3]. This involves specification of a set of position controlled axes and an orthogonal set of force controlled axes. The second approach is aimed at developing a relationship between interaction forces and manipulator positions, [4,5]. This way, by controlling the manipulator position and specifying its relationship to the interaction forces, a designer can ensure that the manipulator will be able to maneuver in a constrained environment while maintaining appropriate contact forces.

In the first group, Paul and Shimano [1] partition the cartesian space and find the best joints to force servo to approximate the desired force and position commands. Raibert and Craig [2] involve all joints in satisfying the cartesian position and force commands simultaneously. Whitney [3] arrives at a single loop velocity control scheme with the net effect of controlling the contact force. In that paper, the impedance matrix approach establishes a connection between the two different approaches mentioned above. In all the above works, the structure of the controller depends on the kinematics and dynamics of the manipulator and of the environment. That is, if the end-effector of a manipulator in motion encounters a point with new constraint, then the controller structure must be changed. In the second group, Salisbury [4] defines a linear static function that relates interaction forces to end-effector position, by a stiffness matrix in a cartesian coordinate frame. Monitoring this relationship ensures that the manipulator will be able to maneuver successfully in a constrained environment. Kazerooni, et. al. [5] extend the previous work [4] and define a generalized mechanical impedance for the manipulator which is used for the compliant motion control. Their approach is an extended frequency domain approach of Salisbury's stiffness control. Also, their design is stable and shows robustness in the face of bounded uncertainties. In the second group approach, the controller's structure does not depend on the kinematics and dynamics of the manipulator and that of the environment. However, in both groups, the controller requires the knowledge of the parameters of the system.

In this work, the concept of mechanical impedance, [4,5] is used in order to relate the external forces to the position and orientation of the end-effector. Hence, the problem of force control is recasted in the position control problem. The objective is to design a controller for the manipulator, so that the perturbed dynamic relationship for the overall system is given by a second order impedance function. For that, a model reference adaptive controller is designed [6,7], where the desired impedance function is used to select the adaptive control model. The direct method of Liapunov is used for the derivation of adaptation laws. This guarantees the stability of the overall system.

3. Manipulator Dynamics

Consider a manipulator with n joints, providing n degrees of freedom. The dynamic equation of such manipulator is given by

$$M(q)\,\ddot{q} + h(q,\dot{q}) + g(q) = T \qquad (1)$$

where q is the n-dimensional vector of joint angular positions, $\dot{q}$ and $\ddot{q}$ are, respectively, the vectors of joint angular velocities and joint angular accelerations, M(q) is the nxn symmetric, positive definite inertia matrix of the manipulator, $h(q,\dot{q})$ is the n-dimensional vector of Coriolis and centrifugal forces, g(q) is the n-dimensional vector of gravitational forces, and T is the n-dimensional vector of torque inputs, applied to the manipulator.

Let δq be the perturbation of the joint angular position vector q, from q_e, and δT be the perturbation of the input torques, from T_e. Then the linearized dynamic equation is given by

$$M(q_e)\,\delta\ddot{q} + G(q_e) = \delta T \qquad (2)$$

where, $G(q_e) = [\partial g/\partial q, \ldots \partial g/\partial q_n]$ for $q = q_e$.

The joint input torques applied to the manipulator, the external forces on the end-effector, and the actuator torques are related by

$$\delta T = L\delta T_a + J_c^T \delta F \qquad (3)$$

where δT, δT_a and δF are n-dimensional perturbations of the joint input torques, the actuator torques and the end-effector external forces, and J_c is the Jacobian matrix which transform joint angle coordinates to end-effector position and orientation. Also, ,the dynamic equation of actuators are approximately given by

$$\delta\dot{T}_a = A_a \delta T_a + B_a \delta U \qquad (4)$$

where

$$A_a = \text{diag}\,[-\lambda_{a1}, \ldots, -\lambda_{an}]$$

$$B_a = \text{diag}\,[b_1, \ldots, b_n]$$

and δU is the n-dimensional vector of actuator inputs, [5].

From equations (2), (3,) and (4), the dynamic equation of the manipulator and the actuator is given by

$$\begin{aligned} \delta\dot{X} &= A\,\delta X + B\,\delta U + D\,\delta F \\ \delta q &= C\,\delta X \end{aligned} \qquad (5)$$

where

$$\delta X = [\delta q^T, \delta\dot{q}^T, \delta T_a^T]^T \in R^{3n}$$

$$A = \begin{bmatrix} 0 & I & 0 \\ -M^{-1}G & 0 & M^{-1}L \\ 0 & 0 & A_a \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ B_a \end{bmatrix} ,$$

$$D = \begin{bmatrix} 0 \\ M^{-1}J_c^T \\ 0 \end{bmatrix} ,$$

$$C = [I \quad 0 \quad 0].$$

and the pairs (A,B) and (A,C) are respectively controllable and observable.

4. Model Reference Adaptive Control

In this section, a controller is designed so that the dynamic perturbation equation of the overall closed-loop manipulator system is given according to the inverse of the desired impedance. To achieve this, a model reference adaptive control strategy is employed. The reference model is chosen such that its transfer matrix is identical to the inverse of the desired mechanical impedance. Hence, the dynamic equation of the reference model is given by

$$\delta\dot{X}_m = A_m \delta X_m + B_m \delta F$$
$$\delta q_m = C_m \delta X_m \qquad (6)$$

where

$$A_m = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ A_{m0} & A_{m1} & A_{m2} \end{bmatrix}, \qquad B_m = \begin{bmatrix} 0 \\ 0 \\ B_{m0} \end{bmatrix}$$
$$C_m = [I \quad 0 \quad 0]$$

such that δX_m is the 3n-dimensional incremental vector of model's joints and actuators values, δF is the n-dimensional vector of incremental external forces. The transfer function matrix of the model is given by

$$G_m(s) = \delta q_m(s)/\delta F(s) = C_m[sI-A_m]^{-1}B_m$$

such that the two dominant poles of the model are given by

$$J_c G_m(s) = (Js^2 + k_1 s + k_0)^{-1} \qquad (7)$$

where J_c is the Jacobian matrix, and J, k_0, and k_1 are respectively the inertia matrix, stiffness matrix, and damping matrix of the desired mechanical impedance, given by

$$\delta F/\delta Y_m = (Js^2 + k_1 s + k_0), \quad \delta Y_m = \text{model's spatial displacement.}$$

Let us define the state error to be

$$e = \delta X_m - \delta X. \qquad (8)$$

Subtracting equation (6) from (5), we get the dynamic equation of the state error as

$$\dot{e} = A_m e + (A_m - A)\delta X + (B_m - D)\delta F - B\delta U. \qquad (9)$$

Let us now choose the input torque to be

$$\delta U = K_x \delta X + K_F \delta F + K_e e \qquad (10)$$

where K_x, K_e, K_F are variable gain matrices with appropriate dimensions. Plugging δU from (10) into (9), we get

$$\dot{e} = (A_m - BK_e)e + (A_m - A - BK_x)\delta X + (B_m - D - BK_F)\delta F. \qquad (11)$$

The problem, now, is how to vary the feedback and the feedforward gain matrics, K_x, K_F and K_e, such that equation (11) is stable and the state error e approaches zero, according to a prespecified transient behavior.

To achieve perfect model following, the state error and its derivative should become zero, that is $e = \dot{e} = 0$. The conditions for perfect model following are given by

$$\begin{aligned} B_M - D - BK_F &= 0 \\ A_m - A - BK_x &= 0 \\ \bar{A} - A_m + BK_e &= 0 \end{aligned} \qquad (12)$$

Furthermore, under perfect model following conditions in (12) the error equation (11) will become

$$\dot{e} = \bar{A}\, e. \qquad (13)$$

that is, the transient behavior of the state error is determined by the constant matrix $\bar{A}$, which is defined by the designer and is Hurwitz.

5. Adaptation Law

The controller gains of the adaptive system should be adjusted such that the overall closed-loop system is stable and follows the reference model. The direct method of Liapunov may be chosen for determining the adaptation law, [6,7].

Let the corresponding Liapunov function for adaptation be given by

$$V = ||e||_P + ||B_m - D - BK_F||_R + ||A_m - A - BK_x||_S + ||\bar{A} - A_m + BK_e||_M \tag{14}$$

where P,R, and S are 3n x 3n arbitrary positive definite symmetric constant matrices. Also, the quadratic norm for any matrix F and any positive definite symmetric constant matrix G is defined by

$||F||_G = tr[F^T GF]$, where tr = trace.

The function V is posi+ive definite, except when there is a perfect model matching it becomes zero. Differentiating V, we get

$$\begin{aligned} \dot{V} = & e^T(P\bar{A}+\bar{A}P)e \\ & +2tr\{(B_m-D-BK_F)^T[Pe\ \delta F^T + R \frac{d}{dt}(B_m-D-BK_F)] \\ & +2tr\{(A_m-A-BK_x)^T[Pe\ \delta X^T + S \frac{d}{dt}(A_m-A-BK_x)] \\ & +2tr\{(\bar{A}-A_m+BK_e)^T[Pe\ e^T + M \frac{d}{dt}(A_m-BK_e-\bar{A})]. \end{aligned} \tag{15}$$

Also notice that, since matrix $\bar{A}$ is Hurwitz, then for any given positive definite symmetric matrix Q there exists a positive definite symmetric matrix P such that

$$P\bar{A} + \bar{A}^T P = -Q$$

Now, for the stability, $\dot{V}$ should be negative. One way to satisfy this is to choose

$$\begin{aligned} \dot{K}_F &= B^\dagger R^{-1} Pe\delta F^T \\ \dot{K}_x &= B^\dagger S^{-1} Pe\delta X^T \\ \dot{K}_e &= -B^\dagger M^{-1} Pe\ e^T \end{aligned} \tag{16}$$

where $B^\dagger = [0,0,B_a^{-1}\}$ is the pseudo-inverse of B. However, since R, S and M are arbitrary matrices, we can choose them such that $RB_a = \alpha I$, $SB_a = \beta I$ and $MB_a = \gamma I$, where α, β, γ are positive scalars. Then denoting $E = [0,0,I]$, the adaptation laws can be given by

$$\begin{aligned} \dot{K}_F &= \alpha EPe\delta F^T \\ \dot{K}_x &= \beta EPe\delta X^T \\ \dot{K}_e &= -\gamma EPee^T \end{aligned} \tag{17}$$

With these adaptation laws, the derivative of the Liapunov function, $\dot{V}$, is given by

$$\dot{V} = -||e||_Q = -e^T Qe < 0$$

which is negative for non-zero state error, (i.e. $e \neq 0$). This guarantees the asymptotic stability of the equilibrium point, $e = 0$.

The proposed design adaptively controls both the position and the end-effector force, and is appropriate for compliant motion of the robotic manipulators. The proposed adaptive controller is shown in Figure 1.

Moreover, if the spatial displacement and velocity can be directly measured, then the knowledge of J_c is not necessary for the implementation of the adaptive controller.

6. Conclusions

In this paper, the definition of mechanical impedance used in [4,5], is employed. The external force and the position of the end-effector are related by a second order impedance function. The force control problem is then translated to a position control problem. An adaptive controller is designed for the latter problem to achieve the compliant motion for the manipulator. The design uses the Liapunov's direct method to derive the adaptation law. The stability of the process is guaranteed from the Liapunov's stability theory. The major advantage of this method is that the controller does not depend on the knowledge of the system parameters and those of the environment. It uses the measured forces at the end-effector and the position and velocity of the end-effector in the joint space. The controller is simple and can be easily implemented by small computers.

7. References

[1] Paul, R.P.C. and Shimano, D., "Compliance and Control", Proc. Joint Auto. Contr. Conf., 1976.

[2] Raibert, M.H. and Craig, J.J., "Hybrid Position/Force Control of Manipulators", ASME, J. Dynamic Syst., Measurement, Contr., Vol. 102, June 1981.

[3] Whitney,D.E., "Force Feedback Control of Manipulator Fine Motions", ASME, J. Dynamic Syst., Measurement, Contr., June 1977.

[4] Salisbury, K.J., "Active Stiffness Control of Manipulator in Cartesian Coordinates", Proc. 19th IEEE Conf. Decision Contr., Dec. 1980.

[5] Kazerooni, H., Sheridan, T.B. and Houpt, P.K., "Robust Compliant Motion for Manipulators, Part I: The Fundamental Concepts of Compliant Motion, Part II: Design Method", IEEE J. Robotics and Automation, Vol. RA-2, No. 2, June 1986.

[6] Seraji, H., "Design of Adaptive End-Effector Controllers for Manipulators", IFAC Symp. Contr. Distributed Para. Syst., UCLA, June 1986.

[7] Pourboghrat, F. ana Lee, H.P., "Adaptive Control for DC Motor Driven Robotic Manipulators", Conf. Appl. Motion Contr., Minneapolis, June 1986.

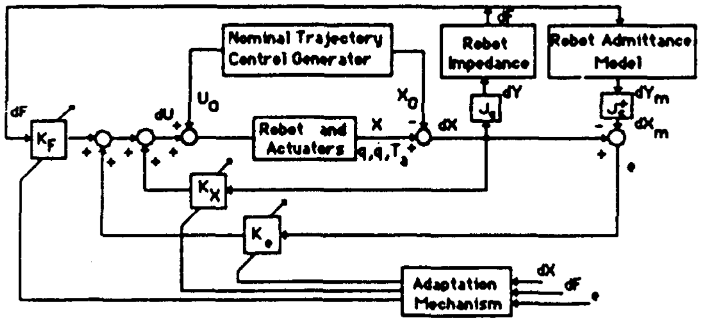


Figure 1. Adaptive Position/Force Controller for Robot

S7-63
98
188196

Flexible Manipulator Control Experiments and Analysis

S. Yurkovich, Ü. Özgüner, A. Tzes, and P.T. Kotnik
Ohio State University
Columbus, OH 43210

ΦM 59328

Abstract

are described.

~~This presentation describes~~ modeling and control design for flexible manipulators, both from an experimental and analytical viewpoint. From the aplication perspective, ~~we report on~~ an ongoing effort within the laboratory environment at The Ohio State University, where experimentation on a single link flexible arm is underway. Several unique features of this study are described here. First, the manipulator arm is slewed by a direct drive dc motor and has a rigid counterbalance appendage. Current experimentation is from two viewpoints: 1) rigid body slewing and vibration control via actuation with the hub motor, and 2) vibration suppression through the use of structure-mounted proof-mass actuation at the tip. Such an application to manipulator control is of interest particularly in design of space-based telerobotic control systems, but has received little attention to date. From an analytical viewpoint, ~~we discuss~~ parameter estimation techniques within the closed-loop for self-tuning adaptive control approaches. ~~Also~~ introduced is a control approach based on output feedback and frequency weighting to counteract effects of spillover in reduced-order model design. A model of the flexible manipulator based on experimental measurements is evaluated for such estimation and control approaches.

is described.

are discussed.

1. Introduction

Traditionally, robotic manipulator arms have been modeled as being composed of rigid links, with co-located actuators and sensors, towards the goal of ensuring stable and reliable control. In order for typical manipulator arms to maintain this rigid property as modeled while carrying payloads, the mechanical design requires large and massive links. This in turn dictates that the torques applied by the joint actuators be large, and heavy, usually geared motors are needed for actuation. Moreover, the controller for such a system is forced to move the arm slowly and deliberately so as to prevent any swaying or vibrations.

In recent years there has been much interest in using light-weight, higher performance arms for both commercial and space-based applications, leading to the study of flexible manipulator control. The advantages of flexible robotic manipulators are many, including faster system response and lower energy consumption, smaller actuators and overall trimmer mechanical design, reduced nonlinearity effects due to elimination of gearing, less overall mass and generally less cost. Obvious tradeoffs, however, complicate the issue of flexible manipulator control, primarily centering on the design of controllers to compensate for, or to be robust in the presence of flexure effects. With the advent of advanced computational resources, strides are currently being made towards solution of the many problems associated with control design.

Control design for lightweight flexible manipulator arms has gained the attention of control theorists only recently, and several approaches have emerged. Most prominant are approaches which either linearize and truncate for controller design, or solve the nonlinear robotics problem for rigid link motion control and treat the flexible dynamics separately. For example, the problem of observation spillover and truncation error effects is treated in [1], where in simulation studies a linear feedback scheme around a reduced order model is introduced for a single-link manipulator. In [2] control of the rigid motion is accomplished via state feedback linearization whereas vibrational dynamics are treated as disturbance effects. Several other analyses have appeared along these basic lines, using various approaches [3,4,5,6,7]. From an applications viewpoint, however, only a few studies have been documented for parameter estimation, system identification, and control. Most prominant among these are the works of Book, *et al.* [8,9,10,11] for time optimal slew experiments, related studies at JPL in flexible beam control [12,13], Schmitz and Canon [14] using non-colocated and tip position sensing in the control algorithm, and several studies conducted at NASA LaRC [15,16].

In this presentation we report on progress made to date on modeling and control design for flexible manipulators, both from an experimental and analytical viewpoint. Specifically, we discuss the ongoing effort within the Control Research Laboratory at The Ohio State University, where experimentation on a single link flexible arm is underway. The manipulator arm is slewed by a direct drive dc motor and has a rigid counterbalance appendage. Current experimentation is from two viewpoints: 1) rigid body slewing and vibration control via actuation with the hub motor, and 2) vibration suppression through the use of structure-mounted proof-mass actuation at the tip. Real-time parameter estimation techniques, within the closed-loop for self-tuning adaptive control, is under investigation and is described briefly here. In these initial studies, a model of the flexible manipulator based on experimental measurements is evaluated.

2. Experimental Setup

2.1 Apparatus

Within the Control Research Laboratory at The Ohio State University, several experimental configurations are under study for system identification and slewing and vibration control for flexible mechanical structures. In this presentation we focus on experimentation and simulation analysis with a single link flexible arm, depicted in Figure 1. The arm is made of 0.0625 inch thick aluminum and is counterbalanced with a rigid aluminum appendage with mass equal to that of the arm. Hub actuation is accomplished by a 3.4 ft-lb *direct drive* dc motor which has an optical encoder with a quadrature digital output to sense motor shaft position, and a tachometer to measure motor shaft speed. This, then, allows both hub position and velocity feedback for control. Strain gauges (for monitoring and parameter estimation) and an accelerometer are placed along the arm, and a 512-elemens CID linear array camera with RS-422 interface is used for sensing the tip position by observing the lamp fixed to the tip of arm. With such a scheme, the tip sensing mechanism (camera) is utilized in verification and tuning of the predicted endpoint position. A related objective for this setup is to achieve control without camera information feedback, with for example rate and acceleration sensing feedback, for application in space-based manipulator systems where off-structure reference for sensing is impractical.

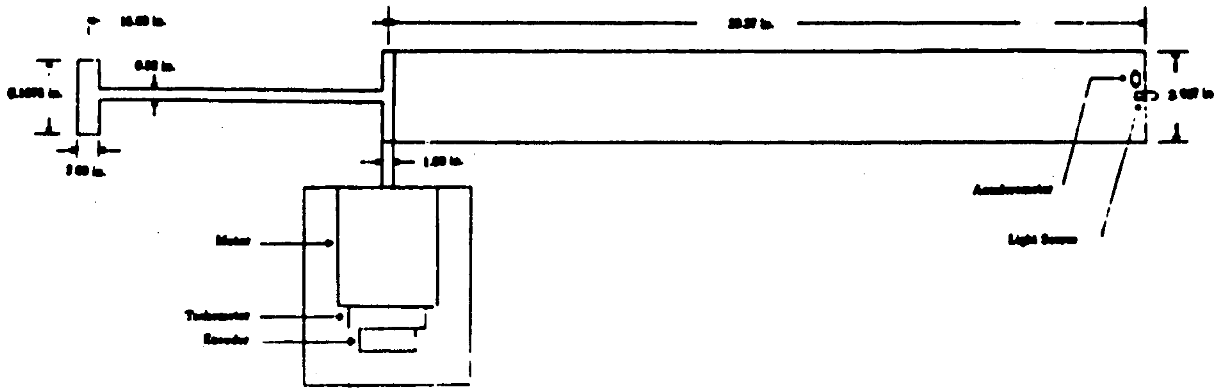


Figure 1: OSU Flexible Arm

Some characteristics of the arm are given below.

Table 1: Arm Characteristics

Material		6061-T6 Aluminum
Modulous of Elasticity		68.944×10^{9} N/m^2
Cross Sectional Area Moment of Inertia		
	Flexible Arm	3.350×10^{-11} m^4
	Rigid Appendage	2.427×10^{-8} m^4
Lengths		
	Flexible Arm	1.0 m
	Rigid Appendage	0.381 m

The unique features of the structure are the direct drive mechanism, chosen to minimize effects of backlash and other nonlinearities due to gearing, and the counterbalance appendage, which provides a more realistic model of application-oriented structures. Such a hybrid structure does, on the other hand, pose unique problems for analytical modeling.

Two computing environments are available in the laboratory for real-time control and data acquisition. The first such system is an IBM AT which uses different combinations of several custom-built cards in addition to the A/D equipment. These cards include a controller for the slewing motor with electronics utilized in processing data received from the linear-array camera. Another card, designed in-house, processes strain gauge and accelerometer data, and includes a low-noise, high-gain amplifier with a low pass filter and excitation to the bridge circuitry. A third custom board is used to drive the proof-mass actuators (discussed in [17] and later in this presentation). The board receives an analog voltage from the D/A and amplifies it to a current which is adequate to drive the actuators. The linear array camera is interfaced to the computer using a custom board which converts the camera's serial data stream to a number corresponding to the position of the beam endpoint. The second computing system is the MicroVax II, equipped with commercially available A/D-D/A boards and real-time operating system software. For the study presented here, the data acquisition is carried out using the IBM AT due to the availability of the camera interface electronics.

2.2 Modeling and Frequency Response

For purposes of finite element modeling studies, the arm is analysed in two separate components, those being the flexible arm itself and the rigid counter balance. The cylindrical mass at the end of the rigid component is modeled as a point mass, and the two components are connected at the pinned joint (motor shaft). For the FEM analysis, each component is modeled as a two-dimensional elastic beam, and the software package ANSYS [18] was used to generate the first five modes of the system shown in Table 2 below. We note that torsional modes were assumed to be insignificant, and were therefore neglected in the analysis. The principle advantage to modeling the arm in this manner is that the effects of the counter balance in the static characteristics are included. Several other approaches were utilized, such as considering the joint at the motor shaft to be a fixed point (clamped free), negating any effect the counter balance may have on the beam dynamics. Experimental results (described below) indicate that the former approach, described above, gives the closest match to measured responses.

For purposes of comparison, several experiments were conducted in testing response characteristics of the apparatus. An open loop frequency response was found by applying a sinusoidal system input torque (varying the motor current), and recording measurements of the tip position; the procedure is similar to that employed in [14]. Data was taken over the range 0.2 Hz to 13.0 Hz, in steps of 0.1 Hz, and the results are shown in Table 3. The system poles and zeros were found by noting the frequencies which produced maximum and minimum tip deflection, respectively. An inherent assumption in this technique is that the damping of the beam is very small (this fact was experimentally verified in an independent study [19]). The damping ratio calculations represented in the table are based on the assumption that excitation near a modal frequency will result in the response showing primarily only that particular modal frequency.

Table 2: FEM Results

Mode	Frequency (Hertz)
1	2.0091
2	8.2509
3	23.1187
4	46.5677
5	79.5244

Table 3: Frequency Response Data

Minimum Tip Response	Maximum Tip Response	Damping Ratio
3.0 Hz	1.2 Hz	0.139
10.3 Hz	7.6 Hz	0.050
11.3 Hz	12.0 Hz	0.008

The open loop step response (in position) of the arm was found by rotating the motor shaft through an angle of 10 degrees and measuring the tip deflection from its nominal value (initial point). After this maneuver the motor holds the new position (that is, is servoing) since the local feedback loop is active. Figure 2 illustrates a plot of the step response. Note that while the torque is applied at the hub at time $t = 0$, the tip deflection response is delayed by approximately 30 milliseconds and, in fact, initially moves in the direction opposite that of the hub rotation. The step responses indicates a settling time of about one minute. A fast Fourier transform of the data allows clear identification of the first two modal frequencies; these occur at 1.18 Hz and 7.5 Hz, respectively. Figure 3 shows the result of the FFT for the tip position in the step response test. We note that the rigid body mode (dc component) due to the pinned joint has been subtracted out of the FFT plot for clarity.

3. Control Analysis

3.1 Problem Formulation

Consider again the single link flexible manipulator system described above, redrawn in Figure 4. The displacement of any point along the arm is given by the hub angle $\theta(t)$ and the deflection $w(x,t)$ measured from the line $\overline{ox}$. We assume that only transverse vibration is present and that the deflection due to this vibration is small. Let L be the arm length so that in general terms

$$y(x,t) = w(x,t) + x\theta(t) \quad . \tag{1}$$

$$\frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right] = -m_d A\frac{\partial^2 w(x,t)}{\partial t^2} \quad . \tag{2}$$

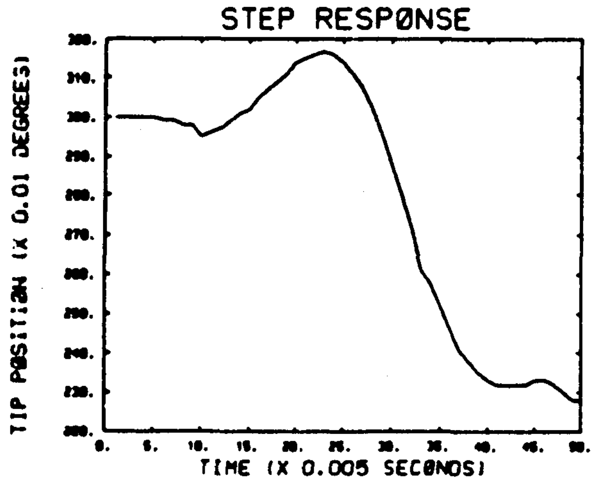


Figure 2

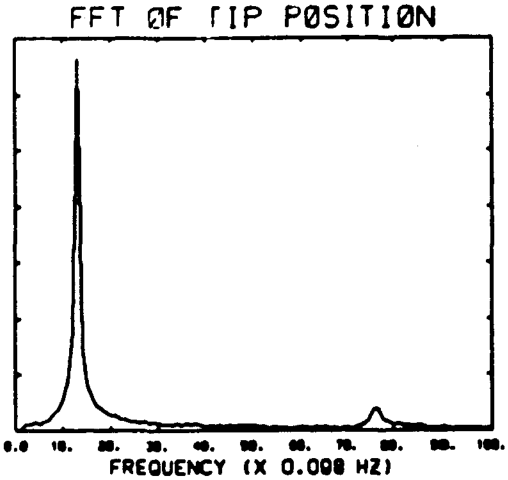


Figure 3

where $EI(x)$ is the elastic stiffness, A is the cross-sectional area, and m_l is the mass density. For the mechanical configuration under consideration, (2) must satisfy the boundary conditions

$$\phi(0,t) = 0 \quad ; \quad EI(x)\frac{\partial^2\phi}{\partial x^2}\bigg|_{x=0} + T - I_H\ddot{\theta} = 0 \quad ; \quad EI(x)\frac{\partial^2\phi}{\partial x^2}\bigg|_{x=L} = 0 \quad ; \quad EI(x)\frac{\partial^3\phi}{\partial x^3}\bigg|_{x=L} = 0 \quad , \tag{3}$$

where T is the torque at the hub and I_H is the actuator inertia. Accordingly, (2) may be put into the familiar form for the generalized modal coordinates $\mathbf{q}(t)$ as

$$M\ddot{\mathbf{q}} + D\dot{\mathbf{q}} + K\mathbf{q} = Bf \quad , \quad \dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n]^T \quad , \tag{4}$$

where M is the mass matrix, K is the stiffness matrix, and D contains terms associated with the damping. For position and velocity measurements (in the y direction) the solution to (4) is approximated by

$$y(x,t) = \sum_{k=0}^{n} \beta_k(t)\varphi_k(x) \tag{5}$$

for the eigenfunctions $\varphi_k(x)$, including the rigid body mode at $k = 0$, and the $\beta_k(t)$ are chosen to minimize the mean-square error upon substitution into (1).

Under the unitary transformation Φ, then, we use the notation $\mathbf{q}$ now to represent the state in the state variable formulation of (4) as

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -\Omega^2 & -\Phi^T D\Phi \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} 0 \\ \Phi^T B \end{bmatrix} f \quad , \quad e(t) = C\begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} \quad , \tag{6}$$

where $\Omega^2 = \Phi^T K\Phi$ is an $n \times n$ diagonal matrix containing the squares of the modal frequencies $\omega_1, \omega_2, \ldots, \omega_n$ along the diagonal, and $e(t)$ is the measurement vector. Upon rearrangement, we may write (6) in the manner

$$\begin{bmatrix} \dot{q}_o \\ \ddot{q}_o \\ \dot{q}_1 \\ \ddot{q}_1 \\ \vdots \\ \vdots \\ \dot{q}_n \\ \ddot{q}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & & & & & \\ 0 & 0 & & & & & \\ & & 0 & 1 & & & \\ & & -\omega_1^2 & -2\zeta_1\omega_1 & & & \\ & & & & \ddots & & \\ & & & & & \ddots & \\ & & & & & 0 & 1 \\ & & & & & -\omega_n^2 & -2\zeta_n\omega_n \end{bmatrix} \begin{bmatrix} q_o \\ \dot{q}_o \\ q_1 \\ \dot{q}_1 \\ \vdots \\ \vdots \\ q_n \\ \dot{q}_n \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ b_1 \\ \vdots \\ \vdots \\ 0 \\ b_n \end{bmatrix} f \quad . \tag{7}$$

We note that in (7) the rigid body mode has been included. Since the only control input for this example is the torque, then $f = T$. Finally, the matrices C and $\Phi^T B$ are given by

$$C = \begin{bmatrix} L & 0 & \varphi_1(L) & 0 & \cdots & \varphi_n(L) & 0 \\ 0 & 1 & 0 & \frac{d\varphi_1}{dx}(0) & \cdots & 0 & \frac{d\varphi_n}{dx}(0) \end{bmatrix} \quad , \quad \Phi^T B = \begin{bmatrix} 1 & \frac{d\varphi_1}{dx}(0) & \cdots & \frac{d\varphi_n}{dx}(0) \end{bmatrix}^T \tag{8}$$

3.2 Parameter Estimation

The fundamental issue in the mathematical formulation of flexible mechanical structures lies in the fact that such distributed parameter systems must be identified (controlled) with only a limited number of sensors (actuators). Indeed, for the analysis of the single-link flexible arm we typically consider hub actuation only, and tip position and/or hub velocity measurements to be employed in modeling and feedback control. Moreover, without reliable models for the control design the analysis becomes even more difficult. Philosophically, there are several different views to take in the control design. One approach is to construct a controller that is reasonably robust in the presence of modeling uncertainties and spillover, and yet simple enough in structure to be easily implementable (for example the variable structure control approach [20]). Another approach is to perform system identification exercises to model the system as accurately as possible prior to control design. A third approach is a combination of the first two: estimate the system parameters on-line (in the closed loop) and base the control design on the resulting model. This last viewpoint is often refered to as Self-Tuning Adaptive Control (STAC).

In the STAC approach, the manipulator dynamics are represented by linear discrete-time models, affording the primary advantage that the controller design is inherently digital in nature. In the application to flexible structures, tuning parameters include combinations of the damping and modal frequencies, or some combination of other free parameters which make up the manipulator model. Our approach to the parameter estimation problem involves recursive least squares methods with covariance resetting. That is, in order to maintain a fast overall convergence rate, the covariance of the estimates is reset at regular intervals in the algorithm. Such a scheme is particularly attractive for the manipulator control problem due to the time-varying nature of the tuning parameters during slew maneuvers and varying payload exercises. Experimental studies of the parameter estimation and STAC approach for the arm described above are presently underway. In the following we present simulation results which indicate avenues to pursue regarding implementation.

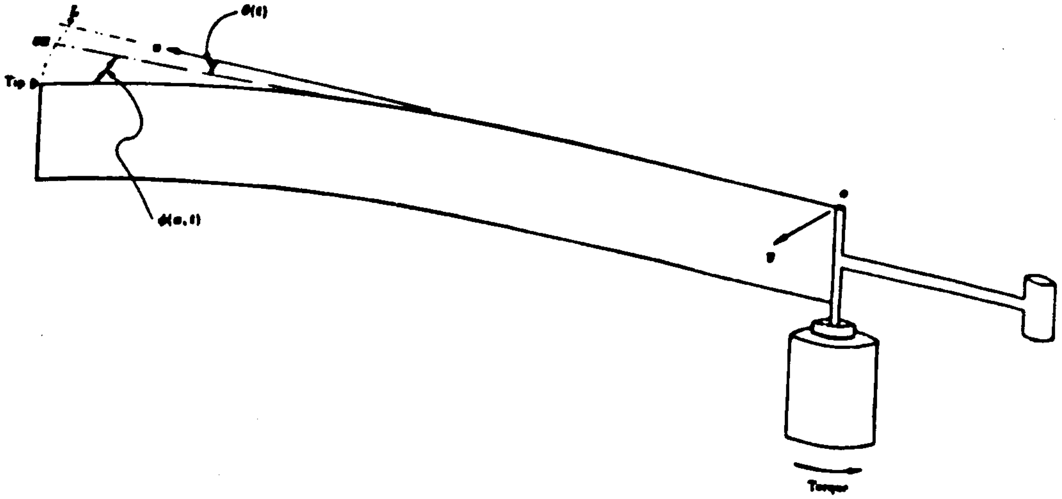


Figure 4: Deflection $\phi(x,t)$, Slew angle $\theta(t)$

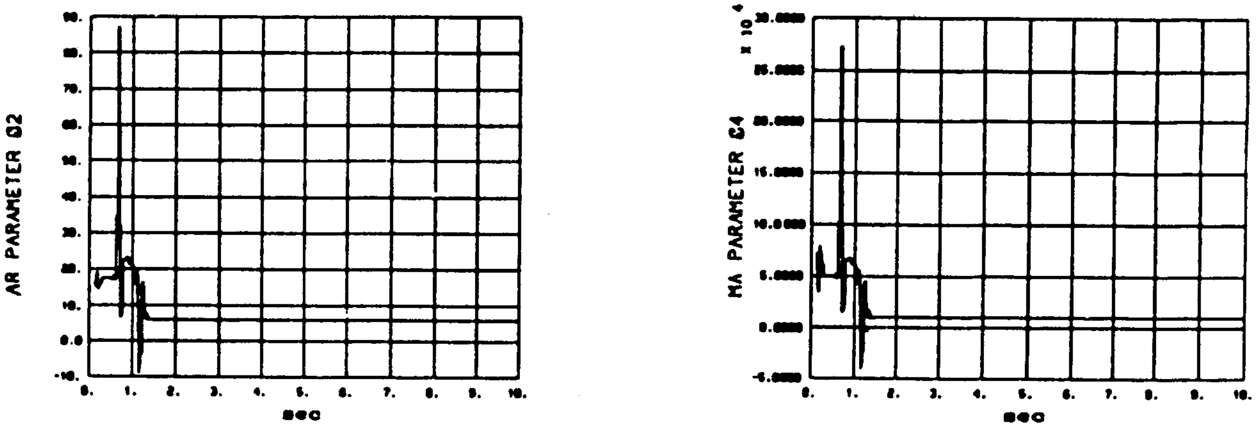


Figure 5: Parameter Estimation Simulation Results

The elements of the C matrix can be found according to [14]

$$\psi_i(L) = \left[\frac{d\psi_i}{dx}(0)\right]\omega_i\left[\frac{y}{d\psi_i/dt}\right] \tag{9}$$

and the $(d\psi_i/dt)(0)$ are solved from a system of nonlinear equations. The modal frequencies can be computed *a priori*, or identified as discussed in the previous section. We model the system as a stochastic ARMA process and excite a fourth-order model of the arm with a white noise input. Such a representation allows a delay (in tip position response, as observed in the actual system) to be inserted into the model. Using a zero-order hold circuit in the model and sampling the simulated FEM model, the AR and MA parameters converge to their nominal values as depicted in the sample plots of Figure 5, which shows time histories of one AR parameter and one MA parameter. Values for damping coefficients are then calculated from these parameters. These results are not useful in closed-loop control however, due to the length of time for convergence to the true parameters. Note also that a primary difficulty results because of the approximate pole-zero cancellation in the system model (indicated by the spike at about 0.7 seconds). Slightly better results are obtained if the rigid mode is removed from the model, which corresponds to exciting the *unforced* system with an initial disturbance. Such an exercise is possible since the motor inertia is considerably greater than that of the arm.

Prior to actual experimentation on the arm, several modifications must be investigated. For example, simulation studies for this and other example systems have indicated improvement for different resetting intervals; for details, the reader may wish to consult [21]. Also, simple digital low-pass filtering of the measured variables has produced improved performance of the parameter estimator. For control purposes the simulations have shown that an algorithm which turns on the control after allowing the estimator to run for a short period of time (for example, as illustrated in the simulations, about 1 to 1.5 seconds) will achieve the control objective. However, we are presently pursuing ways of improving the time to convergence in the closed loop with approaches using state feedback.

3.3 Output Feedback and Frequency Shaping

Generally speaking, high dimensionality and multiplicity of inputs in large-scale systems such as flexible mechanical structures leads to complex centralized controller schemes. One solution to this problem is to simplify the *structure* of the model via decomposition into subsystems with associated subcontrollers in a decentralized output feedback formulation. Moreover, centralized or decentralized output feedback is one of the more straightforward algorithms, from the viewpoint of implementation, for the control of flexible mechanical structures; see, for example, [22,23,20].

For the problem of single-link flexible manipulator control, where only hub actuation is employed in the control action, the output feedback approach to controller implementation is centralized in nature. The problem of spillover is, however, a critical issue to consider in the design. In order to minimize the effects of spillover, we consider a frequency-shaped cost functional [24], where penalties are assigned to the truncated modes and high penalties are assigned to the high harmonics at the input in order to minimize the effects due to excitation of the residual modes.

We consider the cost functional J to be mininized as formulated in the frequency domain utilizing Parseval's Theorem. With infinite time horizon, such a cost is written in the manner

$$J = \int_{-\infty}^{\infty}[X^*(j\omega)Q(j\omega)X(j\omega) + T^*(j\omega)R(j\omega)T(j\omega)]\,d\omega \quad , \tag{10}$$

where $X(j\omega)$ corresponds to the system state (as in (6)) and $T(j\omega)$ the input (torque) of the system. For implementation of such a scheme, consider the diagram of Figure 6, where the parameters K_1, K_2, K_3 are solved for in the minimization of (10), and the filter pole location (γ) is dependent on the system dynamics. In the example under consideration, $Q(j\omega)$ is the system matrix and $R(j\omega) = (j\omega + \gamma)(-j\omega + \gamma)$.

Under this formulation, the open-loop state variable representation of the system has the form

$$\begin{bmatrix} \dot{q}_o \\ \ddot{q}_o \\ \dot{q}_1 \\ \ddot{q}_1 \\ \vdots \\ \vdots \\ \dot{q}_n \\ \ddot{q}_n \\ \dot{T} \end{bmatrix} = \begin{bmatrix} 0 & 1 & & & & & & & 0 \\ 0 & 0 & & & & & & & 1 \\ & & 0 & 1 & & & & & 0 \\ & & -\omega_1^2 & -2\zeta_1\omega_1 & & & & & \frac{d\psi_1}{dx}(0) \\ & & & & \ddots & & & & \\ & & & & & \ddots & & & \\ & & & & & & 0 & 1 & 0 \\ & & & & & & -\omega_n^2 & -2\zeta_n\omega_n & \frac{d\psi_n}{dx}(0) \\ & & & & & & & & -\gamma \end{bmatrix} \begin{bmatrix} q_o \\ \dot{q}_o \\ q_1 \\ \dot{q}_1 \\ \vdots \\ \vdots \\ q_n \\ \dot{q}_n \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} r \quad , \tag{11}$$

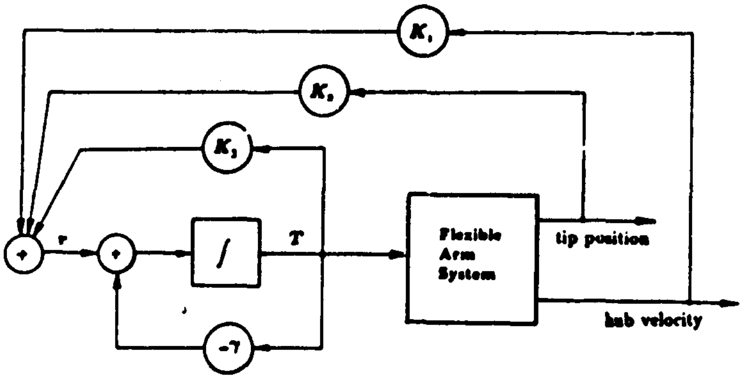


Figure 6: Output Feedback Scheme

$$\hat{v}(t) = \begin{bmatrix} L & 0 & \psi_1(L) & 0 & \cdots & \psi_n(L) & 0 & 0 \\ 0 & 1 & 0 & \frac{d\psi_1}{dx}(0) & \cdots & 0 & \frac{d\psi_n}{dx}(0) & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ T \end{bmatrix} \quad , \tag{12}$$

where the new measurement $\hat{v}(t)$ now includes the torque T. Incorporating the system of (11)-(12) into the output feedback structure of Figure 6 (and subsequent solution of the corresponding Lyapunov equation) allows the off-line calculation of the feedback gains from minimization of (10) by an appropriate nonlinear optimization routine.

We consider now a simulation of the flexible arm system, using a five-mode model from the FEM as the "truth model", from which measurements are taken and fed back in the output feedback scheme. The controller design for this example is based on the reduced system of rigid mode plus first flexible mode, and the resulting control is then tested against the full-order truth model to illustrate the effects of the frequency weighting approach in reducing spillover. A conjugate gradient method is used in the optimization portion of the design, and the final values obtained for the control law (with $\gamma = 4$) are $\tilde{K}_1 = -110.09$, $\tilde{K}_2 = -111.53$, $\tilde{K}_3 = -88.83$ (feasible values for the system under consideration) for the cost which reached a minimum after approximately 3000 iterations. The results using this controller are illustrated in Figure 7 for a step input torque; this applied input is such that the tip rotates through a small angle (of less than 5°, in terms of the rigid position). The values for torque begin at zero, that is, the dc component is subtracted out. In the simulation, the response settles in about ten seconds, whereas the free response decays after about one minute due to damping included in the model.

4. Structure-Mounted Proof-Mass Actuation

Since the large-angle slewing problem is complicated due to the flexibility effects inherent in the structure to be slewed, one is naturally led to investigate the possibility of *relegating*, at least partially, the task of vibration damping to a separate sensor-actuator pair and associated feedback loop. To this end, we have been investigating utilization of a structure-mounted momentum-exchange device mounted near the tip of the single-link manipulator. The practicality of such active vibration damping in a robotic environment is closely coupled to the availability of lightweight and effective devices. The device we have considered in our preliminary studies is a proof-mass actuator developed in our labs to study active vibration in space based flexible mechanical structures.

Non-ground referenced linear actuators are not yet widely available on the market, and this fact led to an in-house development; a general view of the device as mounted on the arm is shown in Figure 8. The device is built around a linear motor manufactured by the Kimco division of BEI Motion Systems which has a total mass of 25 grams, and can deliver a peak force of 2.2 N. The coil (solenoid) is rigidly mounted to a beam clamp which fixes the actuator to the arm. Also connected to the clamp is a rigid aluminum bracket which supports the springs. The proof mass is coupled to the framework through the springs, which are in turn coupled to the framework with adjusting screws so that their rest tension and proof mass rest position can be controlled. There is sufficient adjustment so that springs of different length and stiffness constant can be accomodated. The springs provide a restoring force for the proof mass and transfer force to the structure. A hanger was also mounted to provide strain relief for the feed wires.

The proof mass consists of a rectangular steel ring with a central steel member. This central member passes through the coil and restricts motion to a single axis. Samarium cobalt magnets are fixed to the top and bottom inside edges of the ring (adjacent to the coil) so that the interaction of the permanent magnetic field with a current in the coil results in a force on the proof mass.

Details of the development of the dynamic model of the actuator may be found in [17]. The net force applied to the tip of the arm (where the actuator is mounted) may be given as

$$f = 2kz - K_F i_a + B\dot{z} \quad , \tag{13}$$

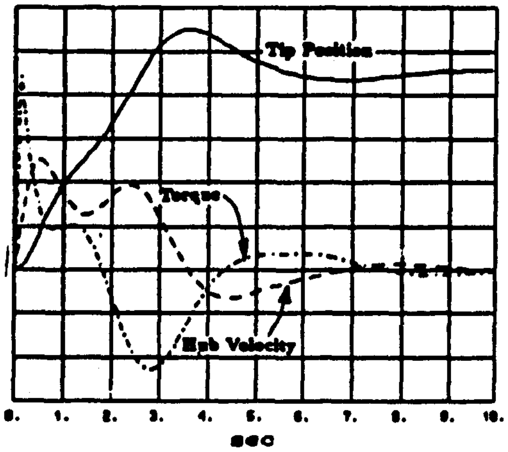


Figure 7: Output Feedback Example

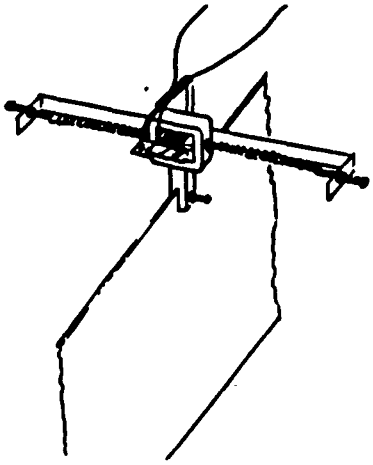


Figure 8: OSU Proof-Mass Actuator

$$m\ddot{z} + B\dot{z} + 2kz = K_F i + m\ddot{y} \quad , \tag{14}$$

where m is the proof mass, y the displacement of the structure at the point of actuator attachment, f the force acting at that point, z the relative displacement of the proof mass, B the viscous damping coefficient, K_F the motor force constant, and i, i_a are the input and armature currents, respectively. The actuator constants taken from the data sheets which accompany the individual components, are $m = 0.0207$ kg, $k = 262.7$ N-m^{-1}, $K_F = 1.112$ N-ampere^{-1}. The incorporation of the above actuator creates a second feedback loop to which the task of vibration damping is *relegated.* The two control loops (for slewing and for vibration damping) can be both designed and implemented in a decentralized manner. Note that y includes the displacement due to both the rigid body mode and the flexibility (see (1)). The principle of *relegation* implies that we design the feedback control only for the latter portion. To this end consider a vibration damping loop for only the first mode, such that the relevant expression is

$$\ddot{q}_1 + \omega_1^2 q_1 = b_1 f \quad , \tag{15}$$

where ω_1 is the natural frequency of the first mode and b_1 is an influence factor determined from the mode shape at the point of interest. Acceleration feedback can be used from the co-located accelerometer and a simple PI controller has been designed. It is evident, however, that the STAC approach or the frequency weighted control approach outlined earlier, can also be used here. The incorporation of the more sophisticated design approaches resulting in more complicated controllers will aid in handling more than the first vibrational mode. Studies along this direction are presently continuing.

5. Conclusion

In this workshop presentation we have described work in progress on modeling, parameter estimation, and control studies for an experimental, one-meter single-link flexible manipulator arm. *Models have been developed for the apparatus based on finite* element analysis and experimental verification. These, with the closed-loop parameter estimation procedures described here, and subsequent STAC approach for control, are being evaluated on the laboratory arm.

Under investigation is experimentation involving local proof-mass actuation for vibration control at the tip of the arm, using a device developed in the Control Research Laboratory at Ohio State for flexible structures control work. The output feedback frequency shaping approach described here may be easily extended to this application, where the formulation is *decentralized* in nature; results on this technique for general flexible structure vibration control will appear in [25]. Finally, various other centralized (for the case of hub actuation only) and decentralized approaches are currently being evaluated in the laboratory.

References

[1] H. Kanoh and H. Lee, "Vibration control of a one-link flexible arm," in *Proceedings of the 24th Conference on Decision and Control*, pp. 1172-1177, Ft. Lauderdale, Florida, December 1985.

[2] P. Kärkkäinen, "Compensation of manipulator flexibility effects by modal space techniques," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 972–977, St. Louis, MO, April 1985.

[3] R. Marino and M. W. Spong, "Nonlinear control techniques for flexible joint manipulators: A single link case study," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1030–1036, San Francisco, April 1986.

[4] A. Barraco, B. Curry, and G. Ishiomin, "Dynamic control for flexible robots: Different approaches," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1038–1043, San Francisco, April 1986.

[5] R. P. Judd and D. R. Falkenburg, "Dynamics of nonrigid articulated robot linkages," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 5, pp. 499–502, May 1985.

[6] K. Khorasani and M. W. Spong, "Invariant manifolds and their application to robot manipulators with flexible joints," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 978–983, St. Louis, MO, April 1985.

[7] A. D. Luca, A. Isidori, and F. Nicolo, "Control of a robot arm with elastic joints via nonlinear dynamic feedback," in *Proceedings of the 24th Conferences on Decision and Control*, pp. 1671–1679, Ft. Lauderdale, December 1985.

[8] W. J. Book and M. Majette, "Controller design for flexible, distributed parameter mechanical arms via combined state space and frequency domain techniques," *Journal of Dynamic System, Measurement, and Control*, vol. 105, pp. 245–254, December 1983.

[9] G. G. Hastings and W. J. Book, "Experiments in optimal control of a flexible arm," in *Proceedings of the 1985 American Control Conference*, pp. 728–729, Boston, MA, June 1985.

[10] G. G. Hastings and W. J. Book, "Verification of a linear dynamic model for flexible robotic manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1024–1029, San Francisco, CA, April 1986.

[11] W. J. Book, S. Cetinkunt, and G. W. Woodruff, "Near optimum control of flexible robot arms on fixed paths," in *Proceedings of the 24th Conference on Decision and Control*, pp. 1522–1528, Ft. Lauderdale, December 1985.

[12] D. Schaechter, "Hardware demonstration of flexible beam control," *Journal of Guidance and Control*, vol. 5, no. 1, pp. 48–53, January 1982.

[13] D. Eldred and D. Schaechter, "Experimental demonstration of static shape control," *Journal of Guidance and Control*, vol. 6, no. 3, pp. 188–192, May 1983.

[14] R. H. Canon and E. Schmitz, "Initial experiments on the end-point control of a flexible one-link robot," *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 62–75, 1984.

[15] J. Juang, L. G. Horta, and H. H. Robertshaw, "A slewing control experiment for flexible structures," in *Proceedings of the Fifth VPI & SU/AIAA Symposium on Dynamics and Control of Large Structures*, Blacksburg, VA, June 1985.

[16] J. N. Juang, J. D. Turner, and H. M. Chun, "Closed-form solutions for feedback control with terminal constraints," *Journal of Guidance and Control*, vol. 8, no. 1, pp. 39–43, January 1985.

[17] J. Martin, U. Özgüner, and S. Yurkovich, "An active vibration damper for flexible structures," in *Proceedings of the Seventeenth Annual Pittsburgh Conference on Modeling and Simulation*, April 1986.

[18] R. W. Gorman, *ANSYS Engineering Analysis System PC/ED, User's Manual.* Swanson Analysis Systems, Inc., 1985.

[19] R. J. Ulman, "System identification studies for flexible structures," M.S. Thesis, The Ohio State University, 1986.

[20] U. Özgüner, S. Yurkovich, J. Martin, and F. Al-Abbass, "Decentralized control experiments on NASA's flexible grid," in *Proceedings of the 1986 American Control Conference*, pp. 1045–1051, Seattle WA, June 1986.

[21] A. Tzes and S. Yurkovich, "A sensitivity analysis approach to the control of manipulators with unknown load," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Raleigh, NC, March 1987. (to appear).

[22] D. Schaechter, "Optimal local control of flexible structures," *Journal of Guidance and Control*, vol. 4, no. 1, pp. 22–26, January 1981.

[23] S. Yurkovich, U. Özgüner, and F. Al-Abbass, "Sliding mode, model reference adaptive control for flexible structures," Technical Report CRL-1008-Su86-P, The Ohio State University, Control Research Laboratory, 1986.

[24] N. K. Gupta, "Frequency-shaped cost functionals: Extension of linear-quadratic-Gaussian design methods," *Journal of Guidance and Control*, vol. 3, no. 6, pp. 529–535, November 1980.

[25] U. Özgüner and S. Yurkovich, "Decentralized frequency shaping and model sensitivities for optimal control of large space structures," in *Proceedings of the 10th IFAC World Congress*, Munich, West Germany, July 1987. (to appear).

S28-37

98

188197

Dual Arm Robotic System With Sensory Input

Ü. Özgüner
Ohio State University
Columbus, OH 43210

9M 573208

1. Abstract

The need for dual arm robots in space station assembly and satellite maintainance is of increasing significance. Such robots will be in greater demand in the future when numerous tasks will be assigned to them to relieve the direct intervention of humans in space. Technological demands from these robots will be high. They will be expected to perform high speed tasks with a certain degree of autonomy. Various levels of sensing will have to be used in a sophisticated control scheme.

~~In this presentation we will briefly describe~~ ongoing research in control, sensing and real-time software to produce a two-arm robotic system that can accomplish generic assembly tasks. T~~he paper will concentrate mostly on~~ the control hierarchy, the specific control approach selected being the Variable Structure (Sliding Mode) Control approach. ~~We will consider~~ a decentralized implementation of model-reference adaptive control using Variable Structure controllers and the incorporation of tactile feedback into it is considered.

are [illegible]

is discussed.

2. Introduction

The need for dual arm robots in space station assembly and satellite maintainance is of increasing significance. Such robots will be in greater demand in the future when numerous tasks will be assigned to them to relieve the direct intervention of humans in space. Technological demands from these robots will be high. They will be expected to perform high speed tasks with a certain degree of autonomy. Various levels of sensing will have to be used in a sophisticated control scheme.

In this presentation we will briefly describe ongoing research in control, sensing and real-time software to produce a two-arm robotic system that can accomplish generic assembly tasks. The paper will concentrate mostly on the control hierarchy, the specific control approach selected being the Variable Structure (Sliding Mode) Control approach. We will consider a decentralized implementation of model-reference adaptive control using Variable Structure controllers and the incorporation of tactile feedback into it.

We assume that multi-arm robotic operations have a hierarchical/decentralized control structure. However, the appropriate control algorithms have to be chosen for feedback to properly fit the special hierarchy of multiple robots with dextrous end effectors. A specific control approach has to be selected, and its requirements can be clearly specified:

- It must easily decompose into a hierarchy.
- It must be ameanable for modular implementation.
- It must posess low real-time computation requirements.
- It must be able to receive changes from sensor data.
- It must be insensitive to modeling errors and load variations.

It is expected that robotic systems will become an important part of future space missions. Orbital Maneuvering Vehicles have been proposed with dual arm systems for space station assembly, satellite servicing, etc.. Although the importance of dual arm robotic systems have been recognized for some time, little work of a general nature has been done in controlling such systems.

Early multi-processor robotic controllers were based on the principle of a simpler low-level processor and a more sophisticated high-level computer. This made interfacing *fairly difficult* and *expansion* almost impossible. With today's processors and appropriate software load distribution, tasks at all levels can be handled by processors of the same family. Coordination of data transfers are extremely simplified. It is apparent that certain improvements will have to be made over conventional control structures (as used say, in the PUMA) if there is hope of accomplishing sophisticated assembly type operations using multiple manipulators. For versatile performance the control hierarchy will exhibit a finer task decomposition. Tasks will have to be relegated to a large number of processors. Sensory inputs will have to be appropriately ascigned.

The control of robots in a precise, reliable and repeatable manner is by itself a hard problem. The problem becomes somewhat more complicated when considering the control of coordinated robot arms. Limited work has been done in the area of multi-arm robot systems [1] [2], [3], [4], [5], [6] and [7].

A method developed for controlling manipulator arms by Young [8], Özgüner and co workers [9], [10], and others [11] utilising variable structure control theory is particularly ameanable to extension to multiple arm systems controlled within a hierarchical framework. Initial work along these lines have already been performed. In this paper we will be reporting on recent developments in the above approach and especially tactile sensing feedback from the end effector as included in the hierarchical control structure.

There appear to be certain generic tasks that are imbedded in many assembly and maintainance operations. These include:

- Pick and place type tasks.
- Pin in the hole type tasks.
- Combined rotation-translation type tasks.

Many complex operations can be partitioned into combinations of these generic tasks. Thus the control algorithm design and related software will concentrate on the above tasks.

Figure 1 summarizes the control hierarchy to be used. At Level I, parsing interpreting and decoding user commands and high level sensory input are accomplished. Error messages to the user are also generated at this level. Level II includes trajectory planning, associated coordinate system transforms and analysis of bounds of the workspace. Joint-level coordination and transfer of information required by control algorithms is carried out at Level III. At Level IV, generation of the feedback control and I/O with actuators and force sensing is accomplished. The control algorithm selected has to be strongly coupled to the information structure selected. The algorithm must be decomposable into the hierarchy imposed and inherently adaptive to load and trajectory variations. The algorithm/control approach we are utilizing is the *Decentralized Model Reference* Adaptive approach using Variable Structure (sliding mode) controllers. It appears that this algorithm with appropriate modifications to accomodate sensory input and user commands can be mapped onto a multiprocessor system.

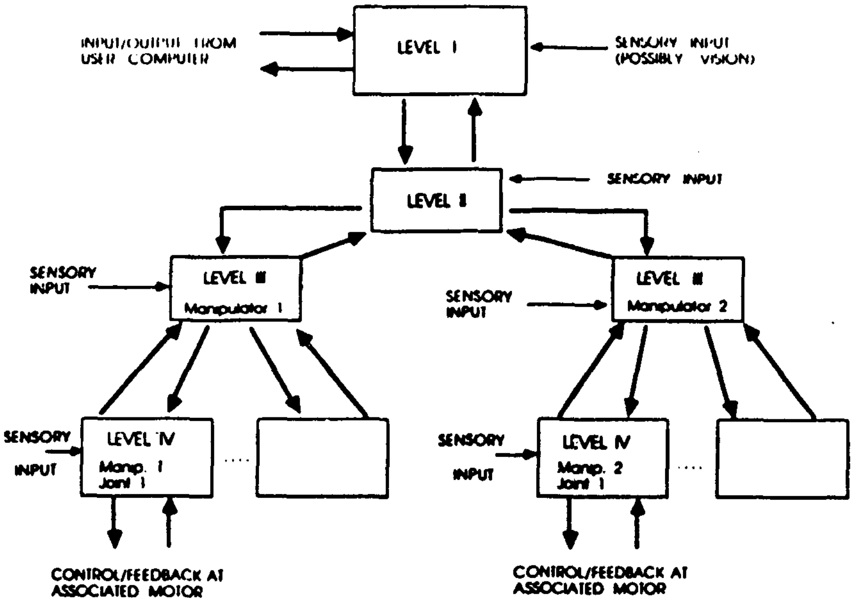


Figure 1: The General Hierarchy for the Control of Two Manipulators

Various results have been recently reported in the utilisation of sensory information from the end–effector in the feedback control structure. In the present work we will be using the force feedback approach (tactile sensing) as reported in [12]. The incorporation of force feedback into the control algorithm is not straight–forward, and in the next section we will introduce the concept of *interaction compensation* which will aid in the analysis.

