A System for Programming and Controlling a Multisensor Robotic Hand

PETER K. ALLEN, MEMBER, IEEE,
PAUL MICHELMAN, STUDENT MEMBER, IEEE,
KENNETH S. ROBERTS, STUDENT MEMBER, IEEE

Abstract—**A system for programming and controlling a multisensor robotic hand (Utah-MIT hand) is described. Using this system, a number of autonomous tasks that are easily programmed and include combinations of hand–arm actuation and force, position and tactile sensing have been implemented. The system is controlled at the software level by a programming language (DIAL) that provides an easy method for expressing the parallel operation of robotic devices. It also provides a convenient way to implement task-level scripts that can then be bound to particular sensors, actuators and methods for accomplishing a generic grasping or manipulation task. Experiments using the system to pick up and pour from a pitcher, unscrew a lightbulb, and explore planar surfaces are presented.**

I. Introduction

A current focus of research in robotics is in building systems that perform complex tasks such as inspection, object recognition and assembly. Towards this end, a number of researchers have focused on developing artificial hands that are anthropomorphic in design. Among these hands are the Salisbury hand [12], the Belgrade hand [22] and the Utah-MIT hand [7]. The engineering of these hands has been an enormous effort, and it has provided the research community with a challenge to use these devices within an existing robotic system.

This paper describes our work in building a comprehensive grasping environment, capable of performing tasks such as locating moving objects and picking them up, manipulating manmade objects such as tools, and recognizing unknown objects. It is becoming increasingly clear that robotic systems need to have capabilities similar to the human haptic system in order to perform complex grasping, manipulation and object recognition tasks using dextrous hands. Towards this end, we have designed and built a system for programming and controlling a Utah-MIT hand that contains the following components:

- A set of low-level system primitives that serve as the basis for the control of the hand.
- A true hand/arm system with many degrees of freedom formed by mounting the hand on a robotic manipulator (a PUMA 560).
- Integrated tactile sensors on the fingertips that provide force-sensitive responses and Cartesian position information.
- Sensing primitives that make use of joint position, tendon force, and tactile array sensing in a number of grasping and manipulation tasks.
- A high-level programming front end that allows task-level scripts for common grasping and manipulation tasks to be written easily and cogently, allowing the natural concurrency of such a hand/arm system to be captured at the programming level.

Manuscript received April 15, 1989; revised January 7, 1990. This work was supported in part by DARPA contract N00039-84-C-0165, in part by NSF grants DMC-86-05065, DCI-86-08845, CCR-86-12709, and IRI-86-57151, in part by North American Philips Laboratories, in part by Rockwell Inc., and in part by the AT&T Foundation. The material in this work was partially presented at the IEEE International Conference on Robotics and Automation, May 14–19, Scottsdale, AZ.

The authors are with the Department of Computer Science, Columbia University, New York, NY 10027.

IEEE Log Number 9037605.

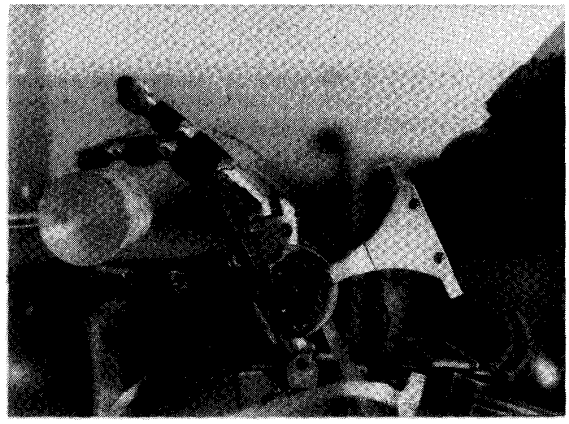

Fig. 1. Utah-MIT hand with tactile sensors mounted.

- A set of haptic exploratory procedures (EP's) that permit active sensing of an object's three-dimensional (3-D) structure.

The main focus of the work described in this paper is in building a system that allows us to integrate hand–arm manipulation and intelligent sensing for a variety of different tasks, using a simple yet powerful programming paradigm. One of the key problems in building robotic systems is to utilize many distributed sensor–actuator devices concurrently. The system described below has integrated a number of actuators (robotic arm and hand) with a powerful sensor suite (joint position, tendon force, tactile arrays) to perform a variety of different tasks, and it forms a framework for implementing other, more complex tasks that include sensing and actuation.

We wish to acknowledge the work of others who have built systems to perform manipulation and grasping tasks. Among these are Geschke's early system to perform integrated robotic manipulation tasks [5], the work of Takase *et al.* [20] in building an integrated robotic teaching and learning system, Salisbury's integrated hand/tactile system [16], Fearing's work with a tactile sensor mounted on a dextrous hand [3], the work of the group at USC integrating the Belgrade hand into an active sensing environment [22], and the original Utah-MIT hand researchers [15] who developed a low-level control system for the hand and a software environment to utilize the low-level control functions.

The outline of this paper is as follows: Section II describes the low-level hardware and software system, Section III describes the high-level task control system, Sections IV–VI describe a number of tasks we have implemented using the system and Section VII summarizes the results.

