October 1989

Control Framework for Hand-Arm Coordination

Sanjay Agrawal
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-70.

Control Framework for Hand-Arm Coordination

Abstract

Co-ordination of multiple manipulators requires cooperation at several levels in the control hierarchy. A distributed processing environment with no hardware dependencies except at the motor servo level, would provide a flexible architecture for coordination. A system on these lines is being built to control an articulated hand and an arm. The four levels of control envisaged include a task decomposition level, a planning level, a scheduling level and a server level. The hand will carry both force and tactile sensors, feedback from these are used to provide adaptive control in grasping tasks. The processing of the sensory information is performed by independent processes, with analyzed information being sent to the relevant layer of the system. The manipulators are also controlled by individual processes. All process can open communications with an active process sending commands or data, or receiving them. We describe the scope of the system and the current setup plus future lines of development.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-70.

Control Framework For Hand-Arm Coordination

MS-CIS-89-70
GRASP LAB 195

Sanjay Agrawal

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

October 1989

Acknowledgements:
This research was supported in part by Sandia 75-1055, Airforce grant AFOSR 88-244, AFOSR 88-0966, Army/DAAG-29-84-K-0061, NSF-CER/DCR82-19196 A02, NASA NAG5-1045, ONR SB-35923-0, NIH 1-R01-NS-23636-01, NSF INT85-14199, ARPA N00014-88-K-0630, NATO grant 0024/85, DuPont Corp., Post Office, IBM Corp. and Lord Corp.

Control Framework For Hand – Arm Coordination

Sanjay Agrawal

GRASP Laboratory

Department of Computer and Information Science

University of Pennsylvania

October 12, 1989

Abstract

Co-ordination of multiple manipulators requires cooperation at several levels in the control hierarchy. A distributed processing environment with no hardware dependencies except at the motor servo level, would provide a flexible architecture for coordination. A system on these lines is being built to control an articulated hand and an arm. The four levels of control envisaged include a task decomposition level, a planning level, a scheduling level and a server level. The hand will carry both force and tactile sensors, feedback from these are used to provide adaptive control in grasping tasks. The processing of the sensory information is performed by independent processes, with analyzed information being sent to the relevant layer of the system. The manipulators are also controlled by individual processes. All process can open communications with an active process sending commands or data, or receiving them. We describe the scope of the system and the current setup plus future lines of development.

*Acknowledgements: This research was supported in part by Sandia 75 1055, Airforce grant AFOSR 88 0244, AfOSR 88-0966, Army/DAAG-29-84-K-0061, NSF-CER/DCR82-19196 Ao2, NASA NAG5-1045, ONR SB-35923-0, NIH 1-RO1-NS-23636-01, NSF INT85-14199, ARPA N0014-88-K-0630, NATO grant No.0224/85, DuPont Corp.,Post Office, IBM Corp. and LORD Corp.

1 Introduction

The motivation for this work came from the desire to develop an environment which having multiple levels of sophistication in terms of user programming as well as understanding. Most robotic environments are only programmable and usable at a single level. That is the user is either provided with a programming environment [Lloyd85, Paul86, Vusk88], or is provided with a computational architecture that is suitable for robot control [Nigam85, Wang86, Chen86].

In order to provide user access at various levels of the system, each level was partitioned in terms of its task and functionality requirements. This meant defining an interface between each level, and in addition creating multiple modules for each level. The communications interface would be defined by the receiver and the sender would comply to the definition.

There were some strong reasons to try and break up the control system into independent modules. The notion of a central processing environment, no matter how computationally powerful, would not solve the need for parallelism, and a massively parallel architecture would introduce gross hardware dependancies from which we were trying to get away from in the first place.

The modules could either contain processes that control a manipulatory device, such as an arm, wrist or hand, or they could be processes that read a tactile array or collect the data from a range scanner. In either case they could send data/commands out to multiple devices and receive data/commands from multiple devices too. This system in no way precludes the addition of new sensors, manipulators or processing hardware, and thus serves the need to expand or modify the objects within the robot environment.

In order to simplify and generalize the means of communication between the various modules, we decide to use the standard UDP protocol [Postel80]. Some justification for UDP is provided in [Corke89]. The protocol itself is not intrinsic in the design of the system and thus can be changed with the only requirement being, the ability to broadcast messages over a network using an Internet protocol.

