N93-11972

AN INTEGRATED DEXTEROUS ROBOTIC TESTBED FOR SPACE APPLICATIONS

Larry C. Li
Hai Nguyen
NASA/Johnson Space Center
Automation and Robotics Division

Edward Sauer
Lockheed Engineering and Science Company

ABSTRACT

An integrated dexterous robotic system was developed as a testbed to evaluate various robotics technologies for advanced space applications. The system configuration consisted of a Utah/MIT Dexterous Hand, a PUMA 562 arm, a stereo vision system, and a multiprocessing computer control system. In addition to these major subsystems, a proximity sensing system was integrated with the Utah/MIT Hand to provide capability for non-contact sensing of a nearby object. A high-speed fiber-optic link was used to transmit digitized proximity sensor signals back to the multiprocessing control system. The hardware system was designed to satisfy the requirements for both teleoperated and autonomous operations. The software system was designed to exploit parallel processing capability, pursue functional modularity, incorporate artificial intelligence for robot control, allow high-level symbolic robot commands, maximize reusable code, minimize compilation requirements, and provide an interactive application development and debugging environment for the end users. This paper presents an overview of the system hardware and software configurations, discusses implementation of subsystem functions, and recaps lessons learned from our work. Current work and future evolution of the system are also discussed.

INTRODUCTION

An integrated dexterous robotic system was developed by the NASA Johnson Space Center (JSC) as a testbed to evaluate various robotics technologies for advanced space applications. The technologies of interest include: dexterous robotic arms and hands, machine vision, tactile and proximity sensing, grasping and manipulation algorithms, parallel computational architecture, and artificial intelligence. The configuration of the testbed system consisted of a 16 degrees-of-freedom (DOF) Utah/MIT Dexterous Hand, a 6 DOF PUMA 562 arm, a stereo vision system, and a VMEbus-based multiprocessing control system. In addition to the position and force sensors already present in the Utah/MIT Hand, an 8-element proximity sensor system was developed and integrated with the hand to provide near-range non-contact sensing capability. A high-speed fiber-optic link, also developed at JSC, was used to transmit digitized proximity sensor signals back to the multiprocessing control system. A hierarchical functional architecture was implemented to provide serial-parallel execution of limb motions. A JSC-developed expert system tool called the C Language Integrated Production System (CLIPS) was used as a rule-based robot programming environment. The system configuration allows autonomous operation and teleoperation. The purpose of this paper is to present an overview of our implementation. First, background and other work in similar areas are reviewed. Then the objectives for developing this testbed system are stated. The system overview covers subsystem implementations. Examples of robot programming are given in the section, Robot Programming Using CLIPS. Finally, the last section summarizes lessons learned from our work.

BACKGROUND

NASA has been active in robotics from the early days of the space program. The Viking mission to Mars is one shining example. Recent progress in robotics technology has allowed NASA to design robots that will help to increase productivity in space. These space robots may be used to perform dangerous or laborious tasks which otherwise would have to be performed by the astronauts. The Shuttle Remote Manipulator System (SRMS), for example, has demonstrated its effectiveness in on-orbit satellite retrieval and repair. The Flight Telerobotic Servicer (FTS), targeted for the Space Station, will be the first advanced multi-function robot in space. The Extravehicular Activity (EVA) Retriever, currently under development at JSC, is an advanced space robot designed for short-range, contingency retrieval and rescue missions. Other advanced robotic systems under development by NASA, such as the Lunar/Mars Rover and the Satellite Servicing System, are other examples of NASA's commitment to further enhance and apply the robotics technology.

The Automation and Robotics Division at JSC has developed a dexterous robotic testbed to develop and evaluate various enabling robotics technologies for space applications. Different from other NASA-developed robotic systems, this testbed emphasizes the development, integration, and application of dexterous robotic hands and arms. Outside NASA, there are many other research efforts with similar emphasis and interests. Clark and Demmel, et. al,[3] has developed a dexterous robotic system consisting of a Utah/MIT Hand, a PUMA 560 arm, a Polhemus 3D Tracker, and a VPL Data Glove. Their implementation was optimized for teleoperation with the VPL Data Glove and the Polhemus 3D Tracker providing position control of hand and arm, respectively. Allen, Michelman, and Roberts[1] described an integrated system for dexterous manipulation. Their implementation also included a Utah/MIT Hand and a PUMA 562 arm. Although similar to the implementation described in reference 3, Allen, Michelman, and Roberts developed their system for autonomous operations. Allen and Roberts[2] have successfully demonstrated haptic object recognition using this system. They have integrated tactile sensors with the Utah/MIT Hand, and used a descriptive language called DIAL for high-level control. Narasimhan, in his master's thesis[9], described a VME multi-processing control system for the Utah/MIT Hand. His system contained several 68020 computer processing units (CPUs) for servo-level and task-level controls. A real-time operating system called Condor was integrated with the system to provide timing, task scheduling, and other process control functions. Salisbury, Brock, and O'Donnell[11] built their dexterous hand system around the Stanford/JPL Hand. They introduced the usage of a LISP machine as the high-level controller providing rule-based programming capability. Stansfield[12,13] developed a dexterous arm/hand system with knowledge-based visually-guided grasping. Our implementation incorporated and enhanced some of the features and concepts found in these research efforts.

