IMPLEMENTATION AND DESIGN OF A TELEOPERATION SYSTEM BASED ON A VMEBUS/68020 PIPELINED ARCHITECTURE

Thomas S. Lee

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

This paper describes a pipelined control design and architecture for a force-feedback teleoperation system that is being implemented at the Jet Propulsion Laboratory and will be integrated with the autonomous portion of the testbed to achieve shared control. At the local site, the operator sees real-time force/torque displays and moves two 6-dof force-reflecting hand-controllers as his hands feel the contact force/torques generated at the remote site where the robots interact with the environment. He also uses a graphical user menu to monitor robot states and specify system options. The teleoperation software is written in the C language and runs on MC68020-based processor boards in the VME chassis, which utilizes a real-time operating system; the hardware is configured to realize a four-stage pipeline configuration. The environment is very flexible, such that the system can easily be configured as a stand-alone facility for performing independent research in human factors, force control, and time-delayed systems.

Introduction

Many existing teleoperation systems are designed to be purely teleoperative (i.e., to receive input commands solely from the operator). In many robotic applications, it is desirable to mix input commands from the operator as well as from a high level planner to have shared or traded control capability [1]. Space tasks such as bolting a screw on a space station platform need not be performed entirely by the astronaut nor under his continuous supervision. For example, he can perform gross motions such as moving the manipulator to the work vicinity, then trade the mode from teleoperation to autonomous control to allow the machine to complete the task by detecting the bolting location, then invoke compliance control as the bolt is being threaded. This saves the astronaut valuable time since he does not continuously monitor the task as it proceeds. This type of situation is referred to as traded control, for there is no mixture of input modes but a complete turnover of control — the robot is either under human or machine control, not a combination of both. There are many situations however when shared control is desirable or even necessary. One instance is when some form of force control is required. For example, as the robot hand moves along the surface in a window-washing situation, the operator can provide positional setpoints while depending on the machine to provide force control setpoints. In a situation of inserting a replacement module into a satellite, the astronaut can provide the positional information but have the machine provide the orientation information (aligning the module automatically as it is being inserted). In space applications, time-delay introduced in the control loop because of transmission delay can be handled by having a form of shared control (e.g., have the autonomous system at the remote site where the robot is situated handle all internally generated forces during the task and have the user on Earth provide positional commands).

The architecture and design of present day teleoperation systems is such that it is not trivial to incorporate inputs from an autonomous system. The major obstacle is to coordinate inputs (i.e., position or force trajectories) from the teleoperation and autonomous sides and synchronize them to ensure that the

resulting trajectories are consistent. There is need for an effective cooperation between the human and the machine in such a way as to have the human informed of what the machine is doing and vice versa. The Teleautonomous Systems Research Laboratory at JPL has adopted a hardware approach that incorporates various elements of shared control. The design of the teleoperation side was much influenced by experiences gained on previous teleoperation systems built at JPL [2, 3], and the desire was to port to it many concepts that were already proven and demonstrated in the teleoperation laboratory. Another objective was to build a system which has a flexible hardware and software development environment that can easily accommodate various modes of shared control in position, orientation, force, and torque domains, obtaining commands from the human operator and/or the autonomous system.

This paper is organized into 5 sections and an appendix. Section 1 provides a description of theoretical aspects of how force-feedback teleoperation is achieved. Section 2 describes the hardware and software environment that exists in the laboratory. In Section 3, the details of pipeline implementation are elaborated, especially the aspects that pertain to timing. Section 4 describes the user interface and how it is achieved. The main text of the paper is concluded in Section 5.

1. Overview of Teleoperation Concepts

The teleoperation system accommodates various modes of operation: joint, Cartesian, rate, index, and force-feedback modes. Joint mode is implemented by having a one-to-one mapping of each hand controller's degrees of freedom (DOF) to that of the robot — this mode is used to test hardware interfaces and to move the robot out of kinematic singular positions. Scaling is involved since the angular ranges of the robot and those of the hand controller are not equivalent. Cartesian mode is when the operator moves the robot in position control mode, having the end-effector referenced with respect to the robot tool frame or the defined world frame. There is a one-to-one correspondence between the motion of the operator's hand and the motion of the robot end-effector. If the robot is in rate mode (either Cartesian or joint space), the robot speed is controlled relative to the amount of deflection of the hand controller handle from its initial start-up (neutral) position. In this mode, the hand controller is in a "spring-return" state such that if the handle is released the hand controller will return to the neutral position (the effect is analogous to a spring-return joystick). Index mode allows the user to extend the workspace of the hand controller. Once a bound of a certain hand controller joint is reached, the user presses the index button to inform the system to disregard hand controller input (i.e., not to move the robot). He then moves the hand controller away from the joint bound and presses the index button to reactivate the robot. Finally, force-feedback is a mode that allows the operator to feel on his hand the forces/torques that are generated at the tool tip of the robot as it interacts with the environment.