2 Robot System Issues

Trying to build a robot system that can perform simple tasks in a unstructured environments is largely an unsolved problem and most solutions have tended to overconstrain the problem. The initial problem

was the lack of an articulated gripper, of which quite a few now exist some of which are described in [Ulr88, Jacob85, Salis84]. A major issue now, is to be able to perform adroitly making use of information gathered from the environment to utilize these articulated grippers that are mounted on the end of an arm.

There are other projects going on at MIT, JPL, Stanford and Sandia [Stans89] where work is being done on grasping and other manipulatory tasks using a robot arm in tandem with articulated hands. Still unknown environments provides the very daunting aspect of having to deal with numerous uncertainties and to overcome them by obtaining information via sensors. The information obtained about the environment or object being manipulated is not very useful in isolation. Only when it has been inserted into a framework with from where this information can be extracted can we obtain a or *picture* of the environment.

The problem of grasping now balloons from grasping using multiple fingers to dealing with a host of devices, each with their own characteristics. With a large number of devices we have a large amount of information flow within the system. If we adopt a distributed processing environment we need to define a hierarchy of control so that at each level we have a particular processing capability, plus provide coordination between levels. To impose this hierarchy the system now needs to divide itself along conceptual lines without any hard dependancies between components in the system.

We can introduce new devices or processes into such a system, or replace obsolete objects without perturbing the system or having to alter existing links. With distributed processing environment with standard interfaces with a high bandwidth communication channel, we should be able to construct an environment to develop a fairly flexible and intricate hand arm control system.

3 System Description

There are essentially four conceptual components to this robot system, a cognitive component, a perceptual component, a scheduling component and a manipulatory component. This paper deals mainly with the details of the manipulatory component. The other components are an essential part of the system but are touched on more briefly in this paper.

We are dealing with a large number of processes or modules that are linked to each other by the fact that they can send messages to and fro. Provided the speed of message passing is adequate, we can close

