N90-25573

A LABORATORY BREADBOARD SYSTEM FOR DUAL-ARM TELEOPERATION

A. K. Bejczy, Z. Szakaly and W. S. Kim
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

ABSTRACT

The computing architecture of a novel dual-arm teleoperation system is described in this paper. The novelty of this system is that (i) the master arm is not a replica of the slave arm, it is unspecific to any manipulator and can be used for the control of various robot arms with software modifications, and (ii) the force feedback to the general purpose master arm is derived from force-torque sensor data originating from the slave hand. The computing architecture of this breadboard system is a fully synchronized pipeline with unique methods for data handling, communication and mathematical transformations. The computing system is modular, thus inherently extendable. The local control loops at both sites operate at 1000 Hz rate, and the end-to-end bilateral (force-reflecting) control loop operates at 200 Hz rate, each loop without interpolation. This provides high-fidelity control. This end-to-end system elevates teleoperation to a new level of capabilities via the use of sensors, microprocessors, novel electronics, and real-time graphics displays. The paper concludes with the description of a graphic simulation system connected to the dual-arm teleoperation breadboard system. High-fidelity graphic simulation of telerobot (called Phantom Robot) is used for preview and predictive displays for planning and for real-time control under several seconds communication time delay conditions. High fidelity graphic simulation is obtained by using appropriate calibration techniques.

INTRODUCTION

A laboratory breadboard system has been developed at JPL for dual-arm teleoperation using a novel generalized bilateral control method for robot (or slave) arm control. Generalized bilateral control of robot arms denotes (i) the ability to control the motion of a robot arm from another, dissimilar robot arm or device and (ii) the ability to reflect the forces sensed by the robot hand back to the hand of the operator. Since the controlling device (the hand controller or HC) is not similar to the robot being controlled, the HC can be designed to perform the task of control and force feedback best, and subsequently, this device can be used for the control of different robot arms[1]. To generate force feedback the HC has to be equipped with motors just like a robot and the control electronics of a robot and a HC can be made identical. In space telerobotic applications the control station may be some distance away from the robot so the control computations have to be carried out at two sites, the local or control station site and the remote or robot site.

An evolving electronic system is under development at the Jet Propulsion Laboratory (JPL) that was designed to solve the motor control and computational tasks of generalized bilateral control. This electronic system (The Universal Motor Controller or UMC) was used to build a generalized bilateral robot control system with PUMA 560 manipulators. These manipulators are equipped with Smart End Effectors (SEE) that sense the wrist and the grasping forces. The signals from the SEE are used to achieve force feedback to the hand controller and to achieve shared manual and automatic control of robot arms. An example of this shared control is when during peg insertion into a hole the robot automatically aligns the peg orientation while the operator translates it into the hole.

It is noted that in conventional teleoperation systems the master arm is a one-to-one or scaled replica of the slave arm and force feedback to the master arm is not derived from forces and moments sensed at the robot hand. Instead, it is essentially derived from position error between master and slave arm joints.

Note also that the control and computational system implied in generalized bilateral control of robot arms also forms a natural base for a supervisory control system of telerobots. In a supervisory control system, manual and automatic control can be traded or shared on the level of task space or work space variables. Thus, evolving capabilities in automation can easily be added to the generalized bilateral control and computational system described in this paper.

The breadboard system currently consists of: (1) two six degree-of-freedom (dof) PUMA 560 robot arms, each equipped with a JPL smart robot hand; the hand is a parallel claw device equipped with a six dof force-torque sensor, grasp force sensors and local processing and control electronics. (2) Two six dof generalized Force-Reflecting Hand Controllers (FRHC), each permits one-hand manual commands in six directions, three translation and three orientation commands either in position or in rate mode; the FRHC is unspecific to any manipulator, it can be used for the control of various robot arms with software modifications. (3) Two computing nodes for control and information display, one at the robot site and one at the FRHC (control station) site. (4) A computer graphics terminal at the control station, utilizing (a) a PARALLAX graphics board to generate real-time sensor information displays and (b) an IRIS graphics super workstation to generate real-time perspective images of robot arm motion either on a mono or on a stereo monitor for preview or predictive displays to aid motion planning or control under communication time delay conditions. The current status of the dual-arm teleoperation system with smart hands and with related control station setting is shown in Fig. 1.

In the first part of the paper the electronic architecture and design choices are discussed. This is followed by the description of the current teleoperation system and the upcoming new developments. The last part of the paper contains the description of a graphics simulation system connected to the dual-arm teleoperation breadboard system. High-fidelity graphic simulation of telerobots (called Phantom Robots) is used to create preview and predictive displays for planning and for real-time control of telerobots under several seconds communication time delay conditions. High-fidelity graphics simulation is obtained through appropriate calibration techniques described at the end of this paper.

ELECTRONIC ARCHITECTURE

The UMC architecture has been described in several publications where it can be found in more detail. See [2] and [3]. There are two tasks that have to be performed by such a system.

- Motor control and feedback signal sensing
- Mathematical computations

In our system an integrated approach was used so that both of the above tasks are carried out by a single electronic system. Since the mathematical transformations involved are complex, they cannot be performed by a single processor. This necessitates inter-processor communication and synchronization besides inter-node communication.

The following are the essential system components for which design choices have to be made:

- Power amplifiers
- Feedback data sensing elements
- Motor control hardware to joint servo processor communication
- Processors
- Inter processor communication
- Inter node communication
- Programming language and development environment
- Motor control algorithm
- Kinematic transformation algorithms

To achieve a compact, integrated package, the power amplifiers and feedback data sensing elements were developed in house. These with the joint processors constitute the UMC and have been described in detail in [2]. In short, this electronics consists of PWM power amplifiers for up to 1 kW motors and provides sensing of motion parameters at servo rates (1000 Hz). Thanks to the NASA technology utilization program, this electronics is now available commercially for up to 10 kW motors either brushed or brushless[4].

The communication from the motor control elements to the joint processor is a private bus called the BLX bus that makes the joint motion parameters memory mapped. It is noteable that with the UMC up to 16 joints can be controlled by a single joint servo processor.