The following paragraph explains the theory of operation when the system is under Cartesian control mode. Refer to Figure 1. The forward loop of the teleoperation system during each sampling interval is the path from the hand controller sending incremental trajectory information to the remote site, which directs the robot to move as commanded by the operator moving the hand controller. Positional information is relayed from the local to the remote site by sending incremental ΔX information. This Cartesian ΔX information is computed by premultiplying the $\Delta\theta$ angular values with the hand controller Jacobian expressed with respect to the tool (or base frame). Once the ΔX information is received at the remote site, it is transformed to be expressed with respect to the robot tool (or base frame). The transformation matrix has the following form:

$$ {}^{R}J_{H} = \begin{bmatrix} R & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & R \end{bmatrix} $$

Typically this transformation is a constant 6×6 rotational matrix and accounts for the orientational difference between the hand controller and the robot. In a space environment, the robot base may have a moving orientation (e.g., the space platform where the robot is placed may move with respect to the shuttle where

the hand controller is placed) — the 6 × 6 transformation matrix would not be constant in this case. The $\Delta\theta$ is calculated for the robot by premultiplying the transformed ΔX_R with the computed inverse Jacobian of the robot. Indexing is accomplished by sending $\Delta X_H = 0$ from the local to the remote site. At times, the robot may drift (i.e., absolute position error will exist between the hand controller and robot positions due to accumulation of θ differentiation (linearization) error). This position error is very difficult to notice since the operator is using teleoperation to move the robot in a relative sense and does not keep track of the ideal robot position). In our system, this error is not handled because we have a high sampling rate and noise-free $\Delta\theta$ data (obtained by differencing two successive encoder values of the hand controller rather than from a velocity approximator); however, one can add an absolute position servo loop at the robot side to account for the error.

The feedback loop originates from the sensor attached at the tip of the robot, where interaction forces and torques are felt, and this sensory information is sent to the local site and reflected onto the hand controller by backdriving the motors that cause the operator's hands to feel the encountered forces and torques. The sensed force/torque information is sent from the remote site and is received at the local site by using the same parallel interface used to pass the ΔX_H information from the local to the remote site. Desired torques to be applied to the hand controller motors are computed as follows. First, the force/torque values from the sensor frame are transformed to the robot tool frame and then premultiplied by a 6 × 6 diagonal matrix to scale and to express reaction forces and torques to be felt by the human. Finally, the forces and torques are premultiplied by the transpose of the hand controller Jacobian express torques that are applied to the motors of the hand controller.

The transformation matrix that converts the force/torque sensor data to resolved Cartesian components expressed with respect to the hand controller handle frame has the following form:

$$^{H}J_{Sensor} = {}^{0}J_{6}^{T} \quad {}^{Reaction\ and\ Scaling}J_{H} \quad {}^{Tool}J_{Sensor}$$

where

$${}^{0}J_{6}^{T} = Hand\ Controller\ Jacobian\ Transpose^{1}$$

$${}^{Reaction\ and\ Scaling}J_{H} = diag(k_1, k_2, k_3, k_4, k_5, k_6)$$

$${}^{Tool}J_{Sensor} = \begin{bmatrix} R & \vdots & 0 \\ \cdots & \cdots & \cdots \\ p \times R & \vdots & R \end{bmatrix}$$

where

$$p\times = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix}$$

Handling Singularity. When a robot is in a singular position, there are multiple kinematic solutions. For example, in the case of a PUMA 560 robot, when joints 4 and 6 axis become aligned, a degree of freedom is lost and in the kinematic sense, only the sum of joint 4 and joint 6 is then important. Therefore, in this situation, the user would have to specify either the joint 4 or 6 value to force the inverse kinematic solution to be unique. When a typical teleoperation scenario is considered, if one observes that the robot is moving toward its singular position, then this indicates two possible actions by the operator. One option is that the operator wants to change the robot pose, and the other is that he has made a mistake by moving the robot near the singular position. This ambiguity can easily be resolved by querying the operator. In our system, the operator specifies his intention to change pose by pressing the middle button when the robot is near a particular singularity. If the operator does not press the pose change button near the singular position,

[1] Refer to the Appendix for the hand controller (JPL's FRHC) kinematic model.

then the robot is forced to remain in the existing pose (i.e., the robot is prevented from ever going into the singularity - a "bouncing off" effect). Pose change can only occur near the singular regions, rather than anywhere in the robot workspace, to avoid large swinging motions.