OBJECTIVES

Our overall objective was to develop, evaluate, demonstrate, and enhance dexterous robotics technologies for space applications. Although our overall objective was somewhat general, it can be broken down into three specific goals: (a) develop and demonstrate capabilities of dexterous robotic systems for space applications; (b) investigate, implement, and evaluate latest advanced robotics technologies; and (c) develop an integrated testbed to support and demonstrate autonomous operation and teleoperation of dexterous robotic systems. Rationales behind these three objectives are explained next.

a. Develop and demonstrate capabilities of dexterous robotic systems for space applications

Future space robots are required to be highly versatile and productive. In order to achieve the required versatility and productivity, these robots should be equipped with intelligent dexterous arms and hands to handle a

multitude of tasks. Because most dexterous robotic hands are modeled after the human hand, their anthropomorphic designs allow the robots to share a common set of tools and handholds with the astronauts, thus minimizing any redesign of existing flight hardware. Other dexterous robotic components such as robotic arms with redundant DOF are also important in providing robots with multiple trajectory options for collision avoidance and path planning.

b. Investigate, implement, and evaluate latest advanced robotics technologies

Besides dexterous robotic hands and arms, other advanced robotics technologies such as parallel computers, machine vision, tactile and proximity sensing, expert systems, and neural networks are also important in the development of an intelligent space robot. Since there is usually an appreciable time gap between emergence of a new technology and the actual implementation of this technology on a flight system, it is important for NASA to keep up with the latest advanced robotics technologies so that the space robots will not be technologically outdated.

c. Develop an integrated testbed to support both autonomous operations and teleoperations

In order to evaluate and demonstrate technologies mentioned above, a testbed system must be developed. It is important for this testbed to be fully integrated so that different technologies can work together to achieve the high-level functions required in an intelligent space robot. An integrated testbed system will also allow us to verify planned operations and identify any unanticipated problems. The testbed system should support both autonomous operations and teleoperations, since both modes of operation are likely to find applications in future space activities.

SYSTEM OVERVIEW

Based on the objectives discussed in the previous section, a testbed system was established. Figures 1a and 1b show the current Smart Hand testbed system configuration. This section provides an overview of the system, which includes descriptions of hardware system architecture, software system architecture,

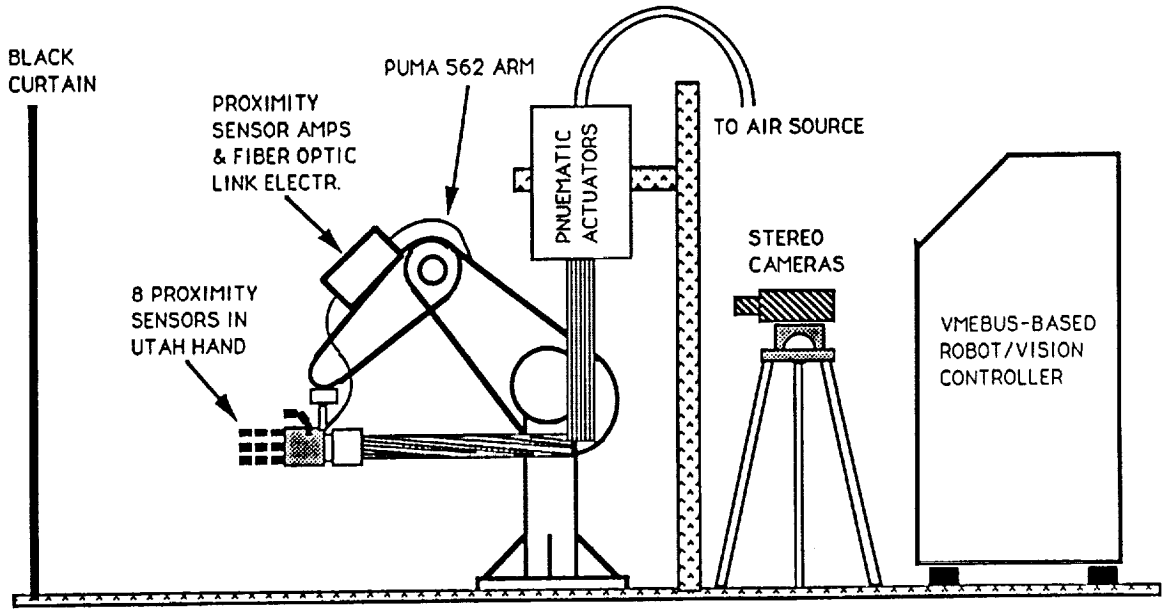

Figure 1a. Current Smart Hand testbed system configuration.

dexterous arm and hand subsystem, vision subsystem, teleoperator interface, and the proximity sensor subsystem.

Figure 1b. Arm and Hand of the testbed system.

Hardware System Architecture

Major hardware components in the system include a Utah/MIT Hand, a PUMA 562 Arm, a stereo vision system, an EXOS Dexterous Hand Master, infrared proximity sensors, and a computer control system. The computer control system is a multiprocessing system with three 68020 single board computers (SBC) mounted inside a 20-slot VMEbus chassis. Each SBC is outfitted with pSOS™ - a real-time multi-tasking operating system kernel, and pRISM™ - a multiprocessing operating system, both of which are products of Software Components Group. Figure 2 shows the functional block diagram of the current system configuration. Each 68020 SBC is responsible for a specific task such as arm control, hand control, and vision control. The software running on these SBCs are discussed in greater detail in the Dexterous Arm and Hand Subsystems and Teleoperator Interface sections. A multi-function VMEbus system controller board arbitrates bus access among the SBCs. The serial port on-board the system controller provides the communication channel to the Unival PUMA arm controller.

Two Data Translation video frame grabbers are used to capture images from the two cameras. The digitized images are processed by the vision controller to produce a 3D vector pointing at the target. This vector is sent to the robot arm and hand controllers for reaching and grasping. The analog-to-digital (A/D) and digital-to-analog (D/A) converters are used to interface with both the Utah/MIT analog controller and the EXOS Dexterous Hand Master. A parallel digital input/output (I/O) board connected with a high-speed fiber-optic link gathers data from the proximity sensor subsystem. Fiber-optic transmission is used to avoid electromagnetic interference (EMI) generated by the large motors located inside the PUMA arm, and to reduce the number of wires bundled at the arm joints. A VMEbus-based 386SX personal computer (PC) is embedded inside the chassis for two purposes: (1) to provide a software development environment for the system programmers, and (2) to host an intelligent rule-based system for developing applications at the symbolic level. These two functions correspond to two phases of operation: development phase and operational phase.