The processor currently used is the NS 32016. There is a large number of processors from which we could choose and the 32000 family has proven to be a very good candidate for our task. The family has a number of processors with a wide performance range and object level compatibility between the members. Its assembly language has proven to be powerful as well as easy to use. The widely used 6800 family would provide less performance, less compatibility between

members and less symmetry in assembly language. Two more advantages of the 32000 family are the relatively small component count and relatively low bus clock rate per unit of performance. The small component count makes it easier to produce a radiation hardened version of a microprocessor, and the relatively slow bus timing makes it possible to time share devices or memory on the bus.

Figure 2 shows the overall architecture of the multibus based distributed computing for our two-node supervisory control system, including the UMC. The main electronic components with the related functions are shown on the board level in Figure 3 (1988 status).

To save development time we used the DB32000 development board which comes with a MULTIBUS interface. This forced us to use MULTIBUS for inter-processor communication. This is a lower bandwidth bus than more recent 32 bit busses. The available bandwidth is, however, more than enough for our application, so the use of MULTIBUS did not hamper the performance of our system. With the upcoming development of new processor boards (still using the 32000 family), a new proprietary bus (the ZBUS) will be introduced that is optimized for high bandwidth shared memory applications.

The inter-node communication currently is performed by a 5 Mbaud fiber-optic link that was developed in-house. Via this fiber optic link a single packet is transmitted every millisecond. This packet carries robot motion commands and also serves as a way of synchronizing all the computations in both the robot and the hand controller nodes. The forward communication link contains a software delay loop to be able to introduce an artificial time delay into the system. This time delay may be set from 0 to 4 seconds in 1 millisecond increments, for time-delay experiments.

Currently the forward packet carries the following information:

- Control mode
- Position change commands for the six degrees of freedom
- Finger grasping force command
- Checksum

Control Modes

The control modes are the following:

- Freeze mode; the robot sets the brake and servos the wrist joints to their positions when freeze mode was entered.
- Neutral mode; the robot is gravity compensated but it may be moved by hand to any position desired. Since the gravity load is compensated by software, when left alone the robot will stay at whatever position it was moved to.
- Current mode; the six bytes following the mode byte will directly command the currents of the six joints. In current mode gravity compensation is still active so at 0 current the robot will not move unless there are external forces acting on it.
- Joint mode; the six motion command bytes will be added to the joint space setpoints, moving the robot in joint space.
- Task mode; the six motion command bytes will be added to the Cartesian setpoints causing robot motion in the Cartesian frame. The so-called task frame is permanently attached to the laboratory, it cannot be redefined.
- Tool mode; the robot is commanded in Cartesian tool frame. This frame is defined by the robot wrist position at the moment the tool mode is activated. This is a Cartesian coordinate system that can be arbitrarily redefined during operation.

If the mode byte of an incoming packet is different from the active mode, the new mode is not entered until 1000 packets come in that all have the same mode bytes. During this intermediate period the robot does not move, any incoming motion bytes are ignored. A new mode has to be active for one second before the robot can be moved in that mode.

For example if the robot is in task mode, the transmitted data carries relative Cartesian coordinates. In every servo loop a change in the range of -D to +D is transmitted, where D is the current speed limit, typically 5 to 10. These changes are added by the receiver to the robot Cartesian setpoint number. This method has a number of merits:

- Small communication bandwidth used
- Error tolerance
- Velocity limiting
- Easy method of indexing the robot

It should be noted that this communication method does not cause any granularity in robot speed whatsoever. It simply limits the granularity of the robot position to 1/10th of a mm. The robot could not be positioned more accurately than that anyway.

The reply packet from the robot side contains the following information:

- Currently active mode

- Wrist forces
- Finger forces
- Finger position
- Joint positions
- Cartesian (task) positions
- Checksum

Development System

The programming language used was the assembly of the 32016 itself since this promised the most performance and the fastest results. It has to be noted that the most convenient development environment such as a C cross compiler and UNIX operating system does not necessarily produce the fastest result and the best program performance. Compilers have the tendency to mask the real world of a processor from the programmer making it harder to generate complex interrupt hierarchies and hardware interfaces. We used a development system that one of us (Szakaly) wrote for the IBM-PC. This system makes it possible to edit and store the assembly source programs in the PC as well as up and download object files. All functions of this development system are integrated so they pass data to each other in the memory of the IBM-AT. If the assembler finds an error for example, it automatically puts the user back into the editor with the cursor on the error. The system also keeps track of the files changed and remembers where each file was modified last. The typical assembly time for a 1000 line program is 15 seconds on a 10 MHz AT which includes the time it takes to write the object output, the symbol table and the memory map files to the disk.

Portions of the teleoperation system such as the force torque display were developed in C using the SYS 32/20 development environment marketed by National Semiconductor.

Control Algorithms

The motor control algorithm is a simple PD control loop. The servo rate is 1000 Hz overall, without interpolation, allowing high gains to be used with the associated high tracking fidelity. The position gains are about 0.1 V/encoder unit. The UMC code generator program is used in the joint level controller. This program assures safe robot control by automatically generating the servo code that controls the joints. There is a set of parameters that have to be specified once for every robot. These parameters are stored in an electrically erasable EEPROM chip. When the program is activated it generates servo code and executes it. There is no possibility of breaking the robot due to human error in the coding.

The code generator is very flexible, it can control any number of motors up to 16, with any combination of hardware elements such as encoders, pots, temperature sensors, motors, brakes. All polarities are menu items so, for example, instead of having to switch the two encoder wires, the user changes the encoder polarity from 'POS' to 'NEG' in the menu. The code generator will use a SUB instruction in place of an ADD in the servo code to accommodate the negative encoder hookup. The motor, the pot, the index and brake polarities can similarly be changed from the menu. The motor control processor interfaces to the rest of the system via the shared memory.

Since the remote node receives Cartesian position setpoints, the inverse kinematic transformation is needed to calculate the robot joint position setpoints. This is carried out by one of the processors on the robot side. This transformation was implemented in integer arithmetic and takes around 700 µsec to execute. Force feedback to the HC is based on robot position error as well as sensor data so the robot end effector Cartesian position has to be computed as well. This is done by computing the robot forward kinematics.