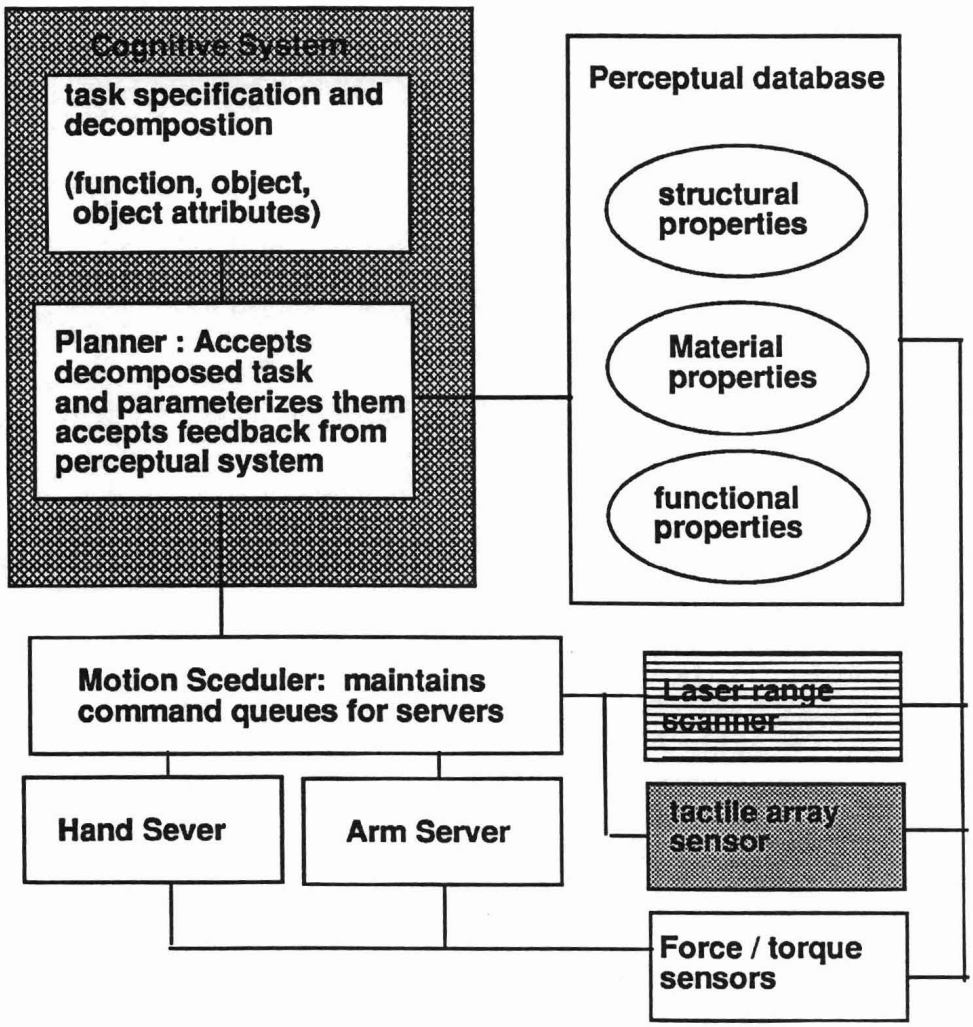

Figure 1: Conceptual Design of Control System

loops about a particular sensory processing module and a manipulator, or close the loop around multiple manipulators and sensors.

Sensor and manipulator modules fall under the manipulatory part of the systems, though modules that process the sensory feedback could either be thought of as belonging to the perceptual component or the manipulatory component depending on whether the processed information was being used to directly control the manipulators, or whether the processing was being used to construct or modify object models stored in the perceptual database.

The next few sections describe the components, detailing the importance of the component and bringing out some of the kinds of interactions that take place between them.

4 Cognitive Component

The cognitive component is the most complex and least defined part of the system. In order to perform the specified task, we need to *understand* the task, the constraints under which it is to be performed, take into account the information we can retrieve and obtain the remaining information. Substantial literature describing of work done in the area of task planning and decomposition exists. In the area of hand arm coordination and primarily grasping, Jeannerod talks about transportation and manipulation as being fairly independent [Jean81], Haun et al, talk about *grasp posture* as a function of stability and manipulabilty [Liu89]. In other work Marteniuk et al suggest that there is an invariability in the reaching component of prehension tasks, though, though there are variances in peak velocities and acceleration profiles [Mar87]. In [Arbib83] they discuss the flow of perceptual information through visual and tactile feedback while performing motor control.

These studies and others indicate that humans seem to decompose tasks into different phases, some of which overlap. Also the decomposition often results in segmentation where there minimal dependancies between two separate segments. Thus in a robot environment we can attempt to segment tasks in a similar manner and to each segment of the task, link a set of sensory information required to perform that segment. If the information is not available a priori i.e. from the perceptual database component, then we need to close a loop around the segment and the sensors that can provide the information. Once the task is segmented the planner needs to set up a framework for the task segments, to allow it to monitor the progress of the individual segments, as well as make any changes if need be to the ordering of the segments.

4.1 Task Specification

At the highest level we provide a task specification level, where the user can input a function that specifies the task to be performed, an object related to the task and associated object attributes. This interface provides a top level which requires a minimal understanding of how the system actually performs, and still has enough generality to allow useful exercises to be performed. The task is specified by means of a function label. The function label represents a set of one or more actions, on the completion of which the task is done. The function labels are self descriptive and are essentially an index into a database of functions that the system contains.

Each action provide the system with a goal for which it must generate a plan for at the next level. The plan will generation is discussed in the planner.

At the task level the system checks whether an object has been specified or not. If the object has not been specified the perceptual database is notified to provide information on possible objects if any that the task could be implicitly specifying.

If on the other hand the object has been specified at the task level, the perceptual database is requested to provide preliminary data on the object from its stored models. This preliminary data would augment any user provided attributes and be sent to the planner.

Thus the task level can be thought of as an interface to any external reasoning or task planning system that can provide functions for the system to perform.

4.2 Planner

The role of the planner is to take a set of actions from the task level specification and fill in the parameters for each action based on the set of object attributes available from the perceptual database. The actions provided to the planner would have a list of parameters that need to be initialized. Typically actions would involve motion of the arm with either a desired position or a desired velocity and for the hand a more complicated setpoint. The action could either specify for the hand to achieve a desired characteristic, where the arm would assist by complying or moving with some velocity, or the action could merely ask the hand to be in a certain configuration when the arm reaches a desired setpoint.

The planner has to formulate a scheme around the set of actions, so that they can be sent to the scheduler and revised if necessary. Revisions to these plans take place, based on the success or failure of previous actions.

The scheduler which is provided with individual queues for each device, keeps the planner informed in case of failure of one the commands in the queue. This enables the planner to decide whether the current queue is still valid, or whether the set of actions need to be revised. The planner does not collect all the sensory feedback, because the feedback is either used directly by the server that controls the manipulator, or by the perceptual database where the processed sensory data is integrated into the current model being used.