2. System Descriptions

Hardware. The hardware is divided between two sites: the local and the remote. Refer to Figure 2 for the hardware configuration. The operator located at the local site moves both the right and the left hand controllers with his hands and observes the corresponding robots located at the remote site moving according to his hand motions. At the local site, the operator moves two six DOF universal Force-Reflecting Hand Controllers (FRHC) [4]. The FRHC that is integrated into the system is the third generation hand controller (version C) built in-house at JPL; it is capable of generating 8 lbs of force and 14 in-lbs of torque in each DOF. The design is such that a six-axis mechanism with a steel-cable/pulley drive system is used to virtually eliminate backlash; various miniature ball bearings and large diameter pulleys were used to reduce mechanism friction. Each axis is driven by DC torque motors with digital incremental encoders for feedback information such as position and velocity. The point of contact for the operator on the FRHC is the handgrip. It is used to specify orientation information while the task is proceeding, and the operator can use three momentary buttons to change system operating modes. The top-left button is called the index button and is used to activate the robot — the operator usually turns the index button off when he interacts with the system menu. The top-right button is used to turn on/off force reflection while a task is proceeding. The middle-bottom button is to confirm a robot pose change. Finally, the trigger is used to open/close the robot gripper.

The hardware unit that interfaces with either the FRHC or the PUMA robot is called the Universal Motor Controller (UMC) [5]. It was built in-house at JPL and can be easily reconfigured to interface with either a hand controller or a robot. It contains joint interface cards that provide encoder and potentiometer information and output desired PWM signals, and uses two National Semiconductor 32016 CPU boards to execute servo and communication software written in the NSC32016 assembly language. Communication between the processor boards and the joint interface cards is made through the Multibus. A special set of protocols to communicate with the UMC from the VME side was developed and tested; the rate of communication is approximately 2 KHz, which is twice the rate at which internal PD motor servoing is performed. A parallel port on one of the CPU boards handles communication and is used to interface with the VME side. The UMC-VME interface allows on-line capability of specifying desired encoder setpoints (position commands) or torque commands (analogous to the PUMA Unimation controller's current commands) in PWM units, reading actual encoder positions and other analog signals (in the case of a hand controller, reading the trigger potentiometer value), setting control loop gains (for position and velocity), controlling the robot gripper, and calibrating the robot or the hand controller.

The VME chassis environment (either L-TELEOP or R-TELEOP) at the local site consists of the following: four MC68020/68881 processor boards (Heurikon HK68/V2F) each running at 20 MHz clock rate, to perform robot/hand controller kinematic computations, communication, graphics, and network interfacing with the user (in the future, to reduce kinematic computation time, a MC68030/68882 Heurikon HK68/V30 card will be used); an Ethernet card (Excelan EXOS 202) to connect to the local network; a system controller card (Motorola MVME025) that handles bus arbitration; a graphics generator card (Parallax 600 VME) for force/torque displays; and two parallel communication cards (Xycom XVME-240) for interfacing with the UMC and the remote site. The graphics card generates real-time force, torque, and grasp force information for operator display.

In the remote site, the hardware includes two Unimation PUMA 560 robot arms. Each arm is equipped with a commercial wrist force torque sensor from the Lord Corporation and a TRI servo gripper. The arms are driven by two UMC's. The VME chassis environment at the remote site consists of the following: five MC68020/68881 cards — two of them perform kinematics for the right and left robots, two perform communication with the right and left robots, and the last one performs network interfacing; a bus arbiter; four parallel communication cards — two interface with the robots and the other two with the Lord

force/torque sensor processor units; an Ethernet card; and a shared memory bus adapter card from the BIT3 Corporation to obtain shared control commands from the autonomous portion of the testbed (SUN 4 running the RCCL robot language under dual-arm configuration).

One of the convenient features of the JPL architecture is the homogenous hardware environment. The VMEbus/68000 architecture was chosen in part to be compatible with the SUN computer backplane environment, which uses the VMEbus architecture — the code for the autonomous portion executes on the SUN4 and an interface exists between the SUN4 and the robot VME chassis by utilizing a VME-VME bus adapter. The choice of the VME architecture seems to be popular, since many research centers have now adopted the architecture for their robotics research. At the motor control level, the local and remote UMC's and VME environments have identical hardware and software setups (i.e., processor and interface cards have the same hardware configurations, and the same code that can handle either the local or remote site is downloaded and executed). Having a homogeneous environment has a number of advantages, namely that the system can be reconfigured easily for many different types of research, and the same resources such as robots and controllers can be shared. Elements of redundancy in the hardware add to fault protection and reliability of the system.