Breadboard Capabilities

As of 6/89 the dual-arm teleoperation system consists of the following major parts (see also Figure 1):

- Two Hand Controller mechanisms
- Local node MULTIBUS cardcages
- Force torque graphic displays
- IRIS workstation with PUMA solid shaded graphic simulation
- IBM-PCs as user interfaces
- PUMA 560 manipulators
- Remote node MULTIBUS cardcages
- Smart End Effectors

The local node cardcage contains the following:

- Two joint interface cards (part of local UMC)
- PWM amplifiers for 8 motors (part of local UMC)
- Joint processor (part of local UMC)
- Kinematic transformation processor
- Communication processor with user interface
- Graphics processor
- Parallax graphics card

The remote node cardcage contains the following:

- Remote node UMC (3 cards and power amplifiers)

- Communication processor
- Smart Hand processor
- Inverse kinematic processor
- Forward kinematic processor

The interfaces are as follows:

- Between cardcages: 5 Mbaud fiber optic links
- From local node to IRIS robot simulation: Fiber optic RS232 at 9600 baud rate.
- From remote node to Smart End Effectors: Fiber optic RS232 at 9600 baud rate for the right hand, fiber optic 3 Mbaud communication for the left hand.

Figure 4 shows the block diagram of the system in its current 1989 status and the interconnections. Figure 5 indicates the timing of events and the sequence of computations. All computations are carried out at a 1000 Hz servo rate. The force feedback signal is currently received at a 125 Hz rate due to the limitation of the RS232 communication channel used. The total round trip time delay is 5 msec for the position error based force feedback and it is around 10 msec for the force-torque sensor based feedback.

The user has a large number of options available through the user interface. Every parameter can be changed on a degree of freedom basis. It is possible to activate a software spring on any degree of freedom that pulls the user's hand back to a center position. Any DOF may be in position or rate mode or it may be turned off. Any degree of freedom can have arbitrary force compliance with a zero or non-zero force setpoint. For example, orientation compliance with zero torque setpoint amounts to automatic peg alignment when performing peg insertion into a hole. An X compliance with non-zero force setpoint will press the end effector against the task board and will maintain contact force. Rate mode is useful when motion over large displacements is desired or when slow, constant velocity motion is the requirement.

The breadboard system multi-mode control flow diagram is shown in Figure 6. The multi-mode control capabilities are described in detail in [5]. Active (that is, force-torque sensor referenced) compliance control and its implementation through a low pass filter is described in detail in [6].

Extensive experiments have been conducted to evaluate the usefulness of these operating modes and force feedback. The data show that force feedback brings an improvement in terms of execution time as well as total force required. The shared control routines also bring about additional improvements. Performance evaluation experiments and results are described in detail in a recent comprehensive report [7].

REAL-TIME GRAPHICS SIMULATION

A real time graphics simulation of the PUMA arm motion has been accomplished by using a Silicon Graphics IRIS-4D GT system. The system is extremely fast both in computation (10 MIPS and 1.1 MFLOPS) and in graphics display. The system can draw 400,000 vectors or 40,000 polygons (4-sided) per second with hidden surface removal and lighting. Thus we could easily achieve the update rate of the PUMA arm graphics simulation to be as fast as the display refresh rate, 60 frames/s for workstation display and 30 frames/s for NTSC video monitor display. Perspective projection was assumed for display, and double buffering was used for the PUMA arm graphics animation to avoid visible flickers or partial drawings. Namely, two display buffers (two 24-bit-per-pixel RGB color image memory buffers) in contrast with a single display buffer were used for display and update in an alternate manner; while one is used for display, the other is used for new drawing, and then the two buffers are switched. Both a solid model with hidden surface removal and a wireframe model with hidden line removal are available for our PUMA arm graphics simulation/animation.

A geometric model of the PUMA 560 arm was constructed by using only 6 object types: 6 boxes, 12 cylinders (frustums), 1 forearm, 1 upperarm, 1 wrist, and 4 fingerhalves. The data structure of the box specifies the box material (color), origin and size. The data structure of the cylinder specifies the cylinder material, origin, bottom and top radii, height, and number of side panels to approximate the side with polygons. The data structure for the other object types were similarly defined. The Denavit-Hartenberg representation was used for the kinematic modeling of the PUMA arm.

Hidden surface removal of the solid model was done by use of the z-buffer of the IRIS graphics system. The z-buffer (24 bits per pixel) contains the z-value data indicating the distance (depth) from the viewpoint for each pixel. At the beginning of each display frame, the z-buffer is initialized to the largest representable z-value (7fffff in hex), while the RGB buffer (24 bits per pixel) contain ing the red, green, and blue color values is initialized to the background color value. Then during the drawing of polygons, lines, points or characters, the

IRIS graphics system updates the RGB buffer and the z-buffer only for those pixels whose new z-value associated with the current drawing is less than the existing z-buffer value.

The lighting calculations were also done by use of the IRIS graphics system hardware. Once the user defines the material properties (diffuse reflectance, specular reflectance, specular shininess, emission color, ambient reflectance, transparency), light source properties (color of the light source, position or direction of the light source, ambient light associated with the light source), and light model properties (ambient light presented in the scene, scene attenuation factor, local viewer property), the IRIS graphics hardware automatically takes care of the lighting calculations.

It is sometimes advantageous to use a wire-frame model with hidden line removal instead of using a solid model. When the wire-frame model of the PUMA arm is overlaid on the camera view, the viewer can still see the actual camera view of the arm. The wire-frame model with hidden lie removal was accomplished by first drawing the arm with filled polygons of the background color and then drawing the arm again with solid lines of white color. In order to avoid appearance of many broken lines, a small positive depth offset (0.001 in the normalized depth coordinate) was introduced during the filled polygon drawing.