5 Perceptual Component

In order to access information about the objects we will be dealing with, we need to have a store of knowledge about them. Object categorization is well studied in the area of human recognition and perception. One paradigm used is natural categories [Ros76]. Objects are classified under superordinate, then divided into basic, and then further divided into subordinate categories. Some of the distinguishing features that place objects in the same category is function and structure. We will use this taxonomy to structure our database and add the material properties as a further distinguishing attribute.

The perceptual system is constructed in a way to enable us to arrange our perceptual information both to perform object recognition, as well as provide operational parameters with reference to objects being manipulated.

5.1 Perceptual Database

One of the key feature of the database is the segmentation along the line of perceptual properties. In addition to perceptual properties, the individual components of the database map neatly onto particular sensory information. There are three components to the database; structural, material and functional. Each component will allow for independent object parameters to be accessed from the database. Thus we can obtain shape and volume to decide the aperture of the hand from the structural database, stiffness and weight of object from the material component of the database, and part and part motion information from the functional database. Multiple object properties can be extracted from haptic sensing [Klat89], thus data obtained from tactile and other contact sensors can provide multiple property descriptions.

The structural description of the database will consist of surface and volumetric description of objects as well as object parts. These descriptions will be provided using superquadrics as the representation.

This representation can be used to model objects utilizing and fusing data from both visual as well as haptic sensors.

The functional description will contain information about the kinds of tasks that the object could be used for. A knife cuts, a hammer hammers, cups can hold liquid would be the kinds of functional relevance that could tell us what object we are dealing with in trying to accomplish the task. In addition functional descriptions can provide information on the best grasp location on the object, or the best grasp to use for a particular object. This grasp information would definitely augment the information we can obtain from the structural and material descriptions. In addition the functional description would contain salient information about moving parts of an object, pivots, and other distinguishing features.

The material description will contain information on intrinsic object properties such as texture and stiffness. These two properties are different in the sense that texture is both a haptic as well as visual property [Led886], where as stiffness or hardness of objects is primarily extracted by touch. Thus the database can be updated by multiple sensors providing the same information, albeit at different resolutions spatially.

6 Scheduler

The planner provides the scheduler with a queue of commands that the devices are to be provided with. The scheduler maintains a separate queue of commands for each separate device within the system. On receiving a command for a device for the scheduler does not have an existing queue, a initialization message is sent first, and on acknowledgement, the scheduler sends out the commands from the devices queue.

The motion queue handler is the heart of the scheduler. On receiving queue of commands for a device from the planner, it translates the commands into queue elements particular to the device, and appends them to the existing queue, or replaces the existing queue by the new commands. If the command is simply to stop the device from what it is doing, the scheduler has to send a halt command to the respective servers. The scheduler is responsible, for maintaining the ordering in the queue, coordinating between the queues, and linking the sensor feedback to the device server.

7 Servers

A server is provided for each device within the system. A device could be a gripper, wrist, arm or a sensor which can be actively commanded. Servers actively monitor the network to check for any commands addressed to them. Each server would have two asynchronous processes running simultaneously. The network process deals with communication over the network, to the scheduler, or a sensor server, that could be providing processed data that is used in controlling the servers device. The device process is engaged in sending commands to the device and monitoring its setpoints. Within the server the two processes would exchange information, using shared variables and buffers to read from and write to.

The servers could run on any machine that had access to the network, be it a PC reading a tactile array, or a systolic array processor running a robot arm. Since the server has a standard interface to the outside world we do not require any specification of the other device. Thus communication is done using the XDR representation which calls for a standard byte ordering in all messages, with machine specific decoding of the data at the host end.

8 Sensors

Sensors within the system can be divided into two categories, intrusive and non-intrusive. Intrusive sensors are those which provide contact data like tactile array sensors or moment/torque sensors.

Non intrusive sensors are devices like laser range scanners, video cameras, that can provide data about the environment without having to be in contact. They typically do not disturb the environment or the objects of interest. These devices can be mounted on an arm and be position to the best advantage of the system, as long as that does not interfere in the manipulatory activity.

There are three modes by which sensors can provide feedback to the manipulators. In the servo feedback mode the sensor is directly monitored by the device process within the server. This mode is associated with position sensors or motor torque sensors where the output is used in the servo loop. This kind of feedback is essentially device specific, and thus not pertinent to the system specification.