Software. All teleoperation software is written in the C language with the exception of the NSC32016 assembly language code that runs on the UMC. Code is developed on a SUN 3/60 (or SUN 4 with a cross compiler) UNIX computer utilizing SUN's C compiler and Wind River's VxWorks/Wind real-time library and is downloaded through Ethernet to the processor boards for immediate execution. Many convenient features of the VxWorks library such as task control, networking, and debugging support save a great deal of development time.

3. Pipeline Architecture Implementation and Timing Data

In this section, a four-stage pipeline design is described. From the hardware point of view, pipeline architecture can be considered modular since functionalities are divided among the processor boards. Referring to the pipeline diagram of Figure 3, each processor has a unique assigned function (e.g., two of the processors COM_H and COM_R are dedicated solely to handling communication, and the other two KIN_H and KIN_R perform kinematic computations). Considering modularity, for a KIN board, the computation time required to perform the assigned function can be reduced by replacing the board with a faster processor board, and as a result, since the most time-consuming (KIN) stage has been speeded up (in pipeline design, the most time-consuming stage determines the pipeline clock period), the pipeline is executed faster. Another consideration is that due to the modular design, available processing power is optimally utilized. Processor assignment can be made according to the required computational power for each stage. For example, a COM stage that requires little number-crunching capability can be assigned to a processor board that holds minimal computation power (enough to handle handshaking with its communicating partner), while a KIN stage should be assigned to a fast processor board that is equipped with an auxiliary floating point processor running at optimal clock rate.

Pipeline design does not increase the closed-loop control loop delay (computation time) but does increase the throughput of the system (i.e., system sampling rate is increased due to the increased rate at which input data is gathered and output data is generated). It is not certain however that pipeline design results in added system stability or performance. Conventionally the sampling and computation times are set equal. In this case, all effects occurring between sampling instances will not be detected. But these effects between each successive sampling will be noted and compensated for if multiple sampling was made during each computation period. It is important to consider the transient effects, especially if an anomaly condition occurs — the faster the system senses the anomaly, the faster the system will respond to it. In this sense, it is intuitive that the higher the sampling rate, the more responsive the system will be in providing more effective control actions.

Various studies have been made concerning time-delay in a force-reflection teleoperation system [6, 7] which is related to lengthening the closed-loop control loop delay in the system. Solutions such as providing

local compliant force control at the remote site and robot-positional-error feedback to reduce instability caused by time-delay have been presented. However, this type of analysis does not address the advantage of having a pipeline design. Pipeline design touches on the issue of multirate sampling theories. Various digital control texts describe sampling concepts. Shannon's sampling theorem [10] states that the original sampled signal can be reconstructed by having a sampling rate that is at least twice the bandwidth of the cutoff frequency of the system. Due to the effects of noise, data quantization, and system resonant frequencies, in their discussion of Shannon's sampling theorem, Houpis and Lamont [10], recommend that the sampling rate be at least eight times greater than the bandwidth of the reference input. Craig [9] considers noise and resonant effects and recommends that the sampling rate be 10 times faster than the correlation time of noise or of the structural resonant frequency of the manipulator mechanism. Avoiding structural resonance is also discussed in Paul [11]. He recommends that the sampling rate should be 15 times the link structural frequency. All these arguments favor having a fast sampling time for the manipulator. In the present design, a four-stage pipeline architecture will be implemented to have the sampling rate be approximately 6 times faster than the closed control loop delay time in force-reflection mode. No conclusive evidence was found by the author as to the advantage of having a faster sampling but still retaining the same closed-loop delay time. Using our configuration, this issue will be studied further, and in connection the effects of multirate sampling in robotic systems will be investigated.

The details of the pipeline design will now be presented. Figure 3 shows a timing diagram and lists the actual timing data that were obtained after implementation. To perform teleoperation computations, four processors are coordinated in a pipeline arrangement; each stage of the pipeline is handled by one processor. Each processor at every 1.6-ms period performs its designated computations (see description boxes underneath the timing diagram). For example, COM_H stage starts by communicating with the robot side to exchange ΔX and force information for 0.3 ms, multiplies the transpose of the hand controller Jacobian to force values (in 0.1 ms), communicates with the UMC to send desired torques and at the same time to receive the present encoder positions, and finally stays idle for 0.6 ms until the next 1.6-ms period begins. The diagram contains a shaded path that traces the closed-loop force control flow. The control loop begins by having the COM_H stage receive the UMC encoder values — this requires 0.6-ms communication time. KIN_H stage then takes the converted robot angular values and calculates the hand controller Jacobian to compute ΔX values. Stages COM_H and COM_R synchronously get invoked to pass the ΔX information to the robot site. Transformation is made to express the ΔX information with respect to the robot base frame, and then the inverse Jacobian of the robot is multiplied to compute the desired robot positional setpoints. After waiting for the force sensor to respond to the robot servoing to the desired setpoints (which is approximately 1.6 ms — usually the Lord force/torque sensor processing unit sends resolved Cartesian force/torque information at every 10 ms, but we are using the raw strain gauge mode with increased clock speed to obtain data much faster), 0.3 ms is used to obtain the raw strain gauge values and multiply these readings with a sensor calibration matrix to compute corresponding force/torque values. The force/torque values are then forwarded to the hand controller side, and finally forces are converted to desired torques and sent to the hand controller UMC for torque servoing. The closed-loop sampling is approximately 104 Hz. In implementation, the multiple processors are synchronized and data is passed through shared memory. In a pipeline situation, the stage that has the worst time delay dictates the pipeline clock rate. In our design, the most time-consuming stages are the kinematics stages KIN_H and KIN_R, which take around 1.6 ms. In the future, each of these stages will be replaced by a faster processor (MC68030/68882) board which has a faster clock speed and floating point capability and which will increase the pipeline clock rate. Note that it is more difficult to improve timing for the communication stages; since the processors do little number-crunching but are used to synchronize the communication protocol, upgrading these processors will not improve the timing.