Pop-up menus were provided for the user interface with the PUMA arm graphics simulation. By using a mouse and selecting appropriate menu/submenu commands, the user can perform view control (view angles, view position, zoom), light position control, PUMA arm motion control (6 joint angles and hand opening), screen selection (workstation screen or NTSC video monitor screen), graphics model selection (solid model or wire-frame model), camera calibration, or graphics overlay.

Graphics Overlay on TV Camera Image

The real time graphics overlay of the IRIS graphics output on the video camera image was achieved by using an IRIS video genlock board. The genlock board enables the IRIS graphics output to be synchronized with the incoming video camera signal. It also provides video switching function. Namely, the video output of the genlock board, which is connected to the video monitor for display, can be switched to either the incoming video camera signal or the IRIS graphics output signal, depending upon the alpha-plane value for each pixel. When the alpha-value of the pixel is 255 (ff in hex), the video camera signal is selected for the genlock board video output. When the alpha-value is 0, the IRIS graphics output is selected. Although the major function of the 8-bit alpha-plane of the IRIS graphics system is to allow blending or mixing of two graphics images, in our application we simply used the alpha-plane to control the video switch for the graphics image overlay (or superimposition) on the camera image. During the IRIS graphics rendering, the alpha-values for the background pixels are assigned 255, while the alpha-values for the pixels associated with the PUMA arm are assigned 0. 1 In this way, the PUMA arm graphics model generated by the IRIS graphics system is overlaid on the real camera view. The graphics overlay procedure is schematically summarized in Figure 7.

Camera Calibration

In order to superimpose the PUMA arm graphics model on the camera view of the actual arm, camera calibration is necessary. In our implementation, camera calibration was achieved by an interactive cooperation between the human operator and the system [8]. The operator provides the correspondences between object model points and camera image points by using a mouse. Thereafter the system computes the camera calibration matrix. The calibration procedure is summarized in Figure 8.

As the human operator selects the data entry mode from the camera calibration menu, the PUMA arm graphics model is overlaid on the real camera view, both the model and the actual camera view appearing on the video monitor screen (Fig. 9). At this stage, the graphics model view and the camera view are not aligned. In fact, the human operator is allowed to change the viewing condition (view angle, view position, zoom) of the model arm at any time during this data entry mode, so that the human operator can find and indicate corresponding points easily. Thirty three vertices (corner points) of the PUMA arm model were pre-selected as object points for camera calibration. As seen in Figure 9, these object points are indicated by square marks on the model arm. For clarity, only visible object points are marked.

The operator first picks an object point by clicking the square with a mouse. When the square is successfully picked, the unfilled square is changed to a filled square. The "pick" function call of the IRIS graphics system is efficiently used to identify which object

point is actually picked. After the identification, the 3-D position of the object point is directly obtained from the geometric model of the PUMA arm. This picking process enables us to determine the 3-D position of the object point, even though a mouse click gives only 2-D screen coordinates. After having picked an object point, the operator indicates, on the camera view of the arm, the location of the corresponding image point by clicking a mouse. This picking-and-clicking procedure is repeated until all desired object points and their corresponding image locations are entered. The data entered are now used to compute the camera calibration matrix.

The 4x3 camera calibration matrix describes the relation between 3-D object points and their corresponding 2-D image points by using homogeneous coordinates. With the assumption that the camera view can be modeled by an ideal pinhole camera model as a perspective projection of the 3-D world onto the 2-D image plane, we can consider the camera calibration matrix M as being composed of several simple transformations. While it is possible to decompose the matrix in a variety of ways, the particular decomposition chosen is as follows:

M = (rotate)(translate)(project)
(scale)(crop) = (3-D viewing transform)(perspective projection)
(2-D viewport transform)

The viewing transformation transforms object coordinates (x,y,z) to camera viewing coordinates (x_v,y_v,z_v) by a rotation and translation. The perspective projection transforms the viewing coordinates to image-plane coordinates (u,v). The viewport transformation (window-to-viewport mapping) maps image-plane coordinates to actual screen coordinates (u_s,v_s) by scaling and cropping (translation of the image center) within the 2-D image plane.

$$u_s = s_x u + c_x,$$
$$v_s = s_y v + c_y$$

There is a standard linear least-squares method that can compute the camera calibration matrix M, when 6 or more object points and their corresponding images are given [9], [10]. Once M is obtained, we can recover both intrinsic (2-D image scaling and cropping parameters including camera focal length) and extrinsic (camera position and orientation) camera parameters [11], [12]. However, our testings indicate that recovering camera parameters by this technique, especially scaling and cropping parameters, is very sensitive to measurement errors.

Fortunately, in or application the camera scaling and cropping parameters can be defined to be identical to the graphics viewport parameters. The full size of the camera view displayed on the video monitor screen is normally equal to the full size of the IRIS graphics output in NTSC mode displayed on the same screen since these two are synchronized by the IRIS genlock board. Thus the scaling and cropping parameters of the camera view are assumed to be identical to the graphics viewport parameters. In the NTSC mode of the IRIS graphics system, the screen size is defined as (XMIN,XMAX, YMIN,YMAX) = (0,645,0,484). Thus, viewport transformation parameters are given by $s_x = c_x = -$ XMAX/2 and $s_y = c_y$ = YMAX/2, and so are the camera scaling and cropping parameters. Thus, instead of computing the camera calibration matrix M, we first transform (u_s,v_s) screen coordinates to (u,v) image-plane coordinates for each image point by

$$u = (u_s - c_x)/s_x,$$
$$v = (v_s - c_y)/s_y,$$

Then, we compute the camera calibration matrix C that relates 3-D object coordinates (x,y,z) and 2-D image-plane coordinates (u,v) without 2-D image scaling and cropping.

C = (rotate)(translate)(project)

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_1 & t_2 & t_3 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

where f is the camera focal length.

A linear least-squares method can be used to determine the 12 elements of the 4x3 camera calibration matrix C, when 6 or more object points and their corresponding images are given [9],[10]. However, the linear method does not guarantee the orthonormality of the rotation matrix. In our graphics overlay application, the orthonormalized rotation matrix may be preferred. Orthonormalization can be applied after the linear method, but this does not yield the least squares solution. In general, a nonlinear least-squares method has to be employed if we wish to obtain the solution that satisfies the orthonormality of the rotation matrix.