During the development phase, the PC operates under MS-DOS. A 68020 C cross-compiler is used to generate executable codes from user-written source codes. The executable codes are then loaded, via the VMEbus backplane, into the SBC's dual-ported memories. During this time, the SBCs are in idle, waiting for signals to begin execution. During the operational phase, all subsystem software is loaded and started. The 386SX PC begins executing an expert system shell, the CLIPS. The CLIPS communicates with the subsystem SBCs through the *system executive*. All applications are developed under this shell, using CLIPS and user-defined syntax.

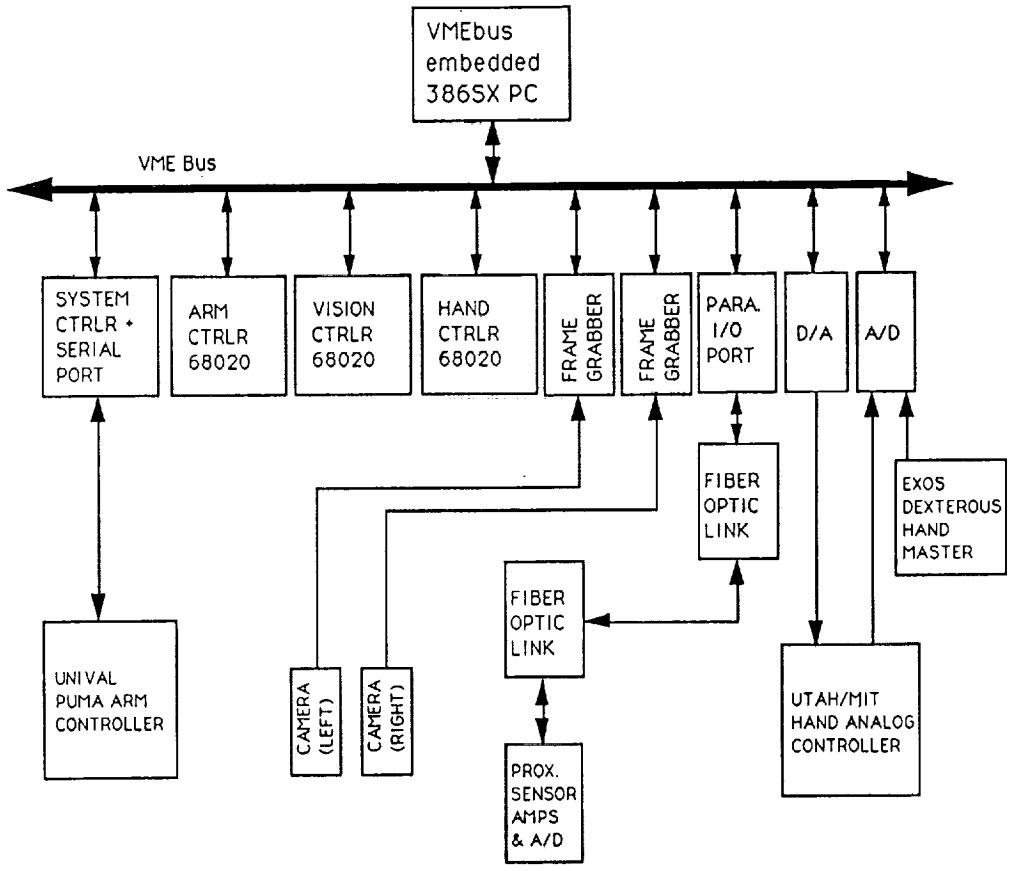

Figure 2. Functional block diagram of the current system configuration.

Software System Architecture

The software system architecture is designed based on the following objectives: (1) exploit parallel processing capability, (2) pursue functional modularity, (3) incorporate artificial intelligence for robot control, (4) develop high-level, symbolic robot commands, (5) maximize reusable code, minimize compilation requirement, and finally, (6) provide an interactive application development and debugging interface. Figure 3 illustrates the software system architecture. Software for the top three layers are running on the 386SX PC, while software for each subsystem in the subsystem layer are running on separate SBCs.

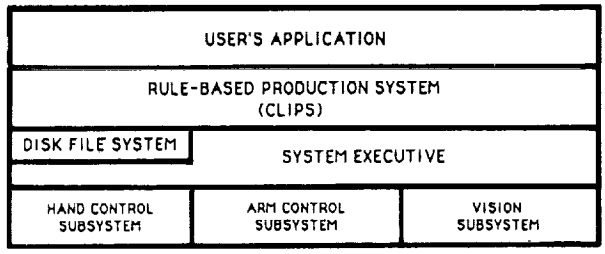

Figure 3. Software system architecture.

Robot arm and hand trajectories are *serial* executions of *parallel* motions. For example, the action of grasping for a ball consists of the following steps:

a. Locate the ball
b. Reach for the ball while opening hand
c. Close hand around the ball
d. Retract arm and hand

Steps *a*, *b*, *c*, and *d* must be executed *serially*, otherwise the task will not be accomplished properly. However, within each serial step, parallel motion takes place. Take Step (*b*) for example, the hand opens while the arm reaches for the ball. During opening of the hand, each finger joint should move concurrently; otherwise the motion would be awkward and time consuming. Understanding this principle, the software architecture was designed to realize *serial-parallel* motions. The overall architecture contains four layers: *user application*, *rule-based production system*, *system executive*, and *subsystem layer*. There are three functional subsystems executing in parallel at the *subsystem layer*. The hand control subsystem performs coordinated control of finger joints, forward and inverse kinematics, acquisition of proximity, position and force sensor data, and automatic calibrations. The arm control subsystem performs predefined motion primitives, and handles serial communications between the Unival PUMA arm controller and the VMEbus control system. The vision subsystem executes a stereo vision algorithm that constantly tracks the target within the visual field. These three subsystem tasks are inherently parallel. The *system executive* is responsible for orchestrating activities among the three subsystems serially to provide fluid arm-hand motions. The *system executive* contains a set of user-defined functions that are frequently called on by CLIPS to carry out any user-defined commands. The mechanism on how these user-defined functions are integrated under CLIPS is described briefly in the Robot Programming in CLIPS section. For a more comprehensive description, one should consult the CLIPS User's Guide[4] and CLIPS Reference Manual[10].