4. Operator Interface

A user interface has been developed using the TCP/IP communication protocol on Ethernet, which allows the operator to execute the teleoperation software and specify system options from any remotely located computer that supports the protocol. For the operator, a graphics menu is displayed on a SUN 3/60 terminal, through which he can interact with the system. The menu consists of a number of windows whose

implementation is based on the SUNVIEW facilities [8]. It is a user-friendly environment where choosing an option can be done simply by moving the SUN mouse and then clicking on a button. It is capable of displaying graphics information in different formats (e.g. bars and scales to display robot information and icons to represent various parts of the system). The operator can use the menu to specify options such as system control modes (position with or without force feedback, and rate, joint, and shared control modes) and reference and view frames. He can monitor the state of the robot by observing the robot angles on the menu display and monitor force/torque data displays on another monitor — observing the robot angles is useful in detecting joint limits. For each robot, a menu window is dedicated for displaying the data about that particular robot. See Figure 4. The robot data is forwarded to the SUN for menu display through a specially designed protocol based on UNIX socket facilities. This data is not forwarded at every sampling instance of the system, but once every tenth or more sampling time, which is sufficient to display varying real-time data. In addition, Cartesian position is forwarded as well as the present robot configuration. Since the data is available on the SUN computer, robot and force/torque data can easily be logged for later analysis.

5. Conclusions

In this paper, a force-feedback teleoperation system based on a pipeline architecture was described. It will be integrated with the autonomous portion of the JPL testbed to support shared control research. Once the system is operational, issues such as multirate sampling effects will be investigated. Future work will include extensive experimentation with the system to examine control, human factors, and time-delay issues.

6. Acknowledgements

The research described in this document was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The author would like to thank Samad Hayati for many useful discussions and Ted Lewis and Zoltan Szakaly for their software and hardware support.

References

[1] Hayati, S., Venkataraman, S.T., "Design and Implementation of a Robot Control System with Traded and Shared Control Capability," submitted for publication to the *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*

[2] Bejczy, A.K., and Hannaford, B., "Man-Machine Interaction in Space Telerobotics," *Proceedings Intl. Symposium on Teleoperation and Control*, Bristol, England, July 1988.

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

[4] McAffee, D., Ohm, T., "Teleoperator Subsystem/Telerobot Demonstrator: Force Reflecting Hand Controller Equipment Manual," JPL D-5172 (internal document), Jet Propulsion Laboratory, Pasadena, California, January 1988.

[5] Bejczy, A.K., and Z. Szakaly, "Universal Computer Control System for Space Telerobotics," *Proc. of the IEEE Conference on Robotics and Automation*, Vol. 1, pp. 318-324, Raleigh, N.C., 1987.

[6] Anderson, R.J., and Spong, M.W., "Bilateral Control of Teleoperators with Time Delay," *Proc. IEEE Int'l. Conf. Systems, Man, and Cybernetics*, Vol. 1, p. 131, Beijing, China, August 1988.

[7] Hannaford, B., "Stability and Performance Tradeoffs in Bilateral Teleoperation," submitted to the *Proceedings of the IEEE Intl. Conf. on Robotics and Automation*, Scottsdale, AZ, May 1989.

[8] Sun Microsystems, Inc., SunView System Programmer's Guide, September 1986.

[9] Craig, J., *Introduction to Robotics*, Addison-Wesley Publishing, 1986, pp. 239-242.

[10] Houpis, C. H., Lamont, G. B., *Digital Control Systems*, McGraw-Hill, 1985.

[11] Paul, R., *Robot Manipulators: Mathematics, Programming, and Control*, The MIT Press, 1981.