II. System Overview

The system we have built consists of a Utah-MIT hand attached to a PUMA 560 manipulator. The hand contains four fingers, each with four degrees of freedom. It resembles the human hand in size and shape, but lacks a number of features that humans find very useful. In particular, it has no palmar degree of freedom (closing of the palm) and the thumb is placed directly opposite the other three fingers, with all fingers identical in size (see Fig. 1). The hand has joint position sensors that yield joint angle data and tendon force sensors that measure forces on each of the two tendons (extensor and flexor) that control a joint. The PUMA adds 6 degrees of freedom to the system (three translation parameters to move the hand in space and three rotational parameters to orient the hand), yielding a 22 degree-of-freedom system. Clearly, such a system is a night-

0018-9472/90/1100-1450$01.00 ©1990 IEEE

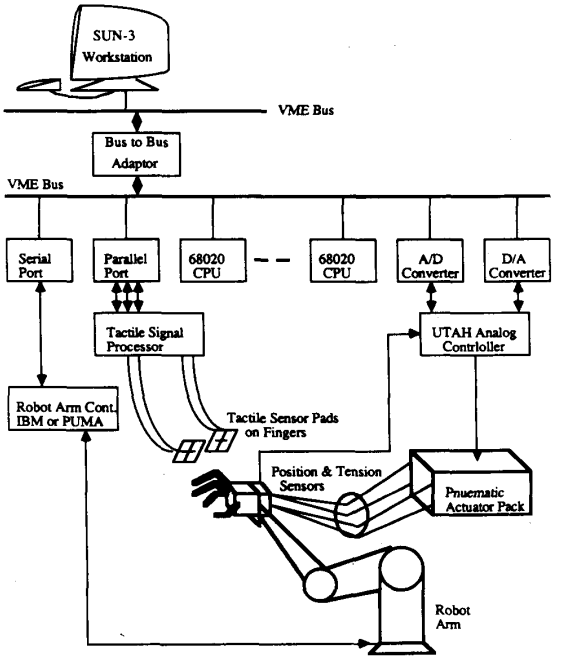

Fig. 2. Hardware overview.

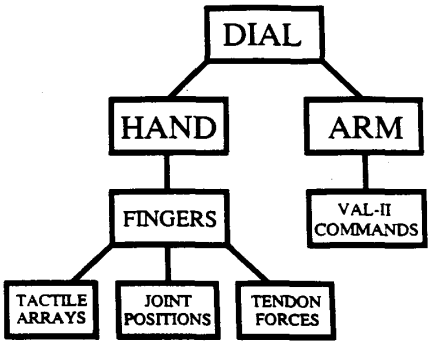

Fig. 3. Software overview.

mare to control at the servo-level in real-time. Our approach is to use the existing controllers in each of these systems rather than to build our own, controlling and communicating with them through an intelligent, high-level controller that links together the movements of arm, hand, and fingers with the feedback sensing of joint positions, tendon forces, and tactile responses on the fingers.

The hardware structure of the system is shown in Fig. 2. The high-level control resides in a SUN-3 processor. The SUN serves as the central controller, and has access to a full UNIX-based system for program development and debugging as well as a set of window-based utilities to allow graphical output and display of the system's various states. The hand is controlled by an analog controller that is commanded through D/A boards from a dedicated 68020 system. The SUN is capable of downloading and executing code on the 68020 and can communicate with it through a shared memory interface [14]. The tactile sensing system is controlled by another dedicated 68020 that monitors the forces on each of the sensor pads. The connection from the SUN to the PUMA is via the VAL-II host control option over a serial interface. The software structure is shown in Fig. 3. The high-level control program is called DIAL and was originally developed by Steven Feiner [4] for use as a graphical animation language. The following sections describe the system in detail.

A. Low-Level System Primitives

The low-level system system primitives are organized along two dimensions, type and domain. Type refers to the type of sensing or actuation (or both) which the primitive implements. Continuous sensing implies a monitoring mode while one-shot sensing implies a static sensor reading. Continuous action implies a synchronized control loop while one-shot action implies an imperative command. Domain refers to the coordinate frame or sensor domain that the primitive operates in. The primitives are:

- GET JOINTS: Reads the joint angles on the hand.
- GET FORCES: Reads the forces (measured at the wrist of the hand) on the flexor and extensor tendons that control each joint.
- GET TACTILE: Reads a 16×16 tactile array on each finger.
- GET WRIST: Reads the Cartesian position of the PUMA wrist.
- GOTO MOVE: A one-shot move that is done atomically. Implemented by setting the desired joint position in the analog control system directly to the final position value of the move for each joint. Move commands have the ability to use symbolic names for specified poses. There is a set of standard pose names available for use by the higher level programming system, particularly for hand pre-shaping operations.
- TRAJECTORY MOVE: Allows a single joint or a number of joints to be moved along an interpolated trajectory from a starting joint space vector to an ending joint space vector. There is also an analogous command that interpolates in Cartesian space for fingertip motion.
- POSITION CONTACT: This is a continuous sensing primitive that is implemented in the Cartesian position domain. When the difference between actual and desired Cartesian fingertip position exceeds a threshold, contact is signaled.
- FORCE CONTACT: Sam as above but tendon forces are monitored for changes. A change in force differentials implies a finger contact.
- TACTILE CONTACT: The tactile arrays are monitored for readings above a prespecified threshold.

B. Composite Functions

A number of composite functions have been built out of the low-level system primitives. Each composite function is described as follows.