Dexterous Arm and Hand Subsystems

The dexterous arm and hand subsystems are responsible for primitive-level control of the Utah/MIT Hand and the PUMA 560 arm. The Utah/MIT Hand is a 16 DOF dexterous hand with 3 fingers and a thumb in an anthropomorphic arrangement (see Figure 4). Embedded within each finger joint is a Hall-Effect sensor for position feedback. Each joint is controlled by two antagonistic tendons. Thirty-two Hall-Effect force sensors located in the wrist are used to detect tendon tensions. The tendons are actuated pneumatically by thirty-two air cylinders. Accompanying the Utah/MIT Hand is an analog servo controller driving the air cylinders. Position and force command signals are received from the D/A converters, and the position and force sensor signals are sent to the A/D converters. The hand controller contains several motion and sensing primitives that may be called upon by the *system executive* software running on the 386SX. The primitives running on the hand controller are listed in Table I. The *system executive* can invoke these primitives by passing command tokens, via the VMEbus, to the dual-ported memories of the hand controllers. When a primitive is invoked, the task is carried out by the hand controller SBC, and the *system executive* is now free to do other tasks. Once the primitive is fully accomplished, a DONE flag is raised to signal task completion. DONE flags are present in the hand controller, the arm controller, and the vision controller. They are extremely important for synchronization of parallel movements. Forward and inverse kinematics are both included in the hand controller software. Primitives such as TIP_MOVE, TIP_ POSITION move and report fingertip locations in Cartesian coordinates. Raw A/D counts for joint and tendon sensor readout are included to facilitate system debugging and calibration.

The arm controller operates very similar to the hand controller. The communication interface between the arm controller and the *system executive* is also through dual-ported memories. The arm controller accepts only two commands: MOVE_ARM_TIP, and MOVE_ARM_JOINT. The first specifies positions in Cartesian space; the second specifies positions in joint space. In VAL II (a PUMA programming language) terminology, they correspond to *transformed* moves and *precision* moves. During actual operation, arm controller accepts

commands from the *system executive*, and formats these commands into proper VAL II syntax, and then sends the formatted VAL II commands to the Unival PUMA arm controller via a RS-232 serial line. A small auto-start program running on Unival controller accepts the formatted commands and moves the PUMA arm to the proper position at specified speed. Upon detection of an error, or a completed move, an appropriate code is sent back to the arm controller via serial line to signal the event.

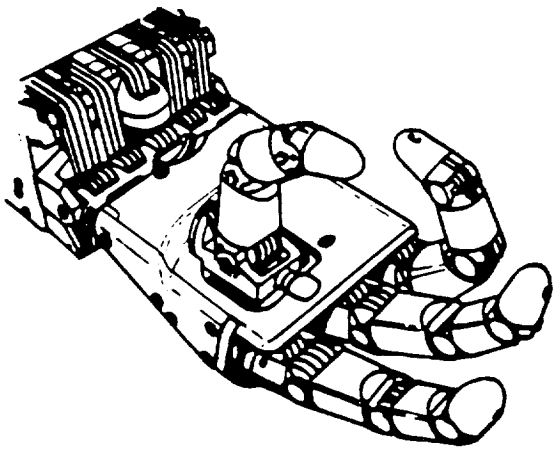

Figure 4. The Utah/MIT Dexterous Hand. (Reprint with permission from SARCOS)

Teleoperator Interface

Currently, the teleoperator interface consists of only the EXOS Dexterous Hand Master. A Polhemus 3D Tracker System, shown in Figure 5a, is being integrated with the hand master to provide a full teleoperation of the arm and hand. The EXOS Dexterous Hand Master, shown in Figure 5b, is an exoskeletal glove controller that can be worn by a human operator. The glove controller is capable of detecting movements of the first three fingers and the thumb, with four analog Hall Effect sensors per each finger and thumb. The sensor signals are amplified and filtered by a custom-built circuit board before they are passed on to the A/D converters. During operation, the *system executive* reads in the joint angles and commands the hand controller to move the robot hand to corresponding positions. The teleoperator control is presently operating in the joint space. Work is being done to include kinematic transform to allow Cartesian space control.

Vision Subsystem

The objective of the vision subsystem is to provide a 3D position of a target at a high update rate. Passive triangulation method is applied to the left and right images of the two video cameras to determine the target distance. Video cameras are used because they can capture images at a fairly high rate of 30 Hz. An active laser scanner vision system was considered. However, it was abandoned because the laser scanner proved to be slower, more costly, and less reliable. The video target tracking algorithm running on the vision controller is composed of the two modules: *centroid discovery module*, and *position calculation module*.