In server mode, the feedback is provided over the network by a sensor whose output has been processed, and a specific attribute is sent back to the device server. An example would be tactile sensors that provided a new finger position, by detecting edges, during contour tracing. Another example could be a visual servoing scheme [Corke89], where an object is tracked, by a camera, mounted on a Puma560. Here the

processing of the camera image is done by one process, which provides the arm server with a new position and orientation.

The third mode of sensory feedback is via the perceptual database. In this mode, the loop is closed around the perceptual database and the planner. Processed data is used by the database to modify an existing object model. this modification of the object model, may influence the task plan, in which case the planner, alters the commands sent to the scheduler.

Below are described two sensors and the modes of processing.

8.1 Visual sensing and processing

Vision is the nonintrusive sensory capability provided to the system. Within this framework we use vision to provide us with information which is used to build the perceptual database. The structural component of this database, requires both volumetric and surface information about objects that need to be manipulated.

We use superquadrics to model objects, and use a schema proposed by Gupta [Gupta89] to obtain both volumetric and surface descriptions from these models. Superquadrics is a compact means on representing a wide range of objects. In addition spatially dense data obtained from any device can be provided to the model generator to enhance the surface contours. Laser range data and tactile sensors are sources of such spatially dense data.

The processing schema of visual data or the mode of representation of objects are not part of the system specification. What is needed are adequate descriptions of objects, which are easily modified and stored. The sensor has a server that accepts commands to relay the image, to the processing server. The processing server could be anywhere on the net, and can either command the sensor to provide more images or could be asked to provide an object model to the perceptual database.

Though the system has not been explicitly designed to perform object recognition, it has the capability of doing so. In the case of a task being specified and no object being provided, the functional database would provide a pointer to the structural database to indicate a group of possible object models. The visual server, would then be required to image the scene aiming to extract at least one of the objects in the group selected.