- GRASP WITH FORCE: Used to grasp objects with a desired grip strength. The command closes all joints specified by a joint mask incrementally while monitoring the tendon forces controlling the joints in the tendon mask. When the difference between the flexor and extensor tendon forces on a joint exceeds the specified threshold, movement of that joint stops. This process continues until the forces on all of the tendons in the tendon mask have surpassed the threshold. This primitive is useful for grasping objects when their precise dimensions are unknown.
- GUARDED MOVE: This composite function combines one or more of the three low-level contact primitives (POSITION CONTACT, FORCE CONTACT, or TACTILE CONTACT) with either the hand finger motion or motion of the PUMA arm. When a contact is detected, the relevant motion ceases.
- LIMP: This primitive is very useful in establishing grasps. It allows a human to interact with the hand and position it manually. It is implemented by comparing the actual joint positions with the previous joint positions, and updating the

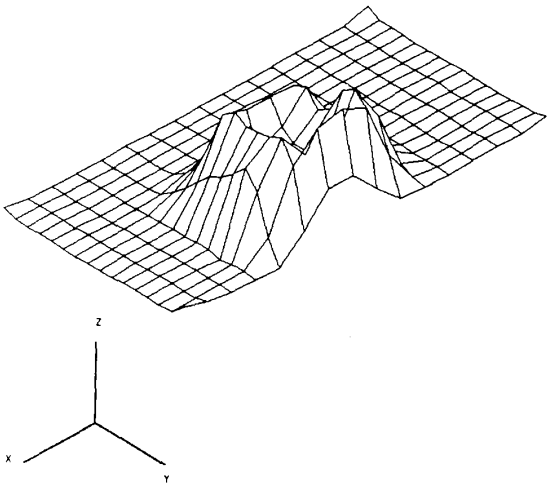

Fig. 4. Tactile impression of pen cap.

```
/* Pitcher script:
        (1)  Move arm/hand to above pitcher,grasp the top
        (2)  Lift top, move to new location,put it on table.
        (3)  Preshape hand to lift pitcher using pitcher's
             handle, move the arm/hand into the position.
        (4)  Grasp the pitcher, lift it,pour out the contents.
        (5)  Replace the pitcher to its original location.

/* DEFINITION LINES */
% prepit         hand_pose       "open_hand"
% pittop         hand_pose       "pitcher_top"   /* preshape */
% get_top        puma_command    "ex get_top"    /* arm to pitcher */
% lift_top       puma_command    "ex lift_top"   /* lift pitcher */
% to_grasp       puma_command    "ex to_grasp"
% pour_pit       puma_command    "ex pour_pit"
% retract_arm    puma_command    "ex retract_arm"
% grabtop        get_force       0x6666 0x0222 5
% grabpit        get_force       0xeeee 0x0222 2


/* EXECUTION LINES */
/*                1         2         3         4         5
/*       12345678901234567890123456789012345678901234567890*/
pittop   #==========          #===
get_top  #==========
grabtop             #
lift_top             #=========
to_grasp                          #======
prepit                            #======          #===
grabpit                                 #
pour_pit                                  #=======
retract_arm                                                 #=====
```

Fig. 5. Dial script for pouring pitcher task.

current desired position to reflect the new actual joint positions. For example, a hand can be placed on an object in a known position and the position can then be recorded for future use. It has a feature to allow the masking out of joints that need to be fixed (not Limp) and those which should be under Limp control.

C. Tactile Primitives

While the level of sensing provided by the joint position and tendon force sensors on the Utah-MIT hand is better than earlier implemented hands, it still falls far short of the requirements for a dextrous manipulation system. In particular, what is desired is accurate positional contact information between the hand and a target object, and a measure of the forces exerted by the fingers at these contact points. The sensor feedback provided by the hand does not allow for localization of contacts. Hence, a requirement for this system is a robust and accurate tactile sensing capability, utilizing sensors mounted on the links of the fingers.

To satisfy this requirement, we have implemented tactile sensors on each of the hand's fingers. The technology being used is a piezo-resistive polymeric material manufactured by Interlink, Inc. [18], [21]. This material has good force response characteristics and is able to be deposited on arbitrary, conformable surfaces. The design of the tactile pads we are using sandwiches the polymer between two pliable sheets of Kapton material that contains electrical etching. The application of forces on the pads provides an increased electrical flow channel between the two sheets as the material within is compressed. Results with this sensor have been good, particularly with respect to signal isolation and hysteresis. We have built an electronics package to multiplex the sensor signals, and hope to achieve a high enough bandwidth in signal processing of the sensors to achieve real-time integration of the tactile data into the grasping procedures. The sensors are monitored by a separate microprocessor board that is responsible for low-level tactile processing, including A/D conversion, and thresholding and normalization of signals. Some of the low-level tactile primitives that have been implemented are:

- TACTILE FILTERS: A number of useful digital filters have been implemented including averaging and median filters which are very useful in processing noisy tactile data [13].
- TACTILE MOMENTS: A useful technique for quickly getting contact information is central moment analysis [6]. The contact area and centroid of the contact can be determined using moments. The second moments are useful for determining the eccentricity of the contact region and the principal axes of the contact.
- EDGE DETECTION: A gradient-based edge detector has been implemented and used for feature extraction from tactile images.
- LINE DETECTION: Lines are detected by using the output of the edge detection procedure in a Hough transform [2].

Fig. 4 shows the output of tactile responses filtered with median and averaging filters when a pen cap is placed on the sensor pads. The signal is very localized (the sensing area is on the order of 0.5 in by 1.0 in). Using this tactile data, we have developed an exploratory procedure of lateral motion on a surface which is described in Section VI.