3. The Concept of Interaction Compensation

It has been previously claimed that the levels of a hierarchy (as in the multi-manipulator system of Figure 1), can be classified so that one identifies "increasing intelligence with decreasing precision", as one moves up [13]. As with most labeling schemes, this may be an over–generalisation and there may be numerous cases where proper *relegation* of (a) Control Authority, and (b) Information Distribution, may result in preferrable operation of the over–all system.

We will consider the regulation of an interconnected system to introduce the concept of Interaction Compensation, which we will subsequently apply to the specific case of multi–manipulator control; under a fixed Control Authority structure and control algorithm.,

Consider a large–scale system consisting of N interconnected subsystems each defined by

$$\dot{x}_i = A_i(x_i)x_i + B_i(x_i)u_i + E_i(x) \tag{1}$$

$$y_i = D_i x_i \quad , \tag{2}$$

for $i = 1,2,\ldots,N$, where $x_i \in \Re^{n_i}$, $u_i \in \Re^1$, $y_i \in \Re^1$, where $\Re$ represents the real Euclidean vector space, $x^T = (x_1^T, x_2^T, \ldots, x_N^T)$ and the matrices are of compatible dimension. $E_i(x)$ denotes the totality of interaction effects from the remaining subsystems to subsystem i. Note that $E_i(x)$ may also include modeling errors.

Let us first define what is meant by *insensitivity to interaction.* Consider again (1)–(2), rewritten for brevity as

$$\dot{x}_i = f(x_i, u_i) + \Delta(x,t) \quad , \tag{3}$$

for the state transition mapping $f : \Re^{n_i} \times \Re \longrightarrow \Re^{n_i}$ and the total interaction term $\Delta(x,t)$. The system (3) is said to be *insensitive* to interaction effects if the solution $x_i(t)$ may be expressed as

$$x_i(t) = \tilde{x}_i(t) + \mathcal{O}(\epsilon_i) \quad ,$$

where $\tilde{x}_i(t)$ solves $\dot{x}_i = f(x_i, u_i)$, for all $\epsilon_i > 0$ and all $t > T$, for some finite time T, and $\mathcal{O}(\epsilon_i)$ represents terms of degree two or higher in ϵ_i.

In reference to measurements available for use at the control inputs of subsystem i we can now consider three possibilities:

1. Full (real-time) interaction information.
2. Partial interaction information.
3. Interaction modeling.

It can be shown that interaction information provides the opportunity of directly negating all effects within the range space of B_i. The more interesting cases are when partial information is available or can be *generated* through dynamic modeling of the interactions.

Given the model, the decentralized control problem is to design a controller to feedback locally available real–time information such that the states of each local subsystem are regulated to zero or track the states of a local reference model.

Let the local reference model for the i-th subsystem be given as

$$\dot{\hat{x}}_i = \hat{A}_i \hat{x}_i + \hat{B}_i r_i \tag{4}$$

$$\hat{y}_i = \hat{x}_i \quad , \tag{5}$$

with $\hat{x}_i \in \Re^{n_i}$, $\hat{A}_i \in \Re^{n_i \times n_i}$, $\hat{B}_i \in \Re^{n_i \times 1}$, where r_i is a scalar reference input. Define the *local* error between the ith subsystem and its local reference model in the manner

$$e_i = \hat{x}_i - x_i \tag{6}$$

so that the local error system dynamics may be written in the form

$$\dot{e}_i = \hat{A}_i e_i + (\hat{A}_i - A_i)x_i + \hat{B}_i r_i - B_i u_i - E_i(x) \quad . \tag{7}$$

Within this framework, assume that

- **Each local controller design is dependent only on the local model.**
- **The systems (1)-(2) and (4)-(5) are controllable.**
- **The states x_i and $\hat{x}_i$ are measureable locally for feedback to the ith input.**
- **Reference trajectory information may be fed to a subsystem from a higher level coordinator but interaction information is only partially available in real-time , and the subsystem is not allowed to communicate with the other local controllers**

4. Variable Structure Controllers

In this presentation we are going to assume that the basics of Variable Structure Control are known. An important feature of Variable Structure Controllers is the fact that, for the decentralised case, the local subsystem is made insensitive not only to local parameter changes but also to dynamical interactions with neighboring subsystems once the sliding surface is reached.

We define the sliding surface corresponding to the system (1)-(2) as

$$\sigma_i = C_i e_i \quad , \tag{8}$$

for C_i in $\Re^{1\times n_i}$. Invariance properties of the sliding surface were given in [14,15]. Refering back now to the error system (7), the control law is formulated in the manner

$$u_i = K_{e_i} e_i + K_{x_i} x_i + K_{r_i} r_i + \delta_i \quad , \tag{9}$$

where K_{r_i} in $\Re^1$, K_{x_i} in $\Re^{1\times n_i}$, and K_{e_i} in $\Re^{1\times n_i}$ can be specified in different regions of the state space, and where δ_i is usually a constant picked according to the norm of the interactions. The elements K_{r_i}, K_{x_i}, K_{e_i}, and δ_i are discontinuous functions of the sliding surface and the coefficients of system and reference model state equations. We furthermore claim that estimates for δ_i can be refined with knowledge on interactions. In the following we derive the appropriate forms based on the reaching condition which, in view of (7), (8) and (9), becomes

$$\begin{aligned} \sigma_i\dot{\sigma}_i &= C_i[\hat{A}_i - B_i K_{e_i}]e_i\sigma_i + C_i\{[\hat{A}_i - A_i] - B_i K_{x_i}\}x_i\sigma_i \\ &+ C_i[\dot{B}_i - K_{r_i} B_i]r_i\sigma_i - C_i\{B_i\delta_i + E_i(x)\}\sigma_i \\ &< 0 \end{aligned} \tag{10}$$

The condition (10) is satisfied provided that the gain parameters and δ_i are chosen so as to make each term negative. Thus,

$$(K_{e_i})_\ell = \alpha_{i\ell}(C_i B_i)^{-1}\left\{\sum_{j=1}^{n_i} c^i_j \hat{a}^i_{j\ell}\right\}\text{sgn}[(e_i)_\ell\sigma_i] \quad ; \tag{11}$$

$$(K_{x_i})_\ell = \beta_{i\ell}(C_i B_i)^{-1}\left\{\sum_{j=1}^{n_i} c^i_j(\hat{a}^i_{j\ell} - a^i_{k\ell})\right\}\text{sgn}[(x_i)_\ell\sigma_i] \quad ; \tag{12}$$

$$K_{r_i} = \left\{\gamma_i(C_i B_i)^{-1} C_i \dot{B}_i\right\}\text{sgn}[r_i\sigma_i] \quad ; \tag{13}$$

where $\alpha_{i\ell}$, $\beta_{i\ell}$, γ_i are positive constants, and $(\,\cdot\,)_\ell$ represents the ℓth element of the indicated vector. Let

$$\Delta_i = \max \;\| E(x) \| \quad , \tag{14}$$

and assuming that the *only* locally available information is Δ_i, we can pick

$$\delta_i = \Delta_i (C_i B_i)^{-1} \| C_i \| \,\text{sgn}[\sigma_i] \quad , \tag{15}$$

On the other hand, if the interaction effects are split into two portions; namely an unknown (but with known bound) portion, and a measurable portion, as given below:

$$E_i(x) = E_{iu}(x) + E_{im}(x) \quad , \tag{16}$$

the concept of *interaction compensation* can be utilised to directly negate the effects of the measured portion. Furthermore, if for specific applications, the measurable interactions are to assume desired values, or follow prespecified trajectories in time, they can be included easily into the model-reference framework above.

5. Robotic System Configuration and Modeling

The system under consideration consists of two different robotic arms, each a planar three-link manipulator (Figure 2). The parameters of each link are shown in the figure where

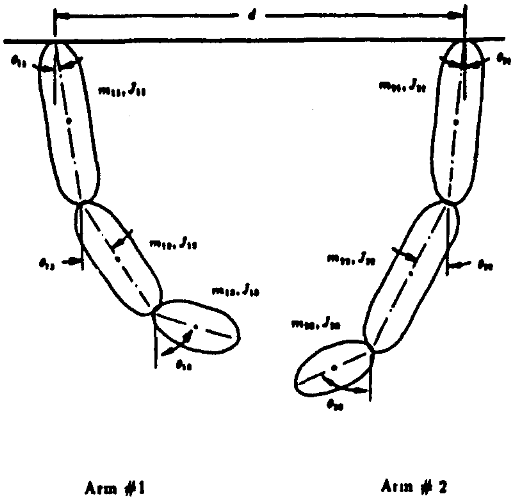


Figure 2: Two three-link, planar manipulators.

L_{ij} $\triangleq$ Length of each link $i = 1,2; j = 1,2,3$
ℓ_{ij} $\triangleq$ Location of center of gravity with respect to the end of the previous link
m_{ij} $\triangleq$ Mass of ij-th link
J_{ij} $\triangleq$ Moment of inertia about the corresponding center of gravity
θ_{ij} $\triangleq$ Angular position measured counterclockwise
T_{ij} $\triangleq$ Torque actuating the ij-th joint
d $\triangleq$ Distance separating both arms (on the base)

The overall task can be divided into three phases: approaching phase, grasping phase, and lifting (coordination) phase. In the approaching phase each arm moves toward the object to be picked. Speed and position control are applied according to the characteristics of each arm. The grasping phase design, although not considered in our study, depends on the sensory system assumed to be available. Force sensing can be utilized to implement a controller using force feedback. The last phase is the lifting phase during which the two-arm robotic system forms a closed-chain mechanical manipulator. Again, tactile feedback can be applied at this phase and we will discuss incorporation of such *interaction information* below. Lagrange's method has been used in finding the dynamical equations of this robotic system, where the equations are written in terms of the total kinetic energy (K) of the system, the total potential energy (P) of the system, and a set of independent coordinates (q_i) chosen to describe the configuration of the system. Furthermore, these equations may include dissipation functions for non-conservative systems. We will not present these equations here but just briefly analyze the conditions while the two arms are in contact. For the first arm, let

$$\begin{aligned} \theta_1 &= (\theta_{11}\ \theta_{12}\ \theta_{13})^t \\ T_1 &= (T_{11}\ T_{12}\ T_{13})^t \quad , \end{aligned} \tag{17}$$

It can then be shown that, in the approaching phase the dynamical equations of the first arm can be written as

$$M_1\ddot{\theta}_1 + F_1(\theta_1\,,\,\dot{\theta}_1)\dot{\theta}_1^2 + G_1\theta_1 = T_1 \quad , \tag{18}$$

where $\dot{\theta}_1^2 = (\dot{\theta}_{11}^2, \dot{\theta}_{12}^2, \dot{\theta}_{13}^2)^t$.

On the other hand, during the grasping and holding phases, a closed chain robot is formed through the continuous contact of the end effectors of both arms with the object. This constraint defines a frictionless manifold which can be expressed in closed form. It is assumed that an additional torque (to be denoted as τ_1) can be defined to maintain the tip of the end effector on the manifold. The dynamical equations of the closed chain may then be written as

$$M_1\ddot{\theta}_1 + F_1(\theta\,,\,\dot{\theta}_1)\dot{\theta}_1^2 + G_1\theta_1 = T_1 + \tau_1 \quad . \tag{19}$$

To find the equation for τ_1, consider a vertical displacement (δs). The corresponding work done by τ_1 is zero; that is,

$$\begin{aligned} \delta W &= \tau_1 \delta s \\ &= -F_{14}^x \delta x + F_{14}^y \delta y \quad , \end{aligned} \tag{20}$$

where F_{14} is a generalized force due to the contact. The above equation can then be expressed in terms of the joint angles since

$$\begin{aligned} x &= L_{11}\sin\theta_{11} + L_{12}\sin\theta_{12} + L_{13}\sin\theta_{13} \\ y &= -L_{11}\cos\theta_{11} - L_{12}\cos\theta_{12} - L_{13}\cos\theta_{13} \\ \delta x &= L_{11}\cos\theta_{11}\delta\theta_{11} + L_{12}\cos\theta_{12}\delta\theta_{12} + L_{13}\cos\theta_{13}\delta\theta_{13} \\ \delta y &= L_{11}\sin\theta_{11}\delta\theta_{11} + L_{12}\sin\theta_{12}\delta\theta_{12} + L_{13}\sin\theta_{13}\delta\theta_{13} \quad . \end{aligned}$$

Substituting the above into (20) results in

$$\tau_1 = \frac{\delta\omega}{\delta\theta} = \begin{bmatrix} (F_{14}^x\cos\theta_{11} + F_{14}^y\sin\theta_{11})\,L_{11} \\ (F_{14}^x\cos\theta_{12} + F_{14}^y\sin\theta_{12})\,L_{12} \\ (F_{14}^x\cos\theta_{13} + F_{14}^y\sin\theta_{13})\,L_{13} \end{bmatrix} = \begin{bmatrix} \tau_{11} \\ \tau_{12} \\ \tau_{13} \end{bmatrix} \quad . \tag{21}$$

Furthermore, since F_{14}^x and F_{14}^y are along the direction of motion and normal to it, they are related by $F_{14}^y = \mu F_{14}^x$ where $\mu \leq 1$ is the coefficient of friction.

Following the same procedure used for deriving the equation of the first arm, one can easily find the dynamical equations of the second arm. Furthermore, a similar equation to (21) can be found for the second arm to satisfy the coordination movement constraint. The decentralized model-reference adaptive controller is now utilised to control the robotic system. To this end, each link is considered an independent subsystem with coupling forces and/or torques being the interactions. Thus,

$S_1 \triangleq$ Link #1 of the first arm with $X_1 = (x_{11}\ x_{12})^t = (\theta_{11}\ \dot{\theta}_{11})^t$
$S_2 \triangleq$ Link #2 of the first arm with $X_2 = (x_{21}\ x_{22})^t = (\theta_{12}\ \dot{\theta}_{12})^t$
$S_3 \triangleq$ Link #3 of the first arm with $X_3 = (x_{31}\ x_{32})^t = (\theta_{13}\ \dot{\theta}_{13})^t$
$S_4 \triangleq$ Link #1 of the second arm with $X_4 = (x_{41}\ x_{42})^t = (\theta_{21}\ \dot{\theta}_{21})^t$
$S_5 \triangleq$ Link #2 of the second arm with $X_5 = (x_{51}\ x_{52})^t = (\theta_{22}\ \dot{\theta}_{22})^t$
$S_6 \triangleq$ Link #3 of the second arm with $X_6 = (x_{61}\ x_{62})^t = (\theta_{23}\ \dot{\theta}_{22})^t$.

Each subsystem of the above has the following general form:

$$\dot{X}_i = A_i(X_i, t)X_i + B_iU_i + \sum_{\substack{j=1 \\ j\neq i}}^{6} A_{ij}(X,t)X_j \quad , \tag{22}$$

where $U_i = T_i$. Detailed derivations of these equations in the above form may be found elsewhere [16]. One can note that any measured torque or force betwean the links can be incorporated into the control structure discusseu previously.

6. Controller Design

The use of Variable Structure Controllers for robotic system control was introduced by Young [8]. Among more recent work in the area one can cite those of Morgan and Özgüner [9] where *decentralized* controllers were employed, of Slotine and Sastry [11] who dwell on the reduction of chattering, and of Young [14] who introduces the design of variable structure model-following control systems. The present work differs from the above in that it uses a Decentralised, Model Reference Adaptive approach and specifically addresses the multiple manipulator control problem. The controller devised for this robotic system is organised in a hierarchical framework. The levels of the hierarchy are divided into two, and the information processed at each level is not directly available to the other level. Figure 3 shows the levels and information flow of the hierarchy, indicating that there are two paths of information flow. Downward moving data presents the flow of commands while upward moving data presents the flow of feedback information. As shown in Figure 4, three tasks are defined at upper level: planning the motion of the end effectors of both arms, defining the local reference inputs for each link, and finding the upper bound of the dynamical interactions between subsystems. Figure 4 schematically depicts the operations done at this level.

The end effectors of both arms are required to move in the work-space in a specific way. A path is a continuous curve in the system workspace connecting the tip initial configuration to the final configuration through all intermediate configurations. On the other hand, a trajectory is a continuous curve in the state space of every link joining the initial state to the final state. In other words, the trajectory contains all the information about the time history of position, velocity, and acceleration for each link. Therefore, a trajectory includes not only a path but also velocity and acceleration at every point at the path.

The first step in generating the two arm robot motion is to characterise the path in some manner, typically by applying physical intuition to some extent. Esentially, the arm should start and stop slowly with a smooth motion. A number of different trajectories may be proposed to satisfy the requirements, such as exponential and polynomial trajectories.

In the present study, the following equations were used

$$x(t) = x(t_f) + [x(t_o) - x(t_f)]e^{-a(t-t_o)^n} \quad (23)$$

$$y(t) = y(t_f) + [y(t_o) - y(t_f)]e^{-b(t-t_o)^m} \quad , \quad (24)$$

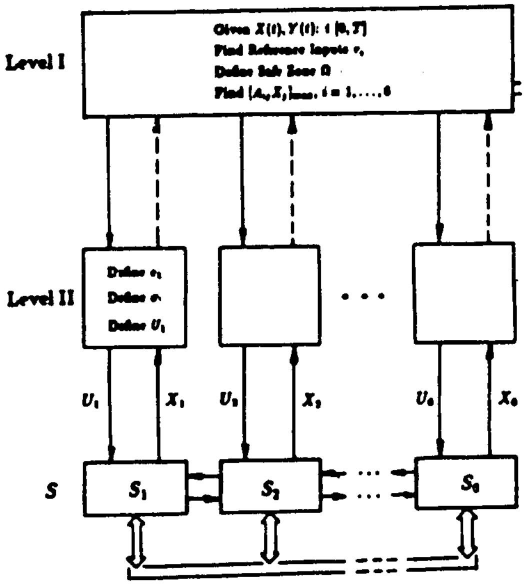
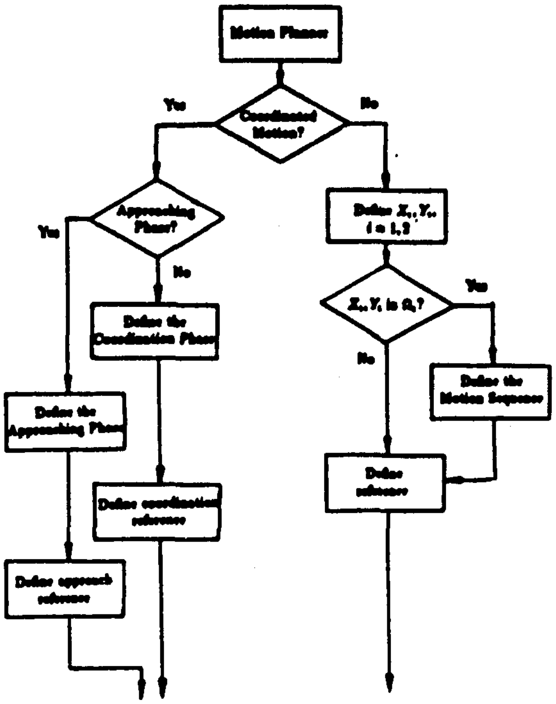


Figure 3: Control Hierarchy



Motion Planner
Yes
Coordinated Motion?
No
Approaching Phase?
Yes
No
Define X_i, Y_i, $i = 1, 2$
X_i, Y_i in Ω_i?
Yes
Define the Coordination Phase
No
Define the Motion Sequence
Define the Approaching Phase
Define reference
Define coordination reference
Define approach reference

Figure 4: High Level Controller

where a, b, m, and n are real numbers. In finding the reference input r_i for every link of each arm, the inverse dynamics approach is utilized. Note that, in the coordination (actual lifting) phase, both end effectors move on the same path. During this phase, if a desired contact force profile is required, this can also be cast in the framework of reference generation in the given formulation.

The low level controller consists of individual controllers for each link. Each controller is designed following the model reference adaptive, variable structure system approach. In this level, each subsystem is controlled separately to follow the reference model. This is to be done using only local information such as position and speed information of both the subsystem and the corresponding local reference model. Furthermore, the local controller required to force the local states to follow the states of the corresponding reference model is designed using the VSS approach presented earlier. Further details and simulation studies of cases without tactile feedback may be found in Ref. [16].

7. Conclusion

In this presentation we have briefly described ongoing research in control, sensing and real-time software to produce a two-arm robotic system that can accomplish generic assembly tasks. We have concentrated mostly on the control hierarchy, the specific control approach selected being the Variable Structure (Sliding Mode) Control approach. The decentralised implementation of model-reference adaptive control using Variable Structure controllers was shown to be particularly suitable for such an application and the incorporation of tactile feedback was possible. Research is presently continuing on adjustments in the feedback gains when a desired torque/force profile is given for end-effectors in contact with each other or other external surfaces.

8. References

[1] S. Hayati, "Hybrid position/force control of multi-arm cooperating robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 82–89, San Francisco, April 1986.

[2] E. Freund, "On the design of multirobot systems," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 477–490, Atlanta, GA, April 1984.

[3] J. Lim and D. H. Chyung, "On a control scheme for two coordinating robot arms," in *Proceedings of the 24th Conference on Decision and Control*, pp. 334–335, Ft. Lauderdale, Florida, December 1985.

[4] Y. F. Zheng and J. Y. S. Luh, "Control of two coordinated robots in motion," in *Proceedings 24th IEEE Conference on Decision and Control*, pp. 1761–1765, Ft. Lauderdale, FL, December 1985.

[5] Y. F. Zheng and J. Y. S. Luh, "Joint torques for control of two coordinated moving robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1375–1380, April 1986.

[6] A. J. Koivo, "Adaptive position–velocity–force control of two manipulators," in *Proceedings of the 24th Conference on Decision and Control*, pp. 1529–1532, Ft. Lauderdale, FL, December 1984.

[7] C. Alford and S. Belyeu, "Coordinated control of two robot arms," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 468–473, Atlanta, GA, April 1984.

[8] K. K. D. Young, "Controller design for a manipulator using theory of variable structure systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-8, pp. 101–105, February 1978.

[9] R. Morgan and U. Özgüner, "A decentralized variable structure control algorithm for robotic manipulators," *IEEE Journal of Robotics*, vol. 1, no. 1, pp. 57–65, 1985.

[10] Ümit Özgüner, S. Yurkovich, and F. Al-Abbass, "Decentralized variable structure control for a two–arm robotic system," Technical Report CRL-1018-Au86-P, The Ohio State University, Control Research Laboratory, 1986. (submitted for publication).

[11] J. H. Slotine and S. S. Sastry, "Tracking control of non-linear systems using sliding surfaces, with application to robot manipulators," *International Journal of Control*, vol. 38, no. 2, pp. 465–492, 1983.

[12] C. Wongchaisuwat, J. D. Donne, Ümit Özgüner, and H. Hemami, "Control of a planar arm by nonlinear feedback gains," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 157–167, 1984.

[13] G. N. Saridis, "Intelligent robot control," *IEEE Transactions on Automatic Control*, vol. AC-28, no. 5, p. , May 1983.

[14] K. K. D. Young, "Design of variable structure model-following control systems," *IEEE Transactions on Automatic Control*, vol. AC-23, no. 6, pp. 1079–1085, December 1978.

[15] F. Al-Abbass, "Decentralized adaptive variable structure control theory and applications," Ph.D. Thesis, The Ohio State University, 1986.

[16] U. Özgüner, S. Yurkovich, and F. Al-Abbass, "Decentralized variable structure control of a two–arm robotic system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Raleigh, NC, March 1987. (to appear).

Geometric Foundations of the Theory of Feedback Equivalence

R. Hermann*
NASA Ames Research Center
Moffett Field, CA 94035

1. INTRODUCTION

For the past ten years, a group of researchers--mathematicians and theoretical engineers, centered at, and partially supported by, the Flight Control group at NASA-AMES--have attempted to push beyond what was done in the 1960's for linear control theory, and develop effective methods for controlling systems whose dynamical equations are fundamentally *nonlinear*. Our applied focus has been the practical problems encountered in designing aircraft and helicopters, but our methodology--based as it is on fundamental mathematical principles--is adaptable to robotic systems.

Conversely, we hope that use of the new ideas under development in the computer science and AI community will help us use computer technology in a more effective way to handle types of control problems--particularly of a "discrete event" nature--that have been difficult to include in a differential-equations based methodology.

Taking a historical view of progress in engineering and engineering-related mathematics, the situation becomes clarified. The breakthroughs of the 1960's in control theory were closely linked to the development of computers, which could solve differential equations very efficiently. Mathematically, assumptions on *linearity* worked well because of the nature of the engineering problems that needed to be solved, especially in the Apollo Program, where the space craft could be treated satisfactorily as point particles, or at worst as rigid bodies. In the 1970's we attempted to adapt the mathematical techniques developed in the 1960's to the more difficult problems of control of aircraft and helicopters in circumstances where the assumptions of linearity of the dynamics could no longer be realistically justified. Recently, there has been a change in computer technology--such as LISP logic-based symbolic computation and greatly increased potentialities for parallelism--that has not yet been fully integrated into the main body of control theory. Further, computer science has achieved greater maturity and substance, and I believe that there are great scientific and technological possibilities in combining the talents and insights in the two communities. What control theory has to offer is a mature, mathematically based overview of a certain class of engineering problems, based on concepts of *differential equations* and *dynamics*, while the youthful vigor of the computer science discipline is generating a lot of energy, but exhibiting the need (in my opinion, at least) for more scientific and mathematical direction.

For the past two years, I have been trying--with George Meyer's advice and support on the engineering questions--to push in two directions. First, to understand how the control techniques of feedback linearization--developed as a useful control algorithm by Hunt, Meyer and Su at NASA-AMES [1,2]--can be integrated into the mainstream of differential geometry and extended in the direction of understanding the relation between *global* and *local* feedback linearization. Second, I have tried to familiarize myself with the LISP and logic-based computer technology and algorithms, and help in the job of introducing it into control theory. Since the first part of this program is further along--a major mathematical paper is now completed [3] and awaits publication--I will describe some of the ideas it contains here, and leave my ideas about developing relations between computer science and control theory to anotner occasion.

2. A VIEW OF FEEDBACK CONTROL IN THE CONTEXT OF DIFFERENTIAL EQUATIONS, DIFFERENTIAL GEOMETRY, AND LIE THEORY

A feedback control system can be taken as an underdetermined system of ordinary differential equations of the following general form:

$$f\left(x, \frac{dx}{dt}, u\right) = 0 \qquad (2.1)$$

$$x \in R^n; \quad u \in R^m$$

$$f \text{ is a map: } R^{2n+m} \rightarrow R^p$$

53 Jordan Road, Brookline, MA 02146
*This work was begun while the author was a National Research Council Senior Research Associate at the Ames Research Center, and continued under grant #NAG2406.

"x" is a vector of R^n describing components of the system (aircraft, helicopter, spacecraft, robot,...) that are fixed in value, such as velocities, positions, angular or linear momenta, etc. "u" are the control variables, which we must choose in some way to achieve a prescribed goal.

Feedback control can be described as follows. A *feedback map* or *law* is a map

$$x \to F(x) = u \qquad (2.2)$$

$$O \to R^m$$

from an open subset O of R^n to the control space R^m. A *trajectory* of the feedback control law (2.2) is a curve

$$t \to x(t) \qquad (2.3)$$

in R^n that satisfies the following ordinary differential equation:

$$f\left(x(t), \frac{dx}{dt}, F(x(t))\right) = 0 \qquad (2.4)$$

In engineering practice, we will want to choose the feedback law (2.2), so that the family of trajectories defined by (2.3) and (2.4) will have certain stability, robustness, and design properties. (For example, for the latter one might want the trajectory (2.3) to start off at time $t = 0$ at a point x_0 and end up exactly or approximately at a point x_1 at $t = t_1$.) *Stabilization* is the property that is best understood mathematically, hence I will use it as a touchstone here.

Much of the work in control theory of the 1960's--which was very successful on both the mathematical and practical fronts--was oriented toward *linear* control systems, i.e., those of the form:

$$\frac{dx}{dt} - Ax - Bu = 0 \qquad (2.5)$$

where A and B are constant matrices of appropriate size. Here, it is natural to require that the feedback (2.1) preserve this linearity. This can be accomplished by specifying that the feedback map (2.2) be of the following form:

$$u = Kx \qquad (2.6)$$

where K is an $m \times n$ real matrix. The trajectory equations (2.2) are then of the following form:

$$\frac{dx}{dt} = (A + BK) \qquad (2.7)$$

One may then require that these trajectories have a prescribed degree of stability. Because (2.7) is a system of differential equations that can be handled with well-known mathematical techniques, we know that this behavior can be specified by imposing conditions on the eigenvalues of the $n \times n$ matrix

$$A + BK \qquad (2.8)$$

In turn, this "pole-placement" problem can be handled with well-known mathematical techniques (matrix Riccati equations or Kronecker pencil theory) that were applied in the 1960's, but that of course go back many years in the mathematical literature.

It is especially interesting that the useful sufficient conditions for stabilization of (2.5) via linear feedback (2.6), i.e., "pole-placement" in the engineering jargon, involve *controllability* of the control system (2.5)and is a mathematical concept that is--as I showed many years ago [4]--essentially *differential-geometric* and *Lie-theoretic* in nature. Thus, it should be no surprise that the problem of stabilization and feedback control of a more general nonlinear system of type (2.1) also involves differential geometry and Lie theory.

Indeed, the work of Hunt, Meyer, and Su [1,2] (preceeded by work of Krener [5], Brockett [6], Sommer [7], Jakubzyk and Respondek [8]) demonstrate this in a decisive way. Their work only dealt with feedback control of a certain class of systems (the *feedback linearizable* ones, with the functions f(, ,) occurring in (2.1) satisfying certain conditions) if the trajectory stayed within a small neighborhood R^n, whose size could not be specified in advance. This posed the question of finding conditions for *global* feedback equivalence. There has been important partial work on this problem by Boothby, Dayawansa, and Elliot [9-11] using the tools of differential topology and foliation theory. In my paper [3] I have begun to develop ways of applying the Ehresmann-Haefliger [12] theory of pseudogroup cohomology to this problem, but there is a long way to go before the results that are useful in practical situations will come forth.

The mathematical heart of the methods I have developed in [3] is the theory of vector field systems (or distributions) on a manifold and their equivalence. I will now sketch some of this basic differential-geometric theory, then return to the control situation.

3. VECTOR FIELD SYSTEMS AND FEEDBACK EQUIVALENCE

I will now use the formalism "calculus on manifolds," particularly the theory of vector fields (i.e., first-order linear partial differential operators) and the Jacobi-Lie bracket [,] (i.e., commutator) of such vector fields. See Isidori's book [13] for an engineer's introduction to these concepts.

Let Z be a manifold, with $\underline{V}(Z)$ the space of vector fields. In terms of coordinates (z^i) for Z, $1 \leq i,j \leq N = \dim Z$, a $V \in \underline{V}(Z)$ is a differential operator of the following form:

$$V = A^i(z) \frac{\partial}{\partial z^i} \tag{3.1}$$

(summation convention in force)

If

$$V' = B^i \frac{\partial}{\partial z^i} \tag{3.2}$$

then

$$[V,V'] = \left(A^i \frac{\partial (B^j)}{\partial z^i} - B^i \frac{\partial A^j}{\partial z^i} \right) \frac{\partial}{\partial z^j} \tag{3.3}$$

Let $\underline{F}(Z)$ be the ring of C^∞, real-valued functions on Z. $\underline{V}(Z)$ is a *module* over $\underline{F}(Z)$, since vector fields can be multiplied by functions:

$$(f,V) \to fA^i \frac{\partial}{\partial z^i} \tag{3.4}$$

Definition. A *vector field system* $\underline{W}$ on Z is a subspace of $\underline{V}(Z)$ satisfying the following condition:

$$fV \in \underline{W} \quad \text{for} \quad V \in \underline{W}$$

$$V_1 + V_2 \in \underline{W} \quad \text{for} \quad V_1, V_2 \in \underline{W}$$

i.e., $\underline{W}$ is a *submodule* of $\underline{V}(Z)$.

Let $\underline{W}$ be such a vector field system. For $z \in Z$, set

$$\underline{W}(z) = \{V(z): V \in \underline{W}\} \tag{3.5}$$

$\underline{W}(z)$ is a linear subspace of the space of tangent vectors at z. Its dimension is called the *rank* of $\underline{W}$ at z. $\underline{W}$ is said to be *nonsingular* if the rank is constant as z ranges over Z. In this paper we will assume that all vector field systems considered have constant rank, unless specified otherwise. The concept defined next will play a basic role in this work.

Definition. Let $\underline{W} \subset \underline{V}$ be a vector field system. Set

$$C(\underline{W}): \{V \in \underline{W}: [V,\underline{W}] \subset \underline{W}\} \tag{3.6}$$

$C(\underline{W})$ is called the *Cauchy Characteristic system* associated with $\underline{W}$.

Theorem 3.1. $C(\underline{W})$ is another vector field system on Z with the following properties:

$$C(\underline{W}) \subset \underline{W} \tag{3.7}$$

$$[C(\underline{W}),C(\underline{W})] \subset C(\underline{W}) \tag{3.8}$$

i.e., $C(\underline{W})$ is *Frobenius integrable* as a vector field system

$$[C(\underline{W}),\underline{W}] \subset \underline{W} \tag{3.9}$$

Proof. Follows from (3.6).

<u>Definition</u>. A curve $t \bullet z(t)$ in Z is called an *orbit curve* of the vector field system $\underline{W}$ if the following condition is satisfied:

There is a vector field V

$$V = A^i \frac{\partial}{\partial z^i}$$

in $\underline{W}$ such that $t \to z(t)$ is an orbit curve of V, i.e., if

$$\frac{dz}{dt} = V(z(t)) \tag{3.10}$$

or, in coordinate terms:

$$\frac{dz}{dt} = A(z(t)) \tag{3.11}$$

In this way, a vector field system defines a family of curves on Z. It is this geometric property that is the key to the usefulness of vector field systems in control theory. As we have seen, control systems are also defined by families of curves, namely solutions of the control equations:

$$f\left(x(t), \frac{dz}{dt}, u(t)\right) = 0 \tag{3.12}$$

$$x \in R^n , \quad u \in R^m$$

Set:

$$Z = R^n \times R^m$$

$$z = (x,u)$$

We can then define a vector field system $\underline{W}$ on Z as the smallest submodule of $\underline{V}(Z)$ whose orbit curves are solutions of the control equation (3.12).

Let X and X' be manifolds. Let $\underline{W}$ and $\underline{W}'$ be nonsingular vector field systems on X and X', respectively. Let

$$\alpha: X \to X'$$

be a diffeomorphism.

<u>Definition</u>. α is called an *equivalence* from the vector field system $\underline{W}$ to the vector field system $\underline{W}'$ if the following condition is satisfied:

$$\alpha_*(\underline{W}(x)) = \underline{W}'(\alpha(x)) \tag{3.13}$$

$$\text{for all } x \in X$$

i.e., if α maps an orbit curve of $\underline{W}$ into an orbit curve of $\underline{W}'$. Our problem is to describe numbers attached to vector field systems that are *invariant* under equivalence. We shall cite (without proofs) some of the theorems from [3] that do provide such invariants.

<u>Theorem 3.2</u>. Let $\alpha: X \to X'$ be a diffeomorphism from X to X' that is an equivalence of vector field system $\underline{W}$ to vector field system $\underline{W}'$. Let $C(\underline{W})$ and $C(\underline{W}')$ be the Cauchy characteristic systems of $\underline{W}$ and $\underline{W}'$, respectively. Then, the following condition is satisfied:

$$\alpha_*(C(\underline{W})) = C(\underline{W}') \tag{3.14}$$

i.e., α is an equivalence between the Cauchy characteristic systems of the given vector field systems.

<u>Definition</u>. For the vector field system $\underline{W}$, set:

$$\underline{W}^1 = \underline{W} + [\underline{W},\underline{W}] \tag{3.15}$$

It is called a *derived system of* $\underline{W}$.

<u>Theorem 3.3</u>. Let α be an isomorphism from $\underline{W}$ to $\underline{W}'$. Then, it is an isomorphism of the derived system $\underline{W}^1$ to $\underline{W}'^1$.

We can now iterate. Set

$$(\underline{W}^1)^1 = \underline{W}^2, \dots \qquad (3.16)$$

to define the *successive derived systems* of the given vector field system $\underline{W}$, denoted as $\underline{W}^1, \underline{W}^2, \dots$ We obtain an increasing filtration of submodules of the module of all vector fields on X:

$$\underline{W} \subset \underline{W}^1 \subset \underline{W}^2 \subset \dots \qquad (3.17)$$

Theorem 3.4. We have:

$$C(\underline{W}) \subset C(\underline{W}^1) \subset C(\underline{W}^2) \subset \dots \qquad (3.18)$$

In words, this says that the Cauchy characteristics of the derived systems also form an ascending, filtered sequence of submodules of the module of all vector fields on X.

We assume that all the modules (3.17) and (3.18) are of constant rank. Set

$$\begin{aligned} r &= \text{rank } \underline{W} \\ c &= \text{rank } C(\underline{W}) \\ r_1 &= \text{rank } \underline{W}^1 \\ c_1 &= \text{rank } C(\underline{W}^2) \end{aligned} \qquad (3.19)$$

and so on.

Theorem 3.5. The sequence of integers

$$\begin{aligned} &r \le r_1 \le r_2 \le \cdots \\ &c \le c_1 \le c_2 \le \cdots \end{aligned} \qquad (3.20)$$

attached to the vector field system $\underline{W}$ are numerical *equivalence invariants*.

Let us now apply these results to control systems in state space form.

4. FEEDBACK INVARIANTS FOR CONTROL SYSTEMS IN STATE SPACE FORM

Let us now specialize the feedback control system to consider those of the following *state space* form:

$$\frac{dx}{dt} = f(x,u) \qquad (4.1)$$

$$y \in R^n, \quad u \in R^m \ .$$

Theorem 4.1. Let

$$\begin{aligned} Z &= R^n \times R^m \\ &= \{(x,u) : x \in R^n, u \in R^m\} \end{aligned}$$

Let $\underline{W}$ be the vector field system Z generated by the components of the following vector-valued vector fields on Z:

$$\underline{W} = \left\{ f(x,u)\frac{\partial}{\partial x}, \frac{\partial}{\partial u} \right\} \qquad (4.2)$$

Then, the orbit curves on $\underline{W}$ are precisely the curves $t \to (x(t),u(t))$ that satisfy the control equation (4.1).

Theorem 4.2. Let $dx/dt = f(x,u)$, and $dz/dt = h(z,v)$ be two feedback control systems with the same number of states and controls. Let

$$\underline{W} = \left\{ f(x,u)\frac{\partial}{\partial x}, \frac{\partial}{\partial u} \right\}$$

and

$$\underline{W}' = \left\{h(y,v)\,\frac{\partial}{\partial y}, \frac{\partial}{\partial v}\right\}$$

be the vector field systems assigned to these control systems. Let

$$T: R^{n+m} \rightarrow R^{n+m} \tag{4.3}$$

be a C^∞ map of the following form:

$$T(x,u) \rightarrow (y,v) \tag{4.4}$$

with

$$\begin{aligned} y &= \alpha(x) \\ v &= \beta(x,u) \end{aligned} \tag{4.5}$$

Then T maps the control system $\{dx/dt = f(x,u)\}$ into the control system $\{dy/dt = n(y,v)\}$, in the sense that it maps solution curves of the first system of ordinary differential equations into solution curves of the second, if and only if T is an equivalence from the vector field system $\underline{W}$ to the vector field system $\underline{W}'$. In particular, the integers $r, r_1, \ldots; c, c_1, \ldots$ assigned by (3.3) to $\underline{W}$ are invariant under feedback equivalence. In the case of a linear control system, these integers can be computed in terms of the *controllability indices.*

Let us now consider the vector field systems associated with a linear, scalar input, control system, i.e., one of the following form:

$$\frac{dx}{dt} = Ax + bu \tag{4.6}$$

$$x \in R^n, \quad u \in R, \quad b \in R^n$$

Associate with that system the following pairs of vector fields on R^n:

$$\begin{aligned} V &= Ax\,\frac{\partial}{\partial x} \\ V_0 &= b\,\frac{\partial}{\partial x} \end{aligned} \tag{4.7}$$

Let $\underline{W}$ be the vector field systems on R^n spanned by these two vector fields and $\partial/\partial u$. Set:

$$\begin{aligned} V_i &= \mathrm{Ad}^i(V)(V_0) \\ &= [V, V_{i-1}] \end{aligned} \tag{4.8}$$

$$\text{for } i \geq 0$$

<u>Theorem 4.3</u>. The following commutation relations hold among these vector fields on R^n:

$$[V, V_i] = V_{i+1}, \qquad \text{for } i = 0,1,2,\ldots \tag{4.9}$$

$$[V_i, V_j] = 0, \qquad \text{for } i,j = 0,1\ldots.$$

The derived system $\underline{W}^j$ is the vector field system generated by

$$\left\{\frac{\partial}{\partial u}, V, V_i;\ i = i,\ldots,j\right\} \tag{4.10}$$

<u>Theorem 4.4</u>. If the system (4.6) is controllable, then

$$C(\underline{W}_{j+1}) = \left\{\frac{\partial}{\partial u}, V_0, V_1, \ldots, V_j\right\} \tag{4.11}$$

As I show in [3], Theorem 4.4 is the geometric heart of the sufficient conditions that Hunt, Meyer, and Su [1,2] provided in their work on feedback linearization, namely:

Theorem 4.5. Let V_0, V_1 be vector fields on R^n generating a single input, controllable control system of the following form

$$\frac{dx}{dt} = V(x) + uV_0(x) \qquad (4.12)$$

Suppose the following condition is satisfied:

> The vector field systems
>
> $$\{V_0, V_1 = [V, V_0], \ldots, V_j = [V, V_{j-1}]\} \qquad (4.13)$$
>
> are Frobenius integrable for all j .

Then, the system (4.12) is *locally* feedback equivalent to a chosen system.

In the Hunt-Meyer-Su work, the transformation T, which establishes the feedback equivalence of (4.12) with a linear system, is obtained as a solution of a system of first order, partial differential equations, and we can only prove existence of such linearizing transformations *locally*. A basic question is:

> How to piece together such local feedback equivalences to find a *global* one?

The answer can be described in terms of cohomology theory [12]. Indeed, this is a typical problem of *global* differential geometry:

> Find conditions for the existence (and computational feasibility!) of a *global* solution of a system of partial differential equations when the conditions for existence of local solutions are satisfied.

What complicates the analysis of the conditions for existence of a global solution is that the cohomology theory one must use involves an algebraic object--the groupoid of feedback automorphisms of the linear control systems--that is *infinite dimensional*, so that standard topological techniques are not very helpful. It is interesting to note that elementary particle physicists at the frontiers--in the so-called *string theory*--are involved with mathematical monstrocities that are very similar to these! Work on this question is in progress.

5. REFERENCES

[1] R. Su, "On the Linear Equivalence of Nonlinear Systems," Syst. and Contr. Lett. 2, 48-52 (1982).

[2] L.R. Hunt, G. Meyer, R. Su, "Design for Multi-Input Nonlinear Systems," in Differential Geometric Control Theory (R.W. Brockett, R.S. Millman, and H. Sussman, eds.), Birkhauser-Boston, 1983.

[3] R. Hermann, "Global Invariants for Feedback Equivalence and Cauchy Characteristics of Nonlinear Control Systems" (to appear) Acta. App. Math.

[4] R. Hermann, "On the Accessibility Problem in Control Theory," in International Symposium on Nonlinear Differential Equations and Nonlinear Mechanics (J.P. LaSalle and S. Leffchetz, eds.), Academic Press, NY, 1963.

[5] A.J. Krener, "On the Equivalence of Control Systems and Linearization of Nonlinear Systems," SIAM J. Control Optim. 11, 670-676 (1973).

[6] R.W. Brockett, "Feedback Invariants for Nonlinear Systems," IFAC Contress, Helsinki, 1978.

[7] R. Sommer, "Control Design for Multivariable Nonlinear Time-Varying Systems," Int. J. Contr. 31, 883-89] (1980).

[8] B. Jakubzyk and W. Respondek, "On Linearization of Nonlinear Control Systems," Bull. Acad. Polonaise Sci. Ser. Math. 28, 517-522 (1980).

[9] W.M. Boothby, "Some Comments on Global Linearization of Nonlinear Systems," Systems and Control Letters 4, 143-147 (1984).

[10] W. Dayawansa, W.M. Boothby, and D.L. Elliot, "Global State and Feedback Equivalence of Nonlinear Systems," Systems and Control Letters 5, 228-234 (1985).

[11] W. Dayawansa, "Geometry of the Feedback Linearization Problem," Ph.D. Thesis, Washington University, 1986.

[12] A. Haefliger, "Structures Feuilletees et Cohomologie a Valeur dans un Faisceau de Groupoides," Comment. Math. Helv. 32, 248-329 (1958).

[13] A. Isidori, Nonlinear Control Systems: An Introduction, Springer-Verlag, 1985.

530 37

183198

10[illegible]

Reducing Model Uncertainty Effects in Flexible Manipulators Through the Addition of Passive Damping

T.E. Alberts
Old Dominion University
Norfolk, VA 23508

Ø 5 5 5 3 2 [illegible]
[illegible]

1. Abstract

An important issue in the control of practical systems is the effect of model uncertainty on closed loop performance. This is of particular concern when flexible structures are to be controlled, due to the fact that states associated with higher frequency vibration modes are truncated in order to make the control problem tractable. ~~In this paper we employ~~ digital simulations of a single-link manipulator system are employed to demonstrate that passive damping added to the flexible member reduces adverse effects associated with model uncertainty. A controller was designed based on a model including only one flexible mode. This controller was applied to larger order systems to evaluate the effects of modal truncation. Simulations using an LQR design assuming full state feedback illustrate the effect of control spillover. Simulations of a system using output feedback illustrate the destabilizing effect of observation spillover. The simulations reveal that the system with passive damping is less susceptible to these effects than the untreated case.

2. Introduction

Many in-space robotic operations will require arms capable of very long reach, while like other space structures, they must be lightweight. Because such arms are likely to be highly compliant (as is the space shuttle RMS arm), control strategies designed to accommodate structural flexibility must be considered. Controlling flexible structures through purely active measures can be cumbersome in terms of hardware and computation time requirements. Moreover, active controllers for flexible structures are subject to instability and other problems associated with model uncertainty. The burden of active control can be reduced by augmenting active control with passive damping. This enhances system stability and reduces the adverse affects of model uncertainty, thereby providing justification for the use of low order dynamic models and controllers.

In this paper we consider a single-link, single-axis arm which rotates in the horizontal plane about a pinned hub in response to a control torque $\tau(t)$. The system, illustrated in Figure 1, and the models employed in this investigation are based upon a laboratory version of the arm that has been used in experimental investigations [1-4] at Georgia Tech. The flexible member is a long slender beam that is assumed infinitely stiff in vertical bending but flexible in horizontal bending.

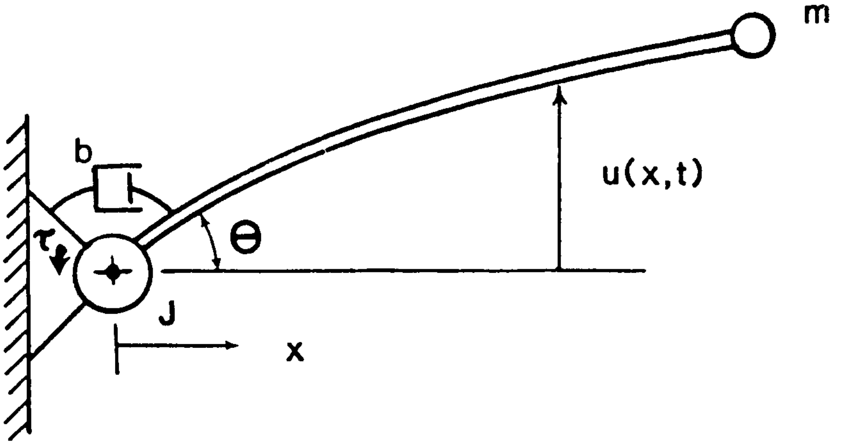


Figure 1. System Configuration

The pinned hub has rotary inertia J. A point payload mass m is fixed to the beam's tip. Although, manipulators sometimes carry payloads that have significant rotary inertia, the effect of this inertia i qualitatively similar to that of a point mass for the configuration considered, and hence payload inertia i not included here. Friction in the pinned joint is represented by a rotary viscous dashpot. Thi configuration is viewed as being representative of lightweight, large payload capacity manipulators. Parameters and dimensions for the arm that the system considered in this paper are tabulated in Appendix A.

Damping augmentation is provided by a constrained viscoelastic layer damping treatment [2,5]. Th approach involves bonding a thin film of viscoelastic material to the flexible member's surface. Thi viscoelastic layer in turn has a stiff elastic constraining layer bonded to its surface. The combine system forms a sandwich-like structure illustrated in Figure 2. When elastic deflection of the structur occurs, shear induced plastic deformation is imposed in the viscoelastic layer. The energy dissipatio associated with the plastic deformation provides the desired mechanical damping. The damping ratio for th untreated beam was approximately constant for all modes at .007. The treatment increased the damping ratio associated with the modes of interest (say the first six modes) by about an order of magnitude. The treate beam had a damping ratio of .03 for the first mode and the values for the 2nd through 6th modes ranged fro .052 to .06. Additional damping is introduced by joint friction.

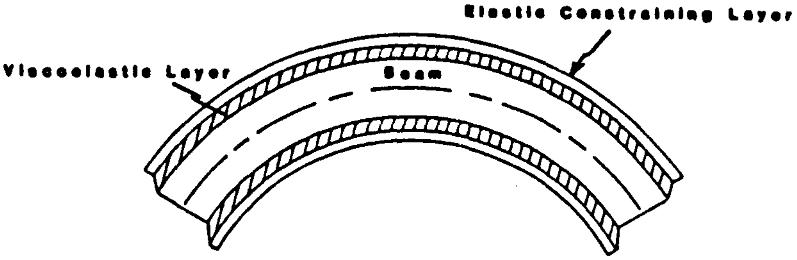


Figure 2. Treated Beam Element Under Flexure

The first step of controller design is usually the development of a "design model" that is a simplifie representation of the actual plant dynamics. The design model serves as the basis for controller design In the case of flexible mechanical systems, the design model is often a truncated representation of th actual plant, retaining only a few critical modes. This implies the assumption that a model based upon small number of vibration modes provides adequate representation of the much larger order actual plant, fo controller design purposes. The modeling error associated with the neglected modes, adversely affect closed loop system performance. In this paper, simulation results are presented to illustrate that the il effects associated with modeling error are reduced somewhat through the addition of passive damping to th system.

We consider a multivariable control system, designed according to the steady state linear quadrati regulator (LQR) approach. A four state model including only one flexible mode and the rigid body mod represents the design model. We consider the consequences of controlling larger order plants representative of the actual system, with a controller derived for the design model.

The regulator is formulated to penalize tip position and control effort. Two cases are considered The first assumes that full state feedback is available. The second case uses output feedback of ti position (v_L), tip velocity, hub angle (θ) and hub angular velocity. The controller designs are kept simpl to facilitate comparisons between the damped and undamped systems.[1]

3. Dynamic Model

Linear transfer function models for the system of interest were developed based on the assumption o small bending deflections and small hub angles. Transfer function modeling for similar systems has bee discussed by several authors [2,4,6-8] and we will not repeat the procedure here. Details on development o the model employed here may be found in [2]. The transfer function poles and zeros used in thi investigation are tabulated in Appendix B.

[1] Although it possesses some light structural damping and is affected by joint friction, we shall designate the untreated arm as "undamped".

In the interest of obtaining a state space realization from the transfer functions it is convenient to work with the partial fraction expansion form given by Equation 1. The ω's are the modal frequencies, the $\underline{\omega}$'s are the damped modal frequencies $\omega\sqrt{1-\zeta^2}$, the ζ's are modal damping ratios and n is the number of flexible modes represented. Damping due to joint friction has not been accounted for in these transfer functions but will introduced later as a form of feedback.[2] The residues λ_0 and μ_0 correspond to the rigid body mode.

$$G_\chi(s) = \frac{\lambda_0}{s} + \frac{\mu_0}{s^2} + \frac{\lambda_1 s + \mu_1}{s^2 + 2\zeta_1\omega_1 s + \underline{\omega}_1^2} + \cdots + \frac{\lambda_n s + \mu_n}{s^2 + 2\zeta_n\omega_n s + \underline{\omega}_n^2} \qquad (1)$$

The subscript χ on G_χ (s) represents the output variable of interest. For the present study four transfer functions were required. Equation 2 summarizes these and defines the notation used here.

$$\begin{bmatrix} \text{hub angular position} \\ \text{hub angular rate} \\ \text{beam tip position} \\ \text{beam tip rate} \end{bmatrix} = \begin{bmatrix} \theta(s) \\ s\theta(s) \\ V_L(s) \\ sV_L(s) \end{bmatrix} = \begin{bmatrix} G_\theta(s) \\ G_{\dot\theta}(s) \\ G_{V_L}(s) \\ G_{\dot V_L}(s) \end{bmatrix} T(s) \qquad (2)$$

Here s is the Laplace operator and T(s), θ(s) and $V_L(s)$ denote the Laplace transforms of the input torque, hub angle and tip position variables, respectively.

Figure 3 is a block diagram representation of the transfer function.

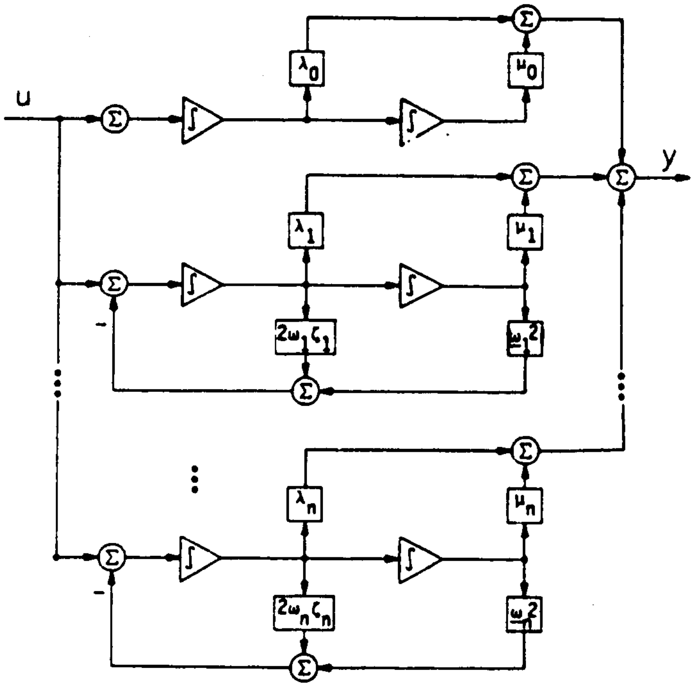


Figure 3. Block Diagram Equivalent of Equation 1

The system matrices corresponding to Figure 3 are as follows:

$$A = \begin{bmatrix} 0 & 1 & & & & & \\ 0 & 0 & & & & & \\ & & 0 & 1 & & & \\ & & -\omega_1^2 & -2\omega_1\zeta & & & \\ & & & & \ddots & & \\ & & & & & 0 & 1 \\ & & & & & -\omega_n^2 & -2\omega_n\zeta_n \end{bmatrix}$$

$$B = [\ 0\ 1\ 0\ 1\ \cdots\ 0\ 1\ ]^T \tag{3}$$

$$\underline{C}_{\chi} = [\ \mu_o\ \lambda_o\ \mu_1\ \lambda_2\ \cdots\ \mu_n\ \lambda_n]$$

The leading principle 2x2 submatrix of A represents the rigid body mode in the tip and hub position transfer functions.

For each of the output variables χ there is a unique output matrix C_{χ}. We will denote these as C_{θ}, $C_{\dot{\theta}}$, C_{vL} and $C_{\dot{v}L}$ with the notation carrying the obvious meaning. These are assembled to form a measurement matrix representing four outputs as indicated in the measurement equation (4) given by:

$$y = \begin{bmatrix} \theta \\ \dot{\theta} \\ v_L \\ \dot{v}_L \end{bmatrix} = \begin{bmatrix} C_{\theta} \\ C_{\dot{\theta}} \\ C_{vL} \\ C_{\dot{v}L} \end{bmatrix} x = \underline{C}x \tag{4}$$

In order to account for viscous joint damping we consider the feedback system illustrated in Figure 4.

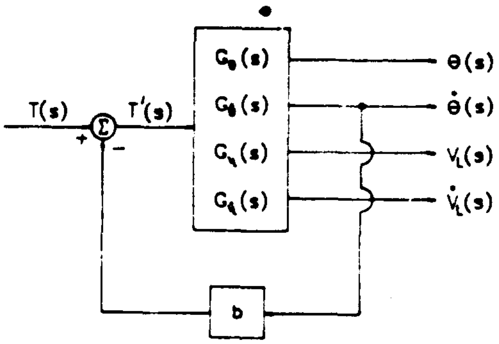


Figure 4. Block Diagram Illustrating Feedback of Joint Damping

When the effect of joint damping is introduced through boundary condition feedback, a new A matrix is formed:

$$\underline{A} = A - BC_{\dot{\theta}}b \tag{5}$$

Where b is the joint damping coefficient. The analysis that follows is based upon a system model of the form ($\underline{A}$, B, $\underline{C}$) .

4. System Representation

We wish to design a controller for a plant with 2(n+1) states, using a design model including only one vibration mode. Here n represents the number of flexible vibration modes required to provide accurate representation of the actual plant, and the additional two states represent rigid body motion. Controllers developed for the single mode system are applied to a model including three vibration modes (8 states), and one including six vibration modes (14 states). These are referred to as the plant models in the text that follows because they are intended to represent actual plants in the simulations presented here.

The system equations for the plant models are given by:

$$\dot{\underline{x}}_n(t) = \underline{A}_n \underline{x}_n(t) + \underline{B}_n u(t) \qquad (6)$$
$$y(t) = \underline{C}_n \underline{x}_n(t)$$

Here $\underline{A}_n$ is the $\underline{A}$ matrix defined in Equation 5, in this case for an n mode approximation of the plant. Similarly the $\underline{C}_n$ matrix is the $\underline{C}$ matrix of Equation 4 for an n mode approximation.