7. Appendix: JPL Force-Reflecting Hand Controller Kinematic Model

The Denavit-Hartenberg parameters for the FRHC are given below:

θ_i	θ_1	θ_2	90°	θ_4	θ_5	θ_6
d_i	0	0	$d_3 + d_3\_offset$	0	0	0
a_i	0	0	0	0	0	0
α_i	0	90°	-90°	0	-90°	90°
Initial Settings	0	-90°	$d_3 = 0°$	90°	-90°	0°

$d_3\_offset = 730.25mm$

$$ {}^0A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^1A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_3 + d_3\_offset \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^3A_4 = \begin{bmatrix} c_4 & -s_4 & -0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^4A_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^5A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Jacobian for the JPL Force Reflecting Hand Controller:

$$ {}^6J_6 = \begin{bmatrix} j_{11} & j_{12} & j_{13} & 0 & 0 & 0 \\ j_{21} & j_{22} & j_{23} & 0 & 0 & 0 \\ j_{31} & j_{32} & j_{33} & 0 & 0 & 0 \\ j_{41} & j_{42} & 0 & j_{44} & j_{45} & 0 \\ j_{51} & j_{52} & 0 & j_{54} & j_{55} & 0 \\ j_{61} & j_{62} & 0 & j_{64} & 0 & 0 \end{bmatrix} $$

$$ \begin{aligned} r_{3o} &= d_3 + d_3\_offset \\ t_1 &= c_2c_4s_5 \\ j_{11} &= r_{3o}(t_1c_6 - c_2s_4s_6) \\ j_{21} &= -r_{3o}(t_1s_6 + c_2s_4c_6) \\ j_{31} &= -r_{3o}(c_2c_4c_5) \\ t_2 &= c_2s_4s_5 + s_2c_5 \\ j_{41} &= c_2c_4s_6 + c_6t_2 \\ j_{51} &= c_2c_4c_6 - s_6t_2 \\ j_{61} &= s_2s_5 - c_2s_4c_5 \\ j_{12} &= r_{3o}(c_4s_6 + s_4s_5c_6) \\ j_{22} &= r_{3o}(c_4c_6 - s_4s_5s_6) \\ j_{32} &= -r_{3o}s_4c_5 \\ j_{42} &= s_4s_6 - c_4s_5c_6 \end{aligned} $$

$$
\begin{aligned}
j_{52} &= s_4c_6 + c_4s_5s_6 \\
j_{62} &= c_4c_5 \\
j_{44} &= j_{13} = c_5c_6 \\
j_{54} &= j_{23} = -c_5s_6 \\
j_{64} &= j_{33} = s_5 \\
j_{45} &= s_6 \\
j_{55} &= c_6
\end{aligned}
$$