D. PUMA Arm Primitives

The arm primitives are already embedded in VAL-II, the programming environment of the PUMA. They include movement primitives, asynchronous interrupt capability, and the ability to establish arbitrary coordinate frames such as the hand coordinate system. It also provides a global coordinate system in which the tasks can take place.

III. DIAL: A HIGH-LEVEL CONTROL SYSTEM

Dial is a diagrammatic language that allows parallel processes to be represented in a compact graphical "time line." It has been used for animation of graphical displays but it has been transported by us to the robotics domain so we may exploit its ability to express parallel operation of robotic devices. It also provides a convenient way to implement task-level scripts which can then be bound to particular sensors, actuators and methods for accomplishing a generic grasping or manipulation task. The instruction set that Dial supports is specified at run-time in a user-supplied backend. We have created a Dial backend to work in our environment which has the need to express parallel actions in the performance of simple grasping and manipulation tasks. Given that we need to control the hand and arm movement concurrently, as well as integrating information from three different sensor systems (joint position, tendon force, tactile array), we found it natural to program tasks in Dial scripts.

Fig. 5 is a Dial script that is used to have the hand/arm system pick the top off of a pitcher on a table, place the top on the table, pick up the pitcher and pour from the pitcher, replacing the pitcher on the table. The horizontal dimension of

Dial's 2-D language represents time, with each column corresponding to what is called a tick. Each event starts at a particular tick and lasts for some integral number of ticks. Dial handles all flow of control by determining which events are to be executed during each tick. Entries in each column of execution lines (described below) specify the events that are to be performed during that column's tick.

Definition lines begin with a "%" in the first column, followed by an event name, an instruction name, and any parameters. Note that this line defines the event, but does not cause it to be executed. Execution lines specify a sequence of successively executed events. Execution lines begin with an event name. Each column in an execution line corresponds to a tick during which an event might be executed. A time line is shown in a comment above the execution lines to aid in discussing the timing of operations.

The appearance of the execution character "#" in the first column causes execution during the first tick. The "#" character is followed by a "=". This is a continuation character. The appearance of a continuation character after the execution character means that the event's execution is to be extended into the next tick. The speed of execution of a script, that is, the length of each tick, can be modified via a system parameter. This enables tasks to be sped up or slowed down with a single parameter.

IV. Task 1: Pouring from a Pitcher

This particular task is implemented with joint space position control, joint space force control, and Cartesian space arm control. The Dial script for such a task is shown in Fig. 5. During ticks 1–13, the arm and hand are moved toward the pitcher (event get_top), and the hand is preshaped to grasp the top (event pittop). In the Dial notation, these events are executed in parallel, with the hand preshaping being done concurrently with the movement of the arm. The hand preshaping is bound to a hand primitive (hand_pose) that takes a predefined setting of the hand joints ("open_hand") as a parameter. In tick 14, the event grab_top is used to grasp the pitcher top securely (Fig. 6). Grab_top is bound to the composite function Grasp with force which instructs the hand to grasp an object and to terminate when the tendon forces exceed a certain predefined threshold. The parameters in the definition line specify which joints to move, which tendons to monitor, and what the value of the force is. The instruction differs from the previously discussed instructions in that there is no way to determine ahead of time how long its execution will take. Therefore, while it is shown as a single-tick operation in the script, the grasping operation continues until the parameter values specified in the definition line are satisfied. In this way, the essential nondeterminism of a robotic task can be captured in Dial. Commands of indeterminate length are implemented as single tick events, but the completion of the event actually determines the tick's length. This allows us to implement a simple form of force control. During ticks 15–24, the hand/arm lifts the top off the pitcher and moves the top to a pre-defined position at which it will release the top (event lift_top). Starting at tick 25, the event pittop (which uses the low-level system primitive hand_pose) is used to move the hand back to its preshape position. This movement has the effect of releasing the top, because in the preshaping pose the hand is not in contact with the top. From tick 29 to 35, the arm moves to a position to grasp the pitcher (event to_grasp, Fig. 7), and simultaneously preshapes the hand to grasp the pitcher (event prepit). The event grabpit (tick 36) grasps the pitcher (Fig. 8); event pourpit (ticks 37–44) lifts the pitcher, turns it to pour the contents (Fig. 9), and then replaces the pitcher in its original position. Event prepit is used to release the pitcher and the PUMA movement event retract_arm removes the hand/arm from the pitcher.

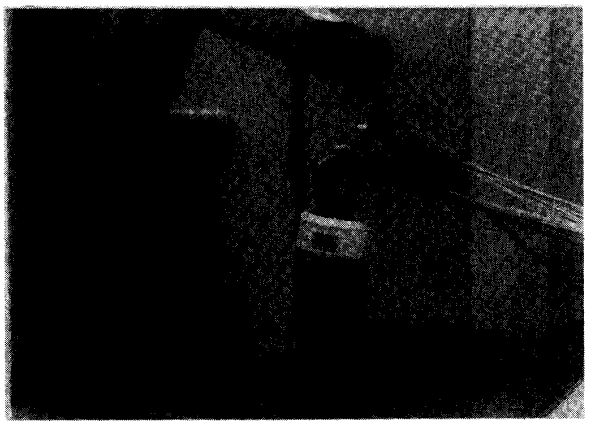

Fig. 6. Grasping the pitcher top.

