Thurston L. Brooks
Thomas B. Sheridan

Superman: A System for Supervisory Manipulation and the Study of Human/Computer Interactions

MIT Sea Grant
College Program

Massachusetts
Institute of Technology
Cambridge
Massachusetts 02139

MITSG 79-20
July 1979

Superman: A System for Supervisory Manipulation and the Study of Human/Computer Interactions

by

Thurston L. Brooks III
Thomas B. Sheridan

Sea Grant College Program
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Report No. MITSG 79-20
Index No. 79-320-Mio

AUTHORS

Thurston L. Brooks wrote this report while a Master of Science degree candidate at MIT.

Thomas B. Sheridan, Professor of Mechanical Engineering, who is the principal investigator for the Sea Grant Project, "Application of Teleoperators to Undersea Tasks," served as Mr. Brooks's thesis supervisor.

MIT SEA GRANT PROGRAM REPORT SERIES
RELATED REPORTS

MITSG 76-7	MIT/Marine Industry Collegium. Telemanipulators for Underwater Tasks: Opportunity Brief #3. $2.50

ABSTRACT

This report considers the need for supervisory control of remote teleoperator vehicles in the ocean environment, and shows that computer controlled systems can increase the effectiveness of remote manipulation.

A distinction is made between absolute tasks -- tasks which have a known spatial relationship to the manipulator base prior to execution -- and relative tasks -- tasks which cannot be spatially defined prior to execution. A second distinction is made between fixed tasks -- tasks which remain fixed with respect to the manipulator base during execution -- and moving tasks -- tasks which continuously move with respect to the manipulator base during execution. Four distinct combinations can be made from this 2 x 2 array: (1) fixed-absolute tasks, (2) fixed-relative tasks, (3) moving-absolute tasks, and (4) moving-relative tasks. Mathematical principles are developed to deal with each of these four possibilities.

A unified theoretical framework of supervisory manipulation is considered to give the designer an overview of: (1) manipulator and processor selection factors, (2) interface design considerations, (3) control language attributes and implementation factors, and (4) control philosophies.

Based on the mathematical and theoretical foundations described in this Sea Grant Report, a supervisory system was developed and demonstrated.

The major conclusion derived from this study is that even under the "best" control conditions, i.e., no time delays, no frame rate problems, high visibility, etc., supervisory control can improve system performance for all forms of manual control except master/slave with force feedback.

ACKNOWLEDGEMENTS

My deepest appreciation must go to my wife, Mudah, for sharing my troubles, giving me comfort and support, and encouraging me forward when there appeared to be no end to this thesis. She also deserves special thanks for invaluable assistance with the experiments and the preparation of this document.

I also wish to express my gratitude to my advisor, Professor Thomas B. Sheridan, for his guidance and advice. He also deserves thanks for his constructive suggestions in the revision of the thesis draft.

I would also like to thank Paul Heckman of the Naval Ocean Systems Center, Cliff Winget of the Woods Hole Oceanographic Institute, Dan Whitney of Draper Labs., and Al Woodin of Draper Labs., for their invaluable aid and feedback.

Further, I would like to thank Leslie Regan whose uncanny typing speed and accuracy is more than deserving of acknowledgement.

Finally, but not least, I would like to thank the government agencies whose support made this research possible. Development of the hardware and software used to demonstrate the basic concepts of supervisory control was supported by the MIT Sea Grant Program. Dean Horn (MIT Sea Grant Director), Norman Doelling, Beth Harding, and others from the Sea Grant Office have faithfully championed my efforts and deserve special thanks. The experimental research in man-machine system performance described in Chapter V, and some of the associated theory, was supported by the Office of Naval Research,Engineering Psychology Programs. Gerald Malecki of the Office of Naval Research provided both technical monitoring and advice.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF FIGURES

LIST OF FIGURES (Con't.)

LIST OF FIGURES (Con't.)

LIST OF FIGURES (Con't.)

LIST OF FIGURES (Con't.)

LIST OF FIGURES (Con't.)

LIST OF TABLES

LIST OF TABLES (Con't.)

CHAPTER I

INTRODUCTION TO SUPERVISORY MANIPULATION

"The bomb was tenuously resting on a craggy slope at the brink of an undersea canyon, and the parachute that was still attached to it was drifting back and forth in the current. There were two dangers here for those attempting a recovery: the first was getting entangled in the parachute shrouds and the second was dislodging the bomb and possibly losing it deeper in the sea. When the bomb was first discovered, the Alvin attached a marking pinger, but it (Alvin) became entangled and there were some nervous moments before it worked itself loose. After that the Alvin preferred to stand back, and the remotely manned CURV I made the necessary attachments and raised the lost bomb to the surface".[1] The incident described was referring to the lost H-bomb off the coast of Palomares, Spain in 1966.

The performance of the CURV clearly indicated the potential of remote vehicles. Although unmanned spacecraft had been in use for eight years, none had directly manipulated its environment to any real extent. Unmanned vehicles had come of age almost unnoticed in the publicity of the event. Within a decade the number of remote vehicles in operational use increased from a handful of simple vehicles to more than 50 sophisticated machines which interact with their environment on a number of levels (sonar, low-light video, sonic imagining, manipulators, etc.).[2]

But with the increasing use of remote unmanned vehicles has come an increasing awareness of the limitations of present technology. Technological advances are needed in navigation and guidance, remote viewing and search, signal processing and transmittal, artificial intelligence, and remote manipulation. This thesis will deal primarily with the last category, remote manipulation, although at times the remaining categories will be mentioned.

1.1 Supervisory Control - What Is It?

A supervisor is normally thought of as an individual who directs the actions of subordinates. The supervisor's function is to plan future courses of action for the subordinates, teach them the proper method of carrying out these actions, monitor their performance, correct their actions whenever they do not meet expectations, and trust the subordinates to perform the actions as directed.

Ferrell and Sheridan[3] in 1967 proposed that a hierarchial man-computer system based on the human supervisor-subordinate relationship could be used in deep space to solve some of the control problems involving time delays. Under supervisory control the human operator directs the subordinate computer by planning the actions and directions it should take, teaching it how to achieve the desired functions, monitoring its performance, intervening whenever it gets into trouble, and trusting it to accomplish the task without continuous assistance.[4]

Figure 1-1 shows the essential difference between direct manual control and supervisory control of a teleoperator. Under direct control the operator's control signals are sent directly to the remote manipulator, and sensor information is fed directly back to the operator. Under supervisory control the operator's control signals are relayed through a local computer to the remote computer, which then processes the signals and acts on the information. The relayed signals are not necessarily the raw signals generated by the operator. In fact the signal is usually a coded instruction of high information density which must be interpreted to be utilized. The operator's input could range from a purely manual analogic command to a highly abstract symbolic command (see Chapter III for details of this distinction). The remote computer not only interprets the local computer's messages but also acts on the sensor information available to it about its environment. The remote computer only relays information to the operator which is deemed important and necessary for effective supervision — the responsibility for the specific details of control is usually left to the subordinate computer.

Figure 1-2 shows how supervisory control fits into the global scheme. Supervisory control combines the best attributes of both machine and man to achieve the desired goal — a clear il-

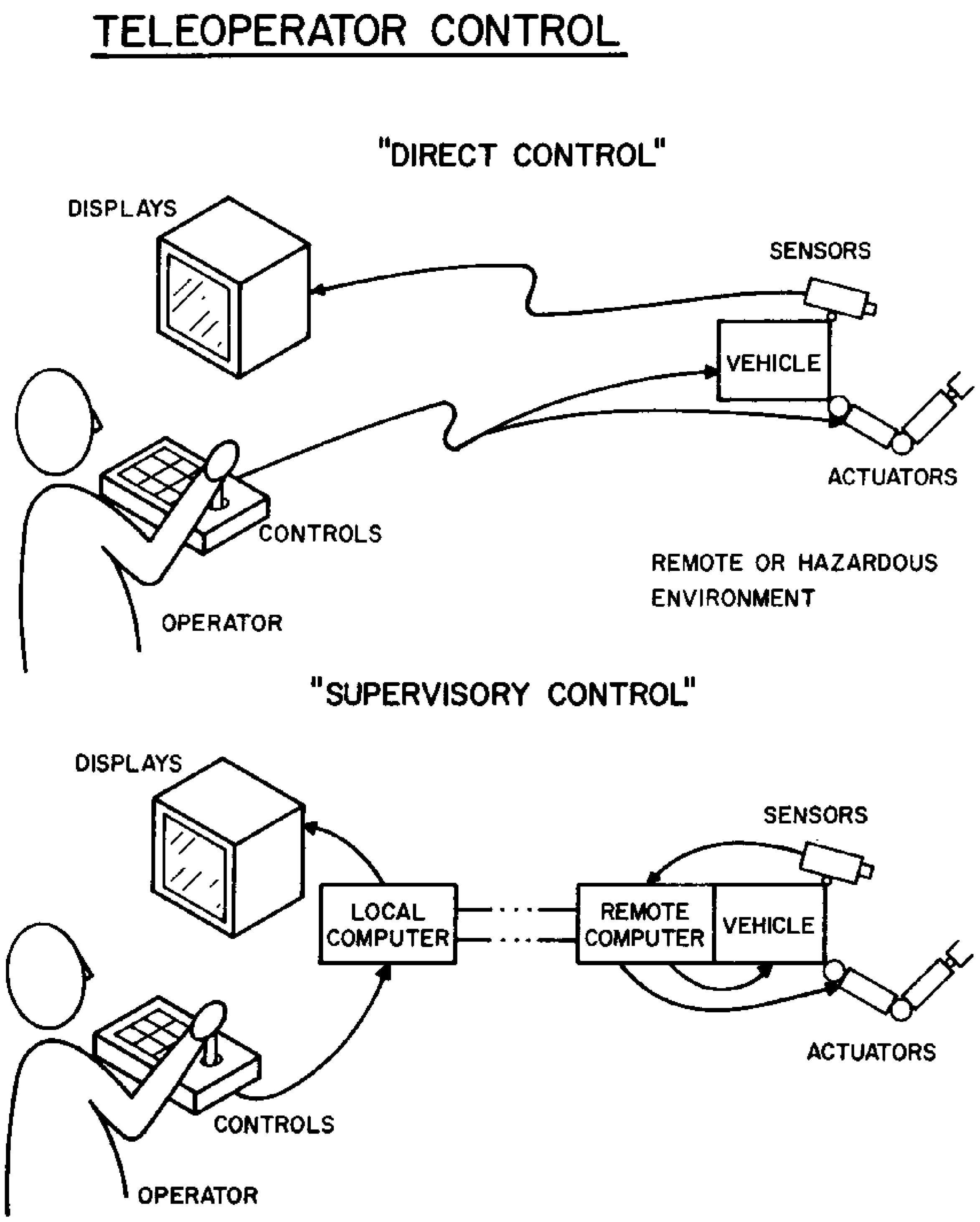


Figure 1-1: Direct Manual Control and Supervisory Control of a Teleoperator (Adapted from Ref. [4])

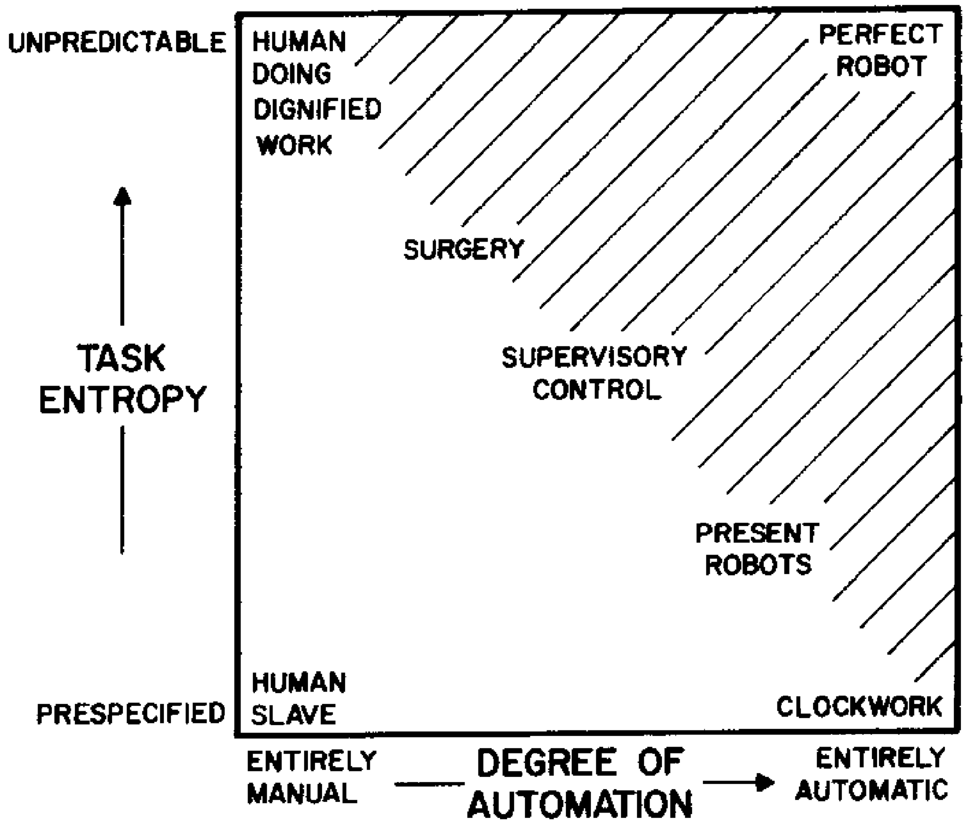


Figure 1-2: Task Predictability Versus Degree of Automation
(Adapted from Ref. [4])

lustration of the axiom "two heads are better than one". Sheridan has said, "human supervisors are used in conjunction with robots because the two are complementary: We automate what we understand and can predict and we hope the human supervisor will take care of what we don't understand and cannot predict".[5]

1.2 Supervisory Control - When Is It Needed Underseas?

The world's oceans cover approximately 70 percent of the globe and of that 70 percent the continental shelves cover less than 8 percent.[6] Since most of the exploration,production and transportation of ocean resources occurs on the continental shelf, this means that 92 percent of the earth's ocean resources remain untapped.

A vehicle which can dive to 6100 m (20,000 ft) will be able to reach 98 percent of the ocean floor. Unfortunately, the pressure at any depth beyond 100 meters requires rather elborate equipment to safely maintain submersible operators or divers for any length of time. Concern for human safety (submersible occupants or divers) requires redundant back-up systems which not only increase the initial capital expenditure, but operational costs as well. Busby[7] gives in-depth accounts of 39 emergency incidents involving 22 submersibles which directly endangered or resulted in loss of life. High mortality rates for commercial divers in the North Sea clearly indicate that human safety is not easily achieved.

Unmanned systems, on the other hand, are not as pressure dependent, and hence, operation costs are less and human safety is not of primary concern, since the operator is on the surface. Figure 1-3 shows the estimated cost in dollars per bottom hour for conventional surface diving, saturation diving, manned submersibles, and remote work vehicles as a function of depth for a welding task. The figure shows that, except for the initial 40 meters, unmanned vehicles have the lowest cost per bottom hour. Unfortunately, the figure does not indicate the productivity of each method; if the remote vehicles require significantly longer times to perform the same task as a diver or manned submersible there will be no economic advantage. In fact Vadus[2] says, "from a cost-effectiveness standpoint, the cross-over point between utilizing a diver with Scuba versus a manned submersible is about 150 meters". The advantage of divers is clearly one of productivity; human divers are on the average 4 times faster than manned or unmanned teleoperator systems.[8] But with the continuing improvements in manipulators and computers, the divers will eventually lose their dexterity advantage. Once the dexterity of manipulators approaches that of divers,the diver's only remaining advantages will be sensing and cognition, both properties which could be done equally as well from the surface.

Man's only function inside a submersible is to provide scene analysis; he can neither feel, hear, nor manipulate his en-

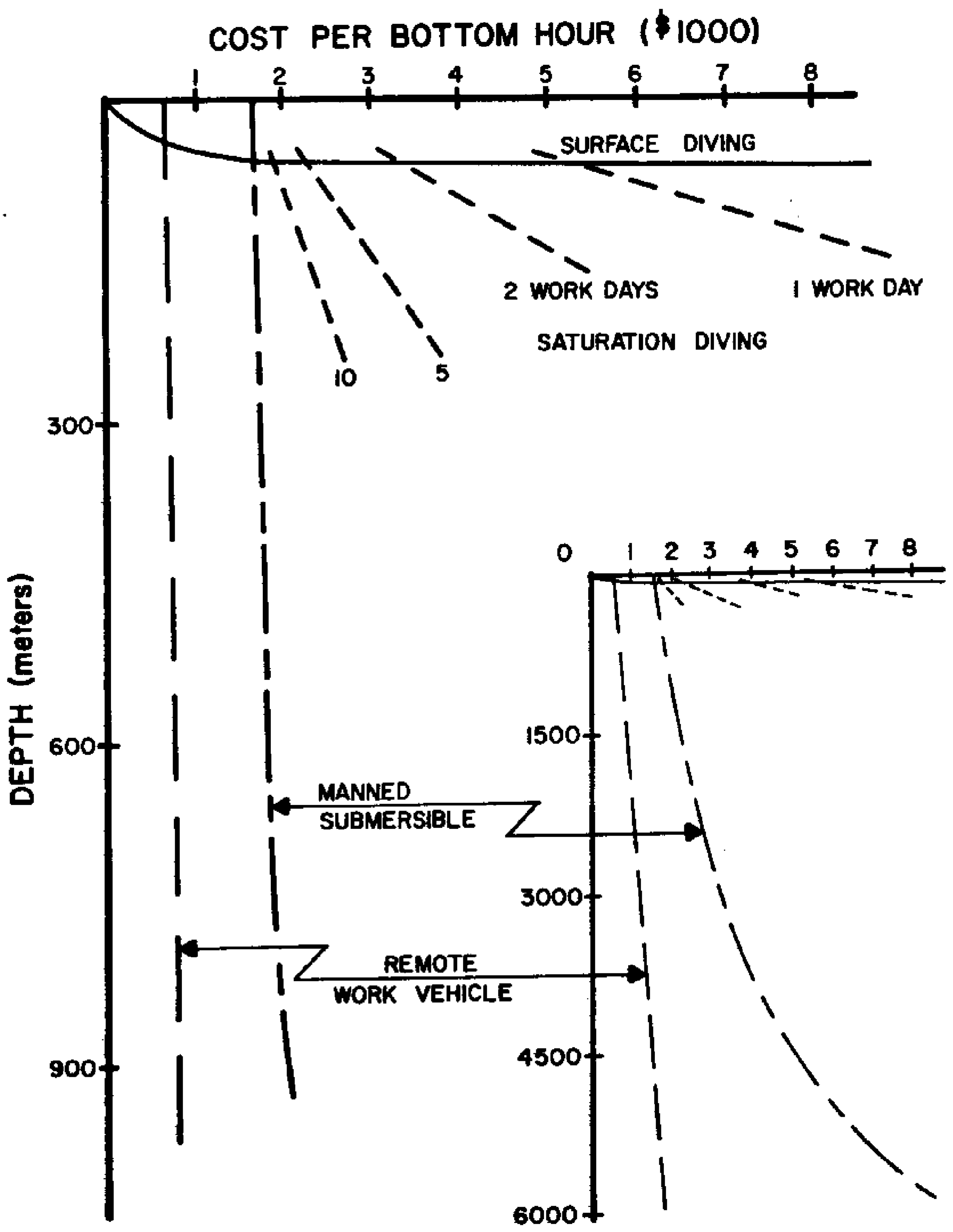


Figure 1-3: Cost Comparisons for Underwater Welding as a Function of Depth and Method (Data taken from Ref. [9] and adjusted for an annual inflation rate of 8 %)

vironment except by artificial transmission of energy and communication through the hull. Until recently the operator's direct visual assessment was irreplaceable, but improved low-light cameras and sonic imaging techniques which can see through turbid water are beginning to encroach on the operator's domain. Man's stereoscopic vision is still unequalled by any other system, but it can be envisioned that within time even this attribute could be sent over the communication link to the surface for operator analysis. If all of man's functions as a diver or submersible operator can be relayed to and from the surface, the need for man to be in direct contact with the hostile ocean environment evaporates. Clearly, the operator should be removed to the safety of the surface vessel since his physical presence will no longer be required.

Once the decision has been made to place the man on the surface it becomes necessary to decide how much "autonomy" the remote vehicle should be endowed with. One of the primary duties of the on-board operator is to make decisions based on the information he has about his environment. If the communication link between the remote vehicle and the surface is instantaneous, the vehicle would need very little automony. Supervisory control would consist of simple routines to aid the operator in a direct manner (e.g., it could obtain tools and return them automatically). But during periods of feedback dropouts (turbid water, etc.), sensor failure, degraded TV, interrupted acoustic imaging, control communication loss, or other failures the remote vehicle should be capable of simple autonomous actions to continue performing the task, stop, or return to the surface.

If the remote vehicle operates at extreme depths or requires complete freedom of movement, the weight and drag of the tether can cause considerable problems. There are three methods which have been suggested to overcome these problems: (1) use of lightweight and small diameter cables, (2) allowing the vehicle to work out of a "garage" which is tethered to the surface, or (3) communicating through sonic links. The cable's weight and diameter can be reduced through the use of fiber optics for communication signals, but the power supply will still require either a bulky cable or the use of limited on-board batteries. The garaging method appears to hold the most promise, since it allows a large, heavy cable to descend from the surface to the garage and a smaller buoyant cable to extend from the garage to the vehicle. The final solution, a sonic link, allows complete freedom of movement, but imposes a severe communication constraint between man and vehicle.

The speed of sound in water is approximately 1600 meters per second. Therefore, an acoustic signal will take a delay time of one second for every 1600 meters of depth. If each "message" is composed of a large number of information bits there can be further restrictions due to the low bandwidth of the sonic channel (e.g., 30k bits per second for the acoustic channel compared to 300k bits per second for a coaxial channel and 3M bits per second for a fiber optic channel[4]). For example, a video picture composed of 128 x 128 pixels with 4 levels of gray scale would take 2 seconds for the entire "message" to be received and assembled on the surface.

This form of limited bandwidth communication is called "frame rate". Clearly, the frame rate is independent of the delay time, and hence, both effects will be observed over an acoustic channel (i.e., at 1600 meters an acoustic channel would take one second for the first bit of information to reach the surface and then another 2 seconds for the entire picture to assemble). Since manual manipulation requires continuous visual feedback to close the loop around the end effector, it would be expected that time delay and frame rate constraints would severely limit the operator. In fact, experiments conducted by Ferrell, Black, Starr, Hill, and others for time delays in space manipulation indicate that task completion time can be increased by a factor of four or more under a time delay.[4] To date, experiments to determine the effects of frame rate on task completion time have not been conducted.

Considering the problems involved with an acoustic communication channel the trade from high bandwidth optical cables does not appear to be warranted. But, if the remote vehicle were controlled under supervisory control, which does not require large amounts of control communication (remember, supervisory control signals are usually coded instructions of high information density —

Section 1.1), the vehicle freedom and performance would offset the loss of direct control.

To summarize, as undersea technology advances there will be an increasing use of unmanned, tethered and untethered vehicles. These remote vehicles can benefit through the use of supervisory control techniques which range from simple direct operator aids (such as automatic tool retrieval and return routines) to more sophisticated supervisory techniques (such as systems that can perform autonomously for brief periods when commanded by high information density directives).

1.3 Supervisory Control - How Can It Be Accomplished?

Now that supervisory control has been defined and the circumstances under which it might be useful have been examined, it will be necessary to determine the methods by which a supervisory system can be built. The remainder of this thesis will be devoted to both the theoretical and practical aspects of developing supervisory manipulation.

In Chapter II mathematical foundations will be developed for performing four major categories of tasks: (1) tasks which have a consistent relationship to the manipulator base at all times. (2) tasks which cannot be spatially defined prior to execution (3) tasks which remain fixed with respect to the vehicle during execution, and (4) tasks which are continuously moving with respect to the vehicle during execution.

Chapter III is devoted to the theoretical aspects of supervisory manipulation from manipulator and processor selection, to the design of the interfaces, language, and control philosophy.

In Chapter IV an explanation of the features of SUPERMAN, a system for supervisory manipulation, is given with visual aids and programming examples.

Chapter V is an evaluation of the SUPERMAN system under supervisory control as a function of viewing conditions (mono and two-view) and manual control modes (switch rate, joystick rate, master/slave without force feedback and master/slave with force feedback). Comparisons between purely manual control and combined manual-computer control are used as a basis for determining the applicability of supervisory control to representative marine tasks.

Conclusions and recommendations for further research appear in Chapter VI.

CHAPTER II

ABSOLUTE, RELATIVE, FIXED AND MOVING MANIPULATION

Through observation of natural human manipulation, one can develop an understanding upon which the design of a supervisory system can be based. For example, when you pull a pen out of your shirt pocket the required joint movements are the same each time regardless of where you are standing. On the other hand, pulling a pen out of your friend's pocket can require different joint actions depending on your friend's spatial location relative to yourself. Hence, through observation of a common human experience, it can be said that one quality of manipulation is represented by the concepts of absolutely defined versus relatively defined joint actions. Another attribute of the human system is the capability to perform a task which is either fixed or moving with respect to the manipulator base. As an example, consider putting a coin in a soda machine versus putting a coin in a turnstile while simultaneously moving through the passage. It is the purpose of this chapter to demonstrate how methods can be developed to give machines the human qualities of absolute, relative, fixed, and moving manipulation.

2.1 Absolute Versus Relative Tasks

Before proceeding, it is necessary to define some of the terminology which will be used:

Frame or Coordinate System - the terms "frame" and "coordinate system" will be used interchangeably to signify a set of three orthogonal unit vectors which define a three dimensional space (e.g., in cartesian space the set of unit vectors $\bar{i}$, $\bar{j}$, and $\bar{k}$).

Joint Angle - the angle or translation which results from a rotational or prismatic actuator between two congruent manipulator links.

Position - the complete spatial location of a coordinate frame by a set of independent variables which are defined by another frame (e.g., in cartesian space the "position" will mean both the displacement of the frame's origin, as well as the frame's orientation, with respect to another coordinate system).

Base or Vehicle Frame - any coordinate frame which is rigidly attached to the most distal manipulator joint from the hand (e.g., the vehicle and anything connected to it would be the base for underwater manipulation).

World or Task Frame - any coordinate frame which is not rigidly attached to the manipulator base (e.g., a valve on a wellhead would have a world or task coordinate frame).

Given the above definitions it is now possible to explicitly define an absolute manipulation task:

> Absolute Task - a task in which the geometric relationship between the task and base frames is known and the joint angles necessary to obtain the required spatial hand positions are always the same.

The only human task which is truly absolute is the act of touching your shoulder, all other tasks are initially relative to some coordinate system — either the world's (e.g., turning on a lamp) or the human operator's body (e.g., pulling out a wallet). But many quasi-absolute tasks can be found for both human and machine manipulation. For example, a quasi-absolute human task would be the functions required to drive a car once the driver has been seated. The gas, brake, clutch, lights and other control inputs are always defined in the same position and require the same joint actions with respect to the driver's seat. Almost every function required of the driver in the cab is absolute, even though the world coordinate system is changing relative to the base (seat) coordinate system.

> Relative Task - a task in which the spatial hand positions always remain fixed with respect to each other, but the joint angles to obtain these positions are a function of the geometric relationship between task and base frames, therefore, requiring the determination of this relationship prior to execution of the task.

Consider the task of removing a nut from a rigid stud located somewhere in space. This task is cognitively defined in a relative mode before the human operator is shown the nut (the human operator knows he must turn the nut in a counter-clockwise manner, pulling back slightly as he turns to determine if the nut is free). But, unlike the quasi-absolute task of turning on the headlights in his car, the human operator does not know, a priori, the location of the nut with respect to his body, and hence, the joint commands required to perform the task.

One of the more obvious and important differences between absolute and relative manipulation is the amount of feedback necessary to perform each. While absolute tasks can be performed without feedback,* relative tasks require input to determine the relation between the base and task frames. For example, before entering your living room at night you know ahead of time that your light switch is always located at the same place on the wall (i.e., a quasi-absolute task assuming you are standing in the doorway). But, you also know that there is a lamp on your endtable, which could have been moved to a new position or orientation during the day (i.e., a relative task). The switch on the wall is fixed and can be located without visual and tactile feedback, and is "absolutely" defined with respect to the doorway before any feedback information has been received (note that usually tactile feedback is used to a small degree to account for errors in memory and proprioception). To turn on the endtable lamp, though, you do not

*Absolute tasks, as well as relative tasks, require proprioceptive feedback to insure the joints are moving as desired (See Chapter III).

know the absolute position of the switch, but only the "relative" action required once the lamp has been located (i.e., once the lamp has been found and the hand oriented to the switch, the action required is a turning or pushing motion). Without feedback most people turn the lights on by a switch which is always fixed with respect to the room rather than blindly search in a dark room for the lamp.

Clearly, a relative task cannot be performed without some form of feedback (visual or tactile in the case of the human operator) to determine the relationship between the two coordinate systems (task versus human operator). An absolute task, on the other hand, declares a priori that the human operator base coordinate system is related to the task coordinate system at the time the task was defined, and hence, requires no other feedback except joint positions (proprioceptive feedback). The majority of people operate in a relative mode whenever feedback is available and the task is not completely and absolutely defined with respect to their internal base system. But note that whenever the visual machinery can be freed from the labor of defining the human operator's base frame with respect to the task coordinate system, most people instinctively resort to quasi-absolute manipulation which does not require the higher level processing needed for relative tasks. For example, tasks such as pulling out a wallet, typing, reaching for a cup of coffee recently set down, automobile control functions, etc. are generally done in a quasi-absolute manner rather than overload the human processing system with unnecessary information.

Most or all industrial tasks are presently defined in an absolute mode (i.e., the part will always be located in the same place at a specified time and will require specific joint actions). But experience has shown that the real, relative and changing world will not be as ideal as the absolutely defined industrial climate. Hence, the failure of many combined computer and manipulator systems in the "real world" where the actions are specified, but the spatial locations and orientations to perform these actions are not known until execution time. To implement supervisory control of a remote manipulator, it becomes obvious that a number of the tasks that the manipulator will be required to perform can be defined prior to execution time, but that the positions at which these tasks are to be accomplished are variables of the environment, and therefore, are usually not explicitly known until the job is located. This drawback has been recognized and much research is being devoted to the development of visual imagery systems to inform the computer of the location of the relative task. But image processing systems are extremely complicated, unreliable, and expensive devices which probably will not be useful for determining the relationship between an arbitrary environment and the manipulator base for many years.

A simpler method which does not rely on digital image processing is needed to fill the gap. Such a method is available with today's technology. The remainder of this chapter will explain the theoretical and mathematical principles necessary to accomplish relative manipulation both with a fixed and moving vehicle.

2.2 The General Transformation Matrix

The transformation between two coordinate systems, p and q (see Figure 2-1b) can be obtained through a combination of translational and rotational transformation matrices. The composite transformation matrix consists of a 3x3 matrix of the direction cosines of coordinate system q with respect to coordinate system p, a column vector which gives the translation of the origin of frame q with respect to frame p, a three element row vector of zeroes, and a translational multiplier, 1. The direction cosines of any vector, $\bar{U}$, are identified as in Figure 2-1a, where ℓ is the cosine of α(the angle the vector $\bar{U}$ makes with the x axis), m is the cosine of β(the angle the vector $\bar{U}$ makes with the y axis), and n is the cosine of γ(the angle the vector $\bar{U}$ makes with the z axis). Using the notation of Figure 2-1b for each of the coordinate axes (i',j',k') of the right handed system to be transformed, the following relationship results:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \ell_{ii'} & \ell_{ij'} & \ell_{ik'} & a \\ m_{ji'} & m_{jj'} & m_{jk'} & b \\ n_{ki'} & n_{kj'} & n_{kk'} & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \qquad (2\text{-}1)$$

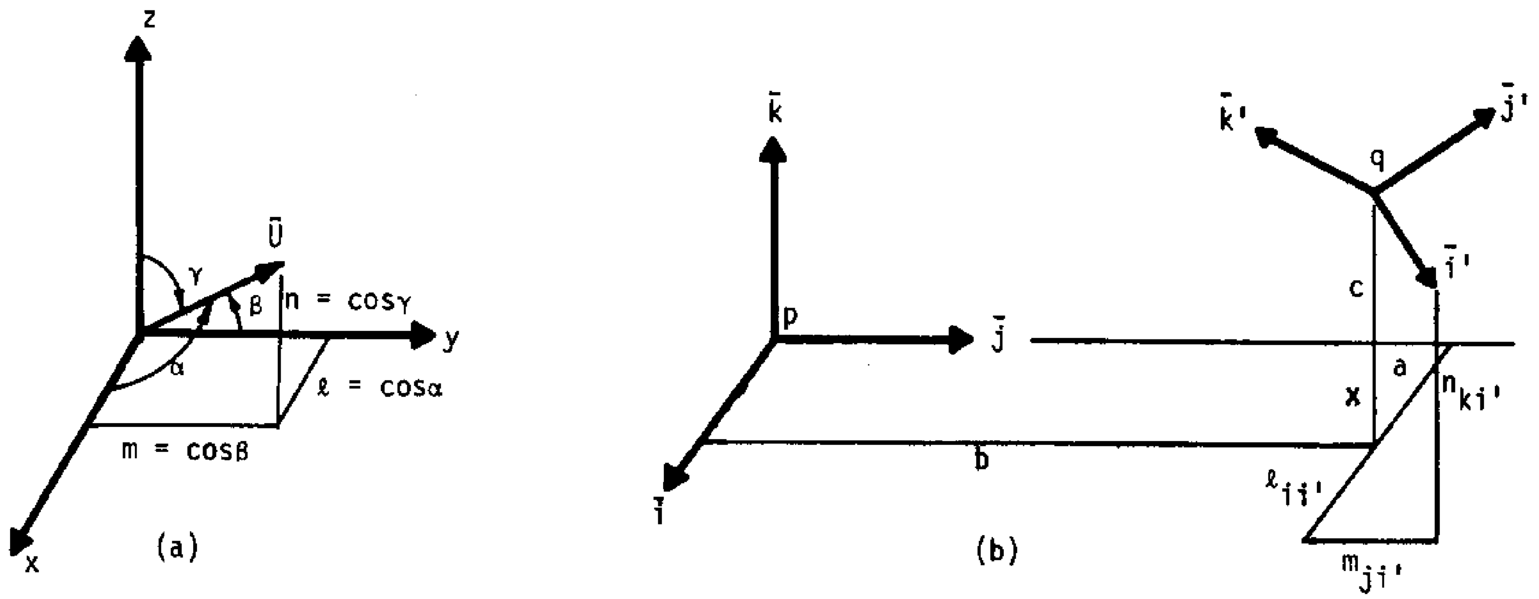


Figure 2-1: Transformation Notation

where,

$\ell_{i\zeta'}$ = the cosine of the angle between each of the three transformed axes (let ζ' = i', j', and k' respectively) and the new coordinate x axis.

$m_{j\zeta'}$ = the cosine of the angle between each of the three transformed axes (let ζ' = i', j', and k' respectively) and the new coordinate y axis.

$n_{k\zeta'}$ = the cosine of the angle between each of the three transformed axes (let ζ' = i', j', and k' respectively) and the new coordinate z axis.

The transformation matrix between coordinate system q and coordinate system p is denoted symbolically by

$$^{p}\underline{A}_{q}$$

which, when substituted into equation 2-1, results in the more succinct form of the transformation law

$$^{p}\underline{X} = {}^{p}\underline{A}_{q}\ {}^{q}\underline{X} \qquad (2\text{-}2)$$

Assuming there is a third coordinate system o and it is desired to express coordinate system q in o, it can be seen that by substituting p for q and o for p in equation 2-2 the transformation from p to o is obtained as

$$^{o}\underline{X} = {}^{o}\underline{A}_{p}\ {}^{p}\underline{X}$$

which when combined with the original result of equation 2-2 (i.e., $^{p}\underline{X}$) gives the desired transformation:

$$^{o}\underline{X} = {}^{o}\underline{A}_p \; {}^{p}\underline{A}_q \; {}^{q}\underline{X} \;=\; {}^{o}\underline{A}_q \; {}^{q}\underline{X}$$

The above equation demonstrates an important feature of the coordinate transform matrices — the transformation matrix from coordinate frame n to coordinate frame 1 can be obtained by multiplying the individual transform matrices from 1 to n

$${}^{1}\underline{A}_n = {}^{1}\underline{A}_2 \, {}^{2}\underline{A}_3 \; \ldots . \, {}^{n-1}\underline{A}_n \qquad (2\text{-}3)$$

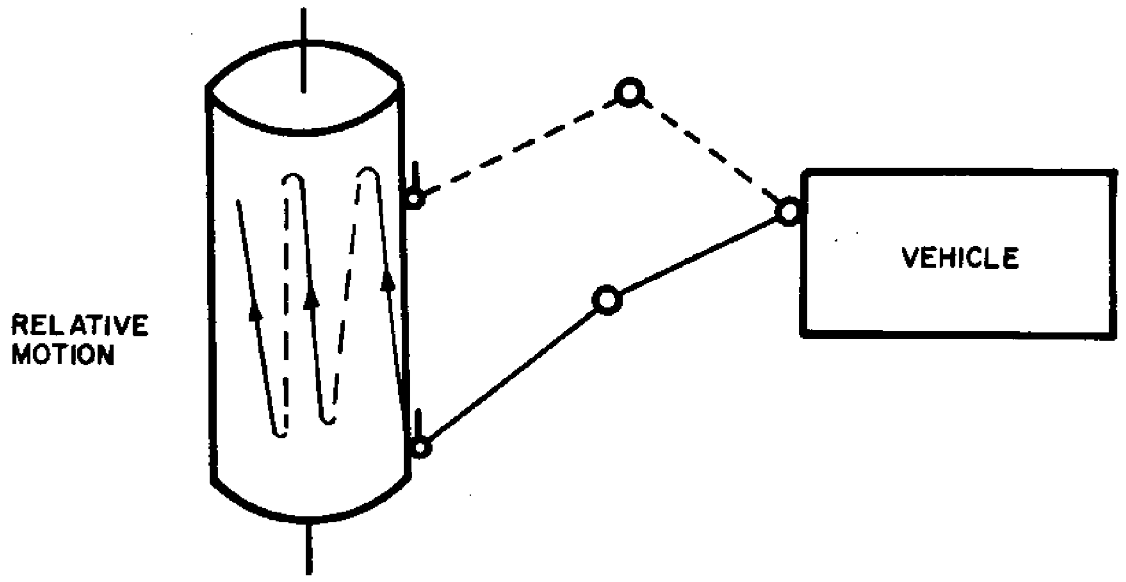
A specialized form of the transformation matrix has been derived by Roth and Pieper for a manipulator link with twist α, length a, revolution θ, and axis offset S.[10,11] Although this specialized matrix allows one to define the transformation parameters with four variables, in practice the notation (α, a, θ, and S) can be confusing if the original source of the matrix is not recognized. It is suggested, therefore, that the general transformation should be used.