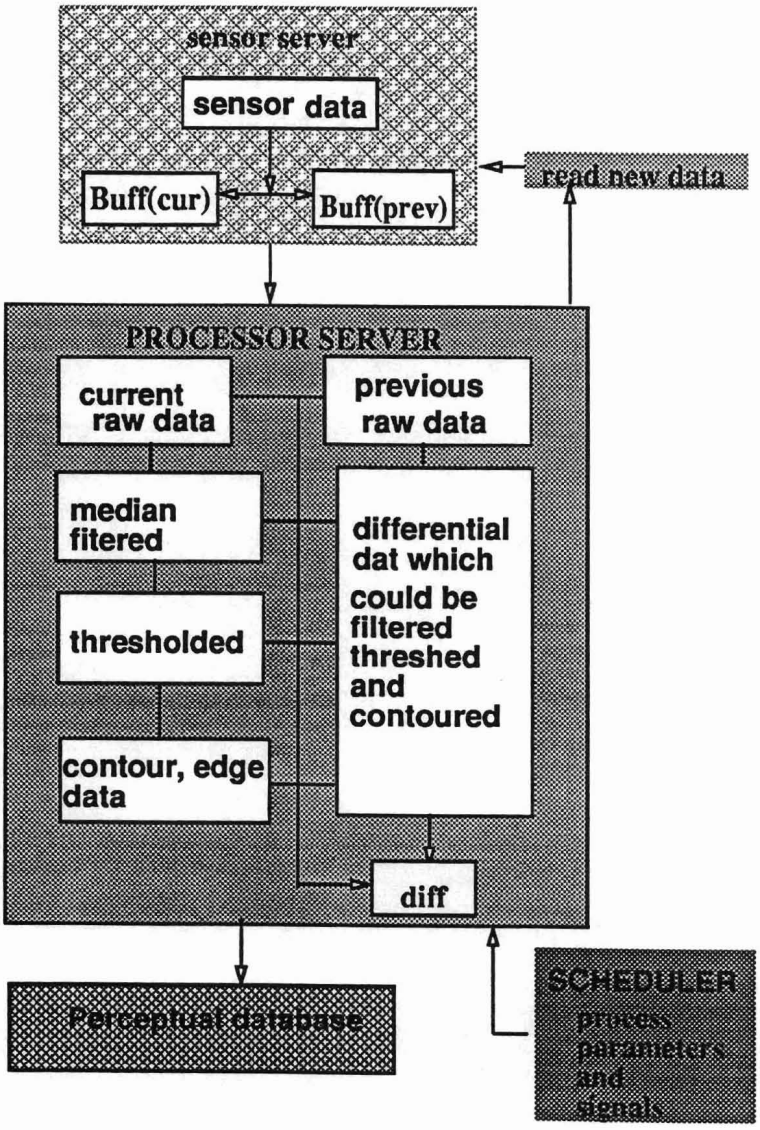

Figure 2: Example of sensor servers

8.2 Tactile sensing and processing

Tactile processing is a crucial intrusive or contact sensor. However it is in the dynamic data gathering and processing area that this system can fruitfully utilize tactile data. One motivation for using tactile data, is to provide edge and surface data to the servers operating the articulated hand in real time. The tactile pads would be placed on the finger tips and palm of the hand, and allow for contour tracing, or surface tracking of objects. In terms of the perceptual database, tactile sensors provide the material properties for objects, such as stiffness and texture.

The processing of tactile data is primarily useful if we can extract edges in real-time. Segmenting the image into a background and foreground to minimize the computation is an important component [Muthu87]. Since tactile feedback is inherently a dynamic process, the environment is constantly being modified even as we obtain data about it. Thus tactile data in conjunction, with position information in world coordinates would be required to accurately construct an object model. Thus along with each tactile image, we need to obtain position and orientation information, from both the arm and the hand. This information can be provided, the arm, hand and tactile servers can synchronize their clocks. Given that we can track edges, we can modify object models, where occlusion has prevented the entire surface description from being generated.

Thus real time processing of tactile data is one of the aims of this system. The tactile image is obtained at a sufficiently high rate from the sensor and then processed by a server that obtains the data and provides edges and patches to the hand server. Again the algorithms used to process the sensors or the sensors themselves are not intrinsic to the system, instead the ability to provide real time edge and patch data to the hand server is crucial.

9 Current Setup

This section describes some of the work that has been done in creating in the individual modules of the system and getting them to work in conjunction with each other. The primary concern was to demonstrate the validity of sending commands over the network to separate servers, and have the devices cooperate in performing tasks. We also attempted to demonstrate the significance of modularizing the various computational processes and provide multiple levels of closed loops to perform the task.

The four levels of the system were created, the task decomposition, the planner, the scheduler and