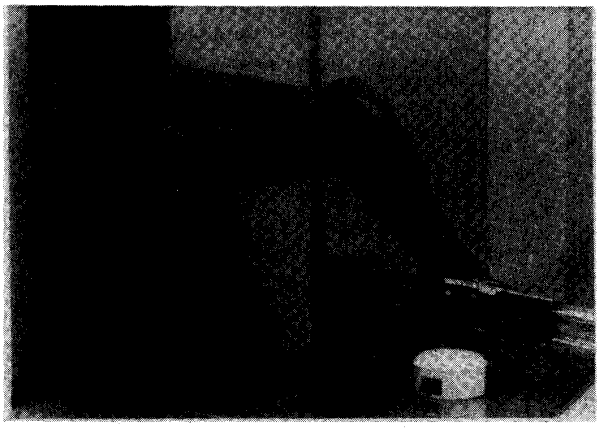

Fig. 7. Preshaping the hand for grasping.

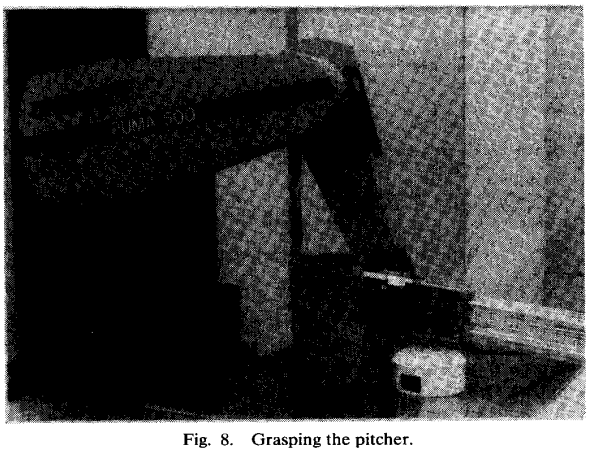

Fig. 8. Grasping the pitcher.

V. Task 2: Removing a Light Bulb

Fig. 10 is another example of a DIAL script—removing a light bulb from a socket (see Fig. 11). This is a task that requires tendon force feedback to determine the grasp strength and coordination of the fingers moving in parallel to unscrew the bulb. The script performs the task by moving the arm to the lightbulb while preshaping the hand, and then repeating

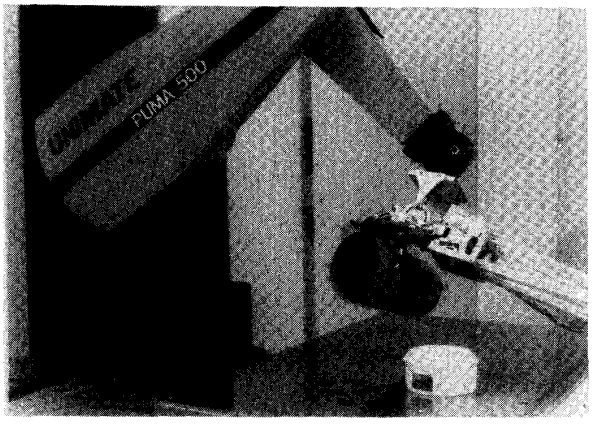

Fig. 9. Pouring the grasped pitcher.

Fig. 11. Hand removing light bulb from socket.

```
/* Script for removing a lightbulb
        (1)  Move arm/hand to above lightbulb and preshape

        (2)  Repeat the following sequence until bulb is
             unscrewed:
                - Grasp the lightbulb tightly
                - Twist the fingers and joint 6 of PUMA
                        counterclockwise
                - Release the grip on the lightbulb
                - Twist fingers and joint 6 of PUMA clockwise back
                        to position at start of (2)

        (3)  Grasp lightbulb and retract arm to lift bulb out
             of socket.                          */

/* DEFINITION LINES */
% preshape       hand_pose       "prelight"
% to_bulb        puma_command    "ex to_bulb"
% grasp_bulb     grasp_force     0x0666 0x0222 400
% twist_hand     hand_pose       "unscrew"
% ungrasp        hand_pose       "ungrasp"
% twist_armCCW   puma_command    "ex twist_wrist"
% twist_armCW    puma_command    "ex twist_CW"
% retract_arm    puma_command    "ex retract_arm"


/* EXECUTION LINES */
/*                  1   */
/*          1234567890   */
to_bulb    #==========
preshape         #====

/* REPEAT THE FOLLOWING SECTION DESIRED NUMBER OF TIMES */
/*              1         2         3  */
/*             12345678901234567890  */
grasp_bulb     #
twist_hand      #========
twist_armCCW    #========
twist_armCW                  #======
ungrasp                    #==
preshape                     #======


/* THE LAST ITERATION, PERFORM THE FOLLOWING SEQUENCE TO REMOVE
        THE BULB FROM THE SOCKET */
grasp_bulb      #
twist_hand       #========
twist_armCCW     #========
retract_arm                #========
```

Fig. 10. Dial script for removing light bulb task.

the following sequence of events: grasp the lightbulb, turn the hand/arm counterclockwise, release the grasp, move back to the initial preshaped position by rotating the hand and arm clockwise. After the bulb becomes loose, the hand graps the bulb and retracts it from the socket. In the script, movement of the arm to the bulb occurs during ticks 1–10. At tick 6, the preshaping of the hand is begun and will execute in parallel with the arm motion, resulting in the arm reaching its destination above the bulb in tick 10 with the hand preshaped. The next set of event lines represent the repeated sequence of grasps and movements that unscrew the bulb from the socket. The event grasp_bulb in tick 11 causes the hand to develop a secure grasp around the bulb. The next events, twist_hand and twist_armCCW, rotate the fingers of the hand and the arm wrist, respectively, in a counter-clockwise motion (ticks 12–20). The hand is moved to the position ungrasp (ticks 21–23) to release the grasp of the hand, and then the hand and arm are returned to their initial positions by the events preshape and twist_armCW in ticks 24–30. This sequence must be repeated a number of times before the light bulb is unscrewed, as shown. At the completion of the script, the event retract_arm is executed to remove the lightbulb from its socket and lift it in air (notice that the hand is still grasping the bulb).