The system equations for the design models are given by:

$$\dot{\underline{x}}_1(t) = \underline{A}_1 \underline{x}_1(t) + \underline{B}_1 u(t) \qquad (7)$$
$$y(t) = \underline{C}_1 \underline{x}_1(t)$$

The system output vector (y) considered is of the form

$$y = [\ \theta\ \dot{\theta}\ v_L\ \dot{v}_L\ ]^T. \qquad (8)$$

5. Regulator Design

The standard form of linear quadratic cost function is

$$J_c = \int_0^\infty (x^T Q x + u^T R u)\, dt \qquad (9)$$

where Q is a symmetric, positive semi-definite state weighting matrix and R is a symmetric, positive definite control effort weighting matrix. Because we seek to regulate tip position, a performance index that penalizes the tip position output variable and control effort was chosen. A cost function for tip position output weighting is expressed as follows:

$$J_c = \int_0^\infty (v_L{}^2 + r\tau^2)\, dt \qquad (10)$$

The output weighted performance index (10) is equivalent to the standard state weighted version (9) with weighting matrices given by:

$$Q = \underline{C}_n{}^T \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 1 & \\ & & & 0 \end{bmatrix} \underline{C}_n \quad , \qquad R = [r] \qquad (11)$$

The system ($\underline{A}_n$, B_n, $\underline{C}_n$) represents an actual plant with dynamics that are either incompletely known or too cumbersome to permit the use of the full model in controller design. The four state design model ($\underline{A}_1$, B_1, $\underline{C}_1$) will serve as an approximation to the actual plant for controiler design purposes. In this case $\underline{C}_1$ replaces $\underline{C}_n$ in the state weighting matrix Q (11)

Two attractive features of the tip position weighted cost function (10) are that it has only one parameter (r) to vary, and that a given value of r can be expected to impose similar performance demands on both systems (damped and undamped). Reducing the value of r decreases the penalty on control effort and is therefore equivalent to demandinǵ higher performance at the expense of increased control energy.

6. State Feedback

In a system with decoupled modes, such as the Jordan canonical realization of Equation 3, a state feedback law

$$u = -K_1 x_1 \qquad (12)$$

designed to stabilize the reduced order system (7) will stabilize the actual system (6), provided that the truncated modes are asymptotically stable. The neglected modes can, however, be excited at their natural frequencies in response to the applied control input. This effect is called control spillover [9,10] in the literature related to controlling flexible spacecraft. The system we are considering has a small amount of modal coupling, due to the introduction of viscous joint damping using Equation 5. Since we normally do not expect viscous damping to destabilize a system, it is reasonable to expect that the state feedback law (12) will stabilize the plant models of interest.

In this section we design a controller based on the output weighted performance index (10). We assume that the first four elements of the state vector are somehow available for feedback. Control is applied to these four state elements according to Equation 12. The time varying gain $K_1(t)$ that minimizes the cost function (9) is given by

$$K_1(t) = R^{-1}B_1^TS(t) \tag{13}$$

where S(t) is the solution of the associated matrix Riccati equation. An often used substitution for the optimal gain $K_1(t)$ is its constant steady state value $K_1 = K_1(\infty)$. The steady state value ^f S(t) is the solution of the algebraic Riccatti equation:

$$-\dot{S}(\infty) = S(\infty)\underline{A}_1 + \underline{A}_1^TS(\infty) - S(\infty)\,B_1R^{-1}B_1^TS(\infty) + Q = 0 \tag{14}$$

The steady state gain solution is used ' :he simulations presented here.

7. Simulations With State Feedback

Figure 5 illustrates the simulated response of the design model (7) to a 4.8 inch step command, for various values of r. The simulations indicate that both the damped and undamped systems are capable of almost arbitrarily good performance as r is decreased. In practice, the limit on the response time is dictated by the strength of the beam and the torque limit of the actuator. A theoretical (assuming in,inite beam strength and motor torque capacity) limit on the speed of response is discussed by Schmitz [6]. This limit is related to the non-minimum phase character of the tip position transfer function. Notice tnat the wrong way start phenomenon typical to systems with non-minimum zeros is indicated in the plots. Schmitz interprets the theoretical response limit as being roughly equivalent to a pure delay associated with the initial period durin, which the tip moves in the direction opoosite to the control command.

When the state feedback law (12) is applied to the larger order systems (Figures 6 and 7), the excitation of the second mode of vibration is readily apparent when $r = 10^{-6}$. In the undamped system (Figures 6a and 7a) the oscillation takes more than two seconds to dic out. Thus, in the case of the undamped system, we find that designing for higher performance (by reducing r) actually results in slower response. The excitation of the second mode also occurs in the damped system (Figure 6b and 7b), however, it dies out in about 0.8 seconds. Although the performance of the actual plant is not as good as that of the design model (Figure 5b), the simulation indicates that the response time for $r = 10^{-6}$ is slightly better than the lower levels of demanded performnce (larger values of r) considered. This is in sharp contrast with the results of Figures 6a and 7a for the undamped system. This example clearly indicates that the damped system is less susceptible to control spillover than the undamped case.

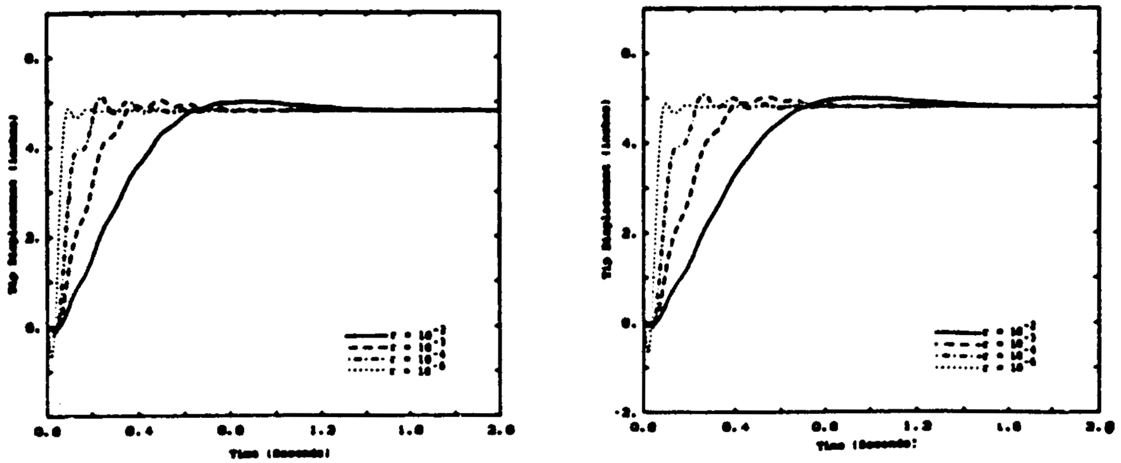
It should be noted that the peak control torque command, when $r = 10^{-6}$, is about 4800 inch pounds. This value is well above the beam's maximum bending moment capacity (~ 175 in. lbf. based on yield) and is about 60 times greater than the rated torque capacity (85 in.lbf.) of the experimental system's motor. In light of these figures, one might argue that control spillover is not a realistic concern for the system of interest. The author concedes to the somewhat articicial nature of this example, however, further consideration of the results adds to their significance. Suppose the initial step command is scaled down by a factor of 50 to about 0.1 inches. Because the system model is linear, we know that the corresponding peak torque is about 100 inch.lbf. This is a realistic figure for the system of interest. In Figures 6a and 7a, peak tip position oscillation amplitude is about 1.5 inches. Scaling this figure down by a factor of 50 gives 30 thousandths of an inch - a significant value in the context of robot accuracy.

8. Output Feedback

The simulations presented in the previous section were based on the assumed availability of states. Practical control systems must depend upon measured outputs for feedback. Frequently the outputs are different entities than the states. In contrast to systems using state feedback, output feedback systems are subject to instability as a consequence of model reduction [9-11] even when the neglected modes are asymptotically stable. This effect is sometimes called observation spillover.

In this section we follow the steady state LQR controller design approach employed in the previous section, however, we implement the controller using output feedback. The design model (A_1, B_1, C_1) has four states and four outputs and the measurement matrix $\underline{C}_1$ is invertible. This allows us to calculate the state x_1 of the desing model from the output vector y according to:

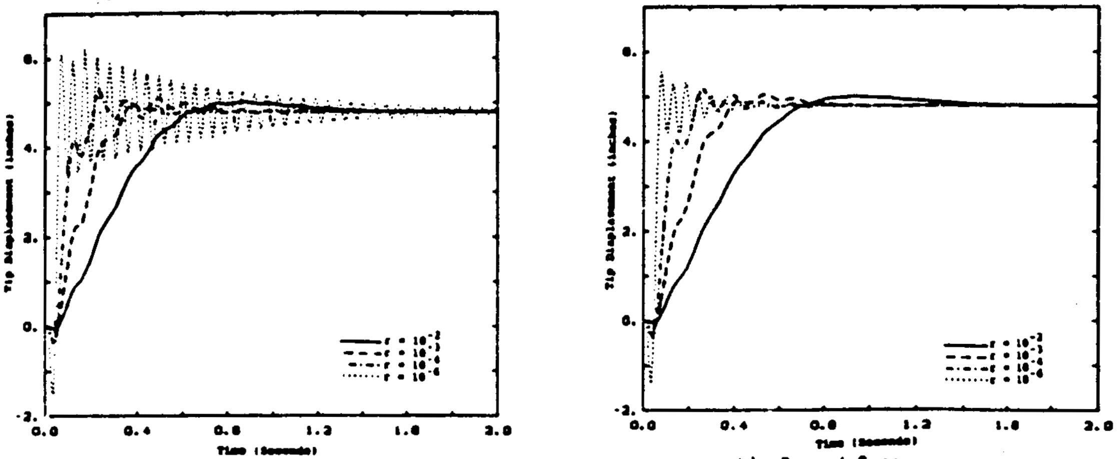
$$x_1 = \underline{C}_1^{-1}\,y \tag{15}$$



a) Undamped Case

b) Damped Case

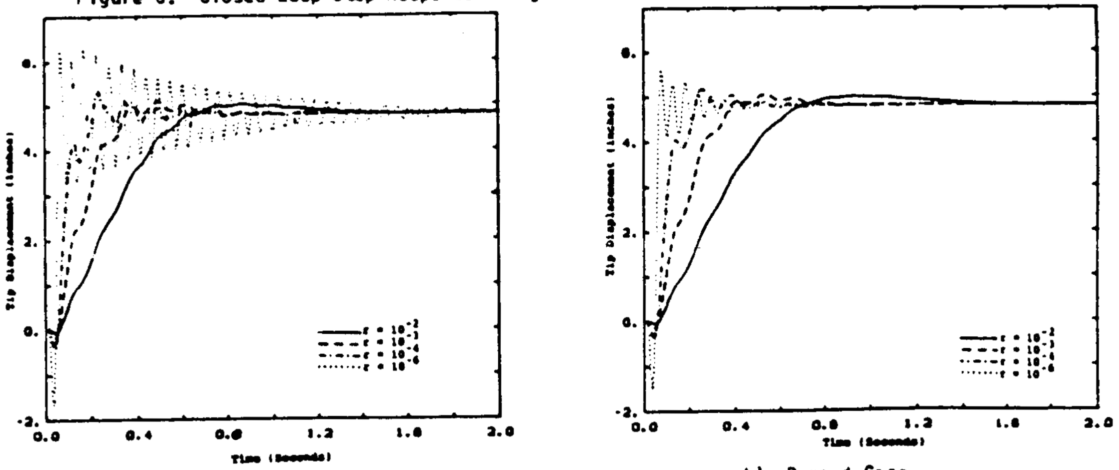
Figure 5. Closed Loop Step Response Using State Feedback, One Vibration Mode Plant Model



a) Undamped Case

b) Damped Case

Figure 6. Closed Loop Step Response Using State Feedback, Three Vibration Mode Plant Model



a) Undamped Case

b) Damped Case

Figure 7. Closed Loop Step Response Using State Feedback, Six Vibration Mode Plant Model

In this case the output feedback control law is given by:

$$u = -K_1 \underline{C}_1^{-1} y \tag{16}$$

The output feedback law (16), when applied to the design model is equivalent to state feedback (12). When applied to controlling the actual model, the output feedback control law (16), expressed in the form of a state feedback law is given by:

$$u = -K_1 \underline{C}_1^{-1} \underline{C}_n x_n \tag{17}$$

Notice that when applied to the plant model, the output feedback law receives information from the states that were neglected in design. This is unwanted input (spillover), and can be viewed as a form of measurement corruption.

9. Simulations With Output Feedback

Simulation results obtained using output feedback are presented in Figures 8 through 10. Figures 8a and 8b are simulations of the design model using the output feedback law (16). When applied to the design model the output feedback law considered is equivalent to state feedback (Figure 5). This case is presented here as a basis for comparison. When the low order output feedback law (16) is applied to the actual system models (Figure 9 and 10) we observe that the performance is limited by the onset of instability. In the undamped system, the first vibration mode is unstable for r = 0.01 and r = 0.005. The damped system remains stable under the same conditions, however, some first mode oscillatory behavior becomes evident as we attempt to design for higher performance. The damped system is not immune to tne instability experienced by the undamped case, however, due to its more favorable open loop pole placement it is more robust.

Upon comparison of the six mode and three mode systems, we find that the stable responses of the six mode plants do not differ noticeably from those of the three mode plants. On the other hand, the divergence rate of the unstable oscillations is greater in the six mode plant (Figure 10a) than in the three mode plant (Figure 9a). This indicates that the presence of the higher, neglected modes (4th, 5th and 6th) do affect system stability slightly.

This example illustrates that the passive damping treatment considered reduces the flexible system's susceptibility to observation spillover induced instability. The peak torque commanded at the highest performance (when the system is stable) was about 80 in.lbs., indicating that the performance demanded was reasonable for the system under consideration. The example employs perhaps the most simplistic of all possible output feedback schemes. Systems employing state estimators also rely on measured data for feedback, and they too are subject to instability due to modeling error.

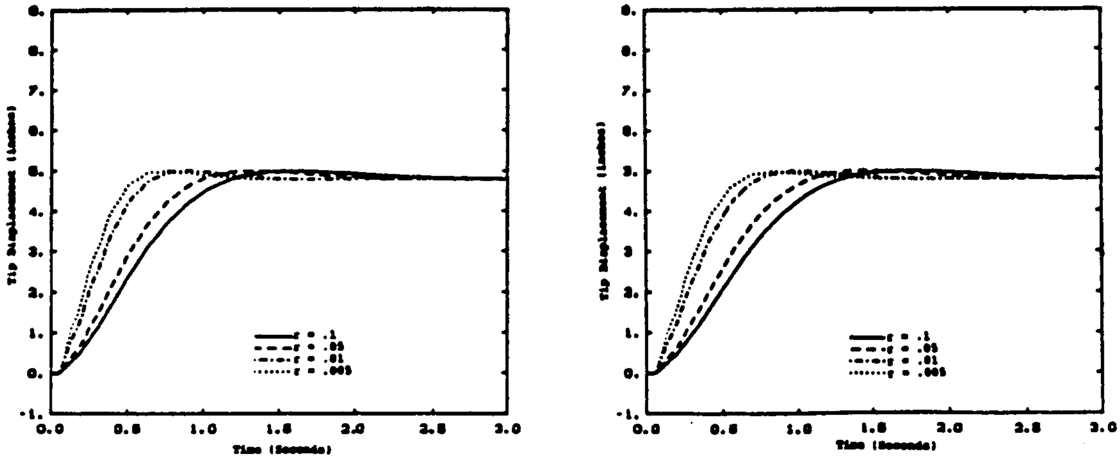
10. Conclusion

One form of modeling error that is relevant for control of flexible structures results from ignoring high order vibration modes in the process of deriving a design model. The effects of this type of modeling error are manifested as control and observation spillover. We have presented simulations of multivariable control a particular flexible arm to illustrate that the addition of passive damping yields a system that is less susceptible to these undesirable effects.

To some degree these results follow intuition, in that one naturally expects that increasing the damping terms of a system's eigenvalues will provide a more stable system with improved performance. The results presented are intended to demonstrate the concept of passive damping on an example that is representative of practical lightweight manipulators.

11. Acknowledgements

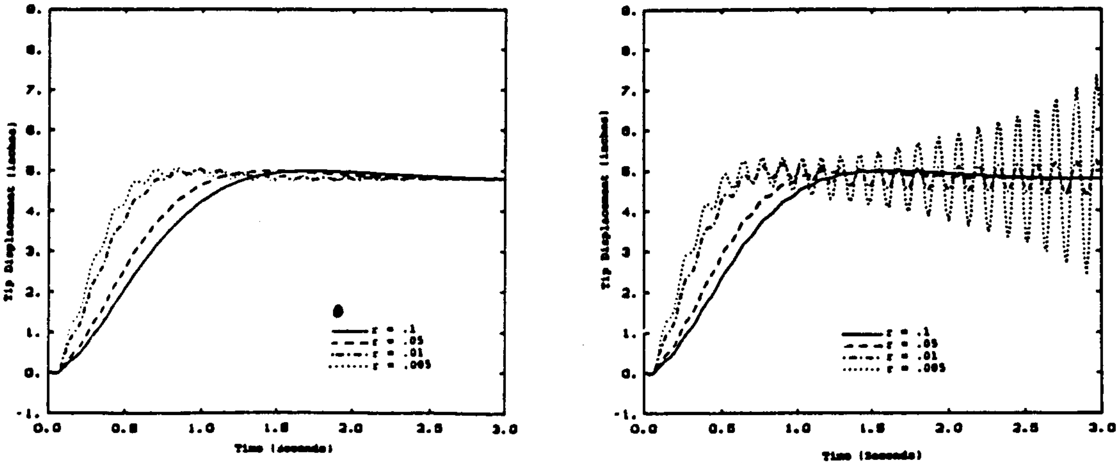
The research described was performed at the Georgia Institute of Technology under the joint advisement of Dr. Wayne J. Book and Dr. Stephen L. Dickerson. The author was supported through a Research Assistantship provided by the Georgia Tech Material Handling Research Center (MHRC).



a) Undamped Case

b) Damped Case

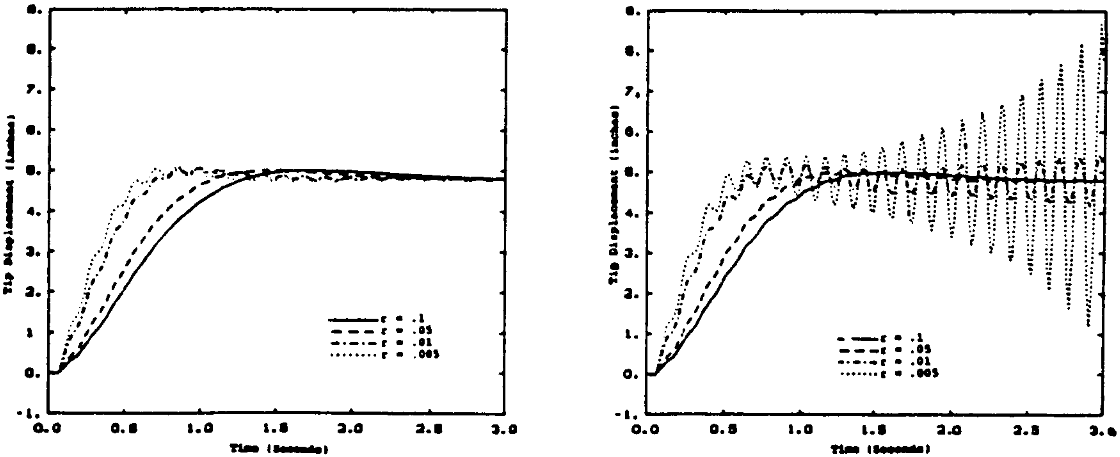
Figure 5. Closed Loop Step Response Using Output Feedback, One Vibration Mode Plant Model



a) Undamped Case

b) Damped Case

Figure 6. Closed Loop Step Response Using Output Feedback, ThreeVibration Mode Plant Model



a) Undamped Case

b) Damped Case

Figure 7. Closed Loop Step Response Using Output Feedback, Six Vibration Mode Plant Model

12. Bibliography

[1] Hastings, G.G., Controlling Flexible Manipulators, An Experimental Investigation, Ph.D. Dissertation, Dept. of Mech. Eng., Georgia Institute of Technology, August, 1986.

[2] Alberts, T.E., Augmenting the Control of a Flexible Manipulator with Passive Mechanical Damping, Ph.D. Dissertation, Ga. Tech., Sept. 1986.

[3] Alberts, T.E., Hastings, G.G., Book, W.J. and Dickerson, S.L., "Experiments in Optimal Control of a Flexible Arm with Passive Damping", Proc. 5th Symp. on Dynamics and Control of Large Structures, June 1985, pp. 423-435.

[4] Alberts, T.E., Book, W.J. and Dickerson, S.L., "On the Transfer Function Modeling of Flexible Structures with Distributed Damping", 1986 ASME Winter Annual Meeting, Dec. 1986.

[5] Plunkett, R. and Lee, C.T., "Length Optimization for Constrained Viscoelastic Layer Damping", J. Accoust. Soc. of Amer., Vol. 48, No. 1, 1970, pp. 150-16.

[6] Schmitz, E., "Experiments on the End-point Position Control of a Very Flexible One Link Manipulator", Ph.D Dissertation, Stanford Univ., Dept. of Aero & Astro., June 1985.

[7] Martin, G.D., On the Control of Flexible Mechanical Systems, Ph.D Dissertation, Stanford Univ., Dept. of E.E., May 1978.

[8] Breakwell, J.A., "Control of Flexible Spacecraft", Ph.D Dissertation, Stanford Univ., Dept. of Aero. & Astro., Aug. 1980.

[9] Balas, M.J., "Feedback Control of Flexible Systems", IEEE Trans. on Automatic Control, Vol. AC-23, No. 4, Aug. 1978, pp. 673-679.

[10] Blas, M.J., "Active Control of Flexible Systems", J. of Optim. Th. and App., Vol. 25, No. 3, July 1978, pp. 415-436.

[11] Khalil, H.K., "On the Robustness of Output Feedback Control Methods to Modeling Errors", IEEE Trans. on Automatic Control, Vol. AC-26, No. 2, April 1981.

APPENDIX A - System Parameters and Dimensions

Joint Inertia	J = 30.2 in^2.lbm.
Payload Mass	m = 0.09 lbm.
Joint Damping Coefficient	b = 0.10 in.lbf. s^2
Beam Dimensions	48" x 3/4" x 3/16"
Material	6065-T6 Aluminum

APPENDIX 8 - Transfer Function Poles and Zeros

Table B-1 Undamped System

Mode	System Poles		Hub Angle T.F. Zeros		Tip Position T.F. Zeros
1	-0.0541	±j7.726	-0.0149	±j2.1129	±8.3410
2	-0.1284	±j18.3456	-0.0985	±j14.0768	±45.0741
3	-0.2957	±j42.2446	-0.2853	±j40.7557	±111.3047
4	-0.5769	±j82.4188	-0.5719	±j81.6939	±206.9715
5	-0.9633	±j137.6207	-0.9603	±j137.1836	±332.0743
6	-1.4532	±j207.5965	-1.4511	±j207.2946	±486.6130

Table B-2 Damped System

Mode	System Poles		Hub Angle T.F. Poles		Tip Position T.F. Poles		
1	-.22197	±j7.1601	-.0176	±j1.9443	1	-7.2785,	+ 7.8830
2	-.9124	±j17.4515	-.6202	±j13.2581	2	-45.9556,	+44.7292
3	-2.3169	±j41.6716	-2.2342	±j40.1610	3	-97.7161,	+113.9306
4	-4.8477	±j83.2589	-4.7987	±j82.5101	4	-263.5548,	+216.0952
5	-8.5293	±j142.4761	-8.5042	±j142.0184	5	-389.2245,	+351.1999
6	-11.7674	±j219.5993	-11.7581	±j219.2929	6	-554.2140,	+519.0317

S31-61

138199

S?

Parallel Processing Architecture for Computing Inverse Differential Kinematic Equations of the PUMA Arm

T.C. Hsia
University of California, Davis
Davis, CA 95616

need code for corpsource

G.Z. Lu and W.H. Han
Nankai University
Tiangjin, China

NA416113

1. Abstract

In advanced robot control problems, on-line computation of inverse Jacobian solution is frequently required. Parallel processing architecture is an effective way to reduce computation time. ~~In this paper~~, a parallel processing architecture is developed for the inverse Jacobian (inverse differential kinematic equation) of PUMA arm ~~[2]~~. The proposed pipeline/parallel algorithm can be implemented on IC chip using systolic linear arrays. This implementation requires 27 processing cells and 25 time units. Computation time is *thus significantly reduced.*

2. Introduction

In many advanced robot control problems, such as with sensor guided manipulations, it is essential that the end effector be appropriately controlled in Cartesian coordinates so that the robot can adapt to a changing environment. This means that we need to compute the inverse Jacobian in real time to provide the required differential change in joint variables for a desired differential change in position and orientation. The speed of this computation directly affects the speed of robot operation. Thus efficient algorithms for computing the inverse Jacobian are needed.

There have been efforts made recently in developing computationally efficient algorthims to solve the Jacobian problem suitable for serial computer implementation [1,2]. In addition some work has been reported in algorithm development for implementation on pipelined or parallel computer [3]. These results show that such parallel algorithms can reduce computation time significantly.

A more important requirement in robot manipulation is the computing of the *inverse Jacobian solution.* This is generally a troublesome problem when we try to invert the Jacobian numerically. A more direct approach is to derive an explicit solution of the inverse Jacobian for a given robot. Paul, Shimano, and Mayer [2] have shown that such solutions can be obtained by differentiating the kinematic equations. This approach has shown to result simpler inverse Jacobian solutions with regard to manipulator degeneracies and joint constraints. The *inverse Jacobian of the PUMA arm* has been solved specifically in [2].

In this paper, we present a pipeline/parallel algorithm and architecture for computing the PUMA arm inverse Jacobian derived in [2]. With rapid advances in VLSI technology, this type of algorithm can be readily implemented on IC chips. These special purpose chips can be connected to a host computer system to achieve real-time Cartesian space control at sufficiently high sample rate. It is noted that a study has been made recently to implement direct kinematic solution on VLSI chips to speed up computation time [4]. The goal here is to further exploit the advantages of VLSI technology for the design of customized chips dedicated to the computing of the inverse Jocobian of PUMA arm.

3. Differential Kinematic Solution of PUMA Arm

Differential changes in joint variables dq_i can be related to the different changes in translation and rotation dx, dy, dz, δ_x, δ_y, and δ_z of the end effector by the relationship

$$[d_x,\ d_y,\ d_z,\ \delta_x,\ \delta_y,\ \delta_z,]^T = J\ [dq_1,\ dq_2,\ldots,dq_n]^T \qquad (1)$$

in which n is the number of joints, and J is the Jacobian matrix. But in advanced robot control problems, we need the solution of dq_i given the desired differential change d_x, d_y, d_z, δ_x, δ_y, δ_z . That is we need to compute the inverse problem

$$[dq_1,\ dq_2,\ldots,dq_n]^T = J^{-1}\ [d_x,\ dy,\ d_z,\ \delta x,\ \delta y,\ \delta z]^T \qquad (2)$$

This represents the inverse differential kinematic solution (inverse Jacobian) of the robot arm.

Instead of relying on the direct computing of the inverse Jacobian matrix J^{-1}, an analytical solution of the inverse Jacobian problem can be frequently obtained, and such a solution for the PUMA arm is given in [2]. For the PUMA arm, the joint variables are the six rotational joint angles θ_1, $\theta_2,\ldots,\theta_6$. Furthermore, the

differential changes in translation and rotation can be related to the differential change of the end effector homogeons matrix T [2]:

$$dT = T \cdot \Delta_T \tag{3}$$

where

$$T = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$dT = \begin{bmatrix} dn_x & do_x & da_x & dp_x \\ dn_y & do_y & da_y & dp_y \\ dn_z & do_z & da_z & dp_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Delta T = \begin{bmatrix} o & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Therefore differential changes in translation and rotation can also be specified in terms of the x, y, z elements of dp, do, and da (dn vector is redundant). The desired solutions of do_i in terms of dp, do, and da for the PUMA arm obtained in [2] are given in the appendix. A pipeline/parallel processing architecture fcr computing these equations is now developed below.

4. Systolic Array Processing

VLSI technology has created a new architecture horizon in implementing parallel algorithms directly on hardware. Central to this architecture is the use of systolic linear arrays which consist of interconnected simple and mostly identical processing cells. Algorithms that can be executed using iaentical operations simultaneously can take advantage of the systolic array architecture to reduce computation time.

The processing cell structure we will employ is the "inner product step processor" which performs matrix-vector multiplication using one-way pipeline algorithms. For example, computing

$$Ab = p \tag{4}$$

where A is nxm and b is mxl, can be carried out in the following recurrence manner:

$$p_i^{(o)} = 0$$

$$p_i^{(K+1)} = p_i^{(k)} + a_{ik} b_k \qquad k = 1,m, \quad i = 1,n$$

$$p_i = p_i^{(m)}$$

This operation can be implemented by a linear array of m inner product step processors shown in Figure 1.

In the following section, we will reformulate the inverse differential kinematic equation given in the appendix in terms of a set of matrix-vector multiplications which can be computed in parallel and pipelining fashion.

5. Algorithm Development

In this section, we present the matrix-vector multiplication processing schemes for computing the differentials do_i, $i = 1,2,\ldots,6$. Here we assume that the trigonometric functions required are available. Typically these functions can be generated by employing ROM look-up techniques [5,6]. The algorithm is broken down into 15 steps as described below. The notation $S_i \triangleq \text{Sin}o_i$ $C_i \triangleq \text{Cos}o_i$ are used.

(1) $A_1 b_1 = p_1$

$$A_1^T = \begin{bmatrix} dp_y & dp_x & p_x & p_y & a_y & a_x & da_x & -a_x & da_y & o_y & o_x & do_x & do_y \\ -dp_x & dp_y & p_y & -p_x & -a_x & a_y & da_y & -a_y & -da_y & -o_x & o_y & do_y & -do_x \end{bmatrix}$$

$b_1^T = [C_1 \; S_1]$ $\quad p_1^T = [p_{11} \cdots\cdot p_{19} \; p_{110} \cdots p_{113}]$

output: $do_1 = p_{11}/p_{13}$

(2) $f_1 \, m_1 + g_1 = h_1$

$f_1^T = [p_{14} \; p_{15} \; p_{18} \; p_{110} \; -p_{111}]$ $\quad m_1 = do_1$

$g_1^T = [p_{12} \; p_{17} \; p_{14} \; p_{112} \;\; p_{113}]$

$h_1^T = [h_{11} \cdots\cdot h_{15}]$

(3) $A_2 b_2 = p_2$

$$A_2^T = \begin{bmatrix} -a_3 & d_4 & p_z & -p_{13} \\ d_4 & a_3 & p_{13} & p_z \end{bmatrix} \qquad b_2 = \begin{bmatrix} S_3 \\ C_3 \end{bmatrix}$$

$p_2^T = [p_{21} \cdots\cdot p_{24}]$

(4) $f_2 \, m_2 + g_2 = h_2$

$f_2^T = [C_3 \; S_3 \; P_{21} \; P_{23} \; P_{24}]$

$g_2^T = [a_3 \; d_4 \; 0 \quad 0 \quad 0]$

$m_2 = a_2, \quad h_2^T = [h_{21} \cdots\cdot h_{25}]$

(5) $A_3 b_3 = p_3$

$A_3 = [-p_z \; p_{13}]$ $\quad b_3^T = [dp_z \; h_{11}]$ $\quad p_3 = p_{31}]$

output: $do_3 = p_{31}/h_{23}$

(6) $A_4 b_4 = p_4$

$$A_4 = \begin{bmatrix} p_z & -p_{13} \\ p_{13} & p_z \end{bmatrix} \qquad b_4 = \begin{bmatrix} h_{21} \\ h_{22} \end{bmatrix} \qquad p_4 = \begin{bmatrix} p_{41} \\ p_{42} \end{bmatrix}$$

(7) $A_5 b_5 = p_5$

$$a_5 = \begin{bmatrix} -h_{21} & h_{22} & -h_{24} \\ h_{22} & h_{21} & h_{25} \end{bmatrix} \qquad b_5^T = [dp_z \; h_{11} \; do_3]$$

$p_5^T = [p_{51} \; p_{52}]$

(8) $A_6 b_6 = p_6$

$$A_6^T = \begin{bmatrix} p_{42} & -a_z & p_{16} & -p_{51} & h_{12} & da_z & p_{111} & -o_z & h_{14} & -do_z \\ -p_{41} & -p_{16} & -a_z & -p_{52} & -da_z & h_{12} & -o_z & -p_{111} & -do_z & -h_{14} \end{bmatrix}$$

$b_6^T = [c_{23} \; s_{23}]$ $\quad p_6^T = [p_{61} \cdots\cdot p_{610}]$

outputs: $do_{23} = p_{64}/p_{61}$ $\quad do_2 = do_{23} - do_3$

(9) $f_3 m_3 + g_3 = h_3$

$f_3^T = [p_3\ \ p_{63}\ p_{68}\ -p_{67}]$ $\qquad m_3 = do_{23}$

$g_3^T = [p_{65}\ p_{66}\ p_{69}\ p_{610}]$

$h_3^T = [h_{31}\cdots\cdot h_{34}]$

(10) $A_7 b_7 = p_7$

$$A_7 = \begin{bmatrix} p_{15} & p_{63} \\ p_{32} & -h_{13} \end{bmatrix} \qquad b_7 = \begin{bmatrix} p_{15} \\ p_{63} \end{bmatrix} \qquad p_7 = \begin{bmatrix} p_{71} \\ p_{72} \end{bmatrix}$$

output: $do_4 = p_{72}/p_{11}$

(11) $A_8 b_8 = p_8$

$$A_8^T = \begin{bmatrix} p_{15} & p_{31} & p_{67} & p_{110} & h_{33} & -p_{67} & h_{15} \\ -p_{63} & h_{13} & p_{116} & -p_{67} & h_{15} & -p_{110} & -h_{33} \end{bmatrix}$$

$b_8^T = [C_4\ S_4]$ $\qquad p_8^T = [p_{81}\cdots\cdot p_{87}]$

(12) $f_4 m_4 + g_4 = h_4$

$f_3^T = [p_{81}\ p_{84}\ p_{86}]$ $\qquad m_4 = do_4$

$g_4^T = [p_{82}\ p_{85}\ p_{87}]$ $\qquad h_4^T = [h_{41}\ h_{42}\ h_{43}]$

(13) $A_9 b_9 = p_9$

$$A_9 = \begin{bmatrix} h_{41} & -h_{32} \\ -p_{68} & p_{83} \\ -h_{42} & -h_{34} \end{bmatrix} \qquad b_9 = \begin{bmatrix} c_5 \\ s_5 \end{bmatrix} \qquad p_9 = \begin{bmatrix} p_{91} \\ p_{92} \\ p_{93} \end{bmatrix}$$

output: $do_5 = p_{91}$

(14) $f_5 m_5 + g_5 = h_5$

$f_5 = [p_{92}]$ $\qquad g_5 = [p_{93}]$ $\qquad m_5 = do_5$ $\qquad h_5 = [h_{51}]$

(15) $A_{10} b_{10} = p_{10}$

$A_{10} = [h_{51}\ -h_{43}]$ $\qquad b_{10}^T = [c_6\ s_6]$ $\qquad p_{10} = [p_{101}]$

output: $do_6 = p_{101}$

The data flow timing table for these computations are given in Tables 1 and 2. It is shown that the solution requires 25 time units and 27 processing cells.

The results of 25 time units is a significant reduction of computation time in comparison with that when a serial computer is used to compute the original solution. The total number of multiplications of that solution is about 150. This is equivalent to 150 time units in the systoic array processing system as opposed to the 25 time units we have achieved by exploiting parallelism.

Table 1. Data flow timing table for steps 1 through 8 which compute $d\theta_1$ $d\theta_3$ and $d\theta_3$. Numbers on top row indicate time units.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
dP_y	$-dP_x$																C_1
	dP_x	dP_y															S_1
		P_x	P_y														
			P_y	$-P_x$													
				a_y	$-a_x$												
					a_x	a_y											
						da_x	da_y										
							$-a_x$	$-a_y$									
								da_y	$-da_x$								
									o_y	o_x							
										o_x	o_y						
											do_x	do_y					
												do_y	$-do_x$				
					p_{14}												do_1, p_{12}
								p_{15}									p_{17}
										p_{18}							p_{19}
													$-p_{110}$				p_{112}
														p_{111}			p_{113}
	$-a_3$	d_4															C_3
		d_4	a_3														S_3
			p_z	p_{13}													
				$-p_{13}$	p_z												
	C_3																a_2, a_3
		S_3															d_4
			p_{21}														
					p_{23}												
						p_{24}											
					$-p_z$												dp_z
						p_{13}											h_{11}
			p_z	$-p_{13}$													h_{21}
				p_{13}	p_z												h_{22}
						h_{21}	$-h_{22}$	$-h_{24}$									dp_z
							h_{22}	h_{21}	h_{25}								h_{11}, do_3
						p_{42}	p_{41}										C_{23}
							$-a_z$	$-p_{16}$									S_{23}
								p_{16}	$-a_z$								
									$-p_{51}$	$-p_{52}$							
										h_{12}	$-da_z$						
											da_z	h_{12}					
												p_{111}	$-o_z$				
													$-o_z$	$-p_{111}$			
														h_{14}	$-do_z$		
															$-do_z$	$-h_{14}$	

Table 2. Data flow timing table for steps 9 through 15 which compute $d\theta_4$, $d\theta_5$, $d\theta_6$.

10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
			p_{64}													p_{65}	
				p_{63}												p_{66}	
							p_{68}									p_{69}	
								$-p_{67}$								p_{610}	
p_{15}	p_{63}															p_{15}	
				h_{31}	$-h_{13}$											p_{63}	
				p_{15}	$-p_{63}$											C_4	
					h_{31}	h_{13}										S_4	
						p_{67}	p_{110}										
							p_{110}	$-p_{67}$									
								h_{33}	h_{15}								
									$-p_{67}$	$-p_{110}$							
									h_{15}	$-h_{33}$							
							p_{81}									$d\theta_4$,	p_{82}
										p_{86}						p_{85}	
												p_{86}				p_{87}	
									h_{41}	$-h_{32}$						C_5	
										-pGG	p_{83}					S_5	
											$-h_{42}$	$-h_{34}$					
													p_{92}			$d\theta_5$,	p_{93}
													$-h_{43}$	h_{51}		C_6,	S_6

6. Conclusion

It has been demonstrated in this paper that parallel computing architecture can be developed for the inverse differential kinematic equation of the PUMA arm. By using systolic linear arrays employed in VLSI chip design, the computation can be completed with 27 processing cells in 25 time units.

The differential kinematic equation in its original form requires about 150 multiplications to compute. If one multiplication is counted as one time unit, the parallel architecture definitely provides a substantial reduction in computaticn time. A customized IC chip dedicated to this algorithm can be fabricated.

7. References

[1] D. E. Orin and W. W. Schrader, "Efficient Jacobian Determination for Robot Manipulators," Robotics Research, Brady and Paul, Editors, pp. 727-734, MIT Press, 1984.

[2] R. P. Paul, B. Shimano, and G. E. Mayer, "Differential Kinematic Control Equations for Simple Manipulators," IEEE Transactions on Systems, Man, and Cybernetics, Vol., SMC-11, No. 6, pp. 456-460, June 1981.

[3] D. E. Orin, H. H. Chao, K. W. Olson, and W. W. Schrader, "Pipeline/Parallel Algorithms for the Jacobian and Inverse Dynamics Computations," proceedings, 1985 IEEE International Conference on Robotics and Automation, pp. 785-789, 1986.

[4] S. S. Leung and M. A. Shanblatt, "A VLSI Chip Architecture for the Real-Time Computation of Direct Kinematics," proceedings, 1986 IEEE International Conference on Robotics and Automation, pp. 1717-1722, 1986.

[5] S. Muroga, VLSI System Design, When and How to Design Very-Large-Scale Integrated Circuits, Wiley, 1982, pp. 313-316.

[6] C. F. Ruoff, "Fast Trig Functions for Robot Control," Robotics Age in the Beginning, C. T. Helmers, ed., Hasbronck Heights, Hayden Book, 1983, pp. 73-79.

Appendix: Inverse Differential Kinematic Equations for PUMA arm

$$d\theta_1 = \frac{C_1dP_y - S_1dP_x}{C_1P_x + S_1P_y}$$

$$d\theta_3 = \frac{d_1}{a_2(d_4C_3 - a_3S_3)}$$

$$d_1 = f_{11}(p)df_{11}(p) - f_{12}(p)df_{12}(p)$$

$$f_{11}(p) = C_1p_x = S_1P_y$$

$$df_{11}(p) = -S_1p_xd\theta_1 + C_1dp_x + C_1p_yd\theta_1 + S_1dp_y$$

$$f_{12}(p) = -dp_z$$

$$d\theta_{23} = \frac{-S_{23}d\upsilon_2 - C_{23}d\upsilon_1}{C_{23}\nu_2 - S_{23}\nu_1}$$

$$\nu_1 = -\omega_2f_{11}(p) + \omega_1p_z$$

$$\nu_2 = \omega_1f_{11}(p) + \omega_2p_z$$

$$\omega_1 = a_2C_3 + a_3$$

$$\omega_2 = d_4 + a_2S_3$$

$$d\upsilon_1 = -\omega_2df_{11}(p) -{}_aA_2f_{11}(p)C_3d\theta_3 + \omega_1dP_z - a_2S_3p_zd\theta_3$$

$$d\upsilon_2 = \omega_1df_{11}(p) - a_2S_3f_{11}(p)d\theta_3 + \omega_2dP_z + a_2C_3p_3d\theta_3$$

$$d\theta_2 = d\theta_{23} - d\theta_3$$

$$d\theta_4 = \frac{NC_4d(NS_4) - NS_4d(NC_4)}{(NS_4)^2 + (NC_4)^2}$$

$$NC_4 = C_{23}D_{41} - S_{23}a_z$$

$$D_{41} = C_1a_x + S_1a_y$$

$$d(NS_4) = -C_1a_xda_1 - S_1a_yd\theta_1 + C_1da_z - S_1da_x$$

$$NS_4 = -S_1a_x + C_1a_y$$

$$d(NC_4) = -S_{23}D_{41}d\theta_{23} + C_{23}dD_{41} \cdot - C_{23}a_zd\theta_{23} - S_{23}da_z$$

$$dD_{41} = -S_1a_xd\theta_1 + C_1da_x + C_1a_yd\theta_1 + S_1da_z$$

$$d\theta_5 = C_5dS_5 - S_5dC_5$$

$$dS_5 = -S_4NC_4d\theta_4 + C_4d(NC_4) + C_4d\theta_4NS_4 + S_4d(NS_4)$$

$$dC_5 = C_{23}d\theta_{23}D_{41} + S_{23}dD_{41} - S_{23}a_zd\theta_{23} + C_{23}da_z$$

$$S_5 = C_4NC_4 + S_4NS_4$$

$$C_5 = S_{23}D_{41} + C_{23}a_z$$

$d\theta_6 = C_6 dS_6 - S_6 dC_6$

$S_6 = -C_5 N_{61} - S_5 N_{612}$

$C_6 = -S_4 N_{611} + C_4 N_{6112}$

$dS_6 = S_5 N_{61} d\theta_5 - C_5 dN_{61} - C_5 N_{612} d\theta_5 - S_5 dN_{612}$

$dC_6 = -dS_4 N_{611} - S_4 dN_{611} + dC_4 N_{6112} + C_4 dN_{6112}$

$= -C_4 N_{611} d\theta_4 - S_4 dN_{611} + - S_4 N_{6112} d\theta_4 + C_4 dN_{6112}$

$N_{61} = C_4 N_{611} + S_4 N_{6112}$

$N_{611} = C_{23} N_{6111} - S_{23} O_z$

$N_{6112} = -S_1 O_x + C_1 O_y$

$dN_{61} = -S_4 N_{611} d\theta_4 + C_4 dN_{611} + C_4 N_{6112} d\theta_4 + S_4 dN_{612}$

$dN_{611} = -S_{23} N_{6111} d\theta_{23} + C_{23} dN_{6111} - C_{23} O_z d\theta_{23} - S_{23} dO_z$

$dN_{6112} = - C_1 O_x d\theta_1 - S_1 dO_x - S_1 O_y d\theta_1 + C_1 dO_y$

$N_{6111} = C_1 O_x + S_1 O_y$

$dN_{6111} = -S_1 O_x d\theta_1 + C_1 dO_x + C_1 O_y d\theta_1 + S_1 dO_y$

$N_{612} = -S_{23} N_{6111} - C_{23} O_z$

$dN_{612} = -C_{23} N_{6111} d\theta_{23} - S_{23} dN_{1111} + S_{23} O_z d\theta_{23} - C_{23} dO_z$

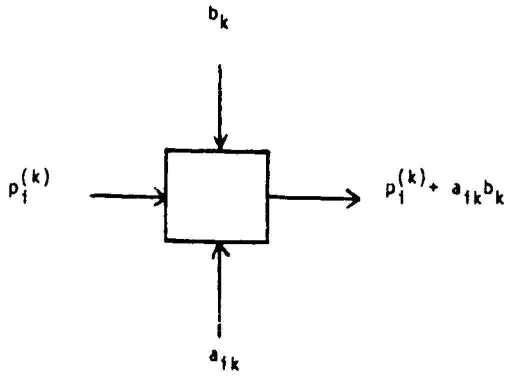


Figure 1. Inner product step processor

NASA S32-63

188200
10P

Manipulator Control by Exact Linearization

K. Kreutz
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

1. Abstract.

Comments on the application to rigid link manipulators of Geometric Control Theory, Resolved Acceleration Control, Operational Space Control, and Nonlinear Decoupling Theory are given, and the essential unity of these techniques for externally linearizing and decoupling end effector dynamics is discussed. Exploiting the fact that the mass matrix of a rigid link manipulator is positive definite - a consequence of rigid link manipulators belonging to the class of natural physical systems - it is shown that a necessary and sufficient condition for a locally externally linearizing and output decoupling feedback law to exist is that the end effector Jacobian matrix be nonsingular. Furthermore, this linearizing feedback is easy to produce.

2. Introduction.

Because of the difficulty in controlling rigid link manipulators, along with a primary concern in controlling end effector (EF) motions, it is natural to ask if a nonlinear feedback law exists which will make an EF behave as if it has linear and decoupled dynamics. It has been known at least since the early 1970s [1]-[5] that exact linearization of manipulators in joint space is readily accomplished by the so-called Inverse or Computed Torque Technique. Efforts to accomplish decoupled linearization of EF motions directly in task space began soon thereafter as is evident in the work of [6]-[14].

The work of [6], although concerned only with controlling the tip location of a three-link manipulator in the plane, is surprisingly prescient in its approach in that it proceeds by the three explicit steps of 1) decoupled linearization of tip behavior; 2) stabilization of the resulting tip dynamics; followed by 3) trajectory control of the now linearly behaving tip. Such clarity of approach will only be retrieved in the latter work of [19]-[22]. The work [6] also presages future work in its dealing with the problems of manipulator redundancy and actuator saturation.

With hindsight, the work [6] can also be viewed as a direct precursor to the development of the Resolved Acceleration Control (RAC) approach to the end effector tracking problem [7][8]. RAC essentially extends the work of [6] to the case of a full six dof manipulator yielding linearized EF positional error dynamics and almost linearized EF attitude error dynamics (the extent to which attitude error dynamics are "almost" linearized will be discussed below). The work of [7][8], however, did not make clear the three steps of [6] and consequently appears to have not been appreciated as a technique for performing decoupled exact linearization of EF motions, but rather as a technique for end effector tracking which has (almost) linear tracking error dynamics. The fact that the attitude error dynamics are not completely linearized also apparently obscured the appreciation of RAC as an exactly linearizing control technique.

The work of [9]-[11] applies Nonlinear Decoupling Theory (NDT) to provide decoupled linearization of a manipulator EF with simultaneous pole placement of the linearized EF dynamics. The abstruse formulation of this approach has apparently discouraged serious comparison with other approaches, the notable exception being [23] where correspondences to RAC and the Computed Torque technique have been noted. The simultaneous pole placement and linearization of EF dynamics represents a blurring of the distinct steps 1 and 2 described above for the approach [6].

In [12]-[14], manipulator dynamics are expressed in the task space, or Operational Space of the EF. The resulting nonlinear effective end effector dynamics are then linearized by the Computed Torque method. Thus, the Operational Space Control (OCS) of [12]-[14] can also be viewed as a Generalized Computed Torque technique. In [12] correspondences to RAC and the Computed Torque technique have been noted.

Recently, Geometric Control Theory (GCT) based techniques for exactly externally linearizing and decoupling general affine-in-the-input nonlinear systems have been developed [15]-[19]. These techniques provide constructive sufficient conditions for local decoupled external linearization which, if satisfied, produces the linearizing feedback law. GCT has been applied to exactly linearizing end effector motions in [19]-[22]. The work of [19]-[22] also provides a clear and mature control perspective which keeps the following steps distinct: 1) Exactly linearize and decouple end effector dynamics to a canonical decoupled double integrator form, i.e. to Brunovsky Canonical Form (BCF); 2) Effect a stabilizing loop (pole placement step); 3) Perform feedforward precompensation to obtain nominal model following performance; 4) Institute an LQR error correcting feedback loop. Unfortunately, to understand the theoretical underpinnings of GCT requires an exposure to differential geometry and Lie algebra/Lie group theory which most practicing engineers are unlikely to have.

It can be shown that all of the above seemingly quite different approaches lead to the same linearising control law for exact external linearisation and decoupling of EF motions [24]. (This equivalence is specific to the nonlinear systems considered here, viz. systems dynamically similar to rigid link manipulators. NDT and GCT apply to a much larger class than this, and so the equivalence to RAC and OSC holds for systems restricted to this class but not in general). Recognising this equivalence enables us to give a simple necessary and sufficient condition for local decoupled external linearisation and to give a simple form for the linearising control which is applicable to a broad class of so-called natural physical dynamical systems [25] [26] of which a serial link manipulator is but a special case. For brevity we do not discuss actuated redundant arms - for discussion of these cases, see [24].

3. Dynamics of Finite Dimensional Natural Systems.

Many physical systems have finite dimensional nonlinear dynamics of the form [25][26]:

$$M(q)\ddot{q} + V(q,\dot{q}) = \tau;\; q \in N^n;\; \dot{q},\ddot{q} \in R^n; \qquad (1$$

$$M(q) \in R^{n\times n};\; M(q) = M^T(q) > 0,\ \forall q$$

where q evolves on a manifold of dimension n. For example $q \in R^n$ for a Cartesian manipulator, while $q \in T^n$ for a revolute manipulator.

Typically (1) arises as a solution to the Lagrange equations:

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}} L - \frac{\partial L}{\partial q} = Q^T$$

where $L = T-U$, $T = 1/2\ \dot{q}^T M(q)\dot{q}$ is positive definite and autonomous, U is a conservative potential function, $Q = \tau + F$ are generalized forces, and F are dissipative or constraint forces. This is exactly how manipulator dynamics are obtained and hence manipulator dynamics are precisely of the form (1). Systems which arise in this way are known as natural systems [25][26]. It is known that for natural systems not only is M(q) positive definite, but V(q,q) of (1) has terms which depend on M(q) in a very special way [27]-[29]. In fact, natural systems are nongeneric in the class of all affine-in-the-input nonlinear systems [38][39]. Although we shall only exploit the fact that M(q) is positive definite for any q, it is worth noting that the nongeneric structure of (1) has recently enabled important statements to be made on the existence of time optimal control laws [38]-[40], on the existence of globally stable control laws [27]-[33], on the existence of robust exponentially stable control laws [34], and on the existence of stable adaptive control laws [35]-[37] for the natural system (1).

Recognizing the special properties of the system (1), it is not surprising that results yielding externall linearizing behavior can be obtained much more easily than by application of NDT or GCT - theories which apply to the whole general class of smooth affine-in-the-input nonlinear systems.

4. End Effector Kinematics and Control After Linearization.

The system (1) is assumed to have a read-out map of either the form

$$y = h(q) \in R^m,\ \mathcal{V} = \dot{y} = J(q)\dot{q} \in R^m,\ J(q) = \frac{\partial h}{\partial q} \in R^{m\times n} \qquad (2$$

or of the more general form

$$y = h(q) \in M^m,\ \mathcal{V} = J_o(q)\dot{q} \in R^m,\ J_o(q) \in R^{m\times n} \qquad (3$$

where $J_o dt - \mathcal{V}dt$ is a general, perhaps nonintegrable, Pfaffian form [25][26], $h(\cdot)$ is C^2 [44][47] and defined on N^n, M^m is some m dimensional output manifold, J or J_o is C^1, and in general m and n have different values. Often $h(\cdot)$ is smooth (i.e. C^∞) or even a diffeomorphism when the domain is suitably restricted. In subsequent discussion $\mathcal{V} = \bar{J}\dot{q}$ will mean that $\bar{J}$ can be either J or J_o. Let the state of system (1) be $(q,\dot{q})$. Then for $y = h(q)$, $\mathcal{V} = \bar{J}(q)\dot{q}$ will be called the "velocity associated with the output y." Note that (2) is a special case of (3) where $\mathcal{V}$, the velocity associated with y, is just $\mathcal{V} = \dot{y}$ and $M^m = R^m$ giving $\bar{J} = J = \partial h/\partial q$. Also note that for the case (3), since h is C^2, it is still meaningful to talk about $\dot{y} = J\dot{q}$ and $J = \partial h/\partial q$, $J(q)$: $T_q N^n = R^n \bullet T_{h(q)}M^m$, but now the case where $\mathcal{V} \neq \dot{y}$ is admitted as a possibility.

For rigid link manipulators moving in Euclidean 3-space, typically $\mathcal{V} = \binom{\dot{x}}{\omega} \in R^6$, where $x \in R^3$ gives the EF location, $\dot{x}$ the EF linear velocity, and $\omega \in R^3$ the EF angular rate of change. It is well known that ω is not the time derivative of any minimal (i.e. 3 dimensional) representation of attitude, so that $\mathcal{V} = \binom{\dot{x}}{\omega} = J_o(q)\dot{q}$ as in (3). In this case, we call $J_o(q)$ the "Standard Jacobian." It is also common to represent EF attitude by a proper orthogonal matrix $A \in R^{3x3}$,

$$A \in SO(3) = \left\{ A \middle| A^TA = AA^T = I, \det A = +1 \right\},$$

where the columns of A determine EF fixed body axes in the usual way. It is well known that $A = \tilde{\omega}A$ where $\tilde{\omega}v: = \omega \times v$ for all $v \in R^3$. Thus EF location and kinematics are often given by

$$y = (x,A) = h(q) \in R^3 \times SO(3), \quad \mathcal{V} = \binom{\dot{x}}{\omega} = J_o(q)q \in R^6 \quad \dot{A} = \tilde{\omega}A,\ A \in SO(3),\ \tilde{\omega}^T = -\tilde{\omega},\ J_o(q) \in R^{6xn}, \tag{4}$$

which should be compared to (3). Alternatively, we can take (cf. (2))

$$y = \binom{x}{\beta} = h(q) \in R^6, \quad \mathcal{V} = \dot{y} = \binom{\dot{x}}{\dot{\beta}} = J(q)\dot{q} \in R^6,\ \beta \in \Omega \subset R^3. \tag{5}$$

$\beta \in \Omega \subset R^3$ is a minimal representation of EF attitude (i.e. of the rotation group SO(3)). In general $\beta = f(A)$ for some function $f(\cdot)$ which is many-to-one or undefined if the domain of $f(\cdot)$ on SO(3) is not properly restricted. That is, because SO(3) cannot be covered by a single coordinate chart, β is not valid for all possible EF orientations and there will be singularity of attitude representation unless we restrict EF attitude to the region of SO(3) for which β is valid [25] [41][42]. This restriction then forces β to be defined in the image of admissible attitudes, namely in some $\Omega \subset R^3$. (It may be true, however, that $\Omega = R^3$ as in the case of Euler-Rodriques parameters where singularity of attitude representation corresponds to $\|\beta\| = \infty$ [42]). Typical β's are roll-pitch-yaw angles, axis/angle variables, Euler angles, Euler parameters, and Euler-Rodriques parameters [25], [41]-[43]. The kinematical relationship between β and ω is given by

$$\dot{\beta} = \Pi(\beta)\omega \tag{6}$$

where $\Pi \in R^{3x3}$ will lose, rank, i.e. become singular, precisely when β becomes a singular representation of EF attitude. Note from (3)-(6) that

$$J = \begin{pmatrix} I & 0 \\ 0 & \Pi \end{pmatrix} J_o.$$

Generally, the standard Jacobian matrix J_o will become singular only at a manipulator kinematic singularity, in which case J will also be singular. Furthermore, J will be singular when $\beta = \beta(q)$ gives a singularity of EF attitude representation. This compounds the trajectory planning problem for EF motions, since now we must plan trajectories which avoid manipulator kinematic singularities and also ensure that $\beta(q) \in \Omega$.

Henceforth the system (1), (2) or (1), (3) will be said to be exactly externally linearized and decoupled if

$$\dot{\mathcal{V}} = u \in R^6. \tag{7}$$

This is somewhat of an abuse of notation as a consideration of the system (1), (4) shows. For $u = \binom{u_1}{u_2}$, $\dot{\mathcal{V}} = u$ yields

$$\ddot{x} = u_1 \in R^3, \quad \dot{\omega} = u_2 \in R^3, \quad \dot{A} = \tilde{\omega}A. \tag{8}$$

Although EF positional dynamics are decoupled and linearized to $\ddot{x} = u$, attitude dynamics are nonlinear and given by $\dot{\omega} = u_2$, $\dot{A} = \tilde{\omega}A$. Eq. (8) is precisely the sense in which RAC can be said to almost "exactly externally linearize and decouple" attitude error dynamics as was discussed in the introduction. In the case of the system (1), (5), $\dot{\mathcal{V}} = \binom{u_1}{u_2}$ gives

$$\ddot{x} = u_1 \in R^3, \quad \ddot{\beta} = u_2 \in R^3, \quad \beta \in \Omega \subset R^3, \tag{9}$$

which can indeed be said to be exactly externally linearized and decoupled. Drawbacks to using (9) are that β must always be controlled to remain in Ω, trajectories involving β may be difficult to visualize, and the generalized force, u_2, which drives β may be nonintuitive. On the other hand, it is obvious how to obtain stable attitude tracking from (9). The advantage to using (8) is that ω and A are easily visualized entities, while u_2 is the ordinary torque that we are all familiar with. Fortunately, despite the nonlinear attitude dynamics, it is possible to use (8) to perform EF attitude tracking with asymptotically vanishing attitude error [7] [8].

Note that once (8) is obtained, it is easy to get (9) by use of the relationship (6). If we have $\dot{\omega} = u$, $\dot{\beta} = \Pi(\beta)\omega$, and $\beta \in \Omega$ so that $\Pi^{-1}(\beta)$ exists, then use of

$$\dot{\omega} = u, \quad u = \Pi^{-1}(\beta)\,(\bar{u} - \dot{\Pi}(\beta)\omega) \tag{10}$$

gives

$$\ddot{\beta} = \Pi(\beta)\dot{\omega} + \dot{\Pi}(\beta)\omega = \bar{u}. \tag{11}$$

Therefore, having (8), we can perform attitude control directly on $\dot{\omega} = u_2$, $\dot{A} = \tilde{\omega}A$ or we can transform to $\ddot{\beta} = \bar{u}_2$ and then control.