FIGURES

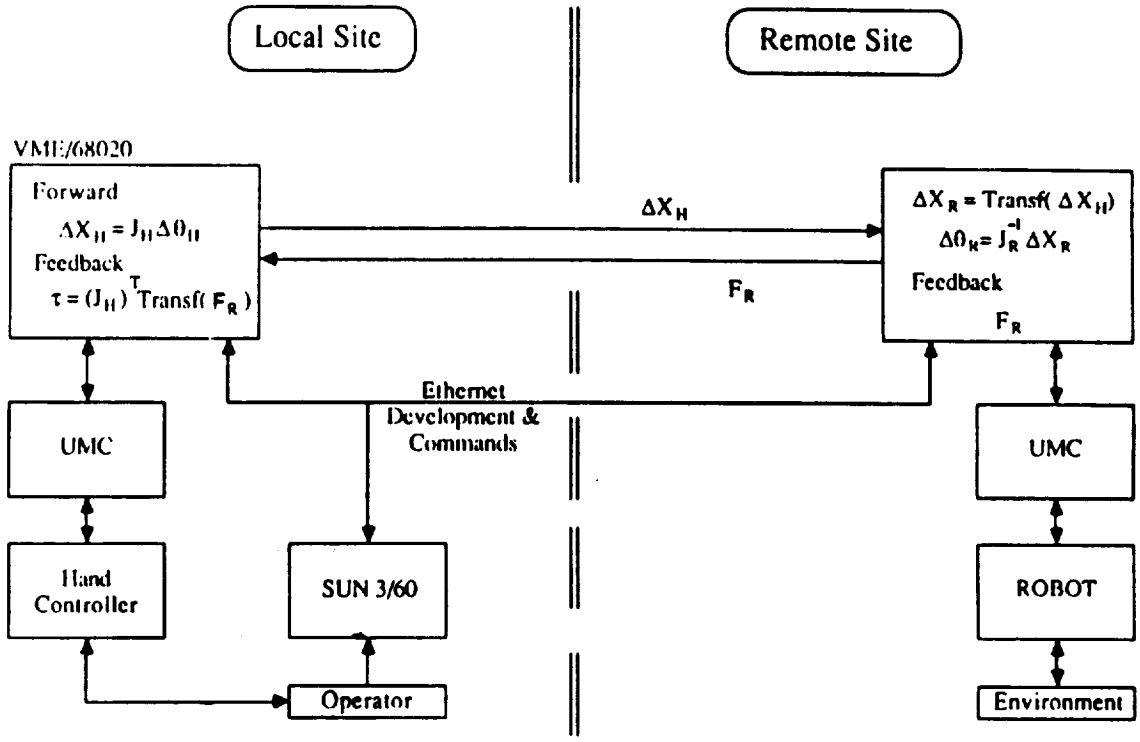

Figure 1: Functional Diagram of the Teleoperation System

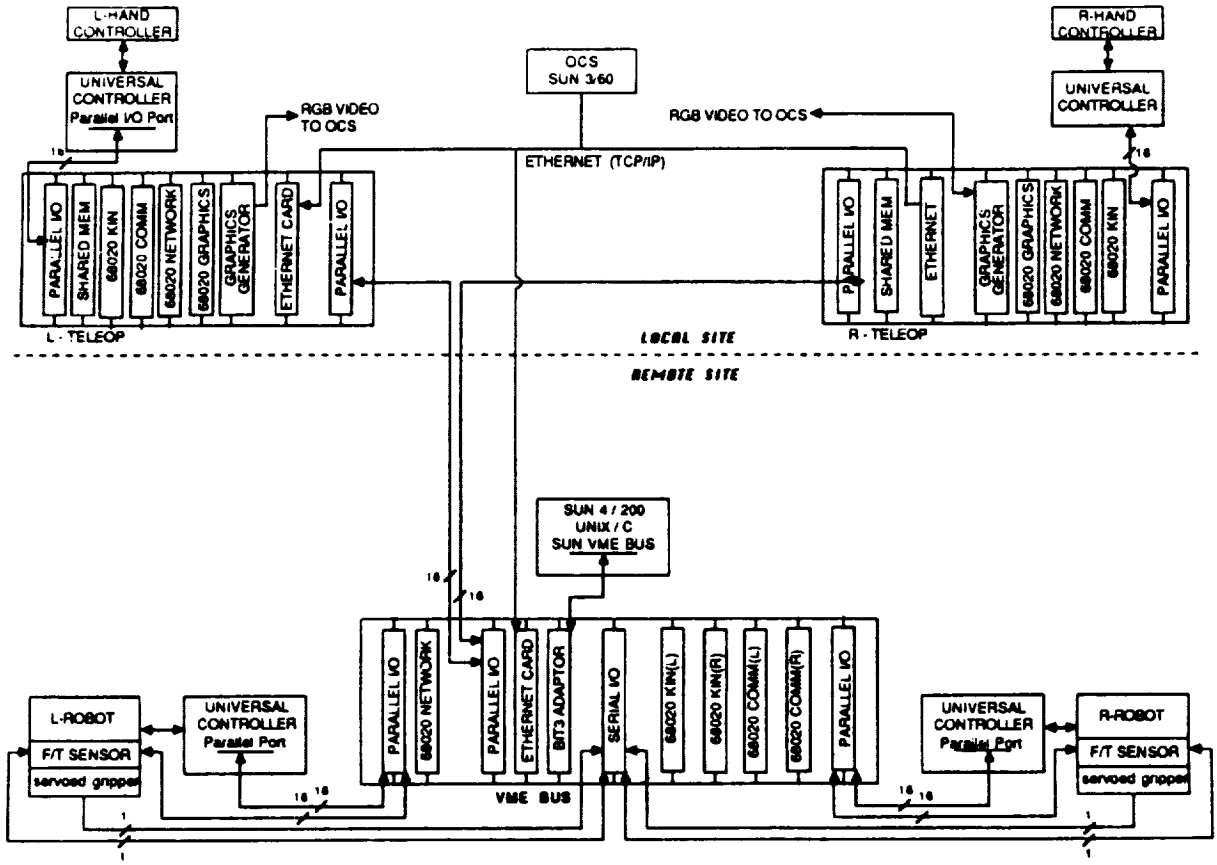