2.3 Relative Manipulation Derivations

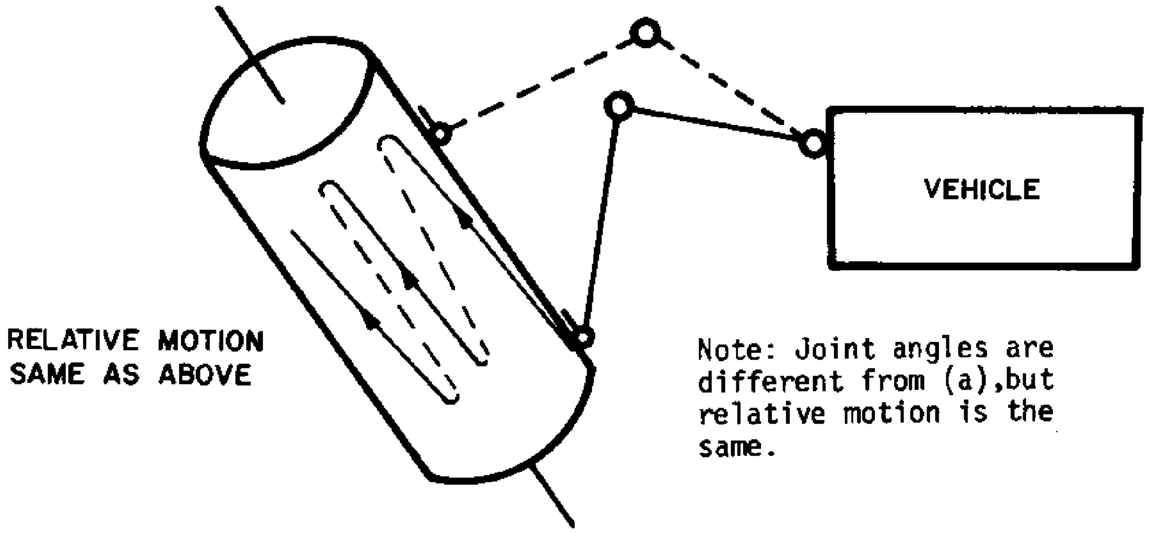
From the discussion in Section 2.1 (Absolute versus Relative Tasks) it should be clear that for an absolute task, coordinate transformations are not required. The only unknowns that have to be determined to perform the task are the desired joint angles, which can be specified when the task is defined. Tool retrieval is a good example of an absolute manipulation task (The tool will always be in the same location when the manipulator retrieves it, and hence, the required joint angles can be recorded once and stored for all time). In contrast to absolute manipulation, relative tasks require coordinate transformations and joint space solutions[*] each time the task is performed. As an example, imagine a program which scrapes a vertical cylindrical surface has been entered into the computer (Figure 2-2a). Later, when the program is executed, the computer is informed that the surface has changed its orientation so that it is skewed with the horizontal (Figure 2-2b). The method that will be presented in this section will allow the computer to adapt to the new orientation and scrape the surface as it originally did for the vertical orientation. This method is general enough to be applied to any relative function. For example, opening or shutting valves; putting on or taking off nuts and bolts; painting, cleaning, scraping, and jetting a surface; drilling; and tapping to name just a few. The key point to note is that the above tasks can be performed without reprogramming the computer each time it is confronted with a task which does not have the same spatial positions and orientations as when the task was initially learned by the computer.

To illustrate the method, imagine the desired relative task requires that a wrench be rotated to free a nut (Figure 2-3a). To obtain the counter-clockwise motion it is necessary for the hand to move from position A to position ε by passing through the inter-

[*]Given a desired hand position with respect to the vehicle, what joint angles will result in this position?



(a) Scraping Vertical Cylindrical Surface


RELATIVE MOTION SAME AS ABOVE
VEHICLE
Note: Joint angles are different from (a),but relative motion is the same.

(b) Scraping Skewed Cylindrical Surface

Figure 2-2

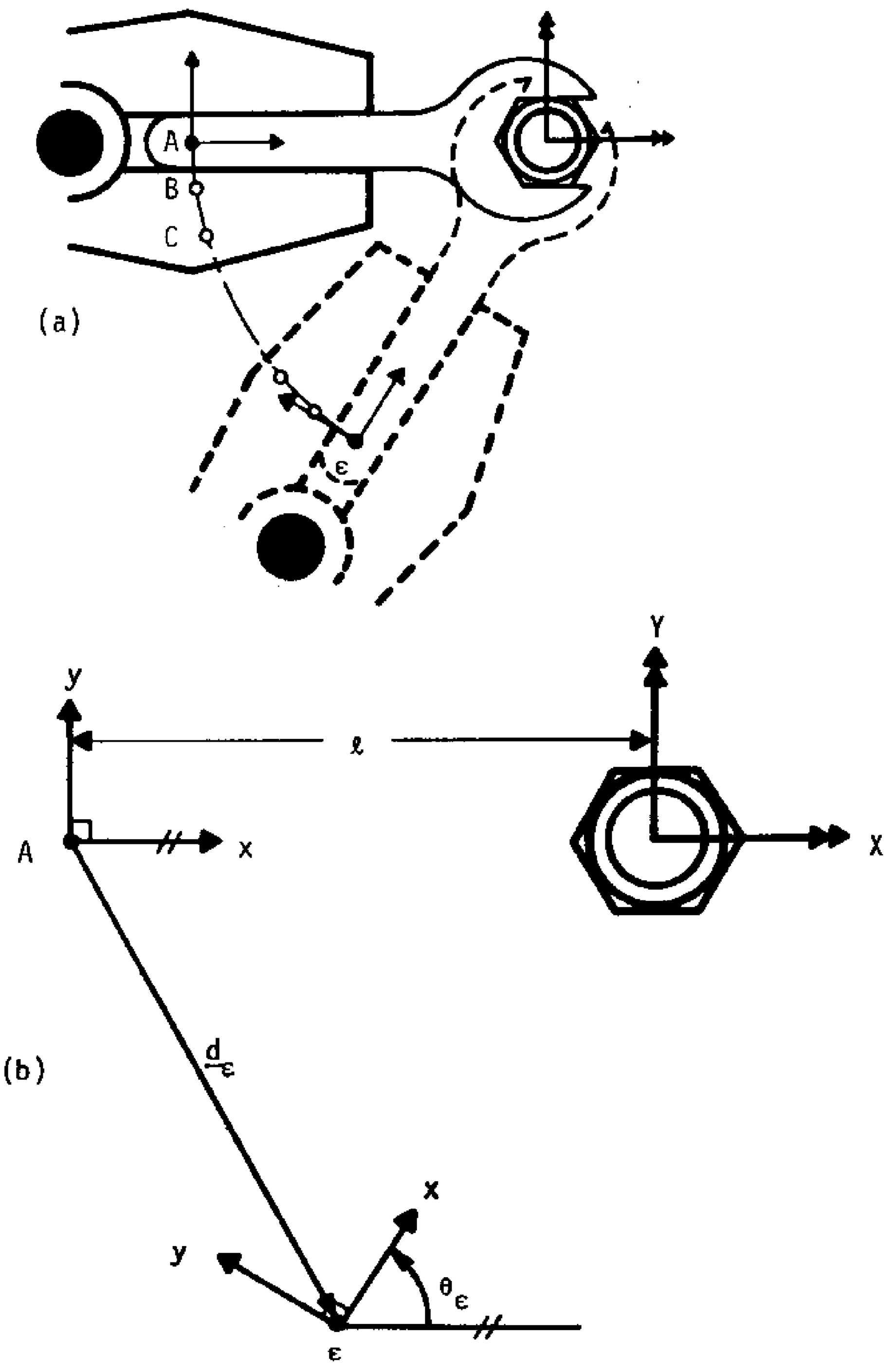
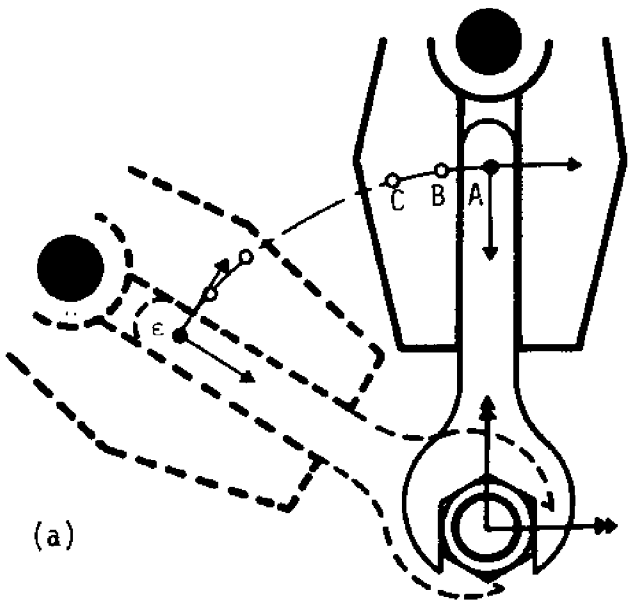


Figure 2-3: Relative Task with Horizontal Initial Position

mediate positions (Although the following development is primarily advanced in terms of points A and ε, it should be recognized that the results should be applied to all points A through ε). Using this simple coplanar example it will be shown that although the desired motion (i.e., turning the nut) requires different absolute positions for various orientations of the nut and wrench, the actions remain constant with respect to the initial hand position. Indeed, as mentioned in Section 2.1, all points in a relative task have the common attribute that each position maintains a fixed relationship with each of the remaining points independently of the world's coordinate system. To demonstrate this constant relationship note that frame ε (Figure 2-3b) is defined with respect to frame A by $\underline{d}_\varepsilon$ and θ_ε where $\underline{d}_\varepsilon$ is a vector in coordinate system A to point ε. Now assume that instead of having approached the nut from the negative x direction the wrench has moved toward the nut from a positive y position as in Figure 2-4a, and that the task still requires a counter-clockwise turning action. In terms of the world coordinates (double arrows in both Figures 2-3 and 2-4) the x and y displacements of A and ε in Figure 2-3 are given by $A(-\ell,0)$ and $\varepsilon(-\ell\cos\theta,-\ell\sin\theta)$, whereas, in Figure 2-4 the displacements of A and ε are given by $A(0,\ell)$ and $\varepsilon(-\ell\sin\theta,\ell\cos\theta)$. In contrast to the different positions of A and ε in the world system, the x and y displacements with respect to the initial hand coordinates, located at point A, remain fixed for both figures, i.e., $A(0,0)$ and $\varepsilon(\ell-\ell\cos\theta,\ell\sin\theta)$. A similar analysis can be used to demonstrate



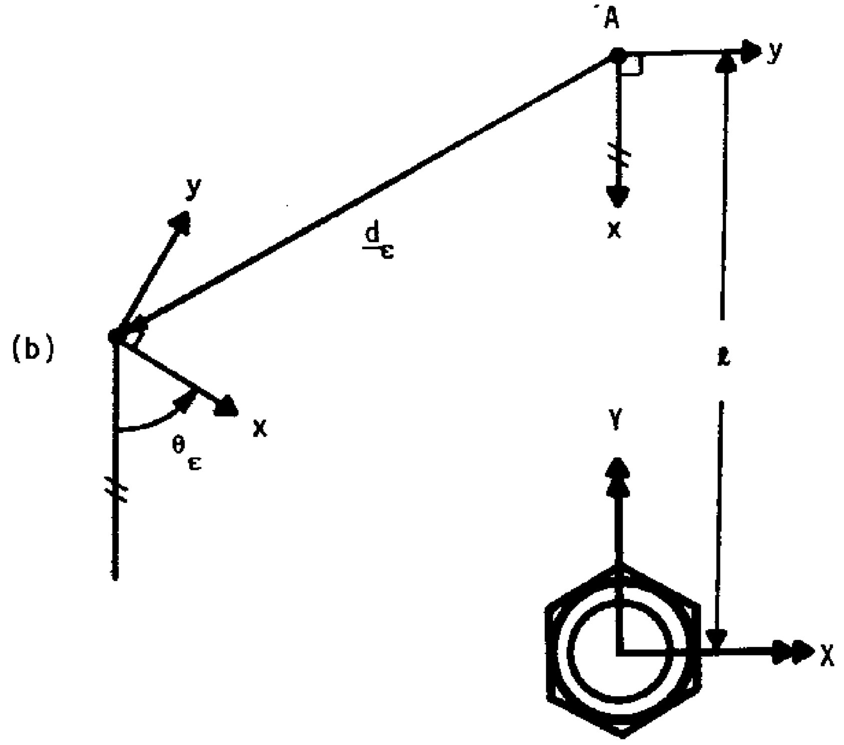


Figure 2-4: Relative Task with Vertical Initial Position

that the orientations also have a constant relationship with respect to the initial hand coordinate system (see figures). Hence, by comparison of the relative motion in Figure 2-3 and the relative motion of Figure 2-4 it is noticed that both tasks maintain constant positions relative to each other but that the world (spatial) positions required to achieve each task are different. Generally, all relative tasks have this feature in common, and it is this consistency which allows a task to be defined in one coordinate frame and through a modification of the original definition obtain the desired manipulator actions for any spatial orientation of the task.

Normally, manipulators do not output the position relative to some original hand orientation, and therefore, if the fixed relationship is to be of any benefit, it will be necessary to transform each position of the task to the initial hand coordinate system. This is accomplished by the use of the transformation matrix given in Section 2-2. Assuming the manipulator has six degrees of freedom, the transformation from hand coordinates to vehicle coordinates, or from vehicle coordinates to hand coordinates, will require six transformation matrices. For any six degree of freedom manipulator (Figure 2-5 and Appendix C), the transformation from the hand to the vehicle frame is given by

$$^{0}\underline{X}_{\varepsilon} = {}^{0}\underline{A}_{1}\,{}^{1}\underline{A}_{2}\,{}^{2}\underline{A}_{3}\,{}^{3}\underline{A}_{4}\,{}^{4}\underline{A}_{5}\,{}^{5}\underline{A}_{6}\,{}^{6}\underline{X}_{\varepsilon} = {}^{0}\underline{A}_{6}\,{}^{6}\underline{X}_{\varepsilon} \qquad (2\text{-}4)$$

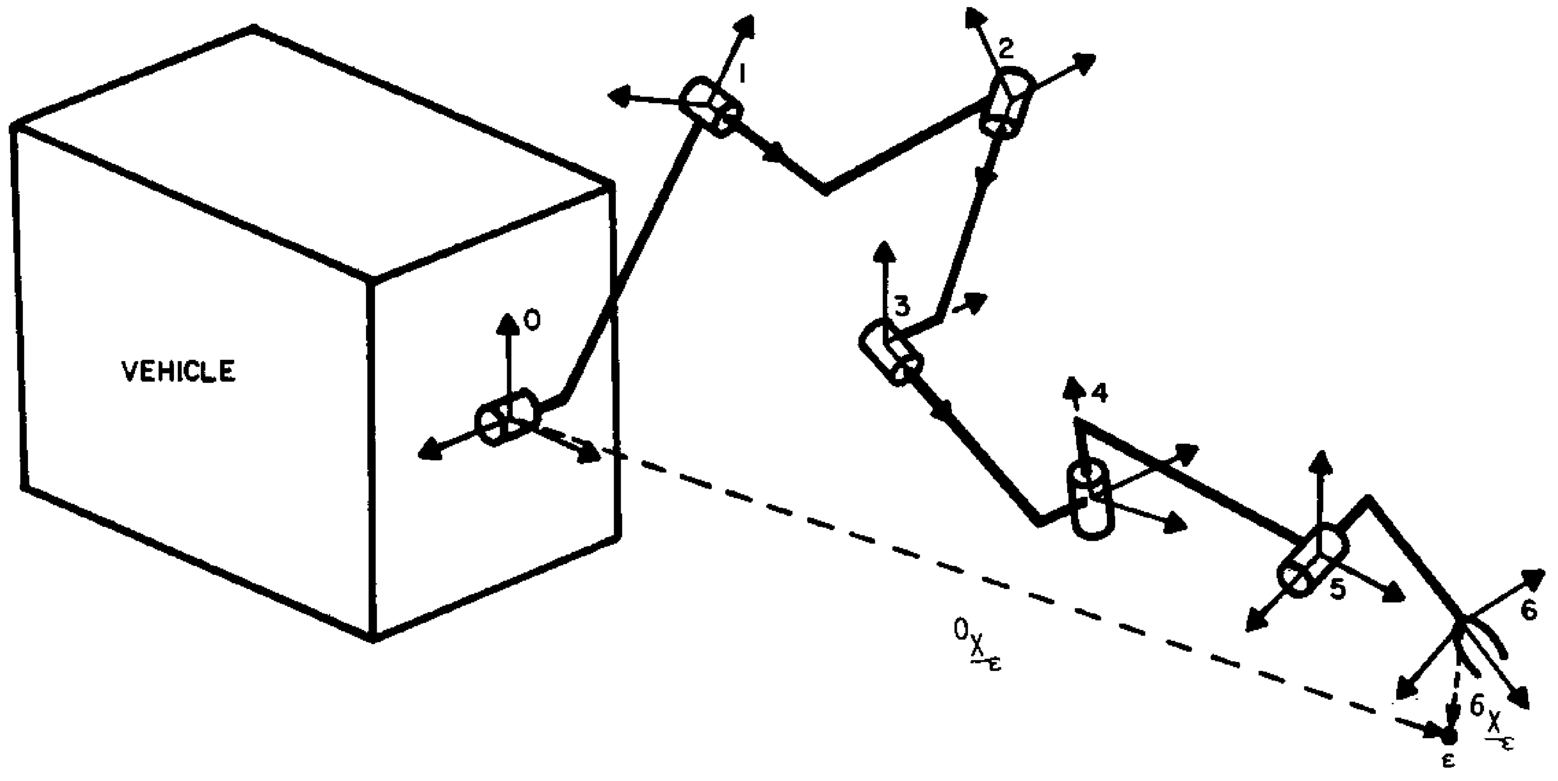


Figure 2-5: General 6 DOF Manipulator and Submersible

and the transformation from the vehicle to the hand is given by

$$^{6}\underline{X}_{\varepsilon} = {}^{6}\underline{A}_{5}\,{}^{5}\underline{A}_{4}\,{}^{4}\underline{A}_{3}\,{}^{3}\underline{A}_{2}\,{}^{2}\underline{A}_{1}\,{}^{1}\underline{A}_{0}\,{}^{0}\underline{X}_{\varepsilon} = {}^{6}\underline{A}_{0}\,{}^{0}\underline{X}_{\varepsilon} \qquad (2\text{-}5)$$

The equation which results by substituting the vehicle coordinates $^{0}\underline{X}_{\varepsilon}$ (equation 2-4) for each point $\varepsilon(\varepsilon = A,B,C\ldots)$ into the transformation from vehicle coordinates to the initial hand coordinates at point A (equation 2-5) is the transformation from the coordinates at point ε in space to the initial hand coordinate system*

$$^{6}\underline{X}_{\varepsilon} = {}^{6di}\underline{A}_{0di}\,{}^{0d\varepsilon}\underline{A}_{6d\varepsilon}\,{}^{6}\underline{X}_{\varepsilon} = {}^{6di}\underline{A}_{6d\varepsilon}\,{}^{6}\underline{X}_{\varepsilon} \qquad (2\text{-}6)$$

where (see also Figure 2-6),

0	-	signifies the vehicle coordinate frame (Figure 2-5)
6	-	signifies the hand coordinate frame (Figure 2-5)
d	-	signifies that the transformations are calculated from the joint rotations recorded at the time the task is defined
i	-	signifies the initial manipulator position (point A in Figure 2-3)
$^{6di}\underline{A}_{0di}$	-	is the transformation from vehicle coordinates (0) to the hand coordinate (6) in the initial position (i) at the time the task is defined (d).
$^{0d\varepsilon}\underline{A}_{6d\varepsilon}$		is the transformation from the hand coordinates (6) at point $\varepsilon(\varepsilon = A,B,C,\ldots,H)$ to the vehicle coordinates (0) at the time the task is defined (d).

*The vehicle coordinates are assumed to remain fixed throughout the learning portion of the task; therefore, since the manipulator base has the same relationship to the task when the hand is in the initial position as when the hand has been moved to position ε, the coordinate systems 0di and 0dε are equivalent.

$$\left[\begin{array}{c}\text{coordinate}\\ \text{frame 1}\end{array}\right]\left[\begin{array}{c}\text{time at which}\\ \text{transformation}\\ \text{occurs}\end{array}\right]\left[\begin{array}{c}\text{position of the}\\ \text{hand frame}\end{array}\right] \mathbf{A}_{\left[\begin{array}{c}\text{coordinate}\\ \text{frame 2}\end{array}\right]\left[\begin{array}{c}\text{time at which}\\ \text{transformation}\\ \text{occurs}\end{array}\right]\left[\begin{array}{c}\text{position of the}\\ \text{hand frame}\end{array}\right]}$$

Figure 2-6: Transformation Matrix Index Codes

It should be clear that the ${}^{6di}\underline{A}_{6d\varepsilon}$ matrices define the frames ε with respect to the initial hand position and that the initial hand coordinate system relative to itself is the identity matrix

$$
{}^{6di}\underline{A}_{6dA} = \underline{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2\text{-}7)
$$

Since the transformation matrix for the initial hand coordinate system relative to itself is always the identity matrix, regardless of the original position and orientation, and since each of the ${}^{6di}\underline{A}_{6d\varepsilon}$ matrices is relative to the initial hand coordinates, the relative position of the transformed points will always be the same regardless of how the task is defined.

The combined transformation matrix

$$
{}^{6di}\underline{A}_{6d\varepsilon}
$$

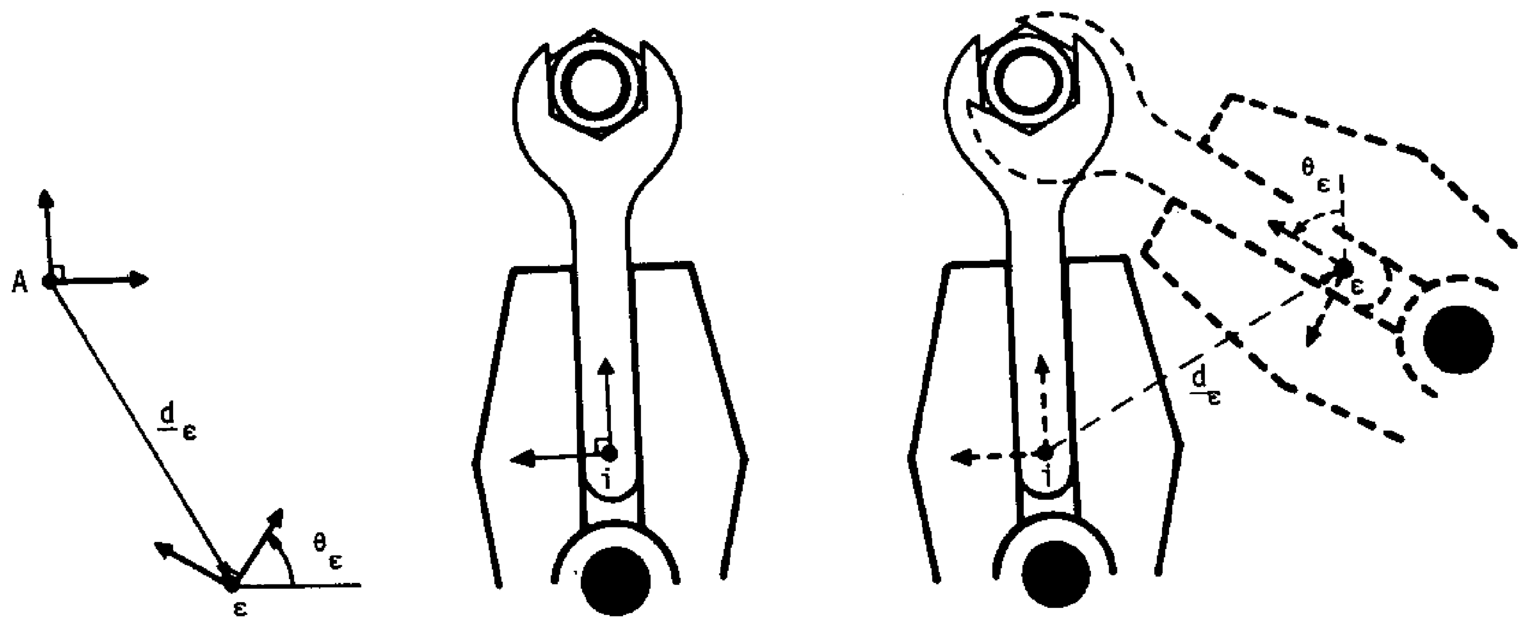
can be partitioned into a 3x3 direction cosine matrix, a 3x1 translational vector, a zero row vector, and a translational multiplier (1) in the same manner as the general transformation matrix in Section 2-2. The partitioned direction cosine matrix clearly gives the orientation of the hand coordinates at point ε with respect to the initial hand coordinate system at A, and the translational vector gives the displacement of the origin of system ε from the initial hand coordinate origin, which is the desired result. For the wrench rotation in Figure 2-3, the partitioned relative matrices would be given by

$$^{6di}\underline{A}_{6d\varepsilon} = \left[\begin{array}{ccc:c} \cos\theta_\varepsilon & \cos(90+\theta_\varepsilon) & 0 & \\ \cos(90-\theta_\varepsilon) & \cos\theta_\varepsilon & 0 & \underline{d}_\varepsilon \\ 0 & 0 & 1 & \\ \hdashline & \underline{0} & & 1 \end{array}\right]$$

where θ_ε and $\underline{d}_\varepsilon$ are defined as shown in Figure 2-3 (ε = A, B, C, ...).

It is important to note at this point in the development that the initial hand coordinate system referred to in the above analysis is the first position defined during the learning stage of the task, and hence, the required calculations can be performed at the time the task is defined and stored for use at a later date. Also, since it is desired that all positions be in terms of the initial manipulator position when the task is defined, the transformation from the vehicle to the initial hand coordinates ($^{6di}\underline{A}_{0di}$) will be constant for all points in the task and will only have to be calculated once. For example, in the wrench rotation task, the manipulator would be moved to position A, the joint angles would be recorded and used to calculate the constant matrix ($^{6di}\underline{A}_{0di}$), and the results would be stored. Then the wrench would be moved to position B, the joint angles would again be recorded, the $^{0dB}\underline{A}_{6dB}$ matrix would be calculated and multiplied by the stored constant matrix, and the result would be stored as the relative transformation between point B and the initial hand coordinates (${}^{6di}\underline{A}_{6dB}$). Each of the successive points C,D,E,... would be converted to relative transformations until all the task points were relatively defined with respect to the initial hand position. Obviously, all of the processing of the relative transformation can take place during the learning mode, and hence, will not slow down the real-time execution of the task.

Although the relative frames, ε have been defined with respect to some initial hand frame, the position of each of the relative frames ε will have to be defined in vehicle coordinates to successfully perform the task at the time of execution. Intuitively, the defined relative motion (Figure 2-3b) and the relative motion desired at execution (Figure 2-4b) are exactly the same, although the absolute spatial positions are not. The fundamental difference between the two tasks is the starting position and orientation. The relative motion required in Figure 2-3 and Figure 2-4 has been redrawn in Figure 2-7a to graphically represent the relative transformation matrices (${}^{6di}\underline{A}_{6d\varepsilon}$) which would be calculated and saved at the time the task was defined (note the absence of the nut in the internal computer's model of the relative task - the computer only knows that a specified relative motion is required given a particular initial hand position). Now, imagine that the manipulator and wrench are moved to the nut as shown in Figure 2-7b. Further, imagine that the relative motion of Figure 2-7a is lifted off the page and that frame A is placed over the initial hand frame in Figure 2-7b. The



(a) Relative Motion Stored in Computer

(b) Initial Execution Hand Frame

(c) Relative Motion Superimposed on Initial Hand Frame

Figure 2-7: Relative Task Positions Determined at Execution

result of superimposing the relative frames on the initial hand frame is the desired relative motion (Figure 2-7c).

Clearly, once the manipulator has been placed in the position at which the task is to be performed (i.e., once the initial execution time hand coordinates are known), the vehicle coordinates of each ε can be found by superimposing the relative transformations $({}^{6di}\underline{A}_{6d\varepsilon})$ on the initial execution frame and transforming from the relative frames to the vehicle frame.

As stated, the initial hand coordinate system relative to itself is the identity matrix

$$ {}^{6di}\underline{A}_{6dA} = \underline{I} \qquad (2\text{-}7) $$

Superimposing the initial hand frame on the execution time hand frame gives

$$ \begin{aligned} {}^{0}\underline{X}_{A} &= {}^{0ei}\underline{A}_{6ei}\,{}^{6di}\underline{A}_{6dA}\,{}^{6}\underline{X}_{A} = {}^{0ei}\underline{A}_{6ei}\,\underline{I}\,{}^{6}\underline{X}_{A} \\ &= {}^{0ei}\underline{A}_{6ei}\,{}^{6}\underline{X}_{A} \qquad (2\text{-}8) \end{aligned} $$

where,

0 - signifies the vehicle coordinate frame (Figure 2-5).

6 - signifies the hand coordinate frame (Figure 2-5).

e - signifies that the transformations are calculated from the joint rotations recorded at the time the task is executed.

i - signifies the initial manipulator position when the transformations are calculated.

${}^{0ei}\underline{A}_{6ei}$ is the transformation from the initial (i) hand coordinates (6) to the vehicle coordinates (0) at the time the task is executed (e).

The remaining relative positions in vehicle coordinates can be obtained by multiplying the relative transformations (${}^{6di}\underline{A}_{6d\varepsilon}$) times the transformation from the initial execution time hand coordinates to vehicle coordinates

$$ {}^{0}\underline{X}_{\varepsilon} = {}^{0ei}\underline{A}_{6ei}\ {}^{6di}\underline{A}_{6d\varepsilon}\ {}^{6}\underline{X}_{\varepsilon} = {}^{0ei}\underline{A}_{6d\varepsilon}\ {}^{6}\underline{X}_{\varepsilon} \tag{2-9}$$

where 6di equals 6ei by superposition of the initially defined hand frame with the initial execution hand frame.*

Once the relative transformation matrices are known and an initial position at the time of execution has been specified, all of the positions with respect to the vehicle can be found. The three displacements and three orientations associated with point ε can be determined directly from the transformation matrix

$$ {}^{0ei}\underline{A}_{6d\varepsilon} = \begin{bmatrix} \ell_{ii'} & \ell_{ij'} & \ell_{ik'} & a \\ m_{ji'} & m_{jj'} & m_{jk'} & b \\ n_{ki'} & n_{kj'} & n_{kk'} & c \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

*Note that the ${}^{0ei}\underline{A}_{6ei}$ matrix is a constant, and therefore, would only have to be calculated once.

by noting that,

a	=	the desired x translation of point ε from the origin of the vehicle coordinates at execution time (Figure 2-2).
b	=	the desired y translation of point ε from the origin of the vehicle coordinates at execution time (Figure 2-2).
c	=	the desired z translation of point ε from the origin of the vehicle coordinates at execution time (Figure 2-2).
$\begin{bmatrix} \ell_{ii'} \\ m_{ji'} \\ n_{ki'} \end{bmatrix}$	=	the direction cosines that the i' axis (the relative x axis at point ε) makes with the respective axes of the vehicle coordinate system.
$\begin{bmatrix} \ell_{ij'} \\ m_{jj'} \\ n_{kj'} \end{bmatrix}$	=	the direction cosines that the j' axis (the relative y axis at point ε) makes with the respective axes of the vehicle coordinate system.
$\begin{bmatrix} \ell_{ik'} \\ m_{jk'} \\ n_{kk'} \end{bmatrix}$	=	the direction cosines that the k' axis (the relative z axis at point ε) makes with the respective axes of the vehicle coordinate system.

Unfortunately, these displacements and orientations are not the required joint angles, but only translations and rotations of the hand with respect to the vehicle frame. It is necessary, therefore, to solve for the joint angles which will result in these hand positions. There have been a number of methods developed to find the joint angles which result in a required position, and therefore, specific details will not be given. But a brief comparison of these methods will be included in Chapter III to demonstrate the need for proper manipulator selection at the earliest stages of planning and development.

2.4 Fixed versus Moving Tasks

Although the two qualities absolute vs relative and fixed vs moving may appear to be related concepts, they are actually independent task variables. A task can be absolute and fixed, or absolute and moving, or relative and fixed, or relative and moving. Consider retrieving a part from a rack attached to the manipulator base (absolute and fixed) in comparison to obtaining a part from a conveyer belt which delivers the part in exactly the same manner everytime (absolute and moving). Both tasks are absolute since neither requires that the manipulator be informed of the geometric relationship between the base and the task coordinate systems. The primary difference between the two tasks is that the task frame changes as a function of time with respect to the base frame during execution for one task and does not for the other. The definition of a fixed manipulation task follows:

> Fixed Task - a task in which the manipulator base remains stationary with respect to the task frame during execution.

To illustrate a fixed relative task imagine that it is necessary for a submersible to settle on the bottom beside a wellhead to loosen a gland nut on a valve. Further assume that due to the location of the nut an impact wrench will not fit over the nut, that the only method of approach is from the side, and that once the nut is free the manipulator gripper has enough force to turn the nut but the gripper cannot initially free the nut. The required functions should be recognized as a relative task as the relationship between task and vehicle can only be ascertained after the vehicle has settled on the bottom at the site. Also once the relationship of the task and vehicle has been determined it remains constant, and hence, the task is fixed.

To perform this task with a computer controlled manipulator the following algorithm, which starts with the gripper closed on the nut, would be defined on the surface as follows:

1) Open end effector.
2) Obtain wrench to free nut (this will be an absolute task as the wrench will always be located in the same position on the tool rack).
3) Place wrench on nut (the initial manipulator position, with the gripper on the nut, defined the nut location).
4) Turn wrench counter-clockwise and free nut (this relative task was defined in Section 2.3).
5) Return wrench to tool rack (absolute task).
6) Return to nut and turn counter-clockwise with gripper until nut is free.
7) Return control to human operator and await further instructions.

Use of the algorithm would proceed as follows:

1) Remote vehicle settles on bottom next to task (fixed manipulation).

2) Operator on surface moves manipulator to nut and grasps it.

3) Operator calls stored program to remove gland nut automatically (the operator is essentially saying to the the computer "Here is the nut — now take it off").

4) Computer takes initial manipulator position, overlays the predefined relative task on that position, and determines the joint commands required to complete the task.

5) Computer performs task and returns control to human operator.

This is an example of a fixed relative task which can be performed with the human operator specifying the initial position. But the human operator's ability to define the initial execution position is extremely limited unless the first position is easily obtained and clearly specified (in the above example, the gripper is placed on the nut with the gripper axis perpendicular to the nut axis). The scraping of a cylinder in Section 2.3 is a good example of the human operator's limited ability to define the initial position of a relative task. Unless the end effector is in exactly the proper position at execution time, small angular errors of the hand could multiply to intolerable levels as the distance from the hand origin increased. To illustrate, imagine that the end effector is slightly skewed with respect to the axis of the cylinder. The computer would assume that the cylinder is also skewed and would scrape the cylinder as shown in Figure 2-8.

Clearly a method is needed to insure that the end effector is oriented properly to perform the task. The device should not be expensive, and it should not require an involved procedure. It was suggested by Ofer Gneezy of the M.I.T. Man Machine Systems Lab that the tool handles used for the experiments (Chapter 5) if permanently mounted on a task would allow the operator to specify the initial position. A special tool evolved from this idea which would plug into a socket mounted on the task (Figure 2-9). With this orienting device the operator would approach the socket and then force steer[12,13] the tool until it mated with the socket. After the orienter is inserted, the relationship of the task to the vehicle would be known and fixed as long as the vehicle does not move. But suppose the vehicle does move?

The definition of a moving manipulation task follows directly from the definition of a fixed task:

> Moving Task - a task in which the task frame changes as a function of time with respect to the base frame during execution.

In many circumstances it is necessary to perform a task which is located in a position that can only be reached when the submersible is off the bottom. Many remote manipulator systems simply ignore

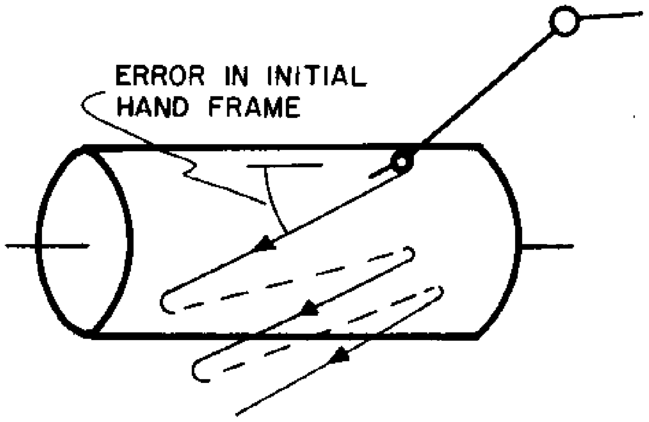


Figure 2-8: Scraping Cylinder with Error in Initial Hand Position

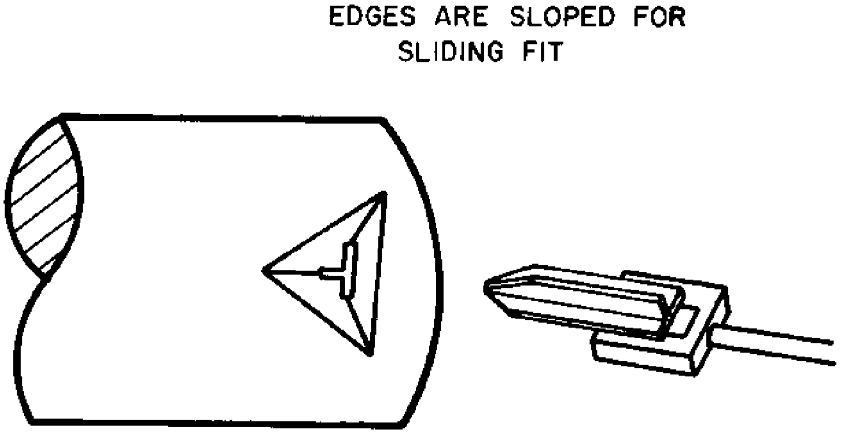


Figure 2-9: Male-Female Orientation Tool

this problem, with the remaining systems attempting to solve the problem by holding the submersible firmly fixed to the structure by means of mechanical grabbers. Unfortunately, fixing the submersible is not always a practical solution — at times there is literally nothing strong enough to grab onto. Also, this method does not completely fix the submersible as the structure, links, and joints have limited stiffness.

If the arm response speed is faster than that of the submersible response to outside disturbances, it is theoretically possible to have the arm correct itself for the motion of the submersible. Obviously, to develop such a method it will be necessary to obtain the position of the vehicle with respect to the task coordinate system. The relationship between the task and base frames can easily be obtained with an orientation manipulator:*

> <u>Orientation Manipulator</u> - a device consisting of a fixed number of links and joints with position sensors which, through the use of transformation matrices, gives the position of the vehicle frame with respect to the task frame.

Possible designs for the orientation manipulator are shown in Figures 2-10 and 2-11. A simple magnetically-coupled orienter (Figure 2-11), which defines the state of the manipulator base with respect to the task, could be used for tasks without the built in orientation sockets (Figure 2-10). The design shown in the figures (universal joints with a telescoping mid-section) was chosen

*Laser and sonic triangulation, or direct machine vision are other possibilities, but the orientation manipulator is economically more attractive.

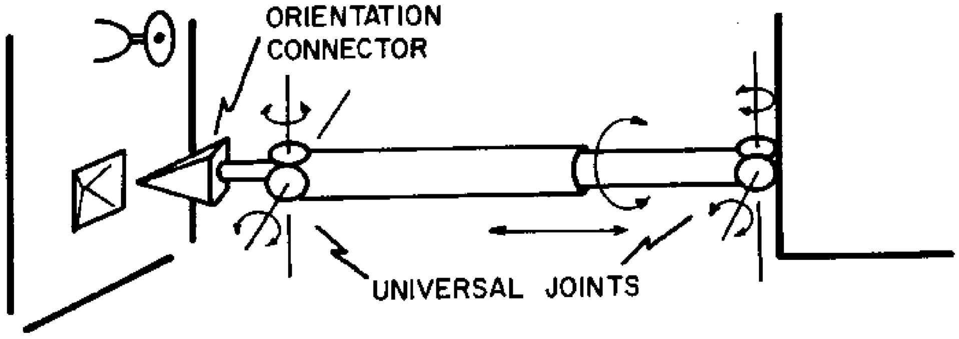


Figure 2-10: Orientation Manipulator for Tasks with Built-In Sockets

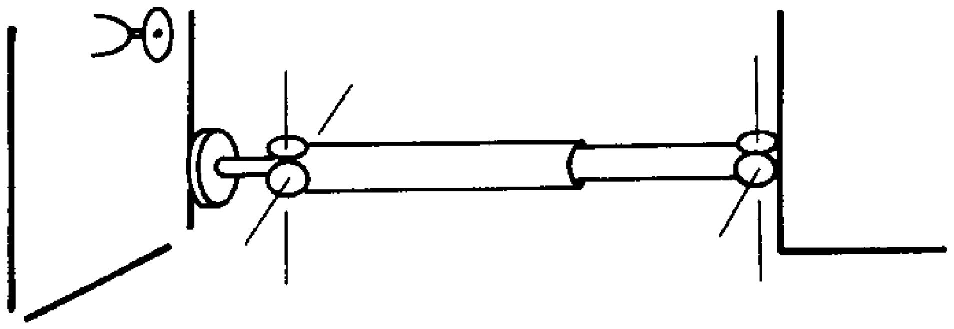


Figure 2-11: Orientation Manipulator with Magnetic Coupler for Tasks without Built-In Sockets

to allow the vehicle to move in all six degrees-of-freedom without locking or binding a joint.

Now that a simple method is available to obtain the vehicle position with respect to the task it is necessary to derive the transformation equations which will result in the required incremental joint angles.