In the nonlinear method, instead of using 9 elements of a rotation matrix, three angles (pan, tilt, swing) are used to represent the rotation. In our current design, all three camera calibration algorithms are available: (i) a linear least-squares method, (ii) orthonormalization after the linear method, (iii) a

nonlinear least-squares method. The algorithms above can be used for both cases: when the camera focal length f is given and when f is unknown. The solutions of the camera calibration matrix C obtained by the above algorithms are stored in different files. The user can pick any one of the camera calibration matrix solutions for rendering the PUMA arm graphics model and superimposing on the camera view.

The PUMA arm graphics model superimposed on the actual camera view after the camera calibration is shown in Figure 10 for the surface model and in Figure 11 for the wire-frame model. Also indicated on these figures is the predictive display "phantom robot" effect under communication time delay condition. As seen on the right side of Figures 10 and 11, the graphics robot image (the "phantom robot") has moved off from the real robot image on the screen to a location commanded by the operator. When the "phantom robot" motion on the screen is controlled by the operator in real time then the operator can see that motion against the real environment on the screen in real time, provided that the environment on the screen is a static one. The real robot image will follow the motion of the "phantom robot" graphics image after some communication time delay and will stop at the location of the "phantom robot" image on the screen, provided that the geometric calibration of the "phantom robot" graphics image relative to the real robot image on the screen was performed correctly before the motion started.

CONCLUSION AND FUTURE PLANS

The main conclusion is that this end-to-end dual-arm breadboard system elevates teleoperation to a new level of capabilities via the use of sensors, microprocessors, novel electronics, and real-time graphics displays. The new control and task performance capabilities have been verified and evaluated for single-arm operation through a statistically significant set of control experiments as reported in [7]. Dual-arm task performance experiments and time-delayed control experiments using predictive display graphics image of robot arm ("phantom robot") will be carried out in the near future.

Future plans in control system and electronics development affects the following areas:

- Processors and bus architecture
- Communication
- Smart end effectors
- Software development environment
- Supervisory control software

The upcoming new devices are the following:

A new processor card containing two of the NS 32016 processors using the new advanced bus interface and 5 Mbit fiber optic links. This processor card can also be used for upcoming flight experiments.

Another processor card using the NS 32332, the new advanced bus interface, 5 Mbit and 15 Mbyte fiber optic links.

A new smart hand featuring very high (10 kHz) data rates with a 12 bit A/D and the new fiber optic link. The actual servo rate will be limited by the host processor to about 5 kHz, this data will be processed to the 1 kHz rate of the rest of the system as described in [13].

After some experience with the new assembler, improvements will be made to the syntax such that the usage will have the appearance of a high level language. This will provide many of the benefits of high level languages without the associated performance and control loss.

When the new hardware is available, the control software will be upgraded to include evolving supervisory control capabilities in model- and sensor-referenced automatic control of the dual-arm system.

The plans also include the upgrade of the dual and non-redundant (six d.o.f.) arm hardware to a dual and redundant (eight d.o.f.) arm system.

Future plans in real-time computer graphics development include (i) the use of computer controlled TV cameras and (ii) graphics overlays of object models on the TV image. Use of computer controlled TV cameras will provide the capability of using a single complete camera calibration for a task scenario since the camera parameters will automatically be known for all different settings of camera position, orientation and zoom. Graphics overlays of object models on the TV image will enable preview/predictive simulation of sensor-referenced control.

ACKNOWLEDGMENTS

Control electronics and software was developed by Z. Szakaly, and the electronics hardware was built by E. Barlow. Graphics image development was done by S. Venema, and graphics overlay calibration techniques were developed by W.S. Kim.

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

[1] Bejczy, A.K. Salisbury, J.K. Jr., "Controlling Remote Manipulators Through Kinesthetic Coupling." Computers in Mechanical Engineering (CIME) Vol. 2, No. 1, July 1983, pp. 48-60.

[2] Bejczy, A.K. Szakaly, Z.F., "A Synchronized Computational Architecture for Generalized Bilateral Control of Robot Arms," Proceedings of the Conference on Advances in Intelligent Robotic systems, by SPIE and the International Society for Optical Engineering Cambridge, MA, Nov. 1-6, 1987.

[3] Bejczy, A.K. Szakaly, Z.F., "Universal Computer control System (UCCS) for Space Telerobots," Proceedings of the IEEE International Conference on Robotics and Automation, Raleigh, NC, March 30-Apr. 3, 1987, pp. 318-324.

[4] Motion Tek, Box 9, Lord Ave., Brunswick, NY 12180.

[5] Bejczy, A.K., Hannaford, B., Szakaly, Z.F., "Multi-Mode Manual Control in Telerobotics," Proceedings of ROMANSY'88, Udine, Italy, Sept. 12-15, 1988.

[6] Szakaly, Z.F., Kim, W.S., Bejczy, A.K., "Force-Reflecting Teleoperated System with Shared and Compliant Control Capabilities," Proceedings of NASA Conference on Space Telerobotics, Pasadena, CA, Jan. 31-Feb. 2, 1989.

[7] Hannaford, B., Wood, L., Guggisberg, B., McAffee, D., Zak, H., "Performance Evaluation of a Six-Axis Generalized Force-Reflecting Teleoperator," JPL Publication 89-18, June 15, 1989.

[8] Kim, W.S., and Stark, L., "Cooperative Control of Visual Displays for Telemanipulation", Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1327-1332, Scottsdale, AZ, 1989.

[9] Sutherland, I.E., "Three-Dimensional Data Input by Tablet", Proc. IEEE, vol. 62, no. 4, pp. 453-461, 1974.

[10] Ballard, D.H., and Brown, C.M., Computer Vision, Prentice-Hall, 1982.

[11] Ganapathy, S., "Decomposition of Transformation Matrices for Robot Vision", IEEE Int. Conf. oin Robotics and Automation, pp. 130-139, 1984.