Figure 2: Hardware Diagram of the Teleoperation System

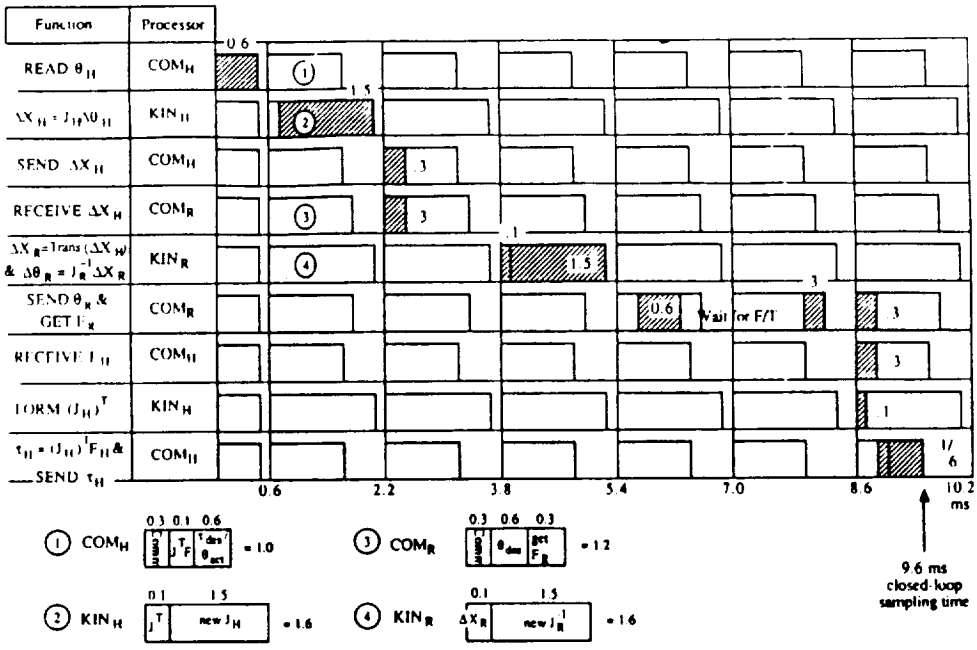

Figure 3: Timing Diagram

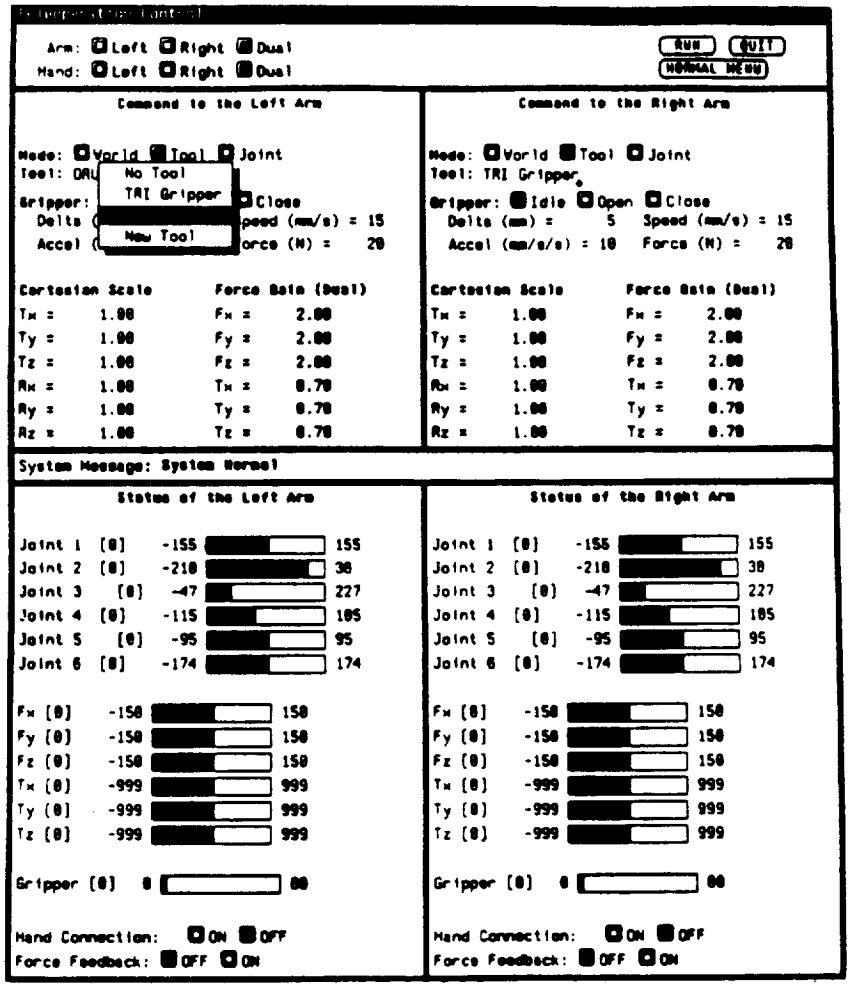

Figure 4: Graphics User Menu