5. Comparison of GCT, NDT, and OSC.

For brevity, we consider the non-redundant manipulator case, taking $n = 6$ in (1), and we omit derivations. A more detailed discussion is given in [24].

Note that the system (1), (5) can be written as

$$\frac{d}{dt}\begin{pmatrix} q \\ \dot{q} \end{pmatrix} = \begin{pmatrix} \dot{q} \\ -M^{-1}V \end{pmatrix} + \begin{pmatrix} 0 \\ M^{-1} \end{pmatrix}\tau, \quad y = h(q) \tag{12}$$

or, taking $Z = \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$,

$$\frac{d}{dt} Z = A(Z) + B(Z)\tau, \; y = H(Z) \tag{13}$$

where the definitions of A, B, and H are obvious. GCT asks: does there exist (i) a nonlinear feedback $\tau = Q(Z) + B(Z)u$ and (ii) a nonlinear change of basis $x = X(Z)$ such that (12) is placed into BCF?:

$$\frac{d}{dt}\begin{pmatrix} y \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix}\begin{pmatrix} y \\ \dot{y} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix}u \iff \ddot{y} = u. \tag{14}$$

The constructive sufficient conditions of [19]-[22] can be applied and give the following linearizing and decoupling feedback law:

$$\tau = -MJ^{-1}\,\partial J\dot{q} + V + MJ^{-1}u$$

where

$$\partial J = \left[\sum_{k=1}^{n} \frac{\partial J_{ik}}{\partial q_j}\,\dot{q}_k\right].$$

Although $\partial J \neq J$, it is true that $\partial J\dot{q} = \dot{J}\dot{q}$ giving

$$\tau = -MJ^{-1}\dot{J}\dot{q} + MJ^{-1}u + V. \tag{15}$$

Note that J must be nonsingular for (15) to exist. This is consistent with the theory of [19]-[12] which provides sufficient conditions for <u>local</u> linearization. Note also that to implement (15), explicit expressions for M, J^{-1}, J, and V are required.

The NDT approach of [11], constructs the linearizing feedback in the following way. For the system (13) define

$$G(Z) = \frac{\partial}{\partial Z}\left(\dot{\left[\frac{\partial}{\partial Z} C(Z)\right]}\cdot A\ (Z)\right),\ D^*(Z) = G(Z)\cdot B(Z),\ C^*(Z) = G(A)\cdot A(Z). \tag{16}$$

The use of

$$\tau = -D^{*-1}(Z)\ C^*(Z) + D^{*-1}(Z)u \tag{17}$$

will transform (12), (13) to $\ddot{y} = u$, i.e. to (14).

It is straightforward to show that, for A, B, and C as in (12) and (13), eq. (17) is precisely eq. (15). Note that in (13) we take $Z = \binom{q}{\dot{q}}$ and not $Z = (q_1, \dot{q}_1, \ldots, q_n, \dot{q}_n)^T$. The latter choice for Z is taken in [11] and serves to obscure the final result – namely that (17) and (15) are equivalent.

Now consider the OSC approach of [12]-[14]. In (1) let V = B-G where B is the coriolis forces and G the gravity forces. Restrict the domain of the system (1), (5) to ensure that $h(\cdot)$, is a bijection (and consequently $\det J(q) \neq 0$ on this restriction). This restriction means that, as for DGC and NDT, the following gives a local result for external linearization. In [12]-[14], the effective EF dynamics are determined to be

$$\Lambda(y)\ddot{y} + C(y,\dot{y}) = F,\ \tau = J^T F, \qquad \Lambda = J^{-T}MJ,\ C = U-P,$$
$$P = J^{-1}G,\ U = J^{-T}B - \Lambda\dot{J}\dot{q}, \quad q = h^{-1}(y),\ \dot{q} = J^{-1}\dot{y} \tag{18}$$

Recall that for the system (1), $M\ddot{q} + V = \tau$, the Computed Torque technique is to take $\tau = Mu + V$, yielding $M(\ddot{q}-u) = 0 \Rightarrow \ddot{q} = u$, since $M(q) > 0, \forall q$. Similarly, in (18) $\Lambda(y) > 0$ for every $y = h(q)$ where q is on the restricted domain. Therefore a choice of

$$F = \Lambda(y)u + C(y,\dot{y}), \quad \tau = J^T F \tag{19}$$

in (18) yields $\Lambda(y)\ (\ddot{y}-u) = 0 \Rightarrow \ddot{y} = u$. In this sense the work in [12]-[14] can be viewed as a Generalized Computed Torque technique. From (18) and (19) it is straight forward to determine that τ of (19) is exactly τ of eq. (15).

6. Derivation of a Feedback Law for Local Exact Decoupled External Linearization and Its Relationship to RAC and GCT.

Recall that the system (1), (7) or (1), (3) is of the form

$$M(q)\ \ddot{q} + V(q,\dot{q}) = \tau;\ q \in N^n;\ \dot{q},\ \ddot{q} \in R^n$$
$$y = h(q) \in M^m;\ h(\cdot)\ \text{is}\ C^2;\ \mathcal{V} = \bar{J}(q)\ \dot{q} \in R^m;\ \bar{J}(q)\ \text{is}\ C^1; \tag{20}$$
$$M(q) \in R^{nxn};\ M(q) = M(q)^T > 0, \quad q \in N^n$$

where in general, it may be that $m \neq n$, $M^m \neq R^m$, $\mathcal{V} \neq \dot{y}$, and $\bar{J} \neq J = \partial h/\partial q$.

It is assumed that a necessary and sufficient condition for h(q) to be onto some neighborhood of y = h(q) in M^m is that the mapping $\bar{J}(q)$ be onto R^m, i.e. we assume that $\bar{J}(q)$ is onto R^m if and only if $J(q) = \partial h(q)/\partial q$ is onto $T_{h(q)}M^m = R^m$. This is a reasonable assumption; for example, when $M^m = R^m$, $\mathcal{V} = \dot{y} = J(q)\dot{q} \in R^m$, and $\bar{J} = J = \partial h/\partial q$ this is trivially true. For the case $y = h(q) = (x,A) \in M^6 = R^3 \times SO(3)$ and $\bar{J} = J_o$ where $\mathcal{V} = \binom{\dot{x}}{\omega} = J_o(q)\dot{q}$, the fact that $\dot{x} \in T_x R^3 = R^3$ and $\dot{A} = \tilde{\omega}A \in T_A SO(3)$ means that for $J_o(q)$ onto, we can fill out a neighborhood of (x,A) and otherwise we cannot. (A general element of $T_A SO(3)$ is precisely of the form $\tilde{\omega}A$, $\tilde{\omega} \in R^{3x3}$ skew-symmetric, so that if $\omega = \omega(q)$ can be mapped onto R^3, $\tilde{\omega}A$ can be mapped onto $T_A SO(3)$ [44][47].)

<u>Definition LEL</u>: The system (20) can be locally exactly linearized and decoupled (LEL) over an open neighborhood $B^m(y') \subset M^m$ of $y' \in h(N^n) \subset M^m$ with the arm in the configuration $q' \in h^{-1}(y')$ if there is an open neighborhood of q', $B^n(q') \subset N^n$, such that $B^m(y') = h(B^n(q'))$ and if for any $u \in R^m$ and $q \in B^n(q')$ there exists

a nonlinear feedback $\tau = F(q,\dot{q},u)$ such that $\mathcal{V}$, the velocity associated with $y = h(q) \in B^m(y')$, obeys $\dot{\mathcal{V}} = u$.

Note that for an EF to be LEL at y' it must be true that y' be in the range of $h(\cdot)$, i.e. y' must be a physically attainable EF position. Also for a given EF location, $y' \in h(N^n)$, a manipulator can physically be in only one of the possible configurations $h^{-1}(y')$. Thus we can interpret $q' \in h^{-1}(y')$ to be the actual physical configuration of a manipulator. If the system (20) is not LEL at y' in the configuration $q' \in h^{-1}(y')$ it may be LEL at a different configuration $q'' \in h^{-1}(y')$.

<u>Theorem LEL</u>: A necessary and sufficient condition for (20) to be LEL at $y' \in h\,(N^n)$ in the configuration $q' \in h^{-1}(y')$ is that $\bar{J}\,(q') \in R^{m\times n}$ be onto, which is true iff $m \le n$ and rank $\bar{J}(q') = m$. Furthermore, the locally exactly linearizing and decoupling feedback is given by

$$\tau = M(q)\ \xi + V(q,\dot{q}) \tag{21}$$

where ξ is any solution to

$$\bar{J}(q)\xi = -\dot{\bar{J}}(q)\dot{q} + u. \tag{22}$$

When $m = n$ this gives

$$\tau = -M(q)\bar{J}(q)^{-1}\dot{\bar{J}}(q)\dot{q} + M(q)\bar{J}(q)^{-1}u + V(q,\dot{q}). \tag{23}$$

□

Proof. Necessity: Suppose that $\dot{\mathcal{V}} = \bar{J}(q')\ddot{q}' + \dot{\bar{J}}(q')\dot{q}' = u$ can be made to hold regardless of the value of $u \in R^m$. This means that there must exist $\ddot{q} \in R^n$ such that

$$\bar{J}(q')\ddot{q}' = -\dot{\bar{J}}(q')\dot{q}' + u. \tag{24}$$

If $\bar{J}(q')$ is not onto, then Im $\bar{J}(q') \subset R^m$ and Im $\bar{J}(q') \neq R^m$. Let u be such that $-\dot{\bar{J}}(q')\dot{q}' + u \notin$ Im $\bar{J}(q')$. Then there is no $\ddot{q}'$ for which (24) holds, yielding a contradiction. Sufficiency: By assumption $\bar{J}(q')$ is full rank and onto $\Leftrightarrow J(q') = \partial h(q')/\partial q$ is full rank and onto. Since $\bar{J}$ and J are C^1, there exists neighborhoods $B^m(y')$ and $B^n(q')$, $y' = h(q')$, such that $B^m(y') = h(B^n(q'))$ and such that $\bar{J}$ is full rank and onto when restricted to $B^n(q')$. Now consider any $q \in B^n(q')$ and its associated $y = h(q) \in B^m(y')$. Then, $\mathcal{V} = \bar{J}(q)\dot{q} \Rightarrow$

$$\dot{\mathcal{V}} = \bar{J}(q)\ddot{q} + \dot{\bar{J}}(q)\dot{q}. \tag{25}$$

Let ξ be any solution to (22). ξ is guaranteed to exist since Im $\bar{J}(q) = R^m$. Take τ to be (21), then

$$M\ddot{q} + V = \tau = M\xi + V \Rightarrow M\ (\ddot{q} - \xi) = 0 \Rightarrow \ddot{q} = \xi,$$

which with (22) and (25) gives $\dot{\mathcal{V}} = u$. □

Comments:

1) Note that this result applies to all systems of the form (20), of which rigid link manipulators are a special case.

2) Note that with $y \in M^m$ and $\tau \in R^n$, the fact that we need $m \le n$ can be interpreted to mean that there must be least as many inputs as outputs.

3) When $\bar{J} = J = \partial h/\partial q$, $\mathcal{V} = \dot{y}$, and $m = n$ we have that $\tau = -MJ^{-1}\dot{J}\dot{q} + MJ^{-1}u + V \Rightarrow \ddot{y} = u$ when det $J \neq 0$. This is the same result provided by GCT, NDT, and OCS as seen in the last section.

4) Note that in the proof we force $\ddot{q} = \xi$ precisely like $\ddot{q} = u$ is forced to happen in the Computed Torque method. In fact, for $y = q$ we have $J = I$ and $\dot{J} = 0$ giving $\xi = u$. Thus the exact linearizing control of (21), (22) is seen to be a generalization of the Computed Torque method in a somewhat different, and perhaps more illuminating, way than OCS.

Let us consider the case of EF control given by the system (1), (4). Here $J = J_o$ where $\mathcal{V} = \begin{pmatrix} \dot{x} \\ \omega \end{pmatrix} = J_o\dot{q}$. In this case, when $m = n$, (23) is

$$\tau = -MJ_o^{-1}\dot{J}_o\dot{q} + MJ_o^{-1}u + V. \tag{26}$$

When $\det J_o \neq 0$, use of τ yields $\begin{pmatrix} \ddot{x} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$. This is precisely RAC [7], [8]. Theorem LEL can be interpreted as an extension of RAC to the redundant arm case which allows for the use of a minimal representation of EF attitude [24]. The more general case $m \leq n$ is given by

$$\tau = Mu + V, \; J_o\xi = -\dot{J}_o\dot{q} + u. \tag{27}$$

By using the indirect form (27), τ can be obtained, after ξ has been found, by use of the Newton-Euler recursion [45]. Furthermore ξ can be obtained recursively - either directly [46], or by first recursively obtaining J_o and $\dot{J}_o$ and then solving for ξ by Gaussian Elimination. The major point to be drawn here, is that (27) shows us how to perform exact external linearization without the need for an explicit manipulator model. After exactly linearizing to $\begin{pmatrix} \ddot{x} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ one can perform EF tracking at this stage [7][8], or one can continue to the form (11) by the use of (10).

When using (26) or (27), the only way that rank $J_o < m$ can occur for $m \leq n$ is when the manipulator is at a mechanically singular configuration. Recall (section 4) that in the case when a minimal representation of EF attitude is used, the resulting Jacobian matrix J will be rank deficient not just for a manipulator singularity, but at a configuration which leads to a singularity of attitude presentation. Thus rank deficiency of J_o is *kinematically cleaner to understand.* The necessity that rank $J_o = m$ in order to use (26) or (27) allows two obvious, but important statements to be made: i) For a manipulator with a workspace boundary (ignoring joint stops), as in the case of a PUMA-type manipulator, exact linearization at the boundary is impossible; ii) For a nonredundant (6 dof) manipulator with workspace interior singularities, there cannot be exact linearization throughout the workspace interior. For a redundant manipulator with workspace interior singularities, it may be possible to avoid workspace interior configurations which cannot be exactly linearized by the use of self motions as described in [48][49]. This is related to the multiplicity of solutions available for ξ in (27).

It is interesting to ask just how the control (23) fulfills the aim of GCT as stated in (12)-(14). We have the nonlinear feedback (taking $\mathcal{V} = \dot{y}$ and $J = \bar{J}$) $\tau = Q(Z) + B(Z)u = (V - MJ^{-1}\dot{J}\dot{q}) + (MJ^{-1})u$ which when applied to (12), (13) gives

$$\frac{d}{dt} \begin{pmatrix} q \\ \dot{q} \end{pmatrix} = \begin{pmatrix} 0 & I \\ 0 & -J^{-1}\dot{J} \end{pmatrix} \begin{pmatrix} q \\ \dot{q} \end{pmatrix} + \begin{pmatrix} 0 \\ J^{-1} \end{pmatrix} u. \tag{28}$$

Consider the local nonlinear change of basis given by

$$\begin{pmatrix} y \\ \dot{y} \end{pmatrix} = \begin{pmatrix} h(q) \\ J\dot{q} \end{pmatrix} \; ; \; \begin{pmatrix} q \\ \dot{q} \end{pmatrix} = \begin{pmatrix} h^{-1}(y) \\ J^{-1}\dot{y} \end{pmatrix} .$$

The fact that $\dot{y} = J\dot{q}$ and $\ddot{y} = J\ddot{q} + \dot{J}\dot{q}$ gives

$$\frac{d}{dt} \begin{pmatrix} y \\ \dot{y} \end{pmatrix} = \begin{pmatrix} J & 0 \\ \dot{J} & J \end{pmatrix} \frac{d}{dt} \begin{pmatrix} q \\ \dot{q} \end{pmatrix} .$$

Writing (28) as

$$\begin{pmatrix} J & 0 \\ \dot{J} & J \end{pmatrix} \frac{d}{dt} \begin{pmatrix} q \\ \dot{q} \end{pmatrix} = \begin{pmatrix} J & 0 \\ \dot{J} & J \end{pmatrix} \begin{pmatrix} 0 & I \\ 0 & -J^{-1}\dot{J} \end{pmatrix} \begin{pmatrix} h^{-1}(y) \\ J^{-1} \dot{y} \end{pmatrix} + \begin{pmatrix} J & 0 \\ \dot{J} & J \end{pmatrix} \begin{pmatrix} 0 \\ J^{-1} \end{pmatrix} u$$

we obtain the BCF

$$\frac{d}{dt} \begin{pmatrix} y \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y \\ \dot{y} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} u, \Leftrightarrow \ddot{y} = u.$$

Of course we are benefiting from the hindsight provided us by GCT [15]-[19].

7. Concluding Remarks

Recognizing the fundamental unity of RAC, GCT, OSC, and NDT [7]-[22] for exact linearization of manipulators, we can focus on their true differences - namely differences in implementation detail and design philosophy. With the awareness that they all produce essentially the same linearizing feedback, we can ask why this particular feedback form is appropriate for manipulator-like systems.

OCS and RAC exploit the specific structure of such systems. Not surprisingly, the solutions arrived at, reflecting the philosophies and implementation perspectives of the researchers involved, are quite distinct in their flavor and presentation. Yet, since the properties specific to manipulator dynamics ultimately forced the solution, they are fundamentally the same. (Actually, apparently only OCS worked with a perspective directed specifically towards decoupled EF motions, RAC is content to stop at a point just shy of the goal. It is also interesting that [12] apparently shows an awareness of the relationship between OCS and RAC, and the degree to which RAC can be said to decouple and linearize EF motions). The important point here is that researchers consciously exploited the specific properties of a system of interest, but without pin-pointing precisely what these properties were which made the system amenable to linearizing control.

GCT and NDT provide techniques for exactly linearizing general smooth affine-in-the-input dynamical systems. These techniques ignore any specific nongeneric structural properties that a system might have and as a consequence the solutions obtained are much less transparent than those of OCS or RAC. The strength of these approaches, particularly GCT, is that they can provide necessary and sufficient conditions for a system to be exactly linearizable and constructive sufficient conditions which produce the linearizing feedback when satisfied. These techniques can be applied to systems which defy our abilities to intuit or comprehend - such as manipulators coupled to complex electromechanical actuation devices. Interestingly, when applied to the problem of manipulator exact linearization the solutions obtained can be shown to be equivalent to those of RAC and OCS. Again the structural properties of the system forced the solution. Once a solution is known to exist, it is reasonable to attempt to produce it from more physical arguments knowing now that the search is not fruitless. This leads to a reexamination of OCS and RAC.

The work of [17]-[22] stresses a perspective which serves to enable a clearer comparison between competing techniques for external linearization: Place the system in a standard linear canonical form before additional control efforts are made - this ensures that the process of linearizing the system is not mixed up with, and confused with, the process of stabilizing and controlling it. This perspective greatly aided the comparison of GCT, OCS, RAC, and NDT which resulted in [24]. In turn, this comparison focuses attention on the structural properties of manipulators.

Much current research makes it apparent that systems dynamically similar to rigid link manipulators have important structural properties which can be exploited to achieve results which are quite strong when compared to those available for general smooth affine-in-the-inputs nonlinear systems [25]-[40]. Here we have seen that exploiting the nongeneric second order form of system (1) with an everywhere positive definite mass matrix and a C^2 locally onto readout map enables a simple form for the linearizing feedback.

8. Acknowledgments

This work was done at the Jet Propulsion Laboratory, California Institue of Technology, under contract with the National Aeronautics and Space Administration.

9. References

[1] Paul, R. C., "Modeling, Trajectory Calculation and Servoing of a Computer Controlled arm.", Stanford A. I. Lab., Stanford U., A. I. Memo 177, Nov. 1972.

[2] Markiewicz, B. R., "Analysis of the Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator." JPL TM 33-601, March 1973.

[3] Bejczy, A. K., "Robot Arm Dynamics and Control," JPL TM 33-669, Feb. 1974.

[4] Hemami, H. and Camana, P. C., "Nonlinear Feedback in Simple Locomotion Systems", IEEE Trans. Automatic Control, 1976, pp. 855-860.

[5] Raibert, M. H. and Horn, B. K., "Manipulator Control Using the Configuration Space Method", Industrial Robot, 5, pp. 69-73, 1978.

[6] Hewit, J. R. and Padovan, J., "Decoupled Feedback Control of Robot and Manipulator arms", Proc. 3rd CISM-IFToMM Symp. on Theory and Practice of Robot Manipulators, Udine, Italy, pp. 251-266, Sept. 1976.

[7] Paul, R., Luh, J. et al., "Advanced Industrial Robot Control Systems", Purdue U., TR-EE 78-25, May 1978.

[8] Luh, J., Walker, M. and Paul, R., "Resolved - Acceleration Control of Mechanical Manipulators", IEE Trans. Aut. Control, AC-25, pp. 468-474, 1980.

[9] Freund, E., "A Nonlinear Control Concept for Computer Controlled Manipulators", Proc IFAC Symp. Multivariable Technological Systems, pp. 395-403, 1977.

[10] Freund, E. and Syrbe, M., "Control of Industrial Robots by Means of Microprocessors", IRIA Conf. Lecture Notes on Information Sciences, Amsterdam, Springer-Verlag, pp. 167-185, 1976.

[11] Freund, E., "Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators", International J. Robotics Research, 3, pp. 76-86, 1982.

[12] Khatib, O., Commande dynamique dans l'espace operational des robots manipulators en presence d'obstacles, Engineering Doctoral Thesis, No. 37, ENSAE, Toulouse, 1980.

[13] Khatib, O., "Dynamic Control of Manipulators in Operational Space", 6th CISM-IFToMM, 1983.

[14] Khatib, O., "The Operational Space Formulation in the Analysis, Design, and Control of Robot Manipulators", 3rd Int'l Symp. Robotics Research, pp. 103-110, 1985.

[15] Isidori, A. and Ruberti, A., "On the Synthesis of Linear Input-output Responses for Nonlinear Systems", Systems and Control letters, 4, pp 17-22, 1984.

[16] Isidori, A., "The Matching of a Prescribed Linear Input-Output Behavior in a Nonlinear System", IEEE Trans. Aut. Control, Vol. AC-30, pp. 258-265, 1985.

[17] Cheng, D., Tarn, T. J., and Isidori, A., "Global External Linearization of Nonlinear Systems Via Feedback", IEEE Trans. Automatic Control.

[18] Isidori, A., Nonlinear Control Systems: An Introduction, Springer-Verlag, 1985.

[19] Chen, Y., Nonlinear Feedback and Computer Control of Robot Arms, D. Sc. Dissertation, Dept. Systems Science and Math, Washington U., St. Louis MO, 1984.

[20] Tarn, T. J., Bejczy, A. K. et al., "Nonlinear Feedback in Robot Arm Control", Proc. 23rd CDC, pp. 736-751, 1984.

[21] Bejczy, A. K., Tarn, T. J., and Chen, Y. L., "Robot Arm Dynamic Control by Computer, 1985 IEEE ICRA, pp. 960-970.

[22] Tarn, T. J., Bejczy, A. K., and Yun, X., "Coordinated Control of Two Robot Arms", 1986 IEEE ICRA, pp. 1193-1202.

[23] Brady, M. et al. Robot Motion Planning and Control, MIT press, 1984.

[24] Kreutz, K., "On Nonlinear Control for Decoupled Exact External Linearization of Robot Manipulators", in Recent Trends in Robotics: Modeling, Control, and Education, Ed. by M. Jamshidi et al. pp. 199-212, North-Holland, 1986.

[25] Meirovitch, L., Methods of Analytical Dynamics, McGraw-Hill, New York 1970.

[26] Gantmacher, F., Lectures in Analytical Mechanics, MIR publishers, Moscow, 1975.

[27] Koditschek, D., "Natural Control of Robot Arms", Center for Systems Science Report, Dept. of EE, Yale U., 1985.

[28] Koditschek, D., "High Gain Feedback and Telerobotic Tracking", This Workshop.

[29] Asada, H. and Slotine, J., Robot Analysis and Control, Wiley, New York, 1986.

[30] Pringle, R. Jr., "On the Stability of a Body with Connected Moving Parts", AIAA Journal, 4, pp. 1395-1404, 1966.

[31] Takegaki, M. and Arimoto, S., "A New Feedback Method for Dynamic Control of Manipulators", J. of Dyn. Sys. Meas. Control, 102, pp. 119-125, 1981.

[32] Arimoto, S. and Miyazaki, F., "Stability and Robotness of PID Feedback Control for Robot Manipulators of Sensory Capacity", First Int'l. Symp. Robotics Research, pp. 783-799, 1983.

[33] Arimoto, S. and Miyazaki, F., "Stability and Robustness of PD Feedback Control With Gravity Compensation for Robot Manipulators", Robotics: Theory and Practice, DSC-Vol. 3, pp. 67-72, 1986.

[34] Wen, J. T. and Bayard, D.S., "Simple Robust Control Laws for Robotic Manipulators - Part I: Nonadaptive Case", This Workshop.

[35] Bayard, D. S. and Wen, J. T., "Simple Robust Control Laws for Robotic Manipulators - Part II: Adaptive Case", This Workshop.

[36] Slotine, J. and Li, W., "On the Adaptive Control of Robot Manipulators", Robotics: Theory and Practice, DSC-Vol. 3, pp. 51-56, 1986.

[37] Paden, B., and Slotine, D., "PD + Robot Controllers: Tracking and Adoptive Control", 1987 IEEE Int'l Conf. Robotics and Automation Raleigh, 1987 (to be presented).

[38] Sontag, E. and Sussman, H., "Time-Optimal Control of Manipulators," IEEE 1986 Int'l. Conf. Robotics and Automation, San Francisco, 1986, pp. 1692-1697.

[39] Sontag, E. and Sussman, H., "Remarks on the Time-Optimal Control of Two-link Manipulators", Proc. 24th IEEE CDC, pp. 1643-1652, 1985.

[40] Wen, J., "On Minimum Time Control for Robotic Manipulators", in Recent Trends in Robotics: Modeling, Control, and Education, Ed. by M. Jamshidi et al., pp. 283-292, North-Holland, 1986.

[41] Stuelpnagel, J., "On the Parameterization of the Three Dimensional Rotation Group", SIAM Review, 6, 422-430, 1964.

[42] Hughes, P.C., Spacecraft Attitude Dynamics, Wiley, New York, 1986.

[43] Craig, J., Introduction to Robotics, Addison-Wesley, 1986.

[44] Boothby, W., An Introduction to Differentiable Manifolds and Riemannian Geometry, 2nd Edition, Academic Press, Orlando, 1986.

[45] Luh, J. Y. S., Walker, M. W., and Paul R. P., "On-line Computational Scheme for Mechanical Manipulators", J. Dynamical Systems, Measurement, and Control, 102, 1980, pp. 69-76.

[46] Hollerbach, J. M. and Sahar, G., "Wrist-Partitioned Inverse Kinematic Accelerations and Manipulator Dynamics", Intl'l. J. Robotics Research, 2, pp. 61-76, 1983.

[47] Von Westenholz, C., Differential Forms in Mathematical Physics, North-Holland, Amsterdam, 1986.

[48] Hollerbach, J., "Optimum Kinematic Design for a Seven Degree of Freedom Manipulator", 2nd Int'l Symp. Robotics Research, 1984.

[49] Baillieul, J. et al., "Programming and Control of Kinematically Redundant Manipulators", PROC. 23rd CDC, pp. 768-774, 1986.

S33-37
153201
10P.

A Virtual Manipulator Model for Space Robotic Systems

S. Dubowsky and Z. Vafa
Massachusetts Institute of Technology
Cambridge, MA 02139

MJ760862
adm

1. A~~bstract~~

Future robotic manipulators carried by a spacecraft will be required to perform complex tasks in space, like repairing satellites. Such applications of robotic manipulators will encounter a number of kinematic, dynamic and control problems due to the dynamic coupling between the manipulators and the spacecraft. ~~This paper presents~~ a new analytical modeling method for studying the kinematics and dynamics of manipulators in space. is presented. The problem is treated by introducing the concept of a Virtual Manipulator (VM). The kinematic and dynamic motions of the manipulator, vehicle and payload, can be described relatively easily in terms of the Virtual Manipulator movements, which have a fixed base in inertial space at a point called a Virtual Ground. It is anticipated that the approach described ~~in this paper~~ here will aid in the design and development of future space manipulator systems.

2. Introduction

Robotic manipulators are potentially very useful for performing complex tasks in non-industrial hostile environments [1,2], such as in space. A number of studies have considered the potential applications of manipulators in space and the capabilities that these systems must have to achieve anticipated mission goals [3-5]. These applications include tasks such as repairing, servicing and constructing space stations in orbit. Currently, these tasks can only be performed by astronaut Extra Vehicular Activity (EVA). Eliminating the need for EVA would obviously reduce hazards to the astronauts and mission costs.

Unfortunately, the use of manipulators in space is complicated by the manipulator/spacecraft dynamic coupling. For example, movements of a manipulator will disturb the attitude of the spacecraft carrying it. This coupling will adversely affect the manipulator's precision, and reduce the on orbit life of the system by consuming excessive attitude control fuel. Also, any motions of the spacecraft, say due to the firing of attitute control jets, will disturb the manipulator. Therefore, new manipulator concepts, designs and control techniques will be required to minimize and compensate for the manipulator/spacecraft dynamic coupling.

Researchers working on the control of space manipulators have focused their attention on issues such as sensor reqirements, path planning algorithms, teleoperator control [6-8]; the problems of vehicle/manipulator dynamic interactions remain unresolved.

This paper presents a new and effective analytical modeling method for studying the kinematics and dynamics of manipulators in space. The problem is treated by introducing the concepts of a Virtual Ground (VG) and Virtual Manipulator (VM). As discussed below the VG is located at the center of mass of the manipulator/spacecraft system, and the VM is an ideal kinematic chain connecting the VG to any point on the real manipulator. Motions of a system, including a vehicle, manipulator and payload can be described easily by the VM. This model has proven to be effective in calculating the kinematic and dynamic properties of the system; such as its inverse kinematic solution and workspaces. This paper shows that the VM approach can also be used to plan the manipulator's motions in order to minimize the degrading consequences of the manipulator/spacecraft dynamic interactions.

3. A Model of Manipulators in Space

Future space manipulator systems will have one or more mechanical arms carried by a vehicle, as shown in Figure 1. The vehicle will be capable of motion in six degrees of freedom, and will have reaction jets for position and attitude control. Although manipulators could be driven by photovoltaicly powered electric actuators, which use no where

$$M_{tot} = \sum_{q=1}^{N} M_q \qquad (2)$$

Since there are no external forces, the VG is stationary in the frame N and the vector $\mathbf{V}_g$ is always constant.

In the following development the VM properties such as link dimensions and joint axes, for initial manipulator configuration are described. Then the rules for its joint movements as a function of the real manipulator joint movements are presented. Referring to Figure 3, which shows the end effector VM for the manipulator shown in Figure 2, the i^{th} link of the Virtual Manipulator is defined by the vector $\mathbf{V}_i$,

$$\begin{aligned} \mathbf{V}_1 &= \mathbf{D}_1 \\ \mathbf{V}_2 &= \mathbf{H}_1 + \mathbf{D}_2 \\ &\vdots \\ \mathbf{V}_N &= \mathbf{H}_{N-1} + \mathbf{D}_N \end{aligned} \qquad (3)$$

where

$$\mathbf{D}_i = \mathbf{R}_i \sum_{q=1}^{i} M_q/M_{tot} \qquad (i = 1, 2, ..., N) \qquad (4)$$

and

$$\mathbf{H}_i = \mathbf{L}_i \sum_{q=1}^{i} M_q/M_{tot} \qquad (i = 1, 2, ..., N-1) \qquad (5)$$

The first VM link represents the vehicle's orientation. This link is attached to the VG by a spherical joint and its motion is equal to the three vehicle rotations with respect to inertial space. The end of the Virtual Manipulator terminates at the end effector, defined by a vector $\mathbf{E}$, fixed in the N^{th} VM link.

The i^{th} VM joint is taken as a revolute or a prismatic joint depending upon whether the i^{th} joint of the real manipulator is revolute or prismatic. The axis of rotation for a revolute VM joint, j_i, is parallel to the axis of the real manipulator joint A_i. Similarly, the translational axes of prismatic VM joints are parallel to the corresponding axes of the real manipulator prismatic joints. Equations (1) through (5) define the VM and its position corresponding to the initial position of the system, as shown in Figure 2. The VM links will all be parallel to the real manipulator links in cases where all the centers of mass for all manipulator links lie on a line connecting the manipulator joints on the corresponding link.

The VM will move as the joints of the real manipulator move. The angular rotations of the VM revolute joints, from their initial position, are equal to the angular rotations of the corresponding revolute joints for the real manipulator. The prismatic virtual joint translations are ratios of the corresponding real prismatic joint translations. For an end effector VM, translation of the virtual joint, $\mathbf{P}_j$, is given by:

$$\mathbf{P}_j = \mathbf{T}_j \sum_{q=1}^{j} M_q/M_{tot} \qquad (6)$$

For the VM in its position of construction, its initial position, the values of $\mathbf{T}_j$ are taken as zero. Hence the initial magnitudes of $\mathbf{P}_j$ are zero. The prismatic joint motions, $\mathbf{T}_j$, are referenced to the initial position.

If a VM that is constructed according to Equations (1) through (5), moves with the real manipulator according to the above description, and its link shapes and lengths remain constant as the manipulator moves, then it can be shown (see Appendix A) that:

1 The axis of the i^{th} virtual joint is always parallel to the i^{th} axis of the real system joint, and

2 The Virtual Manipulator end point will always coincide with the real manipulator's end effector.

These properties enable the kinematic and dynamic motions of a free-floating manipulator system to be described by the motions of a much simpler Virtual Manipulator which has a fixed base in inertial space. The properties of the VM remain the same as long as the mass property of the system does not change. For example, when the manipulator grasps a free-floating pay load, the VM changes. According to Equations (1) through (5), the VM link lengths will be reduced for the addition of a payload. Virtual Manipulators constructed for points other than the end effector have different links than the links defined in Equation (1); and their joint movements maybe different than the ones described above, for example, prismatic joint translations may be the vector $(\mathbf{P}_i - \mathbf{T}_i)$, depending upon location of the point used to construct the VM [9].

reaction fuel, manipulator motions could disturb a vehicle's position and attitude and result in the consumption of excessive amounts of attitude fuel. The useful !ife of spacecraft systems is often limited by the amount of reaction jet fuel they can carry.

Two approaches to solve this problem are: 1) permit the vehicle to move and compensate for the base motions in the manipulator task planning; and 2) plan the manipulator motions so that they do not cause the vehicle to move excessively. The first of these approaches requires the ability to perform inverse kinematic and workspace calculations for a free-floating system [10]. The second requires methods for planning manipulator motions that would self-correct the vehicle's orientation with little or no reaction jet adjustments. These approaches and associated issues are addressed here through the Virtual Manipulator technique. Assumed in this work is that the external forces/torques acting on the system are negligible, and that the system is free floating. Also assumed is that the system elements may be modeled as rigid bodies. The later assumption may not be valid if a manipulator must perform high speed motions.

4. Analytical Development of the Virtual Manipulator

The Virtual Manipulator (VM) is a massless kinematic chain terminating at an arbitrary point on the real manipulator. Its base is the Virtual Ground (VG), which is an imaginary fixed point in inertial space. It is proven below that for a given system the properties of the VM and location of the VG are fixed. VMs exist for many different manipulator structures, such as open or closed chains, single or many branch arms, revolute or prismatic joints [9-11]. The discussion in this paper will be limited to manipulators composed of spatial serial chains with revolute or prismatic connections. Although VMs exist for any point on the real manipulator, this paper deals with VMs whose end points coincide with the real manipulator end effector.

The VG is defined to be the center of mass of the manipulator system. From elementry mechanics, when there are no external forces, such as from reaction jets, the VG will be fixed in an inertial space. It will not move due to any internal forces of the system such as joint torques, or due to any manipulator motions.

Figure 2 shows a schematic drawing of an N body spatial manipulator system. The first body in the chain represents the vehicle which carries the manipulator. The N^{th} body is a combination of the payload and the last link. The i^{th} joint is called J_i, and C_i is the center of mass of the i^{th} body. The vectors $\mathbf{R}_i$ and $\mathbf{L}_i$ connect C_i to J_i and J_i to C_{i+1}, respectively. The vector $\mathbf{R}_N$ connects C_N to the end effector. The vectors $\mathbf{R}_i$ and $\mathbf{L}_{i-1}$ are fixed relative to the i^{th} link, and hence the angle between these vectors is constant for all system configurations. If the i^{th} manipulator joint is a revolute joint, the vector defining the axis of rotation of J_i is called $\mathbf{A}_i$, and the angle θ_i is the rotation of the i^{th} joint. If the i^{th} manipulator joint is a prismatic joint, the vector $\mathbf{T}_i$ is defined to be the translation along the translational axis. If the i^{th} joint is revolute, then the magnitude T_i is equal to zero.

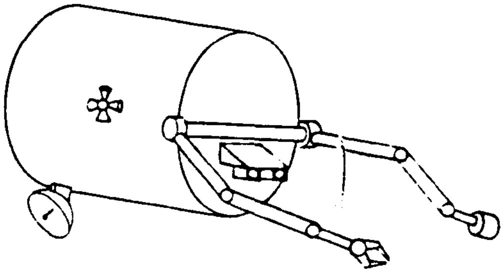


Figure 1: A Space Manipulator System.

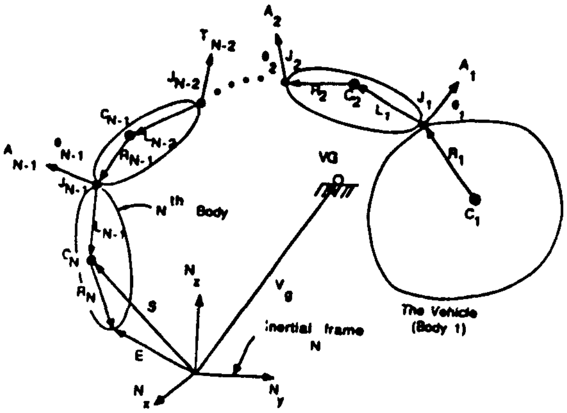


Figure 2: N Body System in Space.

The location of the VG for this system in inertial space, the center of mass of the system, can be found by knowing some initial position of the system. The vector S(0) defines the initial known location of the end effector with respect to an inertial reference frame N. Then the location of the VG, the vector $\mathbf{V}_g$, can be obtained from conservation of linear momentum by:

$$\mathbf{V}_g = \sum_{i=1}^{N}[\mathbf{S}(0) - \sum_{j=1}^{i-1}(\mathbf{R}_j + \mathbf{L}_j + \mathbf{T}_j)]M_i/M_{tot} \quad (1)$$

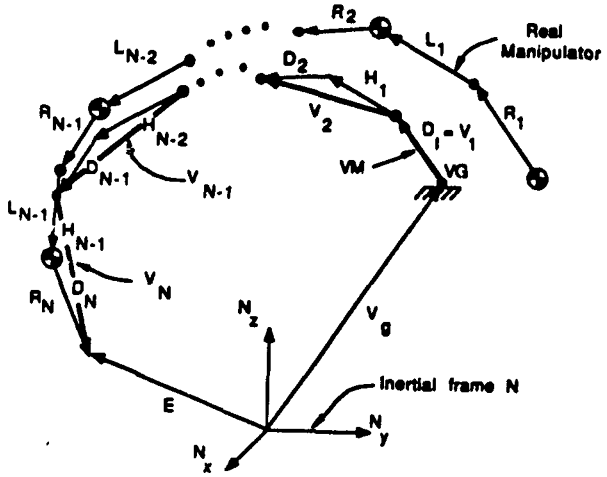


Figure 3: N Body System and its VM.

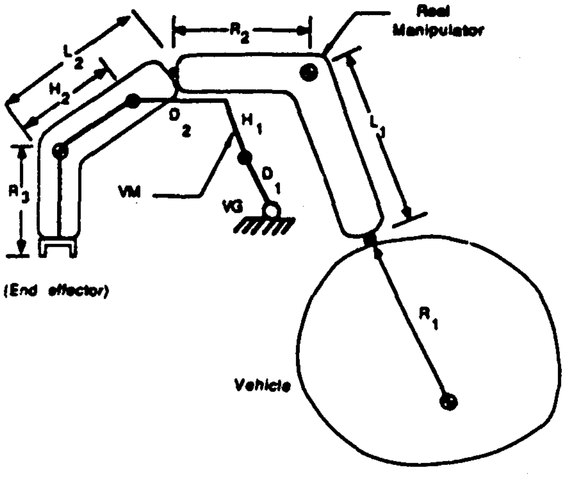


Figure 4: A Three Body Planar System and its VM.

Table 1 gives the properties of a very simple planar manipulator and its Virtual Manipulator, shown in Figure 4. It should be remembered that the method is not restricted to planar systems.

5. Applications of Virtual Manipulators

The Virtual Manipulator approach has a number of possible applications. VMs can be used to simplify the inverse kinematics of space manipulators, calculate their workspaces, plan their motions and formulate the equations of motion [9-11]. It should be noted that using conventional methods, these problems are far more difficult for space manipulators than for industrial manipulators with fixed bases. In the sections below, the use of the VM is shown for workspace analysis and path planning.

A. Workspace Analysis

Since the vehicle and manipulator dynamics are coupled, the manipulator's motions will cause the vehicle to move and this in turn makes it difficult to find the manipulator workspace. In fact several different types of workspaces exist. In this section, a workspace called the constrained workspace, for a manipulator in space is defined, for a more complete discussion of space manipulator workspaces refer to references [9,10]. For the constrained workspace it is assumed that the attitude, but not the location, of the vehicle is controlled. This can be achieved without the use of attitude control fuel by employing reaction wheels, or by using the self correcting maneuvers discussed later in this paper.

To find the constrained workspace, a Virtual Manipulator is constructed to the end effector of the real manipulator. The joint limits of the real manipulator are transformed into VM joint limits. The workspace of the Virtual Manipulator is then found using conventional workspace analysis methods [12]. The real manipulator workspace will be equal to the VM workspace because of the following reasons. The VM end point coincides with the real manipulator end effector, and it is assumed that it is possible to control the orientation of the first VM link, representing the vehicle, with respect to inertial space. The other joints are controlled with their actuators. This workspace will always be a spherical shell, assuming there are no limits on the vehicle orientations. Figure 5 shows the constrained workspace for the simple two link manipulator shown in Figure 4, it was found using its VM.

B. Path Planning

In certain cases, the magnitude of the rotations of the vehicle caused by the manipulator's motion may not be acceptable. For example, vehicle rotations may cause communication devices to loose their signals. Vehicle rotations can be controlled using reaction wheels or reaction jets. However, these devices have the disadvantages of increased mechanical complexity and system weight or increased consumption of attitude control fuel.

It is shown below that the manipulator itself can be moved in such a way as to have the end effector follow a nominal specified path and yet have a prescribed vehicle orientation, with specified limits, without using attitude control fuel or requiring reaction wheels.

Table 1: Characteristics of Planar Manipulator and its VM.

Body no	M (Kg)	R (m)	L (m)	D (m)	H (m)
1	50	1.0	1.0	0.33	0.33
2	50	0.75	0.75	0.5	0.5
3	50	0.5	-	0.5	-

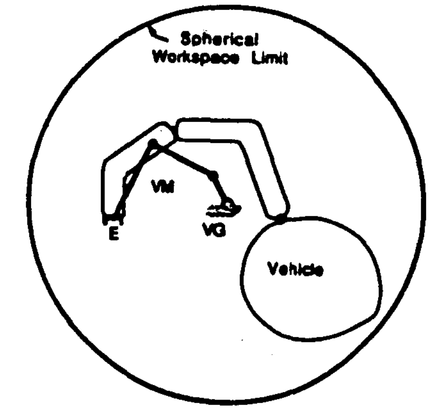


Figure 5: Constrained Workspace of Manipulator Shown in Figure 4.

To find this motion the principle of conservation of angular momentum is applied to the system. For an n degrees of freedom space manipulator the following equation can be writter.

$$\dot{\mathbf{X}} = \mathbf{F}(\mathbf{\Theta},\mathbf{X})\dot{\mathbf{\Theta}}, \qquad (7)$$

$\mathbf{F}$: 3 by n matrix with elements $F_{i,j}$,

$\mathbf{X}$: 3 by 1 vector of vehicle inertial orientations with elements, X_i,

$\mathbf{\Theta}$: n by 1 vector of joint angles with elements θ_i

In general, Equation (7) is non-integratable, that is:

$$\frac{\partial^2 X_i}{\partial\theta_k\partial\theta_j} \neq \frac{\partial^2 X_i}{\partial\theta_j\partial\theta_k} \qquad (8)$$

Therefore, the final vehicle orientation depends on path taken by the manipulator from one position to another. It follows that the final vehicle orientation will change if the manipulator moves along one path in joint space and returns to its initial position by another path. This is a similar notion to the one which permits astronauts to reorient their bodies by moving their limbs [13]. This leads to a strategy for adjusting or correcting motions of the vehicle's orientation. In this strategy nominal trajectories are selected for the end effector and vehicle orientation. Then the joint motions are executed assuming the vehicle follows its trajectory. If at any point the vehicle orientation deviates from its desired path by more than a specific amount, a series of small cyclic motions, selected to correct for the vehicle orientation are added to the joint motions.

To find the cyclical joint motions that achieve the desired vehicle/base orientation corrections, it is assumed that these motions are small enough that the end effector deviates only by a small amount from its nominal trajectory. This small motion assumption permits the use of a nonlinear system model in which nonlinearities of order greater than 2 can be neglected.

First, let $\mathbf{X}$ be a set of Euler angles defining the base orientation with respect to an inertial coordinate frame. The initial and desired final base orientations are $\mathbf{X}_i$ and $\mathbf{X}_d$, respectively. The desired change in the Euler angles is defined by

$$\delta\mathbf{X} = \mathbf{X}_i - \mathbf{X}_d \qquad (9)$$

Let Θ_0 be the vector defining the initial and final joint positions at the beginning and end of the correction maneuver. Also let the vectors $\delta\mathbf{V}$ and $\delta\mathbf{W}$ define small joint movements. The closed correction path is constructed by having the manipulator move along the straight lines, in joint space defined by vectors $\delta\mathbf{V}$ and $\delta\mathbf{W}$ shown in Figure 6.

For small $\delta\mathbf{V}$ and $\delta\mathbf{W}$ the following equation can be obtained from Equation (7).

$$\delta X_k = \sum_{i=1}^{3}\sum_{j=1}^{3}[\sum_{m=1}^{3}(\frac{\partial F_{kj}}{\partial X_m}F_{mi} - \frac{\partial F_{ki}}{\partial X_m}F_{mj}) + \frac{\partial F_{kj}}{\partial\theta_i} - \frac{\partial F_{ki}}{\partial\theta_j}]\delta W_j\delta V_i \qquad (k = 1,2,3) \qquad (10)$$

where δX_i, δV_i and δW_i are elements of the vectors $\delta \mathbf{X}$, $\delta \mathbf{V}$ and $\delta \mathbf{W}$, respectively. In the case of a three DOF spatial manipulator, Equation (10) will yield three equations with six unknowns. Three additional constrain equations are required to solve for $\delta \mathbf{V}$ and $\delta \mathbf{W}$.

If vectors $\delta \mathbf{V}$ and $\delta \mathbf{W}$ are parallel, the cyclic motion will not produce any vehicle rotation. Therefore it is assumed that these vectors are perpendicular:

$$\delta \mathbf{V}^T \cdot \delta \mathbf{W} = 0. \tag{11}$$

Further, the magnitudes of $\delta \mathbf{W}$ and $\delta \mathbf{V}$ are assumed to be equal:

$$\delta \mathbf{V}^T \cdot \delta \mathbf{V} = \delta \mathbf{W}^T \cdot \delta \mathbf{W}, \tag{12}$$

and one of the elements of $\delta \mathbf{V}$ is chosen to be a linear combination of the other two. For example,

$$\delta V_3 = (\delta V_2 + \delta V_1)/2 \tag{13}$$

Equations (10) through (13) yield six scalar equations with six scalar unknowns, which can be solved for the desired joint trajectories, $\delta \mathbf{V}$ and $\delta \mathbf{W}$. If the required correction, $\delta \mathbf{X}$, is large, the values of $\delta \mathbf{V}$ and $\delta \mathbf{W}$ may violate the small joint motion assumption. In this case the desired correction can be achieved by a series of m cyclical correction maneuvers. It is shown below that at each cycle Equations (10) through (13) do not have to be resolved and the final position can be achieved.

Referring to Figure 7, $\mathbf{T}(\mathbf{X}_j)$ is a 3 by 3 matrix which transforms a vector expressed in vehicle body coordinates (x,y,z) into inertial or Newtonian coordinates (N_x,N_y,N_z), when the body is at jth orientation. The transformation matrix for the initial vehicle orientation is $\mathbf{T}(\mathbf{X}_i)$. The transformation matrix for the desired vehicle position to be achieved after m cycles is $\mathbf{T}(\mathbf{X}_d)$. After one correction cycle, the transformation matrix is $\mathbf{T}(\mathbf{X}_i + \delta \mathbf{x})$, where,

$$\mathbf{T}(\mathbf{X}_i + \delta \mathbf{x}) = \mathbf{T}(\mathbf{X}_i)\mathbf{A}, \tag{14}$$

and the matrix $\mathbf{A}$ is the transformation matrix from the vehicle position, one cycle from the initial vehicle position, back to the initial position. The $\mathbf{A}$ matrix will not change with each cycle because the total system, vehicle and manipulator, have been subject only to a rigid body rotation in inertial space. Hence after m cycles the transformation matrix from the desired system position to inertial coordinates is simply:

$$\mathbf{T}(\mathbf{X}_d) = \mathbf{T}(\mathbf{X}_i)\mathbf{A}^m \tag{15}$$

Equation (15) can be solved for $\mathbf{A}$:

$$\mathbf{A} = \mathbf{P}\Lambda^{1/m}\mathbf{P}^{-1} \tag{16}$$

where Λ is a diagonal matrix of the eigenvalues of $\mathbf{T}(\mathbf{X}_i)^{-1}\mathbf{T}(\mathbf{X}_d)$ and $\mathbf{P}$ is a matrix of corresponding eigenvectors.

Using the $\mathbf{A}$ matrix obtained from Equation (16), the change in Euler angles ($\delta \mathbf{x}$) are calculated from Equation (14). Then the joint correction motions for each cycle, $\delta \mathbf{V}$ and $\delta \mathbf{W}$, are obtained by solving Equations (10) through (13). The manipulator should go through the derived joint transformations ($\delta \mathbf{V}$, $\delta \mathbf{W}$) m times to approach the desired vehicle orientation. However, the final vehicle orientation after m cycles will usually be slightly different than the desired orientation because of the neglected higher order nonlinearities. In order to achieve the desired vehicle orientation more precisely, the over all correction may need to be broken into several smaller corrections and the process repeated with a slightly different set of $\delta \mathbf{V}$ and $\delta \mathbf{W}$ for each subcorrection.

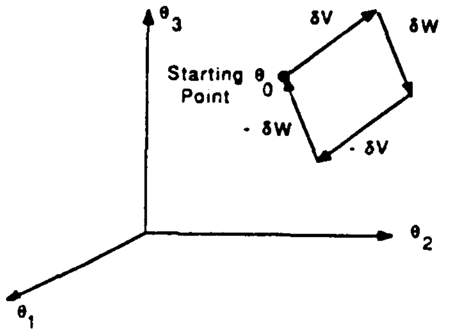


Figure 6: A Closed Path Correction in Joint Space for a Vehicle Rotation.

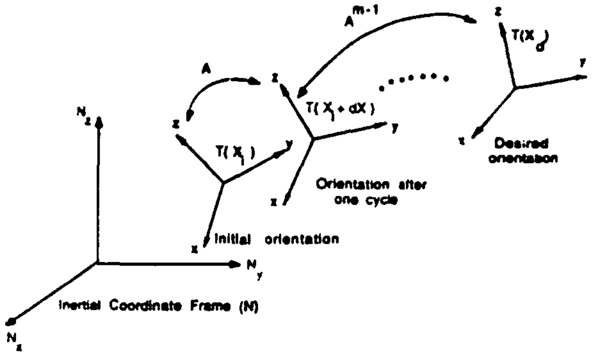


Figure 7: Vehicle Coordinate Rotation Due to Cyclic Manipulator Motion.

The above technique is now demonstrated for a spatial 3 DOF space manipulator shown in Figure 8. The properties of the manipulator are given in Table 2. It is desired to rotate the vehicle from its initial orientation to its final orientation as shown in Table 3. In this example, it was necessary to solve for the joint trajectories (δV and δW) 3 times to precisely obtain the desired vehicle orientation. The joint trajectories for these 3 cycle sets are shown in Figures 9 through 11. Each cycle is repeated 30 times to achieve the desired vehicle orientation. Table 3 shows the system angles after each cycle set. During each cycle the vehicle oscillates in sympathy to the manipulator's motion, see Figure 12. However the mean orientation of the vehicle changes continuously and reaches X_d at the same time the joints return to their initial positions. Figure 13 shows the mean vehicle orientation during the joint cycles. The vehicle movements during the joint motions are ± 0.1 radians from their mean trajectory. Here one can clearly see change in the base orientation as the manipulator joints cycle through their motion.

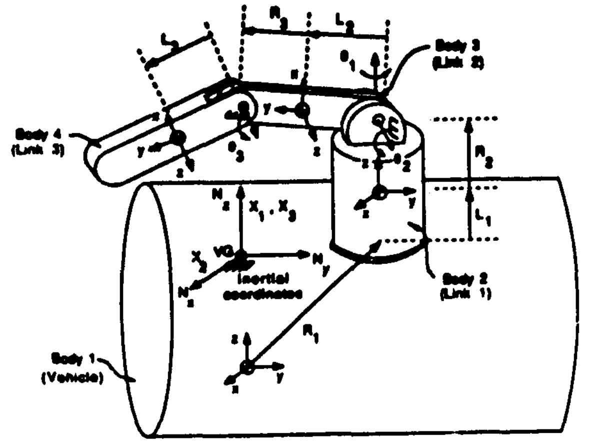


Figure 8: Spatial 3 DOF Space Manipulator.

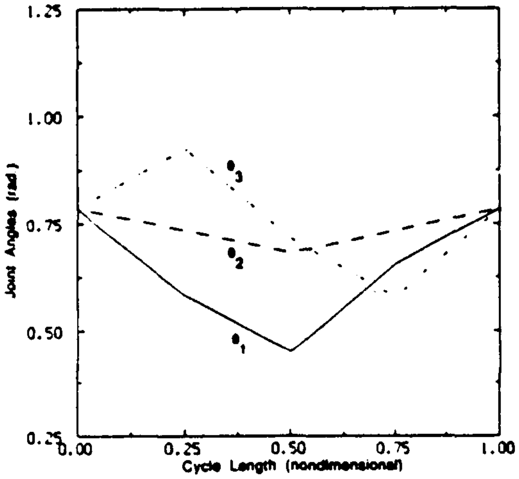


Figure 10: Joint Angle Trajectories for Second Set of Cycles.

Table 3: Manipulator Angles at Different Instances.

Angles (deg)	Initial	desired	After one cycle set	After second cycle set	Final position
X_1	50.	45.	44.7	45.0	45.0
X_2	40.	45.	43.9	45.0	45.0
X_3	35.	35.	37.1	34.8	35.0
θ_1	45.	45.	45.	45.	45.0
θ_2	45	45.	45.	45.	45.0
θ_3	45.	45.	45.	45.	45.0

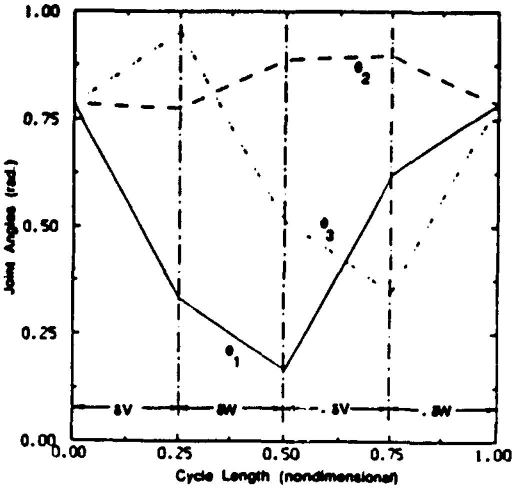


Figure 9: Joint Angle Trajectories for First Set of Cycles.

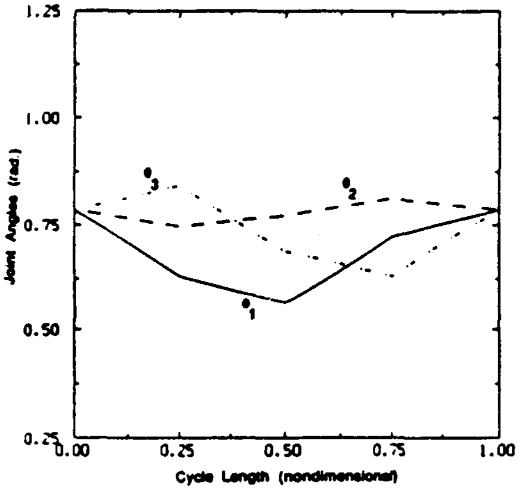


Figure 11: Joint Angle Trajectories for third Set of Cycles.

Table 2: Three DOF Manipulator Characteristics.

Body no.	Mass (kg)	R in local coord. (m)	L in local coord. (m)	Inertia about prin. axs (Kg-m ^{2})
1	20.	-1 i +1 j +2k	0.1i + 0.5j	0.5 i , 0.5 j . 0.5 k
2	7.	0.5i	0.5i	0.5 i , 0.5 j . 0.5 k
3	7.	0.5i +0.1k	0.5i +0.1k	0.5 i , 0.1 j , 0.5 k
4	5.	-	-	0.5 i , 0.1 j . 0.5 k

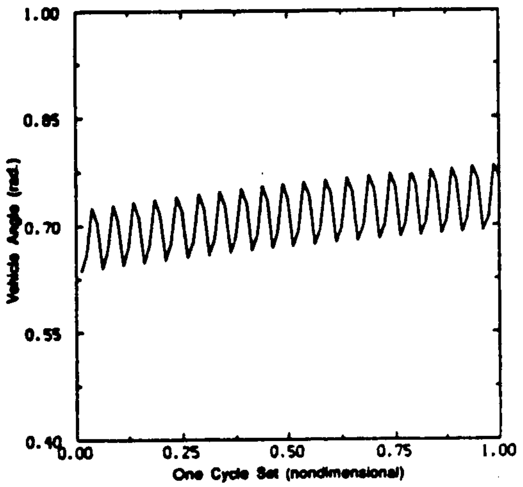


Figure 12: The X_2 Vehicle Coordinate for the First Set of Cycles.

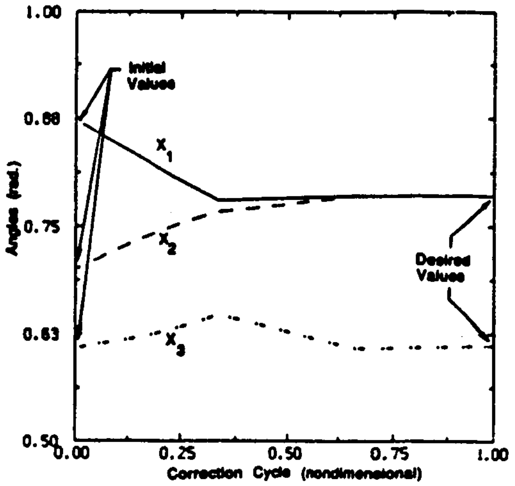


Figure 13: Mean Vehicle Euler Angles During Correction Cycle.