The *centroid discovery module* identifies the target centroid, tracks it, and performs correspondence matching between centroids found in the left and the right cameras. In order to increase search speed, the search algorithm was designed to operate in two modes. The first mode quickly searches the image by scanning from the center outward, skipping five rows at a time, alternating about the center. The search looks for pixels with a value greater than the preset threshold. Once the first of such a pixel is found, the second search mode commences at that row, working outward, comparing every row. The purpose of the search is to determine the maximum and minimum of target extent, and then take the average of the two numbers to determine the vertical coordinate of the target centroid. With the vertical coordinate of the target centroid determined, a side-to-side scanning of that row determines the horizontal coordinate of target centroid. The word *centroid* is used loosely here to denote the center of the projected target area. With the first centroid identified in one image, the second centroid can

be quickly determined from the other image. Since the cameras are co-planar, the second centroid should be located at the same row in the second image. Therefore, only a single row needs to be searched. These two centroids are assumed to be at the same point, and their pixel locations are passed on to the *position calculation module*.

The position calculation module simply calculates the target position using a triangulation method. In order to produce accurate calculations, horizontal and vertical fields-of-view (FOV) of the two cameras were measured. Knowing the FOV, the relationship between the target angle and the pixel location can be characterized by the following equations:

$$h = \frac{H}{512}, \quad v = \frac{V}{480}$$

where H and V are the horizontal and vertical FOV (in radian), respectively. The constants, 512 and 480, are the horizontal and vertical pixel image resolution of the frame grabbers. Parameters h and v are the multiplicative constants that convert pixel positions to target angles. These two equations are used in the triangulation calculations to determine x, y, and z values of the target location. To facilitate triangulation, the cameras are physically configured as illustrated in Figures 6a and 6b. The reference frame used is a right-handed coordinate system with the origin located midway between the two cameras. In this configuration, both cameras are located on the X-Y plane. The equations that determine distance in Z are:

$$Z = \frac{d\tan(\pi-\theta_1)\tan(\theta_2)}{\tan(\pi-\theta_1)-\tan(\theta_2)}; \theta_1 > \pi$$

$$Z = \frac{d\tan(\theta_2)\tan(\theta_1)}{\tan(\theta_1)+\tan(\theta_2)} \theta_1 \le \pi, \theta_2 \le \pi$$

$$Z = \frac{d\tan(\pi-\theta_2)\tan(\theta_1)}{\tan(\pi-\theta_2)-\tan(\theta_1)}; \theta_2 > \pi$$

The two angles are determined by

$$\theta_1 = (\pi - h I_L)$$

$$\theta_2 = h I_R$$

TABLE I. HAND CONTROL PRIMITIVES

COMMAND	TYPE
JOINT_MOVE	JOINT SPACE POSITION CONTROL
TIP_MOVE	CARTESIAN SPACE POSITION CONTROL
JOINT_COUNT	JOINT SPACE POSITION FEEDBACK (RAW A/D COUNT)
JOINT_ANGLE	JOINT SPACE POSITION FEEDBACK
TIP_POSITION	CARTESIAN SPACE POSITION FEEDBACK
TENDON_COUNT	TENDON SPACE FORCE FEEDBACK (RAW A/D COUNT)
TENDON_TENSION	TENDON SPACE FORCE FEEDBACK
JOINT_TORQUE	JOINT SPACE TORQUE FEEDBACK

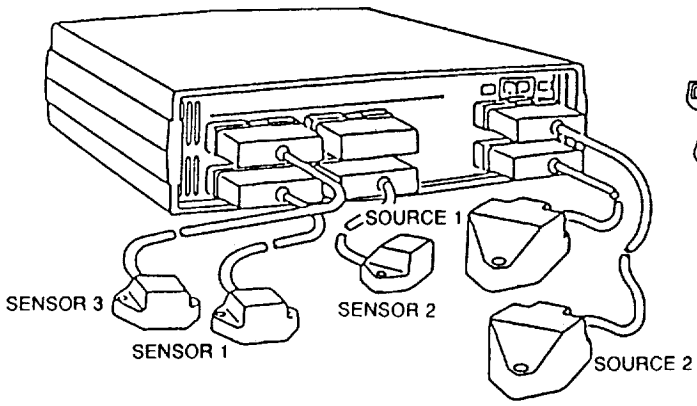

Figure 5a. Polhemus 3D Tracker System.
(Reprint with permission from Polhemus)

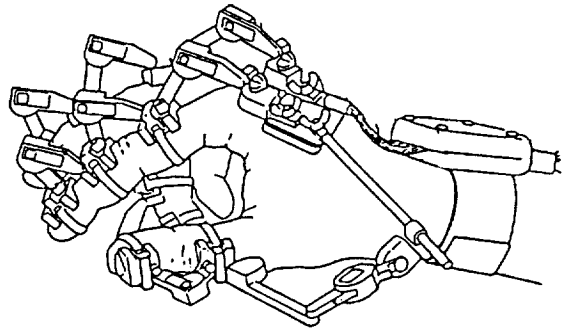

Figure 5b. EXOS Dexterous Hand Master.
(Reprint with permission from EXOS)

Variable I_R and I_L are the horizontal pixel count of target centroids in the right and left images, respectively. Once the Z component is found, values for X and Y are determined by

$$Y = Z \tan\theta_3 \quad ; \theta_3 = v\, J_R$$

$$X = Z \tan\theta_4 - \frac{d}{2} \quad ; \theta_4 = h\, I_R$$

The variable J_R is the vertical pixel count of the target centroid in the right image.