2.5 Moving Manipulation Derivations

To develop the moving transformation equations it will be assumed that the task has been defined in a relative manner as described in Section 2.3, and that the submersible moves with respect to the task.* It will also be assumed that the vehicle has approached the task, connected the orientation sensor, and begun execution of the predefined relative task. Since the task is assumed fixed and the submersible is assumed to move with respect to the task, it is clear that the vehicle base cannot be used to define the relative task positions. Therefore, the relative positions must be transferred to the fixed immovable task frame. The relative positions have been determined in a previous section and are given by equation 2-9,

$$ {}^{0}\underline{X}_{\varepsilon} = {}^{0ei}\underline{A}_{6d\varepsilon}\, {}^{6}\underline{X}_{\varepsilon} \qquad (2\text{-}9) $$

*Whether the task moves, the submersible moves, or both move is irrelevant since all that is required is that a relative motion between the task and vehicle exists.

To define the relative positions in the task coordinate system it is only necessary to transform from the vehicle frame to the task frame at the instant the task is executed,

$$^{T}\underline{X}_{\varepsilon} = {}^{Tei}\underline{A}_{0ei}\,{}^{0ei}\underline{A}_{6d\varepsilon}\,{}^{6}\underline{X}_{\varepsilon} = {}^{Tei}\underline{A}_{6d\varepsilon}\,{}^{6}\underline{X}_{\varepsilon} \qquad (2\text{-}10)$$

where,

T - signifies the task coordinate system (Figure 2-12).

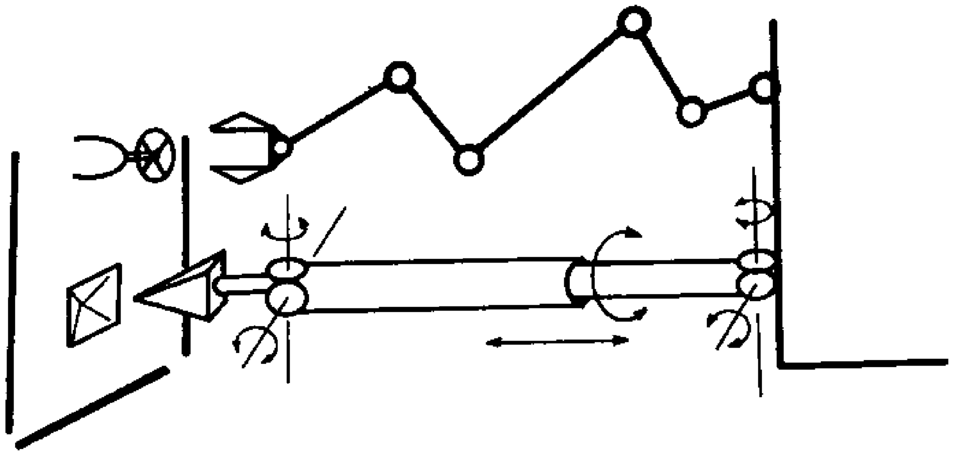
${}^{0ei}\underline{A}_{6d\varepsilon}$ - is defined in Section 2.3

${}^{Tei}\underline{A}_{0ei}$ - is the transformation from vehicle coordinates (0) to task coordinates (T) at the time the task is executed (e) with the hand frame in its initial position (i).

The relative positions are now completely known in the task coordinates. It should be noted before proceeding with the moving derivations that equation 2-10 can be obtained by two distinct methods:

1) If the entire task can be defined prior to submergence, the matrices ${}^{Tei}\underline{A}_{6d\varepsilon}$ can be directly calculated when the task is defined and saved for future use (It should be recognized that to define these positions it is not necessary to enter the data from blueprints — the positions can be obtained by moving the arm through the desired points and calculating the transformation matrices directly).

2) If the task cannot be completely specified in terms of the task frame before submergence, then the matrices ${}^{Tei}\underline{A}_{6d\varepsilon}$ can be calculated on the bottom after the human operator has "shown" the computer the task.


(a) Orientation Manipulator and Work Arm

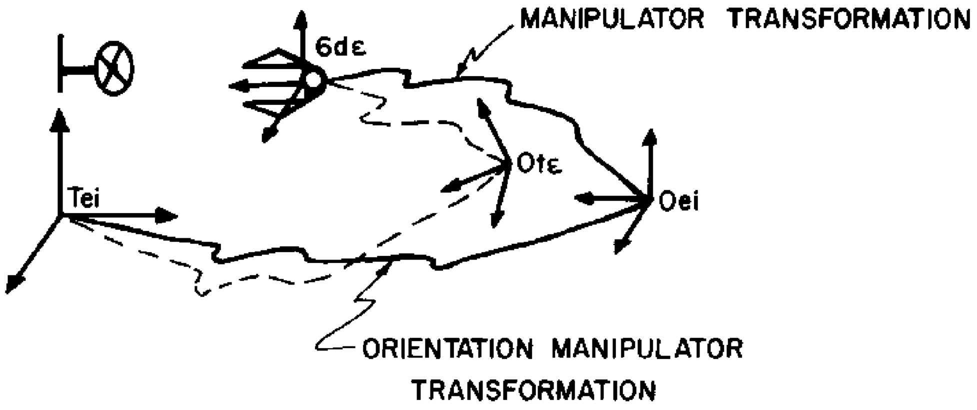

MANIPULATOR TRANSFORMATION
6dε
Otε
Tei
Oei
ORIENTATION MANIPULATOR TRANSFORMATION
(b) Frame Assignments for Moving Task

Figure 2-12

Now assume the vehicle is free to move again and that the vehicle has changed its position by a small increment due to drift, current, etc. Since the relative positions are known in the fixed task frame for any instant of time, the desired task coordinates in "moving" vehicle coordinates are readily given by,

$$ {}^{0}\underline{X}_{\varepsilon} = {}^{0t\varepsilon}\underline{A}_{Tt\varepsilon}\, {}^{Tei}\underline{A}_{0ei}\, {}^{0ei}\underline{A}_{6d\varepsilon}\, {}^{6}\underline{X}_{\varepsilon} \qquad (2\text{-}11) $$

where,

t	-	signifies that the transformation occurs at time $t>0$.
${}^{0t\varepsilon}\underline{A}_{Tt\varepsilon}$	-	is the transformation from the task coordinates (T) to the vehicle coordinates (0) at a particular instant of time (t) with the hand frame in position ε (See Figure 2-12b).

The procedure to use this method would be as follows:

1) Calculate the transformation matrices ${}^{Tei}\underline{A}_{6d\varepsilon}$ which defines the relative task (either on the surface or at the task site).
2) Calculate the required vehicle coordinates of the task at time t by obtaining the matrix ${}^{0t\varepsilon}\underline{A}_{Tt\varepsilon}$ and multiplying it by the matrices ${}^{Tei}\underline{A}_{6d\varepsilon}$.
3) By either analytical or iterative techniques* determine the incremental joint commands (The iterative techniques would appear to lend themselves naturally to this method).
4) Return to 2 and continue until task is finished.

These derivations were advanced in terms of a relative task, but the results can be applied to any task or control mode (manual or computer) with minimal modifications.

*An explanation of these techniques is given in Chapter III.

Once the orientation manipulator is connected to the task, the hand position with respect to the task would be frozen, regardless of the motion of the submersible——only commands from the operator or computer would change the hand position, although the joint angles would be in a continuous state of correction.

It is possible to conceive of a system which, through the use of the orientation manipulator and moving transformation equations, could act as an autonomous robot for extended periods of time. From the moment the orientation manipulator is manually connected to the task, the remote computer would perform the task, or sequence of subtasks, in a completely autonomous manner, only returning to the operator for further instructions or when it gets into trouble. A fleet of these manipulator vehicles could be dropped overboard, plugged in and left to complete the task. Moving manipulations could also be used for space applications.

Now that the definitions and equations for absolute, relative, fixed, and moving manipulation have been determined, we will proceed to investigate some of the design principles of a supervisory system.

CHAPTER III

A UNIFIED THEORETICAL FRAMEWORK FOR SUPERVISORY MANIPULATION

To design an effective supervisory system it is necessary to investigate some of the theoretical aspects of supervisory control. Four factors have been identified which should be considered when developing a supervisory system: (1) manipulator/processor selection, (2) control philosophy; (3) interface design; and (4) language philosophy. Each of these factors interacts with one another, and therefore, none of these factors is independent (e.g., if man-machine interaction is through a single communication channel, the sophistication of the language will be determined by the restraints imposed by that channel, etc). The following sections discuss these factors in more detail.

3.1 Manipulator/Processor Selection

Several investigators have identified the following as important design factors for computer-controlled manipulators:[11,14,15,16,17]

Kinematic Design Factors

1) degrees of freedom
2) joint types and configuration
3) link parameters (length, twist, and offset)
4) workspace
5) approach angles of end effector
6) operation zone of each link
7) obstacle avoidance
8) joint space solvability

Physical Design Factors

1) load capacity
2) link/joint mass
3) link/joint stiffness
4) accuracy/precision
5) repeatability
6) reliability
7) backlash
8) speed
9) acceleration
10) frequency response
11) stability
12) power requirements

Processor Selection Factors

1) computation speed
2) memory capabilities
3) I/O capabilities (man-processor-manipulator interfaces)
4) reliability
5) power requirements

Many of these parameters have been extensively treated in the literature, and therefore, do not require further attention. But one factor, joint space solvability[*], negatively influences many of the other design factors, and hence, warrants further consideration.

For computer manipulations to be successfully performed, it is absolutely essential that the arm have a solution which can be

[*]A solution is the determination of the joint rotations, angular velocity and angular acceleration by an algorithm given a desired spatial position.[11]

obtained under the time constraints of the real-world. But, it should not be immediately assumed that all manipulators have solutions and that a computer can solve the joint space within those real-time constraints. In fact this important parameter, which should be considered in the initial stages of planning a computer controlled manipulator, is often the one design factor which is rarely examined until the physical design has been set.*

Generally there are two methods available to determine the joint space solution; one obtains the solution by iterative technqiues and the other by a closed form analytical expression.

A short discussion of the iterative and analytical methods of joint space solution will be included to demonstrate the need for proper manipulator selection from the start of the program if computer control is to be used.

The iterative techniques find a solution by taking a number of small steps until the resulting values converge on the required position.[10,11,18] But iterative methods have three recognized shortcomings: (1) they consume more time to determine the joint space than the analytic solutions; (2) at certain positions they take an unwieldy number of calculations; and (3) they usually result in only

*This is probably due more to the fact that most computer controlled arms in the past have been converted master/slave, analog, joystick, or rate controlled arms, rather than systems specifically designed for supervisory control.

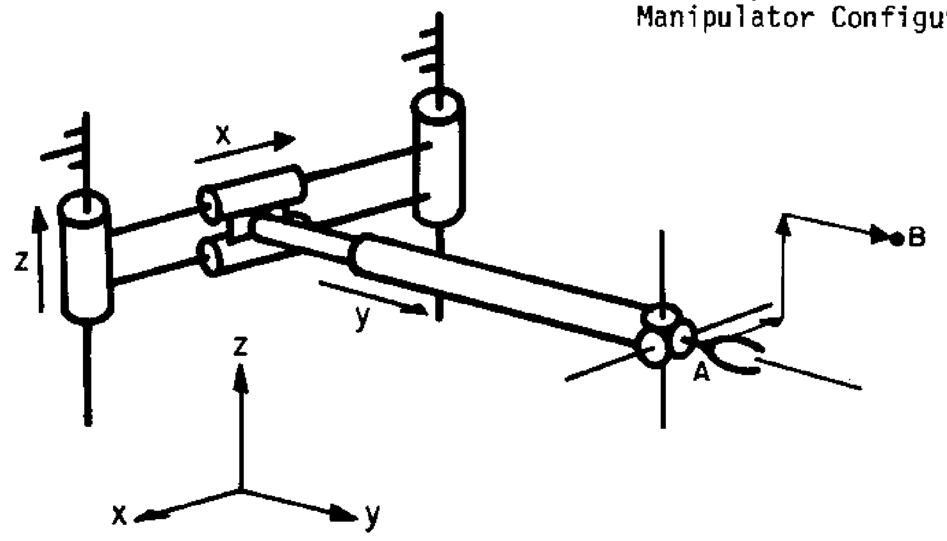
one of many possible configurations.[11] The time drawbacks can be overcome to a large extent by using the incremental result as joint driving signals. If this scheme is used, each iteration results in a driving motion which when properly timed results in almost no computation lag time, with the result that the iterative methods appear faster than many closed form analytic solutions.[18,19]

One of the first attempts to obtain an analytical joint space solution was made by Pieper, who more notably describes solutions for any manipulator with three intersecting axes.[10] Pieper also gives a method to determine if a particular manipulator configuration has a closed form solution — an important attribute for computer manipulations. Generally the method used to solve for the joint space makes use of the fact that the x, y, z, α, β and γ displacements are known, and therefore, since there are six knowns and six unknowns (the required joint angles) the solution can be found. But due to the nature of the equations, multiples of sines and cosines, a polynomial of the 524,288th degree results if a simple substitution and elimination scheme is used.[10] Even if the extraneous roots were removed, the polynomial would still result in approximately 64,000 possible solutions, of which only a few would be valid joint angles for the particular manipulator.[10] A further simplifying assumption must be made to achieve a tenable solution. If the manipulator has three axes which intersect the problem can be broken down into two sets of equations in which three of the unknowns can be solved for independently of the remaining three unknowns.[10] This method generally results in a closed form solution which requires at most the solution of a fourth degree polynomial——a considerable simplification (See Pieper's thesis [10] for a more detailed explanation). Other methods make use of an array of simplifications to arrive at a tenable solution.

Although it should be expected that in the future all manipulator configurations will be solvable, it should be kept in mind that the number of computations increase as the complexity of the arm increases.[11] As an example of the two extremes, trivially solvable and unsolvable, consider the manipulators in Figure 3-1. The manipulator in Figure 3-1a has three translational axes that intersect at the base and three rotational axes that intersect at the wrist. Clearly, the analytic solution to go from position A to position B is easily calculated as the desired motions correspond directly to the joint movements. On the other hand, consider the manipulator in Figure 3-1b which has six degrees of freedom with no intersecting axes. The required joint motions to move the hand from A to B is not immediately evident, and as a matter of interest this case has not been solved in a closed form to this date.

In terms of solvability it would appear that the unsolvable manipulator only offers increased computational complexity. But, comparison with the other design factors (e.g., obstacle avoidance, approach angles and workspace) clearly shows the superiority of the general six degree-of-freedom manipulator over the trival configuration. The general manipulator could easily reach around an obstacle, whereas,

(a) Trivally Solvable
Manipulator Configuration



(b) Unsolvable Manipulator
Configuration

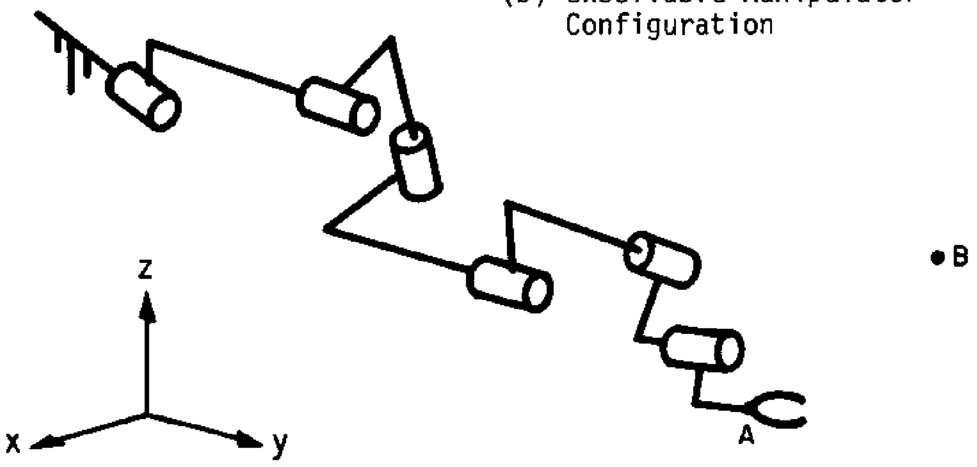


Figure 3-1

the geometrically simpler arm could not. It can also be shown that the workspace and approach angles of the general configuration are not as limited as those of the simpler manipulator. Clearly, as the generality (i.e., the ability to avoid obstacles, to reach more positions in the work area, and to approach an object from different angles) increases, so does the number of calculations required to solve for the joint space in closed form. In constrast to the analytical solution, an iterative technique generally requires the same number of calculations for the trival solution as the general solution (e.g., the linearized equations of motion result in six incremental equations in six unknowns which are easily solved by a 6x6 matrix inversion). But using the iterative method defeats the purpose of generality since it results in only one out of the possible 32 different joint configurations[11]. To obtain the 32 configurations an iterative search of the joint space would have to be performed with the end result that as the generality increases the calculations increase (This method could be extremely expensive computationally, compared to the closed form solution).

Clearly, the two design factors, generality and joint space solvability, oppose each other computationally. The question of how to select the degree of generality versus solvability is still unanswered. In fact Roth[14] has said (speaking on kinematic design of a manipulator) that, ". . . except for simple geometries there exists almost no rational way to make decisions". The decision can only be based on the comparison of the manipulator configuration with the speed of computation. That is, given a specific configuration, how fast must the processor be to generate real-time joint space solutions? Inversely, given a fixed processing speed, how much generality can the manipulator have and still be solvable in real-time?

In many industrial applications it is possible to precalculate the joint-space solution prior to task execution, and hence, processor speed is not as important. But teleoperators under supervisory control are by nature real-time processors. Therefore, anything that reduces the calculation time will increase the available generality.

Some of the methods by which calculation speeds can be increased are:

1) hardware multiply/divide
2) memory look up tables
3) optimized programs
4) special matrix manipulation features
5) parallel/distributed processing
6) precalculation of variables when possible

To conclude this section, there are a number of kinematic, physical, and processor design factors which influence the ultimate configuration of a supervisory controlled remote manipulator. There are undoubtedbly many more which will become apparent as these systems become more sophisticated. To date there is no concise tool for determining the optimal configuration given the desired attributes.

Therefore, design decisions will have to be made on a comparison and tradeoff basis until a better method is available.

3.2 Control Philosophy

The control philosophy is the method by which the manual inputs or supervisory language commands are executed. The control philosophy encompasses every control action from higher level processing to the simplest rudimentary position control. At times it is difficult to distinguish the features of the control philosophy from the features of the other categories —— equipment, interfaces and languages. This is easily understood considering that the equipment, interfaces and language are created to control the manipulator, and hence, cannot be independent of the methods in which it is desired to control it.

There are four state variables which can be directly measured and used to control the system: (1) position/orientation, (2) linear/angular velocity, (3) linear/angular acceleration, and (4) force/torque (Only the linear terms will be used throughout this report with the implicit understanding that these terms will represent both linear and angular terms). State variables used to control the system which cannot be directly measured are kinetic energy, potential energy, and power consumption. State variables can be classified according to whether they are related to the "internal" state or the "external" state of the manipulator.[20] For example, an internal force state would be given by the joint torques, whereas, the external force state would be given by the force applied to the end effector (The external variables are usually referenced to the end effector for convenience).

The control philosophy can be divided into four categories which determine how the state variables are controlled: (1) manual vs computer control; (2) control strategy; (3) control algorithm; and (4) loop closure. Within each of the above categories there are a number of subcategories or implementations. All of the subcategories can be included in a system, but at any one instant only one subcategory can be used (i.e., the subcategories are mutually exclusive at any instant of time).

(a) Control-Manual vs. Computer

The first control philosophy category relates to the degree of manual and computer control to be used by the system. The types of control available are not simply manual or computer, but any combination of the two. Sheridan and Verplank[4] have discussed this in terms of how much of the task-load is carried by each. Figure 3-2 is a modification of their original figure. The figure demonstrates the possible permeations of manual and computer control, from a purely manual control mode with the operator carrying the entire load to a completely autonomous computer mode. For example, forms of shared control are, resolved motion rate control, index control (Chapter IV), or a computer aid which relieves the operator by controlling some of the degrees of freedom. Examples of traded control are emergency takeover by either the human or computer and a supervisory system

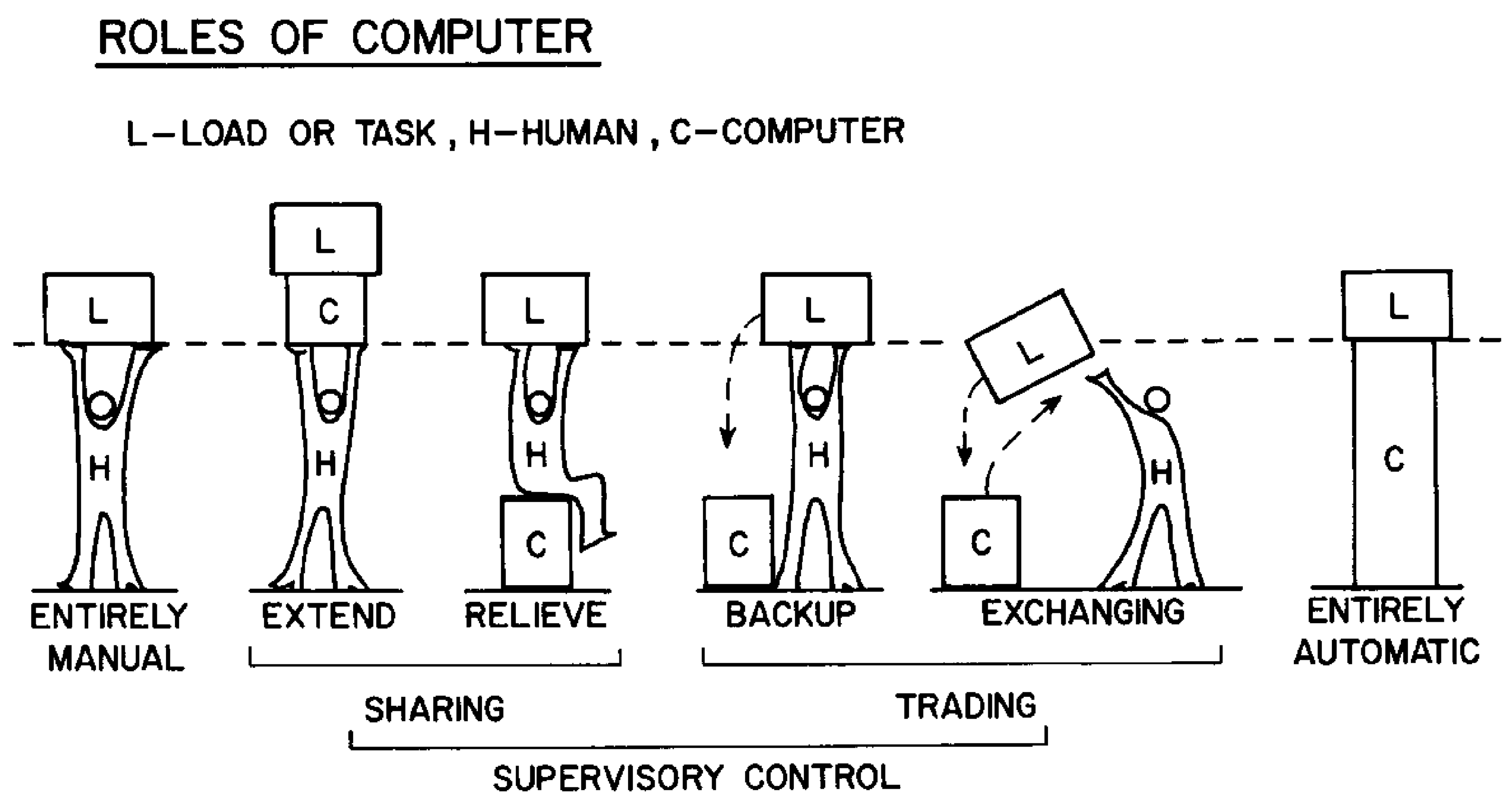


Figure 3-2: Possible Roles of Computer(Modified from Ref. 4)

which takes control for brief periods to perform a task.

(b) Control Strategy

The second category of a control philosophy which can be distinguished is called the control strategy. The following control strategies have been identified: (1) fixed, (2) adaptive*, (3) learning*, and (4) cognitive. These terms, as used here, refer to their conventional meanings and are independent of the intelligence (algorithm) used to achieve these strategies (i.e., to be fixed means to remain constant, to adapt means to modify behavior in response to the environment, to learn means to acquire through experience the ability to discriminate between inputs, and to be cognitive means to be aware of one's world). Before proceeding with explanations and examples of these strategies it is necessary to investigate the significance of the phrase "independent of the intelligence (algorithm) used to achieve these strategies."

Many investigators feel that a machine should only be labelled "adaptive" if it makes a self-adjusting decision based on its environment. Consider a machine that, each time it retrieves a tool, remembers the position, so that it can adjust to changes in the tool rack position. Is the machine adapting to changes in the environment? Consider a relative file (Chapter II) for removing a

*The terms adaptive and learning do not refer to the conventional method by which a servo-control loop modifies its performance to changing electro-mechanical parameters.

a nut which modifies the end effector positions and orientations to the nut. Is the machine adapting to the environment? Neither of these examples makes a "decision", and therefore many investigators relegate these machines to the realm of non-adaptiveness by virtue of their lack of "intelligence." But intuition and common sense says they are adapting. Should biologists consider bacteria to be non-adapting because they don't make decisions? Similar examples can be cited for the remaining strategies. Winston, commenting on machine intelligence, has considered this problem:[21]

> As long as the origin of an idea is obscure; its invention seems profound, but as soon as the explanation surfaces, we wonder, "Why didn't I think of that, its trivial!" As soon as a process is dissected, studied and grasped, the intelligence invariably seems to vanish.
>
> Much the same happens when programs are studied. Vintage performance becomes vin ordinaire once details are exposed and limitations seen. Instead of embracing a system's intelligence, study dilutes it.

This thesis maintains that regardless of whether an algorithm with a "decision process" or "artificial intelligence" can be discerned, if the device "appears" to adapt, learn, or think it will be attributed these qualities.

A fixed strategy can be recognized by a sequence of control commands that maintain a constant relationship or change in a predictable manner as a direct result of a measured state variable. For example, under computer control the commands generated by a fixed-strategy control language would always be executed in a predictable sequence. Even if a branching command is enabled as a result of a measured state variable the outcome is predictable (e.g., GO TO A UNLESS FORCE=10 THEN GO TO B). Under manual control a fixed strategy would be typified by resolved motion rate control which always responds in a predetermined manner for a given input.

An adaptive strategy modifies its behavior in response to the environment by a predetermined method, but, in contrast to a fixed strategy, the result is not predictable to any degree of certainty. For example, compare the fixed strategy given above which follows a predetermined sequence and has a predictable outcome (i.e., either A or B), to the adaptive tool-retrieval example cited previously. Although the method of adapting is predetermined (i.e., record a new position each time tool is retrieved) the result is not (i.e., the new tool position has an infinite number of possibilities). Whenever a manual control mode is used the human operator is usually the adaptive element. But, imagine a strategy that maintains the orientation of the end effector (e.g., to prevent the spillage of a liquid) while allowing the human operator to manually control the spatial degrees of freedom. Is this an adaptive strategy? Although it may appear to be an adaptive strategy, the response is predictable in the same manner as resolved motion rate control, and hence, the strategy is fixed.

As an example of adaptive manual control, consider a task and vehicle which move relative to one another in a random manner.

Through the use of the orientation manipulator (Chapter II) the manual control inputs can be superimposed on the slave arm allowing the operator to perform the task manually while the end effector is continuously adapted to the random task movement.

A learning strategy is similar to an adaptive strategy in that both adapt to the situation, but unlike the adaptive strategy the learning strategy, through experience, acquires the ability to discriminate inputs which previously it could not discriminate. For example, a learning system might watch the operator perform a task until it is sure (from its observation of successes and failures) it understands the task, at which point it progressively relieves the operator as its confidence level increases.[22]

A cognitive strategy involves an "awareness" through a detailed model of the world upon which the algorithm makes decisions (Remember that these decisions can be made by a "dumb" or "intelligent" algorithm, but the strategy will be labelled "cognitive" if the system demonstrates these attributes). This form of control has been called "world-modelling", by some investigators.[4,23,24] The world-modelling in supervisory systems is usually done by the human operator.

(c) Control Algorithm

The control algorithm is the method by which the control strategy is accomplished. Mathematical, logical, statistical, or pseudo-biological procedures can be used by the algorithm to manipulate the state variables.

Control algorithms can be classified as either mechanistic or intelligent. Hence, it is possible to control an arm in either a manual or computer mode using a fixed, adaptive,learning, or cognitive strategy which demonstrates mechanistic or intelligent properties. The "self-adjusting decision based adaption" mentioned in Section 3.2b, is actually an intelligent-adaptive system, whereas the tool-retrieval example is classified as a mechanistic-adaptive system.

The number of available algorithms is so large that only a few will be listed as an indication of the possibilities:[13,22,23,24,25, 26,27,28,29,30,31,32,33,et.al.]

Fixed Strategies

1) Terminal Point Control - the operator specifies the final position of the end effector and the control algorithm determines the trajectory.

2) Path Control - the operator, or processor, specifies path constraints which the control algorithm uses to determine the trajectory. Can be mechanistic or intelligent.

3) Resolved Direction Control - the operator specifies a displacement along a coordinate fixed in the end effector and the algorithm determines the joint actuations necessary to perform that movement.

4) Resolved Rate Control - the operator specifies a rate along a coordinate fixed in the end effector or vehicle and the algorithm determines the joint velocities required for the desired movement.

5) Force Control - the operator, or processor, specifies a desired end effector force vector and the algorithm determines the required joint torques.

Adaptive, Learning, and Cognitive Strategies

1) Simple Model Control - a simple model is continuously updated by the algorithm to adapt to environment changes (e.g., adaptive tool - retrieval). This is a mechanistic algorithm.

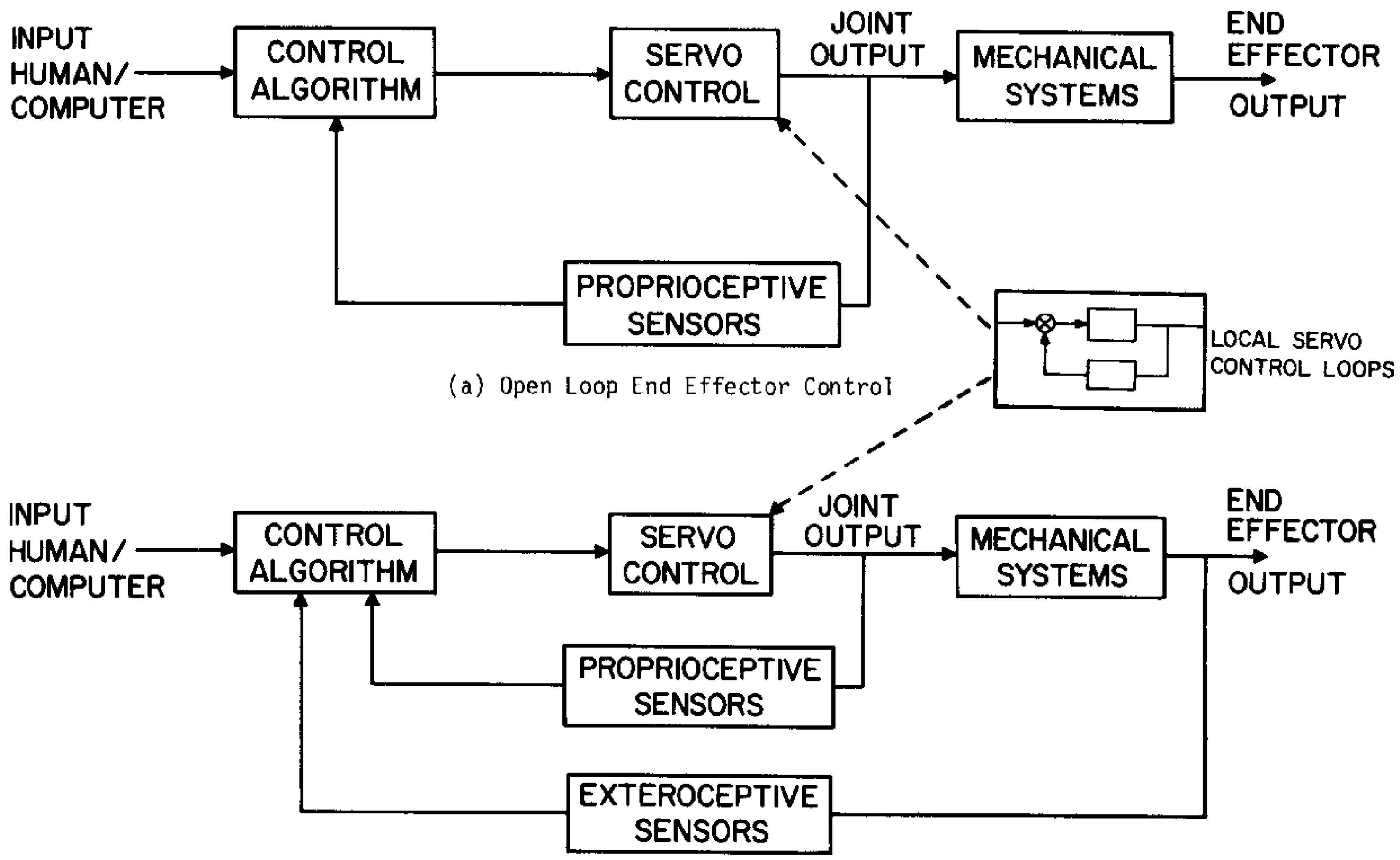
2) Decision Model Control - a statistical algorithm (e.g., maximum likelihood, correlation, etc.) is used to make control decisions. This is an intelligent algorithm.

3) World-Model Control - a detailed model of the manipulator environment is used by a cognitive algorithm for decision and control purposes.

4) Psuedo-Biological Control - an algorithm modelled on theories of biological mechanisms (e.g., the cerebellum).

(d) Loop Closure

The final philosophy category is related to the state variable feedback which the algorithm receives. The state variable feedback determines whether an algorithm is open or closed loop. A generalized block diagram of an open and closed loop system is shown in Figure 3-3 (Note that the local servo control loops that exist in any sizable machine are excluded from this generalization). Control is considered to be open loop when the external state variables (i.e., end effector output) are not directly fed back to the control algorithm. Some manipulators which are considered to be closed loop mechanisms by virtue of internal state feedback (i.e., joint feedback) are actually open loop devices. The loop is not considered to be closed because of the effects of backlash, gravity droop, static link/actuator springiness, dynamic link/actuator wind-up,etc.



(a) Open Loop End Effector Control

(b) Closed Loop End Effector Control

Figure 3-3

A loop is considered to be closed when the actual end effector state variables are fed back to the controller. The external state variables which are fed back to the algorithm to close the loop are obtained by exteroceptive sensors (both exteroceptive and proprioceptive sensors are discussed in Section 3.3).

To conclude this section, a control philosophy has been found to consist of four design categories, human vs. computer control, control strategy, control algorithm, and loop closure. To build a supervisory system, the control philosophy should be specified prior to the design and selection of the equipment, interfaces, and language. This is easily understood considering that the equipment, interfaces and language are created to control the manipulators, and therefore, cannot be independent of the methods by which it is controlled.

3.3 Interface Design

The interfaces which must be considered when building a supervisory system are the man-machine interface and the manipulator-environment interface. Each of these will be considered separately.

(a) Man-Machine Interface

There are two boundaries across which man and machine must communicate — the control interface (man output/machine input) and the display interface (man input/machine output). The control interface consists of any device by which the human operator communicates with the manipulator system. Conversely, the display interface is any device by which the manipulator system communicates with the human operator. The control and display interfaces are the means by which communication occurs, not the method (i.e., language). It is well known that man acts as though he is a single-channel signal detector and processor at a given instant.[34] Therefore, to effectively communicate across these boundaries requires an efficient use of the limited human communication channels.

Some of the variables which have been identified as important criteria for man-machine interface design are:[4,35,36]

1) symbolic/analogic communication
2) apparent/transparent communication
3) stimulus-response compatibility
4) communication channel redundancy
5) consistent format and configuration
6) dedicated or generalized devices
7) ease of use
8) interface compactness

The last two design categories (ease of use and interface compactness) are self-explanatory, and hence, no further discussion is necessary. The remaining factors will be examined on the following pages.

All man-machine communications can be divided into two forms — symbolic and analogic. Symbolic communication has an abstract, coded meaning since it implies communication through symbolic association rather than direct physical analogy. A set of alpha-numeric characters on a screen, a tone signaling an emergency, keys pressed in a particular sequence, and a lit ready-bulb represent man-machine communication through symbolic association. Conversely, analogic communication has a distinctly physical meaning since it represents communication through direct spatiotemporal analogy rather than abstract association. A video monitor showing the manipulator movement, a bar graph of applied force, a tone which varies with pressure, a joystick, a poteniometer for speed adjustment, and a master/slave manipulator represent man-machine communication through direct analogy. Verplank[37] has suggested that an optimal computer—controlled manipulator would use a combination of both symbolic and analogic communication modes. It is still too early to specify the degree of symbolic and analogic communication which should be designed into a supervisory system, but as more functions are relegated to the computer the trend will probably be toward the symbolic end of the spectrum (See Section 3.4).

As shown (Section 3.2), combined manual and computer control of a teleoperator can be classified as either shared or traded control. Sheridan and Verplank[4] have proposed that an important design factor is indicated by the degree of interface transparency (apparent versus transparent) during shared and traded control. The term "transparent" signifies that the operator is unaware of the interface and may actually mentally project himself into the task environment. The term "apparent" indicates that the interface literally forces awareness of itself on the operator and is clearly recognized. A stereo camera which is servoed to the operator's movements allowing the operator to feel as though he were in the remote environment is an example of a transparent display. An example of an apparent display would be an audible signal indicating an emergency. Transparent control is exemplified by a master/slave manipulator with force feedback through which the operator identifies with the remote environment. A sudden jolt or jerk in the master arm signaling return from computer control is an example of an apparent control interface. These examples suggest a design philosophy which can be used to determine when an interface should be transparent or apparent, that is; (1) if control is traded the interface should be apparent, (2) if control is shared the interface should be transparent, (3) if the display requires immediate attention the interface should be apparent, and (4) if the display requires continuous attention the interface should be transparent.

Another important interface design factor is called stimulus-response compatibility. Simply stated, the communication signal, control or display, should have a "natural correspondence" with the operator responses requested or given. If the stimulus and response do not have the required "natural correspondence", the operator will become confused. A form of stimulus-response <u>in</u>-compatibility occurs, for example, when the human operator's internal model of an anticipated hand motion does not coincide with the actual hand motion. This form of stimulus-response _in_compatibility is sometimes referred to as cross coupling (The term "cross coupling" implies an undesirable relationship between the degrees of freedom, that is, motion in the anticipated hand coordinate direction results in a motion in an unexpected direction. See Appendix A). An analog bar graph which decreases as the applied force increases is an example of a display interface with stimulus-response _in_compatibility. Use of analogic controls does not guarantee that stimulus-response compatibility will occur, and conversely, use of symbolic controls does not automatically mean that stimulus-response _in_compatibility will occur. Since stimulus-response compatibility has been extensively treated in the literature [see ref. 4,38,39,et.al.], specific design rules will not be given,except to note that stimulus-response _in_compatibility is detrimental to operator performance and should be removed whenever possible to avoid confusion and errors.

Man-machine communication can also be improved through the use of redundant channels. Redundancy increases the probability that a signal will be received or acknowledged. For example, when control is traded between man and machine it is important that the transfer be apparent to the operator, and hence, redundant visual, auditory, and tactile clues will insure operator recognition of the traded status. It should be realized that these are independent communition channels saying the same thing, and not simply one channel which is displayed or controlled in a redundant manner. As another example of redundant displays, consider the methods by which the machine can signal recognition of a command. It can visually echo the command, it can audibly acknowledge receipt, and it can tactually signal completion. Hence, when a button is pressed a tactile click; an audible tone, and a visual message could all be simultaneously used to enhance the chances of operator recognition. Clearly, whenever possible, redundancy should be used to increase the probability of communication reception or acknowledgement.

Consistency of the control and display format is extremely important for effective man-machine interaction. Symbolic and analogic display and control formats should be structured for easy reference and operator confidence. For example, alpha-numeric feedback should continuously display operational information, such as state, control mode, indicators, etc., in a uniform manner so that the operator can obtain the desired information quickly. Displays and controls should also be arranged in a "tight" pattern within easy reach and view of the operator, but should not obstruct or interfere with each other. The man-machine interfaces should be grouped according to usage with the more heavily used controls and displays given preferential location. Consideration should be given to the amount of attention the user will be required to allocate to each interface. A well known "candy store" phenomena occurs when a child has so many choices that demand his attention that he cannot make-up his mind. Bejczy has an interesting method of defeating the "candy store" effect —— only one display interface with a menu of options is used. Through voice commands the operator can select the display of his choice.