VI. Haptic Perception

Besides grasping tasks, we are interested in using our system to perform haptic perception [19]. A number of interesting properties of the human haptic system have been investigated by Lederman and Klatzky and their colleagues [9]–[11]. This work has shown that an important component of the haptic system is its ability to recognize attributes of three-dimensional objects quickly and accurately. Among these attributes are global shape, hardness, temperature, weight, size, articulation and function [10]. In addition, this work has identified hand movement strategies that are used by humans in discovering different attributes of 3-D objects. These strategies have been labeled as EP's (exploratory procedures), and we have found it natural to extend these human derived capabilities to our robotic domain. However, we must be careful here, in trying to draw too close a comparison between a human hand and devices such as a Utah-MIT hand. Johansson and Vallbo [8] have reported that there are about 17000 mechano receptors in the skin of the human hand; our robotic hand is more limited with 16 joint sensors, 32 tendon force sensors, and 4 16×16 fingertip tactile sensors. In addition, a human hand has two main differences in structure from our robotic hand. The first is a highly flexible, opposable thumb that is mounted to the side of the other digits. The Utah-MIT hand thumb is identical to the other fingers and is mounted directly opposite the other fingers. The second difference is a palmar degree of freedom exists in human hands that is missing in the Utah-MIT hand. Humans find this palmar degree of freedom quite useful, especially for encompassing type graps where the hand is molded to an object, almost independent of the existence of multijointed fingers.

A. Grasping by Containment

Grasping by containment is a method used to understand an object's gross contour and volume by effectively molding the hand to the object. This procedure is described in detail in [1]. In summary, the procedure wraps the hand around an object and collects contact sample points which are then fed into a nonlinear least-squares recovery algorithm developed by Solina [17] that attempts to recover an object's shape as a su-

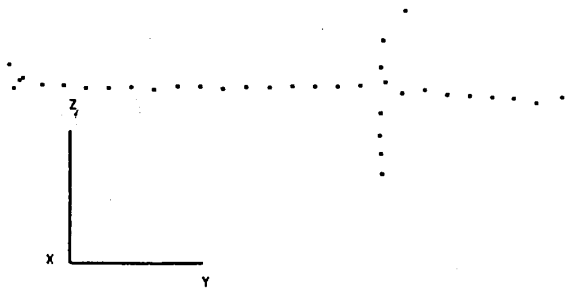

Fig. 12. Surface explorer tactile contacts on a rectangular planar surface.

perquadric. Stable and accurate recovery of shapes such as a wedge, funnel, cylinder, block, etc., has been performed using this EP. A benefit of this method is that the least-squares procedure tends to blur noisy sensor data, much like a low-pass filter, and it has been postulated that human touch receptors are in fact a form of low-pass filter.

B. Surface Exploration

This EP is used to explore planar surfaces with the hand's index finger. While the index finger is held in an extended position, the hand and arm are moved to a specified location. Then the index finger is flexed until the fingertip's tactile sensor detects contact with a surface. (If no contact is detected, the procedure terminates.) After the initial contact, the Cartesian position of the contact point is noted. The hand and arm then begin an iterative search for the boundaries of the surface by performing the following sequence: 1) back the finger off from the surface—that is, lift the finger off the surface until tactile contact is lost; 2) move the arm in a direction parallel to the surface; 3) if the finger is in contact after the movement, note the new contact location, otherwise lower the index finger until it makes contact with the surface again; 4) repeat 1)–3) until the finger fails to make contact in step 3). In step 4), if the finger does not contact the surface, then either the finger has moved beyond the edge of the surface, or the surface is too far away from the finger to be detected. To check for the latter case, the arm must be moved toward the surface. After completing the first collection of data points and finding the edge of the surface, the index finger is moved back to the position of initial contact, and a second mapping of the surface is undertaken in a direction 180° opposite. This procedure continues until a second surface edge is detected. The search now continues as before but in a direction perpendicular to the first two traces. This procedure then is able to map out a set of contact points on the surface, describing its extent. Each time the fingertip contacts the surface, the Cartesian coordinates of the contact are retained. Periodically, a linear, least-squares fit to a plane is performed to derive the normal to the plane and a measure of fit. The surface normal is useful not only for the sake of object recognition, but also in speeding the exploratory procedure by refining the hypothesis of where to move the arm. Fig. 12 shows a pattern of traces on a planar surface of a rectangular block using this EP. The contact points are all within a single plane with the exception of the end points where the finger has dropped off the edge of the block, thus ending the search procedure.

VII. Conclusion

The system we have built is an important step in building an integrated multisensor robotic perception system. Robotic systems need to have the ability to process sensor data from a number of sources and to be programmed in a simple and natural way to accomplish grasping and manipulation tasks. Our system has allowed us to build a number of task-level scripts that embody parallel actuation of devices (hands, fingers, arm, wrist) with sensory feedback from a number of different sources (joint positions, tendon forces, tactile arrays). We are currently adding vision sensors to this system, and these sensors will be integrated through the same set of software as the hand primitives.