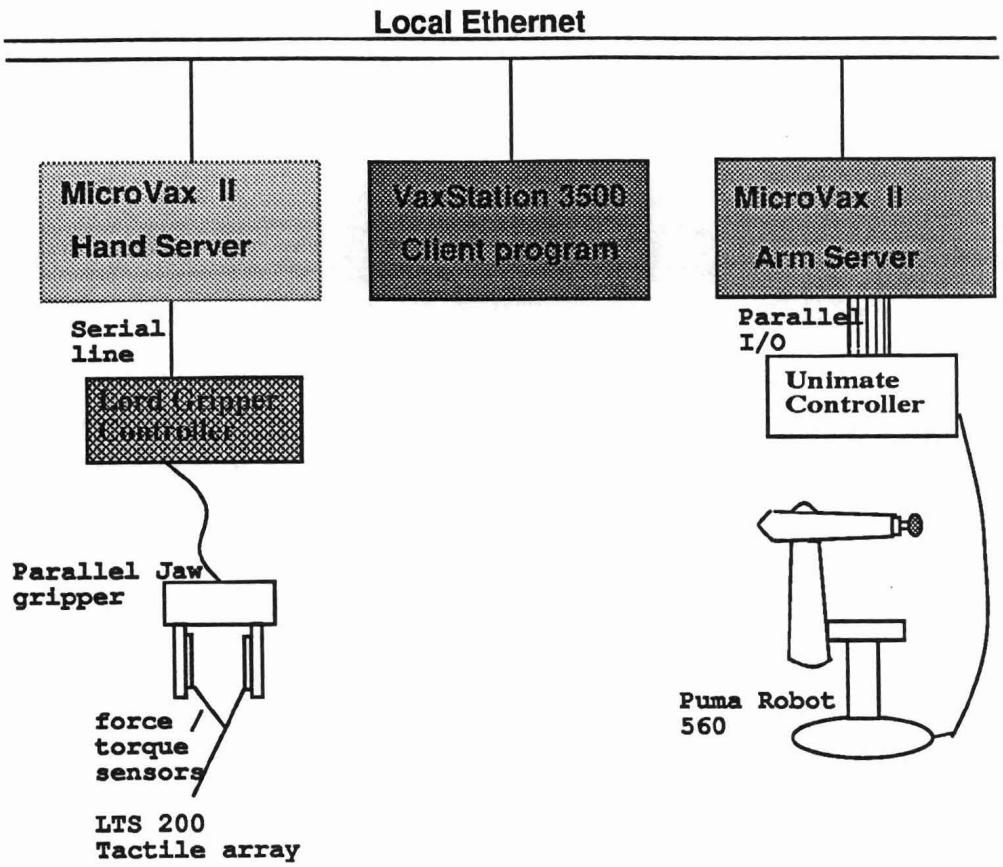

Figure 3: Current Implementation of control system

the server. The set of tasks that we planned to perform were pick and place tasks, where the object location and size and stiffness were specified by the user. Since we did not build the perceptual database, we simplified the set of action parameters to be within the three specified by the user.

The planner consisted of a set of instructions, that determined if the actions were being successfully executed. If the action was not successful, the planner could either issue the command again, with a set of modified parameters, or it could gracefully fail.

The scheduler, accepted a queue for the hand and another for the arm. the queues had elements which depended on the successful completion of each other to provide the coordination, and the scheduler passed the next valid element to the relevant server.

At the server level we had two modules one to send commands to the gripper and the other to send commands to the arm. On receiving the initialization message from the scheduler, the gripper and the arm were put into ready mode.

For each of the modules a input and output interface was defined, and communication established over a local ethernet. Using a local ethernet we found UDP to be very consistent and reliable.

9.1 Hardware and Software

The hardware as shown in figure 3 consists of a Vaxstation 3500, and two MicroVax II's. We used a Puma560 and a Lord Gripper equipped with two LTS-200 force/tactile sensors mounted on the insides of each of the fingers. the fingers had a single degree of freedom, with a maximum aperture of 3 inches. The Puma 560 was controlled by a Unimate controller and the Lord gripper was controlled by a Lord controller.

The gripper was mounted on the wrist of the Puma560. Parallel lines connects the MicroVax running the arm server to the Unimate control box, in order to send the new joint positions to the controller. Similarly a serial line connects the hand server to the Lord controller. An interface was provided to send commands to the Lord controller and receive the current positions and six degrees of force/torque information.

The MicroVax and the Vaxstation were all unix based systems. The task decomposition system and the planner were written in Prolog, while the servers and the scheduler were written in C.

9.2 Hand Server

The hand server has two process running, one that communicates with the lord gripper over a serial line, and the other that talks to any client programs over the network. In the current implementation, the scheduler was the only client. The hand server starts up by opening a socket and listening to the network for any commands. It will accept a numbered command from any client. The command format is based on the number of parameters of the device that can be controlled, and for the gripper, either the desired finger positions, are provided, or the desired force is provided. In addition an object center command can also be provided.

On completing a command the server checks to see if the client was expecting a response, and if so it sends back a message saying that command i, where i is the command number, was successfully completed or failed.

9.3 Arm Server

The arm server is built on similar lines to the hand server. The server starts up the network process and in addition starts up a communication channel with the Unimate controller using the RCI interface [Lloyd85] to send the joint commands. The server will accept commands in either cartesian positions or cartesian velocities in the world coordinates or incremental positions or velocities in the tool frame.

The RCI interface is provided a new joint position every sample period. The inverse jacobian for the puma560 is computed every sampling period and the desired joint angles are obtained using the inverse jacobian and the desired cartesian rate [Corke89, Paul86].

The arm server notifies the client/s of the current wrist position and orientation if the client requests notification. A notification request can specify an update interval when the current command status is sent, or demand the current tool position and orientation in world coordinates. If a joint limit or singularity was encountered the server would be notified by the controller and this information would be provided to the client in turn.

9.4 Results

The pick and place tasks, consisted of grasping cylindrical and cubical shaped objects, from a specific location, and moving them to another point. The coordination was accomplished on a couple of different fronts. We were able to overlap the arm reach motion, with attaining the grasp aperture, based on the object size, so that when we arrived near the object we need not wait for the hand to open. In addition, the delay between the time the hand closed on the object, and the arm began to move away was minimal. Thus it was felt that using a network to communicate between processes, was a feasible method of coordinating control of multiple robot manipulators.

10 Further Research

Research needs to proceed on many fronts. The first task we are working on is to integrate the PENN hand [Ulr88] to take the place of the Lord Gripper, so that we have a articulated hand as one of the devices. Building up the perceptual database and the planner are two large segments of the work. Work on the perceptual database organization is being done as is the work on building the sensor processing environment, in the area of integrating range and tactile data to enhance the object surface descriptions and in extracting edge information in rael time.