6. Summary and Conclusion

In this paper, the concepts of Virtual Manipulators and Virtual Grounds are discussed. The end effector VM characteristics and proof of its properties for serial link with revolute and prismatic joints were presented, and some of its applications were discussed. This is a new concept and further research is required to demonstrate its full capabilities.

7. Acknowledgement

The support of this research by the Automation Branch of NASA Langley Research Center under Grant NAG-1-489 is acknowledged.

8. References

[1] Khatib, O., "Real-time Obstacle Avoidance for Manipulators and Mobile Robots," Proceedings of the 1985 IEEE Inter. Conf. on Robotics and Automation, St. Louis, MO, March, 1985.

[2] Sheriden, T.B., "Human Supervisory Control of Robot Systems," Proceedings of the 1986 IEEE Inter. Conf. on Robotics and Automation, San Francisco, CA, April, 1986.

[3] Akin, D.L., Minsky, M.L., Thiel, E.D., and Kurtzman, C.R., "Space Applications of Automation, Robotics and Machine Intelligence Systems (ARAMIS)," MIT report, NASA Contract NAS8-34381, Cambridge, MA, October 1983.

[4] Meintel, A.J., and Schappell, R.T., "Remote Orbital Servicing System Concept," Presented at the Satellite Services Workshop, NASA Johnson Space Center, TX, June 22-24, 1982.

[5] Bronez, M.A., Clarke, M.M., and Quinn, A., " Requirements Development for a Free-Flying Robot - The 'ROBIN'," Proceedings of the 1986 IEEE Inter. Conf. on Robotics and Automation, San Francisco, CA, April, 1986.

[6] Lee, S., Bekey, G., and Bejczy, A.K., "Computer Control of Space-Borne Teleoperators with sensory feedback," Proceedings of the 1985 IEEE Inter. Conf. on Robotics and Automation, St. Louis, MO, April, 1985.

[7] Kohn, W., and Healey, K., "Trajectory Planner for an autonomous free-flying robot," Proceedings of the 1986 IEEE Inter. Conf. on Robotics and Automation, San Francisco, CA, April, 1986.

[8] French, R., Boyce, B., "Satellite Servicing by Teleoperators," Journal of Engineering for Industry, Vol. 107, pp. 49-54, February 1985.

[9] Vafa, Z., "Dynamics of Manipulators in Space", Ph.D. Thesis, Department of Mechanical Engineering, MIT, Cambridge, MA, 1987.

[10] Vafa, Z., and Dubowsky, S., "On the Dynamics of Manipulators in Space Using the Virtual Manipulator Approach," To appear in the Proceedings of the 1987 IEEE Inter. Conf. on Robotics and Automation, Raleigh, NC, March, 1987.

[11] Vafa, Z., Dubowsky, S., "Kinematic and Dynamic Models of Manipulators for use in Space Environments: The Concept of the Virtual Manipulator," To appear in The Proceedings of Seventh World Congress on Theory of Machines and Mechanisms, Sevilla, Spain, 1987.

[12] Yang, D.C.H., Lee, W.L., "Heuristic Combinatorial Optimization in the Design of Manipulator Workspace," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-14, No. 4, July/August 1984.

[13] Kane, T.R., Headrick, M.R., Yatteau, J.D., "Experimental Investigation of an Astronaut Maneuvering Scheme," Journal of Biomechanics, 1972, Vol. 5, pp. 313-320.

Appendix A: Proof of Virtual Manipulator Properties

First it will be proven that for a VM constructed using the rules presented in section 4 of this paper the VM end point will coincide with the end effector. Then it will be proven that when the manipulator goes through a movement the VM joint motions described in section 4 will keep the VM end point on the end effector.

First, recognizing that the system center of mass is stationary in the inertial frame N, $\mathbf{V}_g$ remains stationary in this frame and referring to Figure 2 yields:

$$M_{tot}\mathbf{V}_g = M_N\mathbf{S} + M_{N-1}[\mathbf{S} - \mathbf{L}_{N-1} - \mathbf{T}_{N-1} - \mathbf{R}_{N-1}] + \ldots + M_1[\mathbf{S} - \sum_{i=1}^{N-1}(\mathbf{L}_i + \mathbf{T}_i + \mathbf{R}_i)] \qquad \text{(A1)}$$

Recall that if the i^{th} joint is revolute $\mathbf{T}_i = 0$, otherwise, its magnitude is equal to the prismatic joint translations from the initial manipulator configuration and its direction is along the translational axis. Equation (A1) can be solved for $\mathbf{S}(t)$ as follows,

$$\mathbf{S}(t) = \mathbf{V}_g + \frac{M_1}{M_{tot}}(\mathbf{R}_1 + \mathbf{T}_1 + \mathbf{L}_1) + \ldots + \frac{M_1 + M_2 + \ldots + M_{N-1}}{M_{tot}}(\mathbf{R}_{N-1} + \mathbf{T}_{N-1} + \mathbf{L}_{N-1}) \qquad \text{(A2)}$$

Equation (A2) can be written in terms of the vectors $\mathbf{D}_i$, $\mathbf{H}_i$ and $\mathbf{P}_i$ by using Equations (4) through (6), to yield:

$$\mathbf{S}(t) = \mathbf{V}_g + (\mathbf{D}_1 + \mathbf{H}_1 + \mathbf{P}_1) + \ldots + (\mathbf{D}_{N-1} + \mathbf{H}_{N-1} + \mathbf{P}_{N-1}) \qquad \text{(A3)}$$

Using Equation set (3) and the fact that the end effector position is always equal to $\mathbf{S}(t) + \mathbf{R}_N$ gives:

$$\mathbf{E}(t) = \mathbf{V}_g + \mathbf{V}_1 + \mathbf{P}_1 + \ldots + \mathbf{P}_{N-1} + \mathbf{V}_N \qquad \text{(A4)}$$

It should be noted that this equation does not depend upon the existance of the Virtual Manipulator. The vector chain represented by Equation (A4) describes the end effector position relative to the N reference frame for all time.

For the initial manipulator position the VM constucted according to the procedure outlined in section 4 has an end point described by the following vector chain,

$$\mathbf{V}_g + \mathbf{V}_1 + \ldots + \mathbf{V}_N \qquad \text{(A5)}$$

Comparing Equations (A4) and (A5) it follows that in the initial position, when the $\mathbf{P}_i$'s are equal to zero, the end effector coincides with the VM end point.

Now it will be proven that as the real manipulator moves the VM joint motions described in section 4 will keep the VM end point on the real end effector. Say the manipulator goes through some joint movement, from section 4, the following vector chain describes the VM end point, where the $\mathbf{P}_i$'s are no longer zero,

$$\mathbf{V}_g^* + \mathbf{V}_1^* + \mathbf{P}_1^* + \ldots + \mathbf{P}_{N-1}^* + \mathbf{V}_N^* \qquad \text{(A6)}$$

In the following paragraphs it will be proven that the vectors $\mathbf{V}_i^*$, $\mathbf{V}_g^*$ and $\mathbf{P}_i^*$ in Equation (A6) are the same as $\mathbf{V}_i$, $\mathbf{V}_g$ and $\mathbf{P}_i$ in Equation (A4), respectively, therefore, the VM end point will coincide with the real end effector. The vector $\mathbf{V}_g$ is always constant, therefore,

$$\mathbf{V}_g^* = \mathbf{V}_g \qquad \text{(A7)}$$

The initial real manipulator links are composed of vectors $\mathbf{L}_{i-1} + \mathbf{R}_i$ and since the manipulator links are rigid, the magnitude of the vectors $\mathbf{L}_{i-1}$ and $\mathbf{R}_i$ and the angles between them are always constant. Since the magnitudes of $\mathbf{L}_{i-1}$ and $\mathbf{R}_i$ are constant, then from Equations (4) and (5) the magnitudes of $\mathbf{H}_{i-1}$ and $\mathbf{D}_i$ will also be constants. It can also be seen that the angles between $\mathbf{H}_{i-1}$ and $\mathbf{D}_i$ are constant. Then

$$|\mathbf{V}_i| = |\mathbf{H}_{i-1} + \mathbf{D}_i| \qquad \forall t, i \tag{A8}$$

The Virtual Manipulator links are composed of the $\mathbf{V}_i^*$ vectors. These links don't change their shapes and lengths as a function of time and since magnitudes of $\mathbf{V}_i^*$ are initially equal to magnitudes of $\mathbf{V}_i$, and magnitudes of $\mathbf{V}_i$ do not change with time it follows that:

$$|\mathbf{V}_i^*| = |\mathbf{H}_{i-1} + \mathbf{D}_i| = |\mathbf{V}_i| \qquad \forall t, i \tag{A9}$$

The magnitude of the vectors $\mathbf{P}_i$ in Equation (A4) and the $\mathbf{P}_i^*$ vectors in Equation (A6) are both obtained from the real manipulator prismatic joint translations, using Equation (6), therefore by definition,

$$|\mathbf{P}_i^*| = |\mathbf{P}_i| \qquad \forall t, i \tag{A10}$$

Now it will be proven that the direction of the vectors $\mathbf{V}_i^*$ and $\mathbf{P}_i^*$ in Equation (A6) are parallel to vectors $\mathbf{V}_i$ and $\mathbf{P}_i$ in Equation (A4), respectively.

First it can be established that the rotations of the first VM link are set equal to the vehicle rotations and hence the first VM link will always be parallel to the vehicle. Therefore,

$$\mathbf{V}_1^* = \mathbf{V}_1 \qquad \forall t \tag{A11}$$

Since axis of rotation or translation of the first real and virtual joints are fixed relative to their corresponding first links, and the rotations of the first VM link is the same as the vehicle, and these axes are initially constructed to be parallel, then they will always be parallel.

Now consider the case when the first real manipulator joint is revolute. The elements of the second VM link $\mathbf{H}_1^*$ and $\mathbf{D}_2^*$ will be parallel to $\mathbf{L}_1$ and $\mathbf{R}_2$ and in turn the vectors $\mathbf{V}_2^*$ and $\mathbf{V}_2$ will be parallel and

$$\mathbf{V}_2^* = \mathbf{V}_2 \qquad \forall t \tag{A12}$$

because the rotational axis of the first VM and real manipulator joints are parallel, as shown above, and the magnitude of their rotations are equal by construction.

In the cases where the first joint is prismatic, the elements of the second VM link $\mathbf{H}_1^*$ and $\mathbf{D}_2^*$ will be parallel to $\mathbf{L}_1$ and $\mathbf{R}_2$ and in turn the vectors $\mathbf{V}_2^*$ and $\mathbf{V}_2$ will be parallel and

$$\mathbf{V}_2^* = \mathbf{V}_2 \qquad \forall t \tag{A13}$$

because the VM and real manipulator translational axis for this joint are parallel. In the same manner it is possible to show that

$$\mathbf{V}_i^* = \mathbf{V}_i \qquad \forall t, i \tag{A14}$$

Also, in a similar manner all the translational axis of the real manipulator and the VM are always parallel, and from Equation (A11),

$$\mathbf{P}_i^* = \mathbf{P}_i \qquad \forall t, i \tag{A15}$$

Substituting Equations (A15), (A14) and (A7) into (A6), and comparing the result with Equation (A4) shows that the end effector will always coincide with the VM end point, and this completes the proof.

S34-37
188202
108

Model Reduction for Discrete Bilinear Systems

A.M. King and R.E. Skelton
Purdue University
West Lafayette, IN 47907

P9 39/092
= aclu

1. Abstract

A model reduction method for discrete bilinear systems is developed which matches q sets of Volterra and covariance parameters. These parameters are shown to represent both deterministic and stochastic attributes of the discrete bilinear system. A reduced order model which matches these q sets of parameters is defined to be a q-Volterra covariance equivalent realization (q-Volterra COVER). An algorithm is presented which constructs a class of q-Volterra COVERs parameterized by solutions to a Hermitian, quadratic, matrix equation. The algorithm is applied to a bilinear model of a robot manipulator.

2. Inroduction

While model reduction of linear systems has been extensively researched over the past few years, little work has been done in the area of model reduction for nonlinear systems. One class of nonlinear systems which is especially appealing are bilinear systems ([1]-[4]). Bilinear systems are linear in the state variables, linear in the control variables, but nonlinear in the state and control. One reason that this class is of interest is that nonlinear systems which are linear in the control variables can be accurately approximated by bilinear models ([5],[6]). Bilinear approximations will in general have a higher order than the original nonlinear system and effective means for reducing bilinear models are needed.

Most approaches to model reduction of linear systems have strived to preserve or approximate a certain characterizing property of the full order model. For deterministic systems this property is typically the impulse response sequence or the system Hankel matrix (e.g., [7]-[10]). Model reduction of linear stochastic systems usually involves the output covariance sequence or the corresponding Hankel matrix (e.g., [11] and [12]). A model reduction technique which considers both deterministic and stochastic properties has also been developed ([13]-[15]) and the resulting reduced order models have been called q Markov COVERs (covariance equivalent realizations). The model reduction problem for discrete bilinear systems has recently received some attention. Hsu et al. [16] develop a method for deterministic, discrete, bilinear systems using a generalized Hankel matrix. Desai has proposed an approach to stochastic model reduction based on his realization theory ([17]).

In this paper we develop a model reduction algorithm analogous to the q Markov covariance equivalent realization approach for linear systems. The algorithm produces a class of reduced order models which exactly match a specified number of deterministic and stochastic parameters. This class of reduced order models is parameterized by the solutions to a Hermitian, quadratic, matrix equation. Section 3 presents the deterministic and stochastic attributes of a bilinear system which we will preserve in our method and defines a q-Volterra covariance equivalent realization. Next, in section 4 the model reduction algorithm is outlined. In section 5 a parameterization of reduced order models which match q Volterra parameters and q covariance parameters is formulated. Section 6 contains an application of the proposed algorithm to a two degree of freedom robot manipulator. The final sections are our concluding remarks, acknowledgements and references.

3. q-Volterra Covariance Equivalent Realizations

Consider the time invariant discrete bilinear system

$$x(k+1) = Ax(k) + \sum_{i=1}^{n_u} (N_i x(k) + b_i) u_i(k)$$

$$y(k) = Cx(k) \tag{1}$$

where A and N_i, $i=1,\ldots,n_u$ are $n_x \times n_x$ matrices, b_i, $i=1,\ldots,n_u$ are $n_x \times 1$ matrices and C is an $n_y \times n_x$ matrix. The state vector $x(.)$ is $n_x \times 1$, the inputs $u_i(.)$, $i=1,\ldots,n_u$ are scalar, zero mean, independent Guassian white noise processes with $Eu_i(j)u_i(k) = \delta_{jk}$ and for $j > k$, $Ex(k)u_i(j) = 0$. The output $y(.)$ is an $n_y \times 1$, zero mean, stationary stochastic process. We assume that the bilinear system driven by unit intensity Guassian white noise is stable in the sense that the state covariance

$$X \overset{\Delta}{=} \lim_{k\to\infty} Ex(k)x^*(k) > 0 \tag{2}$$

is finite. It can be shown that for these input processes the state covariance will satisfy the bilinear Liapunov equation

$$X = AXA^* + \sum_{i=1}^{n_u} N_i X N_i^* + BB^* \ , \ B \overset{\Delta}{=} [\ b_1 \ \ldots \ b_{n_u}\ ] \ . \tag{3}$$

We also assume that there are no redundant inputs or outputs (B has linearly independent columns and C has linearly independent rows).

Ruberti et al. [3] define the product * for an s×1 vector a and an r×1 vector b and the product [] for an m×nr matrix L and an n×s matrix M by

$$a * b \overset{\Delta}{=} \begin{vmatrix} ab_1 \\ \vdots \\ ab_r \end{vmatrix} \ , \ L \,\square\, M = [\ L_1 \ \ldots \ L_r\ ] \,\square\, M \overset{\Delta}{=} [\ L_1 M \ \ldots \ L_r M\ ] \ .$$

They also establish the following identity,

$$L[(Ma * b)] = (L \,\square\, M)(a * b) \ . \tag{4}$$

With these definitions (1) becomes

$$\begin{aligned} x(k+1) &= Ax(k) + N[x(k) * u(k)] + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{5}$$

and (3) may be expressed as

$$X = AXA^* + (N\square X)N^* + BB^* \ , \ N \overset{\Delta}{=} [\ N_1 \ \ldots \ N_{n_u}\ ] \ . \tag{6}$$

The zero initial state response of the bilinear system (5) is an infinite Volterra series [4]. This series in regular form is found to be

$$\begin{aligned} y(k) = &\sum_{i_1=1}^{k} CA^{i_1-1}Bu(k-i_1) + \sum_{i_2=1}^{k}\sum_{i_1=1}^{k-i_2} CA^{i_2-1}N\square A^{i_1-1}B[u(k-i_1-i_2)*u(k-i_2)] + \\ &\sum_{i_3=1}^{k}\sum_{i_2=1}^{k-i_3}\sum_{i_1=1}^{k-i_2-i_3} CA^{i_3-1}N\square A^{i_2-1}N\square A^{i_1-1}B[u(k-i_1-i_2-i_3)*u(k-i_2-i_3)*u(k-i_3)] + \ldots \end{aligned}$$

where identity (4) has been used repeatedly. The matrix valued function in each of the summations is called a Volterra kernel, the j^{th} Volterra kernel in regular form is then

$$h_j(i_j, i_{j-1}, \ldots, i_1) \overset{\Delta}{=} CA^{i_j-1}N\square A^{i_{j-1}-1}N \ldots N\square A^{i_1-1}B \tag{7}$$

where $i_m > 1$, $m=1,\ldots,j$ and the matrix N occurs exactly j-1 times. The step response, $u(k)=1_{n_u}$ for all $k>0$, is

$$y(k) = \sum_{j=1}^{k}\sum_{i_j=1}^{k}\sum_{i_{j-1}=1}^{k-i_j} \ldots \sum_{i_1=1}^{k-i_2\ldots-i_j} h_j(i_j, i_{j-1}, \ldots, i_1)1_{n_u^j} \ .$$

where 1_m is a column vector of ones with m elements. We shall call the coefficients in the step response the Volterra parameters of the bilinear system (5). The Volterra parameters of the j^{th} Volterra kernel are $n_y \times n_u^j$ matrices. We define the set of q^{th} order Volterra parameters as those coefficients in which the matrices A and N occur a total of q times. For example,

$$V_2 \overset{\Delta}{=} \{\ CA^2B \ , \ CAN\square B \ , \ CN\square AB \ , \ CN\square N\square B\ \} \ .$$

We now see that the step response is completely characterized by the sets of Volterra parameters. We also observe that for each k a new set of Volterra parameters effects this response. That is, if a reduced order model matches the first q sets of Volterra parameters of the full order model then it will also match the step response for k=0,1,...,q+1.

In addition to Volterra parameters we are concerned with a covariance sequence for the bilinear system. Desai [17] and Frazho [18] utilize a covariance sequence which includes both second moments of the output and higher moments between the output and input processes in their bilinear stochastic realization theories. We also use this type of sequence, in particular the sequence of concern is

$$R_0(0) \overset{\Delta}{=} Ey(k)y^*(k) = CXC^*$$

$$R_1(1) \overset{\Delta}{=} Ey(k+1)y^*(k) = CAXC^*$$

$$R_1(0,0) \overset{\Delta}{=} Ey(k+1)[y(k) \otimes u(k)]^* = CN\Box XC^*$$

$$R_2(2) \overset{\Delta}{=} Ey(k+2)y^*(k) = CA^2XC^*$$

$$R_2(0,1) \overset{\Delta}{=} Ey(k+2)[y(k) \otimes u(k+1)]^* = CN\Box AXC^*$$

$$R_2(1,0) \overset{\Delta}{=} Ey(k+2)[y(k) \otimes u(k)]^* = CAN\Box XC^*$$

$$R_2(0,0,0) \overset{\Delta}{=} Ey(k+2)[y(k) \otimes u(k) \otimes u(k+1)]^* = CN\Box N\Box XC^*$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

where the subscript indicates the total number of occurrences of A and N, and the integers in parenthesis represent the powers of A from left to right. A typical element of the sequence is then

$$\begin{aligned} &R_{j-1+i_j+i_{j-1}+\ldots+i_1}(i_j,i_{j-1},\ldots,i_1) \\ &\overset{\Delta}{=} Ey(k+j-1+i_j+i_{j-1}+\ldots+i_1)[y(k)*u(k+i_1)*u(k+1+i_2+i_1)*\ldots*u(k+j-2+i_{j-1}+\ldots+i_1)]^* \\ &= CA^{i_j}N\Box A^{i_{j-1}}N\ldots N\Box A^{i_1}XC^* \; . \end{aligned} \tag{8}$$

As with the Volterra parameters we shall define the set of q^{th} order covariance parameters, R_q, as those covariances in which the matrices A and N occur a total of q times, that is the set of second order covariance parameters is

$$R_2 \overset{\Delta}{=} \{ CA^2XC^* \, , CN\Box AXC^* \, , CAN\Box XC^* \, , CN\Box N\Box XC^* \} \; .$$

These sets of covariance parameters completely characterize the stochastic bilinear system. It is worth noting that if a reduced order model matches the first q sets of covariance parameters of the full order model it will also match exactly the mean square value of the output and all output and input correlations up to q steps in time.

Consider now a reduced-order bilinear model

$$\begin{aligned} x_R(k+1) &= A_Rx_R(k) + N_R[x_R(k) \otimes u(k)] + B_Ru(k) \\ y_R(k) &= C_Rx_R(k) \end{aligned} \tag{9}$$

where $x_R(.)$ is an $n_r \times 1$ vector, $n_r < n_x$, $y_R(.)$ is an $n_y \times 1$ vector, and A_R, N_R, B_R, C_R are matrices of appropriate dimensions. In addition, we assume that the state covariance X_R of the reduced model driven by zero mean Guassian white noise is the unique positive definite solution to

$$X_R = A_RX_RA_R^* + (N_R\Box X_R)N_R^* + B_RB_R^* \; . \tag{10}$$

We now define a particular type of reduced order model for discrete bilinear systems.

<u>Definition</u>: The reduced order model (9), with state covariance X_R satisfying (10) is a <u>q-Volterra COVariance Equivalent Realization</u> (q-Volterra COVER) of the bilinear system (5) whenever

$$V_{R_i} = V_i, \; i=0,1,\ldots,q-1$$

and

$$R_{R_i} = R_i, \; i=0,1,\ldots,q-1$$

where V_{R_i} and R_{R_i} denote the sets of i^{th} order Volterra and covariance parameters of the reduced order model, respectively.

An algorithm which constructs the q-Volterra COVERs of a full order model is our main objective. One such algorithm is presented next.

4. A Model Reduction Algorithm

Suppose that a full order model (5) and a state covariance satisfying (6) are given. The q^{th} observabilit; matrix ([3],[4],[16]) of this model is

$$O_q \triangleq \begin{vmatrix} Q_0 \\ Q_1 \\ \vdots \\ Q_{q-1} \end{vmatrix} , Q_0 \triangleq C , Q_i \triangleq \begin{vmatrix} Q_{i-1}A \\ Q_{i-1}N_1 \\ \vdots \\ Q_{i-1}N_{n_u} \end{vmatrix} , i=1,\ldots,q-1 . \qquad (11$$

The matrix partitions Q_i have dimension $(n_u+1)^{i-1}n_y \times n_x$, $i=0,1,\ldots,q-1$. We observe that the matrices Q_qB an Q_qXC^* contain the same information as the sets V_q and R_q, respectively. Using the full order model we construc the following matrices

$$D_q \triangleq O_qXO_q^* \qquad (12$$

$$\overline{D}_q \triangleq O_q(AXA^*+(N\square X)N^*)O_q^* = O_q[\ A\ (N\square X^{1/2})\ ][\ A\ (N\square X^{1/2})\ ]^*O_q^* . \qquad (13$$

As a consequence of the quadratic form and using the Liapunov equation (6) it immediately follows that the rang spaces of these matrices are

$$R(D_q) = R([\ AX^{1/2}\ (N\square X^{1/2})\ B\ ])\ ,\ R(\overline{D}_q) = R([\ AX^{1/2}\ (N\square X^{1/2})\ ]) \qquad (14$$

and it is obvious that $R(\overline{D}_q)$ is contained in $R(D_q)$.

We now compute a full rank factorization of D_q

$$D_q = P\Lambda P^* \qquad (1!$$

where $\mathrm{rank}(D_q) = r <= n_x$. By virtue of the full rank factorization the columns of P form a basis for the rang space of D_q. Introducing P^+, the Moore-Penrose inverse of P, then it is well known that PP^+ is an orthogon projector onto the range of D_q ([19]). We now partition P into blocks whose row dimensions are compatible wi the partitions of O_q (11)

$$P = \begin{vmatrix} P_0 \\ P_1 \\ \vdots \\ P_{q-1} \end{vmatrix} , P_i = \begin{vmatrix} P_i^A \\ P_i^{N_1} \\ \vdots \\ P_i^{N_{n_u}} \end{vmatrix} , i=1,\ldots,q-1 \qquad (1$$

and define new matrices

$$P_A \triangleq \begin{vmatrix} P_1^A \\ \vdots \\ P_{q-1}^A \end{vmatrix} , P_{N_j} \triangleq \begin{vmatrix} P_1^{N_j} \\ \vdots \\ P_{q-1}^{N_j} \end{vmatrix} , j=1,\ldots,n_u , F \triangleq [\ P_A\ P_{N_1}\ \ldots\ P_{N_{n_u}}\ ] , \overline{P} \triangleq \begin{vmatrix} F \\ G \end{vmatrix} . \qquad (1$$

The matrix G is $(n_u+1)^{q-1}n_y \times (n_u+1)r$ and it must be determined such that

$$\overline{D} = \overline{P}\overline{\Lambda}\overline{P}^* , \overline{\Lambda} = \mathrm{diag}(\Lambda)_{n_u+1} . \qquad 1$$

where $\overline{\Lambda}$ is a block diagonal matrix with n_u+1 blocks. Given these constructions we now state our main result.

Theorem 1: Given a discrete bilinear system $\{A,N,B,C,X\}$ and a matrix G in (17) such that (18) is satisfied tl the reduced order model $\{A_R,N_R,B_R,C_R,X_R\}$ of order n_r defined by

$$[\ A_R\ N_R\ ] \triangleq P^+\overline{P}\ ,\ B_R \triangleq P^+O_qB\ ,\ C_R \triangleq P_0\ ,\ X_R \triangleq \Lambda\ ,\ n_r \triangleq r \qquad (:$$

where r, P, Λ are from the full rank decomposition of D_q (15), P_0 is from the partition of P (16), and satisfies (18), is a q-Volterra COVER.

Proof: First we will show that P is the q^{th} observability matrix of the reduced order model (19). Using decompositions (15), (18) and the range space descriptions (14) we find that $\overline{P}$ is in the range space of P so that (19) leads to

$$P[\ A_R\ N_R\ ] = \overline{P}$$

which implies that the partitions of P have the required structure (11)

$$P_0 = C_R\ ,\ P_i = \left|\begin{matrix} P_{i-1}A_R \\ P_{i-1}N_{R_1} \\ \vdots \\ P_{i-1}N_{R_{n_u}} \end{matrix}\right|\ ,\ i=1,\ldots,q-1\ .$$

To show that the reduced order model satisfies the bilinear Liapunov equation we first substitute (19) into (10) which leads to

$$\Lambda = P^{+}\overline{P\Lambda P^{*}}P^{+^{*}} + P^{+}O_qBB^{*}O_q^{*}P^{+^{*}}\ .$$

Using (12), (13), (15), (18), and by pre and post multiply by P and P^*, respectively, we have

$$O_qXO_q^{*} = PP^{+}O_qAXA^{*}O_q^{*}P^{+^{*}}P^{*} + PP^{+}O_q(N\square X)N^{*}O_q^{*}P^{+^{*}}P^{*} + PP^{+}O_qBB^{*}O_q^{*}P^{+^{*}}P^{*}\ .$$

Now using the projection property of PP^+ we find that

$$O_q(X = AXA^{*} + (N\square X)N^{*} + BB^{*})O_q^{*}$$

which is known to be satisfied (6). To show that the model (19) matches Volterra parameters we again use the projection property

$$O_{q_R}B_R = PP^{+}O_qB = O_qB$$

and the matching of covariance parameters follows directly from (12),(15)

$$O_{q_R}X_RO_{q_R}^{*} = P\Lambda P^{*} = O_qXO_q^{*}\ .$$

#

Our remaining task is to determine the unknown matrix G in (17) in order to satisfy (18). This is the topic of the next section.

5. Parameterization of q-Volterra COVERs

To obtain a characterization of the matrix G we first examine the structure of the matrices $\overline{D}_q$ and $\overline{P}$. We observe that $\overline{D}_q$ can be partitioned as

$$\overline{D}_q = \left|\begin{matrix} \overline{D}_{q-1} & \overline{d}_q \\ \overline{d}_q^{*} & \overline{d}_{qq} \end{matrix}\right| \tag{20}$$

and that the partitioned form of the constraint (18) leads to the three relations

$$F\overline{\Lambda}F^{*} = \overline{D}_{q-1}\ ,\ F\overline{\Lambda}G^{*} = \overline{d}_q^{*}\ ,\ G\overline{\Lambda}G^{*} = \overline{d}_{qq}\ . \tag{21}$$

The first relation is satisfied by virtue of the constuction of of $\overline{P}$ (17). It is easily seen that $\overline{d}_q$ is contained in the range space of F so that the second relation is consistent and G^* may be expressed as ([19])

$$G^{*} = \overline{\Lambda}^{-1}(F^{+}\overline{d}_q + (I - F^{+}F)Y) \tag{22}$$

where Y is an unknown matrix with dimension $(n_u+1)r\times(n_u+1)^{q-1}n_y$. Substituting for G in the last relation we find that Y must satisfy the Hermitian, quadratic, matrix equation

$$Y^{*}KY + L^{*}Y + Y^{*}L + M = 0 \tag{23}$$

$$K \triangleq (I - F^{+}F)\overline{\Lambda}^{-1}(I - F^{+}F) = K^{*}\ ,\ L \triangleq (I - F^{+}F)\overline{\Lambda}^{-1}F^{+}\overline{d}_q\ ,\ M \triangleq \overline{d}_q^{*}F^{+^{*}}\overline{\Lambda}^{-1}F^{+}\overline{d}_q - \overline{d}_{qq} = M^{*}\ . \tag{24}$$

By inspection we see that the matrix K is nonnegative definite, and that the columns of the matrix L are contained in the range space of K. Based on these observations we now state a theorem which is motivated by a result of Crone [20].

Theorem 2: Let K be an m×m nonnegative definite matrix with rank t, L an m×n matrix whose columns are containe in the range space of K and M an n×n Hermitian matrix. Then the matrix equation

$$Y^*KY + L^*Y + Y^*L + M = 0 \qquad (25)$$

has a solution if and only if

$$L^*K^+L - M \geq 0 \quad \text{and} \quad \text{rank}(L^*K^+L - M) = s \leq t = \text{rank}(K) . \qquad (26)$$

When these conditions hold Y is a solution if and only if it has the form

$$Y = K^{+/2}(V\Sigma^{1/2}U^* - K^{+/2}L) + (I - K^{+/2}K^{1/2})\bar{Y} \qquad (27)$$

where $K^{1/2}$ is the unique nonnegative definite square root of K and $K^{+/2}$ is the Moore-Penrose inverse of $K^{1/2}$. The matrix V is an m×s matrix, Σ is s×s and U is n×s and they must satisfy

$$V^*V = I \text{ , } R(V) \text{ is contained in } R(K) \text{ , } U^*U = I \text{ , } \Sigma > 0 \text{ , } U\Sigma U^* = L^*K^+L - M \text{ .}$$

$\bar{Y}$ is an arbitrary m×n matrix.

Proof: It is well known that if $K = W\Omega W^*$ is a full rank singular value decomposition (SVD), then

$$K^{1/2} = W\Omega^{1/2}W^* \text{ , } K^{+/2} = W\Omega^{+/2}W^*$$

and it follows that K, $K^{1/2}$, $K^{+/2}$ all have the same range space which is spanned by the columns of W, an m× column unitary matrix. By the hypothesis that the columns of L are in the range space of K, equation (25) i satisfied if and only if

$$(K^{1/2}Y + K^{+/2}L)^*(K^{1/2}Y + K^{+/2}L) = L^*K^+L - M$$

which is consistent if and only if $L^*K^+L-M \geq 0$. All matrix factors of this relation are

$$K^{1/2}Y + K^{+/2}L = V\Sigma^{1/2}U^*$$

where $U\Sigma U^*$ is the full rank SVD of L^*K^+L-M and V is any column unitary matrix of appropriate dimension, m×s. T find a solution Y we must solve the following linear equation

$$K^{1/2}Y = V\Sigma^{1/2}U^* - K^{+/2}L \text{ .} \qquad (28$$

This equation is consistent if and only if V is contained in the range space of K. Since V is column unitary th range space of V may be any m dimensional space with rank s, solutions of (28) exsist if and only i $\text{rank}(L^*K^+L-M) = s \leq t = \text{rank}(K)$. Given that equation (28) is consistent then Y is a solution if and only if i has the following form

$$Y = K^{+/2}(V\Sigma^{1/2}U^* - K^{+/2}L) + (I - K^{+/2}K^{1/2})\bar{Y}$$

where $\bar{Y}$ is an arbitrary m×n matrix.

The results of this theorem show that the matrix L^*K^+L-M is the key to solutions of the quadratic matri equation (23). Substituting for K, L, M from equations (24), and using the rules for the Moore-Penrose inverse o a matrix product ([21]), we find that

$$L^*K^+L - M = \bar{d}_{qq} - \bar{d}_q^*F^{+*}\bar{\Lambda}^{-1/2}(I - (\bar{\Lambda}^{-1/2}(I - F^+F))(\bar{\Lambda}^{-1/2}(I - F^+F))^+)\bar{\Lambda}^{-1/2}F^+\bar{d}_q \text{ .} \qquad (29$$

From this equation we find an interesting result on the Moore-Penrose inverse of a quadratic form which we stat without proof.

Fact: The Moore-Penrose inverse of the quadratic form $F\bar{\Lambda}F^*$ is

$$(F\bar{\Lambda}F^*)^+ = F^{+*}\bar{\Lambda}^{-1/2}(I - (\bar{\Lambda}^{-1/2}(I - F^+F))(\bar{\Lambda}^{-1/2}(I - F^+F))^+)\bar{\Lambda}^{-1/2}F^+ \qquad (30$$

where $\bar{\Lambda}$ is a positive definite matrix and F is any matrix which is multiplication compatible.

Using this result and equation (21) in (29) we find that

$$L^*K^+L - M = \bar{d}_{qq} - \bar{d}_q^*\bar{D}_{q-1}^+\bar{d}_q \qquad (31$$

which is guaranteed to be nonnegative definite ([22]). Thus the first part of the constraint (26) of theorem will always be satisfied.

To show that the second part of the constraint (26) will also be satisfied we note that from the range spac description (14),

$$r = \text{rank}(D_q) \geq \text{rank}(\bar{D}_q) \tag{32}$$

and from the definition of K (24), the relations (21) and the column dimension of F,

$$t \triangleq \text{rank}(K) = \text{rank}(I - F^+F) = (n_u+1)r - \text{rank}(\bar{D}_{q-1}) \ . \tag{33}$$

Rohde [23] has shown that the partitioning (20) of the nonnegative definite matrix $\bar{D}_q$ implies

$$\text{rank}(\bar{D}_q) = \text{rank}(\bar{D}_{q-1}) + \text{rank}(\bar{d}_{qq} - \bar{d}_q^*\bar{D}_{q-1}^+\bar{d}_q) \tag{34}$$

and by using (31)

$$s \triangleq \text{rank}(L^*K^+L - M) = \text{rank}(\bar{D}_q) - \text{rank}(\bar{D}_{q-1}) \ . \tag{35}$$

Collecting equations (32),(33) and (34) we find that $s < t$ and therefore the second part of the constraint (26) in theorem 2 will always be satisfied.

We have shown that solutions of (23),(24) always exist and by theorem 2 they will have the form

$$Y = K^{+/2}(V\Sigma^{1/2}U^* - K^{+/2}L) + (I - K^{+/2}K^{1/2})\bar{Y} \tag{36}$$

where $U\Sigma U^*$ is the full rank singular value decomposition of L^*K^+L-M, V is any column unitary matrix whose range space is contained in the range space of K and $\bar{Y}$ is arbitrary. We observe that the second term of (36) is in the null space of K which is also the range space of F^*. It follows that when (36) is substituted into the expression for G^* (22) that this term will be annihilated by $(I-F^+F)$ which represents a projection onto the null space of F along the range space of F^*. The first term of (36) is in the range space of K, or the null space of F, so that under the projection $(I-F^+F)$ it remains unchanged. Therefore G^* becomes

$$G^* = \bar{\Lambda}^{-1}(F^+\bar{d}_q + K^{+/2}(V\Sigma^{1/2}U^* - K^{+/2}L))$$

or by using equation (24) and conjugate transposing

$$G = \bar{d}_q^*F^{+^*}\bar{\Lambda}^{-1}(I - (I - F^+F)((I - F^+F)\bar{\Lambda}^{-1}(I - F^+F))^+\bar{\Lambda}^{-1}) + \tag{37}$$

$$(\bar{d}_{qq} - \bar{d}_q^*\bar{D}_{q-1}^+\bar{d}_q)^{1/2}UV^*((I - F^+F)\bar{\Lambda}^{-1}(I - F^+F))^{+/2}\bar{\Lambda}^{-1} \ .$$

Equation (37) is an explicit expression for G which was the objective of this section. All of the freedom in G is contained in the column unitary matrix V whose range space is constrained to be in the null space of F.

6. Application to a Robot Manipulator

Consider the two degree of freedom manipulator illustrated in Figure 1. The arm has its center of mass at point C, and it may be translated through or rotated about the fixed point O by the force F and torque T, respectively. The manipulator carries a load at the point L.

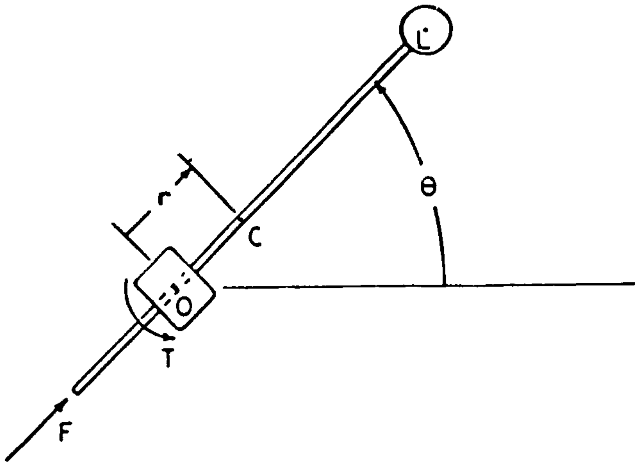


Figure 1. Two Degree of Freedom Manipulator

Treating the load as a point mass and allowing for joint stiffness and damping, the equations of motion are

$$(m + M)\ddot{r} + b_r\dot{r} + k_r r - (m + M)r\dot{\theta}^2 - Ma\dot{\theta}^2 = F$$

$$(J + I \bullet Ma^2 + 2Mar + (m + M)r^2)\ddot{\theta} + b_\theta\dot{\theta} + k_\theta\theta + 2(m + M)r\dot{r}\dot{\theta} + 2Ma\dot{r}\dot{\theta} = T$$

where a is the distance from C to L, M is the mass of the load, J is the moment of inertia of the joint, m is the mass of the arm and I is its moment of inertia about C. Joint stiffness and damping are represented by k_r, k_θ and b_r, b_θ, respectively. Introducing the state vector and the control

$$z = [\ r\ \dot{r}\ \theta\ \dot{\theta}\ ]^T\ ,\ u = [\ F\ T\ ]^T$$

then the equations of motion have the generic form

$$\dot{z} = f(z) + g(z)u\ .$$

A bilinear model of the manipulator can be constructed by expanding each of the functions f(.) and g(.) into a power series and introducing a new state vector which contains higher order terms in z ([4]-[6]). Using the first three terms of the Taylor series expansions of f(.) and g(.) and letting

$$x = [\ r\ \dot{r}\ \theta\ \dot{\theta}\ r^2\ \dot{r}r\ \theta r\ \dot{\theta}r\ \dot{r}^2\ \theta\dot{r}\ \dot{\theta}\dot{r}\ \theta^2\ \dot{\theta}\theta\ \dot{\theta}^2\ r^3\ \ldots\ \dot{\theta}^3\ ]^T$$

then we have a 34th order bilinear model and after discretization it has the form (1).

For purposes of illustration, the following numerical values are chosen: a = 1 m, m = 100 kg, M = 50 kg, J = I = 100 kg-m^2, and k_r = 6 N/m, k_θ = 2.5 N/m, b_r = 3 N-sec/m, b_θ = 5 N-sec/m. Figure 2 shows the step response of the nonlinear equations of motion and the full order bilinear model. The bilinear model provides a fair approximation to the true nonlinear system. A more accurate approximation could be made by retaining higher order terms in the power series expansions.

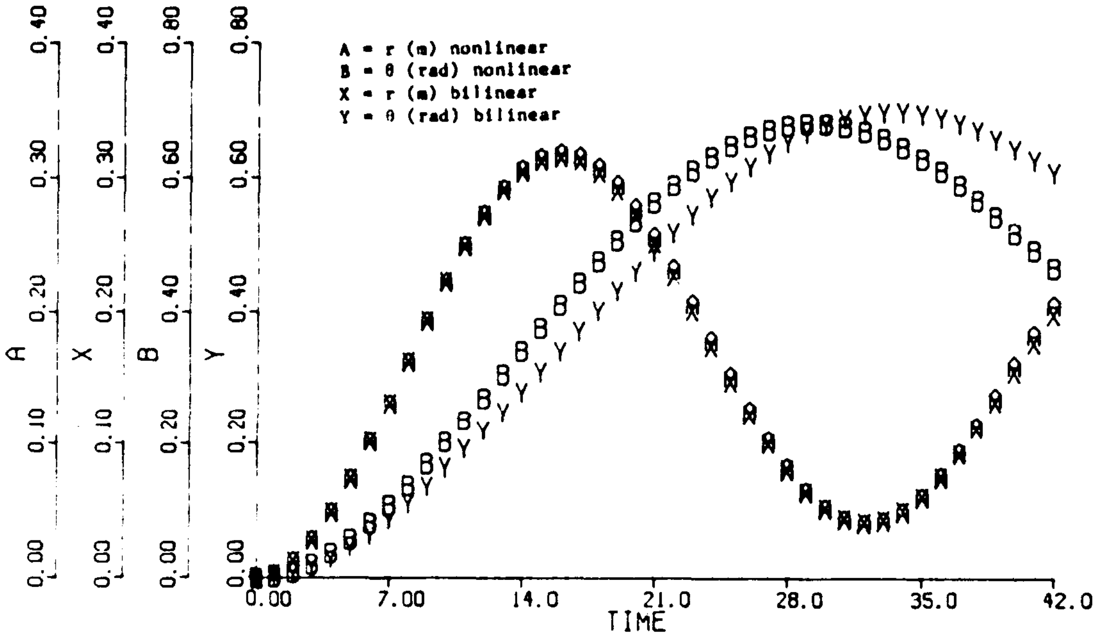


Figure 2. Step Response of Nonlinear and Full Order Bilinear Models

Applying the model reduction algorithm with q=3 (matching three sets of Volterra and covariance parameters) a class of 3-Volterra COVERs was obtained. These reduced models have 14 states which is a greater than fifty percent reduction in model order. Figure 3 shows the response of a reduced model from the class of 3-Volterra COVERs and the response of the full order model to a unit pulse input with a 3 second duration. Figure 4 shows the response of thes models driven by a unit intensity Gaussian white noise process. In Figure 3 we see that the response of the full and reduced order bilinear models are nearly identical for the first 10 seconds. Similarly, in Figure 4 the reduced order model mimics the full order model initially. These observations are in accordance with the theory which states that the response of the reduced order model equals that of the full order system for q steps in time. However, in both cases the quality of the response of the reduced order model deteriorates with time and it eventually goes unstable. This instability is input dependent and possibly in a closed loop setting the model behavior would be acceptable for greater periods in time.

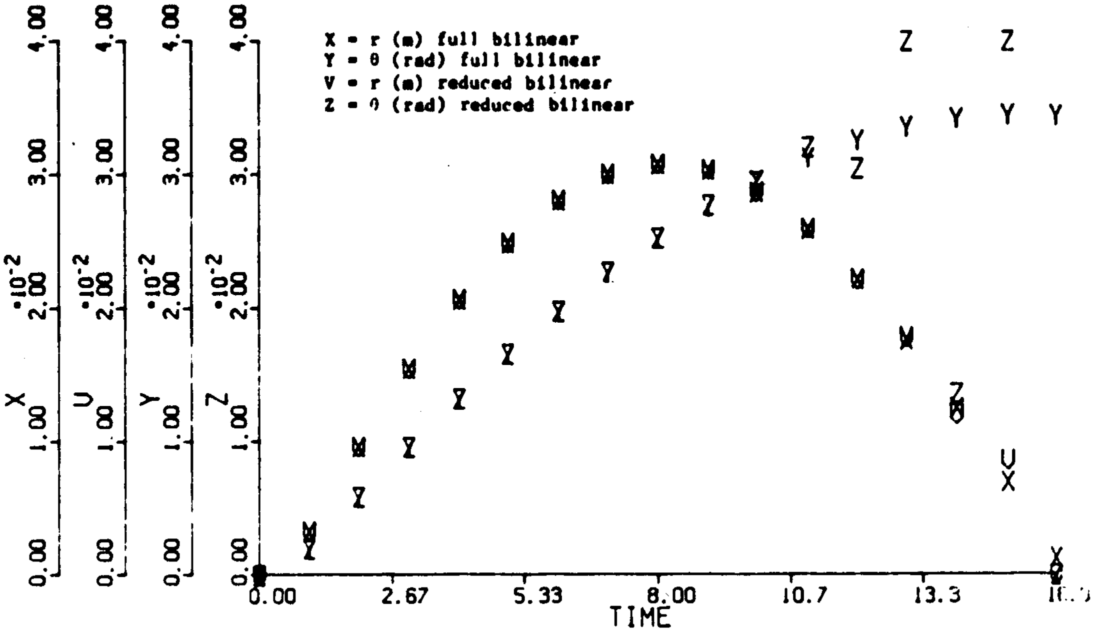


Figure 3. Deterministic Response of Full and Reduced Order Bilinear Models

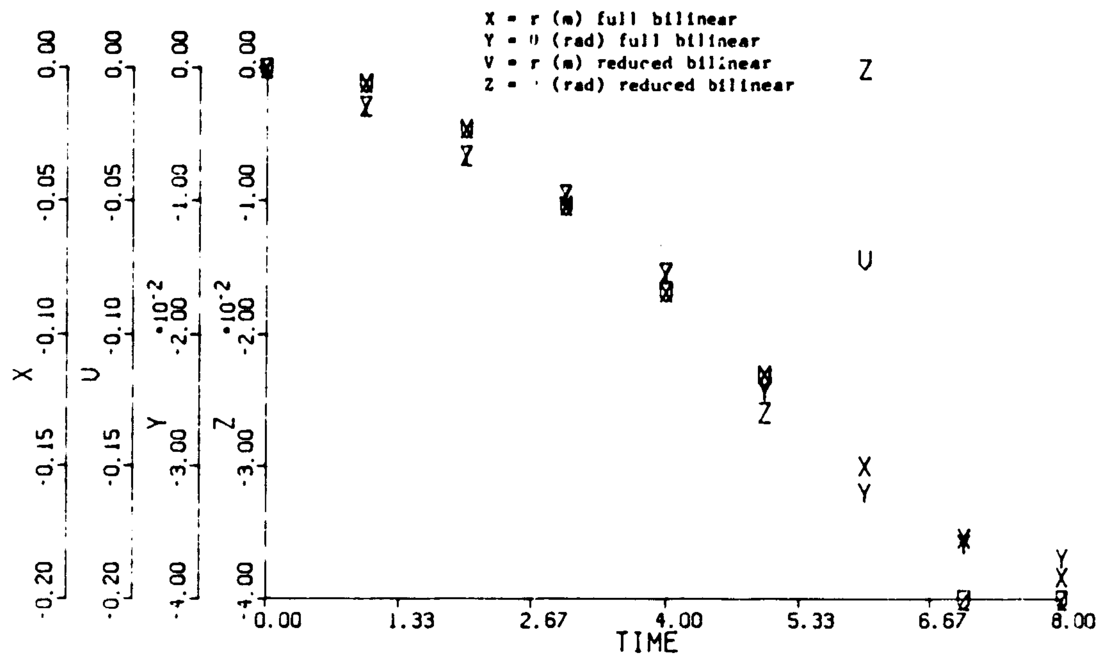


Figure 4. Stochastic Response of Full and Reduced Order Bilinear Models

7. Conclusions

A sequence of sets of Volterra parameters characterizes the deterministic bilinear system, and a sequence of sets of covariance parameters describes the stochastic bilinear system. A model reduction technique was developed for discrete bilinear systems which generates a class of reduced order models which exactly match the first q sets of Volterra and covariance parameters of the full order model. These models are therefore called q-Volterra covariance equivalent realizations, or q-Volterra COVERs. Methods to choose specific models from within the class to satisfy additional modelling considerations is a topic of future research.

8. Acknowledgements

Part of this research was sponsored by the National Science Foundation, grant MEA-8405133, and by the Hughes Aircraft Company through the Hughes Fellowship Program. We also acknowledge Uday B. Desai of Washington State University whose assistance was invaluable.

9. References

[1] C. Bruni,G. DiPillo, and G. Koch, "Bilinear Systems: An Appealing Class of 'Nearly Linear' Systems in Theory and Applications," IEEE Trans. Autom. Contr. Vol. AC-19, No. 4, Aug. 1974, pp. 334-348.

[2] R.R. Mohler, Bilinear Control Processes, Academic Press, New York, U.S.A., 1973.

[3] A. Ruberti, A. Isidori and P. D'Alessandro, Theory of Bilinear Dynamical Systems, Springer-Verlag, New York, U.S.A., 1972.

[4] W. J. Rugh, Nonlinear System Theory: The Volterra/Wiener Approach, Johns Hopkins University Press, Baltimore, U.S.A., 1981.

[5] A. J. Krener, "Linearization and Bilinearization of Control Systems," Proc. Allerton Conf. Circuits and Systems, Champagne-Urbana, U.S.A., 1974, pp. 834-843.

[6] S. Svoronos, G. Stephanopoulos and R. Aris, "Bilinear Approximation of General Nonlinear Dynamic Systems with Linear Inputs," Int. J. Control, Vol. 31, No. 1, 1980, pp. 109-126.

[7] B. C. Moore, "Principle Component Analysis in Linear Sytems: Controllability, Observability, and Model Reduction," IEEE Trans. Auto. Control, Vol. 26, No. 1, Feb. 1981, pp. 17-32.

[8] L. Pernebo, and L. M. Silverman, "Model Reduction via Balanced State Space Representations," IEEE Trans. Auto. Control, Vol. AC-27, No. 2, April 1982, pp. 382-287.

[9] S. Kung and D. Lin, "Optimal Hankel-Norm Model Reductions: Multivariable Systems," IEEE Trans. Auto. Control, Vol. AC-26, No. 4, Aug. 1981, pp. 832-853.

[10] K. Glover, "All Optimal Hankel Norm Approximations of Linear Multivariable Sytems and their L_∞ Error Bounds," Int. J. Control, Vol. 39, No. 6, 1984, pp. 1115-1193.

[11] U. B. Desai, D. Pal, and R Kirkpatrick, "A Realization Approach to Stochastic Model Reduction," Int. J. Control, Vol. 42, No. 4, 1985, pp. 821-838.

[12] B. J. Bacon and A. E. Frazho, "A Hankel Matrix Approach to Stochastic Model Reduction," IEEE Trans. Auto. Control, Vol. 30, No. 11, Nov. 1985, pp. 1138-1140.

[13] R. E. Skelton and B. D. O. Anderson, "q Markov Covariance Equivalent Realizations," Int. J. Control, Vol. 44, No. 5, 1986, pp. 1477-1490.

[14] D. A. Wagie and R. E. Skelton, "A Projection Approach to Covariance Equivalent Realizations of Discrete Systems," IEEE Trans. Auto. Control, Vol. AC-31, No. 12, Dec. 1986, 1114-1120.

[15] A. Yousuff, D. A. Wagie and R. E. Skelton, "Linear System Approximation Via Covariance Equivalent Realizations," J. Math. Analysis and Application, Vol. 106, No.1, Feb. 1985, pp. 91-115.

[16] C. S. Hsu, U. B. Desai and C. A. Crawley, "Realization and Approximation of Discrete Bilinear Systems," in Applied Digital Control, S. G. Tzafestas, Ed. North Holland, Amsterdam, The Netherlands, 1985, pp. 171-187.

[17] U. B. Desai, "Realization of Bilinear Stochastic Systems," IEEE Trans. Auto. Control, Vol. AC-31, No. 2, Feb. 1986, pp. 189-192.

[18] A. E. Frazho, "Schur Contractions and Stochastic Bilinear Systems," Proc. Conf. Inform. Sci. and Syst., Princeton, U.S.A., 1984, pp. 190-196.

[19] A. Ben-Israel and T. N. E. Greville, Generalized Inverses: Theory and Applications, Wiley, New York, U.S.A., 1974.

[20] L. Crone, "Second Order Adjoint Matrix Equations," Linear Algebra and its Applications, Vol. 39, 1981, pp. 61-72.

[21] T. N. E. Greville, "Note on the Generalized Inverse of a Matrix Product," Soc. Indust. Appl. Math. Review, Vol. 8, No. 4, Oct.1966, pp. 518-521.

[22] A. Albert, "Conditions for Positive and Nonnegative Definiteness in Terms of Pseudoinverses," J. Soc. Indust. Appl. Math., Vol. 17, No. 2, Mar. 1969, pp. 434-440.

[23] C. A. Rohde, "Generalized Inverses of Partitioned Matrices," J. Soc. Indust. Appl. Math., Vol. 13, No. 4, Dec. 1965, pp. 1033-1035.

High Gain Feedback and Telerobotic Tracking

D.E. Koditschek*
Yale University
New Haven, CT 06520-2157

Abstract

Asymptotically stable linear time invariant systems are capable of tracking arbitrary reference signals with a bounded error proportional to the magnitude of the reference signal (and its derivatives). It is shown that a similar property holds for a general class of nonlinear dynamical systems which includes all robots. As in the linear case, the error bound may be made "arbitrarily" small by increasing the magnitude of the feedback gains which stabilize the system.

1 Introduction

Tracking is the archetypal pursuit of the control theorist. Given a dynamical system,

$$\begin{aligned} \dot{x} &= f(x,u), \\ y &= h(x) \end{aligned}$$

and a specified "reference signal", $r(t)$, it is required to find a control, $u^*(t)$ such that the forced system, $\dot{x} = f(x,u^*)$ "tracks" r in some sense — usually $\lim_{t\to\infty} y = r$. Solutions to such problems generally involve pre-filtering the reference trajectory through a suitable "feedforward" algorithm, and then adding a compensating error driven "feedback" term to arrive at the input, u^*. If the reference signal is known à priori, then the feedforward algorithm may entail pure differentiation to "pre-compensate" for the lags introduced by the dynamical system itself. However, on-line differentiation of unknown and unpredictable signals has long been eschewed by control theorists as an unreliable technique for both theoretical as well as practical reasons.

This paper considers the problem of tracking in the context of telerobotic manipulators. It is shown that a general class of highly nonlinear control systems which includes all robot models admits tracking algorithms based upon high gain linear state variable feedback. The choice of a pure feedback based algorithm for tracking is surely not optimal in any sense of the word. However, the only other techniques which are known to guarantee tracking for this class of systems make use of feedback algorithms which attempt exact cancellation [1,2,3], (or "nearly" exact cancellation, e.g. [4,5]) of intrinsic nonlinear dynamical terms via feedback, and pure differentiation of the reference trajectory in the feedforward path. In robot applications admitting the use of a "high level" planner it is plausible that the entire future strategy might be made available at once to the "low level" controller in which case tracking schemes requiring pure differentiation of the reference signal might be acceptable. In telerobotic applications the reference signal is, by definition, à priori unknown: it is generated as a record of the unpredicted arbitrary motion of a human agent of control. Schemes which require pure differentiation will probably not be useful in this context.

In a sense, the result reported here simply represents another example of the similarity between general mechanical systems and second order linear systems. It is well known that asymptotically stable linear time invariant systems are capable of tracking arbitrary reference signals with a bounded error proportional to the magnitude of the reference signal (and its derivatives). For a fixed bound on this magnitude, the asymptotic tracking error may be made "arbitrarily" small by increasing the magnitude of the eigenvalues in the left half of the complex plane. In practice, this is accomplished by increasing the gain of linear feedback compensators. In this paper it is shown that the analogous property holds true for the more general class of nonlinear mechanical systems.

As in the theory of linear servomechanisms, a practical obstacle to the systematic use of high gain feedback techniques in telerobotic applications is the inevitable presence of actuator torque limitations. Practical tracking strategies which address this problem while maintaining convergence guarantees are very much needed. This important consideration is entirely ignored here. The problem of characterizing the transient response of feedback compensated nonlinear mechanical systems is the topic of a paper currently in progress.

*This work is supported in part by the National Science Foundation under grant no. DMC-8505160

2 Preliminary Discussion

2.1 Notation and Definitions

If $f : \mathbf{R}^n \to \mathbf{R}^m$ has continuous first partial derivatives, denote its $m \times n$ jacobian matrix as Df. When we require only a subset of derivatives, e.g. when $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, and we desire the jacobian of f with respect to the variables $x_1 \in \mathbf{R}^{n_1}$, as x_2 is held fixed, we may write

$$D_{x_1} f \triangleq Df \begin{bmatrix} I_{n_1 \times n_1} \\ 0 \end{bmatrix}.$$

If $A : J \to \mathbf{R}^{n \times n}$ is a smooth map taking matrix values then let

$$\mu(A) \triangleq \sup_{q \in J} \sup_{\|x\|=1} |x^T A x|$$

and

$$\nu(A) \triangleq \inf_{q \in J} \inf_{\|x\|=1} |x^T A x|.$$

If J is compact, or the entries of A are bounded then both $\nu(A), \mu(A)$ are non-negative real numbers. For any constant matrix, $\mu(A)$ is the square root of the eigenvalue of greatest magnitude, while $\nu(A)$ is the square root of the eigenvalue of least magnitude of $A^T A$, from which it follows that

$$\mu(A) = \sup_{q \in J} \|A(q)\| \qquad 1/\nu(A) = \sup_{q \in J} \|A^{-1}(q)\|,$$

where $\|\cdot\|$ denotes the operator norm induced by the eulcidean norm of $\mathbf{R}^n$.