[12] Strat, T.M., "Recovering the camera parameters from a transformation matrix", Proc. DARPA Image Understanding Workshop, pp. 264-271, 1984.

[13] Bejczy, A.K., Szakaly, Z., Ohm, T., "Impact of End Effector Technology on Telemanipulation Performance," - see elsewhere in this Proceedings.

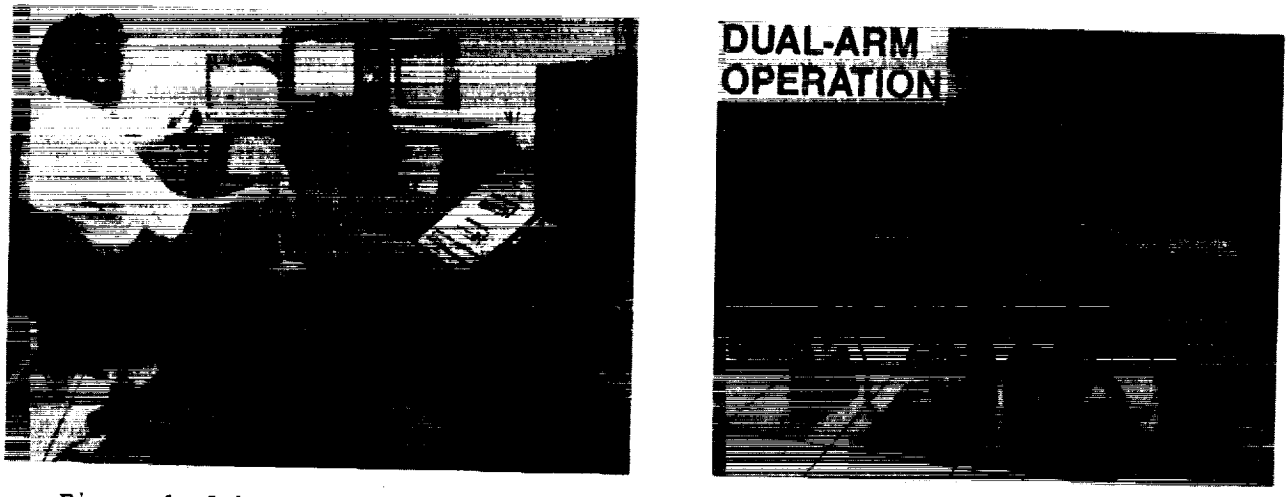

Figure 1. Laboratory Breadboard System for Advanced Dual-Arm Teleoperation (1989)

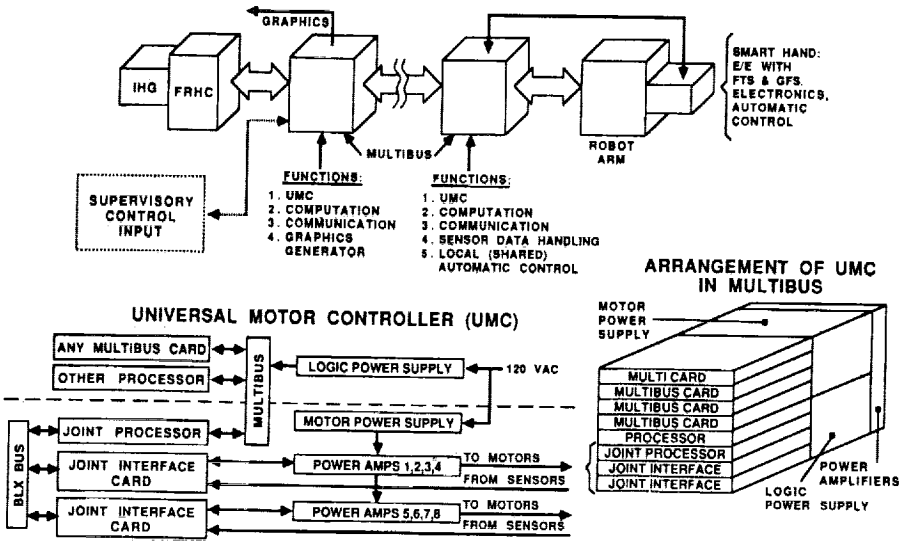

Figure 2. Electronics Architecture of Distributed Two-Node Supervisory Control System

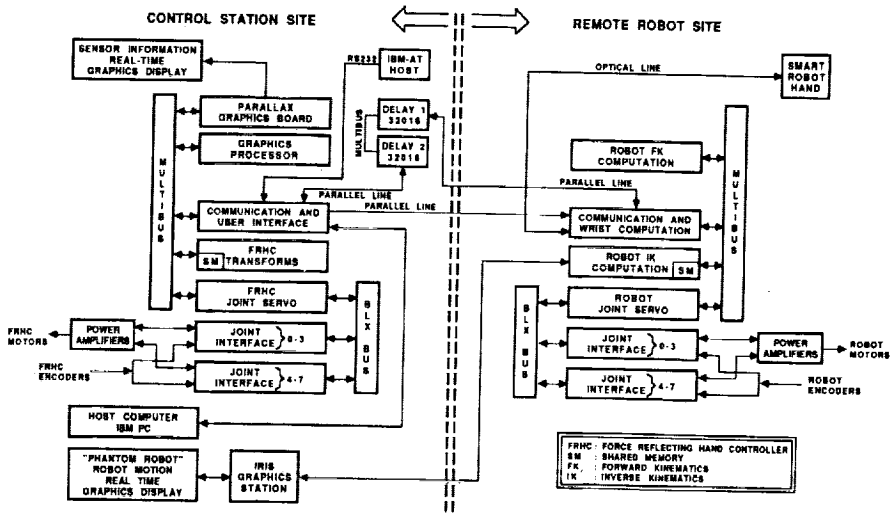

Figure 3. Board-Level Components of Distributed Two-Node Supervisory Control System Electronics (1988)