The question of whether display and control interfaces should be dedicated or generalized devices has remained unanswered for many years. For example, although a general purpose keyboard is flexible, a dedicated keyboard is often more efficient (see Language Philosophy Section). Which is more important, generality or efficiency? An interesting solution to this problem has been achieved by using a virtual image to label the keys, giving the dedicated keyboard the flexibility of a general purpose keyboard.[35] As another example, consider an analog control interface —— should a generalized position controller which is manipulator independent be used, or a dedicated master?

To conclude, the man-machine interface involves two boundaries— the control and display interfaces. Although these two boundaries are distinctly different (input versus output) there are eight common design considerations which must be evaluated to achieve effective communication through the limited human channels available at each boundary.

(b) Manipulator-Environment Interface

The manipulator-environment interface is responsible for sensing the internal and external states of the manipulator. Sensing of the internal manipulator state is called proprioception. Sensing of the external state of the manipulator is called exteroception.[20]

The term proprioception is defined as being aware of stimuli produced within oneself. Proprioceptive sensing of the internal manipulator state usually concerns geometric/kinematic properties (i.e., sensors for measuring the positions,velocities,accelerations and torques of the joints). Proprioceptive feedback is open loop information with respect to the end effector (see Figure 3-3), since the end-effector output is neither directly measured nor fed back for comparison with the commanded value. Clearly, the accuracy and precision will depend heavily on the calibration and repeatability of the system. For example, elastic arms with large masses toward the end-effector will be uncontrollable.[17] Many of the important manipulator selection factors (e.g., backlash, link/joint mass and stiffness) clearly result from systems which use only proprioceptive information for control.

The term exteroception is defined as being aware of stimuli produced outside of oneself. Exteroceptive feedback is commonly provided by proximity, tactile, texture, slippage, force/torque, and spatiotemporal optical sensors (i.e., digitized cameras). In contrast to proprioceptive feedback, exteroceptive feedback closes the loop directly around the end effector. Since the end effector output is compared to the commanded value, exteroceptive sensors reduce the necessity for a highly calibrated and repeatable system (e.g., if the end-effector is approximately on target, precision align with exteroceptive sensors). Some of the uses of exteroceptive information are[40]: (1) correcting position errors, (2) constrained motion, (e.g. sliding along a surface), (3) error detections (e.g. collisions), (4) training (e.g., a learning system that is taught how a block feels), and (5) classification of objects (e.g., size, weight, etc.). It should be noted that exteroceptive sensors should not be considered as a substitute for good manipulator design — proprioceptive and exteroceptive sensors should be used to complement one another.

In conclusion, the responsibility of the manipulator-environment interface is to measure the internal and external state of the arm. The design of proprioceptive and exteroceptive sensors should be based on reliability, accuracy, precision, stability, and repeatability of the end effector.

3.4 Language Philosophies

According to a classification described by Park[23], manipulator systems can be divided into two categories —explicit and world-modelling. Explicit systems assume "someone" (man or machine) will be there to teach the machine what to do in a step-by-step manner. World-modelling systems, on the other hand, attempt to give the computer an internal "picture" of its environment upon which it can act and plan the required step-by-step instructions (i.e., artificial intelligence). For example, the human operator using a world-modelling system would say, "open the valve" leaving the details to the computer, whereas the human operator using an explicit system would describe "how" to open the valve.

Although these two categories appear to be distinct, it is possible to conceive of a system which, after being explicitly programmed to perform a task, can adapt to new environments by modifying its instruction set on the basis of an internal world-model. Would this be an explicit or world-modelling system? Clearly, this classification deteriorates as explicit systems with world-modelling capabilities proliferate. In fact, Park indicates that the boundary separating the two categories is no longer distinct:

> "The two kinds of systems are becoming more alike. Recent world-modelling programs permit the user to decide on the tactics to be employed, and they can also deal with more uncertainty in their world models. Explicitly-programmed robots are beginning to make some strategic decisions for the user, such as planning pick-and-place motions to avoid obstacles."[23]

There is also a tendency in the literature to label a system as "world-modelling" (implying a high degree of planning and artificial intelligence) when in reality it is an adaptive or learning strategy which simply adjusts an algorithm. (This is not meant to imply that all adaptive and learning strategies are devoid of artifical intelligence, but a system which records the operator's movements over a period of time and takes control upon recognition of a pattern (i.e., the operator did the same thing twice) is little more than a mechanistic record/ playback strategy).

Clearly, it is necessary to devise a new or modified classification system which will remove the ambiguities and vague usage of the term "world-modelling". Before proceeding with this classification, note that the "world-model" is used for two distinct purposes---planning and execution. The confusion results from the fact that planning is a function of the task description (language philosophy) and execution is a function of manipulator control (control philosophy). Hence, it is possible to have an apparent contradiction by defining a task "explicitly" and executing it on a "world-model" basis. The term "explicit" refers to the language used for task description---not the system, and the term "world-modeller" refers to the method of execution---not the language.

It is suggested, therefore, that the term "world-modelling" be used as an adjective to define a <u>system</u> which bases its decisions on a model of its world (task). Whenever speaking of the <u>language</u> by which the system is programmed the terms "explicit" and "implicit" will be used. As before, the explicit language assumes either man or a world-modelling machine will specify the commands necessary to perform the task. The implicit language, though, assumes each command is a generalized instruction which should be translated by the machine into step-by-step instructions. Notice that the term "world-model" was not used to define an implicit language, since it is possible for a system to develop a set of instructions strictly on the basis of a "dumb" algorithm (e.g., a cross-assembler which simply translates higher level language commands to lower level commands). Since the world-model is defined as a system parameter (i.e., data base), it should be clear that both the explicit and implicit languages can now use world-models without ambiguity in definition.

The explicit and implicit languages have the following attributes:

<u>EXPLICIT LANGUAGE</u>

1) Human operator responsible for planning and program order
2) Instructions are detailed and specific
3) Complex tasks require many instructions

<u>IMPLICIT LANGUAGE</u>

1) Robot responsible for planning and program order
2) Instructions are simple and abstract
3) Complex tasks can be described in a few instructions

As a method of comparing languages, Grossman and Taylor[41], have suggested that, "the level of manipulator languages is best measured not by the richness by their computer science content, but rather by the number of source statements required to code specific applications programs". The value of this method as a means of classifying languages which range from simple explicit commands to abstract implicit commands has not been proven. But this method could at least provide a hypothetical indication of the time required of the human operator to define the task. Since supervisory manipulation requires real-time interaction, the time the operator spends programming directly effects the task completion time (see Figure

5-1 which shows hypothetical task completion times as a function of task complexity). Hence, one design goal for a supervisory language should be to allow efficient communication between man and machine (it should be remembered that this discussion deals only with the method of communication and not the means, i.e., language and not interfaces). This design criterion appears to indicate that an implicit language is the better choice. But if the computer planning time is greater than that for a human operator, the advantage of an implicit language would be lost.

Explicit and implicit languages can be further broken down into two categories of code——grammatical/syntactic code versus program code (e.g., "I want you to get me the scalpel now" versus "scalpel" often used by doctors in an operating room). A programming code is a highly specialized language which must be learned with all its idiosyncrasies before command entry can begin, while a grammatical/syntactic language strives for a natural conversation between machine and operator with the ultimate goal of simple, active man-machine communication. Unfortunately, "natural" conversations tend to be extremely awkward methods of code entry when input must be entered through a keyboard (Indeed, even verbal commands tend to be slow for specialized conditions such as an operating room). Clearly, a programming code has a higher information density for each operator input compared to the grammatical/syntactic code. For example, an explicit verb-noun-parameter-terminator language used by Perceptronics[36] requires the following input commands to define a point which the manipulator will be required to return to:

```
        DEFINE          POINT           7           DO
```

The same commands could be written as one instruction by allowing the computer to assume that one keystroke means "define this point, call it a sequential name, and terminate the stroke automatically if the required information has been obtained" (see DPATH in the DEFINE section). Clearly the advantages of grammatical/syntactic code can be overshadowed by the greater number of keystrokes or verbal commands required for data entry.

Many designers believe that if the operator must adapt his behavior to the demands of the machine, optimal communication can not be achieved. But a long list of "natural" situations can be cited where the human language is deliberately constrained for precise communication. For example, doctor-nurse communication in the operating room, pilot-air traffic controller communication, pilot-gunner communication (e.g., why say, "There's a silver mirage MIG fighter coming over our starboard wing at 700 mph firing...", instead of "bandit at 3 o'clock high"?), etc.

Regardless of whether the language (explicit or implicit) is natural or structured, there are basic components which are common to all. In collaboration with Verplank the fundamental elements of manipulator language have been identified:

VARIABLES - A quantity of data identified by a symbolic name. There are two types of data quantities associated with manipulator languages—state and program variables. The appropriate state variables have been identified previously (Section 3.3). Examples of program variables are counters, flags, etc.

DECLARATION STATEMENTS - Non-executable statements (i.e., they do not perform an action or operation) which simply specify a given condition. For example, a declaration could state that the entire command string should be interpreted in joint coordinates versus, say, vehicle coordinates.

ASSIGNMENT COMMANDS - Replaces the current values of a variable with the quantity specified by the command. Assignment commands can directly assign a value, call for input from sensor readings, or request symbolic/analogic input from the operator.

ACTION COMMANDS - Primitive manipulator commands that request a physical response from the arm. These commands control the state variables (e.g., position, angular velocity, force, etc.). The requested action can be either absolute or relative to the current state and expressed in any state space (e.g., joint coordinates, hand coordinates, vehicle coordinates, etc.).

OPERATION COMMANDS - Commands used to modify, transform, and manipulate both the state and program variables. As examples, an operation can request a transformation from one coordinate frame to another, an addition of two variables, etc.

FLOW-CONTROL COMMANDS - Higher level language commands that regulate the direction of the program based on tests and branching. Flow-control commands allow the user or implicit compilier to exercise external control over the sequence of execution. Tests are either performed on the state variables (position, force, etc.) or the program variables (counters, flags, etc.). For example, program flow could be redirected if a touch sensor is activated or a force encountered.

COMMUNICATION COMMANDS - Commands used to request or deliver some form of man-machine interaction. An example, would be a command, that when executed, displays a message or outputs an audible signal.

SUBROUTINE CALLS - Sequences of programming elements which have been identified by a symbolic name so that the sequence can be executed by one reference (Note - A subroutine call is not an implicit language as a translation is not being performed). Subroutine calls are important because they allow a complicated procedure to be executed as a sequence of less complicated tasks (subroutines).

Several of these fundamental elements are similar to other programming languages (FORTRAN, APL, etc.), and, therefore, many of the procedural and design rules which apply to computer science are also applicable here. For example, individually named variables should be used to allow personal labeling versus fixed names such as "1" through "9", programs should be structured to avoid confusion but not at the cost of execution speed, etc. Many design questions are still actively debated. For example, what is the best method to terminate a command or line——a SEND command, a carriage return, or an automatic termination when the expected information has been obtained?

Generally, a supervisory manipulator language should have the following attributes:[4,24,42,43,44]

1) Easily learned, read, debugged, and used
2) Constrained and standardized code
3) Real-time command modification (editing)
4) Real-time code generation
5) Manipulator independent
6) Easily commented
7) Easily upgraded
8) Application (task) flexibility

In conclusion, it has been shown that there are two methods of generating code for manipulator languages——explicit (human planning) and implicit (machine planning). Although implicit languages appear to be more efficient communication modes, the machine planning time can significantly reduce their effectiveness for real-time supervisory systems. It is too early in the development of supervisory manipulation to define an optimal language, but a compromise between the two which allows the operator to perform the higher functions of planning and world-modelling, letting the computer do the simpler planning and world-modelling, would probably result in an optimal system. It is possible, though, to project the future development of supervisory manipulation as a gradual change from explicit languages to implicit languages as the computer planning time approaches that of the operator.

Languages can be further divided into constrained and natural communications. Constrained languages are more efficient compared to natural languages but require a complete knowledge of the specific code. Again, it is too early to predict which will result in the optimal system, but it can be assumed that a combination between the two will become more prevalent.

Manipulator languages have basic components which are fundamental——variables, declaration statements, assignment commands, action commands, operation commands, flow-control commands, communication commands, and subroutine calls. Although many procedural and design rules have been determined there are still many questions which remain unanswered.

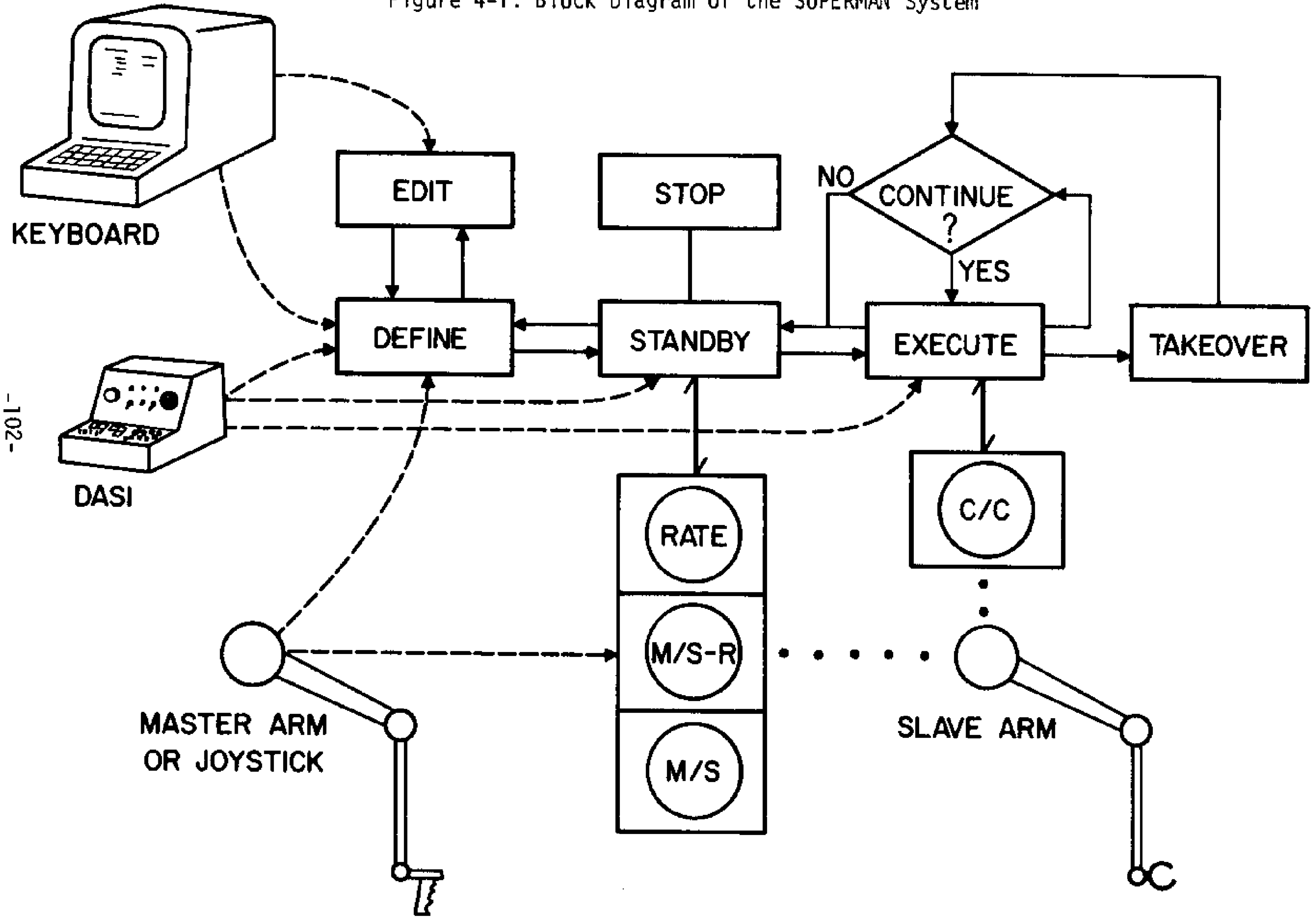
CHAPTER IV

SUPERMAN: A SYSTEM FOR SUPERVISORY MANIPULATION

To investigate supervisory control of a remote manipulator, an experimental system called SUPERMAN was created. The system was built on the theoretical foundation outlined in Chapter 3. Before proceeding with the specific SUPERMAN design details, a summary explanation of the system will be necessary.

Figure 4-1 shows the general relationships between the multiple inputs (keyboard, dedicated symbolic keys, and analogic inputs), the computer states (STANDBY, DEFINE, EDIT, EXECUTE,STOP, and TAKEOVER) and the control modes (RATE, MIXED MASTER/SLAVE AND RATE, MASTER/SLAVE, and COMPUTER control). The solid rectangles in the figure represent computer states with the exception of the rectangles with circles which represent control modes. The man-machine interfaces are represented graphically on the left of the figure (The joystick, TV monitors and switches on the computer interface are not shown). The solid arrows in the figure indicate transitions between states, the dashed arrows represent input signals, the half-arrows indicate control mode communications, and the dottled lines represent output signals. Control normally resides in the STANDBY state. Through this state the operator can enter one of the three primary states —— STOP, DEFINE, or EXECUTE. Secondary states (TAKEOVER and EDIT) can only be entered through one of the primary states.

Figure 4-1: Block Diagram of the SUPERMAN System



With the basic foundation established, the factors which influenced the design of the SUPERMAN system will now be considered.

4.1 SUPERMAN Design Considerations

In Chapter 3 it was stated that there are four design factors which should be considered when building a supervisory system; (a) manipulator/processor selection; (b) control philosophy; (c) interface design; and (d) control language. Each of these factors will be dealt with separately.

(a) Manipulator/Processor Selection

Unfortunately, the selection of the manipulator was not determined by geometric constraints (i.e., solvability and generality) but rather as a matter of availability. The selection of the processor was also predetermined. Therefore, to achieve better performance of the overall system these components were modified.

The SUPERMAN system uses an Argonne National Laboratory E2 master/slave manipulator (Figure 4-2). The modifications to the arm consisted of mechanical and electronic alterations which were the direct result of a change from syncro/resolvers to potentiometers for position feedback. Modifications to the geometry of the existing arm were impractical, and therefore, only minor changes in gearing were done. Details of the modifications made to the arm can be found in Appendix D.

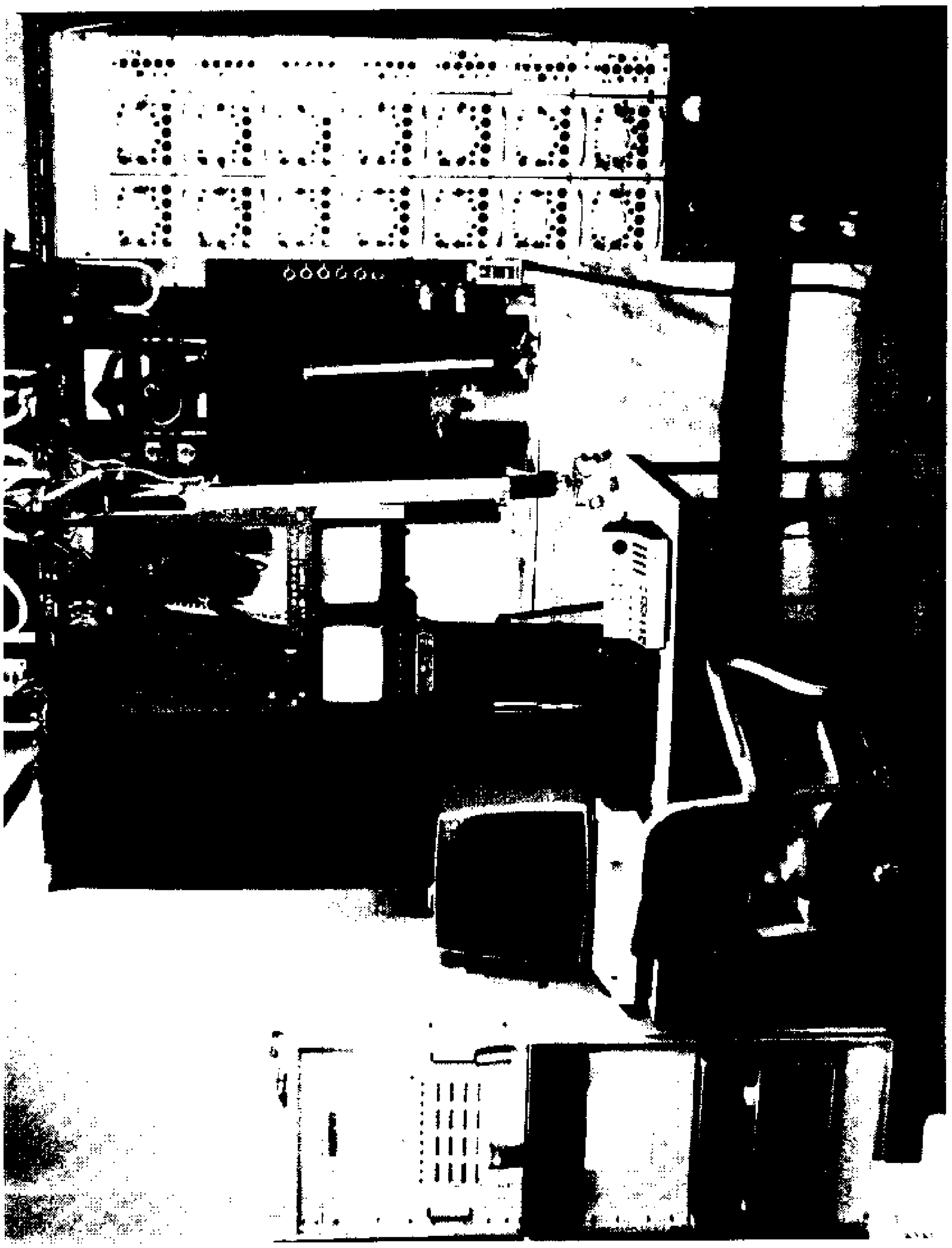


Figure 4-2: Teleoperator Laboratory

The manipulator/computer interface designed for the project has 32 analog to digital inputs, 16 digital to analog outputs, 32 digital inputs and 16 digital outputs. The central processor, affectionately known as Murphy, is an Interdata Model 70 with 64K bytes of memory. The M70 is interfaced with two Diablo disk drives and a modified Imlac vector plotting scope. The processor performs a hardware floating point multiply in 54.0 μ seconds and has a basic register-to-register instruction time of 1.0 μ seconds. Many of the program simplifications would not have been necessary had a faster processor been available. On the basis of these recommendations the Man-Machine Systems Lab is purchasing a PDP-11/34 for future studies. The future SUPERMAN system may even use a form of distributed processing by assigning display and joint driving functions to microprocessors.

(b) Control Philosophy

The control modes which were implemented on the SUPERMAN system are explained in the STANDBY section. It was decided that only the fixed and adaptive control strategies would be used as the learning and cognitive control strategies are beyond the realm of this study. The adaptive strategies consist of a command which records the slave position for use during the next execution and the implementation of the relative technique discussed in Chapter 2.

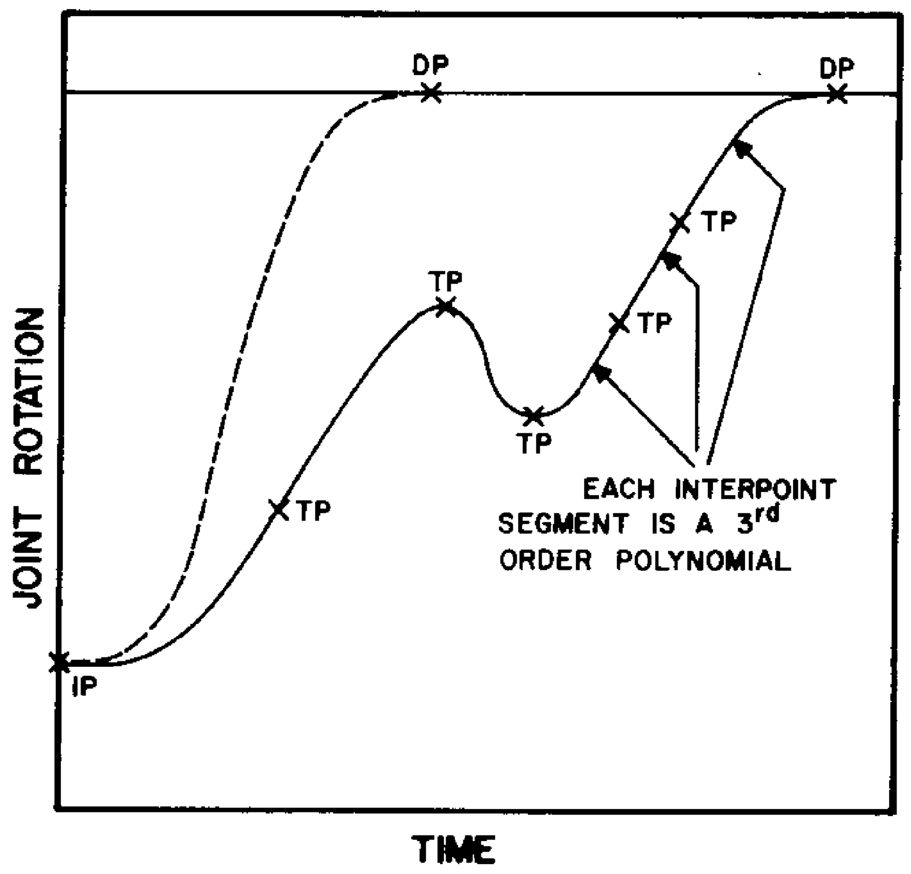
The SUPERMAN system presently uses proprioceptive sensors for position, velocity,and torque. Although the system does not use any exteroceptive sensors, it performs many functions which would normally be done through exteroceptive information; the difference being a matter of convenience and accuracy. For example, the end effector force vector can be approximated through proprioceptive joint torques.

The complexity of many of the control algorithms requires that the details be explained in the sections dealing with the SUPERMAN language. Hence, only the mathematical control algorithms for position/orientation and linear/angular velocity control will be considered in this section.

The three algorithms identified by Bejczy[26], terminal point control, path control and resolved position/rate control, were implemented on the system to drive the slave servos. The first control algorithm, terminal point control, is obtained when the operator specifies an end point through the DPATH command. When executed, the algorithm outputs a third order polynomial between the starting position and the desired end position. The parabolic velocity curve (derivative of the polynomial) is given zero initial and final velocities as boundary conditions. The control algorithm calculates the time required for each joint to reach its final position and then uses the slowest joint time as the total path time for all six degrees-of-freedom. Using the slowest joint time insures that all the joints have sufficient time to reach the final position. Hence, the algorithm chooses an optimal path time to insure that all joints arrive at the desired position at the same time.

The operator has the option of using a path control algorithm through the TPATH (through-path) command. The operator can specify any number of through-points that the manipulator is to traverse on its way to the final point. The maximum velocity that the end effector can have as it moves through the point is calculated from an empirically derived curve. With the initial and through-point velocities known, the algorithm is able to fit a smooth third-order polynomial between the two points in real-time. For the calculation of the next through-point the previous point's velocity becomes the initial velocity for the new curve. This process is repeated until an end point is reached with a final velocity of zero. If the processor does not detect an end-point (DPATH), the computer automatically gives the last through-point in the sequence a zero velocity. The path control algorithm calculates the slowest joint and then drives the joints to arrive at the through-point at the same moment in time. (The equations used for the terminal point algorithm are actually a subset of the path control algorithm since the initial and final velocities are set to zero for the terminal point).

Figure 4-3 shows the joint rotations for a number of through-points (TP) and an end-point (DP) given an initial execution manipulator position (IP). Note that the second through-point in the path has a zero velocity because of a change in path direction. The dashed line in the figure is the joint rotation as a function of time for the terminal point algorithm.



TP-THROUGH POINT

DP-END POINT

IP-INITIAL MANIPULATOR POSITION

Figure 4-3: Illustration of the Terminal and Path Control Algorithms

Resolved position/rate control algorithms are employed in many of the manual control modes (time delay, resolved rate control, indexed manual control, etc.). See the STANDBY section for further details of the position and rate control algorithms.

(c) Interface Design

The man-machine interfaces were designed to use multiple communication channels for redundancy and effective interaction. Whenever possible, tactile, visual and audio signals are returned to the operator to insure detection of the action. Inputs to the SUPERMAN system include an ASCII keyboard, a dedicated analog-symbolic interface (DASI), a three axis spring-centered joystick, and a six degree-of-freedom replica master manipulator. Outputs from the system include audible warning tones, graphic and alpha-numeric visual displays, lights for binary on/off information, force feedback, and meters for joint torque levels.

To communicate with the manipulator a dedicated analog-symbolic interface (DASI) was created with efficient man-machine interaction as the design criterion. The DASI keyboard is shown in Figures 4-4 and 4-5. The buttons numbered 0 through 15 on the right of the control box (Figure 4-5) are used to define computer commands. Each of the round DEFINE state buttons on the interface responds with a tactile "click" to give the operator positive feedback that the command has been entered. For feedback redundancy all DASI buttons alert the operator with a short 50 ms "beep" from a sonalert tone

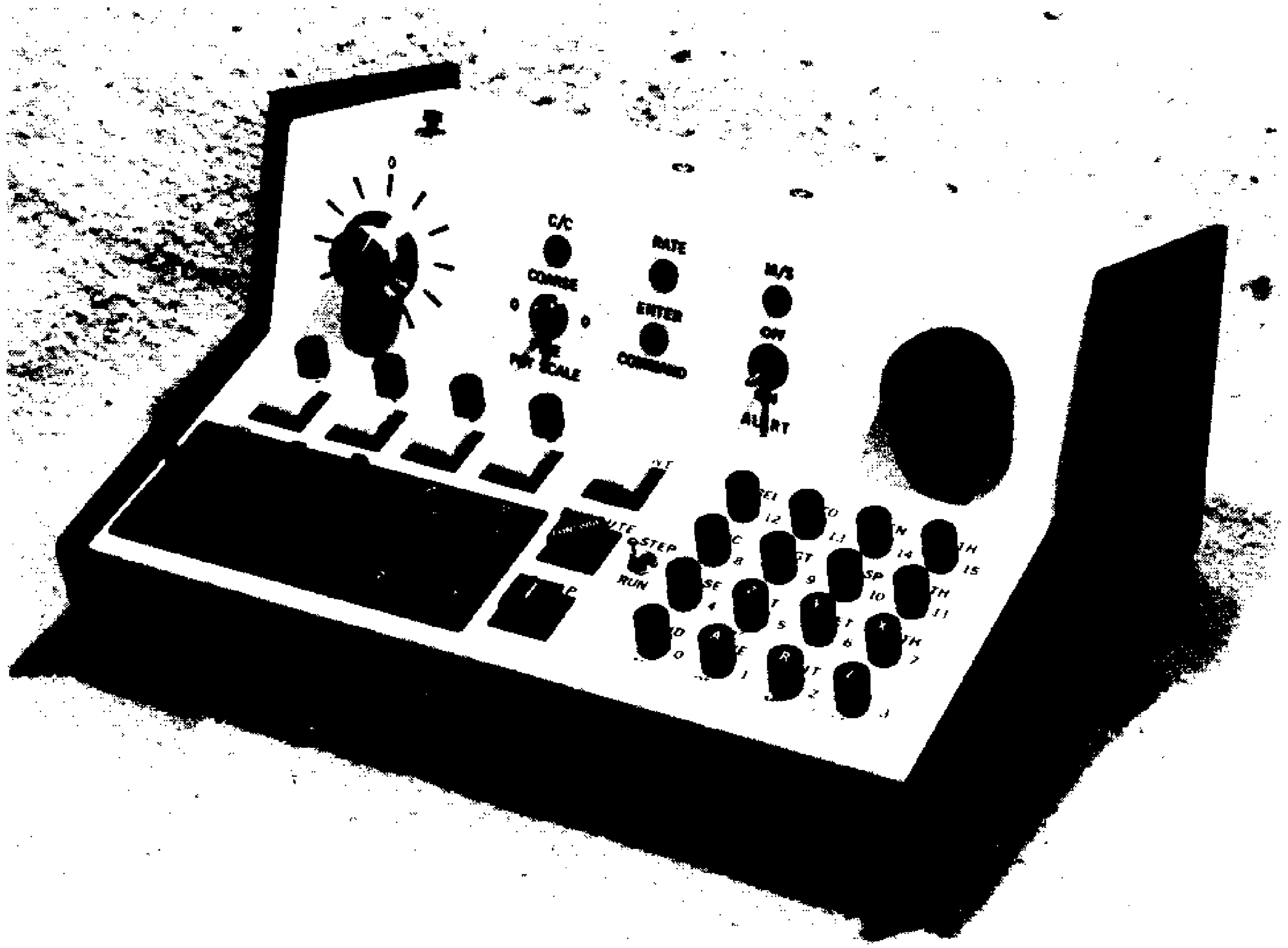


Figure 4-4: DASI (Dedicated Analog-Symbolic Interface)

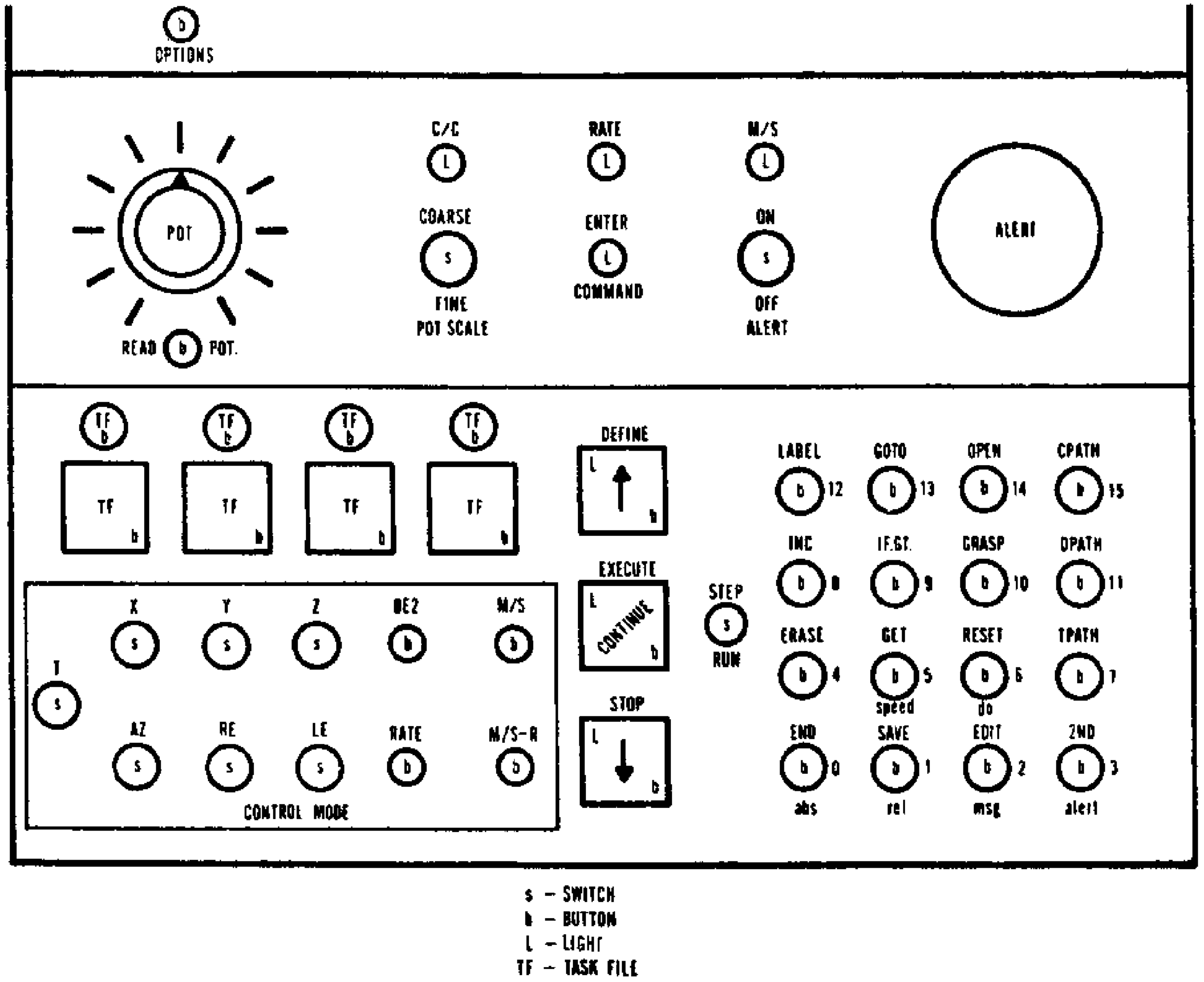


Figure 4-5: Diagram of DASI Functions

generator and when appropriate, the commanded action is echoed on the monitor screen. The buttons are grouped according to computer states and control modes (The grouping of commands will be discussed further in the individual sections on each state). A potentiometer with fine and coarse adjustments is also incorporated into the DASI keyboard for analog input. The potentiometer is used for data entry in the DEFINE state, as a scaling factor for rate control modes, and to adjust the arm speed during execution.

A piezoelectric transducer (sonalert) is used to give audible warning signals with a sound intensity of 50 to 80 db. The sonalert is used to signal (1) manipulator collisions with the environment, (2) manual takeover with mismatch, (3) imminent movement of the master arm, (4) execution of a relative task which exceeds the physical limitations of the arm geometry, as well as (5) keystroke entry.

The screen of the Imlac vector scope is used for display of (1) the control mode which is in effect or temporarily suspended, (2) the computer state, (3) the listing of the file being executed, (4) the task file button assignments, (5) position, velocity and force information, and (6) operator cues (see Figure 4-6). Lights on the DASI control box are used to indicate control modes, computer states, and operator cues. Visual torque feedback for each joint is indicated by meters located on each servo-amplifier. Visual spatio-

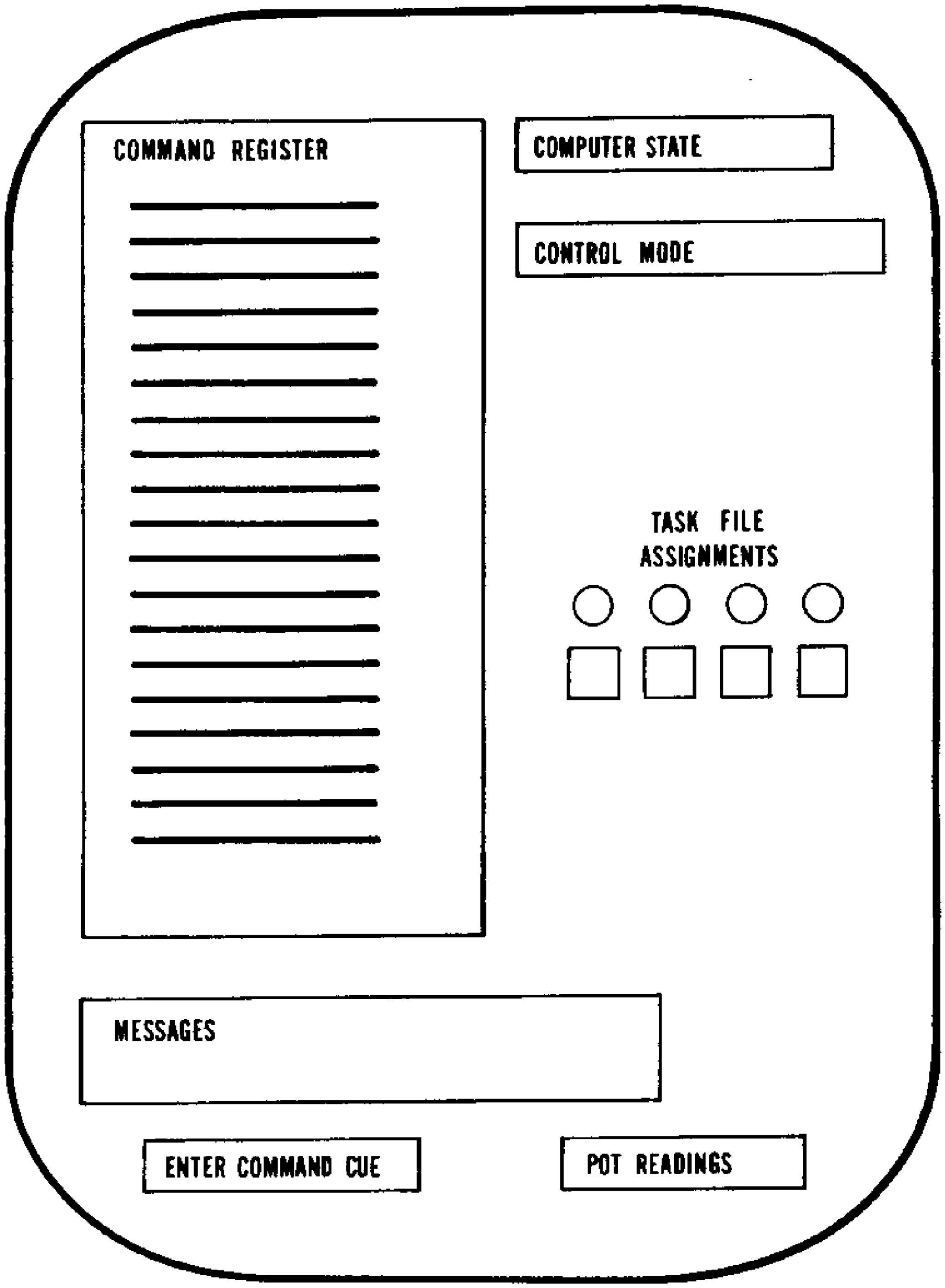


Figure 4-6: Display Format

temporal feedback consists of two video monitors with one fixed camera and one zoom camera with remote pan and tilt controls.