Given a set $\mathcal{P}$, a smooth scalar valued map, $v : \mathcal{P} \to \mathbf{R}$ is said to be *positive definite at a point* $p \in \mathcal{P}$ if $v(p) = 0$, and $v > 0$ in some open neighborhood of p. Given a smooth (time invariant) vector field, f, on some phase, space, $\mathcal{P}$, we shall say that, v, a positive definite map at $p_d \in \mathcal{P}$, constitutes a *Lyapunov function for f at p_d* if the time derivative along any motion of the vector field is non-positive,

$$\dot{v} = D_p v \; f(p) \leq 0,$$

in some neighborhood of p_d, and that it constitutes a *strict Lyapunov function for f* if the inequality is strict [6,7]. The *domain* of v with respect to p_d is the largest neighborhood around p which is free of additional critical points and upon which the derivative is still non-positive.

The existence of a strict Lyapunov function at a point is a sufficient condition for asymptotic stability of that equilibrium state. If a strict Lyapunov function has not been found, asymptotic stability may, nevertheless, be assured if a further condition on the possible limiting set holds. This is "LaSalle's Invariance Principle" [7]. It is possible, as well, to draw conclusions about the tracking capability of a forced dynamical system in consequence of of the stability properties of the unforced vector field at a particular equilibrium state. However, this seems to require the use of a strict Lyapunov function.

It has been known for quite some time that the total energy of a mechanical system may be interpreted as a Lyapunov function [8]. Unfortunately, this choice of Lyapunov function is never strict. The central contribution of this paper rests upon the construction of a strict Lyapunov function for the general class of nonlinear mechanical systems described below, (1). The tracking results follow as a standard consequence.

2.2 Dynamical Equations of Kinematic Chains

The equations of motion of a kinematic chain have been extensively discussed in the robotics literature, and this paper will rely upon the standard rigid body model of an open chain with revolute joints. Thus, we consider a robot to be a particular member of the class of *mechanical systems*,

$$M[q]\ddot{q} + B[\dot{q}, q]\dot{q} + k(q) = \tau \tag{1}$$

where the generalized positions take values in a configuration space, $q \in J$, and M is a positive definite invertible symmetric matrix for all $q \in J$. As shown in the appendix, in the case of kinematic chains, M, the "inertial" terms, B, the "coriolis and centrifugal" terms, and k, the gravitational disturbance vector, all vary in q by polynomials of transcendental functions. It follows that $\nu(M) > 0$ and $\mu(M) < \infty$.

This system may be rewritten in the form

$$\begin{aligned} \dot{q}_1 &= q_2 \\ \dot{q}_2 &= M^{-1}[Bq_2 + k - \tau] \end{aligned} \tag{2}$$

where the generalized positions and velocities take values $p \triangleq \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \in \mathcal{P} \triangleq TJ$ in *phase space* — the tangent bundle over J.

While M, k are always bounded, the coriolis and centripetal forces are quadratic in the velocity — i.e. B is linear in $\dot{q}$ — and, therefore, may become unbounded. It is, however, bounded with respect to q, as the following technical result shows.

Lemma 1 *For any $k \in \mathbb{R}^n$,*

$$k^T B q_2 = q_2^T \hat{M}(k) q_2$$

where

$$\hat{M}(k) \triangleq \begin{bmatrix} k^T D_{q_{11}} M \\ \vdots \\ k^T D_{q_{1n}} M \end{bmatrix} - \frac{1}{2} \sum_{i=1}^{n} \kappa_i D_{q_{1i}} M .$$

is bounded above.

Proof:

$$B q_2 = (q_2 \otimes I)^T D_q M^S \, q_2 - \frac{1}{2} \left[(q_2 \otimes I)^T D_q M^S \right]^T q_2$$

and, hence,

$$\begin{aligned} q_2^T B^T k &= q_2^T \left[D_q M^S \right]^T (q_2 \otimes I) k - \tfrac{1}{2} q_2^T (q_2 \otimes I)^T D_q M^S k \\ &= q_2^T \left[D_q M^S \right]^T \left(k q_2^T \right)^S - \tfrac{1}{2} q_2^T (q_2 \otimes I)^T D_q M^S k \\ &= q_2^T \left(\begin{bmatrix} k^T D_{q_{11}} M \\ \vdots \\ k^T D_{q_{1n}} M \end{bmatrix} - \tfrac{1}{2} \textstyle\sum_{i=1}^{n} \kappa_i D_{q_{1i}} M \right) q_2 . \end{aligned}$$

Since M contains transcendental functions in q, all of its derivatives in q must be bounded.

□

It follows that for some $\hat{\mu} < \infty$,

$$\|\hat{M}(k)\| \leq \hat{\mu} \|k\|. \tag{3}$$

Corollary 2 *For all $p \in \mathcal{P}$,*

$$q_2^T [\frac{1}{2} \dot{M} - B] q_2 \equiv 0.$$

Proof:

$$\begin{aligned} e_2^T [\tfrac{1}{2} \dot{M} - B] e_2 &= e_2^T [\tfrac{1}{2} \dot{M} - \hat{M}(e_2)] e_2 \\ &= \tfrac{1}{2} q_2^T \left[\left[D_q M^S \right]^T (q_2 \otimes I) - (q_2 \otimes I)^T D_q M^S \right] q_2 \\ &\equiv 0. \end{aligned}$$

□

2.3 Stability Properties of "PD" Compensated Systems

Suppose we are presented with the mechanical system (2), and a desired point,

$$p_d \triangleq \begin{bmatrix} q_d \\ 0 \end{bmatrix} \in \mathcal{P}.$$

Choose two positive definite matrices, $K_1, K_2 > 0$, and define the "PD" algorithm

$$\tau = k(q) - K_1 [q_d - q] - K_2 \dot{q}. \tag{4}$$

In terms of the translated "error coordinate system"for $\mathcal{P}$,

$$e(p) \triangleq \begin{bmatrix} q - q_d \\ \dot{q} \end{bmatrix}, \tag{5}$$

the resulting closed loop system has the form

$$\begin{aligned} \dot{e} &= \begin{bmatrix} 0 & I \\ -M^{-1} K_1 & -M^{-1}(B + K_2) \end{bmatrix} e \\ &\triangleq A[\dot{q}, q] e. \end{aligned} \tag{6}$$

Proposition 3 *For all $\gamma_0 > 0$,*

$$\tilde{v}(e) = \frac{1}{2}e^T\tilde{P}(q)e \triangleq \frac{1}{2}e^T\begin{bmatrix} \gamma_0 K_1 & 0 \\ 0 & \gamma_0 M(q) \end{bmatrix}e$$

is a Lyapunov function for the closed loop system (6).

Proof: It is clear that $\tilde{v}$ is positive definite at the origin of the error system. Taking the time derivatives along the solutions of the closed loop system, (6),

$$\begin{aligned} \dot{\tilde{v}} &= \tfrac{1}{2}e^T[\tilde{P}A + A^T\tilde{P} + \dot{\tilde{P}}]e \\ &= -e^T\begin{bmatrix} 0 & 0 \\ 0 & \gamma_0 K_2 \end{bmatrix}e + \gamma_0 e_2^T[\tfrac{1}{2}\dot{M} - B]e_2. \end{aligned}$$

Noting that $e_2 \equiv q_2$, it follows from Corollary 2 , that the second term is identically zero.

□

There follows the desirable result that proportional and derivative linear state feedback stabilizes a mechanical system, after the gravitational disturbance torques have been removed.

Theorem 1 ([9,10,11]) *The origin of the closed loop error coordinate system (6) is asymptotically stable.*

Proof: The existence of a Lyapunov Function, $\tilde{v}$, assures stability. According to LaSalle's invariance principle, the attracting set is the largest invariant set contained in $\{(e_1, e_2) \in \mathcal{P} : \dot{v} \equiv 0\}$, which, evidently, is the origin, since the vector field is oriented away from $\{e_2 \equiv 0\}$ everywhere else on that hyperplane.

□

Notice that the proof of attractivity requires an appeal to LaSalle's invariance principle in consequence of the fact that $\tilde{v}$ is not a strict Lyapunov function. In order to obtain the desired extension to tracking problems it is necessary to construct one. Unfortunately, the constructions devised to date require the artificial limitation to decoupled PD feedback. Namely, in the sequel, it will be assumed that the gain matrices of (6) are specified as

$$K_1 \triangleq \omega^2 I; \qquad K_2 \triangleq 2\varsigma\omega I \tag{7}$$

given two positive real numbers, ω, ς.

2.4 A Strict Lyapunov Function for Nonlinear Mechanical Systems

The following technical lemma will be of use in the main result, below.

Lemma 4 *For $M(q)$ as in (1) and any positive scalars, $\alpha, \beta, \gamma \in \mathbb{R}^+$,*

$$\inf_{\|e\|=1} e^T\begin{bmatrix} \alpha I & \beta I \\ \beta I & \gamma M(q) \end{bmatrix}e \geq \nu(K)$$

where

$$K \triangleq \begin{bmatrix} \alpha & \beta \\ \beta & \gamma\nu(M) \end{bmatrix}.$$

In particular, the matrix is positive definite when

$$\alpha\gamma\nu(M) > \beta^2 \tag{8}$$

Proof: Since

$$\begin{bmatrix} \alpha I & \beta I \\ \beta I & \gamma M(q) \end{bmatrix} > \begin{bmatrix} \alpha I & \beta I \\ \beta I & \gamma\nu(M) \end{bmatrix} = K \otimes I,$$

it will suffice to show that

$$\nu(K \otimes I) = \nu(K).$$

This follows since all eigenvalues of $K \otimes I$ are eigenvalues of K, according to Lemma 12 in the appendix. The particular conclusion obtains by taking the determinant of K.

□

Proposition 5 *For all $p_d \in \mathcal{P}$ and $\omega, \varsigma > 0$, given any bounded set, $\mathcal{B} \subset \mathcal{P}$, containing p_d there exists a scalar $\gamma_0 > 0$ such that*

$$v(e) \triangleq \frac{1}{2}e^T P(q)e = \frac{1}{2}e^T\begin{bmatrix} \omega^2\gamma_0 I & \omega\varsigma I \\ \omega\varsigma I & \gamma_0 M(q) \end{bmatrix}e$$

is a strict Lyapunov Function for the closed loop system, (6) on the domain $\mathcal{B}$, assuming the decoupled feedback gain matrices specified in (7).

Proof: Letting

$$\beta \triangleq \sup_{e \in \mathcal{B}} \|e\|,$$

find some γ_0 satisfying

$$\gamma_0 > \max\left\{\frac{\varsigma}{\sqrt{\nu(M)}}, \frac{1}{\nu(M)}, \frac{\mu(\hat{M})\beta}{\nu(M)} + 1\right\}. \qquad (9)$$

According to Lemma 4 and the inequality involving the first entry of the inferior set in (9), it follows that P is a positive definite matrix for all $q \in J$, hence v, is positive defnite at p_d.

Taking time derivatives along the solutions of system (6), we have

$$\dot{v} \;= \tfrac{1}{2}e^T[PA + A^TP + \dot{P}]e,$$

which may be expanded as

$$\begin{aligned}\dot{v} \;&= -\omega\varsigma e^T \begin{bmatrix} \omega^2 M^{-1} & \omega M^{-1} \\ \omega M^{-1} & \gamma_0 I \end{bmatrix} e \\ &\quad -\omega\varsigma(\gamma_0 - 1)e_2^Te_2 - \omega\varsigma e_1^T M^{-1} B e_2 \\ &\quad +\gamma_0 e_2^T[\tfrac{1}{2}\dot{M} - B]e_2.\end{aligned}$$

The term in the last line vanishes according to Corollary 2 . Moreover, the block matrix in the first line is positive definite according to the inequality (9) and the result of Lemma 4 since

$$\begin{bmatrix} \omega^2 M^{-1} & \omega M^{-1} \\ \omega M^{-1} & \gamma_0 I \end{bmatrix} = \begin{bmatrix} M^{-\frac{1}{2}} & 0 \\ 0 & M^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \omega^2 I & \omega I \\ \omega I & \gamma_0 M \end{bmatrix} \begin{bmatrix} M^{-\frac{1}{2}} & 0 \\ 0 & M^{-\frac{1}{2}} \end{bmatrix}.$$

Finally, according to Lemma 1 , the term in the middle may be rewritten as

$$\omega\varsigma[(\gamma_0 - 1)e_2^Te_2 + e_1^TM^{-1}Be_2] = \omega\varsigma e_2^T[(\gamma_0 - 1)I + \hat{M}(M^{-1}e_1)]e_2 > 0,$$

where $k \triangleq M^{-1}e_1$, and the result follows from the inequality involving the last entry of the set in (9).

□

3 Consequences for Tracking Unknown Reference Signals

Now consider the decoupled "PD" compensated system forced by a continuously differentiable reference signal, $q_d(t)$,

$$\tau = k(q) - \omega^2[q_d(t) - q] - 2\omega\varsigma\dot{q}. \qquad (10)$$

Assume that the reference trajectory is "unpredictable" — i.e. its first and second derivatives are unknown — but there is available an à priori bound on the maximum rate of change,

$$\|\dot{q}_d\| \leq \rho_0.$$

Notice that the forced closed loop system may be written in the same error coordinates as (5), above,

$$\dot{e} = A[\dot{q}, q]e + d, \qquad (11)$$

where $d \triangleq \begin{bmatrix} \dot{q}_d(t) \\ 0 \end{bmatrix}$, is a "disturbance" input due to the unknown but non-zero reference derivative.

Theorem 2 *The closed loop "disturbed" error system (11) has bounded trajectories which asymptotically approach the set*

$$\left\{e \in \mathcal{P} : \|e\| \leq \frac{\mu(M)}{\nu(K)} 2\rho_0 \sqrt{(\gamma_0/\varsigma)^2 + 1/\omega^2}\right\}$$

where

$$K \triangleq \begin{bmatrix} \omega & 1 \\ 1 & \gamma_0\nu(M)/\omega \end{bmatrix}.$$

Proof: We have

$$\begin{aligned}\dot{v} \;&= \tfrac{1}{2}e^T[PA + A^TP + \dot{P}]e + e^TPd, \\ &\leq -\omega^2\varsigma e^T\left(\tfrac{1}{\mu(M)}K \otimes I\right)e + \omega^2\varsigma \begin{bmatrix} \gamma_0 e_1^T\dot{q}_d/\varsigma \\ e_2^T\dot{q}_d/\omega \end{bmatrix}. \\ &\leq -\tfrac{\omega^2\varsigma}{2\mu(M)}\|e\|\left[\|e\|\nu(K) - 2\rho_0\mu(M)\sqrt{(\gamma_0/\varsigma)^2 + 1/\omega^2}\right].\end{aligned}$$

This is negative whenever e is outside the set indicated in the statement of the theorem.

□

Corollary 6 *The asymptotic tracking bound may be made arbitrarily small by increasing the magnitudes of the feedback gains in (10)*

Proof: For a sufficiently large value of ω it is possible to choose two real numbers $\kappa_1, \kappa_2 \in (0,1)$ such that

$$\varsigma = \kappa_2\gamma_0\sqrt{\nu(M)} \qquad ; \gamma_0 = \kappa_1\omega$$

and the inequality (9) still holds. Using these definitions and the results of the theorem, the attracting region is bounded by the magnitude

$$\frac{\mu(M)}{\nu(K)} 2\rho_0\sqrt{(\kappa_2^2/\nu(M)) + 1/\omega^2},$$

Note that

$$\begin{aligned}\nu(K) &= \omega + \gamma_0\nu(M)/\omega - \sqrt{(\omega - \gamma_0\nu(M)/\omega)^2 + 4} \\ &= \omega + \kappa_1\nu(M) - \sqrt{(\omega - \kappa_1\nu(M))^2 + 4},\end{aligned}$$

hence,

$$\frac{d\nu(K)}{d\omega} = 1 - \frac{\omega - \kappa_1\nu(M)}{\sqrt{(\omega - \kappa_1\nu(M))^2 + 4}} > 0$$

and $\nu(K)$ is bounded from below as ω increases. Since κ_2 may be made as small as desired without violating (9), the result follows.

□

A The Stack Representation

If $A \in \mathbf{R}^{n\times m}$, the "stack" representation of $A \in \mathbf{R}^{nm}$ formed by stacking each column below the previous will be denoted A^S [12].

If $B \in \mathbf{R}^{p\times q}$, and A is as above then the *kronecker product* of A and B is

$$A \otimes B \triangleq \begin{bmatrix} a_{11}B & \dots & a_{1m}B \\ a_{21}B & \dots & a_{2m}B \\ & \vdots & \\ a_{n1}B & \dots & a_{nm}B \end{bmatrix} \in \mathbf{R}^{np\times mq}.$$

The kronecker product is not, in general, commutative. Note that while the transpose "distributes" over kronecker products,

$$(A \otimes B)^T = (A^T \otimes B^T),$$

the stack operator, in general, does not.

Lemma 7 *If $A \in \mathbf{R}^{n\times m}$ then there exists a nonsingular linear transformation of $\mathbf{R}^{nm}$, T, such that*

$$\left(A^T\right)^S = TA^S$$

Proof: For $p = nm$, let $\mathcal{B} \triangleq \{b_1, ..., b_p\}$ denote the canonical basis of $\mathbf{R}^p$ — i.e., b_i is a column of p entries with a single entry, 1, in position i, and the other $p-1$ entries set equal to zero. The transpose operator is a reordering of the canonical basis elements, hence may be represented by the elementary matrix,

$$T \triangleq \left[b_1, b_{n+1}, b_{2n+1}, ..., b_{(m-1)n+1}, b_2, b_{n+2}, b_{2n+2}, ..., b_{(m-1)n+2}, ...b_n, b_{2n}, b_{3n}, ..., b_{mn}\right].$$

□

For $n = m$, if we define $P_+ \triangleq I + T$, $P_- \triangleq I - T$ then both operators are projections onto the set of "skew-symmetric" , "symmetric" operators of $\mathbf{R}^n$, repsectively, since $P_\pm^2 = P_\pm$. Note that Ker $P_\pm$ = Im $P_\mp$.

The kronecker product does "distribute" over ordinary matrix multiplication in the appropriate fashion.

Lemma 8 *If* $A \in \mathbf{R}^{n\times m}, B \in \mathbf{R}^{p\times q}, C \in \mathbf{R}^{m\times k}, D \in \mathbf{R}^{q\times l}$ *then*

$$(A \otimes B)(C \otimes D) = (AC \otimes BD).$$

Lemma 9 ([12]) *If* $B \in \mathbf{R}^{m\times p}$, $A \in \mathbf{R}^{n\times m}$, *and* $C \in \mathbf{R}^{p\times q}$ *then*

$$[ABC]^S = (C^T \otimes A)B^S.$$

Noting that for any column, $c \in \mathbf{R}^{p\times 1}$, we have

$$c^S = \left[(c)^T\right]^S = c,$$

there follows the corollary

Corollary 10 *If* $B \in \mathcal{R}^{m\times p}$, $c \in \mathbf{R}^p$ *then*

$$\begin{aligned} Bc &= Bc^S = (c^T \otimes I)B^S \\ &= \left([Bc]^T\right)^S = \left(c^T B^T\right)^S = (I \otimes c^T)\left(B^T\right)^S. \end{aligned}$$

,

Noting, moreover, that

$$tr\ \{A\} = \left(I^S\right)^T A^S,$$

there follows the additional result

Corollary 11 *If* $A \in \mathbf{R}^{n\times m}, B \in \mathbf{R}^{p\times m}$ *then*

$$tr\ \left\{AB^T\right\} = \left(A^S\right)^T B^S.$$

Proof:

$$\begin{aligned} tr\ \left\{AB^T\right\} &= \left(I^S\right)^T \left(AB^T\right)^S \\ &= \left(I^S\right)^T (B \otimes I)A^S \\ &= \left(A^S\right)^T (B^T \otimes I)I^S \\ &= \left(A^S\right)^T B^S. \end{aligned}$$

□

Lemma 12 *For any square array,* $A \in \mathbf{R}^{n\times n}$, *if* I_m *is the identity on* $\mathbf{R}^m$ *then the spectrum of* $(A \otimes I_m)$ *is contained in the spectrum of* A.

Proof: Suppose λ is an eigenvalue of $(A \otimes I_m)$. There must be some non-zero vector, $x \in \mathbf{R}^{mn}$ in the kernel of $\lambda(I_n \otimes I_m) - (A \otimes I)$ Since $x = X^S \in \mathbf{R}^{n\times m}$, it follows that

$$\begin{aligned} 0 &= [\lambda(I_n \otimes I_m) - (A \otimes I)]x \\ &= \left[\lambda X - XA^T\right]^S \\ &= \left[X(\lambda I_n - A^T)\right]^S. \end{aligned}$$

This implies that $\text{Im}\, X^T \subset \text{Ker}\ \lambda I_n - A$, and since the former subspace has dimension at least 1 (according to the assumption that $X \not\equiv 0$), the latter must as well. Thus, λ is an eigenvalue of A.

□

B General Robot Arm Dynamics

The rigid body model of robot arm dynamics may be most quickly derived by appeal to the lagrangian formulation of Newton's Equations. If a scalar function, termed a *lagrangian*, $\lambda = \kappa - \upsilon$, is defined as the difference between total kinetic energy, κ, and total potential energy, υ, in a system, then the equations of motion obtain from

$$\frac{d}{dt}D_{\dot{q}}\lambda - D_{q}\lambda = \tau^{\mathsf{T}},$$

where τ is a vector of external torques and forces [13,14].

First consider the kinetic energy contributed by a small volume of mass δm_i at position p in link $\mathcal{L}_i$,.

$$\delta\kappa_i = \frac{1}{2}{}^{0}\dot{p}_i^{\mathsf{T}}{}^{0}\dot{\mathrm{p}}_{\mathrm{i}}\delta \mathrm{m}_{\mathrm{i}}$$

where ${}^{0}p_i = {}^{0}F_i\,{}^{i}p$ is the matrix representation of the position p in the base frame of reference, ${}^{0}F_i$ is the matrix representation of the frame of reference of link $\mathcal{L}_i$ in the base frame, and ${}^{i}p$ is the matrix representation of the point in the link frame of reference, and, hence, [1]

$$\dot{p}_i = \dot{F}_i\,{}^{i}p,$$

since the position in the body is independent of the generalized coordinates. The total kinetic energy contributed by this link may now be written

$$\begin{aligned}
\kappa_i &= \textstyle\int_{\mathcal{L}_i} \frac{1}{2}\left[\dot{F}_i\,{}^{i}p\right]^{\mathsf{T}}\dot{F}_i\,{}^{i}p\,dm_i \\
&= \textstyle\int_{\mathcal{L}_i} \frac{1}{2}\,trace\{\dot{F}_i\,{}^{i}p\left[\dot{F}_i\,{}^{i}p\right]^{\mathsf{T}}\}dm_i, \\
&= \tfrac{1}{2}\,trace\{\dot{F}_i \textstyle\int_{\mathcal{L}_i}^{i} p^{i}p^{\mathsf{T}}dm_i\left[\dot{F}_i\right]^{\mathsf{T}}\} \\
&= \tfrac{1}{2}\,trace\,\{\dot{F}_i\overline{P}_i\dot{F}_i^{\mathsf{T}}\},
\end{aligned}$$

(since the frame matrix is constant over the integration), where $\overline{P}_i$ is a symmetric matrix of *dynamical parameters* for the link. Explicitly, if the link has mass, $\overline{\mu_i}$, center of gravity (in the local link coordinate system) $\overline{p_i}$, and inertia matrix, $\overline{J_i}$, then

$$\overline{P}_i \triangleq \begin{bmatrix} \overline{J_i} & \overline{\mu_i p_i} \\ \overline{\mu_i p_i}^{\mathsf{T}} & \overline{\mu_i} \end{bmatrix}.$$

Passing to the *stack representation* (refer to Appendix A)

$$\begin{aligned}
2\kappa_i &= trace\,\{\dot{F}_i\overline{P}_i\dot{F}_i^{\mathsf{T}}\} \\
&= \left[(\dot{F}_i\overline{P}_i)^{S}\right]^{\mathsf{T}}\dot{F}_i^{S} \\
&= \left[(\overline{P}_i\otimes I)^{\mathsf{T}}\dot{F}_i^{S}\right]^{\mathsf{T}}\dot{F}_i^{S} \\
&= \left[\dot{F}_i^{S}\right]^{\mathsf{T}}\tilde{P}_i\dot{F}_i^{S} \\
&= \left[(D_q F_i^{S})\dot{q}\right]^{\mathsf{T}}\tilde{P}_i(D_q F_i^{S})\dot{q} \\
&= \dot{q}^{\mathsf{T}}M_i\dot{q},
\end{aligned}$$

where we have implicitly defined

$$M_i(q) \triangleq \left[D_q F_i^{S}\right]^{\mathsf{T}}\tilde{P}_i D_q F_i^{S}; \qquad \tilde{P}_i \triangleq \overline{P}_i^{\mathsf{T}}\otimes I.$$

It follows that the total kinetic energy of the entire chain is given as

$$\kappa = \frac{1}{2}\dot{q}^{\mathsf{T}}M(q)\dot{q}; \qquad M(q) \triangleq \sum_{i=1}^{n} M_i(q).$$

The potential energy contributed by δm_i in $\mathcal{L}_i$ is

$$\delta\upsilon_i = z_0^{\mathsf{T}}F_i\,{}^{i}pg\delta m_i$$

where g is the acceleration of gravity, hence the potential energy contributed by the entire link is

$$\upsilon_i = z_0^{\mathsf{T}}F_i\int_{\lambda_i}{}^{i}pg\,dm_i = z_0^{\mathsf{T}}F_i\overline{p_i}g,$$

and $\upsilon = \sum_{i=1}^{n} z_C^{\mathsf{T}}F_i\overline{p_i}g$. [2]

To proceed with the computation, note that $D_{\dot{q}}\lambda = D_{\dot{q}}\kappa = \dot{q}^{\mathsf{T}}M(q)$, hence,

$$\frac{d}{dt}D_{\dot{q}}\lambda = \ddot{q}^{\mathsf{T}}M(q) + \dot{q}^{\mathsf{T}}\dot{M}(q).$$

[1] We will omit the prior superscript, 0, when it is clear the the coordinate system of reference is the base

[2] Assume that z_0 "points up" in a direction opposing the gravitational field.

Moreover,

$$\begin{aligned} D_q \kappa &= \tfrac{1}{2}\dot{q}^T D_q[M(q)\dot{q}] \\ &= \tfrac{1}{2}\dot{q}^T[\dot{q}^T \otimes I]D_q M^s , \end{aligned}$$

hence,if all terms from Lagrange's equation involving the generalized velocity are collected, we may express them in the form $\dot{q}^T B^T$, where

$$B(q,\dot{q})^T \triangleq \dot{M}(q) - \frac{1}{2}[\dot{q}^T \otimes I]D_q M^s .$$

Finally, by defining $k(q) \triangleq [D_q v]^T$, Lagrange's equation may be written in the form (1)

$$M(q)\ddot{q} + B(q,\dot{q})\dot{q} + k(q) = \tau.$$

M, called the "inertia" matrix, may be shown to be positive definite over the entire workspace as well as bounded from above since it contains only polynomials involving transcendental functions of q. B contains terms arising from "coriolis" and "centripetal" forces, hence is linear in $\dot{q}$ (these forces are quadratic in the generalized velocities), and bounded in q, since it involves only polynomials of transcendental functions in the generalized position. Finally, k arises from gravitational forces, is bounded, and may be observed to have much simpler structure (still polynomial in transcendental terms involving q) than the other expressions. An important study of the form of these terms was conducted by Bejczy [15].

References

[1] E. Freund. Fast nonlinear control with arbitrary pole placement for industrial robots and manipulators. *The International Journal of Robotics Research*, 1(1):65–78, 1983.

[2] J. Y. S. Luh, M. W. Walker, and R. P. Paul. Resolved acceleration control of mechanical manipulators. *IEEE Transaction on Automatic Control*, AC-25:468–474, 1980.

[3] T. J. Tarn, A. K. Bejcsy, A. Isidori, and Y. Chen. Nonlinear feedback in robot arm control. In *Proc. 23rd IEEE Conference on Decision and Control*, pages 736–751, Las Vegas, Nev., Dec 1984.

[4] Jean-Jaques E. Slotine. The robust control of robot manipulators. *The International Journal of Robotics Research*, 4(2):49–64, Summer 1985.

[5] In Joong Ha and Elmer G. Gilbert. Robust tracking in nonlinear systems and it application to robotics. In *Proc. 24th IEEE Conference on Decision and Control*, pages 1009–1017, IEEE, Ft. Lauderdale, Fla., 1985.

[6] Morris W. Hirsch and Stephen Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, Inc., Orlando, Fla., 1974.

[7] J. P. Lasalle. *The Stability of Dynamical Systems.* Volume 25 of *Regional Conference Series in Applied Mathematics*, SIAM, Philadelphia, PA, 1976.

[8] Sir W. Thompson and P. G. Tait. *Treatise on Natural Philosophy.* University of Cambridge Press, 1886, Cambridge.

[9] Morikazu Takegaki and Suguru Arimoto. A new feedback method for dynamic control of manipulators. *ASME Journal of Dynamics Systems,Measurement, and Control*, 102:119–125, 1981.

[10] Daniel E. Koditschek. Natural motion for robot arms. In *IEEE Proceedings 23rd Conference on Decision and Control*, pages 733–735, Las Vegas, Dec 1984.

[11] A. J. Van Der Schaft. *Stabilization of Hamiltonian Systems.* Memo 470, Technische Hogeschool Twente, Twente, Netherlands, Jan 1985.

[12] R. Bellman. *Introduction to Matrix Analysis.* McGraw Hill, New York, 1965.

[13] V. I. Arnold. *Mathematical Methods of Classical Mechanics.* Springer-Verlag, N.Y., 1978.

[14] H. Goldstein. *Classical Mechanics.* Addison-Wesley, Reading, MA, 1980.

[15] Antal K. Bejczy. *Robot Arm Dynamics and Control.* Technical Report 33-699, Jet Propulsion Laboratory, Pasadena, CA, 1974.

Concept Development of a Tendon Arm Manipulator and Anthropomorphic Robotic Hand

C.T. Tolman
AMETEK
Santa Barbara, CA 93160

1. Abstract

~~This paper summarizes~~ AMETEK/ORED inhouse research and development efforts leading toward a "next-generation" robotic manipulator arm and end-effector technology is summarized. Manipulator arm development has been directed toward a multiple-degree-of-freedom, flexible, tendon-driven concept which we refer to as a Tendon Arm Manipulator (TAM). End-effector development has been directed toward a three-fingered, dextrous, tendon-driven, anthropomorphic configuration which ~~we refer~~ is Referred to as an Anthropomorphic Robotic Hand (ARH). Key technology issues are identified for both concepts.

2. Introduction

The background, rationale, and requirements for a next-generation manipulator arm and end-effector are noted in order establish the foundational assumptions upon which the inhouse R&D program is based. In order to relate the context for development, this background includes a brief synopsis of the projected telerobotics evolutionary path which AMETEK/ORED has advocated since its inception in 1977.

Over the past five years, AMETEK/ORED has pursued concept development of a Tendon Arm Manipulator (TAM) through a low-level-of-effort inhouse R&D program. This development has included three conceptual design configurations and two limited engineering development models. Results of this program to date are summarized. The latest TAM design configuration is illustrated and discussed, including performance design goals. Technical issues and enabling technology development are noted.

The original R&D for the TAM included some preliminary work on a dextrous three-fingered end-effector concept. About two years ago, in response to a planned NASA program, this work was formulated into a concept design for an Anthropomorphic Robotic Hand (ARH). The concept was further refined and some preliminary design performed in response to the proposed DARPA Advanced Robotic Manipulator program. The baseline ARH concept design is illustrated and described. Technology issues and key enabling technology development are summarized.

3. Background

The first robotic manipulator arm and end-effector was adapted to a subsea remotely operated vehicle (ROV) in 1961. Over the succeeding 25 years, increasingly capable manipulators have been designed and applied to subsea ROV's; in general, control of these manipulators has been limited to master-slave teleoperation, but has included bilateral force feedback on the more sophisticated systems. AMETEK has been involved in this applications arena for many years.

In 1979, AMETEK/ORED initiated an inhouse study program to forecast next-generation manipulator technology.

Technology advancement of articulated (revolute-coordinate) manipulator arms appeared to be well covered, but we identified operations in unstructured subsea environments, e.g., around wellheads, where articulated arms were severely constrained in accessibility. In these cases, what was needed was a flexible "snake-like" multiple degree-of-freedom (DOF) configuration to work through and around a maze of obstructions. This need was not being addressed, and thus became a goal for further inhouse work. For end-effectors, other than specialized, task-specific end-effectors and tools, there appeared to be a driving need for a general, dextrous end-effector with kinesthetic and haptic capabilities approaching that of the human operator. We elected to parallel research on dextrous end-effectors along with our manipulator arm research. Progress to date on both the manipulator arm and end-effector is summarized in Section 4.

In order to establish a context for this program, it is useful to briefly note the differences in orientation and approach between robotics technology development directed toward teleoperation and that aimed for autonomous applications. The issues, particularly with respect to control, are significantly different. The most notable difference is in the nature of the pacing robotics technologies: for autonomous operation, higher levels of control [1] are the pacing item, and mechanical systems with dextrous capabilities cannot be fully utilized as yet; under teleoperation, the operator provides higher level control, so highly-capable mechanical systems can more readily be utilized. Hence the motivation to prioritize such advanced mechanical systems development is greater for teleoperation.

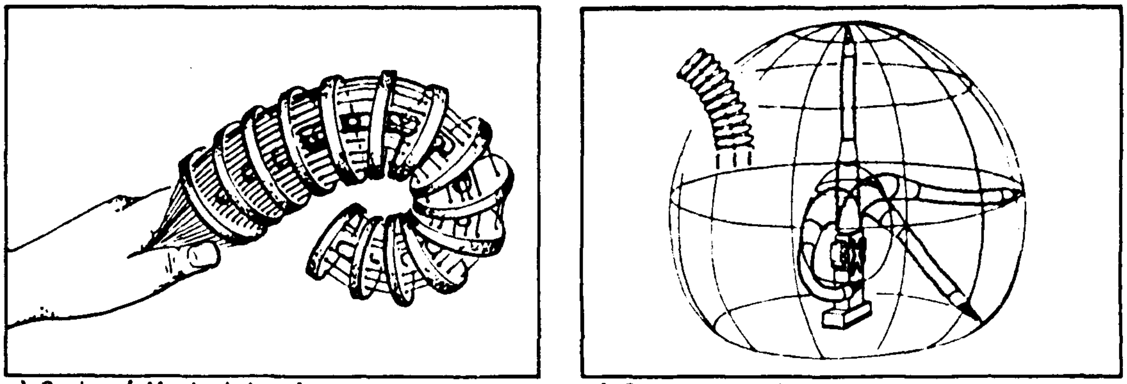
AMETEK/ORED advocates a view of the evolution of robotics technologies from teleoperation toward fully-autonomous systems, through progressive implementation of supervised autonomous modes of operation, as sensing, control, and computational technologies mature. An informative technical paper on this subject was written by J. Vertut, Manager of the Advanced Teleoperation Program in French Advanced Robotics and Automation project [2]. Our views were expressed by AMETEK/ORED General Manager Jack Stone in his article in ROV Magazine [3]. A more exhaustive treatment, including specific examples for space telerobotics, was provided recently by NASA/Montemerlo [4].

4. Concept Development: TAM and ARH

AMETEK/ORED has separated the inhouse IR&D program into two related concept development initiatives, the Tendon Arm Manipulator (TAM) and the Anthropomorphic Robotic Hand (ARH). The TAM concept is discussed first, followed by a discussion of the ARH concept.

Tendon Arm Manipulator (TAM):

Inspiration for the TAM concept originated with Tensor Arm Manipulator Design (Figure 1a) by the Scripps Institution of Oceanography [5]. We also examined with interest the Spine manipulator arm (Figure 1b) developed by Spine Robotics [6].



a) Scripps' Manipulator Arm

b) Spine Manipulator Arm

Figure 1, Flexible Manipulator Arm Configurations

Both of these configurations utilize a number of jointed discs, the planes of which can be rotated in two dimensions with respect to one another. Each disc is driven by four tendons, two for each degree of freedom. Thus the arm has a maximum of 2(n-1) DOF, where n represents the number of discs (the first disc is fixed to the base or world frame, while the last disc serves as the base plate for the wrist). The number of independent DOF can be reduced, as desired, by establishing an angular relationship between disc rotations, e.g., the two sections of the Spine arm form only circular arcs of varying radius, so that the arm has only four independent DOF.

The original TAM design configuration (Figure 2) resembled the Scripps design because of its adaptability to multiple DOF and more arbitrary shapes. Four joint configurations were considered, varying the relative placement and connection of the joint with respect to the discs. A simple engineering model was built in order to duplicate some of the results of the Scripps work. As a follow-on, a larger engineering model was constructed, specifically to emperically examine loading of tendons and joints and instabilities.

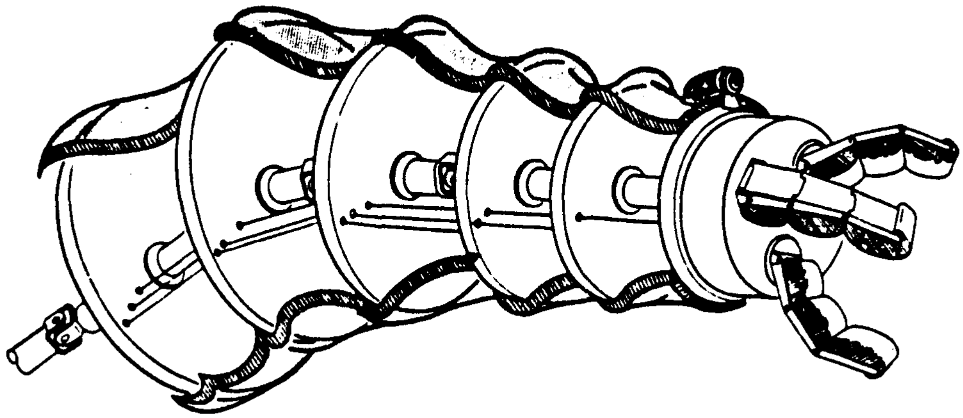


Figure 2, AMETEK/ORED TAM, Original Design Concept

Two major deficiencies of the basic Scripps configuration were confirmed: (1) a buckling instability (also noted by Scripps) between discs; and (2) high torsional loading of the joints under certain loading conditions.

The current TAM design, illustrated in Figure 3, draws on the latest advances in "Serpentine Arm" technology, summarized in a recent Intelligent Task Automation report [7], and addresses and corrects the deficiencies exhibited by the TAM engineering models.

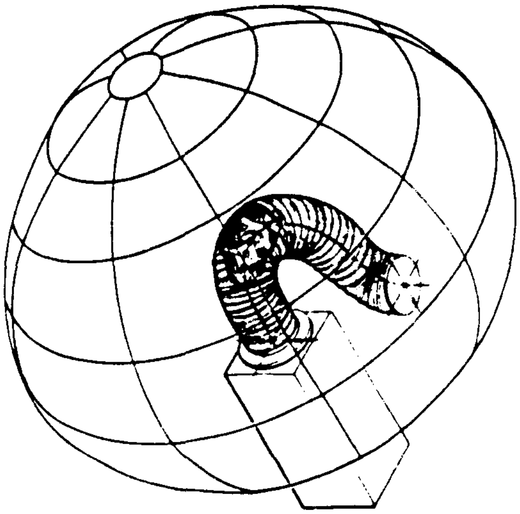


Figure 3, AMETEK/ORED Tendon Arm Manipulator (TAM)
Baseline Concept Design

In order to eliminate the buckling instability, sheathed cables are used for the tendons, each sheath terminating at the disc preceding that being displaced by the tendon; this makes each displacement determinate and precludes the buckling exhibited by the previous TAM models. To reduce the high stresses generated by torsion, the joints were reconfigured to form large-diameter double-gimballed rings; this not only increases the effective radius for reacting torsional moments but also provides a convenient center-arm space for routing of actuation cables.

Performance goals for the baseline TAM design include the following:

- Length: 36" from shoulder to wrist base plates.
- Weight: 20 lbs incl structure and tendons.
- Payload: 50 lbs excl wrist and end-effector.
- Speed: 180 degrees/sec, 1/4-load (all joints).
- Accuracy: 0.050" or better.
- Operational Envelope: approximately hemispherical.

The current TAM baseline design, with each joint limited to ± 30 degrees angular deflection (as our research has indicated is a practical deflection upper limit for a tendon-driven configuration), requires nine segments to achieve a hemispherical operational envelope -- actually somewhat more than hemispherical as shown in Figure 3, closely corresponding to an optimal operational envelope for an articulated manipulator arm.

Obviously, given the current state of the art, control of such an arm, with up to 18 DOF for the baseline design, is a major issue. We have generated control scenarios, however, to account for this limitation:

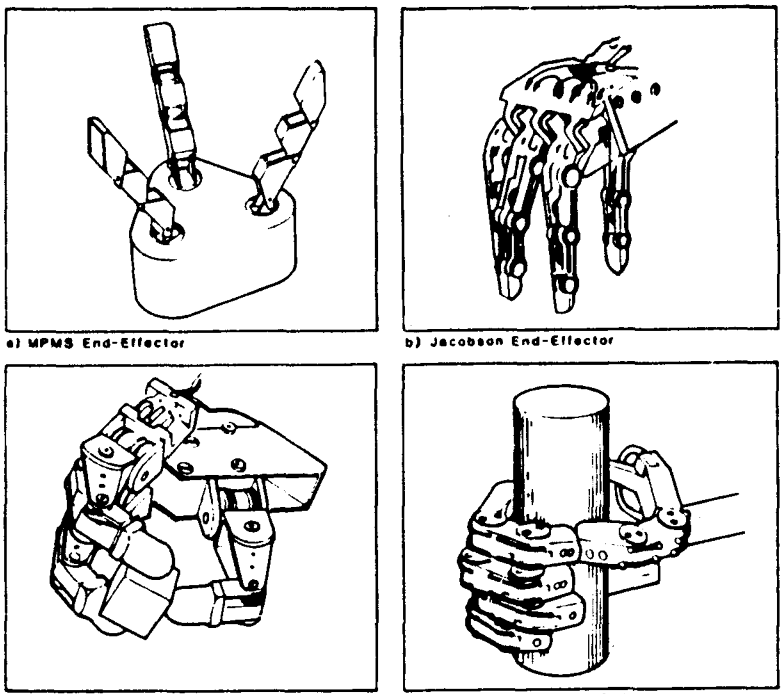
<u>For teleoperation</u>, each joint can be servo-controlled with a scaled replica master, with which the operator "shapes" the spacially-correspondent TAM. If, after positioning the arm at a work site the operator subsequently displaces the master arm such that the TAM contacts some obstruction, the bilateral control system compliantly reshapes the TAM around the obstruction and simultaneously conforms the master to the new shape. This represents a simple extrapolation of current technology.

<u>For autonomous operation</u>, the TAM can be limited in independent DOF by controlling groups of discs in a relational manner, as with the Spine arm. Such grouping may be accomplished mechanically or electronically. Initially, as few as four independent DOF may be used (determinate), with increasing DOF and shape capabilities implemented as sensing, control, and computational technology advances. Ultimately, with the control loop closed around the end-point through sophisticated sensing and control, and with control strategies for indeterminate arm configurations (e.g., world modeling with sensing updates and spacial distribution of allowable arm shapes and trajectories within the world model), the TAM should be able to achieve accuracies and capabilities rivaling articulated arms.

<u>Anthropomorphic Robotic Hand (ARH):</u>

Much relevant work has been done over the past thirty years on dextrous end-effectors. For the first twenty years, this work was almost exclusively in the area of prosthetic devices. An interesting example is the Belgrade hand [8]. Over the past ten years or so, there has been considerable interest and effort directed toward dextrous end-effectors suitable for robotic (autonomous) or mixed-mode (teleoperation/autonomous) applications. "Teleoperation", in this case, includes close-coupled prosthetic applications. The previously-noted report for the Intelligent Task Automation program [7] includes a comprehensive summary of dextrous end-effectors.

AMETEK/ORED's initial work on a dextrous end-effector concept for the TAM focussed on the Multiple Prehension Manipulator System (MPMS) [9] design circa 1974. This hand, illustrated in Figure 4a, has three fingers, each with base rotation and link curl (total of six independent DOF). It is able to simulate all six prehensile modes of the human (as defined in the referenced article), but is not anthropomorphic.



a) MPMS End-Effector

b) Jacobson End-Effector

c) Salisbury End-Effector

d) Caporali/Shahinpoor End-Effector

Figure 4, Dextrous End-Effector Designs

Using the prehensile modes analysis, along with an analysis of the configurations of existing dextrous end-effector designs, notably the Jacobsen [10], Salisbury [11], and Caporali/Shahinpoor [12] hands (illustrated in Figure 4), we derived a unique design, specifically directed toward anthropomorphicity and simplicity. Because of our orientation toward teleoperation, we gave anthropomorphicity a high priority. AMETEK/ORED designated this design concept the Anthropomorphic Robotic Hand (ARH).

The baseline ARH design concept, as illustrated in Figure 5, utilizes three fingers (configured as a thumb and two fingers) and a fixed palm. In order to directly mimic the grasping modes of the human hand, of particlar advantage for teleoperation, the thumb has the capability to rotate from opposition with the fingers to planarity with the palm. In addition, the digit joints of the thumb independently rotate to curl the thumb as does a human thumb. Each of the two fingers has three joints; the knuckle and middle joints have independent rotation, while the end joint rotation is ratioed to the rotation of the middle joint (approximately 2:1). No lateral rotation is provided for the knuckles of the two fingers, but the base rotational axes are oriented with such that the tips of the fingers converge during curl to meet at contact with the palm.

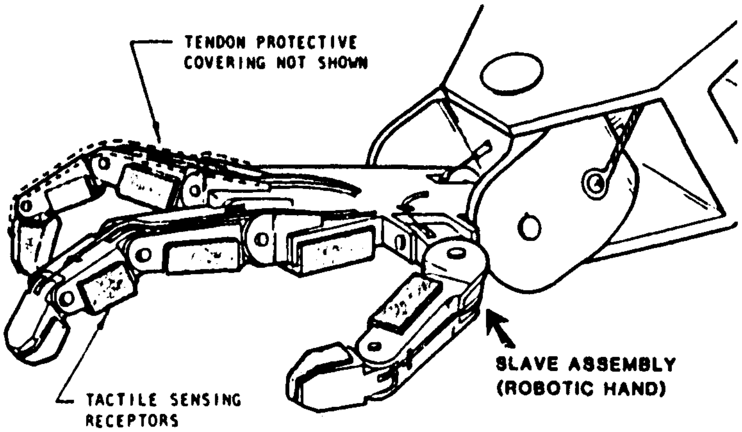


Figure 5, AMETEK/ORED Anthropomorphic Robotic Hand (ARH)
Base'ine Concept Design

Thus, the ARH baseline configuration has a minimum number of independent DOF (seven, as compared to nine for the Salisbury hand and sixteen for the Jacobsen hand), and is able to achieve all the prehensile modes of the human in a direct anthropomorphic manner.

The ability of the thumb to rotate to the plane of the palm uniquely provides a hook grasping mode in an anthropomorphic manner. Spherical and cylindrical grasp and closure are provided with thumb opposition and coordinated curl of thumb and fingers. Direct pinch with both or either finger tip(s) is enabled by proper rotation and curl of the thumb with respect to the curl of each or both of the fingers, and coordination of these movements will allow pinch transfer. Finally, lateral pinch is enabled by rotating the thumb midway and closing onto the finger.

Key technology issues in the areas of actuation, sensing, and control are, in general, being addressed through ongoing research throughout the community. Most of the critical elements currently exist commercially or are near transition from the laboratory. A complete review of the ARH baseline design is beyond the scope of this paper, but some of the key technology issues for both the TAM and ARH are noted in Section 5.

5. Technology Development

Preliminary design and development of the TAM and ARH have included research on pacing technologies, actuation, sensing, and control. Key issues, results, and recommendations follow:

Actuation for both the TAM and ARH is provided from actuator mechanisms located in the base through sheathed cable tendons. The actuators could be electrical motors, shape memory alloy (SMA), hydraulic or pneumatic mechanisms. A particularly interesting actuation technnology is that referred to as "mechanical muscle" technology. SMA appears relatively less attractive because of the adverse relationship of force vs response, and high hysteresis.

In general, although "cleaner", pneumatics seem less suitable than hydraulics for actuators because of working fluid compressibility, resulting in compliance ("sponginess") and deflection rate ("stiction") characteristics which are difficult to control. New high-torque rare-earth DC motors offer a very competitive alternative for actuation.

Suitable sensing receptors for force/torque are generally available, as well as position sensors, although current technology advances promise significant improvements. Tactile sensing elements for proximity (stretching the definition of "tactile"), contact, force, imaging, surface and material characteristics, and slip are the subjects of much current research.

Breakthroughs are in order to be really applicable and useful for the ARH, but very promising devices are on the horizon, e.g., thin micromachined silicon arrays with both normal and shear measurement at each array site on 1mm x 1mm spacing. By comparison, currently available commercial tactile sensors have only normal force resolution capability at each site, and are approximately 1/2" thick (Lord tactile sensors).

For teleoperation, sensing must include force/torque and tactile feedback stimulation for the operator. Force/torque feedback is state of the art for bilateral control systems, but tactile feedback is another matter. By comparison with tactile sensing receptors, relatively little work is being done in this area. An example of what might be done is to adapt the soleniod-actuated pin-matrix technology used for Braille readers to a hand controller for the operator. Such a device, perhaps fitted into a "glove" controller, could potentially provide the operator with simulated contact, imaging, and force tactile feedback. Another technique has been suggested by AMETEK/ORED: thermal simulation of contact, imaging, and possibly force tactile feedback using a Peletier junction array.

Control is a very complex issue, being addressed through many research projects in the community. We are generally only tracking technology developments in these areas for relevancy to the TAM and ARH. We have, however, developed short- and long-term strategies for control, focussing initially on teleoperation for the short-term, and looking ahead for compatibility with likely future technological approaches for telerobotics and full automation for the long-term. This has been noted in preceding discussion.

6. Conclusion

This paper has presented an overview of AMETEK/Offshore Research and Engineering Division inhouse technology development efforts on an advanced manipulator arm (Tendon Arm Manipulator) and a dextrous end-effector concept (Anthropomorphic Robotic Hand). The current baseline design concepts for the TAM and ARH were presented and discussed, and key enabling technological issues were summarized.

7. Acknowledgements

Much of the technical work reflected in this paper was contributed by William Walker and Steven Gates of AMETEK/ORED, and Mark Danna, currently working with the University of California at Santa Barbara and the Center for Robotic Systems in Microelectronics, Santa Barbara, California. In addition, Mark Danna provided a valuable critique of this technical paper.

8. References

[1] J. Albus, "Hierarchical Control for Robots and Teleoperators" presented at IEEE Workshop on Intelligent Control, August 26-27, 1985.

[2] J. Vertut et al, "Human Factor Aspects of Computer Enhanced Teleoperation," presented to the 31st Conference on Remote Systems Technology, Proceedings, Volume 1, 1983.

[3] J. Stone, "Robotics: One View of the Evolutionary Path," Subnotes, March 1986.

[4] M. Montemerlo, "NASA's Automation and Robotics Technology Development Program," Headquarters, National Aeronautics and Space Administration.

[5] V.A. Anderson and R.C. Horn, "Tensor Arm Manipulator Design," presented at the American Society of Mechanical Engineers' Conference and Show, N.Y, N.Y., May 15-18, 1967.

[6] T.J. Drozda, "The Spine Robot...The Verdict's Yet to Come," Manufacturing Engineering, September, 1984.

[7] Martin Marietta, "Phase I - Intelligent Task Automation ITA)" AFWAL-TR-85-4062, April, 1986.

[8] R. Tomovic and G. Boni, "An Adaptive Artificial Hand," Trans. IRE, AC-7, 1962.

[9] F. Skinner, "Design of a Multiple Prehension Manipulator System," ASME Pub. # 74-DET-25, presented to ASME Design Technical Conference, October 5-9, 1974.

[10] S. Jacobsen et al., "The Utah/MIT Dextrous Hand: Work in Progress," Robotics Research, MIT Press, 1982.

[11] J. Salisbury, Jr., "Design and Control of an Articulated Hand," presented to the International Symposium on Design and Synthesis, Tokyo, Japan, July, 1984.

[12] M. Caporali and M. Shahinpoor, "Design and Construction of a Five-Fingered Robotic Hand," Robotics Age, February, 1984.

Future Research Directions

Moderator: G.A. Bekey
University of Southern California

Panel Members: R. Bajcsy, University of Pennsylvania,
J.Y.S. Luh, Clemson University,
A. Sanderson, Carnegie-Mellon University,
G. Saridis, Rensselaer Polytechnic Institute,
T.B. Sheridan, Massachusetts Institute of Technology,
C. Weisbin, Oak Ridge National Laboratories

The intent of this session was to provide views on the state of telerobotics research and to draw together a collection of suggested research opportunities to present to NASA. The panel members gave opening statements summarizing their views, the results of earlier sessions, and discussion periods. This was followed by a general discussion with the audience. The moderator concluded the session by stating a synthesized set of recommendations.

G. A. BEKEY: Welcome to the closing session of this symposium. I suggest each of the panel members take about ten to fifteen minutes to tell you about the research directions that they see in their own field, based on their experience or on the topics that they heard at this symposium. Then I suggest they take a ten-year leap into the future to give you a more distant extrapolation about where these research directions might or might not lead. After everyone has spoken, we will open the floor for comments, questions, contradictions, arguments and additional extrapolations. I am sure that some of you would like us to extrapolate an additional twenty years forward beyond the initial ten years.

C. WEISBIN: As most of you know, after being here for three days and hearing all of these talks, any attempt to summarize near-term and long-term research directions in fifteen minutes would be a rather formidable challenge, so I will respectfully defer and do something else. I propose to tell you about the research items that I think are interesting and not worked on very much, some things I have heard at this conference that I would express some cautions about, and some things that might surprise us. So, what I cannot do is to state all of the research that needs to be done in vision, multi-sensor integration, and so on. Many of these programs have been ongoing for some time. My comments will obviously reflect my very personal biases. In a meeting with parallel sessions such as this, one cannot be at every session, and therefore cannot always be fair to everything that was presented. So, three types of comments: high-priority, cautions, and surprises.

High Priority Issues. The first one has to do with the Minnesota work which deals with unexpected events and self-understanding. We are working in this area also. I happen to think that this is very important and it is an area which has not received a lot of attention. Most of us in this room realize that systems are not necessarily going to work the way we expect them to every time. But if one counts the number of papers or articles that address the issues of coping with contingencies and understanding self-awareness, I think we would find they are very few. A second area has to do with three-dimensional world modeling and understanding the environment. We heard papers for example from Hans Moravac and others on characterizing the world in which the robot lives. There, I suspect that people may assume that it is an easy problem, whereas in reality it is quite difficult. A third kind of problem is the issue of man-machine cooperative problem solving which was alluded to in several talks. The

PAGE 372 INTENTIONALLY BLANK

problem consists of having man and machine(s) act as a composite system to try to accomplish a specific task while equitably sharing the workload. One of the central problems is knowledge representation. When the man tells the machine: "you take over vision, I am going to assume the planning role," that assumes that each of them can communicate effectively at the same level. I do not think that we can do that today very well. Those are three topics (or topical areas) within the scope of the AI discipline that I regard as high priority issues. I do not think these areas are receiving adequate attention, although I feel they are very important.

Cautions. We heard a few papers which deal with the issue of reasoning with uncertainty. There has been a lot of time and effort spent on how to propagate uncertainty. One caution there, at least in my limited experience, is due to the potential difficulty in obtaining the uncertainties in the first place. We are not just talking about statistical uncertainties in data measurements. There are also uncertainties in rules and in heuristics which must be assessed. We must also be able to differentiate uncertainties from mistakes. There is much basic data that needs to be obtained. The limited efforts that we have made in that direction have indicated clearly that the problem of getting and characterizing the basic uncertainties data is just as important as that of propagating it and using it in a decision-making environment. We also heard some statements such as "this is an NP complete problem and we cannot solve it" and questions concerning the lack of formal methodologies to deal with the combinatorial explosion. But what if we are looking for a solution, not the best solution, just a satisifying solution. The warnings about combinatorial explosion are still appropriate and need to be heeded. But, in many cases, these problems might be made easier if we do not absolutely insist on getting the best solution and find an adequate solution instead. We also heard several papers which deal with qualitative physics. This has been very enigmatic to me. Sometimes when you try to form a hybrid of that type, (hard science ["physics"] tempered by heuristic or integral approaches ["qualitative"]), you can escape with the worst of both worlds. Qualitative physics has a very noble goal, but it can also be fraught with difficulties. Then we heard some papers on learning by discovery or by analogy. There I worry a little bit about the applicability to real-time problems. I think that is really hard, and I am not really sure it is possible.

Surprises. The first surprise is that, out of the whole conference, I saw one or at most two papers in the area of concurrent computation and parallel computing. That surprised me because this is an area that is absolutely fundamental to getting anything done. There may have been many more papers in sessions that I did not attend. At least in those sessions that I attended there was surprisingly little discussion of implementation of parallel algorithms. In a similar vein, you might be aware that there is a renewed interest in neural networks. The UCLA paper is an example of that. But, given the amount of interest that there is currently in neural networks, the number of papers on that topic that were discussed up at this conference was unusually few.

I have several remarks about the conference as a whole. First of all, I wanted to thank the organizers. I thought that it was a really interesting and well organized conference. You inevitably run into a difficulty when you do things too well. The large attendance forces you to go to parallel sessions. That is the normal problem. The accommodations, the people who were here, the subjects, the papers, were well worth coming. It was all very interesting as far as we were concerned. We would look forward to another meeting of this type.

T. B. SHERIDAN: Each of us looks at the world with glasses colored in a different part of the spectrum. My part of the spectrum is supposed to be that of the man-machine interface, which I have been interested in for a long time. From that point of view, let me categorize my comments into several parts. One of the most human things is that we use words. In certain emerging fields, words sometimes get a bit fuzzy and imply things that we, if we are honest with ourselves, may not quite muster. But we persist in using those words anyway. In a conference like this, and really for a long time, I have been a bit concerned about working toward using the English language in such a way that we are not kidding ourselves and are meaning the same thing. A term like teleoperator, which has been around for a long time means to do something at a distance. It has come to mean doing something with more or less continuous human control at a distance, although I suppose you could have a computer teleoperating another computer. I think that term is more or less settled in terms of human control.

The word telerobot means controlling from a distance something which at least has some autonomy. It means human control, in presumably supervisory fashion, of a robot which to some extent is autonomous. But, it seems to me here NASA or somebody might define what the term telerobot means. Let us not use it to mean all of teleoperation.