The high-level control of the system through Dial needs to be further modified to extend its capabilities in the robotics domain. One extension is to include the ability to dynamically alter script sequences. Currently, Dial must continue on a single script thread. For example, in haptic exploration tasks, the feedback from the hand–arm system is used to generate hypotheses about an object's shape, which can drive new rounds of sensory exploration. This can be implemented by having multiple Dial scripts available, and dynamically executing a particular script sequence based upon the outcome of the previous exploration.

This system will serve as an experimental test bed that will allow us to continue to explore new grasping strategies, task-level control, and further extensions of haptic exploratory procedures.

Acknowledgment

We would like to thank Steve Feiner and cliff Beshers for bringing Dial to our attention and helping us port it to our environment.

References

[1] P. Allen and K. S. Roberts, "Haptic object recognition using a multi-fingered dextrous hand," *IEEE Conf. Robotics Automat.*, pp. 342–347, Scottsdale, AZ, May 15–19, 1989.
[2] D. Ballard and C. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[3] R. Fearing, "Simplified grasping and manipulation with dextrous robot hands," *IEEE J. Robotics Automat.*, vol. RA-2, no. 4, pp. 188–195, 1986.
[4] S. Feiner, D. Salesin, and T. Banchoff, "DIAL: A diagrammatic animation language," *IEEE Computer Graphics and Animation*, vol. 2, no. 7, pp. 43–54, Sept. 1982.
[5] C. C. Geschke, "A system for programming and controlling sensor-based robot manipulators," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 1, pp. 1–7, Jan. 1983.
[6] B. K. P. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.
[7] S. C. Jacobsen, E. K. Iversen, D. F. Knutti, R. T. Johnson, and K. B. Biggers, "Design of the Utah/MIT dextrous hand," *IEEE Conf. Robotics Automat.*, pp. 1520–1532, San Francisco, April 7–10, 1986.
[8] R. S. Johansson and A. B. Vallbo, "Tactile sensory coding in the glabrous skin of the human hand," *Trends in Neurosciences*, vol. 6, pp. 27–32, 1983.
[9] R. Klatzky, S. Lederman, and V. Metzger, "Identifying objects by touch: An 'expert system'," *Perception and Psychophysics*, vol. 37, pp. 299–302.
[10] S. Lederman and R. Klatzy, "Hand movements: A window into haptic object recognition," *Cognitive Psych.*, vol. 19, pp. 342–368, 1987.
[11] S. J. Lederman and R. A. Browse, "The physiology and psychophysics of touch," *Sensors and Sensory Systems for Advanced Robots*, NATO ASI series. New York: Springer-Verlag.
[12] M. T. Mason and J. K. Salisbury, *Robot Hands and the Mechanics of Manipulation*. Cambridge, MA: M.I.T. Press, 1985.
[13] C. Muthukrishnan, D. Smith, D. Myers, J. Rebman, and A. Koivo, "Edge detection in tactile images," *IEEE Int. Conf. Robotics Automat.*, pp. 1500–1505, Raleigh, NC, 1987.
[14] S. Narasimhan, D. M. Siegel, and J. M. Hollerbach, "Condor: A revised architecture for controlling the Utah-MIT hand," *IEEE Conf. Robotics Automat.*, pp. 446–449, Philadelphia, PA, April 24–29, 1988.
[15] S. Narasimhan, D. M. Siegel, J. M. Hollerbach, K. Biggers, and G. E. Gerpheide, "Implementation of control methodologies on the computational architecture for the Utah/MIT hand," *IEEE Int. Conf. Robotics Automat.*, pp. 1884–1889, San Francisco, CA, Apr. 7–10, 1986.
[16] K. Salisbury, D. Brock, and S. Chu, "Integrated language, sensing and control for a robot hand," *Int. Symp. Robotics Research*, Gouvieax, France, Oct. 1985.
[17] F. Solina, "Shape recovery and segmentation with deformable part models," Ph.D. Dissertation, Dept. Comput. Sci., Univ. Pennsylvania, Dec. 1987.
[18] T. Speeter, "Flexible piezo-resistive touch sensing array," *SPIE Conf. Optics, Illumination and Image Sensing for Machine Vision III*, Cambridge, MA, Nov. 1988.

[19] S. Stansfield, "Visually-guided haptic object recognition," Ph.D. dissertation, Dept. Comput. Inform. Sci., Univ Pennsylvania, Oct. 1987.
[20] K. Takase, R. Paul, and E. Berg, "A structured approach to robot programming and teaching," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 4, pp. 274–289.
[21] B. Tise, "A compact high resolution piezo-resistive digital tactile sensor," *IEEE Conf. Robotics Automat.*, pp. 760–764, Philadelphia, PA, Apr. 24–29, 1988.
[22] R. Tomovic, G. Bekey, and W. Karplus, "A strategy for grasp synthesis with multi-fingered robot hands," *IEEE Int. Conf. Robotics Automat.*, pp. 83–89, Raleigh, NC, Mar. 31–Apr. 3, 1987.