(d) Control Language

On the basis of the previously cited language considerations (Chapter 3), it was decided that the SUPERMAN system would use an explicit language with a constrained programming code. Whenever required, a psuedo-grammatical/syntatic code is used for clear and precise entry (e.g., IF FORCE > XXX, INCREMENT DOF XXX, etc.). The hierarchy of the SUPERMAN code is shown in Figure 4-7. The heavy lines indicate computer states and the light lines represent transitions. The figure shows that there are two methods by which the EXECUTE state can be entered —— the command register or a task file.

The command register is a general purpose file which is executed through the use of the execute button. The command register is an efficient file through which a task can be defined, tested, edited, and finally saved as either a named subroutine or a task file.

A named subroutine is a string of commands which has been saved under a user specified "name". These files can only be executed by (1) retrieving them from the disk and inserting them into the command register in the DEFINE state, or (2) by calling them from the main program during execution.

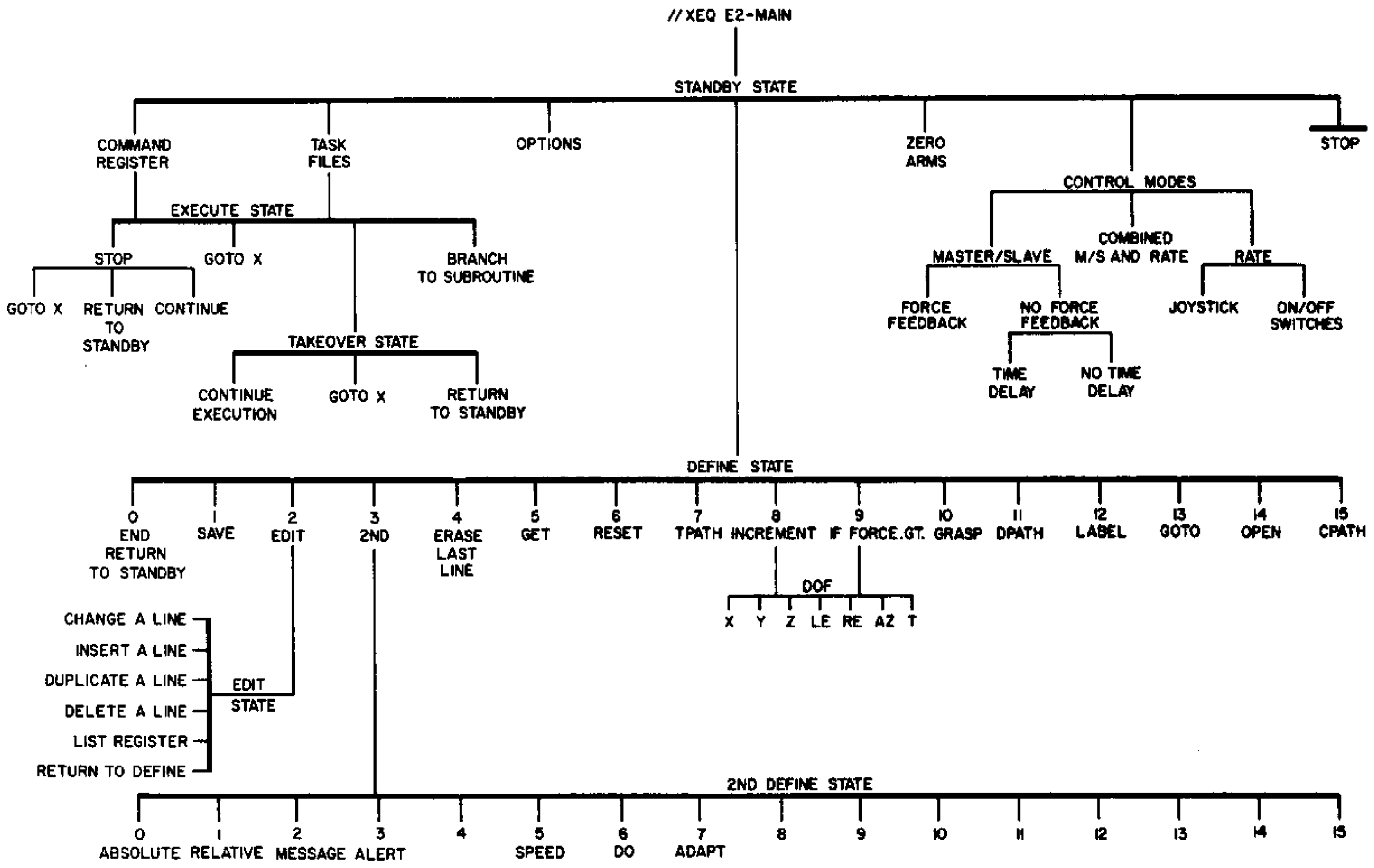


Figure 4-7: SUPERMAN Code Hierarchy

A task file, on the other hand, is a string of commands which has been through the debugging stage and has been saved under a specific button for easy execution. Once a file has been saved as a task file it cannot be edited (The user can change the file by executing it and then entering the DEFINE state, thereby moving the task file commands into the command register).

In the block diagram at the beginning of this chapter (Figure 4-1) it was shown that the SUPERMAN system has six computer states - STANDBY, DEFINE, EDIT, EXECUTE, TAKEOVER, and STOP. The STOP state is self explanatory and no further consideration is needed. The remaining states will be discussed in the following sections.

4.2 STANDBY State

When the computer is in this state, control resides with the main program and the operator. By pressing the proper control console button the user can invoke a control mode, specify a control constraint, execute the command register or a task file, zero the arms, list the options currently available, or transfer to the DEFINE state (Figure 4-8).

The control mode, as the name implies, is the method by which the primitive signals required by the slave arm to perform the desired function (task) are generated. In general, the control mode can be divided into two categories on the basis of whether the primary con-

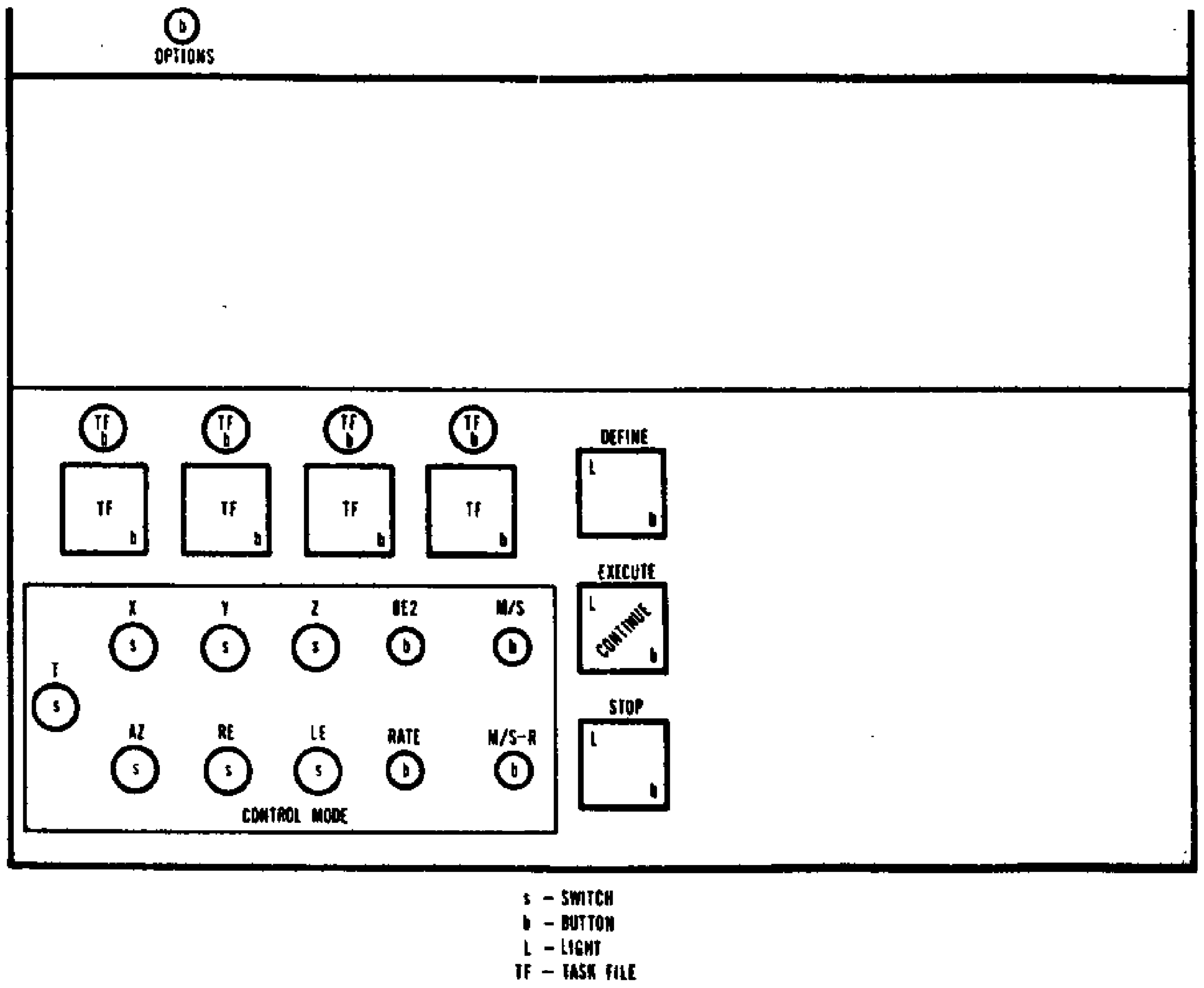


Figure 4-8: Primary DASI Input for STANDBY State

troller of the arm is the human operator or computer.[*] Under computer control the processor has complete control over the slave manipulator. The human operator can only interrupt the computer (see TAKE-OVER or EXECUTE state) or change its goal (see, for example, GOTO in DEFINE state). Under a manual control mode the human operator is the primary signal generator. Manual control modes are generally independent of state (e.g., the control mode might be master/slave while the state is EDIT). Note, though, that during the EXECUTE state the manual control mode is temporarily suspended while the computer control mode is in effect (i.e., during execution the control mode is state dependent). Six methods of manual control have been incorporated into the SUPERMAN system:

1) Switch Rate - Each degree of freedom is rate controlled through a spring-centered on/off switch on the DASI console. The individual rates are adjusted as a percent of the maximum rates by the DASI potentiometer.

2) Mixed Master/Slave and Rate - The master acts as a springloaded joystick in the X, Y, and Z axes, giving rate commands to the X, Y, and Z axes of the slave proportional to the displacement of the master. The rate

[*]The term "primary controller" indicates that in general the human operator is never in complete control of the slave arm in any control mode. For example, in RMRC the human operator is the primary controller, or action giver, but the computer is ultimately interpreting and relaying the commands to the slave. This is a form of shared control between the human operator and computer described by Sheridan and Verplank [4].

of the slave arm is then reflected in the force-feedback level which the operator feels in the master. The remaining degrees of freedom (rotation, elevation, azimuth, and end-effector) are controlled in a master/slave mode. The potentiometer on the control console can be adjusted to set the sensitivity of the joystick.

3) Variable Rate Joystick - A three degree of freedom springloaded joystick (Figure 4-9) outputs rate signals to the X, Y, and Z axes of the slave proportional to the joystick displacement. The X, Y, and Z axes of the master arm are then locked in position, creating another joystick for the remaining rotational degrees of freedom. The potentiometer on the DASI console can be adjusted to set the sensitivity of the joysticks.

4) Master/Slave without Force-Feedback - The slave manipulator duplicates the position of the master, but the master is completely unaware of the slave position. The force the slave exerts on its environment is proportional to the difference in position between the master and slave.

5) Master/Slave with Force-Feedback - Any position error between master and slave applies a driving force to the corresponding motors on both master and slave —— but in opposite directions to nullify the error.

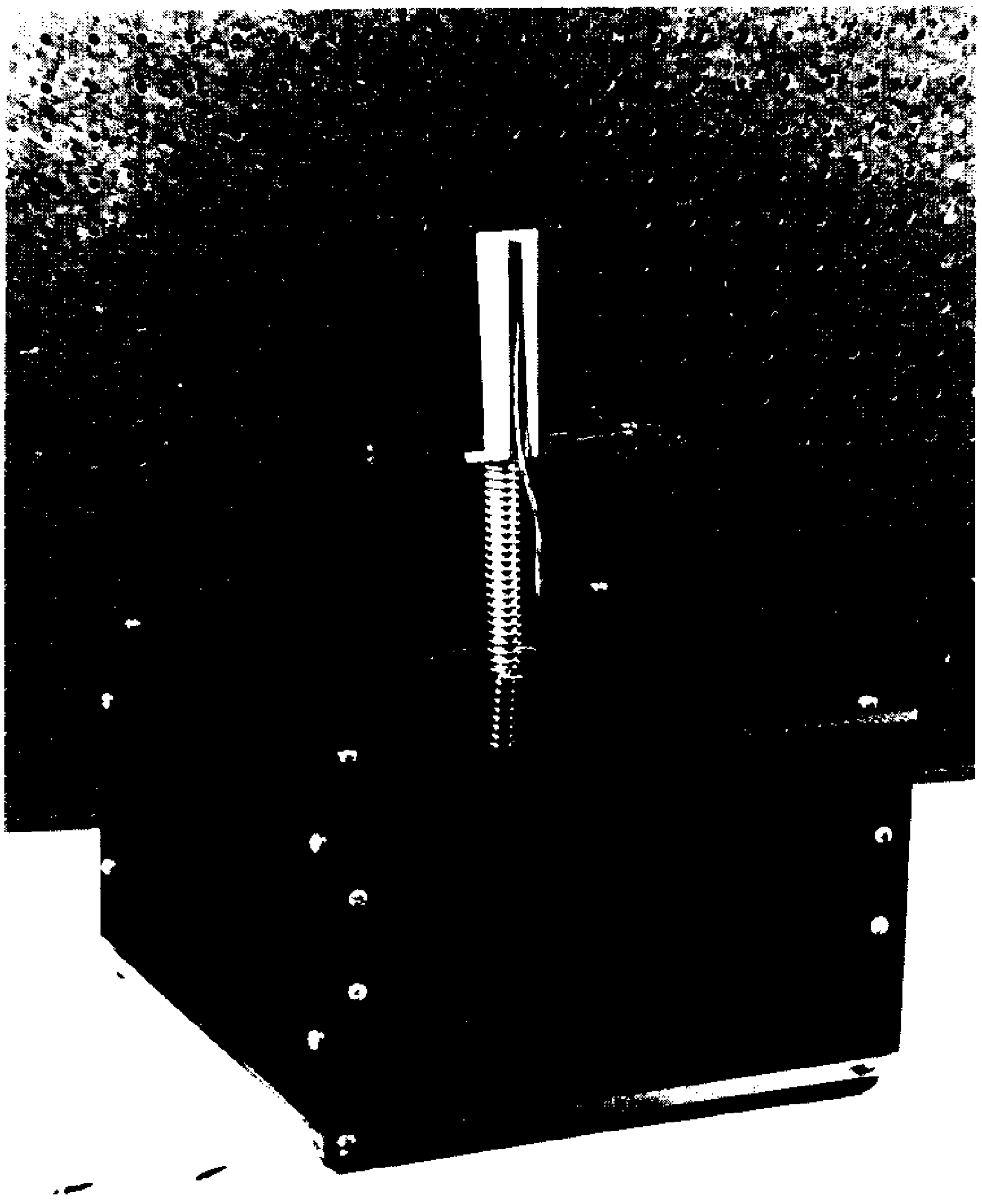


Figure 4-9: Three Degree-of-Freedom Joystick

Thus, any force exerted on the slave is reflected to the master giving the operator the impression of actual contact with the environment. The force exerted by either arm is proportional to the position disparity between the master and slave. Potentiometers on each degree of freedom can be used to adjust the amount of force-feedback (gain), damping, and tach feedforward of both manipulators (See Appendix D).

6) Indexed Master/Slave and Rate - Within a specified boundary the master arm gives direct incremental position commands to the slave. But once the master is pushed beyond the imaginary boundary, the master controller changes to a proportional rate joystick and the operator feels a counteracting force which is proportional to the rate of the slave. The operator can return to position control by moving the master back into the boundary. This combination allows the operator to efficiently switch between the two major forms of manual control —— rate and position. Since this mode frequently trades between rate and position control, an offset will usually exist between the master and slave positions.

Whenever a control mode is changed it is necessary to initialize the master position to the current slave position (when a master/slave mode has been selected) or to some zero reference position (when a rate mode has been selected). To avoid possible operator harm the computer warns the user through the DASI sonalert before movement of the master. During initialization movements the master arm is continuously checked to determine if a collision has occurred to prevent unnecessary damage to the arms and further protect the operator. Whenever control is passed to the master the arm gives a small jerk to indicate the trade (i.e., this is a form of apparent trading of control).

Although a time delay is not a form of manual control, it is a control constraint which may be imposed for experimental purposes (under a time delay the slave arm would be driven to duplicate the position of the master after an interval of time had elapsed). The time delay is entered through the STANDBY state at the same time the control mode is specified and can be set to any time between 0 and 6 seconds by the DASI potentiometer. Other control constraints which will be implemented in the future on the SUPERMAN system are communication blackouts, limited communication bandwidths, and limited or frozen degrees of freedom.

Through the STANDBY state the operator can zero both the master and the slave manipulators (ØE2 button on the control mode plate). The zeroed position can be used to calibrate the arms or as a convenient reference. A list of system options is also available by pressing the "OPTIONS" button (Figure 4-8). Each option is obtained through the LDAT switches on the Interdata front panel. The following options are currently available:

1) LDATS(12) - Time Delay
2) LDATS(13) - Disregard Takeover Commands
3) LDATS(14) - Rate with Joystick
4) LDATS(15) - No Force-Feedback

The operator can also execute the command register, execute a task file, or transfer to the DEFINE state from the STANDBY state. Each of these actions can be initiated through the corresponding button, signaling a change of state from STANDBY to EXECUTE or DEFINE.

4.3 DEFINE State

DEFINE is the primary state through which the operator enters a string of commands to be executed at a later time. Once the DEFINE key has been pressed, commands are entered by one of the sixteen specially dedicated buttons for each function (Figure 4-10). Each of the buttons used in the DEFINE state has dual functions. The second function for each button is denoted in gold letters below the button, whereas the major function is in black letters above the button (The small lettered commands in Figure 4-10 represent the gold commands on DASI). To enter a second command the operator simply pushes the 2nd button and then the desired function key.

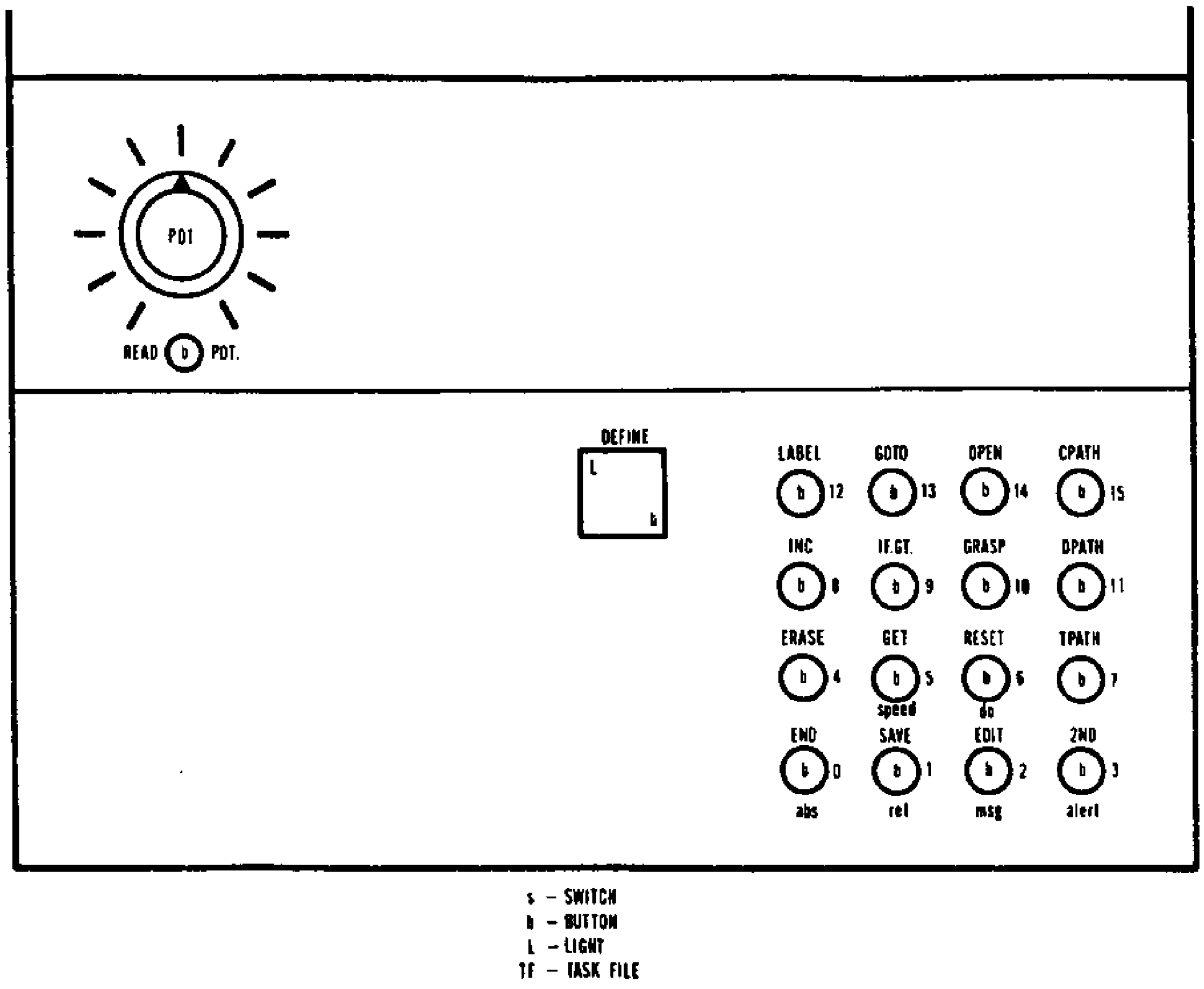


Figure 4-10: Primary DASI Input for DEFINE State

A listing of each button and a definition of its function is given on the following pages. The general format that will be used throughout this text will be as follows:

[Button Push]	Button keystrokes are denoted by brackets.
(Pot Inputs)	Potentiometer inputs are denoted by parentheses.
"Keyboard Entries"	Keyboard entries are denoted by quotes.
COMPUTER REPLIES	Computer replies are denoted by capital letters.

The dedicated-button commands associated with the DEFINE state are:

Button Number	Command and Definition	Usage
0	END	[END]
	Final command used to signal completion of DEFINE state.	
1	SAVE	[SAVE] "NAME" [SAVE][TASK FILE N]
	Used to save the command register on the disk as either a task file or a named file. A task file can be recalled only by one of eight buttons in the STANDBY state, whereas a named file is saved under a user-designated title and can only be recalled by the same name through the GET button (5) in the DEFINE state or as a subroutine in the EXECUTE state.	
2	EDIT	[EDIT]
	Enters the EDIT state (Section 4.4).	
3	2ND	[2ND]
	Used to signal that the second function of the dual command keys will be used.	
4	ERASE LAST LINE	[ERASE]
	Used to erase the last entry in the command register.	

5 GET [GET]"NAME"

Used to retrieve a named command file from the disk either as a subroutine in the EXECUTE state or immediately in the DEFINE state. GET asks for the name of the file to be recalled and then locates the file. If GET is to retrieve the file immediately it will read the file, string it on the end of the command register, modify the statement numbers, and return for further input. Otherwise it will insert a subroutine call in the command register which will retrieve the file at execution.

6 RESET [RESET]

Used to initialize the necessary internal variables and the command register to zero.

7 THROUGH PATH [TPATH]

Records the present position of the arm for use in EXECUTE as a through-point. (A through-point is a position which the operator desires the arm to move through without stopping, i.e. non-zero velocity point.)

8 INCREMENT DOF XXXX [INC][DOF](XXXX)

Makes an incremental motion in the desired degree of freedom by a selected value. The user enters the INCREMENT command, then the degree of freedom (DOF), adjusts the desired increment XXXX through the potentiometer and presses the READ POT button directly beneath the potentiometer.

9 IF DOF FORCE.GT. [IF.GT.][DOF](XXXX)

EXECUTE NEXT COMMAND

If the force level in the desired degree of freedom (DOF) is greater than the level set by the operator (XXXX) the following command is executed. If the force level is less than the level set by the operator, the command is skipped during execution. The user enters the IF FORCE.GT. command, then the desired degree of freedom, and adjusts the force level through the potentiometer.

10 GRASP WITH FORCE XXXX [GRASP](XXXX)

The user enters the GRASP command and adjusts the force level through the potentiometer.

11	<u>DISCRETE PATH</u>	[DPATH]

Records the present position of the arm for use in EXECUTE as a terminal point. During execution, the slave arm is moved from its current position to the recorded position with zero final velocity.

12	<u>LABEL N</u>	[LABEL][N]

Labels a position in the command register which can be returned to through a GOTO command. The user presses the LABEL button and then the number N of the desired label.

13	<u>GOTO N</u>	[GOTO][N]

GOTO is a conditional command which moves to label N unless the operator signals during execution to change the branch to [M] by pressing a different button. To enter the command the operator presses the GOTO button and then the number N of the label to which GOTO should branch.

14	<u>OPEN</u>	[OPEN]

Open jaws.

15	<u>CONTINUOUS PATH</u>	[CPATH]

Records the position of the master manipulator every 0.1 second for use in EXECUTE. A continuous path is achieved by interpolating between the recorded positions.

16	<u>ABSOLUTE</u>	[abs]

Informs the execution compiler that the command register is to be executed exactly as recorded (see RELATIVE). The user enters the absolute command by pressing the 2ND button [#3] and then the ABSOLUTE button [#0].

17	<u>RELATIVE</u>	[rel]

Informs the execution compiler that the positions in the command register are to keep the same relative displacement with respect to each other, but are to be transformed so that the first position following the RELATIVE command corresponds to the position of the slave at the time of execution. A RELATIVE command can be cancelled by an ABSOLUTE command, with the result that only the positions between the RELATIVE and ABSOLUTE commands are transformed. The user presses the 2ND button [#3] and then the RELATIVE button [#1] to enter the command in the register.

2ND - 2	<u>MESSAGE</u>	[msg] "MESSAGE"
	Allows the operator to enter a message which will be displayed on the monitor screen when the message command is reached in the command register.	
2ND - 3	<u>ALERT</u>	[alert]
	Sounds the sonalert on the DASI control console for 1/2 second when this command is executed.	
2ND - 4	Available button command presently not assigned.	
2ND - 5	<u>SPEED ADJUST</u>	[speed]
	Allows the user to fix the velocity of the arm during execution from 0 to 100 percent of the maximum velocity. The speed is permanently set until another speed adjustment is encountered in the command file to turn it off.	
2ND - 6	<u>DO LOOP</u>	[do][N][I]
	During execution the processor will execute all commands through label N, I times before continuing beyond label N.	
2ND - 7	<u>ADAPT XYZARL</u>	[adapt][X][Y]...
	When executed this command records the slave position and stores the appropriate degrees of freedom, [X][Y]..etc., in the calling file. This command results in a permament modification of the calling file, and therefore, should be used with discretion. When the file is executed again the manipulator will return to the modified position instead of the originally recorded position.	

Throughout the DEFINE state the yellow DEFINE button remains lit indicating that the current computer state is DEFINE. The computer visually signals that it is ready for command entry at the lower left corner of the monitor screen with, "ENTER DEFINE COMMAND." The ENTER COMMAND light on the DASI console also signals the ready condition.

To assist the operator in command entry the dedicated buttons respond with a positive click and a beep from the DASI sonalert. As each command is entered they are echoed on the screen in the command register (Figure 4-6) to give visual confirmation of the command to the operator. Whenever input from the DASI potentiometer is required the scaled values (force, distance velocity, etc.) are shown on the lower right corner of the screen with an appropriate title. The computer also checks the logic of command entry to insure that the operator has entered the required keystrokes. If an illogical sequence of strokes is encountered the computer responds with an error message and resets the register for continued entry.

Since the first entry in a command string must declare the file as either absolute or relative, the processor automatically inserts an absolute (ABS) command in the register if the first button pressed is not a relative (REL) declaration.

Upon completion of the DEFINE state by the operator the system compiles the command register and preprocesses all relative commands and force statements. Preprocessing in the DEFINE state decreases the number of execution calculations, and hence, the time required to execute the task (see Chapter II for the appropriate preprocessed calculations). As noted in Figure 4-1 the EDIT state is entered through the DEFINE state. The EDIT state will be discussed in the following section.

4.4 EDIT State

The EDIT state allows the operator to modify a command register which has been previously defined. Since it is required that the file be defined before editing, this state can only be entered through the DEFINE state. Once in the EDIT state, input to the system is primarily through the keyboard. Whenever DASI entries are required the computer cues the operator with, "ENTER DASI COMMAND." The following options are available after entering the EDIT state:

1) CHANGE A LINE
2) INSERT A LINE
3) DUPLICATE A LINE
4) DELETE A LINE
5) LIST COMMAND REGISTER
6) RETURN TO DEFINE

The editor checks the logic of keyboard and command entry to insure the proper sequence of keystrokes has been entered. If an illogical sequence is encountered the editor returns with an error message and initializes the editor for further command entry. Errors encountered in the EDIT state are shown in the message area of the screen (Figure 4-6). Some of the editor messages are:

```
COMMAND REGISTER IS FULL —— LINE INSERTION IS NOT POSSIBLE
ILLEGAL EDITOR COMMAND
FILE RETRIEVAL NOT ALLOWED BY EDITOR
SAVING OF FILES NOT ALLOWED BY EDITOR
```

WRONG KEY NUMBER ENTERED —— TRY AGAIN

CONTINUOUS PATH EDITS NOT ALLOWED

4.5 EXECUTE State

This state suspends the manual control mode in effect and executes the string of commands in the command register or task file. The EXECUTE state is entered through the STANDBY state by either the EXECUTE button (i.e., the command register) or one of the TASK FILE buttons. When a file is executed the master arm position is frozen (master/slave or combined modes), the computer performs the required relative calculations, executes the file, returns the slave arm to the initial execution position, and returns to the suspended manual control mode in STANDBY. The green EXECUTE button remains lit throughout the EXECUTE state regardless of whether the command register or a task file is being executed. Figure 4-11 shows the relevant button inputs for the EXECUTE state.

To facilitate in the testing of a file the command register allows the operator to step through each line of the program as well as to vary the maximum speed of the manipulator. During step execution only one command in the register is executed each time the EXECUTE button is pressed. The operator can also move up or down in the register by pressing either of the arrowed buttons (Figure 4-11). Whether in step mode or run mode the computer's position in the execution register is indicated by two congruent arrowheads (> >) which dynamically move through the register as the file is executed.

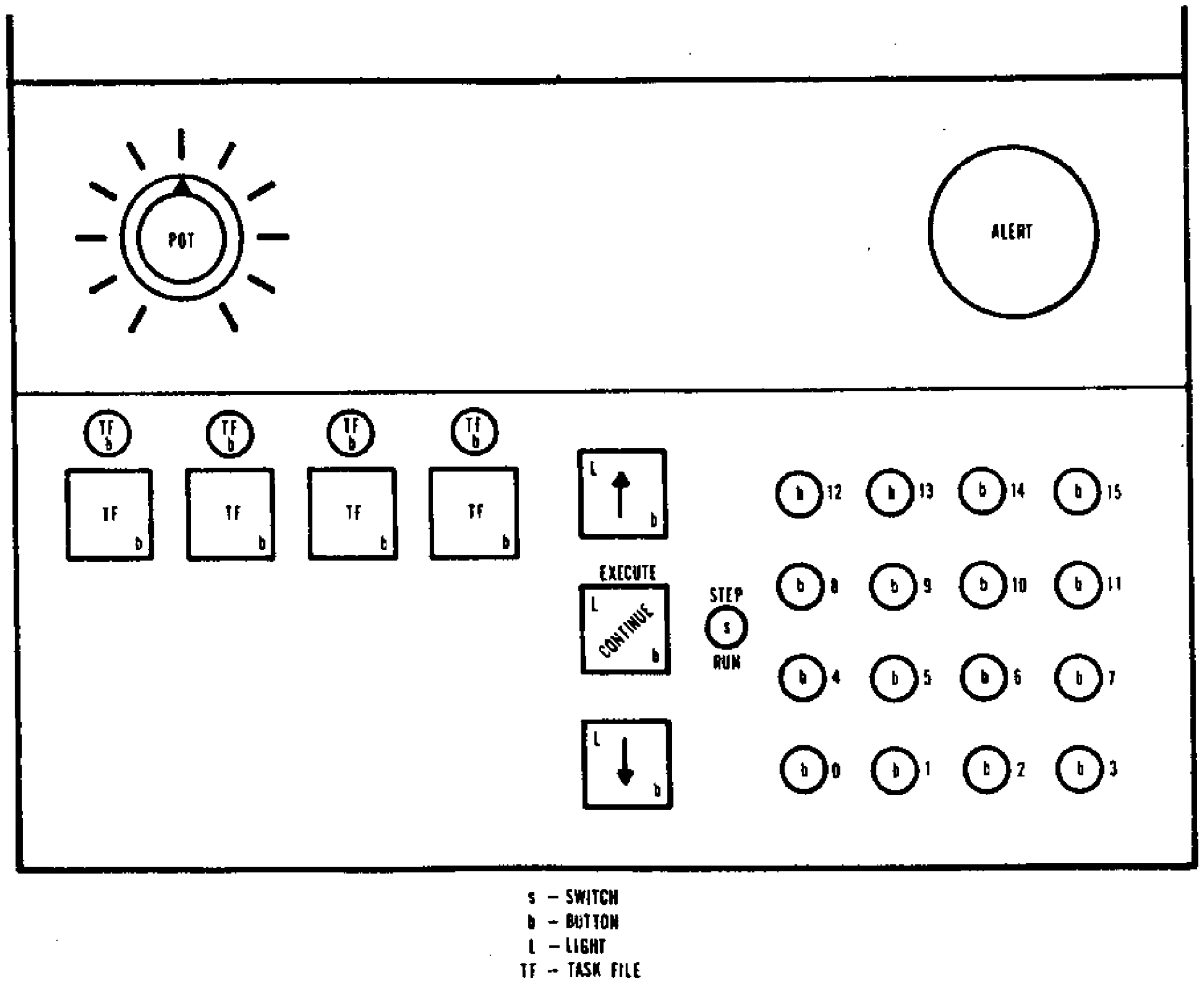


Figure 4-11: DASI I/O for EXECUTE State

The computer performs all relative calculations immediately after the EXECUTE button has been pressed. If the execution file (command register or task file) contains a relative declaration and the calculated arm configuration cannot be physically satisfied, control is returned to the STANDBY state and the following message is given to alert the user:

```
EXECUTION CANCELLED

EXECUTION FILE HAS A RELATIVE DISPLACEMENT
WHICH EXCEEDS THE MANIPULATOR WORKSPACE
```

During execution the processor continually checks the status of the button labels (Figure 4-11, numbered buttons) to determine if the operator has requested a transfer to a specified label in the execution register. If the label has not been defined previously, the computer returns with an error message (LABEL X UNDEFINED) and waits for further instructions before proceeding.

If the operator desires to take control during execution there are two methods available. The user can take immediate control: (1) by pulling on the appropriate manual control stick (see TAKEOVER section), or (2) by pressing the STOP button. All action ceases after the STOP button has been pressed until the human operator signals for continuation (green EXECUTE button), branch to a specified label in the register (GOTO button and then label number), or return to STANDBY and the suppressed control mode. Under an emergency situation the operator can also stop the manipulator action by hitting a red panic button which completely powers down the manipulator and servoelectronics.

Whenever the computer is in control of the arms the force levels of each joint are continuously monitored for excessive levels to protect the arms and environment. The manipulators are further protected by independent fuses on each degree of freedom (For inexperienced operators a lower amperage fuse is used than for an experienced user).

If the arm collides with its environment the processor responds by alerting the operator with a series of rapid beeps from the DASI sonalert and a flashing message on the monitor screen, while simultaneously backing away from the object to relieve the static forces. The message which flashes on the screen is as follows:

```
MAX FORCE LEVEL EXCEEDED
```

After the static forces have been relieved, the sonalert stops and the flashing message is replaced by the following message:

```
MAX FORCE LEVEL EXCEEDED
EXECUTION STOPPED ---- PRESS CONTINUE TO PROCEED
```

By pressing the STOP button the user is transferred immediately to the suspended manual control mode in the STANDBY state. If the CONTINUE button is pressed the computer will attempt to continue execution of the task from the location that it encountered the excessive force levels.

As mentioned in the DEFINE section, SUPERMAN has the capability of recalling subroutines by name during execution. Whenever a subroutine call is encountered in the execution register the cue pointer (> >) stops on that call as the file is retrieved. Once the subroutine has been located, the execution register on the screen is replaced by the subroutine register. The subroutine register is then executed as if it were a command file or task file. Upon completion of the subroutine the computer returns to the calling register, restores the original calling values and proceeds with the mainline execution. If the subroutine cannot be located on the disk, the computer returns with the following error message and returns control to the operator through the STANDBY state:

```
"name"  UNDEFINED
```

Subroutine resting is presently not allowed. Therefore, if a subroutine file is encountered which calls another subroutine the following error message is returned and the computer continues execution of the calling register:

```
SUBROUTINE NESTTING IS NOT ALLOWED
```

As noted in Figure 4-1, the TAKEOVER state is entered through the EXECUTE state. The TAKEOVER state will be discussed in the following section.

4.6 TAKEOVER State

TAKEOVER is a transition state between the two primary control modes, i.e., from computer control to the manual control mode in effect before the EXECUTE state. The operator enters this state and overrides the computer by pulling on the appropriate control input (i.e., the master whenever the suspended control mode involves position control; or the joystick or rate switches whenever the suspended control mode involves rate control). The TAKEOVER option is available in all six manual control modes.

Manual override of an autopilot has been studied in considerable detail in the past, but manual takeover from a computer controlled manipulator is essentially an uncharted realm. User intervention with an autopilot is a relatively simple task which involves disengaging the autopilot control and moving the joystick in the direction of the desired rate change. The control stick signals represent rate commands, and therefore, discontinuities in displacement do not occur. Similar attributes (i.e., simplicity of takeover and continuous displacement commands) are necessary for user intervention of a computer controlled manipulator. Rate control of a manipulator closely resembles the rate control of an airplane; therefore, manual override of a rate controlled manipulator can be implemented in much the same way as an autopilot. But if control were suddenly passed to the master/slave mode, the position signal from the master would probably be different from the previous computer signal to the slave and there would be a discontinuity in the displacement command. Clearly, manual override of a position controlled manipulator will present a new set of problems.

To prevent a jump in the position commands during manual override, either the master must continuously follow the slave's movements so that at the moment of takeover the two are properly aligned, or the difference in the position of the master and slave arms at the instant of takeover must be artificially maintained. It is not practical to have the master follow the slave, as some motions are too complicated for the human operator to follow. Therefore, the only other possibility, if position control is to be used, is to allow an initial mismatch between the master and slave.