We come to the word telepresence, and there is an open opportunity for chaos and confusion. Telepresence can mean two things. It can mean feeling like you are present somewhere else. You could also use the word telepresence to mean you are operating at a distance just as though, and just as well, as if you were there. But, you can be telepresent and still be doing supervisory telerobotic control. Or, you can be telepresent controlling and not feeling the present. What I am saying is that there are various interpretations of that term needing better definition. One can begin to name other terms. It may be time to draft a semi-official glossary defining such words. I think that is important, because people confuse each other with words.

Some things I would like to applaud. We have talked about end-effectors for a long time, but we are beginning to see a real experimental science emerge on the use of more complex end-effectors. We are beginning to understand what it means to have multiple degrees of freedom in an end-effector. The work of Ken Salisbury, Larry Leifer, and others I could name is beginning to give us some nice science. Some of David Aiken's work on the combined use of the body and the hands is providing data. We need data. We have not had much data in this area. We are beginning to get some microdata. Blake Hannaford, Larry Stark and others reported on microforce data. That is, within a task, there are different forces that do different things at different times. I really applaud getting instruments right into the active manipulation site and getting some good hard data. Also we are beginning to understand this whole problem of impedance control. Not just force control. Not just position control. It may be some kind of a hybrid that generalizes the whole story. These things all go together.

As a next category, let us look at vision. Some very exciting things are coming along with head-mounted virtual displays. These help on the sensory side to achieve some telepresence. They also permit us to do other things that we have not been able to do before: wider fields of view, certain kinds of simulation experiences, use of computer graphics to help an operator see things that he or she normally could not see, superimposing video graphics on TV, rendering multiple views coming in part from a model, etc. These are all things that are relatively close in.

We jump a little bit farther and get the computer operating at a more profound level. We are beginning to see some nice work done in areas of linguistic interfaces with semi-automatic telerobotic systems. Here, we should not lose sight that we need both: what I would call symbolic language, which I am using now as I speak; and analogic language, which we use with steering wheels, joysticks, and our hands when we point, pull, push, etc. In our everyday life we use both of these languages when we communicate with each other and with the physical world. I am sure we need to understand how to combine the use of both of those languages.

Computers for a number of years now have helped us resolve coordinate systems. That kind of thing is very well in hand. They are going to begin helping coordinate two arms, helping us coordinate arms plus vehicles that have to be controlled at the same time, and indeed helping us coordinate any linkage or set of rigid interconnected bodies with more than six degrees of freedom. There are some tough problems still to be resolved there.

Looking into the future a little bit more, we see computer aids for sensing and planning. Now that is a bigger package. But, I think that as long as we consider that these are really, in large measure, aids to help the human sense and plan, we come to realize from looking at human anatomy that there are many more nerve cells for sensing and planning than there are for motor control. Indeed, in telerobotic systems, that is going to be the same. We are going to have much more cost, expense and complexity in sensing, planning and decision than we are going to have in purely motor control.

Another topic which I believe has been neglected is finding good performance measures for teleoperators and telerobots. It is still very difficult to determine that any given telerobot is better than another. Of course, you have to be able to say that it is better at what. Now, you could say that there is no way of answering that question unless you are talking about a very specific task. That may be true in an ultimate sense. I would assert that there is, in-between, a possibility involving a battery of tests, intelligence tests or motor-skill performance tests. These tests would consider a whole range of different kinds of tasks that you can do with hands, eyes and sensors, automatic control system, or planning, etc. These might be a better foundation for comparing the performance of different telerobotic systems. We do not have that yet. In fact, if you look at the literature, very little research has been done on it. Everybody has a different task board, in effect.

Finally, I will touch on one of the problems that has been a concern to a lot of people. The way that AI researchers, and other researchers that work on heuristic programming and develop LISP programs, talk about models and control is somehow different than the way in which control engineers talk about modeling and control (which usually is in terms of differential equations and things like that). I think that it is time that we interconnected these two ways of thinking: the analog and the digital. There is a lot of hard work to be done there. I see that as a very long range problem.

G. SARIDIS: I will be trying to express my feelings, feelings is the right word, about the area that I am interested and concerned with. This is the theory of intellegent machines. In this meeting, as in several previous meetings that I have attended, the underlying idea is that of associating some kind of intelligence with the robots or machines. I think that there are three basic disciplines that contribute toward a theory of intelligent machines: artificial intelligence, operations research, and system theory. All of us probably agree that this is where we should be looking into, to develop such a theory. I was asked to express the present situation in my area and to discuss where we stand regarding intelligent machines. Let me be a little more specific. My interests are autonomous systems, in contrast to man-machine interactive systems. Autonomous systems, typically called robots, are systems that can perform autonomously without interaction with the human operator.

In terms of actual applications, I can say only one thing: there are very few. There are very few places where one can find a machine that has some kind of intelligence. There is a lot of research going on. For instance, I just came back from a visit to JPL, and I was really impressed. I saw a "smart" hand, and I saw some other components of intelligent machines. For quite a while, JPL has been a pioneer in this area. In general, NASA has been interested because of the unique work in space that it is doing. This includes both manned and unmanned space exploration. I think we have been de-emphasizing unmanned space exploration. The funding seems to have decreased, although great work has been done. Instead it should have been increased. Industry is another area where these machines are applicable. We are talking about the factory of the future. It is going to be modularized. There will be no workers there doing things that the machine can do better. The human operators will be in a role involving primarily monitoring and supervision. There might be some maintenance crews. The humans will be doing the more intellectual work of designing and building those machines. Underwater and underground exploration are other very important applications. Automation of mining operations is another application area in which there is much interest. Nuclear handling is another area. This has been explored for quite a while. There are, for example, interesting applications for smart robots in safeguarding nuclear plants. The problem of safety could be improved. There is also some role for automation and robotics in medicine. Other organizations (e.g. government agencies) have also established efforts in automation. Substantial research is also being conducted in universities. Most of the major universities have a strong program in automation. The automotive industry has also been pioneering in research to create an automated environment in the factory of the future. I would like to make a parenthetical remark. An intelligent machine can be any machine that can perform tasks (including anthropomorphic tasks) without interaction with a human operator. Some of the machines used in the automotive industry are examples of non-anthropomorphic machines that can perform anthropomorphic tasks. Work in Japan has shown the first level of intelligent or semi-intelligent (i.e., smart) machines that will remove the worker from the factory floor. Now they are working on the second level which will demonstrate more intelligence. They have different stations and robots that can move from one station to another and be reprogrammed.

I would like to now turn to the issue of how to build those machines. There exist, of course, various components. There is the motion hardware. There is sensing - different types of sensing such as vision, tactile and proximity sensing, etc. One issue is how to integrate them. How do you get the people involved in these areas to communicate with each other? How do you get what I call equalization of communications? How do you establish a common language so that there can be meaningful exchange among the various research groups? I am a strong believer in a mathematically structured intelligent machine. How is that going to be materialized? I proposed a probabilistic model. Other people have proposed fuzzy or possibilistic models. AI researchers want to have a purely heuristic model based on AI principles. Are we going to have a hierarchical structure or a distributed structure? What are we going to emphasize? Kinematics? Dynamics? Both? Is control going to be adaptive? Is trajectory planning going to be done using simple kinematics? How are the actuators going to be incorporated? How is the hardware going to be improved? If there is some kind of heirarchy, how are the higher levels going to be constructed? Is there some additional intelligence at the higher levels? How are tasks going to be executed autonomously? Is the task to be decomposed or synthesized for comparison with the command?

Finally, I will make some brief comments about the future. What is the final goal of an intelligent machine? How can we tell that the machines that we are building are really intelligent? How can we measure that? In response to this question, I will offer the following remark. We will have succeeded in building machines that are really intelligent when we can build robots that can by themselves build more robots.

A. SANDERSON: I will comment on the topics in artificial intelligence that have been discussed at the meeting. Those topics have focused on issues of task planning and trajectory planning, as well as navigation of mobile robots. This is a subset of the broader spectrum of issues in artificial intelligence, but it seems to be a subset that is particularly relevant to the kind of tasks that NASA has defined. In this area, there is no shortage of buzzwords and jargon. For example, the discussions reflected the issues of: what is scheduling vs real-time planning vs what is off-line planning; what is reactive planning. There is a whole set of terminology which reflects, in a sense, the rather nebulous view not only of the solutions but of the problems themselves. I think it also reflects the clear view that these kinds of techniques are going to be the keys to the evolution of the technology, but in fact there are basic concepts and basic issues that need to be worked out. It is not a case of taking existing tools and applying them appropriately. It is the case of working out some quite basic principles and understanding how in the case of particular domains and applications they can be made useful.

This background slide suggests a task domain where issues of task and trajectory planning are relevant to space applications.

BACKGROUND

Space-based Diagnosis, Repair, and Assembly Tasks:

- Materials Handling
- Fault Diagnosis
- Reasoning about the Origin of Faults
- Hypothesis Formation and Testing
- Planning and Execution of Repairs
- Disassembly and Assembly
- Replacement of Parts

This slide suggests some topics or tasks in diagnosis, repair and assembly of systems. The kind of issue which arises here is the mixture of complex tasks that need to be accomplished in a relatively isolated environment. There are problems of materials handling, reasoning about faults and diagnosis of faults, forming hypotheses and developing strategies to test systems, planning and execution of repairs of those systems (which in itself involves assembly and disassembly of parts and replacement of those parts). These are all subtasks which go along with the idea of having systems in isolated environments, and other systems with sufficient capability to keep those systems repaired, maintained and serviced.

You can look at this list and ask what is different about doing this in a space environment, as opposed to the problem of repairing your computer when it breaks down. I think there are several dimensions to that. One is that the physical environment in which you need to carry out these operations is quite different. The vacuum, zero-gravity, lighting conditions, the physics of sensing and manipulation are quite different. So you need to be able to do the manipulation and do the tasks in a different environment than might be done in other situations. Secondly, the situation itself tends to be more isolated and less interactive than in other cases. It really forces you into the issue of asking what it takes to do things in an autonomous or semi-autonomous way, where you do not have the fall-back position of someone walking in and interacting with the system. So it really forces the issue of how to accomplish useful work and tasks in an autonomous way. The third aspect is the nature of the systems themselves. That is, you are not typically dealing with a line of computers or television sets that you see the same things with the same faults every day. There are many unique systems which have to be addressed, unique designs, unique requirements; and you have to be prepared to cope with those unique requirements. So that set of issues really defines these tasks as quite different from any other related tasks. The collection of those poses a considerable burden on how you make systems, and autonomous systems, to be able to cope with these issues. I could make a similar list for another area such as navigation, such as exploration, and with similar kinds of constraints that are imposed that make the task in fact quite difficult and challenging. The technical challenge is, I think, apparent. It is a combination of reasoning and manipulation, and how to combine those together.

I will use one other slide which summarizes some of the research issues which are posed by this kind of task.

RESEARCH ISSUES

REPRESENTATION

- Geometrical, Physical, Functional
- Uncertainty
- Dynamics

PLANNING

- Diagnosis and Repair
- Sequence Level
- Discrete Task Level
- Motion Level

CONTROL: MANIPULATION AND SENSING

- Adaptive Sensor-based Systems
- Learning Systems
- Multisensors

This summarizes many of the issues which came up during technical sessions. I will not try to summarize all those in detail. Let me just make a few points. I think that there is a clear relationship between representation and planning and control. As someone suggested, planning and control are not as different as many people sometimes pretend. If you think about what even classical control attempts to do, it really largely has to do with resolving uncertainties. It has to do with being able to predict what those uncertainties are going to be, in a way that you can take timely actions to correct them. That has to do with the fact that, even in relatively simple control systems, you do not have a perfect model of the world, and you cannot devise your actions totally in advance in order to accomplish a task. So, you have to work with uncertainties, and you have to be able to develop predictions of the world in order to accomplish the task. This has a lot to do with planning, and planning, as it is used typically in AI, is a kind of reasoning in a symbolic world. It is a representation of the world which tends to be symbolic and relational. The kind of issues of representation includes geometry, physics, functions of systems, temporal aspects, etc. Representation of uncertainty is fundamental to that, in the same way that it is in control. It is uncertainty, again, not in the sense that there are things that you simply do not know. But, the more that you can explicitly describe the nature of that uncertainty, the more knowledge that you have to incorporate into the system. The dynamics are increasingly important as you look at the kinds of systems which are being discussed here. So, as you look at the strategies toward planning, you typically see a hierarchical approach. You typically see various levels of abstraction defined. In the case of manipulation, you can define sequences of tasks. You can define sets of discrete operations. You can define explicit motions, in the sense of trajectory planning. It is easy in a sense to link those together in a block diagram. It is much more difficult to be able to make systems that work and accomplish those tasks. One of the issues, if you look at the work on planning, is that you do not see at this point complete working systems. You see pieces of systems and concepts toward working systems. At the lower levels of control, manipulation and sensing, you see a trend toward more adaptive and learning systems with the ability to integrate information and resolve uncertainties among multiple sensors.

I have a few notes on specific topics, which I think are worth mentioning. One is the problem of validation and evaluation of systems. How do you know when a plan works? When it is working, how do you know when to go back and replan? Again, in a dynamic and changing environment, that is important. The question of unexpected events and error recovery is implicit in this. Someone pointed out that, in many robot programs, 90% of the program code deals with the unexpected events. The question of why is planning difficult, again, we could go on and on about. One of the basic issues is the computational and combinatorial barriers that you reach, given the complexities of the task. Trying to evaluate all the possible alternatives simply yields a combinatorial explosion. In practice, the solution to that is to look for efficient representations, and often domain-specific representations and control mechanisms.

I will finish with a couple of remarks. One is to reiterate that the payoff I see from this technology is really in the evolution of systems. That is, it is important to have the core technology which becomes embedded in the telerobot systems and evolves into autonomous systems. The final payoff as we expect is perhaps with autonomous systems, but issues such as representation, such as system architecture, we cannot simply wait and implement in autonomous systems. I think the concepts have to be embedded earlier in the evolution of the systems. The final remark is a rather philosophical view of robotics. One can think of robotics as a collection of devices (robots, hands, sensors, etc.), or one can think of it as a science which deals with principles. In fact, it is a mixture of those things. In many ways, I think of it as an experimental science, one in which you have to build systems and try to find out why they work and why they do not work in order to make progress. I think in this area in particular you have to build systems and use planners with real robots in order to make progress and come up with useful results.

R. BAJCSY: I will concentrate on stating my views on future research directions in sensing and perception. Here is the world which we live in. Here are the sensors, i.e. , hands, ears, eyes, etc. Here is the representation. To set the stage, I assume that I am interested only in the world of physics and geometry. I assume, for the purpose of this discussion, that we have only contact and non-contact sensors. This is not a severe limitation, since in fact there are no other types of sensors. I would like to impress upon you that sensors sense only partially through a window of the world and only some aspect of the world. So one sensor can never sense the full world. So, having set the stage, I proceed to the mission impossible. The job is to recover the world from the partial measurements. Here are my questions for the next two or three years. Then, I have another set of questions for the following two or three years. Then, I have another slide for the next ten years. Question number one, which is dear to my heart, what is the information that you lose through these transducers? From which follows, models of hardware and/or software. In my view, software is equivalent to hardware, i.e., an edge detector could be implemented either in hardware or in software. These models represent the processes that have to take place as you go from the world representation to the sensors and to the sensor data. So, one issue is what is the information that is lost, and hence the modeling question. The second question is what are the rules, or principles, of recovery. In the first question, things are being taken apart, whereas in the second, things are being put together. The third question, which is very clear in tactile sensing and information processing, is that of data acquisition. This is an essential part of the recognition, or manipulation, or whatever use is made of the information. This issue, in my view, is just as important in vision, although it may not be as readily apparent. Hence, the control issues are a critical part of the requirements for machine sensing. We must get away from an approach to machine sensing in which there are no global goals that involve how the information is to be used for the purpose of control (as an example). This is my view of the present state and of the important immediate issues.

I would like to share with you what I think is a plan for the next ten years. I really view space exploration and space robotics as representing a great opportunity as a research laboratory, or as a research environment, for discovering rules of evolution. Why? Well, space represents different physics and physical laws. Different radiation. Our sensors have been developed or have evolved to be sensitive to a certain spectral range. In space, different sensors will be needed perhaps. But, of more interest to me, is the different representations that will evolve from the different sensors and physical laws. Different rules for update of this knowledge will also evolve, as the sensors take measurements and interact with the world. Hence, the rules of evolution.

J. Y. S. LUH: Being the last person in the panel has advantages (I will not say disadvantages). All of the important and interesting issues are well taken care of. That means that I do not have to say anything. On the other hand, I have noticed a few minor things which have not been mentioned by the previous panelists. These are down-to-earth, for instance, kinematics, dynamics and control. These are basic issues. Without kinematics, dynamics and control, there would be no robots, either in space or in industry. Of course, there has been a lot of research on these topics for many years. Nonetheless, there are still open issues that need to be studied and investigated in order to improve space operations.

For instance, one issue is that of robots with closed-kinematic-chains. These robots, it seems to me, would be very useful. This topic was studied many years ago in a very general sense. But, very few results were obtained that were specific to a robot and that were relevant to space applications. In order to investigate the closed-chain robot, extensive computations (for instance, to do inverse kinematics) are required quite often. To shorten the computations, parallel computation is one of the topics that needs to be looked into. Redundant robots is another topic which I think is important. I only judge from my experience. Everybody knows that the human body has redundant kinematic chains. Without the redundancy, you could not do a lot of things that you normally like to do, such as scratching your back when it itches. If we look at dynamics, we need to do a lot of work on analysis of flexible joints and flexible links. So far, the most complicated flexible robot that, to my knowledge, has been analyzed consists of two links. This is not really enough. There is a need to pursue that kind of research.

If we look at control, we use a lot of words and names. For instance, one of the oldest control schemes in robotics is that of computed torque. That was done ten or fifteen years ago. Then, later we developed resolved rate, resolved acceleration, nonlinear transformations, adaptive, force control, impedance control, etc. But, there is a result in a paper presented here by K. Kreutz of JPL in which he found a lot of equivalence among the various concepts. By changing the names and equations, a lot of control schemes which seem to be different are in fact equivalent.

But, whether the control schemes are equal or not, the most analytical results are very seldom applied or implemented in hardware. As in several of the earlier comments made by the panel members, robotics is an experimental science. It is nice to do analysis to guide your experiments. But the final judgement really comes through experimentation. All the control schemes may look beautiful in analysis (and I expect to get a lot of comments from the audience on this), but we also need the implementation and experimentation to prove feasibility and performance based on our current hardware technology.

In space applications, I listened to a paper presented on a virtual robot. I think this type of work is particularly relevant to space applications. Similar or equivalent work should be promoted.

Now I move to teleoperation and telerobotics. I agree with comments made earlier about terminology. Telerobot implies remote control of a robot. Teleoperation can involve two robots or even multiple robots. There are a lot of papers on that topic. There are two items which are basic issues and which have not been mentioned. It is my opinion that the main issue in the coordination of two or more robots is how do you divide the work. This is the task decomposition which is the main issue before we even start to do anything else. There is a Chinese story that is pertinent here. There is a monk in the monastery. If he wants to drink water, he carries the water from the well to the monastery in two buckets of water, one bucket at each end of a beam, and the beam is balanced on top of his shoulder. If there are two monks in the monastery, then the same beam can be used. Each of the monks supports one end of the beam. Now, however, there is only one bucket at the mid-point of the beam. If you have three monks living in the monastery, there will be no water. The point of the story is that in the monastery they must decompose the task to determine who is doing what. Otherwise, the three monks (or robots) will just sit there and argue who is doing what. A second topic which is also basic, in my view, and which has not been mentioned is that of limb coordination. We are talking now about two robots and about perhaps extending this to walking robots. There is a lot of interest in walking robots, instead of wheeled robots. Multiple arm robots with dexterous end-effectors are also being discussed for space applications. I believe that the coordination of arms with the legs and end-effectors is also a very important issue.

G. A. BEKEY: I would like to thank the panelists for their contributions. We have about 45 minutes now for comments, rebuttals, questions, suggestions, etc. If anyone would like to make a comment, I would like to suggest that he or she step to the microphone.

A. J. BEJCZY, JPL: I would like to make a comment on intelligent machines and their relation to man. How do we communicate with intelligent machines? I assume that intelligence is also measured not only by the action that a machine can perform, but by the way that a machine can communicate with other machines, or in our case with an operator or with someone that is supposed to use it. In a telerobot sense, an intelligent machine is for a user. I would like to ask a sharp question on the issues of how to communicate with intelligent machines and how is intelligence defined from that point of view.

G. SARIDIS: This was a loaded question. First, let me start with the intelligent machine itself. I do not think that we have built an intelligent machine. Therefore, we have not explored all the capabilities of our machines regarding intelligence. But, suppose for the moment that we could do that. A good idea is to use the same approach that we are using in dealing with expert systems. With expert systems, we have some intelligence stored in the computer. The operator can, by means of if-then statements, communicate with that particular computer. This assumes that the computer in which the expert system resides has what I would call static intelligence. An intelligent machine would have what I would call dynamic intelligence. I think that it would be an improvement on an expert system to do that job.

R. BAJCSY: Communication with an intelligent machine depends on the representation that the particular machine has. It was precisely this point that I was trying to promote in my presentation. The world is out there. We have these limited sensors which reduce the data and store it in memory in some organized fashion. This we call representation. There are also additional rules and laws that control both how to gather the data and how to store the data. In space, we have this wonderful opportunity because we do not have a preconceived a priori representation of that world. Therefore, we have an opportunity to gather this knowledge iteratively and to learn something about learning itself and also about evolution.

A. LOKSHIN, JPL: I would like to share my views on the problem of robot error recovery after what are referred to typically as unexpected events. I believe that they are quite close to the views that the panel expressed, but I would like to introduce additional ideas. I believe that the only unexpected event that could be properly dealt with is an event that is expected well in advance. If you look for example at our society, people (such as fire-fighters, commanders, etc.) that are supposed to deal with real unexpected events, have very rigorous training to be able to foresee any kind of possibility in advance. If a person behaves properly in real unexpected events, he usually gets a medal. I do not think that the PUMA 560 deserves a medal every time that it does a simple task such as acquiring and grasping a tool. Therefore, a more precise term than unexpected events is that of abnormal events. If we use this term, then work that focuses on case studies in a particular task domain will be encouraged and more readily accepted. Such case studies that address the issue of dealing with abnormal events in particular task domains are needed in robotics. For example, a robot could try a thousand times to unscrew a bolt and then tell us all of the problems that it encountered in all of those attempts. This experimental approach would be very similar to the one used in the very early stages of the development of coding theory. There, a lot of very simple statistical experiments (such as counting the number of times a particular letter of the alphabet was used in average speech) were conducted. I think this type of work and presentations should be encouraged in robotics also.

A. SANDERSON: I agree. There are levels of abstraction in terms of thinking about what is expected and about what is normal or abnormal. The key question is what is the capacity of the system to interpret any kind of event and how does it decide what to do about them. In some sense what you are referring to with the PUMA is that, when you program it to read a monitor that is tripped, and you have a fixed subroutine that it goes into, you can argue that it is error recovery, but it is not intelligent error recovery. The more general case that you are referring to is where you have a deeper representation of the system and where, instead of having a kind of subroutine, you have some capacity to reason about the nature of the event and its origin and to think in a deeper sense about the appropriate reaction, given all the constraints, the states of the system, and the goals.

C. WEISBIN: I think that it would be rather bold to suggest that we could deal with unknown unknowns, which is what I think is at issue. When you try to put a framework in place to deal with classes of potential events, as opposed to specific recovery modes, you need a framework for broad on-line dynamic replanning, which is more than simple error recovery.

I fully agree with your question. If I did not ever anticipate something to happen, it would be unlikely that our robot would cope with it properly. On the other hand, I do believe that you can anticipate broad classes of events that you can handle on-line through a replanner, without having prescribed in detail the particular event and the particular reaction to the event. No matter what we call that, whether it is error recovery, unanticipated events, robust planning, etc., the ability to perform this function is essential.

G. A. BEKEY: Let me add another comment to that from my personal experience. Many years ago I worked on issues of trying to model human performance in particular kinds of systems. Tom Sheridan has done similar work. One time, we were interested in modeling the way in which human pilots adapted to particular catastrophic failures in aircraft, under situations where recovery was possible. There are obviously times, such as the extreme case in which the vertical stabilizer falls off, when you cannot recover. There are situations where the stability augmentation system fails, and recovery is possible, but the control strategy that the pilot has to follow is different. The issue here is very similar to that raised by the question. Is this an unknown unknown, or is it an unknown for which previous learning took place and there are strategies of recovery stored somewhere in the system? What we did was to try to model the recovery strategy, and it was obviously highly dependent on previous training. This raises very complex issues in AI and in the way in which knowledge is represented. It is a very interesting question with a lot of implications in many of the areas that we are dealing with.

R. SCOTTI, ORI: I would like to raise the issue of what is the ultimate model; what sort of ultimate model are we thinking about, when we think about robotics and artificial intelligence. The question came up in one of the sessions, where we discussed the prospect that perhaps it is an anthropological model that we are talking about if we follow the idea that man is the model. It is an issue, what is the model. As in our experiences with science in the past, solving a problem really revolves about understanding and having a clear idea about what the problem is. History has shown us so many times that once we get to that stage, then the answer just comes out. The issue of what is the model is really important, and I would like to hear something on that.

Another issue that has come up is that in many other fields of human activity, specifically sports, there is an entirely different understanding about the relationship of thinking, acting and accomplishing. This is the concept of visualization. I have seen some really wonderful demonstrations of optical simulation which has been proposed as a method for doing things where you cannot see. But, the whole issue keeps coming back to me of how we can investigate the relationship between what a person conceives and visualizes and what that person can actually accomplish. Or, turn it around in the other direction. I think that we have not yet begun to scratch the surface of this very interesting area.

A third area, which may be a bit sensitive, but which I feel strong enough and bold enough to mention: what is our responsibility as sincere dedicated people to unfolding these concepts and putting them before humanity in our evolution? What is our responsibility to the morality and to the integrity of these things to which we are dedicating our lives' energy. Specifically, what is our responsibility to the future environment in which we have to work. It is definitely an economic, financial, environment. Space station for example is going to require funds and guidance coming down from the top. We are all of the age where we have seen these projects come off from a very sound position of integrity and morality. Then, for some other reasons, the scientific results are used to address other issues.

G. A. BEKEY: Let us respond to the three issues one at a time. Does anyone want to comment on what the right model for robots in AI is? Is a human the right model for a robot?

T. B. SHERIDAN: We probably all have something to say about that. Each of us cannot get out of his own skin. So, to some extent, our model of the world starts out being ourselves. There is no other way it can be, for starters. The better we define and understand a problem, the less the thing we build to solve the problem looks like ourselves. This has been generally true in human technology as it has evolved over the centuries. It seems to me that we always start out anthropomorphically. But the better we understand it, the less it looks like a human because we refine the technology to serve that particular need. At some point, it may stop looking like a human. In so far as it is so robust that it handles things that we cannot anticipate as human beings, we get back to our unexpected event, and there is no solution. My feeling about unexpected events, to come back to that for a moment, is that there is an expectation of zero on the density function. It involves infinite information, and there is zero apprehension on the part of the human. We are always in this sort of scale, starting from ourselves and what we totally expect as humans, and sliding toward specific technology for specific tasks as we understand them.

C. WEISBIN: I would take the position, as an engineer, that we are trying to get a job done. Alternatively, one could take the classical AI approach that aims to study how humans reason and think. I do not think that there is a right or a wrong answer. However, I would clearly come down on the side of getting the job done. I am not sure if you were asking whether the physical object that gets the job done should have anthropomorphic characteristics, or that the cognitive aspects should have anthropomorphic characteristics, or both. I think that we would be doing ourselves a disservice if we insisted that robots have only two arms, that they be able to lift only a certain amount weight, etc. - in order just to follow the human model. One could argue that, since most of our current plants are built for humans, we ought to build robots that look like humans because they would fit into current plants. This might be short-sighted. I would therefore come out <u>against</u> the view of physical anthropomorphism and the view of exclusively confining ourselves to paradigms of human reasoning. Where you can solve differential equations and accurately get the right answer, this should be the approach. To take the view that we should reason with heuristics because people do, is not always the best approach, in terms of getting the job done. If you are trying to study human beings, then the context is different and it would make sense.

G. SARIDIS: I would like to bring up an example that I encountered when I was in West Germany last summer. There was a project about plugging some holes on the body of a Mercedes-Benz with rubber plugs. Humans used to do that, but it was to cumbersome and unhealthy. So, they decided to try to replace the human. They did a study to understand if a vision system could be used to center and align the arm to the hole and if an arm control system could then be used to push the arm into the hole. They were very unsuccessful. They replaced that with a magnetic scanner, and they were able to do the task with only one attempt. This proves the point that C. Weisbin made.

R. BAJCSY: If I assume that in your question you are equating model with representation, then I wish to make a very strong statement. There is not one unique representation. There are multiple representations for anything that you wish to do. A second remark is that: I do not think that Homo sapiens should be so pretentious to think that they are always the best models for a given job. I think models should be task-driven.

G. A. BBKBY: There is a cartoon that many of you may have seen. It shows a human being whose function it was to push the emergency button in case of a problem, and he was replaced with an anthropomorphic looking robot with its finger ready to push the same button. This is clearly not the right kind of a model.

The second question had to do with the relationship between thinking and accomplishment. Since human beings can improve their tennis by playing inner tennis in the sense of using visualization to improve their performance), can robotics be improved by using a similar approach?

C. WBISBIN: Speaking as a rather poor athlete, who is uniformly poor in a variety of sports, I believe that athletes improve performance through rote practice without necessarily visualizing each detailed step. When you throw a bowling ball 27 times, or 2000 times, after a while it becomes a routine. If you are trying to get a kill shot in racket-ball, you do not necessarily do that very well through visualization, you do it pretty much by rote after significant amounts of practice. I make no scientific claim that this is true. It would be quite reasonable to take some intelligent machine for example, and have it practice in a variety of related configurations and try to improve skills through perturbation and credit assignment reward, rather than differential model building. That would get us into the area of learning. I think the athlete analogy is more one where humans improve pretty much through repetition.

A. SANDBRSON: Let me propose one possible scenario. You could think of a simulation as a kind of visualization in which the robot does not actually execute the command physically but could execute parts of it in conjunction with simulating other parts. In fact, it might be able to practice and learn without actually doing the execution itself. Some of the psychology that goes with the visualization approach says that humans may be able to do that.

G. A. BBKBY: Is any one willing to deal with the third and more difficult question, the one that deals with responsibility?

T. B. SHBRIDAN: First, I wanted to commend you for asking such a question, because it is terribly important. It is the kind of question that is so embarrassing to talk about, that we do not talk about it at scientific meetings like this. I think however that it is ultimately very important in AI and robotics. One of the reasons that we behave ourselves in the world is that, when we start encroaching on the rights of other people, we put our own bodies, in effect (personal reputations, health, etc.), on the line. Once we are able, in a telerobotic sense, to guide or to initiate the actions of semi-autonomous devices that can go out and creep around the world, and make trouble in the backyards of others, it will be very easy to abandon responsibility and to claim that the driver is not us. In fact, responsibility will be increasingly more difficult to trace back. We are getting into a territory here which is fraught with social problems. It is also my belief that SDI is the tip of that iceberg. I hope sincerely that NASA does not get in that business.

G. A. BBKBY: I think it is also important to know, however, that much research in robotics, as G. Saridis pointed out earlier, is concerned precisely with removing humans from some of those jobs which are the most hazardous to them, whether it is in mines, or in the presence of hazardous chemicals, etc. I think that clearly is an issue in which robotics is contributing to human welfare.

G. SARIDIS: I think there are two questions involved in that. One is our responsibility in creating machines that might be used by others in detriment to humanity. Second, suppose that we create intelligent machines, and these machines get out of control. Will these machines have the built-in potential to create some kind of chaos? Not because the humans ordered them so, but because of themselves. This type of hell is reminiscent of the movie 2001 Odyssey. We should keep this in mind as we aim toward intelligent machines. I think however, that education can solve the problem. It is a matter of developing social responsibility through education. We have, in the past, been very successful in creating instruments that can be used either to help humanity or to destroy humanity. Take the simplest of things, as an example. Take a glass, a small glass, that you can use to drink water. I can break that and cut somebody's throat. I can use that as a weapon. Unless I control myself, and I have to be educated to do that, I can use even the simple things as weapons. The second problem is much more serious, and I do not have an answer for it. But then,... I do not know either if we are going to ever create intelligent machines.

L. MARTIN, Telerobotics International: I would like to state an alternative view on anthropomorphic kinematics. The thoughts behind anthropomorphic kinematics is that we are trying to replace a man in an EVA atmosphere trying to do a variety of tasks. No doubt there are machines that can do specific tasks (such as punching buttons, plugging holes, etc..) in a much more simplistic manner. To be able to replace a man in that environment, with the design standards that are already in place for NASA to be used in satellites that will be put in place decades from now, we must look at machines that replicate the kinematic characteristics of the human (in size, lift capacity, etc.). I would like to make one observation in general. I think there is not a whole lot of hardware work being done at the present time. I would like to take you back about eighty years by analogy. I would challenge this group, that if eighty years ago you had the challenge that the Wright brothers had, you would be designing the automatic pilot before designing the airplane. I would suggest the more pragmatic approach.

G. A. BEKEY: Thank you for your comment. That will not go unanswered here.

G. SARIDIS: Well, I am for parallel processing, instead of serial processing. The question was whether we were going to do that serially or in parallel. We do both research and experimentation, in parallel. If we were smart enough ten years ago, when the Japanese announced their automation program, and we were doing basic research as they did, our industries would not be in the bind that they are in now. We are trying to catch-up and establish a more competitive role with their industry.

R. BAJCSY: I am sorry you got this impression. I am in academia and we do very hard hands-on experimentation. So do most of my colleagues in robotics. While we are doing hands-on experimentation, we have to look ahead. We have to look for generic problems, for problems that can be generalized. We have to look for principles. We have to make contributions to the general knowledge in robotics. While we are doing that, we are also testing on concrete examples. We are not avoiding doing hands-on experiments.

J. Y. S. LUH: That is a very good answer. I just want to add specific comments about the issue of kinematics. The problems of kinematics is well studied, and there are many results. The important topic currently is that of redundancy. The redundant kinematic robot has been studied by many researchers. There are to my knowledge anywhere between twenty to thirty papers, several of them even in journals. However, more experimentation is required.

G. A. BEKEY: I can also make one observation of my own. My colleague, Professor R. Tomovic of Yugoslavia, and I are in the process of not only studying but building a five-finger anthropomorphic hand for robotic applications. With any luck at all, you will see it at the forthcoming IEEE Robotics Conference in Raleigh. But, at the same time, I am very supportive of the efforts of my colleague Professor Lee, (who is sitting out here) who has designed a hand with two thumbs. Some of us may be all thumbs, but his is meant to be an alternative robotic hand which is not at all anthropomorphic. So, I think we need to keep the principles that various people have brought here all in mind. I think you would be surprised, back to the questioner, how much hardware work goes on. And, if we had the resources to do more prototyping and experimentation, we would do a lot more. Developing prototype hardware is very expensive, and most of us simply do not have the resources. But this is not from a lack of desire to do so.

D. PIVIROTTO, JPL: I wanted to touch on the point on why there were so few papers on parallel processing. One of the things that I have been doing is to search for someone who is actually putting together flight computer systems that we might be able to use for space robots. There is no one, as far as I can determine. Do you have any knowledge about research in computer systems for robotic applications in space? As probably everybody knows, space computers have to be reliable, fault-tolerant, radiation protected, etc. You cannot just take a PC and fly it. I also notice that there does not seem to be a whole lot of work in computational systems for practical telerobotics applications in space. There was a paper by R. Travasos, at a recent robotics conference, who tried to address the problem of linking up different kinds of computers to solve the overall telerobotics problem. I have not seen very much of that going on. This work is interdisciplinary. I was wondering whether the panel could shed any light on this. Are there research programs of that nature?

A. SANDERSON: I cannot say that I know of a real effort at space-hardened multi or parallel processing. The efforts at the research level tend to be at architectures for applications, and there are selected prototypes that have been built. At CMU, there is a WARP processor, a systolic array machine, which is being used on a mobile vehicle. It resides in the mobile vehicle. It is being used for low-level image processing. It is being used as part of the autonomous system. I do not know of an example which includes all of the aspects needed for space-flight applications.

G. A. BEKEY : It is very simple. Just get a space-hardened connection machine.

D. PIVIROTTO: Actually, the DARPA machines appear to be based on the following approach. People have a theory of what makes a good AI processor. Somebody else then has a theory about how to link machines together to do processing. Actually, none appear to meet the anticipated performance. I have heard papers on several machines. None of them are designed from the point of view of defining a system that we are trying to make work, and putting together a complement of computers to do the job. I think that people get seduced into thinking that, if we have the connection machine, all of our problems will be solved. From the papers I have seen, that does not sound right.

C. WEISBIN: What I had in mind when I brought up that question was that, if you attended many of these sessions, you would see a myriad of boxes: vision, planning, control, etc.. Many of the boxes would be running in parallel. The implication is that there is research on-going in which such an implementation is being considered. Without beginning to consider space-hardening or anything like that, our limited experience with a very narrow class of machines indicates to me that there is significant potential out there in hardware. However, there is a major problem in software. If you had ten, a hundred, a thousand processors, the scheduling problem of how to allocate tasks to processors, and to determine how they would communicate, is a very legitimate research area. My comments were that I expected to see more papers here in this area. Regarding your question about whether anyone is to the stage of fielding something (and I am not even touching space now), there is one military project that I am aware of where that is being considered. Clearly, the conditions that you would have in space would be quite different. In any event, I think it is premature. I do not think that we currently know how to effectively schedule and determine the communication modes in general for an arbitrary number of processors.

J. C. HORVATH, JPL: I am from the Hypercube group at JPL. I would like to respond collectively to all of this. There has been a project underway for quite a few years at JPL and Caltech to do parallel processing. We have built hardware which is meant and designed specifically to be space-qualifiable, eventually. Sometimes we have been criticized for being too simplistic just because we are trying to stay simple, so that we can deal with real-world problems. We have a substantive effort ongoing in software and have solved quite a few real problems. I am here at this conference mostly to raise interest within the robotics community in using the Hypercube. So, we are ready if you are (Laughter).

C. WEISBIN: I would like to ask a question of the speaker. I applaud the effort in using Hypercubes. It is fully consistent with what we are doing. I just want to know whether you disagree with the statement I made that, as far as I knew, task allocation and scheduling of a variety of tasks (planning, manipulation, control, sensing, etc.), with real-time feedback, is something that is still in the research stage and not ready to go. Would you disagree?

J. HORVATH: I would hesitate in this forum to say yes or no. I know that there are other members of our group that want to do it, but I do not really know how far along they are. I would have to refer you to them.

A. GOFORTH, NASA ARC: There is a program at Ames that is developing a space-borne symbolic processor. You might see a paper in a year or so about its applicability to telerobotics. We considered it for the strawman design of the Flight Telerobotic Servicer. Yes, there is work on processors targetted for space. It is rather premature to present papers, when the details of the hardware have not yet been worked out. The big problem for a space-qualified system is the power consumption. If you want to use a parallel processor, the real problem is the bus or the interconnections. That uses a lot of power. So, a lot of power is required to support parallelism. I am not saying that it cannot be done, but that is the real tall pole. With regard to the issue of tasking software for parallel operations, it is true that no one knows how to do that. But, for pure master-slave teleoperations, that is easy. It may be as easy as taking what is being done with VAX machines in laboratories and moving it into space-qualified equipment.

G. A. BEKEY: Let me suggest that it may be a good time to stop and perhaps continue on an individual basis. Let me summarize in a few sentences what I think has happened here. This has been an extremely enlightening discussion, at least for me. It is interesting to see how some of the issues came up again and again in the conversations.

- Many of the issues involve how we model the world. How do we extract information from it? Do we do it by using sensors? What kind of abstract models do we use for representation? Then, there is the issue of how knowledge is represented, transmitted and used for the design of systems.

- There is a question of where the intelligence resides, and the relation between building intelligent machines and, conversely, using machines which in some sense imitate intelligent behavior. Those two things may or may not be equivalent.

- The question of uncertainty came up again and again - ways of modeling it, of detecting it, and incorporating it in systems.

- The question of evaluation and validation of systems, particularly man-machine systems, came up several times. It is clearly a very important one, if we are going to be able to assess our successes or our failures.

- The above issues of modeling and representation, intelligence, uncertainty, evaluation, and validation are all very closely related to those of planning and scheduling.

- I appreciate the comment by Dr. Sanderson that robotics is an experimental science, and that there is a great deal to learn by experimentation.

- Finally, we should not forget that we do in fact have a responsibility to humanity and to ourselves.

One of the panelists said that one of the important areas is to improve communication among researchers. This conference has contributed admirably to that goal. I would like to thank Dr. Rodriguez for organizing it and bringing us all together.

APPENDIX A

PROGRAM SCHEDULE

SPACE TELEROBOTICS WORKSHOP

FINAL PROGRAM

(UPDATED TITLES FOR PROCEEDINGS)

January 20-22, 1987
The Pasadena Center
Pasadena, California

Jet Propulsion Laboratory
California Institute of Technology

National Aeronautics and
Space Administration

PROGRAM COMMITTEE

G. Rodriguez, Chairman
R. S. Doshi
K. Kreutz
D. R. Meldrum
M. H. Milman
H. Seraji
B. H. Wilcox

ADMINISTRATIVE STAFF

P. B. McLane
S. K. Owen

For Information Regarding the **Program** Contact:

Guillermo Rodriguez, M/S 198-330
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109
Phone: (818) 354-4057

For Information Regarding **Registration** Contact:

Pat McLane, M/S 180-205
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109
Phone: (818) 354-5556

Program at a Glance*

MONDAY
January 19, 1987

7:00	HILTON ROSÉ ROOM	Pre-Workshop Registration & Reception
9:30		

TUESDAY
January 20, 1987

7:30	LOBBY	Registration
8:30	LITTLE THEATRE	Workshop Welcome
8:45	LITTLE THEATRE	Plenary Session
9:45	LOBBY	Coffee Break
10:10		Concurrent Sessions
	C101	TA1: System Architecture I
	C102	TA2: Manipulator Control I
	C103	TA3: Spatial & Geometric Representation & Reasoning
	C104	TA4: Man/Machine Interface I
12:20		Lunch
1:30		Concurrent Sessions
	C101	TP1: System Architecture II
	C102	TP2: Manipulator Control II
	C103	TP3: Trajectory Planning for Manipulators
	C104	TP4: Man/Machine Interface II
	C105	TP5: Robot Simulation & Control
4:30		
5:00		Bus to JPL
5:30		Visit to JPL
7:00		Reception at JPL
8:00		

WEDNESDAY
January 21, 1987

7:30	LOBBY	Registration
8:00	LITTLE THEATRE	Plenary Session
8:45		
8:50		Concurrent Sessions
	C101	WA1: System Architecture III
	C102	WA2: Manipulator Control III
	C103	WA3: Planning & Scheduling: Basic Research & Tools I
	C104	WA4: Sensing & Perception I
12:10		Lunch
1:15	LITTLE THEATRE	Plenary Session
2:00		Concurrent Sessions
	C101	WP1: System Concepts
	C102	WP2: Manipulator Control IV
	C103	WP3: Planning & Scheduling: Basic Research & Tools II
	C104	WP4: Sensing & Perception II
	C105	WP5: Telerobots
6:20		
6:30	HILTON INT'L BALLROOM	No-Host Reception
7:00	HILTON INT'L BALLROOM	Banquet
9:00		

THURSDAY
January 22, 1987

7:30	LOBBY	Registration
8:00	LITTLE THEATRE	Plenary Session
8:45	LOBBY	Coffee Break
9:00		Concurrent Sessions
	C101	THA1: System Architecture IV
	C102	THA2: Manipulator Control V
	C103	THA3: Autonomous Systems
	C104	THA4: Sensing & Perception III
12:00		Lunch
1:30	LITTLE THEATRE	Panel Discussion
3:30	LOBBY	Closing Reception

*Unless otherwise indicated, all events are in the Pasadena Center

Technical Sessions, Tuesday Morning, January 20

7:30 am — 8:30 am   Registration

8:30 am — 8:45 am   WORKSHOP WELCOME — Little Theatre
Lew Allen, Jr.; Director, Jet Propulsion Laboratory

8:45 am — 9:45 am   PLENARY SESSION — Little Theatre
Paul S. Schenker, Jet Propulsion Laboratory
NASA Telerobotics Research — Program Objectives and Technology Outreach

9:45 am — 10:00 am   Coffee Break

SESSION TA1 — Room C101
SYSTEM ARCHITECTURE I

10:00 am — 12:20 pm

Chair: A.K. Bejczy, Jet Propulsion Laboratory

10:00 — 10:40
Task Allocation Among Multiple Intelligent Robots
L. Gasser and G.A. Bekey, University of Southern California

10:40 — 11:20
Robot Design for a Vacuum Environment
S. Belinsky, W. Trento, R. Imani-Shikhabadi, and S. Hackwood, University of California at Santa Barbara

11:20 — 11:40
Execution Environment for Intelligent Real-Time Control Systems
J. Sztipanovits, Vanderbilt University

11:40 — 12:00
Distributed Intelligence for Supervisory Control
W.J. Wolfe and S.D. Raney, Martin Marietta Denver Aerospace

12:00 — 12:20
An Architecture for Heuristic Control of Real-Time Processes
P. Raulefs and P.W. Thorndyke, FMC Corporation

SESSION TA2 — Room C102
MANIPULATOR CONTROL I

Chair: H. Seraji, Jet Propulsion Laboratory

10:00 — 10:40
Adaptive Force-Position Control for Teleoperated Manipulators
A.J. Koivo, Purdue University

10:40 — 11:00
Adaptive Control of Dual-Arm Robots
H. Seraji, Jet Propulsion Laboratory

11:00 — 11:20
Control Strategy for a Dual-Arm Maneuverable Space Robot
P.K.C. Wang, University of California at Los Angeles

11:20 — 11:40
Nonlinear Feedback Control of Multiple Robot Arms
T.J. Tarn and X. Yan, Washington University
A.K. Bejczy, Jet Propulsion Laboratory

11:40 — 12:00
Dynamics and Control of Coordinated Multiple Manipulators
S.A. Hayati, Jet Propulsion Laboratory

12:00 — 12:20
Design of a Reconfigurable Modular Manipulator System
D. Schmitz and T. Kanade, Carnegie-Mellon University

SESSION TA3 — Room C101
SPATIAL AND GEOMETRIC REPRESENTATION AND REASONING

Chair: I. Oppenheim, Carnegie Mellon University

10:00 — 10:20
Telerobot Task Planning and Reasoning: Introduction to JPL Artificial Intelligence Research
D.J. Atkinson, Jet Propulsion Laboratory

10:20 — 11:00
Geometric Reasoning
R.F. Woodbury and I.J. Oppenheim, Carnegie Mellon University

11:00 — 11:40
A Qualitative Approach for Recovering Relative Depths in Dynamic Scenes
S.M. Haynes and R. Jain, University of Michigan

11:40 — 12:20
A Framework for Qualitative Reasoning About Solid Objects
E. Davis, Courant Institute, New York University

12:20 pm — 1:30 pm   Lunch Break

SESSION TA4 Room C104
MAN/MACHINE INTERFACE I

Chair: T. B. Sheridan, Massachusetts Institute of Technology

10:00 — 10:40
Issues in Human/Computer Control of Dexterous Remote Hands
K. Salisbury, Massachusetts Institute of Technology

10:40 — 11:00
Recent Results of Integrated Telerobotic Systems Tests*
D. Aiken, Massachusetts Institute of Technology

11:00 — 11:20
Task Level Testing of the JPL-OMV Smart End Effector
B. Hannaford, Jet Propulsion Laboratory

11:20 — 11:40
A Natural Language Interface to a Mobile Robot
S. Michalowski, V. A. Medical Center, Palo Alto
C. Crangle and L. Liang, Stanford University

11:40 — 12:00
Types of Verbal Interaction with Instructable Robots
C. Crangle and P. Suppes, Stanford University
S. Michalowski, VA Medical Center, Palo Alto

12:00 — 12:20
Design Development and Evaluation of an EVA Prehensor*
L. Leifer, J. Jameson, M. LeBlanc, D. Wilson, E. E. Sabelman, and D. Schwandt, Stanford University

*No paper received for proceedings

Technical Sessions, Tuesday Afternoon, January 20

SESSION TP1 — Room C101
SYSTEM ARCHITECTURE II

1:30 pm — 4:30 pm

Chair: C.A. Bekey, University of Southern California

1:30 — 2:10
Hierarchical Control of Intelligent Machines Applied to Space Station Telerobots
J.S. Albus, R. Lunia, and H. McCain, National Bureau of Standards

2:10 — 2:50
Electronic Prototyping
J. Hopcroft, Cornell University

2:50 — 3:10
Programming Methodology for a General Purpose Automation Controller
M.C. Sturzenbecker, J.U. Korein, and R.H. Taylor, IBM T.J. Watson Research Center

3:10 — 3:30
Real-Time Hierarchically Distributed Processing Network Interaction Simulation
W. Zimmerman and C. Wu, Jet Propulsion Laboratory

3:30 — 4:30
Panel Discussion

SESSION TP2 — Room C102
MANIPULATOR CONTROL II

Chair: S. Dubowsky, Massachusetts Institute of Technology

1:30 — 2:10
A Survey of Adaptive Control Technology in Robotics
D. Tesar and S. Tosunoglu, University of Texas at Austin

2:10 — 2:30
The Operational Space Formulation and Artificial Potential Field Concept in Robot Control*
O. Khatib, Stanford University

2:30 — 2:50
Simple Robust Control Laws for Robot Manipulators, Part I: Non-adaptive Case
J.T. Wen and D.S. Bayard, Jet Propulsion Laboratory

2:50 — 3:10
Simple Robust Control Laws for Robot Manipulators, Part II: Adaptive Case
D.S. Bayard and J.T. Wen, Jet Propulsion Laboratory

3:10 — 3:30
Algorithms for Adaptive Control of Two-Arm Flexible Manipulators Under Uncertainty
J.M. Skowronski, University of Queensland, Australia

3:30 — 4:30
Panel Discussion

SESSION TP3 — Room C103
TRAJECTORY PLANNING FOR MANIPULATORS

Chair: A. Sanderson, Robotics Institute, Carnegie-Mellon University

1:30 — 2:10
The Use of 3-D Sensing Techniques for On-Line Collision-Free Path Planning
V. Hayward, S. Aubry, and Z. Jasiukajc, McGill University

2:10 — 2:30
Collision-Free Trajectory Planning Algorithm for Manipulators
F. Pourboghrat and J. Han, Southern Illinois University

2:30 — 3:10
Task Planning and Control Synthesis for Robotic Manipulation in Space Applications
A.C. Sanderson, M.A. Peshkin, and L.S. Homem-de-Mello, Carnegie-Mellon University

3:10 — 3:30
Using Automatic Robot Programming for Space Telerobotics
E. Mazer, J. Jones, A. Lanusse, T Lozano-Perez, P. O'Donnell, and P. Tournassoud, Massachusetts Institute of Technology

3:30 — 4:30
Panel Dicussion

5:00 pm — 5:30 pm	Bus to JPL
5:30 pm — 7:00 pm	Visit to JPL
7:00 pm — 8:00 pm	Reception, von Karman Auditorium, JPL

*No paper received for proceedings

SESSION TP4 Room C104
MAN/MACHINE INTERFACE II

Chair: T.B. Sheridan, Massachusetts Institute of Technology

1:30 — 2:10
MIT Research in Telerobotics
T.B. Sheridan, Massachusetts Institute of Technology

2:10 — 2:50
Telerobotics: A Simulation Facility for University Research
L. Stark et al., University of California at Berkeley

2:50 — 3:10
Mixed Initiative Control of Intelligent Systems
G.C. Borchardt, Jet Propulsion Laboratory

3:10 — 3:30
Report on the Stanford/Ames Direct-Link Space Suit Prehensor
J.W. Jameson, Stanford University

3:30 — 4:30
Panel Discussion

SESSION TP5 Room C105
ROBOT SIMULATION AND CONTROL

Chair: W.A. Wolovich, Brown University

1:30 — 2:10
Inverse Kinematic-Based Robot Control
W.A. Wolovich and K.W. Flueckiger, Brown University

Spatially Random Models, Estimation Theory, and Robot Arm Dynamics
G. Rodriguez, Jet Propulsion Laboratory

2:10 — 2:30
Task Oriented Nonlinear Control Laws for Telerobotic Assembly Operations
R.A. Walker, L.S. Ward, and C.F. Ells, Integrated Systems Inc.

2:30 — 2:50
A Universal Six-Joint Robot Controller
D.G. Bihn and T. C. Hsia, University of California at Davis

2:50 — 3:10
Real-Time Graphic Simulation for Space Telerobotics Applications
E.W. Baumann, McDonnell Douglas

3:10 — 3:30
Manipulator Control and Mechanization: A Telerobot Subsystem
S. Hayati and B. Wilcox, Jet Propulsion Laboratory

3:30 — 4:30
Panel Discussion

*No paper received for proceedings

Technical Sessions, Wednesday Morning, January 21

7:30 am — 8:00 am Registration

8:00 am — 8:45 am PLENARY SESSION — Little Theatre
Peter Friedland, Ames Research Center
Building Intelligent Systems — AI Research at NASA Ames

SESSION WA1 Room C101
SYSTEM ARCHITECTURE III

8:50 am — 12:10 pm

Chair: R.A. Volz, University of Michigan

8:50 — 9:30
An Approach to Distributed Execution of ADA Programs
R.A. Volz, P. Krishnan, and R. Theriault, University of Michigan

9:30 — 10:10
Computational Structures for Robotic Computations
C.S.G. Lee and P.R. Chang, Purdue University

10:10 — 10:30
A Run-Time Control Architecture for the JPL Telerobot
J. Balaram, A. Lokshin, K. Kreutz, and J. Beahan
Jet Propulsion Laboratory

10:30 — 10:50
Implementation of the JPL Run-Time Controller*
J. Balaram, J. Beahan, K. Kreutz, and A. Lokshin,
Jet Propulsion Laboratory

10:50 — 11:10
Distributed Control Architecture for Real-Time Telerobotic Operation
H.L. Martin, P. Satterlee, Jr., and R.F. Spille,
Telerobotics International, Inc.

11:10 — 11:30
High Performance Architecture for Robot Control
E. Syler and J. Peterson, Grumman Space Systems Division

11:30 — 12:10
Robot Action Planning from Geometric Databases*
T. Premack, Goddard Space Flight Center

SESSION WA2 Room C102
MANIPULATOR CONTROL III

Chair: J.Y.S. Luh, Clemson University

8:50 — 9:30
Problems and Research Issues Associated With the Hybrid Control of Force and Displacement
R.P. Paul, University of Pennsylvania

9:30 — 9:50
Adaptive Hybrid Control of Manipulators
H. Seraji, Jet Propulsion Laboratory

9:50 — 10:10
Adaptive Hybrid Position/Force Control of Robotic Manipulators
F. Pourboghrat, Southern Illinois University

10:10 — 10:30
Chaos Motion in Robot Manipulators
A. Lokshin and M. Zak, Jet Propulsion Laboratory

10:30 — 10:50
Reduced Model Adaptive Inverse Control for Accurate Robot Arm Path Tracking*
S. Lee, University of Southern California

10:50 — 11:10
Effect of Control Sampling Rates on Model-Based Manipulator Control Schemes
P. Khosla, Carnegie-Mellon University

11:10 — 11:30
Flexible Manipulator Control Experiments and Analysis
S. Yurkovich, Ü. Özgüner, A. Tzes and
P.T. Kotnick, Ohio State University

11:30 — 11:50
Tracking 3-D Body Motion for Docking and Robot Control
M. Donath, B. Sorenson, G. Ben Yang, and R. Starr,
University of Minnesota

11:50 — 12:10
Kinematically Redundant Robot Manipulators
J. Baillieul, R. Brockett, J. Hollerbach, D. Martin,
R. Percy, and R. Thomas, Scientific Systems, Inc.

SESSION WA3 Room C103
PLANNING AND SCHEDULING BASIC RESEARCH AND TOOLS I

Chair: P. Cheeseman, Ames Research Center

8:50 — 9:30
Overview of Planning/Scheduling Problems and Procedures*
P. Cheeseman, Ames Research Center

9:30 — 9:50
Planning and Scheduling: Is There a Difference?
K.G. Kempf, FMC

9:50 — 10.10
The Mechanisms of Temporal Inference
B.R. Fox and S.R. Green, McDonnell Douglas Research Laboratories

10:10 — 10:50
Contingent Plan Structures for Spacecraft
M. Drummond, K. Currie, and A. Tate, University of Edinburgh

10:50 — 11:30
Reasoning and Planning in Dynamic Domains: an Experiment With a Mobile Robot
M.P. Georgeff, A.L. Lansky, and M.J. Schoppers,
SRI International

11:30 — 12:10
Route Planning in a Four-Dimensional Environment
M.G. Slack and D.P. Miller, Virginia Polytechnic Institute

12:10 pm — 1:15 pm Lunch Break

*No paper received for proceedings

SESSION WA4 Room C104
SENSING AND PERCEPTION I

Chair: T. Kanade, Carnegie-Mellon University

8:50 — 9:30
Possibilities and Problems of Computer Vision for Space Applications*
R. Nevatia and S. Lee, University of Southern California

9:30 — 10:10
The Sensing and Perception Subsystem of the NASA Research Telerobot
B. Wilcox, D.B. Gennery, B. Bon, and T. Litwin, Jet Propulsion Laboratory

10:10 — 10:30
Knowledge-Based Vision for Space Station Object Motion Detection, Recognition, and Tracking
P. Symosek, D. Panda, S. Yalamanchili, and W. Wehner, III, Honeywell Systems and Research Center

10:30 — 10:50
Sensor Systems Tested for Telerobotic Navigation
A.W. Thiele, D.E. Gjellum, R.H. Rattner, and D. Manouchehri, Rockwell International Science Center

10:50 — 11:10
Monovision Techniques for Telerobots
P.W. Goode and K. Cornils, LaRC
D.M. Krauss, University of South Florida

11:10 — 11:30
A Vision System for Recognizing and Tracking Known Objects*
W.J. Wolfe and G. Duane, Martin Marietta Denver Aerospace
M. Nathan, University of Colorado at Boulder

11:30 — 12:10
The CMU NAVLAB Experiment*
T. Kanade, Carnegie-Mellon University

*No paper received for proceedings

Technical Sessions, Wednesday Afternoon, January 21

1:15 pm — 2:00 pm

PLENARY SESSION — Little Theatre
John Oberright, Goddard Space Flight Center
Space Station Flight Telerobotics Service Program Overview

SESSION WP1 Room C101
SYSTEM CONCEPTS

2:00 pm — 6:20 pm

Chair: D. Pivirotto, Jet Propulsion Laboratory

2:00 — 2:20
Space Assembly Maintenance and Servicing*
Capt. J. Wong, SDYOD, Los Angeles Air Force Station

2:20 — 2:40
Space Telerobotics Systems: Applications and Concepts
L. Jenkins, Johnson Space Center

2:40 — 3:00
Technology Requirements of Telerobotic Satellite Servicing in Space*
H.F. Meissinger, TRW Federal Systems Division

3:00 — 3:20
Laboratory Testing of Candidate Robotic Applications for Space
R.B. Purves, Boeing Aerospace Corporation

3:20 — 3:40
Telerobotic Assembly of Space Station Truss Structure
G. Fischer, Grumman Space Systems

3:40 — 4:00
Robotic Mobile Servicing Platform for Space Station
S.H. Loewenthal and L. Van Erden, Lockheed Missiles and Space Corp.