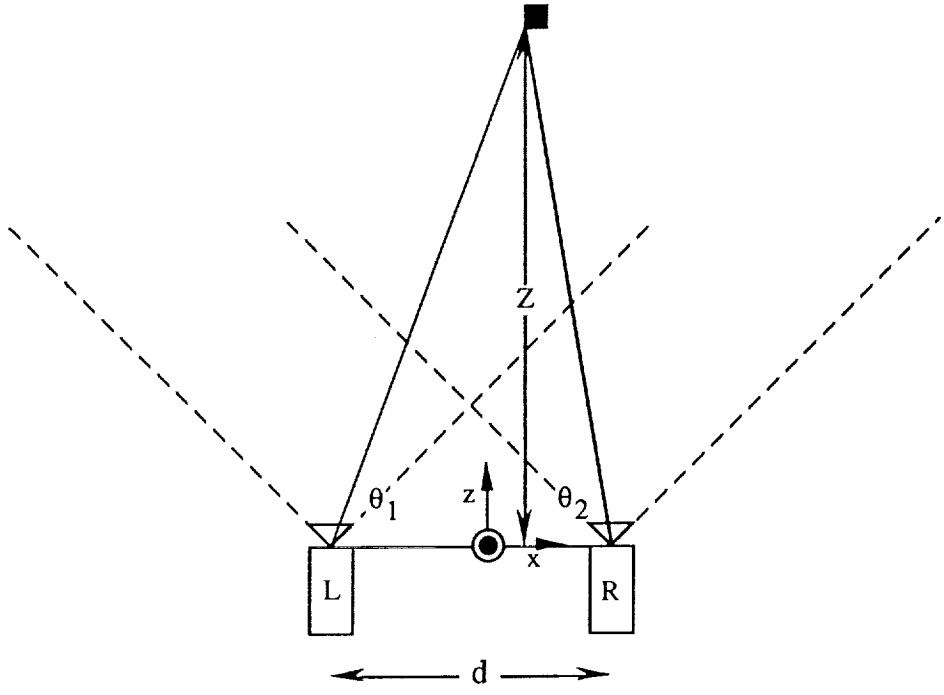

Figure 6a. Schematic of the stereo vision algorithm.

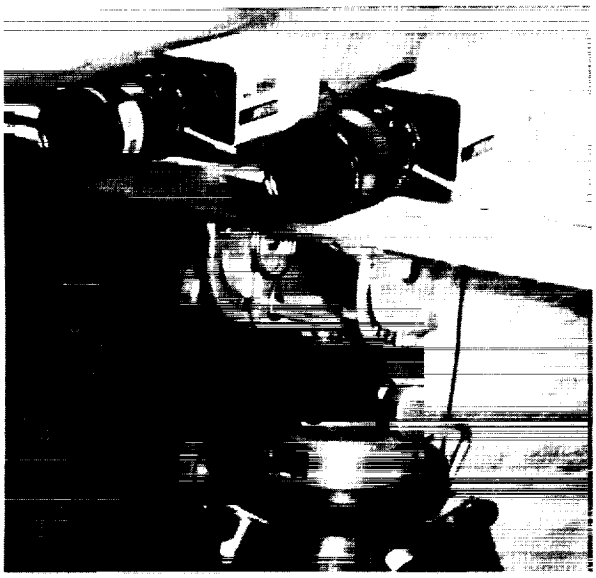

Figure 6b. Video camera configuration.

The stereo vision tracking algorithm was developed on a Sun 3/260 workstation using an Oasis C compiler. The cameras are made by Javeline, Model JE7362. The two frame grabbers are made by Data Translation, Model DT1451. The tracking algorithm, running on a 20 MHz 68020 SBC, was able to provide a target position update rate of 10 frames per second.

Proximity Sensor Subsystem

Although the vision system is capable of target tracking, it is currently not capable of dealing with visual obscurity caused by having the arm and the hand coming between the cameras and the target during an act of reaching and grasping. Without guidance from the vision system, other means of determining when to grasp (i.e., close hand) is necessary. Hess and Li[5,6] (U.S. Patent No. 4,980,626) described a method of using a proximity sensor to determine when and how to grasp a target with a dexterous end effector. Figure 7a is a drawing of the infrared proximity sensor which was originally an optoelectronics part developed by Optek, formerly a division of TRW (Part No. OPM102T). However, this part has become obsolete since then. There are eight sensors embedded inside the Utah/MIT Hand, with two in each finger, as is shown in Figure 7b. The sensors are reflective, with transmitter and receiver co-located on the same substrate. Due to its low power, the sensor does not have a very long detection range. A special signal amplifier was built to increase the signal power and to filter out noises. As a result, the sensors can now detect objects at approximately 1 to 2 inches away. The amplified signals are digitized by an A/D converter and transmitted, via a high-speed serial fiber-optic link, to the VME multiprocessing controller. No special software communication protocol was required because the fiber-optic link multiplexes and transmits signals at a high frequency of 125 MHz, thus making the interface on both sides of the link appear fully parallel.

ROBOT PROGRAMMING IN CLIPS

One of the most interesting features of the testbed system is the incorporation of an expert system shell, CLIPS, for high-level control. CLIPS has added many interesting and useful features to our system:

a. Intelligent rule-based control
b. Interactive user interface
c. User-defined functions
d. Portable C source codes
e. English-like syntax
f. Command clustering

The first feature provides machine reasoning capability for the robot. The interactive user interface allows the robot programmers to call on different robot primitives without having to recompile code every time. This feature is very important during application development and debugging. Once the entire algorithm (i.e. set of rules) is developed, it can be converted into binary form for batch mode operations. As part of the CLIPS package, a simple software interface was provided to link into user-defined subroutines. In our implementation, these subroutines provide primitive robot control functions such as MOVE-ARM or MOVE-HAND. Once the user-defined subroutines are linked into CLIPS, they become part of the

CLIPS command set. Table II lists some of the user-defined functions added to the CLIPS command set.