If the joints of the master and slave are not equal when manual takeover occurs, geometric cross coupling* can occur, destroying the spatial correspondence between the master and slave end effectors. Since position control requires that an offset be put into the control loop when control is passed to the master to maintain the initial mismatch, it would be expected that a stimulus-response incompatibility will occur. If the offset is allowed to remain throughout the emergency task, this incompatibility could have serious effects on the human operator's ability to function properly if the emergency task requires anything more than a simple retreat motion. Clearly,

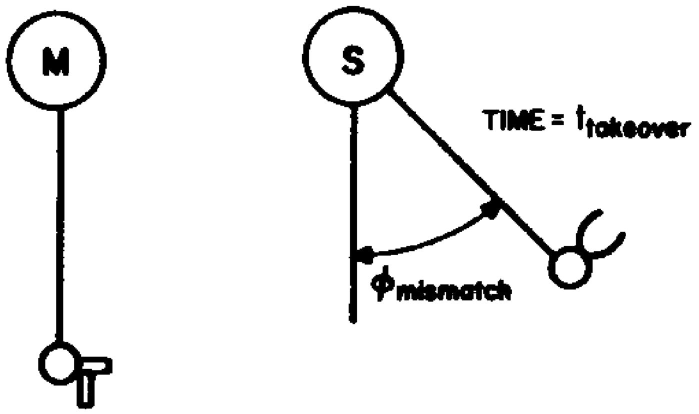
*Geometric cross coupling is a phenomenon where a master motion in one hand coordinate corresponds to a slave movement in an unexpected hand coordinate due to a geometric difference between the master and slave joints (see Appendix A).

the mismatch should be removed as quickly as possible. But, if the rate of decrease is too fast, the change in mismatch would be more detrimental than the offset alone, and if the rate is too slow the offset would remain throughout the duration of the emergency task. It is necessary, therefore, to find a rate of mismatch decrease which does not interfere with the operator's normal response, and yet, reduces the offset to zero in a reasonable time.

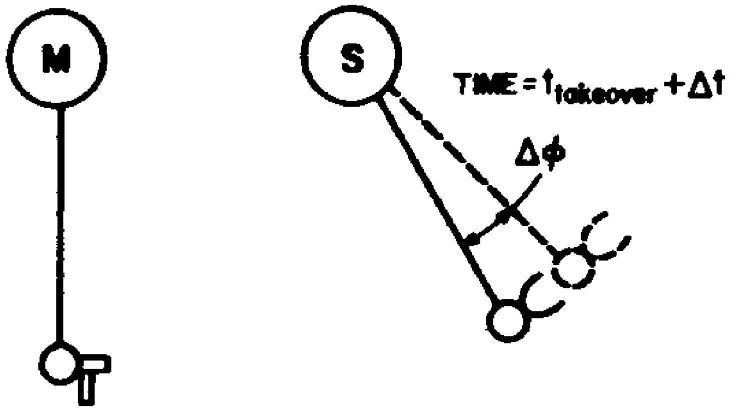
Figure 4-12a demonstrates a master/slave manipulator at the instant of takeover (takeover). After some increment of time Δt, the offset of the slave arm with respect to the master will decrease by an increment $\Delta\phi$ (Figure 4-12b). Eventually, the total number of increments will equal the original offset and the mismatch will be completely removed (Figure 4-12c). During the removal of the offset a transient condition will exist between the initially mismatched takeover and the final master/slave mode. The period of time during which the transient effects will be observed ($n\Delta t$) is a function of the initial mismatch distance ($\phi_{mismatch}$) and the rate of offset decrease ($\Delta\phi/\Delta t$). Since the initial offset cannot be directly controlled the total mismatch time can only be adjusted by the offset decrease rate as mentioned previously.

Experiments were performed to study some of the characteristics of the takeover problem (see Appendix B). Through these experiments a subjective decrease rate was determined using accuracy and path time as criterion. Clearly, if the emergency task does not

(a) Master/Slave Manipulator at Instant of Takeover



(b) Master/Slave Manipulator After Time Δt



(c) Master/Slave Manipulator After Mismatch is Dissolved

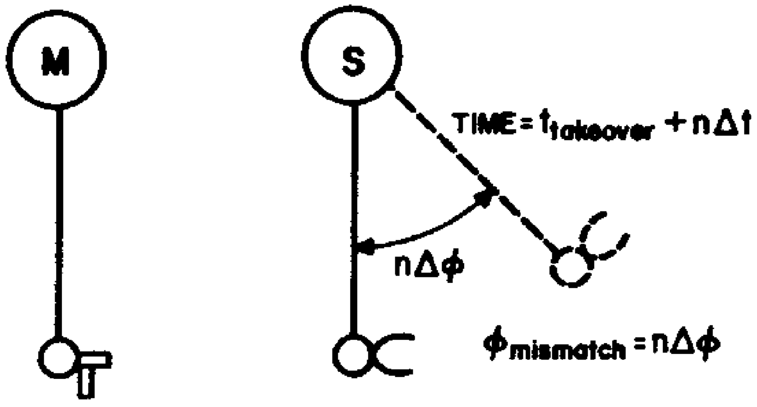

M
S
TIME = $t_{takeover} + n\Delta t$
$n\Delta\phi$
$\phi_{mismatch} = n\Delta\phi$

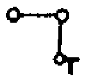
Figure 4-12

require a high degree of accuracy, a faster decrease rate would allow the arms to approach a true master/slave mode more quickly. But the computer cannot determine the accuracy needed by the human operator to complete an unpredictable task (i.e., an emergency task). To avoid this problem the SUPERMAN system begins cancelling the offset with a fixed rate, and then, if the operator wishes to increase the rate, he can do so by pressing the STOP button until the desired speed is obtained. If the initial mismatch is small the offset is usually cancelled before the operator is aware of it. If the offset is larger, the operator has the option of (1) adjusting the rate to a value suitable for the emergency task or (2) working with the initial decrease rate. Since the mismatch must be dissolved before control is passed to the STANDBY state, the computer will not allow the operator to exit the TAKEOVER state until the offset has vanished. The diamond in Figure 4-1 signifies that after the mismatch has been removed, the operator has the option of moving into the STANDBY state by pressing the STOP button, or continuing the EXECUTION state by pressing the CONTINUE button.

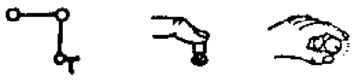
4.7 Example SUPERMAN Programs

To demonstrate some of the features of the SUPERMAN system a number of simple programs will be considered. The examples chosen characterize the relative ease and speed of program entry——important attributes for a real-time supervisory system.

Each line of the command string depicts the symbolic and analogic inputs required of the operator to graphically demonstrate program entry. The following symbols will be used to represent the actions required of the human operator:

[BUTTON PUSH]	
(POT READINGS)	
"KEYBOARD ENTRIES"	
ANALOG INPUTS	

The order of each symbol from left to right indicates the sequence of command entry. For example,



would mean that the operator positions the slave, presses a button and then adjusts the DASI potentiometer. To further clarify the program entry procedure actual pictures of the manipulator are included whenever a position is recorded (DPATH, TPATH and CPATH).

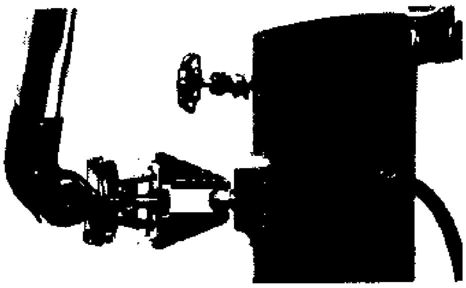
As the first example consider a string of commands to take a nut off of a bolt and put it in a box. This program can be broken down into two major sections; one removes the nut and the other places it in the box. Since the user would prefer one nut removal program to be used for all nuts regardless of the orientation of the nut, a RELATIVE command should obviously be the first command in the register (The RELATIVE command and all of the following commands are described in the section on the DEFINE state).

1 [RELATIVE]

Relative file declaration. Location of nut in space is not known prior to execution.

2 [LABEL] [1]

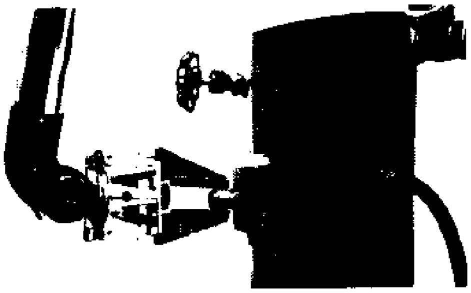
3 [DPATH]



Place the slave on the nut and record that position by pressing the DPATH button.

4 [GRASP] (200)

5 [DPATH]



Turn the end effector 180° and record that position.

6	[INCREMENT] [Y] (-300)	Increment the slave by 300 counts in the direction that would pull the nut off.
7	[IF FORCE.GT.] [Y] (100)	If the force in the Y direction is greater than 100, the nut is still on the bolt, therefore execute the next command.
8	[GO TO] [2]	
9	[GO TO] [3]	If the force is less than 100 in the Y direction, the nut is free and this command would be executed.
10	[LABEL] [2]	
11	[INCREMENT] [Y] (300)	Return the arm to position before incrementing in command #6.
12	[OPEN]	Release the nut.

13 [GO TO] [1]

Return to LABEL 1 and continue turning the nut.

14 [LABEL] [3]

End of the first part of task. The nut is off.

[SAVE] "NUT-OFF"

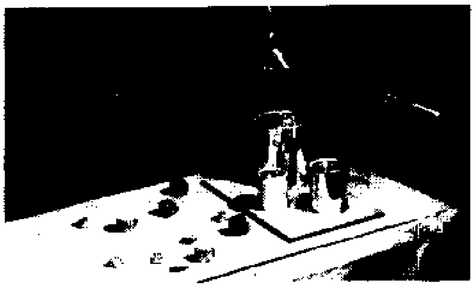
Save command register as the named file "NUT-OFF" (typed in at the keyboard).

The second part of the task requires the manipulator to place the nut in a box. The entire command register for the program to put the nut in the box would be as follows:

1 [ABSOLUTE]

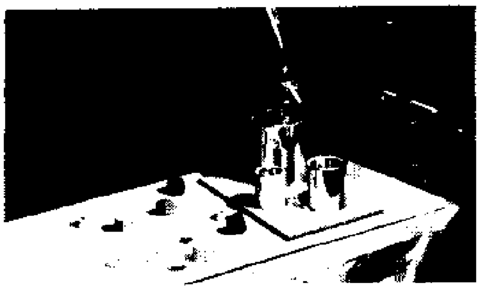
Declare file as absolute. The box will always be in the same location

2 [TPATH]



Move the slave to a position just over and above the outside edge of the bucket and record this position by pressing the TPATH button.

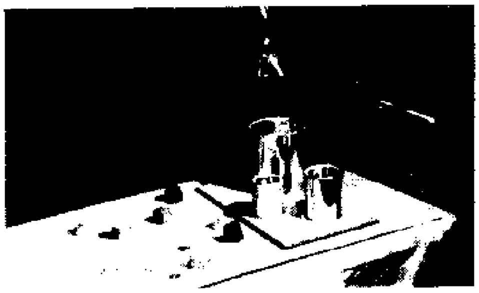
3 [DPATH]



Move the slave to a position over the center of the bucket and record the position.

4 [OPEN]

5 [TPATH]



Enter the same position as recorded in command #2 by duplicating line 2.

[SAVE] "NUT-IN-BOX"

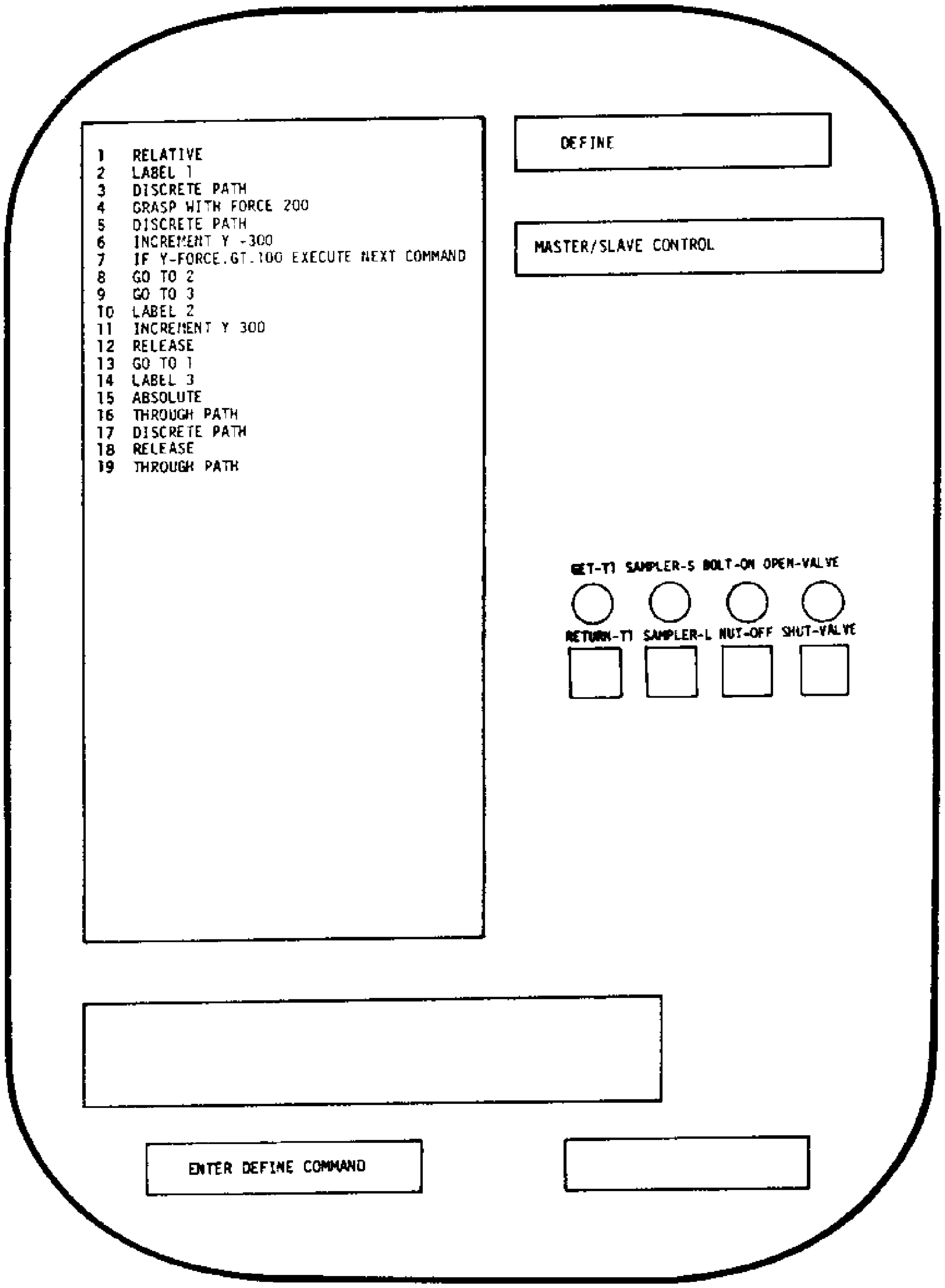
At this point the operator could call either program as a subroutine and execute it. The NUT-OFF program would simply take the nut off and return control to the calling program as soon as the nut was free. The NUT-IN-BOX program would move to the bucket, release whatever was held in the end effector, return to the position it was called from, and continue with the calling program. Hence, the NUT-IN-BOX program could be used as a generalized sampler to put whatever was required in a sample basket. But the present status of these files (i.e. a named file) requires that the operator type in each name to obtain the file to execute it or call them from within another program as subroutines. If the operator performs the following commands the file will be saved as a task file which can be immediately executed at the touch of a button:

```
[GET] "NUT-OFF"
[GET] "NUT-IN-BOX"
```

The computer will reply by stringing the two files together in the command register (renumbering GOTO and LABEL statements if necessary) as shown in Figure 4-13. The operator would then enter:

```
[SAVE] "TASK-FILE"
```

and press the button which will retrieve the file (e.g., button #1). To remove a nut and put it in the box the operator simply presses the same button, the execution compiler transforms the first half of the register relative to the position of the slave at the instant the button is pressed and then executes the program. After the nut is removed and placed in the box the slave returns to the operator's position and the computer relinquishes control.



```
RELATIVE
LABEL 1
DISCRETE PATH
GRASP WITH FORCE 200
DISCRETE PATH
INCREMENT Y -300
IF Y-FORCE.GT.100 EXECUTE NEXT COMMAND
GO TO 2
GO TO 3
LABEL 2
INCREMENT Y 300
RELEASE
GO TO 1
LABEL 3
ABSOLUTE
THROUGH PATH
DISCRETE PATH
RELEASE
THROUGH PATH
```

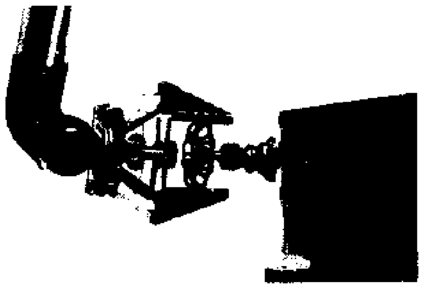
Figure 4-13: Display for Combined NUT-OFF and NUT-IN-BOX Program

To further demonstrate the SUPERMAN programming capabilities a string of commands to open a valve will be considered. Again, since the user would prefer a simple program which could be used on any valve regardless of its orientation, a relative declaration will be the first command in the register.

1 [RELATIVE]

2 [LABEL] [1]

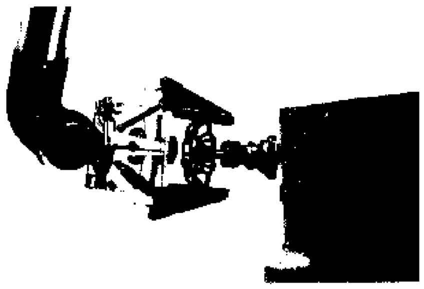
3 [DPATH]



Place the slave on the valve and record that position by pressing the DPATH button.

4 [GRASP] (200)

5 [DPATH]



Turn the end effector 180° and record the position.

6 [IF FORCE.GT.] [RE] (20)

If the wrist rotation force during the above motion is greater than 20 the valve is open and the next command should be executed.

7 [GO TO] [2]

8 [OPEN]

If the wrist rotation force is less than 20 the valve is still turning, so re- lease the valve and continue opening.

9 [GO TO] [1]

10 [LABEL] [2]

The valve is open.

11 [OPEN]	Release the valve.
12 [MESSAGE] "VALVE OPEN"	Alert the operator that the valve is open with a message.
13 [END]	End DEFINE state.

The computer will string the following commands on the file and return to the STANDBY state:

```
DPATH TO RETURN CONTROL
END
```

The "DPATH TO RETURN CONTROL" statement causes the slave to return to the initial execution position before relinquishing control to the operator so that the master and slave are matched. The operator would execute the command register in either the step mode or the run mode (see EXECUTE section).

The tasks which have been implemented on the SUPERMAN system are: (1) tool retrieval, (2) tool return, (3) auto-sampler, (4) open/shut valve, (5) nut-off, (6) auto-digger, and (7) bolt-on (Listings of each of the command registers for each of these programs are given in Appendix F).

The bolt-on program is particularly sophisticated in that it recognizes three possible results — cross threading, inability to engage threads, and completion of task. These patterns are easily recognized through logical statements (i.e., IF FORCE.GT., etc.). Figure 4-14 shows a bolt being threaded into a nut by the computer.

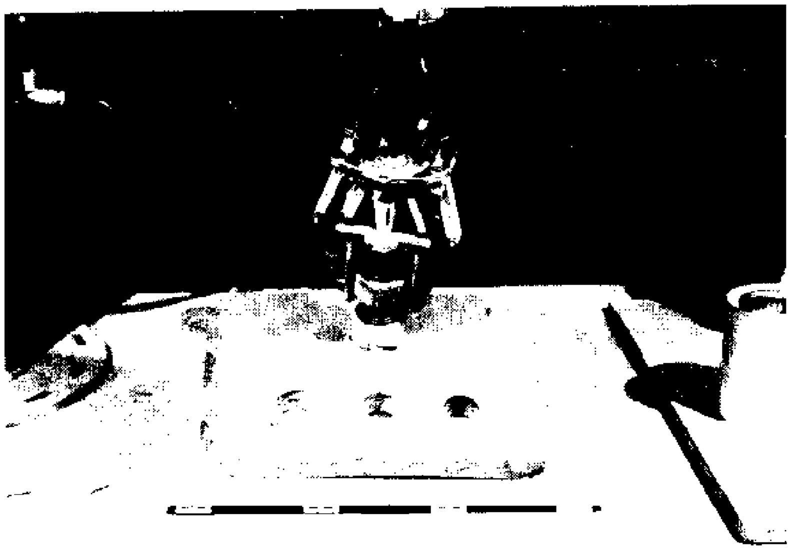


Figure 4-14: Bolting Operation under Supervisory Control

The program initially turns the bolt through 360 degrees while simultaneously checking the rotation torque level. If at any time during the first turn the torque level becomes excessive the bolt is cross threaded. After the first turn the program increments away from the bolt along the rotation axis and checks the force. If the force is less than some prespecified threshold level the threads have not started. Finally, if the force is greater than the threshold level the computer continues to turn until the rotation torque increases to the specified tightness and returns control to the operator.

The tool retieval and tool return programs use the ADAPT command to detect minor changes in the position of the tool rack. When the tool task-file is executed the ADAPT command records the tool rack position and stores the Y,Z,A,R, and L degrees-of-freedom in the calling file. When the file is executed again the manipulator will return to the modified position instead of the originally recorded position.

CHAPTER V
EXPERIMENTS, DATA, AND RESULTS

The purpose of this study was to determine the applicability of supervisory control to remote ocean work. Therefore, the objective of the experiments was to build a framework upon which comparisons of supervisory and manual control could be made. Unfortunately, two of the suggested uses for supervisory control (i.e., limited frame rate and time delayed conditions) were not implemented in time to be experimentally investigated. Hopefully, these results will be obtained in the near future. In any case these restricted conditions can only degrade direct manual control and make supervisory control look better. Therefore the experiments which follow provide a "best case" for direct manual control relative to supervisory control (or a "worst case" for supervisory control in such a comparison).

The manual control modes presently available to the ocean industry range across the spectrum from switch rate to force reflecting master/slave. Therefore a fair comparison should at the very least, include the major control divisions (position and rate). It was decided that two representative control modes from each class would be used. The two forms of position control were master/slave with force feedback and master/slave without force feedback. The two forms of rate control were switch rate control and joystick variable rate control.

5.1 Experimental Considerations

Previous investigators have found that task completion time under manual control increases with task complexity. According to a hypothesis advanced by Sheridan and Verplank[4] it would be expected that the combined programming and execution time for a supervisory system would also increase with the complexity of the task. As shown in Figure 5-1, at some level of task complexity the supervisory control scheme should become faster than the direct control method. But at what level of complexity will this occur if it occurs? Or must time delays and limited frame rates be present for it to occur? The experiments were performed with the expectation that a relationship such as Figure 5-1 might be found for some of the control modes. It should be noted though, that the primary purpose of these experiments was to demonstrate the usefuleness of supervisory control for remote undersea tasks and not the verification or negation of this hypothesis. Therefore, the experiments were designed to be representative of underwater tasks and were not intended to be relative tests of how complexity affects the completion time (i.e., the tasks were not designed on the criterion of complexity, but rather applicability to the marine environment).

Some of the tasks which could be required of a manipulator system in an undersea environment are:[6,45,46,47]

1) assessing damage (poking, prying, etc.)
2) bolting/unbolting

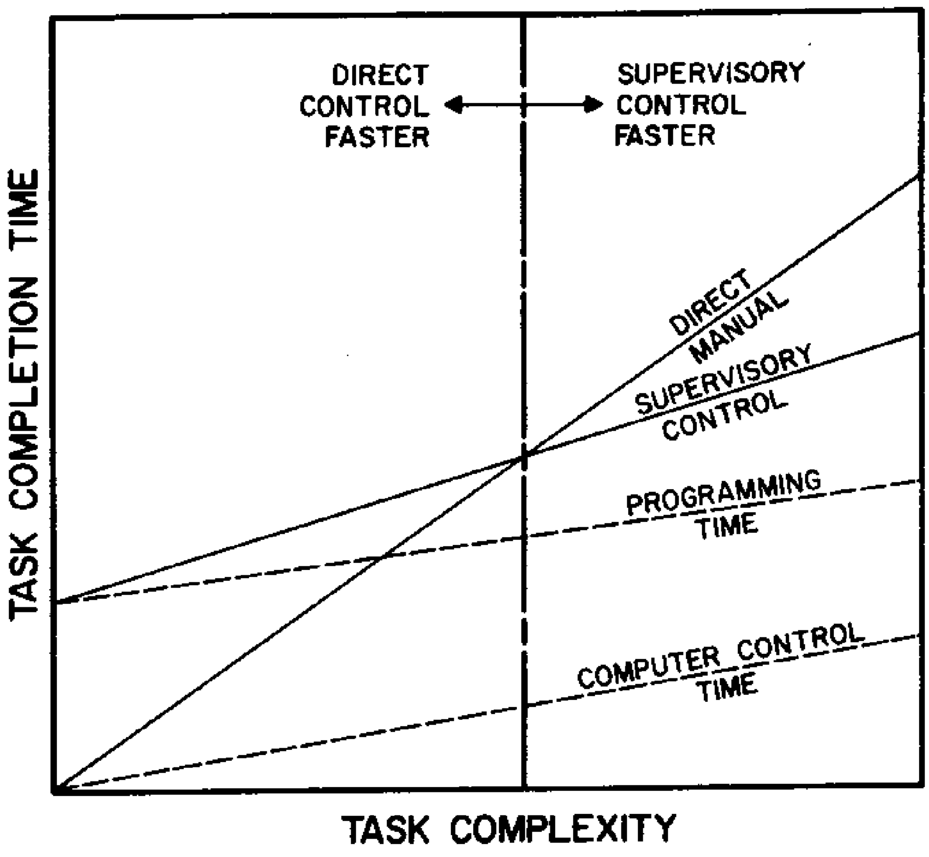


Figure 5-1: Hypothetical Curves of Task Completion Time Versus Task Complexity (Modified from Ref. [4])

3) welding
4) connecting hoses
5) cutting (pipes, wire, cable, etc.)
6) digging
7) drilling
8) tapping
9) fastening
10) lifting objects
11) pulling
12) recovery
13) reaching into confined spaces
14) threading cables
15) untangling cables
16) water jetting
17) wire brushing
18) opening/closing valves
19) sampling
20) coring

Several of the tasks in this list require special tools. Usually the tools perform the desired function directly, requiring little or no movement of the manipulator once the tool is in place. Hence, a major portion of the task requires the operator to retrieve and return the tool.

The following tasks were chosen as representative of the requirements of marine manipulation (A brief description of each will be given as a foundation; detailed explanations of the tasks and experiments will be given in Section 5.4):

GET TOOL - starting from a prespecified position, move to the rack, grasp the tool, being sure it is properly seated in the hand, and return with the tool to the initial position.

RETURN TOOL - starting from a prespecified position with tool in hand, move to the rack, replace the tool on the rack being sure it is properly seated, and return to the initial position.

NUT-OFF - starting from a prespecified position, move to the nut, orient the hand, and remove the nut without dropping it.

SAMPLER - starting from a prespecified position, pick-up thirteen randomly placed samples and put them in buckets according to their size.

OPEN/CLOSE VALVE - starting from a prespecified position, move to the valve, orient the hand, and open or close the valve as required.

DIGGER - starting with a shovel grasped firmly in the end effector, fill a bucket with simulated soil.

Bejczy has stated that there are three experimental factors which should be considered for the evaluation of a manipulator system:[48]

1) Effectiveness of control measured by the binary categories of "success or failure".

2) Quality of control evaluated by the combined measure of "accuracy and time".

3) Cost of control measured through the "consumption of applied resources".

It was decided that the experiments would only consider effectiveness and quality of control as the cost criteria could not be implemented within the time constraints of the project.

It has been shown by a number of investigators that the time required to perform a task can be broken into a number of distinctly different motions. For example, one classification divides the task time for control with a time delay into get, transport, and position motions.[49] For a peg-in-the-hole task Hill[50] has shown that the task has two independent motions which determine the total task time - trajectory (i.e., gross travel) and precision. A scheme similar to Hill's will be used to describe the task completion time for a supervisory system using manual and/or computer control:

$$TT = f\begin{pmatrix}\text{task complexity \&}\\ \text{control philosophy}\end{pmatrix} + g\begin{pmatrix}\text{task location \&}\\ \text{control philosophy}\end{pmatrix} \qquad (5\text{-}1)$$

where,

TT	=	Task Time
g	=	Time to <u>locate</u> the task. This time is a function of the initial hand/task locations and the control philosophy used to locate the task (Figure 5-2).
f	=	Time to <u>perform</u> the task. This time is a function of the task complexity and control philosophy used to perform the task (Figure 5-2).

These equations apply to all forms of control, both manual and computer. When a task is executed by the computer, the location actions can be performed by manual control, computer control, or a combination of the two (Figure 5-2). For example, manual location (g_m) of the task is usually done for relative tasks (Chapter 2). Computer location (g_c)

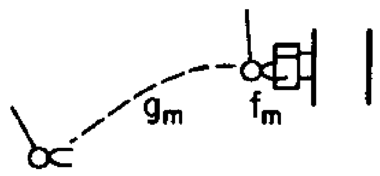
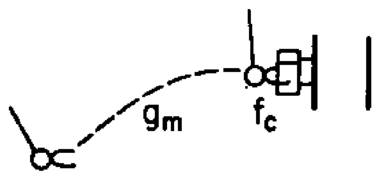
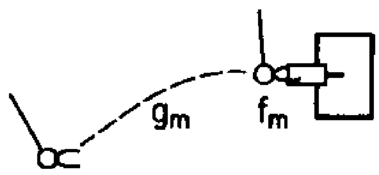
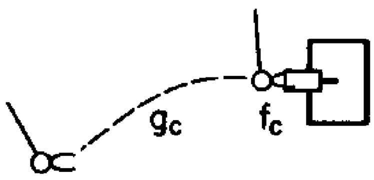
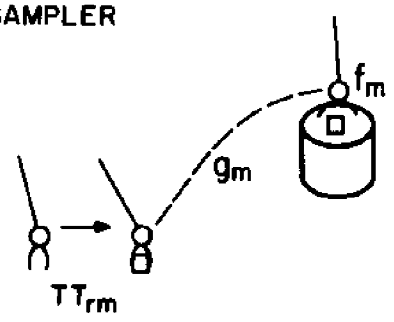
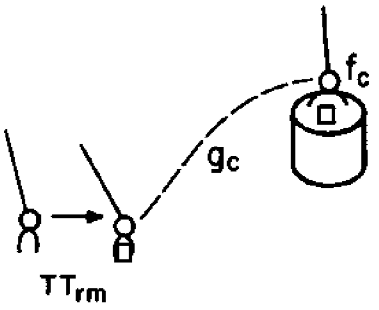
	MANUAL	COMPUTER
RELATIVE	NUT-OFF  $TT_m = f_m + g_m$	 $TT_c = f_c + g_m$
ABSOLUTE	GET-TOOL  $TT_m = f_m + g_m$	 $TT_c = f_c + g_c$
COMBINED	SAMPLER  $TT_m = f_m + g_m + TT_{rm}$	 $TT_c = f_c + g_c + TT_{rm}$

Figure 5-2: Definition of Location and Performance Time

of the task occurs when the position of the task can be identified or is fixed and known (e.g., absolute tasks). Finally, combined location actions occur when a task is composed of a number of relative and absolute subtasks. For example, when the task requires samples to be placed in a bucket, the action of grabbing the block is a relative subtask (TT_{rm} = task time to manually grab the relative block in Figure 5-2), and the action of moving the block to the bucket is an absolute subtask (Figure 5-2). The total task completion time under computer control consists of the time it takes to manually find the sample (TT_{rm}), the time it takes the computer to locate the bucket (g_c), and the time it takes the computer to perform the task (f_c).

Under manual control the trade from location to performance actions is continuous, and therefore the determination of the boundary between these task times (i.e., location and performance) can be difficult. When a relative task is executed under computer control, the location actions are usually done by manual control and the task performance actions by the computer. During the trade from manual location movements to computer execution movements a discontinuity in control occurs making the determination of these two times rather simple (remember, this "discontinuity" is a desired result since trading of control should be "apparent"). If the location time of a task can be determined under manual control, it should be possible to use this time as the location time of the computer controlled tasks. If the computer execution time is constant, a prediction of task completion time would be given by:

$$TT_c = E_c + g_m\begin{pmatrix}\text{task location \&}\\ \text{manual control mode}\end{pmatrix} \qquad (5\text{-}2)$$

where,

E_c = Constant computer execution time.

From the above equation it is seen that the total computer task completion time can be obtained by adding the manual location time to the execution time.

As stated the experiments were chosen on the basis that they were representative of marine manipulation tasks. Although the tasks were not designed to be relative indices of complexity, it was hoped that the hypothesis advanced by Sheridan and Verplank[4] could be demonstrated. It was decided that the evaluation of the experiments would be based on effectiveness and quality of control. The time measure would be divided into location and task performance times to indicate the time spent by each action under computer control.

5.2 Equipment

The specifics of the SUPERMAN system have been described elsewhere (Chapter 4), and therefore, will not be discussed here. The manipulator laboratory was arranged as shown in Figure 5-3 during the experiments. Many of the objects schematically represented in the figure can be identified in Figure 4-2. Figures 5-4 through 5-7 show the movable task hub, sample buckets, task board, simulated tools, and tool rack built for these experiments (Note, the tool rack in Figure 5-5 is normally mounted to the base of the manipulator by a

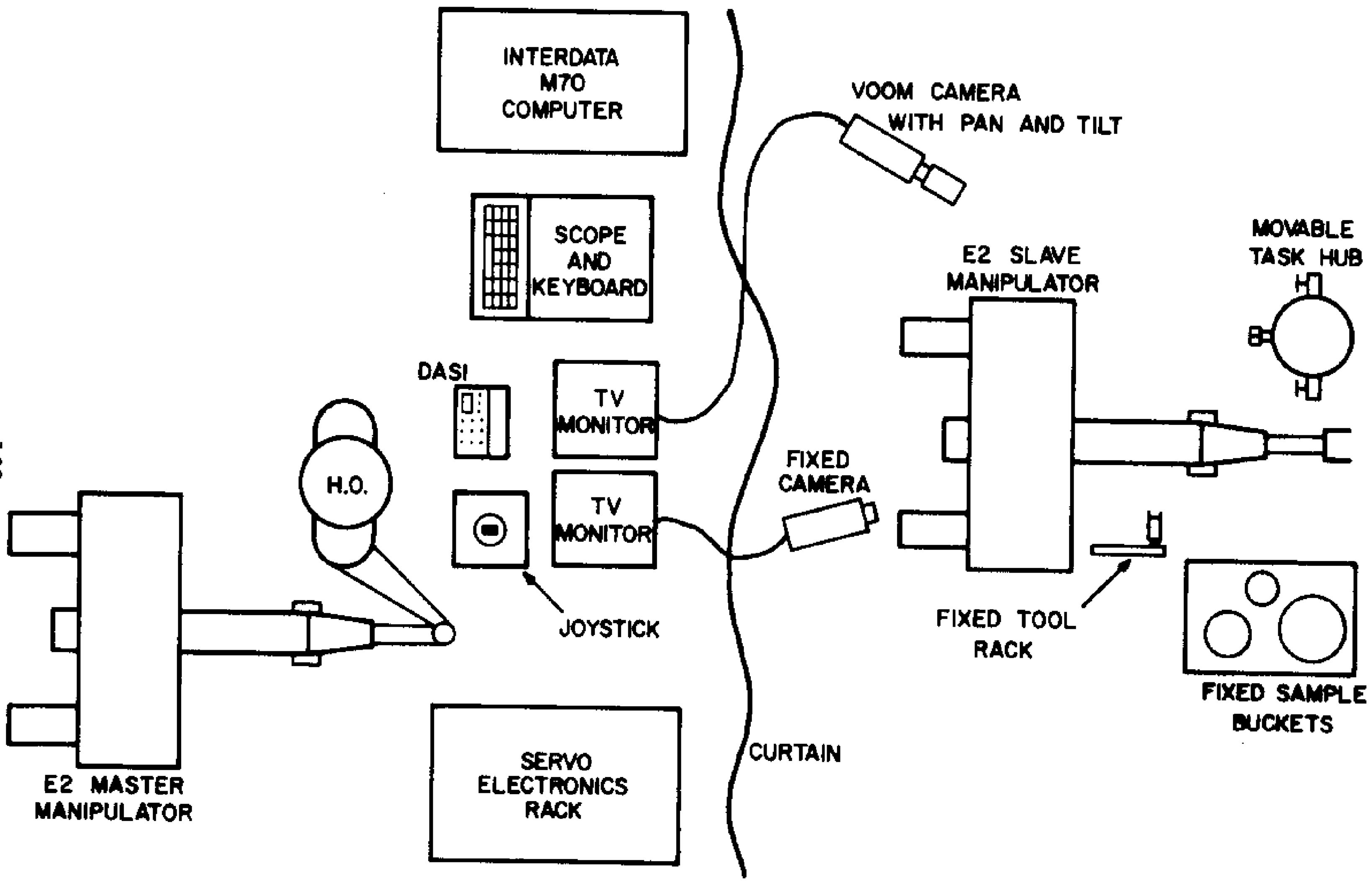


Figure 5-3: Schematic of Experimental Layout

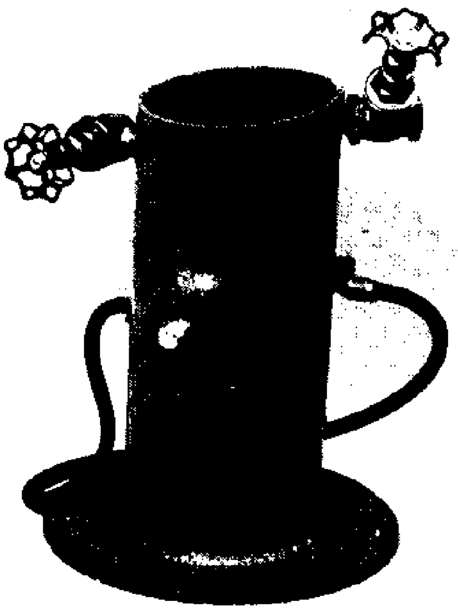


Figure 5-4: Task Hub Designed for Experiments

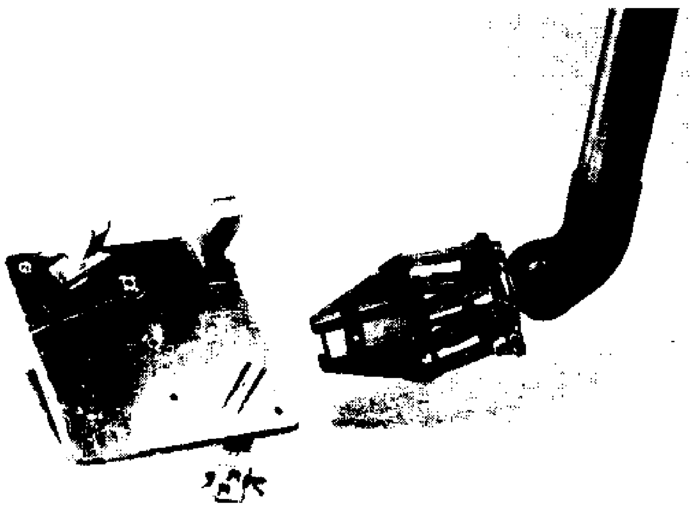


Figure 5-5: Tool Handles and Rack Designed for Experiments

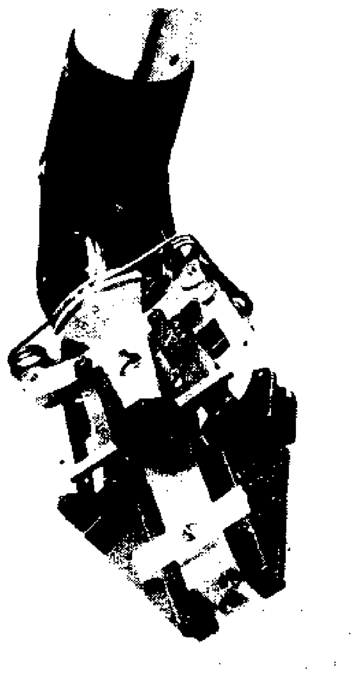


Figure 5-6: Proper Seating of Tool Handle in End Effector

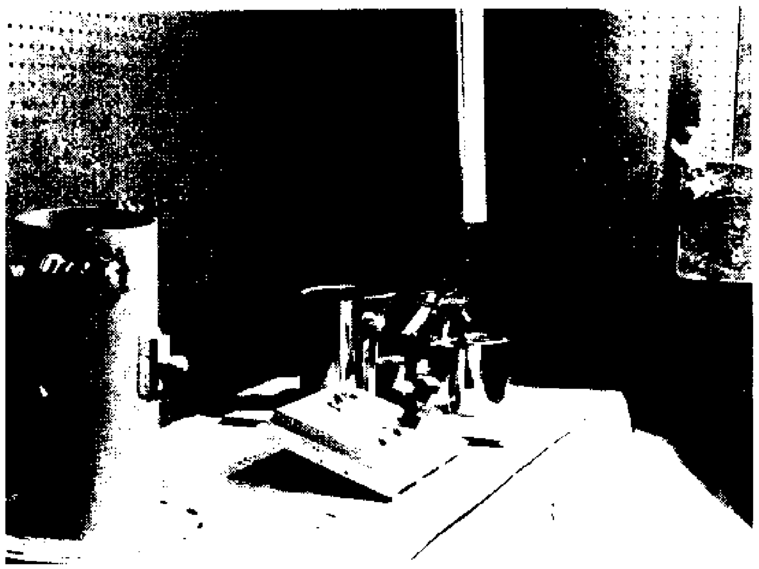


Figure 5-7: Movable Task Hub,Task Board,Sample Buckets,and Tool Rack

centilever beam. See the shadow in Figure 5-7).