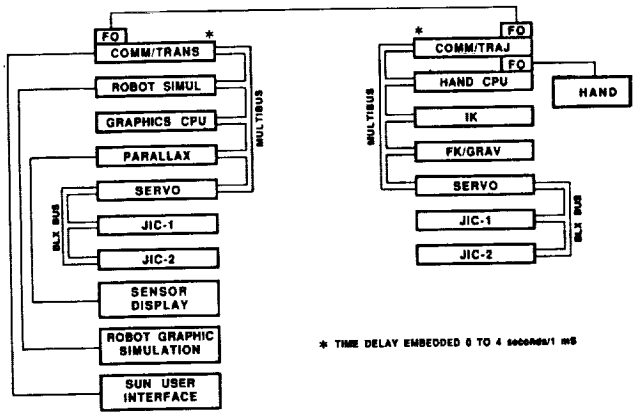

Figure 4. Supervisory Control System Electronics Upgrades (1989)

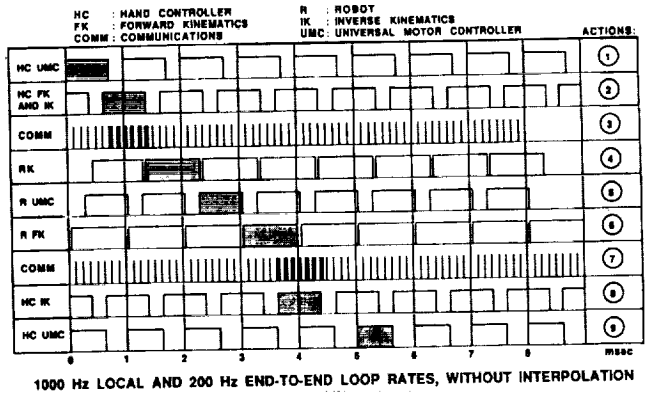

Figure 5. Bilateral Control Communication Timing Diagram

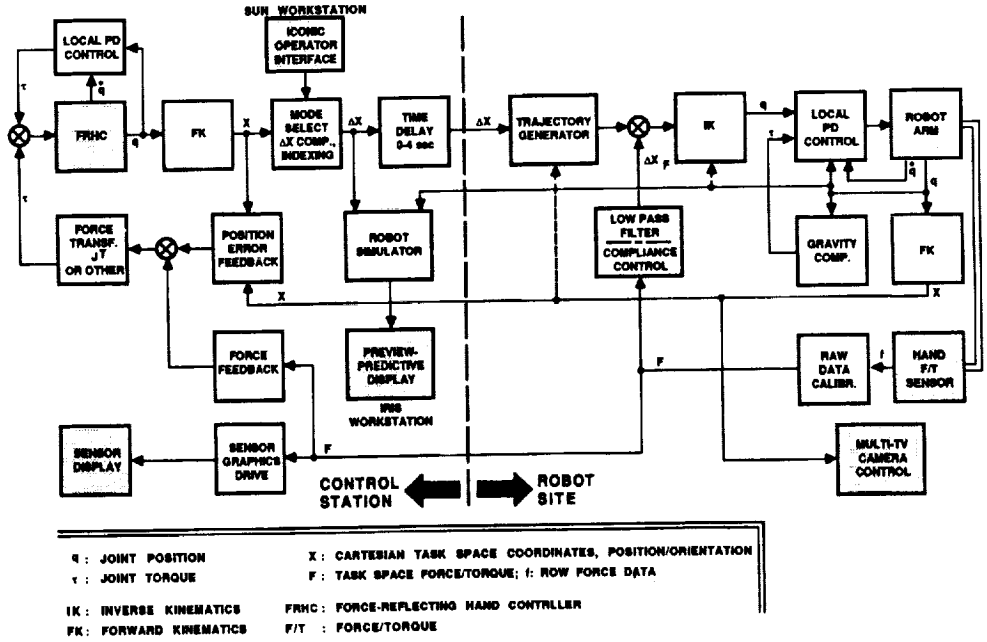

Figure 6. Advanced Teleoperation Control System Block Diagram

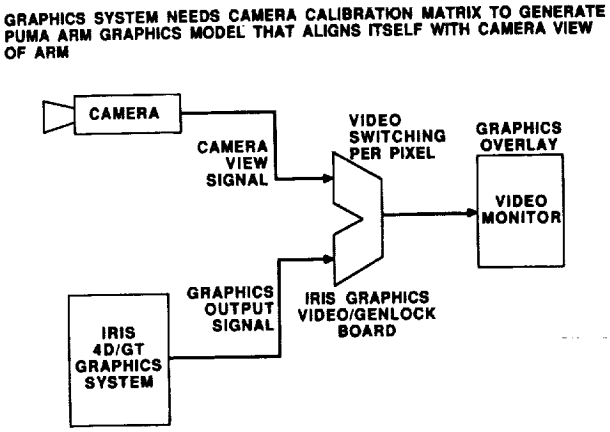

Figure 7. Graphics Video Overlay Procedure

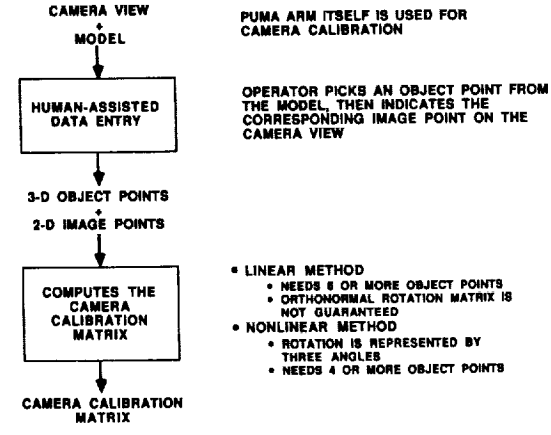