Source codes written in C are provided with the CLIPS software package. The codes were written in such a way that minimizes operating system and hardware dependency. The software has been successfully ported to PC, Sun 3 Unix workstations, VAX/VMS, Macintosh, and other machines. Since CLIPS is a symbolic production system with English-like syntax, it allows the robot application developer to program the robot in a descriptive language. The user-defined commands listed in Table II are some good examples. Furthermore, with CLIPS, new robot commands with a higher level of abstraction may be constructed from lower level primitives. Consider the following set of CLIPS rules:

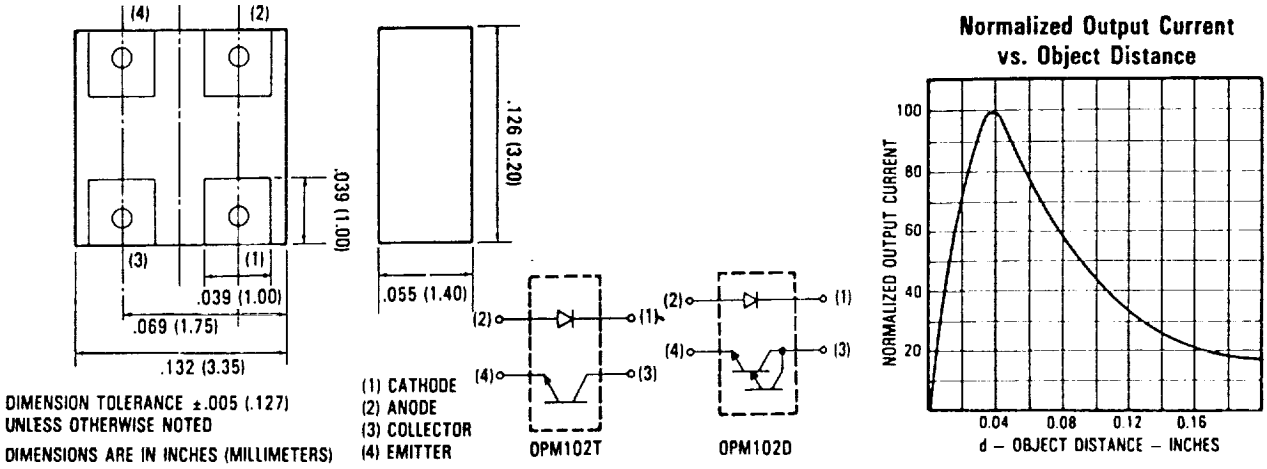

Figure 7a. Infrared proximity sensors. (Reprint with permission from Optek)

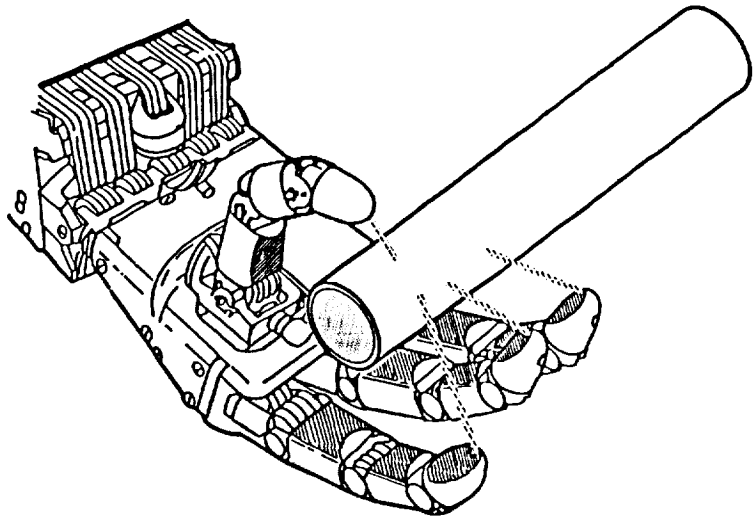

Figure 7b. Utah/MIT Hand with proximity sensors.

Rules 1, 2, and 3 are considered primitives. Rule 4 is a higher level command that produces a combined effect of having Rules 1, 2, and 3 firing together. To fire Rule 4, one simply loads in the set of rules listed above and executes the following CLIPS command:

(assert (acquire the target))

This assertion will cause Rules 1, 2, and 3 to fire, thus producing a combined action of looking, reaching, and grasping.

FUTURE WORK

The past effort has resulted in the completion of a single arm/hand system operating in both autonomous and partially teleoperated modes. The current focus is on perfecting the teleoperator system by integrating the Polhemus 3D Tracker into the system for arm control. Furthermore, the development of a dexterous robotic system with two arms and two hands is being pursued. Computer control is currently being developed for the Stanford/JPL Hand. We will be mounting the Stanford/JPL Hand on a second PUMA to form a second arm/hand system. The PC 386SX computer will be replaced by a Sun 3/160 workstation running Unix BSD 4.3. The Sun 3/160 workstation will provide a better software development environment for the programmers due to its window environment (Graphical User Interface and X Window). Neural network algorithms for learning the mapping between visual inputs and arm/hand position commands will be investigated in detail this year. Design and fabrication of a VMEbus-based neural network