References

[Arbib83] Michael A. Arbib, Thea Iberall, and Damion Lyons *Coordinated Control Programs for Movements of the Hand* In COINS Technical Report 83-25, University of Massachusetts, Amherst

[Chen86] J. B. Chen, R. S. Fearing, B. S. Armstrong and J. W. Burdick *NYMPH: A Multiprocessor for Manipulation Applications* Proceedings IEEE Conference on Robotics and Automation, 1986, 1731-1736

[Corke89] Peter I. Corke and Richard Paul *Video-Rate Visual Servoing for Robots* GRASP Lab, University of Pennsylvania Technical Report MS-CIS–89–33

[Gupta89] Alok Gupta. *Part Description and Segmentation Using Contour, Surface and Volumetric Primitives.* Dissertation Proposal, University of Pennsylvania, 1989. Technical Report MS–CIS–89–33.

[Jacob85] S. C. Jacobsen, J. E. Wood, D. F. Knutti, K. B. Biggers and E. K. Iversen The version I Utah/MIT Dextrous Hand Robotics Research: Second International Symposium, ed. H. Hanufsa and H. Inoue, MIT Press, 1985, 39-54

[Jean81] Marc Jeannerod *Intersegmental Coordination During Reaching at Natural Visual Objects* Attention and Performance Vol. 9 Erlbaum, Hillsdale NJ, 1981 153-168

[Klat89] Roberta L. Klatzky, Susan Lederman and Catherine Reed. *Haptic Integration of Object Properties: Texture Hardness and Planar Contour* In Journal of Experimental Psychology: Human Perception and Performance 1989, Vol. 15, No. 1,45-57

[Led886] Susan Lederman, Georgie Thorne and Bill Jones. *Perception of Texture by Vision and Touch: Multidimensionality and Intersensory Integration* In Journal of Experimental Psychology: Human Perception and Performance 1986, Vol. 12, No.2, 169-180

[Liu89] Huan Liu, Thea Iberall and George Bekey *The Multi-dimensional Quality of Task Requirements for Dextrous Robot Hand Control.* IEEE International Conference of Robotics and Automation, 1989

[Lloyd85] John Lloyd *Implementation of a Robot Control Development Environment* Masters Thesis, McGill University, December 1985.

[Mar87] Ronald G. Marteniuk and Christine L. Mackenzie *Invariance and Variability in Human Prehension* Proceedings CIAR, University of Waterloo, May 1987

[Muthu87] Chellappa Muthukrishnan, David Smith, Donald Myers, Jack rebman, Antti Kovio *Edge Detection in tactile Images* Proceedings IEEE Conference on Robotics and Automation, 1987

[Nigam85] R. Nigam and C. S. G. Lee *A multiprocessor-Based controller for the Control of Mechanical Manipulators* IEEE J. of Robotics and Automation, Vol 1, no. 4 Dec. 1985

[Paul86] Richard P. Paul and Hong Zhang *Computationally Efficient Kinematics for Manipulators with Spherical Wrists* International Journal of Robotics Research 5(2) 32-44, 1986

[Postel80] Jon Postel *User Datagram Protocol* SRI International. August 1980. (Sun 800-1054-01)

[Ros76] E. Rosch, C. B. Mervis, W. Gray, D. Johnson and P. Bayes-Braem *Basic Objects in Natural Categories* Cognition and Categorization, (Eds) E. Rosch and B. Lloyds, Erlbaum, Hillsdale NJ

[Salis84] J. K. Salisbury *Design and Control of an Articulated Hand* Proc. International Symposium on Design and Synthesis, Tokyo 1984

[Stans89] Sharon Stansfield *Robotic Grasping of Unknown Objects: A knowledge-Based Approach* Sandia Report, SAN89-1087 UC-32, 1989

[Ulr88] Nathan Ulrich, Richard Paul and Ruzena Bajcsy A Medium Complexity Compliant End-effector Proceedings IEEE Conference on Robotics and Automation, 1988, 434-437

[Vusk88] Marko I. Vuskovic *R-Shell: A Unix based Development Environment for Robotics* Proceedings IEEE Conference on Robotics and Automation, 1988, 457 - 460

[Wang86] Yulin Wang and Steven Butner *A new Architecture for Robot Control* Proceedings IEEE Conference on Robotics and Automation, 1987, 664-670