5.3 Subject Training

Three classes of subjects were used for these experiments, one experienced, four well trained, and two untrained subjects. Due to time constraints only three subjects were used for four of the tasks (tool retrieval, tool return, nut-off and sampler), and only one subject was used for the remaining two (open/closed valve and digger).

The well trained subjects had an average of 20 hours training given in 15 minute intervals for each of the control modes. On the average the trained subjects performed 1.5 out of the possible 6 tasks. Due to the variations in time required by each subject to reach a plateau of performance, some of the subjects performed more than the average while others performed less. The four trained subjects were given incentives to perform well in the form of bonuses which would be awarded to the best combined time and error rates in any control category.

Generally, after the subjects practiced for 15 minutes with a particular control mode a simulated task was performed. Unfortunately, during the initial stages of the experiments the simulated tasks were not run in earnest, and therefore, the subjects tended to "take it easy". After the author became aware of this effect, the tasks were run under actual experimental conditions to insure subject cooperation.

When the subjects appeared to show a plateau, experiments were begun. But since the experiments usually stretched over a period of several days, the subjects were asked to "reperform" some of the tasks due to a "mistake". If the subjects showed marked improvement the tasks were performed again until the learning curve levelled off. One of the subjects was remarkable in that after five minutes of practice he was achieving faster times than the expert and with a lower error rate. This subject exhibited almost no learning curve at all.

The author was used as the baseline experienced subject. He has over 200 hours of practice on a number of manipulator systems and intimate knowledge of the SUPERMAN system. It may be reasonably assumed that the experienced subject underwent little or no learning. The experienced subject performed all six of the tasks without a "warm-up" period.

The untrained subjects had a total of 3 hours training time for all control modes (i.e., 30 minutes per control mode and viewing condition). The learning curves of the untrained subjects were not observed. The only requirement placed on their training sessions was to insure that each control mode was given equal training time. After the 3 one-hour familiarity and adjustment periods were over the subjects were allowed 24 hours of rest and the experiments were begun. None of the subjects had any previous manipulator experience. The mean times of the untrained subjects were always above the maximum value of the trained subjects for the same task and control mode.

5.4 Tasks and Experiments

As mentioned, the tasks chosen as representative of marine manipulation were: (1) tool retrieval; (2) tool return; (3) taking a nut off; (4) sampling; (5) opening or closing a valve; and (6) digging. The experiments involved four different control modes: (1) master/slave with force feedback; (2) master/slave without force feedback; (3) variable rate control with a joystick; and (4) variable rate control with switches. With regard to the video arrangement, both mono and two-view conditions were tested for comparison. Each experiment was performed 5 times to obtain a statistical mean and standard deviation (see Appendix G). Both purely manual control and computer control were used. These conditions, combined with those mentioned in the previous section, give:

(2 viewing conditions) x (4 control modes) x (2 control philosophies — manual and supervisory) x (5 runs) x (3 subjects x 4 tasks + 1 subject x 2 tasks)= 1120 (required runs)

Two of these experiments (tool retrieval and tool return) have constant computer execution times regardless of the manual control mode used since they do not require manual location time. This reduces the matrix to 900 runs.

To reduce the number of runs further, the constant task time equation (Equation 5-2) was used whenever the location time of the task could be identified under manual control. Since the "nut-off" and "valve" experiments are relative tasks with readily distinguishable location times, these tasks were only performed under manual control, thus reducing the number of runs to 680.

The experiments were scored on the basis of recorded time and errors. The subjects were not given specific instructions to minimize either quality, but only to weigh them equally. The experiments were not redone if errors occurred (regardless of the magnitude) unless it was impossible to proceed with the task (e.g., a collision with an object that blew a fuse, etc.). The tasks were randomized whenever possible to insure that variables which the experimenter was not aware of (e.g., particularly easy or difficult task positions, short term learning effects, etc.) could be negated. All tasks started from a prespecified position so that comparisons of location times could be made across control modes.

Before proceeding with an explanation of each task and presentation of the experimental results, a description of the nomenclature of the graphs which follow will be given:

MS	-	master/slave with force feedback
MS NO FFB	-	master/slave without force feedback
JVRC	-	joystick variable rate control
SVRC	-	switch variable rate control
SC	-	supervisory control

The first graph on the top of the page will give the task completion time as a function of the control mode and viewing condition (see, for example, Figure 5-10). Task completion times are given on the vertical axis and the manual control modes are given on the horizontal axis. The control modes increase in complexity from left to right within each viewing condition. The number above each bar gives the mean task completion time for that control condition. The lines to the left of the manual bars indicate the minimum and maximum completion times for that control mode. The second graph at the bottom of the page will give the expected errors as a function of control mode and viewing condition.

(a) Tool-Retrieval Task

The first task required the subject to start with the end effector positioned over the nut (task hub). On the experimenter's signal, the subject moved the end effector to the tool rack, obtained the tool, being sure it was properly seated in the hand, and returned to the nut. This task simulated tool-retrieval for many of the tasks listed in Section 5.1, but in particular it simulated the retrieval of an impact wrench to free a frozen nut. Figure 5-8 shows the experienced subject retrieving a tool handle from the rack (The curtain is normally closed during experiments but was left open here for visualization). An action photo of the manipulator retrieving a tool handle under computer control is shown in Figure 5-9.

The average tool-retrieval times as a function of control mode and viewing condition for both manual and computer control are shown in Figure 5-10 (see Appendix G for the individual subject means and standard deviations). The results for various control modes under mono and two-view conditions can be found to the left

Figure 5-8: Author Retrieving Tool from Rack

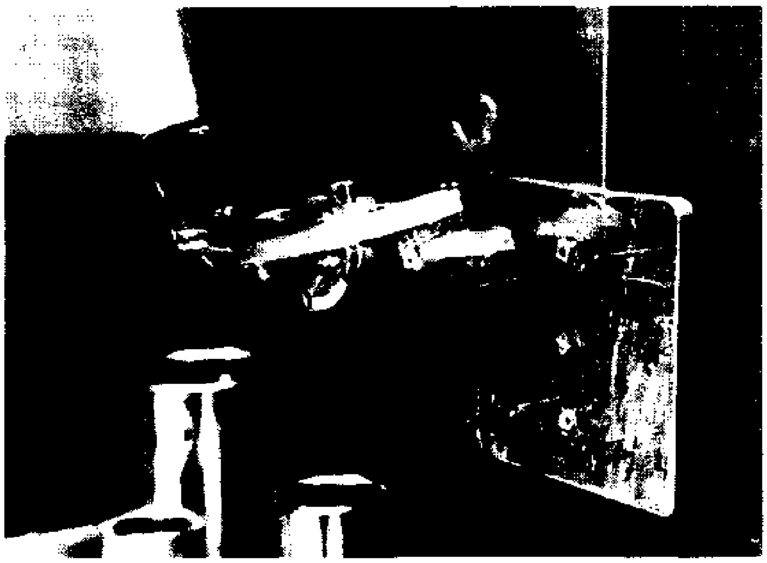


Figure 5-9: Manipulator Retrieving a Tool Handle under Computer Control

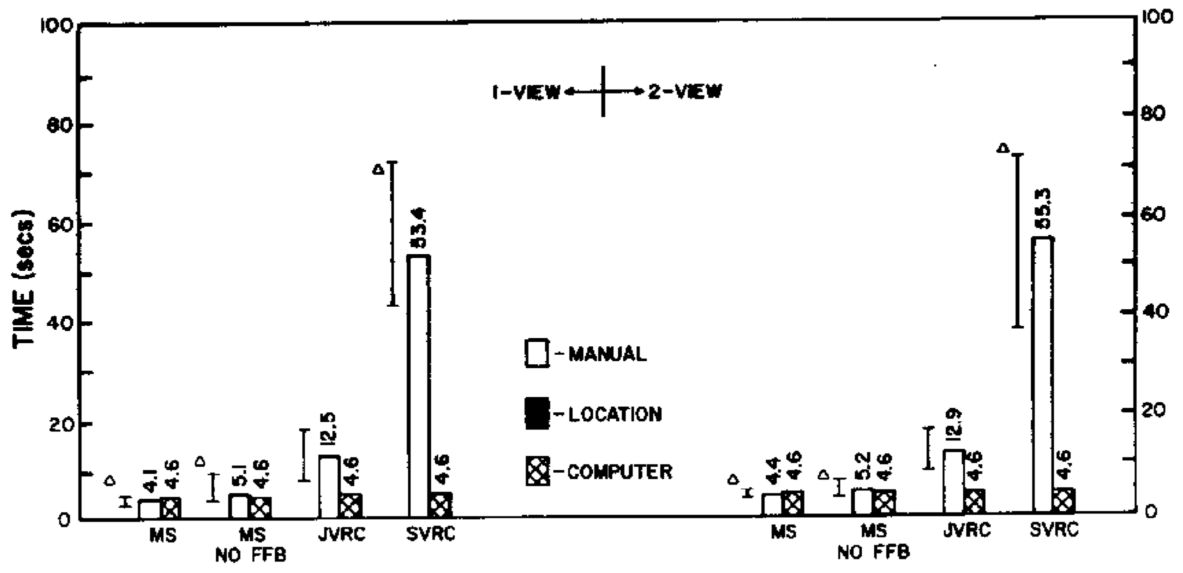


Figure 5-10: Average Tool-Retrieval Time (Each bar gives the average time of two subjects. The Δ symbol represents the mean time for an untrained subject. The capped lines show the total range of data for the trained subjects.)

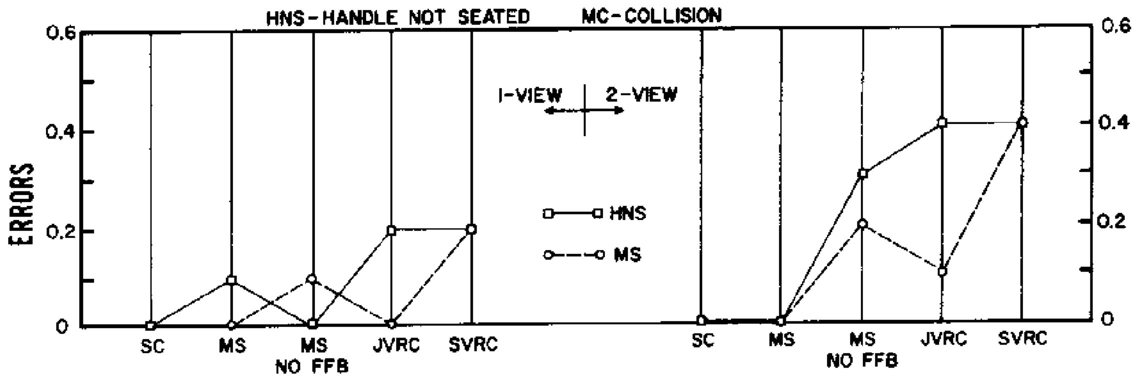


Figure 5-11: Expected Number of Tool-Retrieval Errors (Averaged over two trained subjects.)

and right of the figure respectively. Since tool-retrieval is an absolute task, the absence of manual location time is an expected result. Each of the manual control mode bars is the result of data averaged over two subjects with five trials each. For comparison, the average time over five trials for an inexperienced subject to obtain a tool is denoted by a triangle. As previously mentioned, the untrained subjects consistently averaged higher than the maximum time for any of the trained subjects (Note, the one exception to this observation occurred during this task under switch rate control (SVRC) with one-view, but the untrained subject was still well above the trained subject's average).

There were three kinds of errors that a subject could make while performing this task——collisions, dropping the tool, and not seating the handle in the end effector properly. The subjects were told that the success or failure of the task was measured by whether a solid connection between the tool handle and end effector was achieved (Hydraulic or electrical connections require the tool to be properly aligned in the hand). Figure 5-11 plots the mean number of tool-retrieval errors averaged over two trained subjects (The trained subjects never dropped a tool, and therefore, none of these errors is noted in the figure). This figure can be used to give a rough indication of the number of errors which can be expected from experienced operators (For the unexperienced subject's errors see Appendix G). Generally as the control complexity increased (i.e., from master/slave to switch rate) the error rate increased. Note that the errors for two views are consistently higher than one view. It is suspected that this phenomenon is a result of the shared attention and internal orientation readjustment required of the operator when switching from one view to the other. The one-view task completion times were also lower than the two-view, although the statistical significance of this was not demonstrated (i.e., significance tests were not performed).

By comparing the results between manual and computer control for each control mode and viewing condition it will be noted that supervisory control can improve the time required for tool retrieval for all but one of the control modes —master/slave with force feedback. Also, the expected errors for master/slave manipulation are only slightly higher than those under supervisory control and probably are not statistically supportable.

(b) Tool-Return Task

For the second task the subject started from a specified position (next to the task hub nut) with the tool in hand, and then on the experimenter's signal, the subject moved to the rack, replaced the tool insuring that it was properly seated, and returned to the initial position. To properly seat the tool on the rack required that both of the rack pins (see Figure 5-12 below and Figure 5-5) were engaged in the handle and that the tool was completely pushed onto the pins.

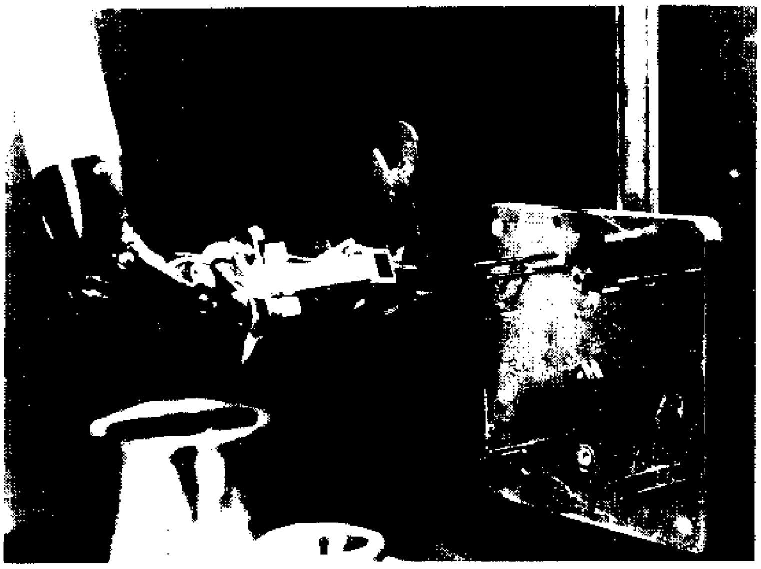


Figure 5-12: Tool Being Replaced on Rack

The average tool-return times as a function of control mode and viewing condition for both manual and computer control are shown in Figure 5-13 (see Appendix G for individual subject means and standard deviations). Again, the lines to the right of the manual control mode bars give the range over which the two trained subjects performed the task. Since tool-return is an absolute task, the absence of manual location time under computer control is not a surprise. These experiments were performed in conjunction with the tool-retrieval experiments (i.e.,

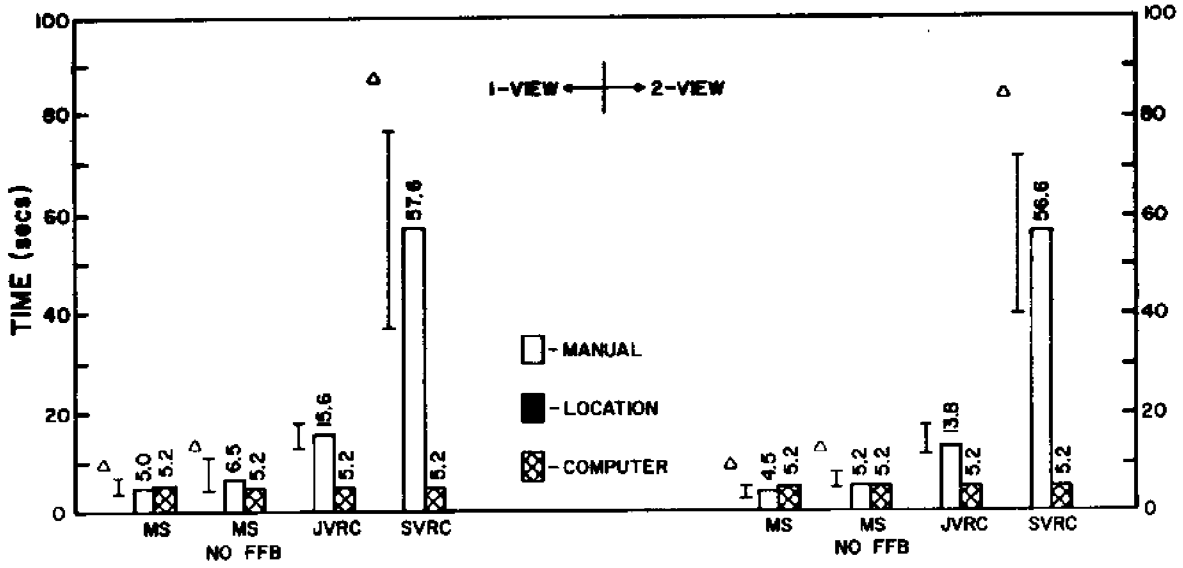


Figure 5-13: Average Tool-Return Time (Each bar represents the average time of two trained subjects and each Δ gives the mean time for an untrained subject. The capped lines represent the total range of data for the trained subjects.)

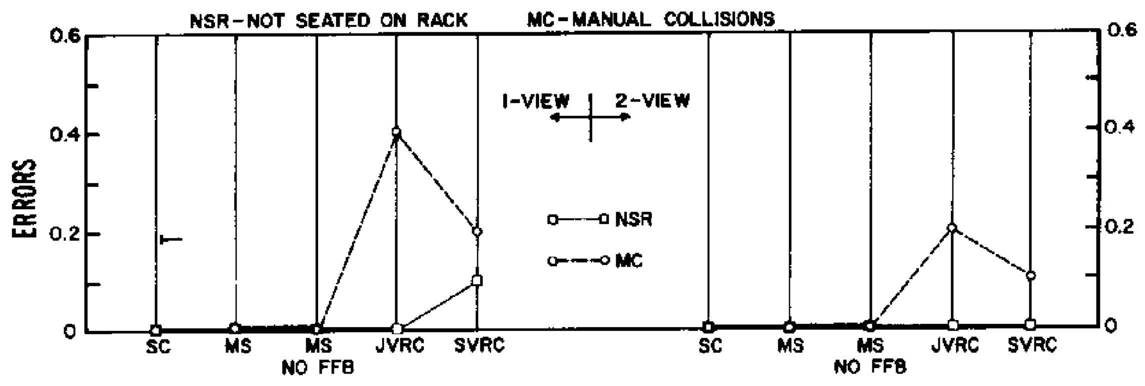


Figure 5-14: Expected Number of Tool-Return Errors (Averaged over two trained subjects.)

the tool was retrieved and then replaced), and hence, many of the experimental conditions were the same. For example, the same subjects were used for both tasks. The untrained subject's mean is again seen to be greater than the highest maximum time for any of the trained subjects.

The errors which the subjects could make during the experiments were the same as those for the tool-retrieval task (collision, dropping, and seating errors). The operators were told that the success or failure of the task was determined by whether or not the tool was properly replaced on the rack. Figure 5-14 shows the expected number of errors for the tool-return task. Generally, as the control mode complexity increased the number of collisions increased, but note the significant drop in collisions for switch variable rate control (SVRC). Also, tool/rack seating problems were not as significant as the hand/tool seating problems of the previous task. Interestingly enough, the errors for one view were higher than those for two views, the exact opposite of the result obtained for the tool-retrieval task. Also, the mono-view was slower than the two-view condition which might not be expected. Since the tool rack is stationary and the viewing conditions were not changed between experiments, the only explanation for these reversals is that the subjects were making more efficient use of the two views in the return task than in the retrieval task. To clarify, note that for the return task the subjects were required to locate two 1/8 inch pins and holes, whereas, for the retrieval task the subjects were required to locate a $\frac{7}{8} \times \frac{3}{4}$ inch tool handle with the end effector docking plate. Clearly, a second close-up view of the small pins would be more useful than a global mono view, whereas, a second view of the large handle is probably not needed as much by an experienced operator.

Even without time delay or frame rate limits the results indicate that supervisory control can improve the expected error rate and time required for tool retrieval for all of the control modes, except master/slave with force feedback.

(c) Nut-Removal Task

This experiment required the subject to position the end effector over the valve on the task hub, and then on the experimenter's signal, the subject moved the end effector from the valve to the nut, oriented the hand, and removed the nut. The general procedure used by the subjects and computer was to turn 180°, pull back to test if the nut was off, and then either reverse 180° and continue, or remove the nut.

The average times to remove the nut as a function of control mode and viewing condition for both manual and computer control are shown in Figure 5-15 (see Appendix G for the individual subject means and standard deviations). Each of the bars in Figure 5-15 is the result of the average of two trained subjects with five

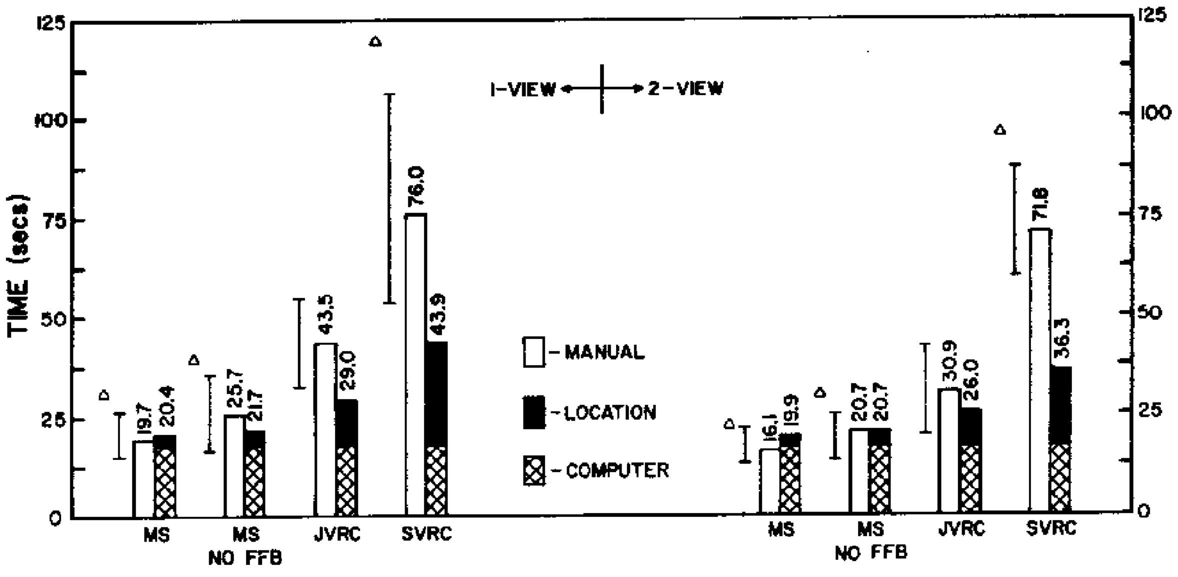


Figure 5-15: Average Nut-Removal Time (Each bar represents the average time of two trained subjects and each Δ gives the mean time for an untrained subject. The capped lines represent the total range of data for the trained subject.)

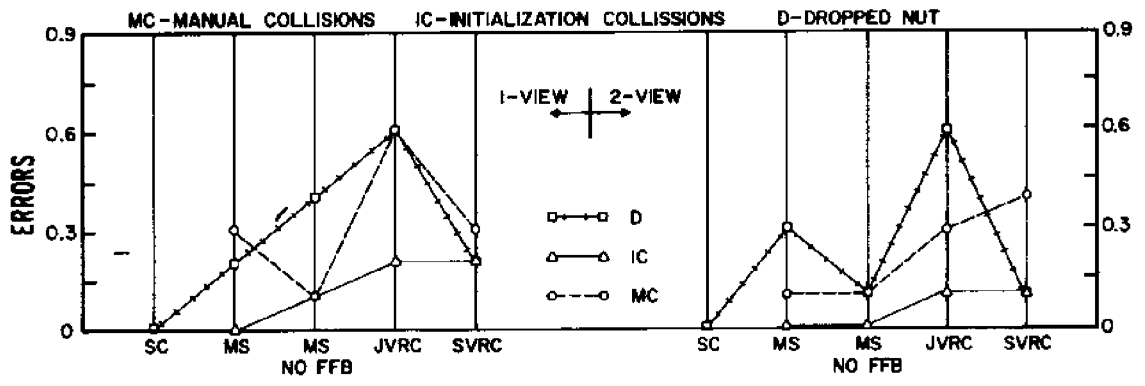


Figure 5-16: Expected Number of Nut-Removal Errors (Averaged over two trained subjects.)

trials each. For comparison, the average time over five trials for an inexperienced subject is included. The untrained subject for this task exhibited the same characteristics as the untrained subject used for the tool retrieval and return task, that is, he consistently averaged higher than the maximum time for any of the trained subjects.

Since nut removal is a relative task (i.e., the location of the nut in space is not known prior to execution), the computer time is a combination of the manual location time (g_m) and the computer execution time (f_c). To reduce some of the required experiments, two distinct times and error rates were recorded when the subjects performed the task manually: (1) the time to locate the nut; and (2) the time to take the nut off. Then, through the use of equation 5-2, the time to perform the task by supervisory control was calculated for each control mode. As seen from the figure, under computer control the proportion of time spent locating the task to the time spent executing the task increased with increasing control complexity.

There were two errors that a subject could make while performing this task——collision and loss of nut. Prior to the task, the operators were told that the task would be considered to be successfully completed if the nut could be removed without losing it. Figure 5-16 plots the mean number of errors averaged over two subjects. This figure gives an indication of the errors which can be expected from a trained operator (For the unexperienced subject's errors see Appendix G). Generally as the control complexity increased, the frequency of errors increased, although a sharp drop is noted for switch rate control (SVRC). The initialization collisions shown in the figure are the errors recorded during the manual location of the task. These errors indicate the number of collisions which would be expected to occur under supervisory control (remember that under supervisory control the task is manually located prior to execution).

Comparing the results for both viewing conditions clearly shows the advantage of the two-view system under any control mode for this task. With respect to task completion times, master/slave control with force feedback is the only control mode which did not benefit from a supervisory system. But, supervisory control improved the expected errors in all control modes.

(d) Sampling Task

The fourth task required the subject to pick-up thirteen randomly placed samples (blocks) and put them in one of two buckets according to their size (see figures in Section 4.7). This task simulated many of the common storage tasks encountered in the ocean environment (e.g., research sampling, saving a nut recently removed, etc.).

Figure 5-17 shows the average time to place one sample in a bucket as a function of the control mode and viewing condition

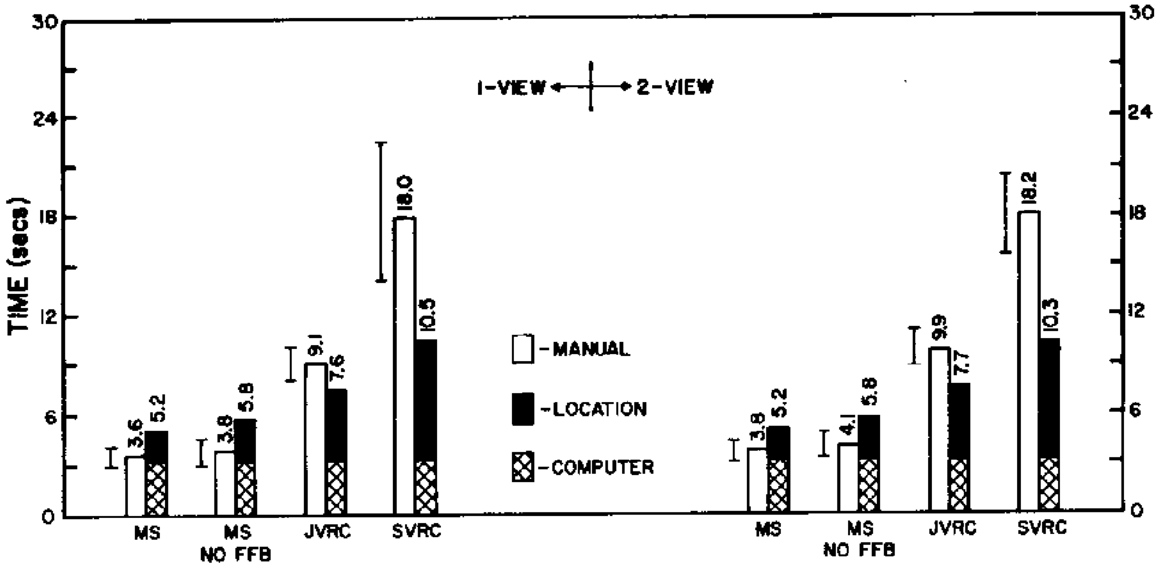


Figure 5-17: Average Sampling Time (Each bar represents the mean time of three trained subjects. The capped lines represent the total range of data for the subjects.)

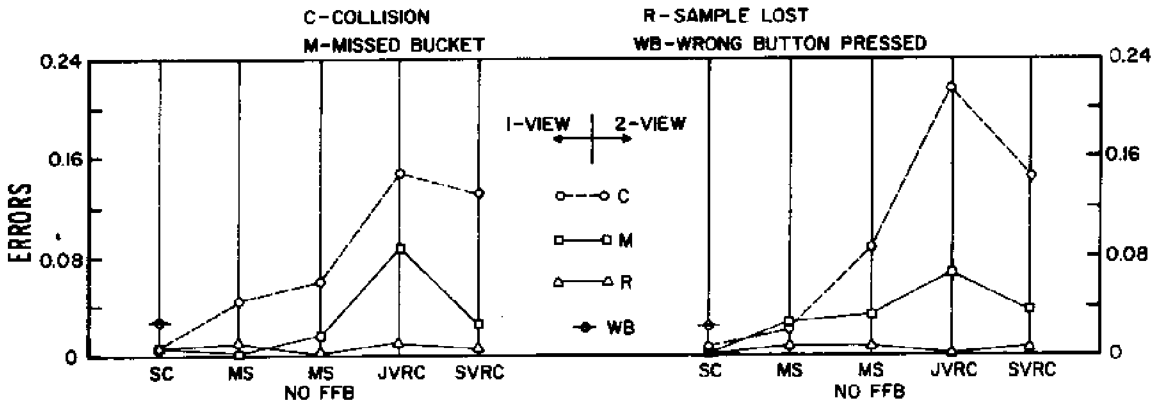


Figure 5-18: Expected Number of Sampling Errors (Mean errors for three subjects.)

for both manual and computer control (See Appendix G for the individual subject means and standard deviations). Each of the bars in the figure is the result of the average of three trained subjects with five runs each. The capped line to the left of each time-bar, gives the range of the data recorded for that manual control mode (i.e., minimum and maximum values).

Sampling is a combined relative and absolute task, and hence, the overall location time is a combination of the sample (relative) and bucket (absolute) location times. Under manual control the proportion of the total location time which occurs during sample grabbing or bucket location is not easily determined. Therefore, the methods used by the relative tasks to reduce the number of experiments could not be applied here and independent experiments for both manual and supervisory control had to be performed. But once the times for supervisory control had been determined and the time to execute the sampling routine was known, the relative grasping subtask time could be obtained by subtracting the two times (Figure 5-2). The location time given in Figure 5-17 was determined by this method, and hence, it is the manual grasping time.

There were four errors that could be made when performing this task——collisions, missed buckets, lost samples, and pressing the wrong button. The subjects were told that their success or failure to complete the task would be measured by how many samples were successfully placed in the proper buckets. Figure 5-18 plots each of these errors as a function of control mode and viewing condition. The sampling task demonstrated the same error trends as the previous tasks (i.e., the errors generally increased as the control complexity increased, but the most complex control mode, SVRC, shows a drop from the joystick control mode, JVRC). The errors noted in the supervisory control column are the average errors over all control modes. Since the subjects were required to pick-up 13 samples in rapid succession, an interesting error appeared when using supervisory control — on occasion the subjects pressed the wrong button, sending the sample to the wrong bucket.

It would appear from the results that both one and two-view conditions are equally suited for this task. The task completion times indicate both forms of master/slave manipulation (with and without force feedback) can perform sampling tasks faster than supervisory control. Obviously, force feedback is not required for this task. The expected errors under supervisory control are less than manual control.

(e) Open/Close Valve Task

This experiment required the subject to position the end effector over the nut on the task hub, and then, on the experimenter's signal, the subject moved to the valve, oriented the hand, and opened or closed the valve as required (opening and closing tasks were switched after each experiment). Since this is a relative task, the location time and the task performance time were recorded separately, when the subject performed the task manually, to reduce the number of required experiments.

The average time to open or close the valve as a function of control mode and viewing condition for both manual and computer control is shown in Figure 5-19. Each of the bars in the figure represents the average of five experimental runs using one subject. The lines to the left of the manual task times give the minimum and maximum completion times under manual control.

The only error the subject could make was to allow the slave arm to collide with its environment. Task "success or failure" was not measured since the subject was required to continue until the task was finished. Figure 5-20 shows the mean number of collisions averaged over each of the five experiments. The initilization errors shown in the figure are the errors recorded during the manual location of the task. As mentioned for the nut-off task, these errors indicate the number of collisions which would be expected to occur under supervisory control. The decrease in manual collision errors for joystick and switch rate control, compared to master/slave without force feedback, can be explained by the fact that the rotational degrees of freedom (α,β, and γ) under rate control are independent of the translational degrees of freedom (x, y, and z). Therefore, once the valve was properly located under rate control, the translational degrees of freedom could be

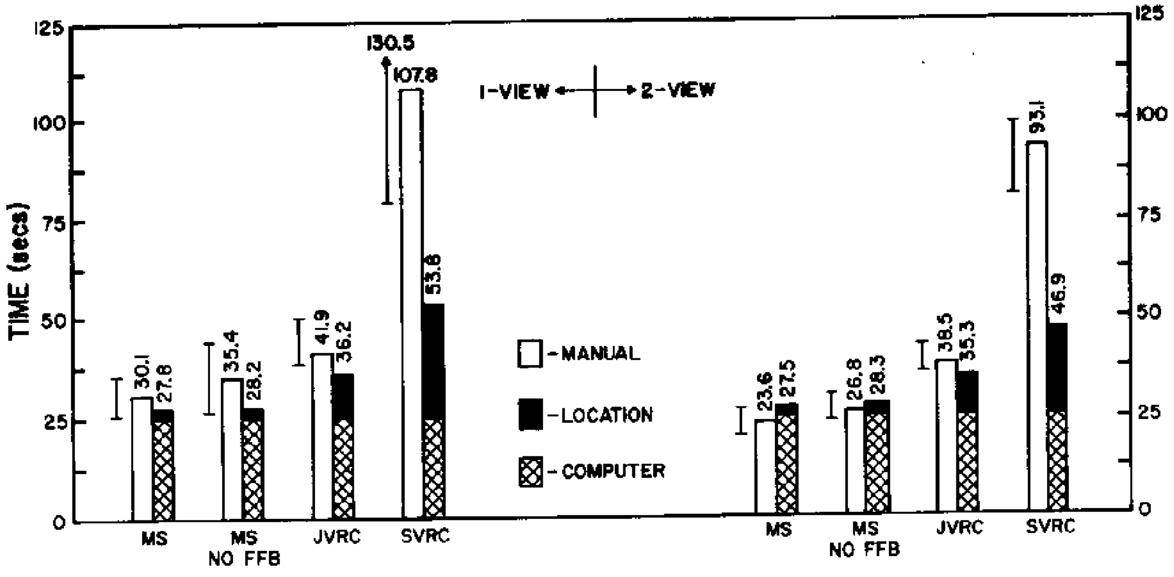


Figure 5-19: Average Valve Operation Time (Each bar represents the average time for one subject to open or close a valve. The capped lines represent the total range of the subject.)

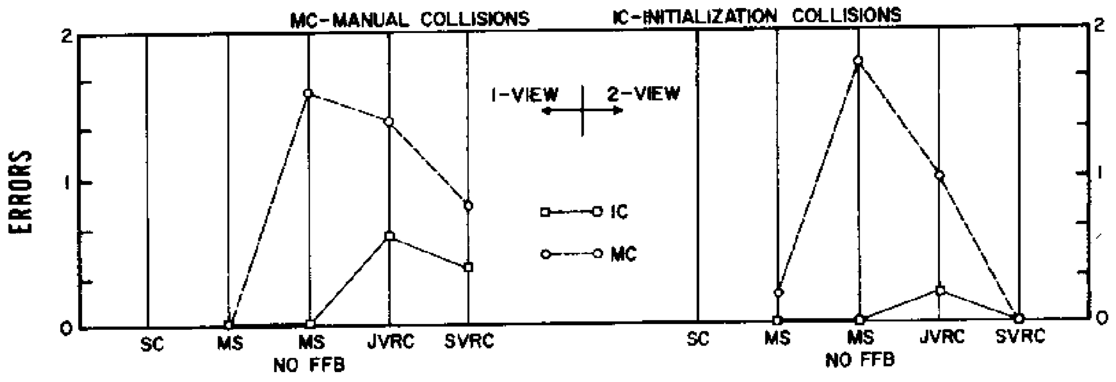


Figure 5-20: Expected Number of Valve Operation Errors

frozen and the operator was only required to focus his attention on the remaining rotational degrees of freedom. This is clearly indicated by the fact that there are no manual location errors under master/slave without force feedback, and hence, all the collisions must have occurred during the turning portion of the task. The reduced errors for the switch mode over the joystick mode can also be explained by the same reasoning, since the switch mode locks all of the degrees of freedom except the one in use, which in this case happens to correspond to the rotational axis of the valve.

As would be expected, the two-view condition is faster and shows fewer errors than the mono-view. Supervisory control was faster than any manual control mode under mono viewing conditions, but is seen to benefit only the more complex control modes under two-view conditions. The error rates of the supervisory control modes show significant reductions over those of the purely manual control modes.

(f) Digging Task

The final task required the subject to remove a specified amount of soil from a box by filling a bucket with a shovel (Figure 5-21). This task is a combination of two relative subtasks and one absolute subtask: (1) the shovel is placed in position to remove the soil in a relative manner, (2) the shovel is pushed into the soil and lifted out in a relative manner, and (3) the movement to the bucket and dropping of the soil is absolutely defined in space. Under supervisory control, the positioning of the shovel is clearly a

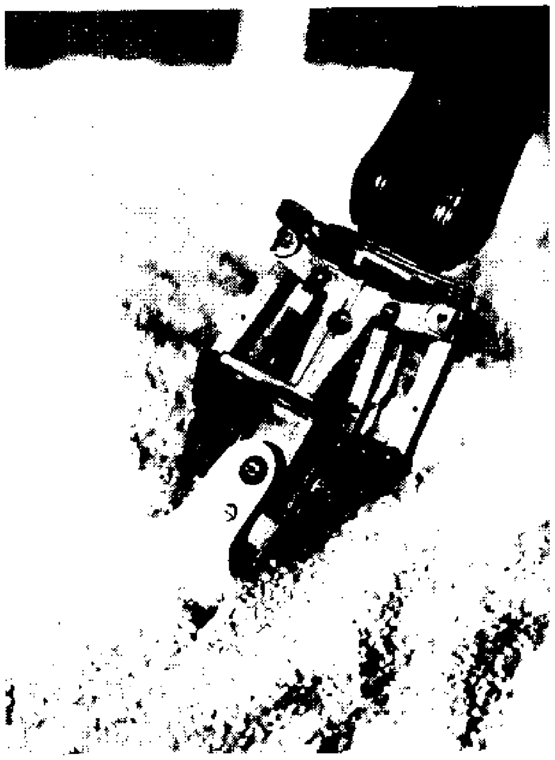


Figure 5-21: Shovel Scooping-Up the Plastic Beads Used to Simulate Marine Soil

manual task (i.e., the operator decides where and when to dig),but the relative scooping actions and absolute dropping actions are easily executed by the computer.