4:00 — 4:20
System Engineering Techniques for Establishing Balanced Design and Performance Guidelines for the Advanced Telerobotics Testbed
W.F. Zimmerman and J.R. Matijevic, Jet Propulsion Laboratory

4:20 — 4:40
Dedicated Robotic Servicing for the Space Station System
R.F. Thompson, G. Arnold, and D. Gutow, Rockwell International Corporation

4:40 — 5:00
Shuttle Bay Telerobotics Demonstration
W. Chum and P. Cogeos, Martin Marietta Denver Aerospace

5:00 — 5:20
Telerobotics: Research Needs for Evolving Space Stations
L. Stark, University of California at Berkeley

5:20 — 6:20
Panel Discussion

SESSION WP2 Room C102
MANIPULATOR CONTROL IV

Chair: R. Paul, University of Pennsylvania

2:00 — 2:40
Geometric Foundations of the Theory of Feedback Equivalence
R. Hermann, Ames Research Center

2:40 — 3:00
Reducing Model Uncertainty Effects in Flexible Manipulators Through the Addition of Passive Damping
T.E. Alberts, Old Dominion University

3:00 — 3:20
Parallel Processing Architecture for Computing Differential Kinematic Equations of PUMA Arm
T.C. Hsia, University of California at Davis
G.Z. Lu and W.H. Han, Nankai University, China

3:20 — 3:40
Manipulator Control by Exact Linearization
K. Kreutz, Jet Propulsion Laboratory

3:40 — 4:00
A Virtual Manipulator Model for Space Robotic Systems
S. Dubowsky and Z. Vafa, Massachusetts Institute of Technology

4:00 — 4:20
Model Reduction for Discrete Bilinear Systems
A.M. King and R.E. Skelton, Purdue University

4:20 — 4:40
High Gain Feedback and Telerobotic Tracking
D.E. Koditschek, Yale University

4:40 — 5:00
Multi-Limbed Locomotion Systems for Space Construction and Maintenance
K.J. Waldron and C.A. Klein, Ohio State University

5:00 — 5:20
Concept Development of a Tendon Arm Manipulator and Anthropomorphic Robotic Hand
C.T. Tolman, AMETEK

5:20 — 6:20
Panel Discussion

SESSION WP3 Room C103
PLANNING AND SCHEDULING BASIC RESEARCH AND TOOLS II

Chair: P. Cheeseman, Ames Research Center

2:00 — 2:40
Prediction and Causal Reasoning in Planning
T. Dean and M. Boddy, Brown University

2:40 — 3:20
The Generic Task Toolset: High Level Languages for the Construction of Planning and Problem Solving Systems
B. Chandrasekaran, J. Josephson, and D. Herman, Ohio State University

3:20 — 3:40
Monitoring Robot Actions for Error Detection and Recovery
M. Gini and R. Smith, University of Minnesota

3:40 — 4:20
Recovering From Execution Errors in SIPE
D.E. Wilkins, SRI International

4:20 — 4:40
Incremental Planning to Control a Blackboard-Based Problem Solver
E.H. Durfee and V.R. Lesser, University of Massachusetts at Amherst

4:40 — 5:00
Planning in Automatic Robot Planning*
G.F. DeCosta, Northrop Research and Technology

5:00 — 5:20
Knowledge Representation System for Assembly Using Robots
A. Jain and M. Donath, University of Minnesota

5:20 — 6:20
Panel Discussion

6:30 pm — 7:00 pm No-Host Reception, International Ballroom, Hilton Hotel

7:00 pm — 9:00 pm Banquet, International Ballroom, Hilton Hotel

*No paper received for proceedings

SESSION WP4 Room C104
SENSING AND PERCEPTION II

Chair: T. Kanade, Carnegie-Mellon University

2:00 — 2:40
Object Apprehension Using Vision and Touch
R. Bajcsy and S.A. Stansfield, University of Pennsylvania

2:40 — 3:10
Sensory Substitution for Space Gloves and for Space Robots
P. Bach-y-Rita, J.G. Webster, and W.J. Tompkins, University of Wisconsin
T. Crabb, Astronautics Corp. of America

3:10 — 3:40
Dual Arm Robotic System with Sensory Input
Ü. Özgüner, Ohio State University

3:40 — 4:00
Multiple Degree of Freedom Optical Pattern Recognition
D. Casasent, Carnegie-Mellon University

4:00 — 4:40
Maximum Likelihood Estimation of Parameterized 3-D Surfaces Using a Moving Camera
Y. Hung, B. Cernuschi-Frias, and D.B. Cooper, Brown University

4:40 — 5:00
The Architecture of a Video Image Processor for the Space Station
S. Yalamanchili, D. Lee, K. Fritze, T. Carpenter, K. Hoyme, and N. Murray, Honeywell Systems and Research Center

5:00 — 5:20
An Optical Processor for Object Recognition and Tracking
J.A. Sloan and S. Udomkesmalee, Perkin-Elmer Corporation

5:20 — 6:20
Panel Discussion

SESSION WP5 Room C105
TELEROBOTS

Chair: J.E. Pennington, Langley Research Center

2:00 — 2:40
A Flexible Telerobotic System for Space Operations
N.O. Orlando Sliwa and R.W. Will, LaRC

2:40 — 3:20
Systems Simulations Supporting NASA Telerobotics
F.W. Harrison, Jr. and J.E. Pennington, LaRC

3:20 — 3:40
Coordination of Multiple Robot Arms
L.K. Barker and D. Soloway, LaRC

3:40 — 4:00
Development of Semi-autonomous Service Robots With Telerobotic Capabilities
J.E. Jones, Colorado School of Mines
D.R. White, MTS Systems Corporation

4:00 — 4:20
A Task-Based Metric for Telerobotic Performance Assessment
J.F. Barnes, Martin Marietta Denver Aerospace

4:20 — 4:40
Communications in Hierarchical Systems*
M. Simon

4:40 — 5:00
Environmental Modeling and Recognition for an Autonomous Land Vehicle
D.T. Lawton, T.S. Levitt, C.C. McConnell, and P.C. Nelson, Advanced Decision Systems

5:00 — 5:20
Metalevel Programming in Robotics: Some Issues
A. Kumar N. and N. Parmeswaran, Indian Institute of Technology

5:20 — 6:20
Panel Discussion

*No paper received for proceedings

Technical Sessions, Thursday Morning, January 22

7:30 am — 8:00 am	Registration
8:00 am — 8:45 am	PLENARY SESSION — Little Theatre **Jack E. Pennington, Langley Research Center** Overview of Langley Space Telerobotics Program
8:45 am — 9:00 am	Coffee Break

SESSION THA1 Room C101
SYSTEM ARCHITECTURE IV

9:00 am — 12:00 noon

Chair: A. Meystel, Drexel University

9:00 — 9:40
Software and Hardware for Intelligent Robots
G.N. Saridis, Rensselaer Polytechnic Institute
K.P. Valavanis, Northeastern University

9:40 — 10:20
Nested Hierarchical Controller with Partial Autonomy
A. Meystel, Drexel University

10:20 — 10:40
A Space Systems Perspective of Graphics Simulation Integration
R. Brown, C. Gott and G. Sabionski,
Johnson Space Center
D. Bochsler, LinCom Corporation

10:40 — 11:00
Application of Operational Experience to Telerobotic System Design*
J.S. Kuehn and E.D. Selle, ORINTEC

11:00 — 11:20
A Hierarchical Algorithm for Manipulator Planning*
M. Thomas and W.J. Wolfe, Martin Marietta
Denver Aerospace

11:20 — 12:00
Panel Discussion

SESSION THA2 Room C102
MANIPULATOR CONTROL V

Chair: A.J. Koivo, Purdue University

9:00 — 9:40
Motion Coordination and Programmable Teleoperation Between Two Industrial Robots
J.Y.S. Luh and Y.F. Zheng, Clemson University

9:40 — 10:00
Use of Control Umbilicals as a Deployment Mode for Free Flying Telerobotic Work Systems
J.S. Kuehn and E.D. Selle, ORINTEC

10:00 — 10:20
The Design of a Nine-String Six-Degree-of-Freedom Force Feedback Joystick for Telemanipulation
M.L. Agronin, Jet Propulsion Laboratory

10:20 — 10:40
A Learning Controller for Nonrepetitive Robotic Operations
W.T. Miller, III, University of New Hampshire

10:40 — 12:00
Panel Discussion

SESSION THA3 Room C103
AUTONOMOUS SYSTEMS

Chair: H. Moravec, Carnegie-Mellon University

9:00 — 9:40
3-D World Modeling Based on Combinational Geo for Autonomous Robot Navigation
M. Goldstein, F.G. Pin, G. de Saussure and C. Weisbin
Oak Ridge National Laboratory

9:40 — 10:20
On Autonomous Terrain Model Acquisition by a Mobile Robot
N.S.V. Rao, S.S. Iyengar and C.R. Weisbin,
Louisiana State University

10:20 — 10:40
Recognizing Noisy, Geometrically Distorted and Partially Occluded Objects*
H. Wechsler, University of Minnesota

10:40 — 11:00
Solon: An Autonomous Vehicle Mission Planner
M. Dudziak, Martin-Marietta Baltimore Aerospace

11:00 — 11:20
Mission Planning for Autonomous Systems
G. Pearson, FMC Corporation

11:20 — 12:00
Certainty Grids for Mobile Robots
H. Moravec, Carnegie-Mellon University

*No paper received for proceedings

12:00 noon — 1:30 pm	Lunch Break

SESSION THA4 Room C104
SENSING AND PERCEPTION III

9:00 am — 12:00 noon

Chair: B. Wilcox, Jet Propulsion Laboratory

9:00 — 9:40
Improved Shape-Signature and Matching Methods for Model-Based Robotic Vision
J.T. Schwartz and H.J. Wolfson, Courant Institute, New York University

9:40 — 10:20
A Technique for 3-D Robot Vision for Space Applications
V. Markandey, H. Tagare and R.J.P. de Figueiredo, Rice University

10:20 — 10:40
Differential Surface Models for Tactile Perception of Shape and On-Line Tracking of Features
H. Hemami, Ohio State University

10:40 — 11:00
Constraint-Based Stereo Matching
D.T. Kuan, FMC Corporation

11:00 — 11:20
A Database/Knowledge Structure for Robotics Vision System
D.W. Deerholt, New Mexico State University

11:20 — 11:40
Sensing and Perception: Connectionist Approaches to "Subcognitive" Computing
J. Skrzypek, University of California at Los Angeles

11:40 — 12:00
Parallel Processing for Digital Picture Comparison
H.D. Cheng and L.T. Kou, University of California at Davis

Technical Sessions, Thursday Afternoon, January 22

1:30 pm — 3:30 pm	Panel Discussion — Little Theatre **Future Research Directions** Moderator: G. A. Bekey, University of Southern California The intent of this session is to provide views on the state of telerobotics research and to draw together a collection of suggested research opportunities to present to NASA. The panel members will give opening statements summarizing the results of earlier sessions and discussion periods. This will be followed by a general discussion with the audience. The moderator will conclude the session by stating a synthesized set of recommendations.
3:30 pm — 5:30 pm	Closing Reception, The Pasadena Center Lobby

APPENDIX B
INDEX BY AUTHOR

Agronin, M.L. II-341
Alberts, T.E. III-307
Albus, J.S. I-155
Arnold, G. I-75
Atkinson, D.J. I-339
Aubry, S. III-119
Bach-y-Rita, P. II-51
Baillieul, J. II-245
Bajcsy, R. II-45
Balaram, J. I-211
Barakat, W. II-413
Barker, L.K. II-301
Barnes, J.F. II-317
Baumann, E.W. II-207
Bayard, D.S. III-215, 231
Beahan, J. I-211
Bejczy, A.K. II-361
Bejczy, A.K. III-179
Bekey, G. I-127
Belinsky, S. I-95
Bergengruen, O. II-413
Bihn, D.G. II-197
Bochsler, D. II-267
Boddy, M. III-49
Bon, B. II-3
Borchardt, G.C. II-423
Braddi, L. II-413
Brockett, R. II-245
Brown, R. II-267
Byler, E. I-233
Carpenter, T. II-85
Casasent, D. II-65
Cernuschi-Frias, B. II-71
Chandrasekaran, B. III-59
Chang, P.R. I-199
Cheng, H.D. II-155
Chun, W. I-83
Cogeos, P. I-83
Cooper, D.B. II-71
Cornils, K. II-23
Crabb, T. II-51
Crangle, C. II-381, 393
Currie, K. III-17
Davis, E. I-369
Dean, T. III-49
Dearholt, D.W. II-135
deFigueiredo, R.J.P. II-111
de Saussure, G. I-273
Donath, M. II-31
Donath, M. III-101
Doshi, R.S. III-115
Drummond, M. III-17
Dubowsky, S. III-335
Dudziak, M.J. I-289
Durfee, E.H. III-91
Eisenberg, J. II-413
Elia, C.F. II-185
Ellis, S. II-413
Ethier, S. II-413
Fischer, G. I-45
Flora, D. II-413
Flueckiger, K.F. II-165
Fox, B.R. III-9
Friedland, P. I-19
Fritze, K. II-85
Gasser, L. I-127
Gennery, D.B. II-3
Georgeff, M.P. III-27
Gidwani, S. II-413
Gini, M. III-67
Gjellum, D.E. II-19
Goldstein, M. I-273
Gonzales, N.N. II-135
Goode, P.W. II-23
Gott, C. II-267
Green, S.R. III-9
Gutow, D. I-75
Hackwood, S. I-95
Han, J.Y. III-125
Han, W.H. III-317
Hannaford, B. II-371, 413
Harrison, F.W., Jr. II-293
Hayati, S. II-219
Hayati, S.A. III-193
Haynes, S.M. I-361
Hayward, V. III-119
Heglie, R. II-413
Hemami, H. II-125
Herman, D. III-59
Hermann, R. III-299
Hoemem-de-Mello, L.S. III-129
Hollerbach, J. II-245
Hopcroft, J. II-59
Hoyme, K. II-85
Hsia, T.C. II-197
Hsia, T.C. III-317
Hung, Y. II-71
Imani-Shikhabadi, R. I-95
Iyengar, S.S. I-283
Jain, A. III-101
Jain, R. I-361
Jameson, J.W. II-433
Jasiukajc, Z. III-119
Jenkins, L. I-29
Jones, J. III-139
Jones, J.E. II-307
Jordan, T. II-413
Josephson, J. III-59

Kanade, T. III-171
Kempf, D.G. III-3
Khosla, P.K. II-235
Kim, N.H. II-413
Kim, W. II-413
King, A.M. III-345
Klein, C.A. I-105
Koditschek, D.E. III-355
Koivo, A.J. III-155
Korein, J.U. I-167
Kotnik, P.T. III-279
Kou, L.T. II-155
Kreutz, K. I-211
Kreutz, K. III-325
Krishnan, P. I-187
Kuan, D.T. II-131
Kuehn, J.S. II-335
Kumar N., A. I-181
Lansky, A.L. III-27
Lanusse, A. III-139
Lawton, D.T. I-313
Lee, C.S.G. I-199
Lee, D. II-85
Lesser, V.R. III-91
Levitt, T.S I-313
Liang, L. II-381
Litwin, T. II-3
Lokshin, A. I-211
Lokshin, A. II-229
Lowenthal, S.H. I-55
Lozano-Perez, T. III-139
Lu, G.Z. III-317
Luh, J.Y.S. II-325
Lum, H. I-19
Lumia, R. I-155
Manouchehri, D. II-19
Markandey, V. II-111
Martel, B. II-413
Martin, D. II-245
Martin, H.L. I-223
Matijevic, J.R. I-67
Mazer, E. III-139
McCain, H. I-155
McConnell, C.C. I-313
Meystel, A. I-251
Michalowski, S. II-381, 393
Miller, D.P. III-41
Miller, W.T., III II-273
Misplon, M. II-413
Moore, E. II-413
Moore, S. II-413
Moravec, H.P. I-307
Murray, N. II-85
Nelson, P.C. I-313
Nguyen, A. II-413
Nguyen, C. II-413
C onnell, P. III-139
Oppenheim, I.J. I-351
Orlosky, S. II-413
Ozguner, U. III-279, 289
Panda, D. II-9
Parameswaran, N. I-181
Patel, G. II-413
Paul, R.P. III-255
Pearson, G. I-303
Pennington, J.E. II-293
Percy, R. II-245
Peshkin, M.A. III-129
Peterson, J. I-233
Pin, F.G. I-273
Pourboghrat, F. III-125, 273
Purvis, R.B. I-35
Raney, S.D. I-139
Rao, N.S.V. I-283
Rattner, R.H. II-19
Raulefs, P. I-149
Rizzi, M. II-413
Rodriguez, G. II-177
Sabionski, G. II-267
Salisbury, K. II-351
Sanderson, A.C. III-129
Saridis, G.N. I-241
Satterlee, P.E., Jr. I-223
Schenker, P.S. I-3
Schmitz, D. III-171
Schoppers, M.J. III-27
Schwartz, J.T. II-103
Selle, E.D. II-335
Seraji, H. III-153,
159, 261
Shaffer, E. II-413
Sheridan, T.B. II-403
Skelton, R.E. III-345
Skowronski, J.M. III-245
Skrzypek, J. II-145
Slack, M.G. III-41
Sliwa, N.O. II-285
Sloan, J. II-95
Smith, R. III-67
Soloway, D. II-301
Sorensen, B. II-31
Spille, R.F. I-223
Stansfield, S.A. II-45
Stark, L. I-91
Stark, L. II-413
Starr, R. II-31
Sturzenbecker, M.C. I-167
Suppes, P. II-393

Sutter, M.	II-413
Symosek, P.	II-9
Sztipanovits, J.	I-131
Tagare, H.	II-111
Tarn, T.J.	III-179
Tate, A.	III-17
Taylor, R.H.	I-167
Tendick, F.	II-413
Tesar, D.	III-205
Theriault, R.	I-187
Thiele, A.W.	II-19
Thomas, R	II-245
Thompson, R.F.	I-75
Thorndyke, P.W.	I-149
Tolman, C.T.	III-365
Tompkins, W.J.	II-51
Tosunoglu, S.	III-205
Tournassoud, P.	III-139
Trento, W.	I-95
Tyler, M.	II-413
Tzes, A.	III-279
Udomkesmalee, S.	II-95
Vafa, Z.	III-335
Valavanis, K.P.	I-241
Van Erden, L.	I-55
Volz, R.A.	I-187
Waldron, K.J.	I-105
Walker, R.A.	II-185
Wang, P.K.C.	II-257
Ward, L.S.	II-185
Webster, J.G.	II-51
Wehner, W. III	II-9
Weisbin, C.R.	I-273, 283
Wen, J.T.	III-215, 231
White, D.R.	II-307
Wilcox, B.	II-3, 219
Wilkins, D.E.	III-79
Will, R.W.	II-285
Wolfe, W.J.	I-139
Wolfson, H.J.	II-103
Wolovich, W.A.	II-165
Wong, H.	II-413
Woodbury, R.F.	I-351
Wu, C.	I-173
Yalamanchili, S.	II-9, 85
Yang, G.B.	II-31
Yun, X.	III-179
Yurkovich, S.	III-279
Zak, M.	II-229
Zheng, Y.F.	II-325
Zimmerman, W.F.	I-67, 173

APPENDIX C

ATTENDEES/PARTICIPANTS

Adams, Richard H.
Dover Sargent Co.
Central Research Laboratories
P.O. Box 75
Red Wing, MN 55066

Ahmad, Shaheen
Purdue Univ.
School of Electrical Engrg.
West Lafayette, IN 47907

Aiken, David A.
Massachusetts Inst. of Technology,
Space Systems Lab
Cambridge, MA 02139

Alberts, Thomas E.
Dept. of Mech. Engrg.
Old Dominion Univ.
Norfolk, VA 23508

Albus, James
National Bureau of Standards
Bldg. 220, Room B-124
Gaithersburg, MD 20899

Alexander, Harold L.
Standford Univ. Room 250
Durand Bldg.
Stanford, CA 94305

Anselmo, Victor J.
Rand Corporation
1700 Main Street
Santa Monica, CA 90406

Antonsson, Erik K.
Mechanical Engrg. Dept. 104-44
California Institute of Technology
Pasadena, CA 91125

Antsaklis, Panagiotis J.
Dept. of Electrical and Computer
Engrg.
Univ. of Notre Dame
Notre Dame, IN 46556

*Argonin, Mike L.
JPL MS 251

Askew, Scott R.
NASA Johnson Space Center
ES 63
Houston, TX 77058

*Atkinson, David J.
JPL MS 251

Bach-Y-Rita, Paul
Univ. of Wisconsin, Rehab Med.
1415 Johnson Drive
Madison, WI. 53706

*Bachman, William E.
JPL MS 198-112

Bahram, Ravani
Univ. of Wisconsin-Madison
Space Automation & Robotics
Mech. Engrg. Dept.
1513 University Ave.
Madison, WI 53706

Bajczy, Ruzena
Univ. of Pennsylvania
Cmptr & Info Science Dept.
200 S. 33RD St.
Philadelphia, PA 19104

*Balaram, J.
JPL MS 198-330

Barnes, John
Martin-Marietta Corp.
Denver Division
M/S 0427, P.O. Box 179
Denver, CO 80123

Bauer, Frank H.
NASA Goddard Space Center
Code 712.3
Greenbelt, MD 20771

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

PAGE 408 INTENTIONALLY BLANK

Baumann, Erv W.
McDonnell Douglas, Aerospace
Information Services Co.
Bldg. 271D/1/G13, Dept. W314
P.O. Box 516
St. Louis, MO 63166

*Bayard, David S.
JPL MS 198-330

*Beahan, John
JPL MS 198-330

Begy, Steve
Begy Corporation
5 Claret Ash
Littleton, CO 80127

*Bejczy, Antal K.
JPL MS 198-330

Bekey, George A.
Computer Science Dept.
Univ. of Southern Calif.
University Park
Los Angeles, CA 90089-0782

*Bell, Charles E.
JPL MS 198-326

Bender, Welcome W.
Center for Robotics in
Microelectronics
Univ. of Calif., Santa Barbara
Santa Barbara, CA 93106

Bennett, William S.
The Aerospace Corp.
P.O. Box 3430
Sunnyvale, CA 94086-3430

Berenji, Hamid R.
NASA Ames Research Center/Heer Assoc.
M/S 244-7
Moffett Field, CA 94035

*Biefeld, Eric W.
JPL MS 510-202

Bihn, Daniel
Hewlett Packard
Info Technology Group Core Systems
P.O. Box 44035
Cupertino, CA 95015-4035

Blanco, Wilfredo G.
Goddard Space Flight Center
Code 116.2
Greenbelt Road
Greenbelt, MD 20771

Bochsler, Daniel
Lincom Corporation
18100 Upper Bay Road
Suite 208
Houston, TX 77058

*Bon, Bruce
JPL MS 23

*Borchardt, Gary C.
JPL MS 168-522

Bradford, Kayland
Martin-Marietta Corporation
Denver Division, M/S 0427
P.O. Box 179, Adv. Automation
Technology Group
Denver, CO 80123

*Briggs, Hugh (Clark)
JPL MS 185-105

Burman, Dennis A.
Athtec Systems, Inc.
4225 Northgate Blvd., #4
Sacramento, CA 95834

Byler, Eric A.
Grumman Aerospace
M/S A09-25,
Bethpage, NY 11714

*Cameron, Jonathan M.
JPL MS 23

Cannon, David J.
Consultant
84 Escondido Village
Stanford, CA 94305

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Cannon, Robert
Aeronautics and Astronautics
Stanford Univ.
Stanford, CA 94305

Carpenter, Gil C.
Grumman Aerospace
M/S A08-35
Bethpage, NY 11714

Casasent, David
Carnegie-Mellon Univ.
Dept. ECE
Pittsburgh, PA
15213

Cass, Dwight E.
Northrop Research and Technology Ctr.
One Research Park,
Palos Verdes, CA 90274

Catie, Randy
Digital Equipment Corp.
8301 Prof. Pl.
Landover, MD 20904

*Chafin, Roy L.
JPL MS 278

Cheeseman, Peter
Ames Research Center
MS 244-7,
Moffett Field, CA 94035

*Chen, Gun-Shing
JPL MS 157-410

Chitty, Richard
Fairchild Space Company
20301 Century Blvd.
Germantown, MD 20874

Clarke, Margaret M.
Rockwell International
12214 Lakewood Blvd.
Downey, CA 90241

Cleland, John G.
Research Triangle Inst.
Research Triangle Park, NC 27709

Cogeos, Paul
Martin Marietta,
P.O. Box 179
Denver, CO 80201

Cooper, David
Brown Univ.
Div. of Engrg
Providence, RI 02912

*Cooper, Brian K.
JPL MS 23

Crangle, Colleen
Stanford Univ., Medical Center
IMSSS, Ventura Hall
Stanford, CA 94305

Croft, John W.
NASA Goddard Space Flight Center,
Code 712.3, Bldg. 11
Greenbelt, MD 20771

Da Costa, Francis
Northrop, 1 Research Park
Palos Verdes, CA 90274

*Dahlgren, John
JPL MS 198-330

Danielson, Roger T.
Lockheed Missiles & Space Co.
Org. 57-10, Bldg. 529
Sunnyvale, CA 94088

Davis, Robert E.
NASA Goddard Space Flight Center
Code 408
Greenbelt, MD 20771

Davis, Ernest
New York Univ., Courant Inst.
251 Mercer St.
New York, NY 10012

Dean, Thomas
Brown Univ., Dept. of Computer
Science
Box 1910
Providence, RI 02912

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Dearholt, Don
New Mexico State Univ.
Computing Research Lab
Dept. of Computer Science
Box 3CU, NMSU
Las Cruces, NM 88003

Delpech, Michel
Mechanical Engrg. Dept., M/S 104-44
California Institute of Technology
Pasadena, CA 91125

*Diner, Daniel B.
JPL MS T-1201

Divona, Charles
NSA Electro-Mechanical Systems Div.
2701 Saturn Street
Brea, CA 92621

*Dominey, Peter
JPL MS T-1300

Donath, Max
Univ. of Minnesota
111 Church St., S.E.
Minneapolis, MN 55455

*Doshi, Rajkumar S.
JPL MS 510-202

*Doyle, Richard J.
JPL MS 510-202

Drummond, Mark
Artificial Intell. Appls. Inst.
Univ. of Edinburgh
80 South Bridge
Edinburgh EH1 1HN, UK

Dubowsky, Steven
Mechanical Engrg.
Room 3-469
Massachusetts Institute of Technology
Cambridge, MA 02139

Dudziak, Martin J.
Martin Marietta Baltimore
Aerospace, M/S E-315
103 Chesapeake Park Plaza
Baltimore, MD 21220

Dunagan, Brad C.
Consultant
5360 Repecho Dr.
M-107
San Diego, CA 92124

Durfee, Edmund H.
Univ. of Massachusetts
Dept. of Computer and
Information Science
Amherst, MA 01003

Eaton, Wayne A.
NASA Johnson Space Center
NASA RD 1
Houston, TX 77058

Edwards, Laurence J.
Stanford Univ.
Center for Design Research
Terman Engrg. Center
Stanford, CA 94305

*Eggemeyer, William C.
JPL MS 301-250

*Ehrlich, Raymond L.
JPL MS 506-328

Elia, Curtis F.
Integrated Systems, Inc.
101 University Ave.
Palo Alto, CA 94301

Endo, Scott M.
Advanced Technology, Inc.
Suite 105
2120 San Diego Avenue
San Diego, CA 92110

Engle, Steven W.
Expert-Ease Systems
1301 Shoreway Rd.
Belmont, CA 94002

Erwin, Harry
NASA Headquarters, Code MT
Washington, DC 20546

*Fang, Wai Chi
JPL MS 111-208

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Farrell, James D.
Robotics Research Corp.
20008 Ford Circle
Milford, CA 45150

Fernhout, Paul D.
Princeton Univ.
Robotics and Expert Systems Lab
E220 E-QUA, Civil Engrg. Dept.
Princeton, NJ 08544

Fini, John N.
Metex, Inc.
6301 Stevenson Ave., #3111
Alexandria, VA 22304

Fischer, Grahme
Grumman Corp., A09-25
Space Systems Division
Bethpage, NY 11714-3588

Flatav, Carl R.
Telerobotics, Inc.
45 Knickerbocker Avenue
Bohemia, NY 11216

Fox, Barry R.
McDonnell Douglas, Dpt. 225
Bldg. 305/Level 2E, P.O. Box 516
St. Louis, MO 63166

*French, Robert L.
JPL MS 23

Friedland, Peter
Ames Research Center
M/S 244-7
Artificial Intelligence Branch
Moffett Field, CA 94035

Friedman, Mark B.
Carnegie Mellon Univ.
Robotics Institute
Pittsburgh, PA 15213

Fung, Patrick
Spar Aerospace
1700 Armont Drive
Weston, Ontario, CN M9L-2W7

Gardner, Bruce E.
The Aerospace Corp.
M4/915, P.O. Box 92957
Los Angeles, CA 90009

Gasser, Les
Univ. of Southern California
Computer Science Dept.
Los Angeles, CA 90089-0782

*Gennery, Donald B.
JPL MS 23

*Gershman, Bob
JPL MS 264-122

Gilbert, Ray L.
NASA Headquarters, Code I
600 Independence, S.W.
Washington, D.C. 20546

Gilstrap, Vance
U.S. Air Force Space Div.
1733 Ford Avenue
Redondo Beach, CA 90278

Gini, Maria
Univ. of Minnesota
Dept. of Computer Science
136 Lind Hall
207 Church St., SE
Minneapolis, MN 55455-0191

Goforth, Andy
NASA Ames Research Center
M/S 244-7, Bldg. 244
Moffett Field, CA 94035

Goldstein, Moshe
Oak Ridge National Laboratory
Bldg. 6025
P.O. Box X
Oak Ridge, TN 37831-6364

Golub, Alexander
TRW, Space and Technology Group
82/2325
One Space Park
Redondo Beach, CA 90278

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Goode, Plesent W.
NASA Langley Research Center
ISD Autom. Tech Branch
Bldg. 1268A, Room 1150
Hampton, VA 23665-5225

Gott, Charles
NASA Johnson Space Center
FM 7, NASA RD. 1
Houston, TX 77058

Green, David
Tracor Aerospace, Inc.
M/S 28-7
6500 Tracor Lane
Austin, TX 78725-2070

Green, Stanley R.
McDonnell Douglas Astronautics Co.
Dept. F882
P.O. Box 21233
Kennedy Space Center, FL 32815

Grover, Jeffrey L.
Georgia Inst. of Technology
Research Inst.
347 Ferst Drive, N.W.
Atlanta, GA 30332

Guida, Giovanni
Dipartimento di Elettronica
Politecnico di Milano
20133 Milano (Italia)
Piazza Leonardo DaVinci 32, IT

Gutow, David
Rockwell, Rocketdyne Div.
AC-43, 6633 Canoga Ave.
Canoga Park, CA 91303

Hackwood, Susan
Univ. of California, Santa Barbara
Center for Robotic Systems
Santa Barbara, CA 93116

*Gyanfi, Max A.
MS 301-285

*Hadaegh, Fred
JPL MS 198-330

Haley, Dennis C.
Martin Marietta Denver Aerospace
M/S 0428
P.O. Box 179
Denver, CO 80201

Hamel, William R.
Oak Ridge National Laboratory
Bldg. 3500, P.O. Box X
Oak Ridge, TN 37831

Han, Jia-Yuan
Dept. of Electrical Engrg.
Southern Illinois Univ.
Carbondale, IL

*Hannaford, Blake
JPL MS 198-330

*Hansen, Bert
JPL MS 185-105

Harkins, Robert H.
Martin Marietta Denver Aerospace
20 Mesa Oak
Littleton, CO 80127

Harmon, Scott
Robot Intelligence International
4660 Long Branch Avenue
San Diego, CA 92107

Harrigan, Raymond
Sandia National Labs
Div. 1411
Albuquerque, NM 87185

Hayward, Vincent
McGill Univ., Dept. of Elec. Engrg.
3980 University
Montreal, Quebec, CA H3A 2A7

Healey, Anthony J.
Naval Postgraduate School
Code 69WH
Monterey, CA 93943

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Heckathorne, Craig W.
Northwestern Univ. Rehab.
Engrg. Program, Room 1441
345 Superior St.
Chicago, IL 60611

Heer, Ewald
Heer Associates
5329 Crown Avenue
La Canada, CA 91011

Hemami, Hooshang
The Ohio State Univ.
Dept. of Elec. Engrg.
2015 Neil Ave.
Columbus, OH 43210

Hennessey, Michael P.
FMC/Northern Ordnance, C4012
4800 East River Road
Minneapolis, MN 55421

Herberg, Joseph R.
Westinghouse Electric
105 Mall Blvd.
Expo Mart Bay 200W
Monroeville, PA 15146

Hermann, Robert
53 Jordan Road
Brookline, MA 02146

Herndon, Joseph N.
Oak Ridge National Laboratory
Bldg. 3546, P.O. Box X
Oak Ridge, TN 37831

Hirai, Shigeoko
Univ. of California, Cntr for
Robot. Syst. in Microelec.
6740 Cortona Drive
Goleta, CA 93117

Hodge, David C.
Human Engineering Laboratory
Bldg. 459, Robotics Sciences &
Military Appl. Team
Aberdeen Proving Ground, MD
21005-5001

*Hollander, Mike
JPL MS 301-250D

Hollis, Ralph L.
IBM Research, P.O. Box 218
Yorktown Hts, NY 10598

Homem-de-Mello, Luiz S.
Carnegie Mellon Univ.
P.O. Box 112
Pittsburgh, PA 15213-3890

Hopcroft, John
Cornell Univ., Dept. of
Computer Science
311 Upson Hall
Ithaca, NY 14853

*Horvath, Joan C.
JPL MS 138-208

Houpt, Paul
General Electric Corp.
Research and Development
Room 37-2031
1 River Road
Schenectady, NY 12345

Hsia, Tien C.
Univ. of California, Davis
Dept. of Electrical and
Computer Engrg.
Davis, CA 95616

Hunter, David
Space Mechanics Directorate
Dept. of Comm., Canada
Communications Research Centre
3701 Carling Ave.
Ottawa, Ontario, CN K2H 852

*Ih, Che-Hang (Charles)
JPL MS 198-330

Ioannou, Petros A.
Univ. of So. California
MC 0781, EE Systems, SAL 300
Los Angeles, CA 90089

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Iyenger, Sitharama S.
Louisiana State Univ.
Dept. of Computer Science
Baton Rouge, LA 70805

Jain, Santeev
Athtec Systems Inc.
4225 Northgate Blvd., #4
Sacramento, CA 95833

Jain, Anil
Univ. of Minnesota
Dept. of Mech. Engrg.
111 Church Street., S.E.
Minneapolis, MN 55455

Jain, Ramesh
Univ. of Michigan
Dept. of Elec. Engrg. and
Computer Science
Ann Arbor, MI 48109

Jarvis, John F.
AT&T Bell Laboratories
Crawfords Corner Road
Holmdel, NJ 07733

*Jau, Bruno
JPL MS 23

Jelatis, Demetrius G.
Central Research Laboratories
Dover Sargent Co.
P.O. Box 75
Red Wing, MN 55066

Jenkin, Alan
Aerospace Corp., D8 M4/972
2350 E. El Segundo Blvd.
El Segundo, CA 90245

Jenkins, Lyle M.
NASA Johnson Space Center
Code EX2, Bldg. 15, Room 126B
Houston, TX 77058

Jennings, Mel
Lockheed Missiles and Space Co.
1111 Lockheed Way, 0/55-30/B/591
Sunnyvale, CA 94089-3504

Jones, Jerry E.
American Welding Inst.
New Topside Rd.
Route 4, Box 90
Louisville, TN 37777

Josephson, John
Ohio State Univ., Lab for AI
Research, CIS Dept.
228 CAE Bldg., 2036 Neil Ave.
Columbus, OH 43210

Julian, Ron
Armstrong Aerospace Medical
Research Lab, AAMRL/BBA
Wright Patterson AFB, OH 45433-6573

*Kan, Edwin P.
JPL MS 23

Kanade, Takeo
Carnegie-Mellon Univ.
Schenley Park
Pittsburgh, PA 15213

Karlen, James P.
Robotics Research Corp.
2000B Ford Circle
Technecenter
Milford, OH 45150

Kempf, K.
FMC Corporation
1205 Coleman Avenue
Santa Clara, CA 95052

Keshavan, Kesh
Northrop Research and Tech Center
One Research Park
Palos Verdes, CA 90274

Khosla, Pradeep K.
Carnegie Mellon Univ.
Schenley Park
Pittsburgh, PA 15213

King, Francis G.
Ford Motor Co.
5305 Wing Lake Road
Bloomfield Hills, MI 48013

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

King, Andrew M.
Purdue Univ., School of
Aero. & Astro., Grissom Hall
West Lafayette, IN 47907

Koditschek, Daniel E.
Yale Univ., Dept. of
Electrical Eng.
P.O. Box 2157, Yale Station
New Haven, CT 06520-2157

Koivo, A. J.
Purdue Univ., Dept. of
Electrical Engrg.
West Lafayette, IN 47906

Koningstein, Ross
Stanford Univ., Dept. of
Aeronautics
Stanford, CA 94305

*Kreutz, Kenneth
JPL MS 198-330

Kuan, Darwin
FMC Corporation
1205 Coleman Avenue
Santa Clara, CA 95052

Kuehn, Judson
Orintec
P.O. Box 3092
Vallejo, CA 94590

Kumar, Vijay
Ohio State Univ.
Columbus, OH 43210

*Lam, Raymond K.
JPL MS 510-202

Lane, Frank
Fluor Technology, Inc.
Adv. Technology Div.
3333 Michelson Drive
Irvine, CA 92730

Lansky, Amy L.
SRI International, AI Center
333 Ravenswood Ave.
Menlo Park, CA 94025

Larsen, Ronald L.
Univ. of Maryland
Central Administration
Adelphia, MD 20783

Larson, Tim
Odetics
1515 S. Manchester Ave.
Anaheim, CA 92801-2907

Lawton, Daryl
Advanced Decision Systems
201 San Antonio Circle/286
Mountain View, CA 94043

*Lawton, Teri
JPL MS 198-330

Leach, Eugene
Caterpiller, Inc., Research, TC-A
100 N.E. Adams St.
Peoria, IL 61629

*Lear, Tom
JPL MS 278

Leaver, Scott
Univ. of California, Irvine
616 Engineering
Irvine, CA 92717

Lee, Albert W.
TRW
16064 Gallatin St.
Fountain Valley, CA 92708

Lee, Sukhan
Univ. of Southern California
University Park
Los Angeles, CA 90089

Lee, C.S. George
Purdue Univ.
West Lafayette, IN 47907

*Lee, Jun Ji
JPL MS 111-208

*Lee, Roger A.
JPL MS 238-208

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

*Lee, Thomas S.
JPL MS 301-285

Leifer, Larry J.
Stanford Univ., ME Design Div.
Stanford, CA 94305-3030

Leventhal, Lance A.
Consortium for Space Automation
and Robotics
11722-D2 Sorrento Valley Rd.
San Diego, CA 92121

Lewis, Andrew M.
Princeton Univ. Robotics and
Expert Systems Lab, E220 E-QUA
Civil Engrg. Dept.
Princeton, NJ 08544

*Lewis, Stephen L.
JPL MS 198-326

*Li, Yinghan P.
JPL MS 510-264

Lin, Chun-Shin
San Diego State Univ.
Dept. of Electrical &
Computer Engrg.
San Diego, CA 92182

*Liu, Hua-Kuang
JPL MS 248-100

*Kiu, K.Y.
JPL MS 156-246

*Livingston, Floyd R.
JPL MS 507-201

Loewenthal, Stuart H.
Lockheed Missiles & Space Co.
Bldg. 551, Dept. 62-14
Sunnyvale, CA 94088

*Lokshin, Anatole
JPL MS 198-330

Long, Arlie D.
NASA Goddard Flight Center
Org. 735.2, Bldg. 11, Rm 109
Greenbelt, MD 20771

Luh, John
Clemson Univ., Dept. of
Electrical and Computer Engrg.
Clemson, SC 29634

Lumia, Ron
National Bureau of Standards
Room B-124, Meterology Bldg.
Gaithersburg, MD 20899

*Macchio, Gregory
JPL MS 138-208

Marcus, Beth A.
Arthur D. Little, Inc.
20 Acorn Park
Cambridge, MA 02140

Martin, H. Lee
Telerobotics Int. Inc.
8410 Oak Ridge Highway
Knoxville, TN 37931

*Marzell, Neville I.
JPL MS 198-330

*Matijevic, Jacob R.
JPL MS 185

Mazer, Emmanuel
Massachusetts Inst. of Technology
MIT AI Lab
545 Technology Square
Cambridge, MA 02139

McCarthy, J.M.
Univ. of Calif., Irvine
616 Engineering
Irvine, CA 92717

McConnell, Chris
ADS
201 San Antonio Cir.
Suite #205
Mt. View, CA 94040

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

McLaughlin, John S.
Aerospace Corp., MS M4/915
2350 E. El Segundo Blvd.
El Segundo, CA 90245-4651

Meissinger, Hans F.
TRW Electronics and Defense
One Space Park
Redondo Beach, CA 90278

Meldrum, Deirdre
420 N. Polk Street
Moscow, ID 83843

Meystel, Alex
Drexel Univ., Lab of Appl.
Machine Intell. & Robotics
Dept. of ECE
31 St At Market
Philadelphia, PA 19104

Michalowski, Stefan
Stanford Univ., Design Div.
Dept. of Mechanical Engrg.
Stanford, CA 94305

Michelson, Robert C.
Georgia Tech Research Inst.
Raid/TDD, GTRFCC
Atlanta, GA 30332

Miller III, Thomas
Univ. of New Hampshire
Dept. of Electrical and
Cmptr Engrg., Kingsbury Hall
Durham, NH 03824

*Milman, Mark H.
JPL MS 198-330

*Mishkin, Andrew
JPL MS 23

*Mittman, Dave
JPL MS 301-250

Moravac, Hans
Carnegie-Mellon Univ.
Robotics Inst.
Pittsburgh, PA 15213

Myers, Craig J.
Univ. of Tennesee Inst.
Engineering Science Dept.,
Upper G Wing
Tullahoma, TN 37388

Myers, John K.
SRI International
333 Ravenswood
Menlo Park, CA 94025

McLay, Michael J.
Fairchild Space Co.
MS A12
20301 Century Blvd.
Germantown, MO 20855

Nakamura, Yoshihiko
Univ. of Santa Barbara
Santa Barbara, CA 93106

Naser, Joseph A.
Electric Power Research Inst.
3412 Hillview Avenue
P.O. Box 10412
Palo Alto, CA 94303

Nease, Ardell
Rockwell International
Space Station Syst. Div., M/S AE 99
12214 Lakewood Blvd.
Downey, CA 90241

Nevatia, Ramakant
Univ. of Southern California
PHE 204, Dept. of Electrical Engrg.
Los Angeles, CA 90089-0273

Newbauer, John
AIAA
1633 Broadway
New York, NY 10019

*Nguyen, Tam
JPL MS 278

*Nickersen, Jack A.
JPL MS 144-201

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Drive, Pasadena, CA 91109

Oberright, John
Goddard Space Flight Center
19 Watkins Park Dr.
Greenbelt, MD 20772

Ollendorf, Stan
Goddard Space Flight Center
Greenbelt, MD 20771

Olsen, Roy E.
Grumman Aerospace, A09-25
Bethpage, NY 11714

Olson, Noreen S.
FMC Corporation
4800 E. River Rd.
Minneapolis, MN 55412

Oppenheim, Irving
Carnegie-Mellon Univ.
Dept. of Civil Engrg.
Pittsburgh, PA 15213

Ottinger, Brian D.
Westinghouse Electric Corp.
1310 Beulah Rd.
Pittsburgh, PA 15235

Overton, John D.
Eagle Engineering, Inc.
711 Bay Area Blvd., St. 315
Webster, TX 77598

Ozguner, Umit
The Ohio State Univ.
Dept. of Electrical Engrg.
2015 Neil Ave.
Columbus, OH 43210

Passino, Kevin M.
Dept. of Electrical and
Computer Engrg.
Univ. of Notre Dame
Notre Dame, IN 46556

Paul, R.
CIS Dept., Moore School
Univ. of Pennsylvania
Philadelphia, PA 19104

Pearson, Glenn
FMC Corp., Central Eng. Lab
P.O. Box 580
Santa Clara, CA 95052

Pennington, Jack E.
NASA Langley Research Center
M/S 152D, Bldg. 1268A, Rm. 1148
Hampton, VA 23665-5225

Pepper, Ross L.
NOSC
1306 Kainvi Drive
Kailua, HI 96734

*Peters, Steve
JPL MS 301-250

Peterson, John H.
58 King Ave.
Selden, NY 11784

Pfeffer, Lawrence E.
Stanford Univ.
Aero/Astro Dept.
Stanford, CA 94305

Pierson, Paul B.
GE/RCA Advanced Technolgy Labs
Moorestown Corporate Center
RTE. 38
Moorestown, NJ 08057

*Pivirotto, Donna L.
JPL MS 180-701

*Porta, Harry J.
JPL MS 510-202

Pourboghrat, Farzad
Southern Illinois Univ.
Dept. of Elec. Engrg.
Carbondale, IL 62901

Price, Charles R.
Johnson Space Center, EF2
Houston, TX 77089

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Purves, Robert B.
Boeing, JA-68
P.O. Box 1470
Huntsville, AL 35807-3701

Radtke, Robert P.
Tracor
9810 FM 19o0
Bypass-Humble, TX 77339

Rambin, Ronald
Vista Controls Corp.
27825 W. Fremont Court
Valencia, CA 91355

Rao, Nageswara SV
Louisiana State Univ.
Dept. of Computer Science
Baton Rouge, LA 70803

Raulefs, Peter
FMC Corp.
Artificial Intell. Center
P.O. Box 580
Santa Clara, CA 95052

Raymus, Steven D.
General Electric, Astro Space Div.
P.O. Box 800
Princeton, NJ 08540

Regan, Howard
Fairchild Space Co.
20301 Century Blvd.
Germantown, MD 20874

Reynaud, Howard
Communication Research Center
3701 Carling Ave.
P.O. Box 11490, Sta. H
Ottawa, Ontario, CN K2H 8S2

Richardson, Dennis C.
Westinghouse
105 Mall Blvd., Expomart 339E
Monroeville, PA 15146

*Robinson Lee
JPL MS 278

Rogers, Arthur
Integrated Automation Corp.
8300 Professional Place
Landover, MD 20785

*Rokey, Mark J.
JPL MS 301-250

Rosenfeld, Robert
DARPA
1400 Wilson Blvd.
Arlington, CA 2209-2308

Roth, Zvi
Florida Atlantic Univ.
Dept. of ECE
500 S.W. 20th St.
Boca Raton, FL 33431

*Rubenstein, Louis D.
JPL MS 301-250

Russell, Paul
Ford Aerospace and
  Communications Corp.
3939 Fabian Way
Palo Alto, CA 94303

*Salganicoff, Marco
JPL MS 198-330

Salisbury, Kenneth
Massachusetts Inst. of Technology
AI Laboratory
545 Technology Square
Cambridge, MA 02139

Samadani, Ramin
Stanford Univ.
202 Durand Bldg.
Stanford, CA 94305

Sanderson. Arthur
Robotics Inst., Dept. of
  Elec. and Comptr. Engrg.
Schenley Park
Pittsburgh, PA 15213

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Drive, Pasadena, CA 91109

Saridis, George N.
Rensselaer Polytechnic Institute
Dept. of ECSE
Troy, NY 12180-3590

Sauerwein, Timothy A.
NASA Goddard Space Flight Center
Code 735.2
Greenbelt, MD 20771

Schafer, Dr. Bernd
Deutsche Forschungs-und
Versuchsanstalt fur Luft-und
Raum, DFVLR
Oberpfaffenhofen, 8031
Webling, West Germany

Schappell, Roger T.
Martin Marietta Denver Aerospace
MS 0427
P.O. Box 179
Denver, CO 80201

*Scheid, Robert E.
JPL MS 198-330

*Schenker, Paul S.
JPL MS 198-330

Schmitz, Don
Carnegie-Mellon Univ.
Comptr. Science Dept.
Wean Hall 1327
Schenley Park
Pittsburgh, PA 15213

Schmuter, Samson L.
520 East 81st St., Apt 6M
New York, NY 10028

Schnakenberg, George H.
U.S. Dept. of Interior
Bureau of Mines
P.O. Box 18070
Pittsburgh, PA 15236

Schneider, Stanley A.
Stanford Univ.
923 Menlo Avenue, #4
Menlo Park, CA 94025

*Schober, Wayne R.
JPL MS 180-701

Schoenwald, Jeff S.
Rockwell International
1049 Camino Dos Rios
P.O. Box 1085
Thousand Oaks, CA 91360

Schwartz, Jack T.
Courant Institute, New York Univ.
251 Mercer Street
New York City, NY 10012

Scotti, Richard
Ori, Inc.
1375 Piccard Drive
Rockville, MD 20850

Scruggs, Roy M.
Georgia Tech
Electronics Res. Bldg., ECSL/CTAD
Atlanta, GA 30332

Sebok, Kenneth
Perry Offshore, Inc.
275 W. 10th St.
Riviera Beach, FL 33404

Seering, Warren
Massachusetts Inst. of Technology
Mechanical Engrg.
325 Thomas Laboratory
Cambridge, MA 02139

Seidman, L.P.
Ford Aerospace and
Communications Corp.
3939 Fabian Way
Palo Alto, CA 94303

Senger, Robert D.
Westinghouse
1 Carriage Road
Greensburg, PA 15601

*Seraji, Homayoun
JPL MS 23

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Shahidi, Reza
Univ. of Maryland
College Park, MD 20742

Shapiro, Daniel
Advanced Decision Systems
201 San Antonio Circle, Suite 286
Mountain View, CA 94040

Sheridan, Thomas B.
Massachussetts Inst. of Technology
3-346
Cambridge, MA 02139

Shimoyama, Isao
Carnegie Mellon Univ.
Civil Engrg.
Schenley Park
Pittsburgh, PA 15213

Shoemaker, Patrick A
Naval Ocean Systems Center
Code 551
San Diego, CA 92152

Siciliano, Bruno
Universita' de Napoli
Dipartimento di Informatica
e Sistemi
Via Caludio 21
80125 Napoli, Italy

*Sirlin, Sam
JPL MS 198-326

*Sirota, Allen
JPL MS 278

Skowronski, Jan M.
Univ. of Queensland
Physical Sciences & Engrg. Group
St. Lucia, AU 4067

Skrzypek, Josef
UCLA, Machine Perception Lab, CSD
3532D Boelter Hall
Los Angeles, CA 90024

Slack, Marc G.
Virginia Polytechnic Inst.
and State Univ.
Blacksburg. VA 24061

Sliwa, Nancy E.
NASA Langley Research Center
MS 152D
Hampton, VA 23665-5225

Sloan, Jeffrey A.
Perkin-Elmer Corp.
7421 Orangewood Avenue
Garden Grove, CA 92641

Smith, Richard E.
Univ. of Minnesota
Cmptr Science Dept.
136 Lind Hall
207 Church Street, SE
Minneapolis, MN 55455

*Smith, Jeff
JPL MS 301-280

*Snow, Edward R.
JPL MS 278

*Solloway, Carleton B.
JPL MS 510-271

Soloway, Donald
NASA Langley Research Center
MS 152D, Bldg. 1268A, Room 1148
Norfolk, VA 23503

Soroka, Barry I.
Univ. of Southern California
M/C 0781
Los Angeles, CA 90089

*Spagnuolo, John N.
JPL MS 510-202

Spector, Victor A.
TRW
One Space Park
Redondo Beach, CA 90278

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Stark, Lawrence
Univ. of California
Minor Hall
Berkeley, CA 94720

Staunton, Brian D.
The Aerospace Corp.
2350 E. El Segundo Blvd.
El Segundo, CA 90245

*Stephenson, R. Rhoads
JPL MS 98-105

Stokes, Lebarian
NASA Johnson Space Center
M/C ES63
Houston, TX 77058

Stuart, David G.
TRW
R5/2020
One Space Park Blvd.
Redondo Beach, CA 90278

Sturzenbecker, Martin C.
IBM
T.J. Watson Research Center
Yorktown Heights, NY 10598

Sudey, John
NASA Goddard Space Flight Center
Code 716
Greenbelt, MD 20771

Symosek, Peter
Honeywell Systems and Research Center
3660 Technology Drive
Minneapolis, MN 55418

*Szirmay, Steve Z.
JPL MS 198-330

Sztipanovits, Janos
Vanderbilt Univ.
Dept. of Electrical Engrg.
Nashville, TN 34235

Tarn, T.J.
Washington Univ.
Box 1040
St. Louis, MO 63130

Taylor, Edith C.
McDonnell Douglas Astronautics
Co-Houston Div.
16055 Space Center Blvd.
Houston, TX 77062

Taylor III, Eugene A.
Vista Controls Corporation
27825 W. Fremont Court
Valencia, CA 91355

Teeter, Ronald R.
Astronautics Technology Center
5800 Cottage Grove Dr.
Madison, WI 53558

Tesar, Delbert
Univ. of Texas at Austin
Rm. 4.146, Mechanical Engrg. Dept.
Austin, TX 78712

Thiele, Alfred W.
Rockwell International
Science Center
1049 Camino Dos Rios
P.O. Box 1085
Thousand Oaks, CA 91360

Thomas, Donald J.
Westinghouse Electric Corp.
Gov't. Systems Business Devlpmt.
Bldg. 602
1310 Beulah Road
Pittsburgh, PA 15235

Thomas, Richard G.
Scientific Systems, Inc.
One Alewife Place
Cambridge, MA 02140

Thompson, Richard
Rocketdyne Div.
Rockwell International
6637 Canoga Avenue
Canoga Park, CA 91303

Thompson, William
NSA Electro-Mechanical Systems Div.
2701 Saturn St.
Brea, CA 92621

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Thorndyke, Perry W.
FMC Corporation
1205 Coleman Ave.
Box 580
Santa Clara, CA 95052

Tilley, Scott
Ford Aerospace and
Communications Corporations
3939 Fabian Way
Palo Alto, CA 94303

Tollander, Carl
Advanced Decision Systems
201 San Antonio Circle, Suite 286
Mountain View, CA 94040

Tolman, C. Thomas
665 N. La Patera Lane
Goleta, CA 93117

Tosunoglu, Sabri
Univ. of Texas
Mechanical Engrg. Dept.
Austin, TX 78712

*Tso, Kam S.
JPL MS 23

Ullman, Marc A.
Stanford Univ. Dept. of
Aeronautics and Astronautics
Durand Bldg., Room 250
Stanford, CA 94305

Utzerath, James
Astronautics Corp. of America
4115 N. Teutonia Avenue
Milwaukee, WI 53201

Vafa, Zia
Massachusetts Inst. of Technology
305 Memorial Dr.
Cambridge, MA 02139

Valavanis, Kimon P.
Northeastern Univ.
Dept. of ECE
Boston, MA 02115

Varsi, Giulio
NASA Headquarters
Code ST
Washington, DC 20546

*Vasquez, Sharon E.
JPL MS 198-326

Vilain, Marc
BBN Labs
10 Moulton St.
Cambridge, MA 02238

Virag, David
Naval Ocean Systems Center
P.O. Box 997
Kailua, HI 96734

*Volkmer, Kent
JPL MS 506-328

Volz, Richard
Univ. of Michigan
501 East Engineering Bldg.
Ann Arbor, MI 48109

*Von Sydow, Marika
JPL MS 278

Von Wolfsheild, Reichart K.
1227 South Central
Glendale, CA 91204

Vuskovic, Marko
San Diego State Univ.
Dept. of Mathematics
San Diego, CA 92182-0314

Waffenschmidt, Eberhard
Dornier-System GMBH
Postbox 1360
7990 Friedrichshafen/Bode
West Germany

Waldron, Kenneth
Stanford Univ., Design. Div.
Dept. of Mech. Engrg.
Stanford, CA 94305

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

Walker, Robert A.
Integrated Systems, Inc.
101 University Ave.
Palo Alto, CA 94301

*Walters, Tom
JPL MS 180-701

Wang, Paul K.C.
UCLA, Boelter Hall
Dept. of Electrical Engrg., Rm. 6731C
Los Angeles, CA 90024

Ward, Lawrence S.
Integrated Systems, Inc.
101 University Ave.
Palo Alto, CA 94301

Watson, John T.
Eagle Engineering
711 Bay Area Blvd., Ste. 315
Webster, TX 77578

*Weeks, Ralph C.
JPL MS 198-330

Wehner, William
Honeywell Inc.
3660 Technology Drive
Minneapolis, MN 55418

Weisbin, Charles R.
Oak Ridge National Laboratory
Dir. Robotics & Intell. Syst.
Bldg. 6025, P.O. Box X
Oak Ridge, TN 37831-6364

*Wen, John
JPL MS 198-330

Werneth, Russell L.
Navy Postgraduate School
Code 69WH
Monterey, CA 93950

White, Dawn
MTS Systems, Advanced Tech.
Development Div.
Box 24012
Minneapolis, MN 55424

*White, James E.
JPL MS 510-202

*White, Leslie A.
JPL MS 233-301

Widjaja, Davy
Lockheed-EMSCO
2400 NASA Road 1, C-16
Houston, TX 77258

Wiker, Steven F.
Naval Ocean Systems Center
Code 5301
San Diego, CA 92152-5000

*Wilcox, Brian H.
JPL MS 23

Wilkins, David E.
SRI International
333 Ravenswood
Menlo Park, CA 94061

Wilson, David M.
Stanford Univ.
940 Scott St.
Palo Alto, CA 94301

Wolfe, William
Martin Marietta Corp., Denver Div.
M/S 0427, P.O. Box 179
Adv. Automation Technology Group
Denver, CO 80123

Wolovich, William A.
Brown Univ.
182 Hope Street
Providence, RI 02912

Wong, Joseph
Los Angeles Air Force Station
SD/CLT
Los Angeles, CA 90009-2960

*Woo, Raymond
JPL MS 198-330

Woodbury, Robert
Carnegie-Mellon Univ.
Dept. of Architecture
Pittsburgh, PA 15213

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

*Wu, Chung-I
JPL MS 198-133

Yalamanchili, Sudhakar
Honeywell Inc., MN65-2100
3660 Technology Drive
Minneapolis, MN 55418

*Zimmerman, Wayne F.
JPL MS 198-330

de Figueiredo, Rui J. P.
Rice Univ.
Dept. of Elec. & Computer Engrg.
Houston, TX 77251-1892

*Affiliated with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109