Figure 8. Graphics Calibration Procedure

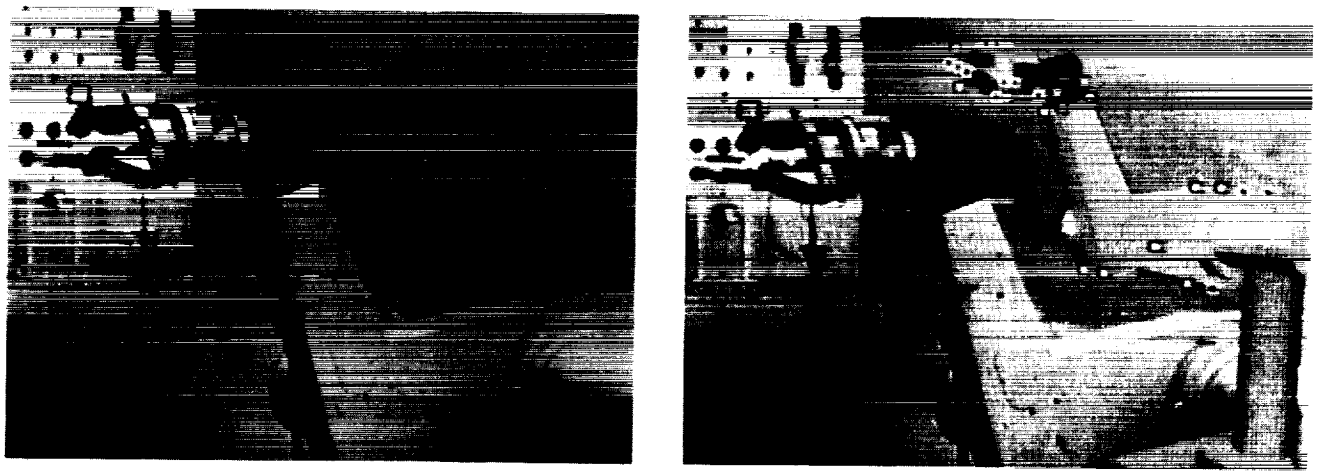

Figure 9. Visual/Manual Calibration of Graphics Overlay

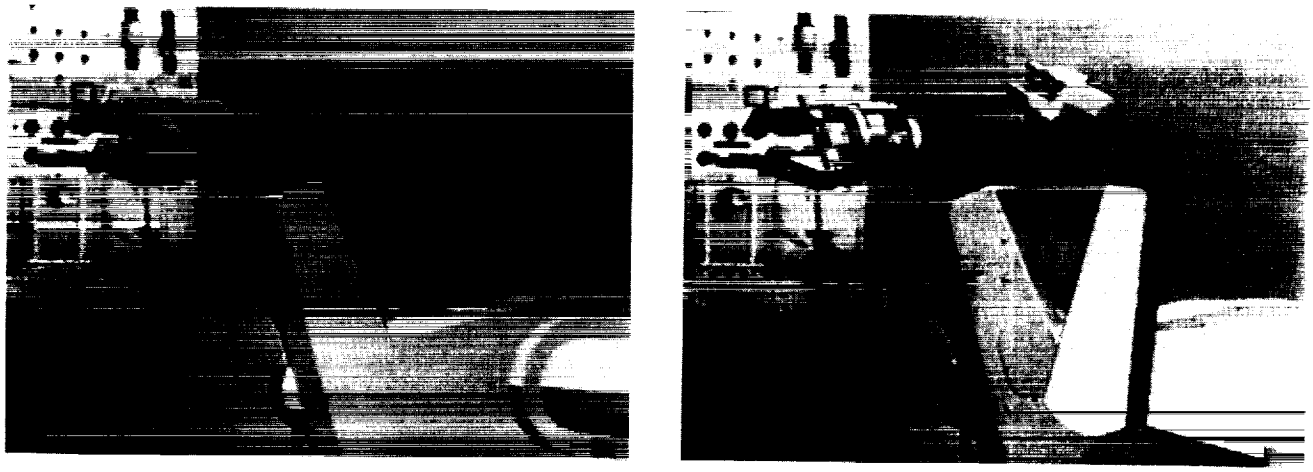

Figure 10. Solid Shaded Polygon Graphics "Phantom Robot" Calibrated Overlay and Predictive Display

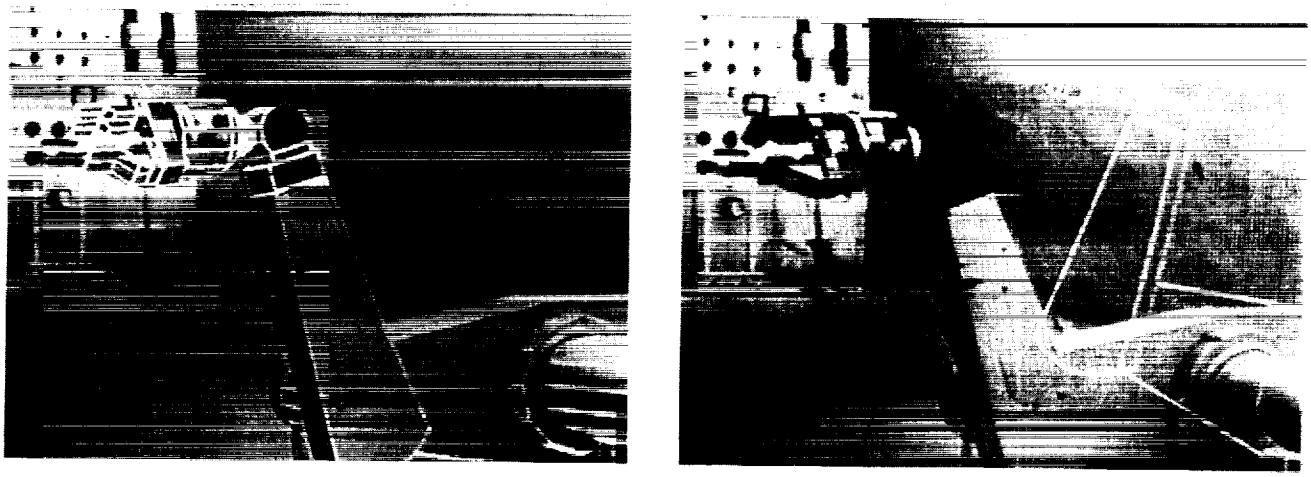

Figure 11. Wire-Frame Graphics "Phantom Robot" Calibrated Overlay and Predictive Display