TABLE II. SOME EXAMPLES OF USER-DEFINED CLIPS COMMANDS

USER-DEFINED CLIPS COMMANDS	DESCRIPTION
(move-arm *speed x y z rx ry rz*)	Move arm to the specified position and orientation with a given speed
(move-arm-home *speed*)	Move arm to predefined home position
(move-tips *speed x0 y0 z0 x1 y1 z1....x3 y3 z3*)	Move fingertips to specified Cartesian locations at the specified speed
(move-joints *speed J0 J1 J2... J15*)	Move finger joints to specified joint angles at the specified speed
(get-joint-pos) (get-visual-pos)	Get joint/target position and assert the information in the fact-list
(arm-move-done) (hand-move-done)	Returns DONE flag for arm/hand moves
(read-hand-ir)	Invoke real-time proximity sensor readout for check-out and debugging purposes
(grasp-with-ir *speed ir__threshold*)	Grasp at specified speed if infrared (IR) sensors total activity exceeds threshold
(see-reach-grasp *speed approach__distance o a t*)	Move arm/hand to visually determined target position and grasp default speed

```
; Rule 1: get target position
        (defrule  get__target__position__primitive
            ?old__fact <- (get target position)   ;   fact-list trigger pattern
        ⇒
            (retract ?old__fact)          ;delete old fact
            (get-target-pos)        ; call on user-defined subroutine to assert target position)
                                          ; information onto the fact-list

; Rule 2: arm movement primitive
        (defrule  arm__move__primitive
            ?old__fact1 <- (target position is ?x ?y ?z))   ;fact-list trigger pattern #1
            ?old__fact2 <- (move arm to target with speed ?speed)          ; fact-list trigger pattern
        ⇒
            (retract ?old__fact1 ?old__fact2)         ;delete the old facts
            (move-arm ?speed ?x ?y ?z 90 -180 0) ;call user-defined arm move subroutine)

; Rule 3: hand movement primitive
        (defrule  hand__move__primitive
            ?old__fact <- (grasp object with ir) ;fact-list trigger pattern
        ⇒
            (while (= (arm__move__done) do)  ;         wait 'til arm move is complete
            (retract ?old__fact)    ;delete the old fact
            (grasp-with-ir 1 25)  ;call  user-defined grasp subroutine for grasp)
        )

;Rule 4: acquire the target -- combining Rule 1, 2, and 3.
        (defrule  acquire__the__target__primitive
            ?old__fact <- (acquire the target)
        ⇒
            (retract ?old__fact)
            (assert (get target position))     ;this assertion causes Rule 1 to fire, producing
                                                    ;(target position is x y z) fact
            (assert (move arm to target with speed 100))      ; this assertion causes
                                                                ; Rule 2 to fire, moving the arm
            (assert (grasp object with ir))    ;this assertion causes Rule 3 to fire, closing hand
        )
```

board is planned for this year. New vision hardware and software will be acquired and developed to increase the vision system performance. Finally, we will continue to investigate and to develop space robotics applications. Demonstration of candidate space-related tasks are planned for the last quarter of 1991.

CONCLUSIONS

This paper has described the implementation of a dexterous robotic testbed system developed at the NASA Johnson Space Center. The system included several desirable features found in other systems. The system is unique in the fact that all these desirable features are integrated into one system, thus making the system capable and flexible. These features include dexterous hand and arm, stereo vision, multiprocessing, rule-based programming, local hand control using proximity sensor feedback, and autonomous and teleoperator capabilities in co-existence for shared and traded controls. Some of these features were enhanced to increase system flexibility and capability. Subsystem implementations were described in detail; and examples of rule-based robot

programming in CLIPS were also given. Our experience in the past year has revealed the need for additional development in the following areas: target tracking, image understanding, dexterous manipulation, force-reflective dexterous hand/arm masters, and tactile sensing. Although more vigorous tests still need to be conducted, the initial evaluation of the system has demonstrated that dexterous robots provide more capability and flexibility than conventional robots, and therefore, have an important role in future space applications.

ACKNOWLEDGEMENT

Special thanks to Reginald Dawson and Dagoberto Rodriguez for developing the fiber-optic electronics used in our system. The assistance provided by Robert Davis, Issa Zaid, and Frank Moore is greatly appreciated. We also would like to thank Dr. Jon Erickson and Cliff Hess for reviewing the paper, and providing valuable comments and criticisms.

REFERENCES

1 Allen, P., Michelman, P., Roberts, K., "An Integrated System for Dexterous Manipulation," Department of Computer Science, Columbia University, New York, NY, 1989.

2 Allen, P., Roberts, K., "Haptic Object Recognition Using a Multi-Fingered Dexterous Hand," Department of Computer Science, Columbia University, New York, NY, 1989.

3 Clark, D., Demmel, J., Hong, J., Laferriere, G., Salkind, L., Tan, X., "Teleoperation Experiments with a Utah/MIT Hand and a VPL Data Glove," *Proceedings: NASA/JPL Space Telerobotics Conference*, Pasadena, CA, January 1989.

4 Giarratano, J., *CLIPS User's Guide, Version 4.3*, NASA/Johnson Space Center, June 1989.

5 Hess, C., Li, L., "Proximity Sensors Make Robot Dexterous," *NASA Tech Briefs*, October 1990, pp 50.

6 Hess, C., Li, L., "Smart Hands for the EVA Retriever." *Proceedings: The Third Annual Workshop on Space Operations Automation and Robotics, 1989*, NASA Reference Publication 3059, pp 441-446.

7 Jacobsen, S., C., Wood, J., E., Knutti, D., F., Biggers, K., B., "The Utah/MIT Dexterous Hand: Work in Progress," *The International Journal of Robotics Research*, Vol. 3, No. 4, Winter 1984, pp 21-50.

8 Marr, D., *Vision*, Freeman and Company, New York, NY, 1982.

9 Narasimhan, S., "Dexterous Robotic Hands: Kinematics and Control." *Master Thesis*, Department of Electrical Engineering and Computer Science, MIT, January 1988.

10 NASA/Johnson Space Center, *CLIPS Reference Manual, Version 4.3*, JSC-22948, May 1989.

11 Salisbury, K., Brock, D., O'Donnell, P., "Using an Articulated Hand to Manipulate Objects," *Proceedings of the 1987 SDF Benchmark Symposium on Robotics Research*, Santa Cruz, CA, August 1987.

12 Stansfield, S. A., "Knowledge-Based Robotic Grasping," *Proceedings: IEEE Conference on Automation and Robotics, May 1990.*

13 Stansfield, S. A., "Reasoning About Grasping," *Proceedings: 7th AAAI Conference*, St. Paul, MINN. August 1988.

14 Venkataraman, S. T., Iberall, T., (Eds), *Dexterous Robotic Hands*, Spring-Verlag, New York, NY, 1990.

15 Wolovich, W., *Robotics: Basic Analysis and Design*, HRW, New York, NY 1987.