Representing Global World of a Mobile Robot with Relational Local Maps

MINORU ASADA, MEMBER, IEEE, YASUHITO FUKUI, AND SABURO TSUJI, MEMBER, IEEE

Abstract—**Many mobile robot systems adopt a two-dimensional (2-D) map, a top view of the space, represented with a Cartesian coordinate system fixed to an object in the scene, and plan the route to the destination on it. These approaches tend to be brittle, accumulate error and utilize little or no qualitative information. Instead of numerical integration of local maps into a global map, a method for representing a global map consisting of local map representations and relations between them has been developed. Sensor maps (maps of which origins are fixed to the moving sensor) viewed at locations close to each other are integrated into a local map representation in a Cartesian coordinate system fixed to an object. First, three-dimensional (3-D) information of the edges on the floor is obtained at each sensor map by assuming the camera model and the flatness of the floor. A reliable feature is selected as a reference in a sensor map, which has important roles in finding the correspondence between the current sensor map and the following ones and in building a local map with these sensor maps. During the motion of the robot, the local map is updated by the motion stereo method unless the current reference point disappears from the sensor map. Farther edges should be represented in other local maps when the robot approaches to them since the precise estimation of their locations in the current local map is difficult. Finally, the relation between local maps that represents the relative orientation and the approximate distance from the previous local map to the current one is included in the global map. The method is tested in an indoor environment and experimental results are shown.**

I. Introduction

The capabilities for representing and reasoning about 3-D world from sensory data are essential for intelligent mobile robot systems to accomplish various tasks such as visual navigation, obstacle avoidance, and landmark (and/or object) recognition. So far, several approaches to representing of the scene in the mobile robot systems have been studied [1]–[7] etc., where various kinds of sensors such as TV camera(s), a range finder, and a dead reckoning are used to build a 3-D world model (map). Whichever sensor(s) is (are) used, these systems have to cope with the following problems: (1) how should 3-D information from sensory data be represented, in other words, what kind of world model (map) is suitable for representing the 3-D information, and (2) how should this model be updated and/or integrated?

Tsuji and Zheng [1] have developed a stereo-vision-based mobile robot system that constructs a perspective map where the 3-D information obtained from stereo vision is represented in the image coordinate system, and used it for predicting the location of each obstacle in the next perspective map. Also, Dunlay [2] has adopted a perspective map of height information transformed from an ERIM range image, and used it to detect obstacles and to determine the steering angle and the vehicle speed. In these systems, the 3-D information from sensory data is represented in a sensor-based coordinate system fixed to the mobile robot.

Herbert and Kanade [3] have analyzed ERIM range images and constructed a surface property map represented in a Cartesian coordinate system viewed from top, which yields surface type of each point and its geometric parameters for segmenting the scene map into traversable regions and obstacle ones. Also, Daily, Harris, and Reiser [4] have constructed a top view of height information (which they call Cartesian Elevation Map) transformed from ERIM range images, and planned a number of candidates for vehicle trajectories on it for cross-country navigation. In these systems, a two-dimensional (2-D) map viewed from top is used to represent the three-dimensional (3-D) information from a range finder, and is fixed to static objects in the world so as to integrate the 3-D information observed at different locations. Since the resolution in a map of this kind is different from that of the sensor output (e.g., a range image), smoothing and/or interpolation of the sensor outputs are needed.

Differences between the sensor-based perspective maps [1], [2] and the object-centered 2-D maps (a top view of the scene) [3], [4] are discussed in [1], [5]. They argue that the 2-D maps are easy to understand but do not naturally capture sensor resolution and accuracy, and that for real-time navigation, adjusting the steering angle and controlling the vehicle speed from the perspective maps are much easier than from the 2-D maps. However, integration of the perspective maps obtained at different locations into a single perspective map seems difficult.

Elfes [6] has developed a sonar-based mapping and navigation system that constructs sonar maps of the environments viewed from top and updates them with recently acquired sonar information. He proposed a hierarchical representation of sonar map that includes three kinds of axes; an abstract axis, a resolution axis, and a geographic axis. Since the outputs of sonar sensors are directly mapped to a 2-D map, the difference between the sensor maps and the object-centered 2-D maps is implicit in his system. Asada [7] has presented a method for building a 3-D world model using the range information. He adopted the hierarchy of the scene map by Elfes [6], but extended three levels of the geographic axis (a sensor map, a local map, and a global map) so that other sensor outputs can be represented in this hierarchy, and that the relation between three kinds of coordinate systems can be explicit. In his system, local maps are integrated into a final global map by matching geometrical properties of obstacle regions in the local maps.

Brooks [8] pointed out that in such a system, a traveling of long distance often results in a significant amount of positional error in the global map and that the robot cannot correct this error unless landmarks in the environment are given in advance. He proposed an interesting idea to use rubbery and stretchy relational maps, making the spational reasoning robust to errors

Manuscript received May 1, 1990; revised February 9, 1990. This work was supported by the Grant-In-Aid for Scientific Research from the Ministry of Education, Science, and Culture, Japanese Government. This work was partially presented at the 9th International Conference on Pattern Recognition, Rome, Italy, November 1988.

M. Asada is with the Department of Mechanical Engineering for Computer-Controlled Machinery, Osaka University, Suita, Osaka 565, Japan.
Y. Fukui is with the Audio and Video Research Lab., Matsushita Electric Industrial Co., Ltd., Kadoma, Osaka 571, Japan.
S. Tsuji is with the Department Control Engineering, Osaka University, Toyonaka, Osaka 560, Japan.

IEEE Log Number 9037606.