Figure 5-22 shows the average time to place one shovel load of soil in the bucket for various control modes and viewing conditions (see Appendix G for subject means,standard deviations and error counts). Each of the manual control bars in the figure is the result of five experimental runs which required approximately nine

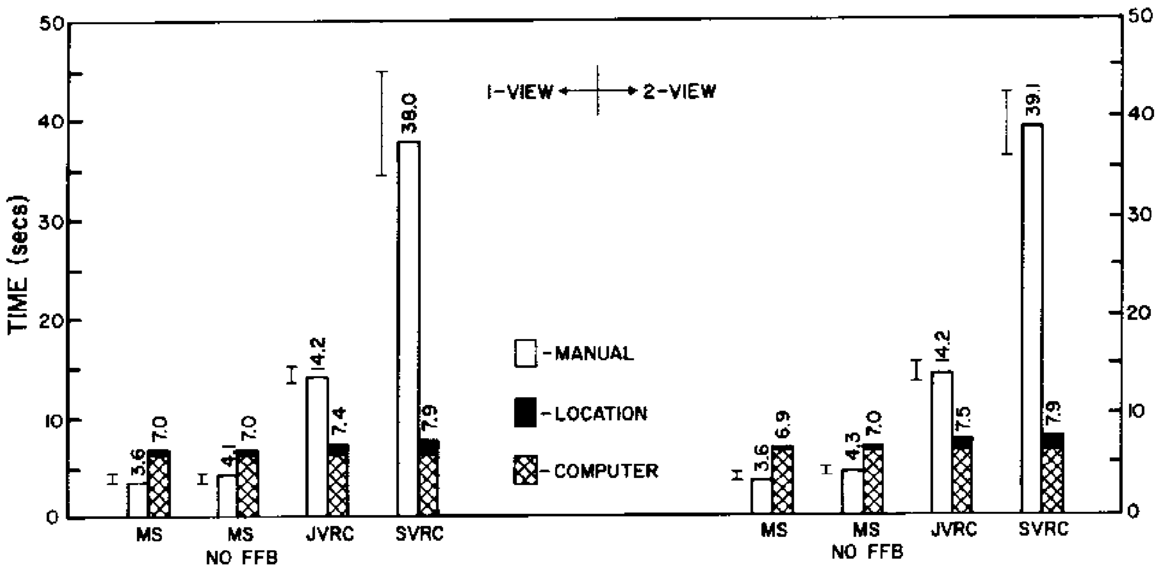


Figure 5-22: Average Digging Time (Each bar represents the mean time for one subject to obtain a shovel full of soil and place it in the sample bucket. The capped lines represent the total range of data.)

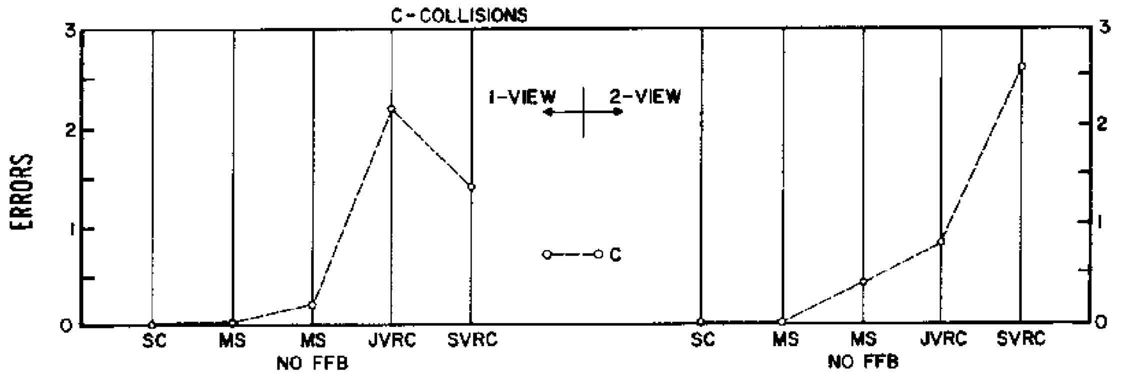


Figure 5-23: Expected Number of Digging Errors

shovel loads to fill the bucket. The mean supervisory control time for each control mode was obtained by one experimental run which required nine automatic scoops to fill the bucket. The manual positioning of the shovel is seen to be a small fraction of the total task under computer control, and hence, the variation of these times were considered small enough to warrant only one run through the task (remember, the computer execution time is constant for all practical purposes).

Collision with the environment was the only error which occurred during this experiment. Task "success or failure" was not measured in this task as the subject was required to continue until the bucket was full. Figure 5-23 plots these errors as a function of control mode and viewing condition. The digging task demonstrated the same error pattern as the previous tasks, that is, the number of collisions increased with increasing control complexity. None of the supervisory control runs showed any errors.

The results of this task can be directly compared to those obtained for the sampling task. For example, both one and two-view conditions proved to be equally suited for the task. Both tasks also demonstrated that master/slave with and without force feedback can perform faster than supervisory control. These similarities are hardly surprising, as the digging task is really nothing more than an involved sampling procedure (i.e., dig-in instead of grab). The expected errors for supervisory control were less than those for manual control except master/slave.

5.5 Summary of Results

Both the task completion time and the number of errors increased with control complexity for all of the tasks. Viewing conditions (mono and two-view) appeared to affect tasks which required precision movements (e.g., return tool, nut-off, and open/close valve), but had little or no affect on the less precise tasks (e.g., sampling and digger). For many of the tasks a sharp decrease in errors was noticed between joystick and switch rate control. This effect is attributable to the increased attention and care each operator exhibited during switch rate control modes (To move from point A to point B requires considerable thought and effort with switch rate control, but under joystick rate control the desired movement only requires a push on the stick). In some tasks the reduction in errors from joystick to switch rate control can be attributed to the coincidental matching of the task degrees of freedom and control degrees of freedom (e.g., valve and nut-off).

Table 5-1 gives the ratios of task completion times for each control mode with respect to the "best" control case, master/slave with force feedback. The ratios are given for each subject, task and viewing condition. The untrained subjects are denoted by U1 and U2, the trained subjects are denoted by T1, T2, T3 and T4, and the experienced subject is denoted by E1. The table shows a number of interesting trends: (1) the ratios increase with in-

	1-VIEW				2-VIEW				
	MS	MS NO FFB	JVRC	SVRC	MS	MS NO FFB	JVRC	SVRC	
VALVE 1-DOF	1.0	1.2	1.4	3.6	1.0	1.1	1.6	3.9	E1
NUT-OFF 2-DOF	1.0 1.0 1.0	1.2 1.4 1.3	2.2 2.2 *	4.3 3.3 3.9	1.0 1.0 1.0	1.3 1.2 1.4	2.0 1.9 *	4.4 4.5 4.3	T1 E1 U2
SAMPLER 3-DOF	1.0 1.0 1.0	1.1 1.0 1.0	2.5 2.8 2.4	5.1 5.3 4.7	1.0 1.0 1.0	1.1 1.0 1.1	2.7 2.5 2.6	4.4 5.1 4.9	T2 T3 E1
SCOOPER ?-DOF	1.0	1.1	3.9	10.6	1.0	1.2	3.9	10.9	E1
RETURN-TOOL 6-DOF	1.0 1.0 1.0	1.1 1.4 1.4	3.6 2.7 *	10.7 12.2 9.0	1.0 1.0 1.0	1.2 1.3 1.4	3.4 2.7 *	12.1 13.2 9.7	T4 E1 U1
GET-TOOL 6-DOF	1.0 1.0 1.0	1.1 1.4 1.4	3.8 2.4 *	13.0 13.0 8.3	1.0 1.0 1.0	1.2 1.2 1.3	3.3 2.6 *	12.1 13.2 11.8	T4 E1 U1

Table 5-1: Ratio of Time to Perform Task Under Given Control Mode to Time to Perform Task Under Master/Slave with Force Feedback

creasing control complexity, (2) the ratios are approximately constant across subjects (both trained and untrained) within a given task and viewing condition, and (3) the ratios are not constant across tasks (The tasks have been arranged in the table so that the ratio increases as the page is read from top to bottom). A number of other investigators have found similar trends.[8,51,52]

It is interesting to examine each task in terms of its major functions. For example, the only movement required to open or close a valve after the task has been located is a turning motion. The nut-off task, on the other hand, not only requires a turning motion, it also requires a pull to see if the nut has come off. The valve task could be classified as a one degree-of-freedom task once the valve has been located, whereas the nut-off task could be classified as a two degree-of-freedom task once the nut has been found. Similarly, the sampling task requires three degrees-of-freedom to both locate the task and drop it in the bucket. The tools were designed so that all six degrees-of-freedom are required to remove or replace a tool (if any of the arm joints are fixed, the tool cannot be obtained unless by coincidence the degree of freedom was fixed in the proper position).

Hence, from these observations it is seen that the degrees of freedom required to perform the major functions of the task indicate the task difficulty (i.e., the relative position in Table 5-1). Conversely, the position of a task on the chart (i.e., task difficulty) indicates the degrees of freedom used to perform the task. The ratios given in Table 5-1 have been plotted in Figures 5-24 and 5-25 as a function of control complexity.

It is important, at this point, to note that the task difficulty is not the same as task complexity. It is not the task degrees-of-freedom which determine the task difficulty, but the actual degrees-of-freedom required by the operator to complete the task. For example, imagine a task which requires an operator to make a complicated six degree-of-freedom motion in space, and by coincidence that same motion can be performed by moving only one joint. Are both tasks equally difficult and complex? As far as the operator is concerned the latter task is easier than the former, but from the point of view of task complexity the task has not changed.

Unfortunately, defining difficulty only in terms of the degrees-of-freedom used to perform the task is too simplistic. For example, in both figures the index of difficulty for the sampling task (3 DOF) is seen to be greater than the nut-off task (2 DOF) for the rate control modes, but less for the nut-off task (2 DOF) under master/slave control without force feedback. Clearly, the index of difficulty is a function of more than just the spatial degrees of freedom. The explanation for this is simple —— picking up blocks does not require force feedback, and hence, the ratio reflects this. Since task difficulty is a relative ratio of task completion times,

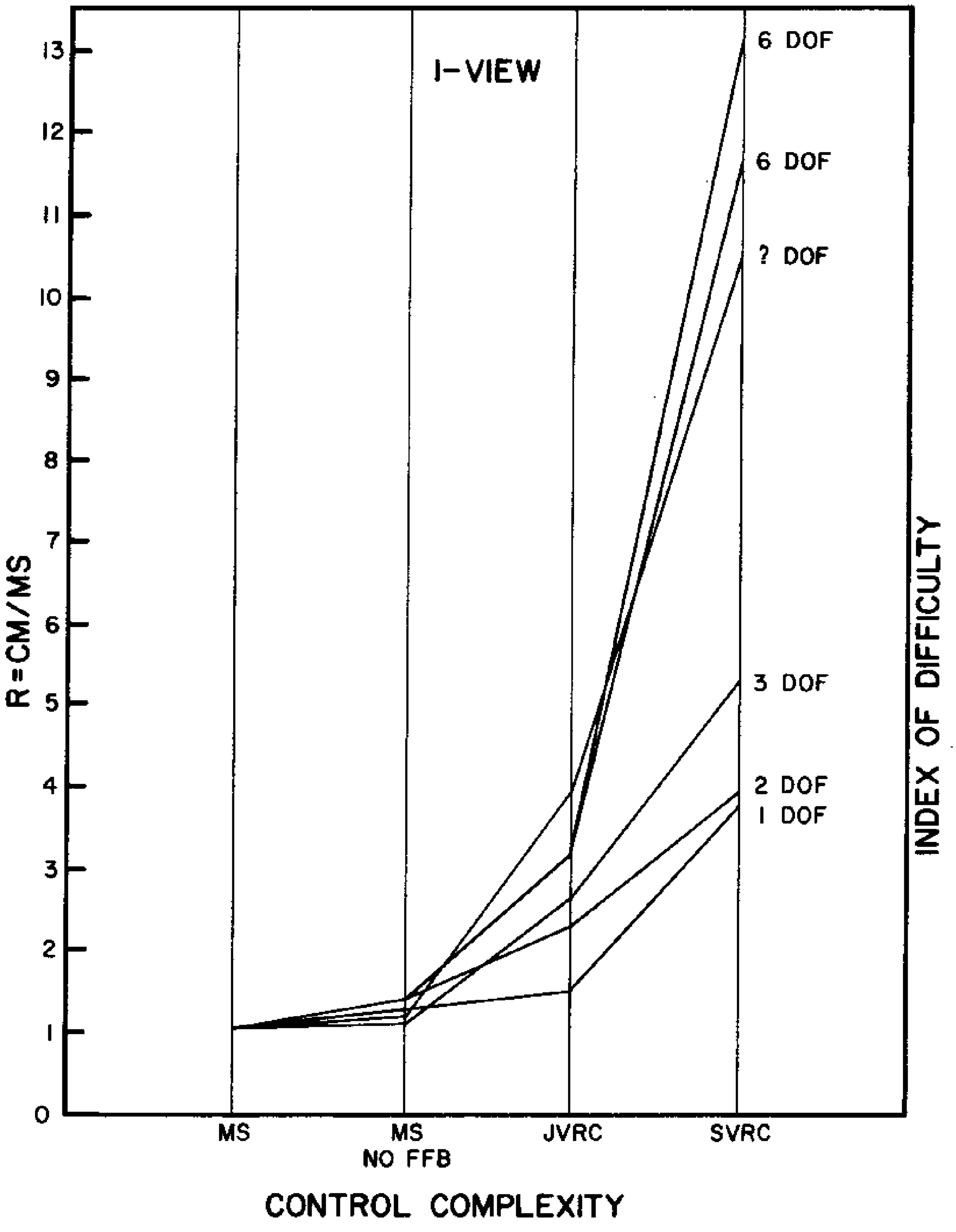


Figure 5-24: Control Mode Ratios and Index of Difficulty Versus Control Complexity for One View

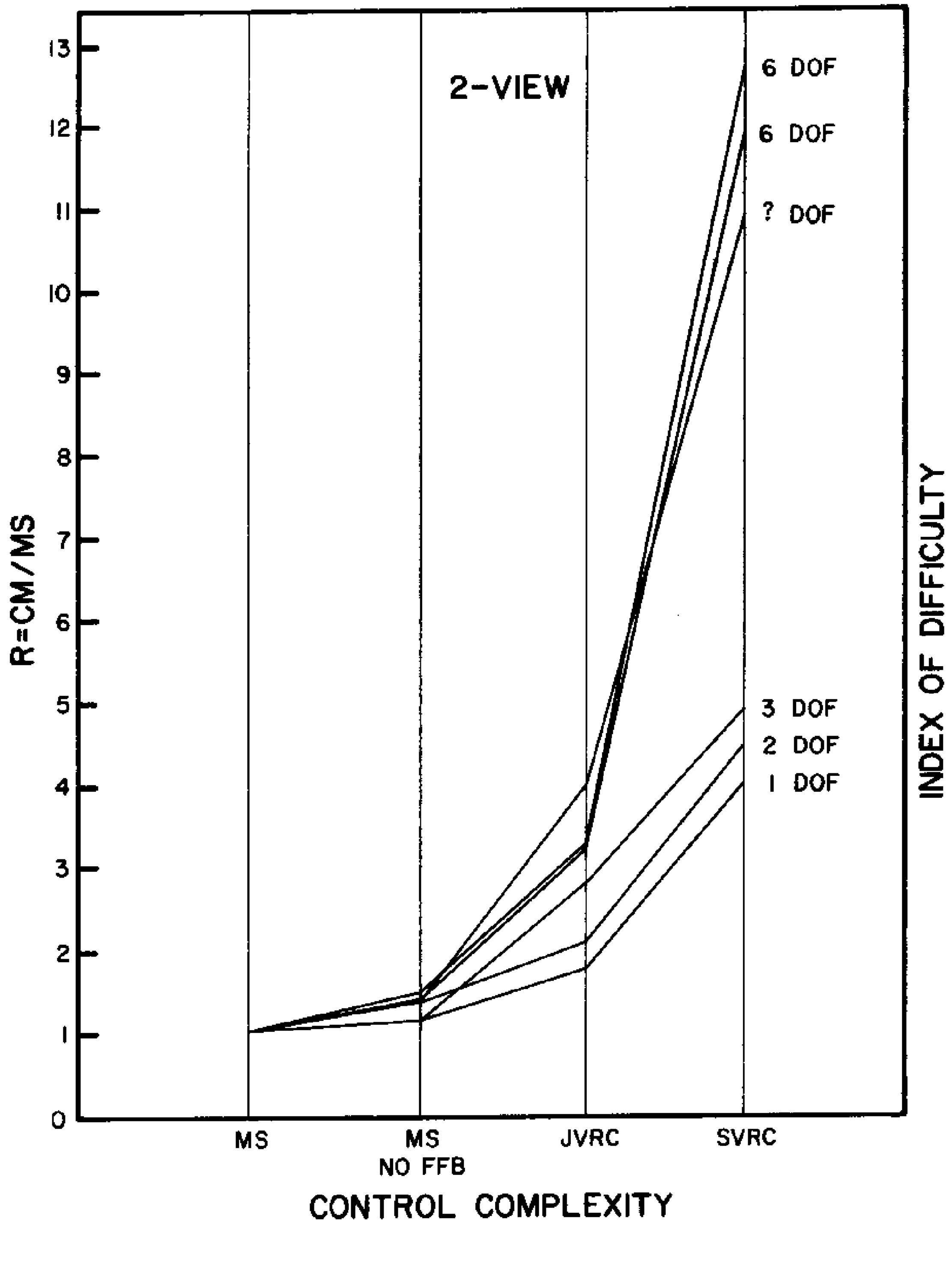


Figure 5-25: Control Mode Ratios and Index of Difficulty Versus Control Complexity for Two Views

any factor which consistently affects one task but not the other will change the relative ordering of the indexes.

Table 5-2 gives the ratios of the task completion times under manual control to the times under supervisory control. The ratios are given for each subject, task and viewing condition. The ratios relative to computer control (Table 5-2) do not show the same trends as those relative to master/slave control (Table 5-1). It is interesting to note that in contrast to the consistent ratios of Table 5-1, the computer control ratios of the untrained subjects are significantly higher than the trained subjects (i.e., untrained subjects gain more from supervisory control than trained subjects). Gains from supervisory control for any manual mode are seen to be most significant for absolute tasks (retrieval and return tool). The control mode columns clearly indicate the results of the experiments: (1) master/slave with force feedback rarely benefits from supervisory control, (2) master/slave without force feedback can profit from supervisory control in tasks which require force feedback, and (3) both forms of rate control can be aided by supervisory routines regardless of the task.

As stated previously, the tasks were designed for applicability to the marine environment and not generalized tests of task complexity. Therefore, it is impossible to define a quantitative measure of the complexity of each task. Clearly the tasks could be ordered as shown for the ratios of difficulty; but this index is

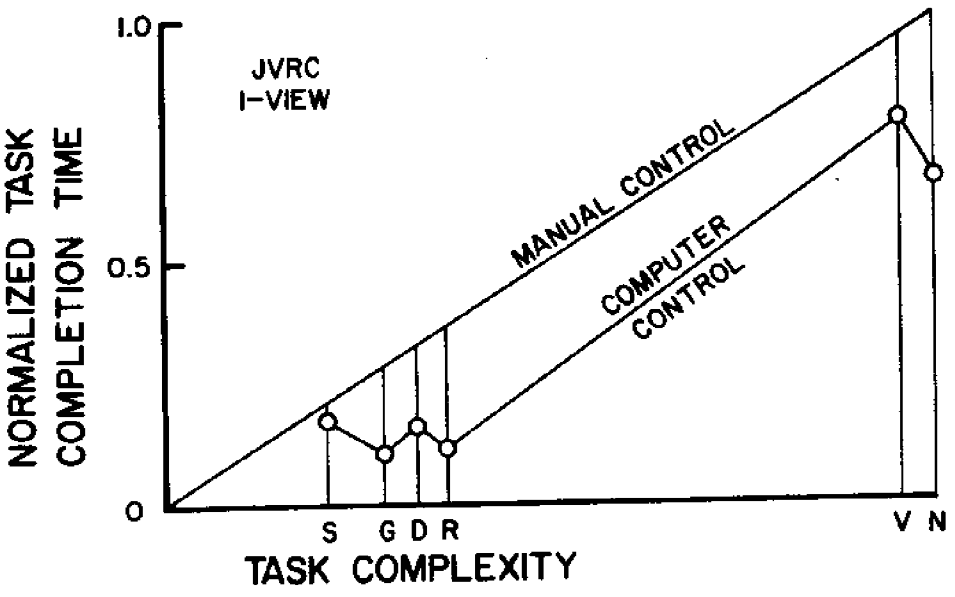
	1-VIEW				2-VIEW				
	MS	MS NO FFB	JVRC	SVRC	MS	MS NO FFB	JVRC	SVRC	
VALVE 1-DOF	1.1	1.3	1.2	2.0	0.8	0.9	1.1	2.0	E1
NUT-OFF 2-DOF	1.0 0.9 1.5	1.2 1.2 1.9	1.6 1.4 *	2.0 1.5 2.1	0.8 0.8 1.1	1.1 0.9 1.5	1.2 1.1 *	2.1 1.9 2.0	T1 E1 U2
SAMPLER 3-DOF	0.7 0.7 0.6	0.7 0.7 0.6	1.2 1.2 1.2	1.8 1.6 1.7	0.8 0.7 0.7	0.8 0.7 0.7	1.3 1.3 1.3	1.9 1.7 1.8	T2 T3 E1
SCOOPER ?-DOF	0.5	0.6	1.9	4.8	0.5	0.6	1.9	4.9	E1
RETURN-TOOL 6-DOF	0.9 1.1 1.9	1.0 1.5 2.6	3.2 2.8 *	9.4 12.8 16.9	0.9 0.9 1.7	1.0 1.1 2.4	3.0 2.3 *	10.6 11.3 16.3	T4 E1 U1
GET-TOOL 6-DOF	0.8 1.0 1.8	0.9 1.3 2.6	3.2 2.3 *	10.8 12.5 15.3	1.0 0.9 1.4	1.2 1.1 1.7	3.2 2.4 *	11.7 12.4 16.0	T4 E1 U1

Table 5-2: Ratios of Time to Perform Task Under Manual Control to Time to Perform Task Under Supervisory Control

not a true measure of task complexity. Therefore, the relationship between direct manual control and supervisory control described in Section 5.1 will have to be obtained through an indirect method.

One method of obtaining a graph such as Figure 5-1 is to assume a slope for the direct manual control line and assign an arbitrary scale of time to the vertical axis. Then, to find the complexity of the task, determine the position of each task completion time on the assumed line and descend vertically to the horizontal task complexity axis. Now plot the computer control time for the same task along this vertical line (remember, task complexity, being a function of the task only, does not change from one control form to another).

Using this procedure, graphs of task completion time as a function of task complexity (Figure 5-26) were obtained for each of the experimental tasks under joystick and switch rate control with mono viewing conditions (the remaining control modes and viewing conditions have similar graphs). The nonlinearity of the computer control curve could possibly be due to normal variations of the human subjects. But, since task complexity does not change from one control mode to another, the complete reversal of some of the tasks (e.g., nut-off and valve) on the horizontal complexity axis cannot be explained as easily. The only explanation that can be offered is that either the relationship between direct control and supervisory control is



G - Get Tool	S - Sampler
R - Return Tool	V - Valve
N - Nut-Off	D - Digger

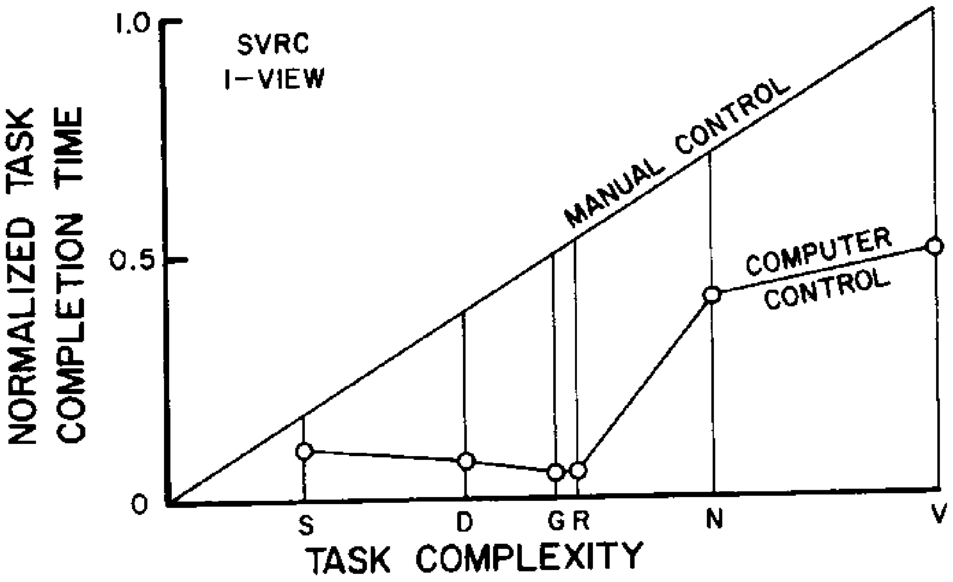


Figure 5-26: Plots of Actual task Completion Time Versus Task Complexity

not linear, as might be suggested by the qualitative plot of Sheridan and Verplank, or the horizontal axis is not task complexity but some other variable or combination of variables. The results neither verify nor negate the hypothesis, but only say that the relationship is more complicated than the plot suggested by Sheridan and Verplank.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

6.1 Conclusions

This thesis has considered the need for supervisory control of remote teleoperator systems, and has shown that supervisory systems can increase the effectiveness of remote manipulation. A mathematical foundation has been developed for performing four major classes of tasks: (1) absolute, (2) relative, (3) fixed, and (4) moving. The theoretical aspects of supervisory manipulation were covered to give the designer an overview of: (1) manipulator and processor selection factors, (2) interface design considerations, (3) language attributes and implementation factors, and (4) control philosophies. Based on the mathematical and theoretical foundations described in this thesis a supervisory system was developed and demonstrated.

The major conclusion of this study is that under the "ideal" conditions of master/slave manipulation with force feedback and real-time, undegraded, continuous viewing conditions, supervisory control offers little or no benefit. But once control complexity is increased or viewing conditions are degraded, supervisory control offers real advantages and can increase a system's capabilities. Even under the "best" control conditions (i.e., no time delays, no frame rate problems, high visability, etc.) supervisory control has been demonstrated to improve system performance for all forms of manual control except master/slave with force feedback.

6.2 Recommendations

Design recommendations have been considered in Chapters II and III, and therefore, will not be considered here. The following is a list of additional recommendations.

1. To date, moving manipulation (Chapter II) has not been investigated to determine the applicability of this form of shared control to the ocean environment. It is suggested that a simple implementation of manual control using an orientation manipulator would be warranted.*

2. It was noted that traded control between manual control modes can increase an operator's ability to perform certain tasks. For example, when performing a task that requires accurate positioning, it is best to "grossly" move the manipulator to the task by master/slave and then change to switch rate for precise movements. These forms of traded control warrant further investigation.

3. During the two-view experiments subjects occasionally had to pan and tilt the moveable camera for a better view. It would be expected that automatic slaving of the camera to the end effector position would decrease errors and completion times for two-view conditions, and therefore, this form of computer aid should be studied further (Derivations for a slaved camera are included in Appendix E).

*Remember that the orientation manipulator gives the orientation of the task with respect to the moving vehicle. Though the use of the orientation manipulator and the moving transformation equations, the hand position with respect to the task can be frozen while the computer continuously corrects the joint actuators to adapt to the changing positions between the task and vehicle —— only manual commands from the operator change the hand position (see Chapter II).

4. Whenever a number of tasks can be strung together, the only manual location time involved would be the initial set-up. Hence, the task would not be limited by the human operator's location and performance times for each of the individual subtasks and it might be expected that supervisory control would have an advantage over even the "ideal" case of master/slave with force feedback.

5. Tasks which require long periods of time to complete should be investigated to determine if operator fatigue could be reduced by supervisory control.

6. Force/torque wrist sensors should be used in lieu of sensing the joint torques and transforming to equivalent end effector forces and torques. Use of these sensors will require considerable thought to insure reliability in the hostile ocean environment.

7. Experiments to determine the effects of frame rate and resolution on task completion time and the expected number of errors should be performed under both manual and supervisory control.

APPENDIX A

INCOMPATIBILITY INDEX

Cross coupling is a phenomenon which occurs when the human operator's internal model of an anticipated motion does not coincide with the observed motion. As defined here, cross coupling will be specified in terms of the expected versus the observed end effector motion (i.e., the hand coordinates of the end effector; not the individual joint rotations of the arm). Some investigators have called this stimulus-response mismatch and orientation-display incompatibility. Although both of these terms are appropriate, the term cross coupling implies a relationship between the degrees of freedom, and hence, is felt to relay more information (remember that the applicable degrees of freedom are not the individual joint rotations but the coordinates of the hand frame).

There are three forms of cross coupling which have been identified —— mechanical, geometrical, and observational. Figure A-1 shows how these forms of cross coupling occur.

In Figure A-1a, the operator desires to move the end effector to the left. But due to mechanical cross coupling of the degrees of freedom in the control stick, the operator's command drives the end effector in a direction which is skewed with respect to the commanded motion (i.e., the actual motion and the commanded motion are not the same). The operator expects the end effector to move horizontally across the screen (see figure in operators head), but

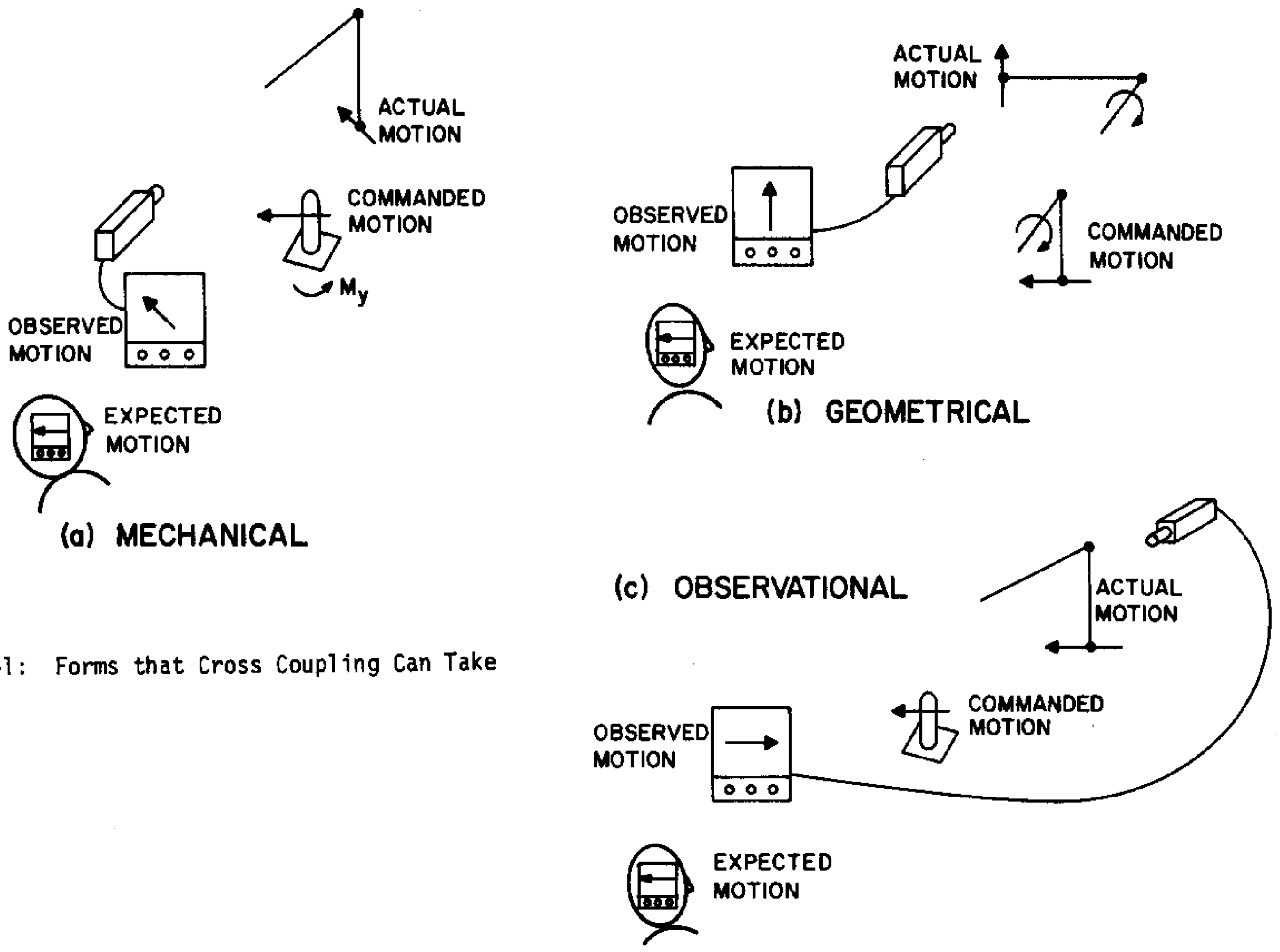


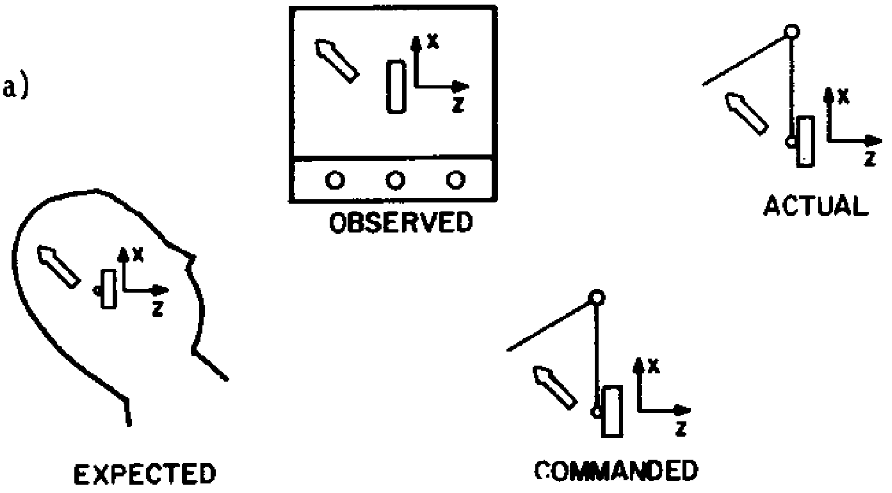
Figure A-1: Forms that Cross Coupling Can Take the observed motion is in an unexpected direction. Hence, the observed end effector motion is cross coupled with the operator's internal model of the expected motion.

In Figure A-1b, the operator desires to move the end effector to the left, but due to a geometrical difference between the master and the slave arm, the operator's command drives the slave end effector in a vertical direction. The operator expects the end effector to move horizontally across the screen (see figure in operator's head), but the observed motion is vertically upward. Again, it is seen that the observed end effector motion is cross coupled with the operator's internal model (i.e., the expected motion).

In Figure A-1c, the operator desires to move the end effector to the left, but due to the positioning of the camera the observed motion is contrary to what the operator expects. Hence, the observed end effector motion is cross coupled with the operator's internal model of the expected motion.

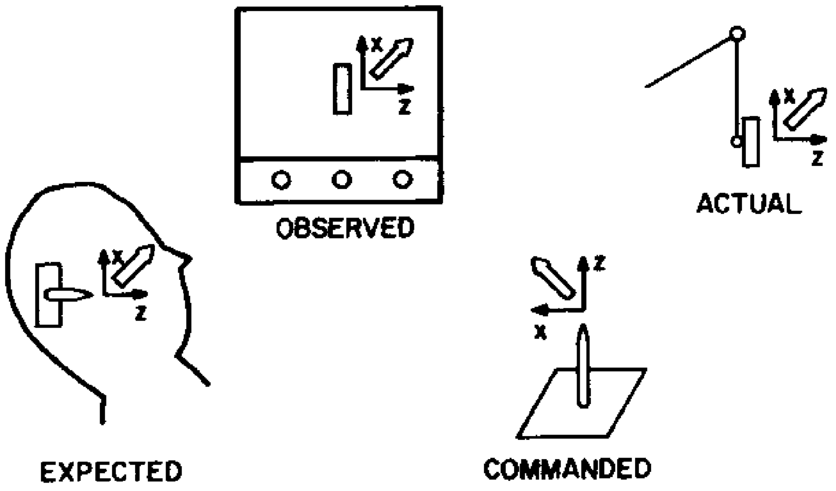
Occasionally, the expected motion is physically identifiable with the controller. At other times the expected motion is only an abstraction in the human operator's mind. As an example of these possibilities, consider master/slave control (Figure A-2a) versus resolved motion rate control (Figure A-2b). Under master/slave control (Figure A-2a), the commanded motion, the actual motion, the observed motion, and the expected motion coincide. All of the

(a)



Master/slave manipulator with end effector rotated 90° from conventional (Z axis is normally vertical). This arbitrary change makes no difference in operator's ability to control,since the expected motion is physically identifiable with the commanded motion.

(b)



Resolved motion rate control with end effector rotated 90°. Although the joystick is not physically rotated in space,as was the master hand in (a) above, the operator imagines the joystick is rotated, and hence, cross coupling does not occur.

Figure A-2: Physically Identifiable and Abstract Expected Motions

motions coincide because the slave end effector orientation is physically identifiable with the master controller orientation (i.e., the master XZ frame has the same orientation as the observed slave XZ frame). But under resolved motion rate control (Figure A-2b),the observed and actual motions will be different from the commanded direction of motion (The joystick's fixed in space and cannot be physically rotated to make the XZ frame coincide with the slave end effector XZ frame). But the observed motion is exactly what the human operator expected, since the operator imagines that the joystick is riding on the slave hand coordinate system. Clearly, the expected motion is only an abstraction in the operator's mind. But as long as the operator can maintain this abstraction, cross coupling will not occur. As another example, consider cross coupling due to a geometrical difference between the master and slave. If the human operator is aware of the geometric difference, he might be able to anticipate the required commands which would result in the desired motion, and hence, the expected motion and observed motion would be the same (i.e., there would be no cross coupling even though the arms are not geometrically similar).

At this point a simple formula which indicates the degree of cross coupling or simulus-response incompatibility will be presented without proof.

$$\mathrm{ICI} = \frac{1 - c\alpha|c\alpha|}{2}$$

where,

$$c\alpha = \frac{dr_o \cdot dr_e}{|dr_o|\ |dr_e|}$$

ICI = Incompatibility Index*

$c\alpha$ = the cosine of the absolute angle between the <u>observed</u> motion vector and the <u>expected</u> motion vector.

dr_o = the observed motion vector

dr_e = the expected motion vector

If,

$ICI = 0$	cross coupling will not occur
$0 < ICI \le \frac{1}{4}$	cross coupling will occur but the operator can compensate for the effects
$ICI > \frac{1}{4}$	cross coupling will cause the operator's performance to go unstable

This rule was observed during the takeover experiments in Appendix B, but experimental verification was not done due to time considerations. An indication of the validity of this rule has been found in the literature: "Vertut showed that if the master end of a master/slave manipulator is rotated 30° [ICI = $\frac{1}{8}$]** relative to the slave (and all other correspondences left undisturbed) the operator

*The incompatibility index was given this form so that the entire range from 0 to 180° would result in a singular function which continually increases to a maximum of one (1) as the cross coupling approaches 180° phase difference.

**The comments in brackets are mine.

could compensate, but as the disparity went beyond 45° [ICI > $\frac{1}{4}$]* performance deteriorated badly."[4] Vertut's example used one joint degree-of-freedom and only accounted for geometric cross coupling, whereas the incompatibility index advanced here is in terms of spatial coordinates and is hypothetically valid for all forms of cross coupling (mechanical, geometric and observational) and analog stimulus-response incompatibility. For example, the analog bar graph display mentioned in Section 3.3a which decreases with increasing force level would have an incompatibility index of one (ICI = 1) as long as the operator expects the display to increase with increasing force (i.e., the expect motion is an increase but the observed motion is a decrease). But, as soon as the operator adjusts to the graph and expects the graph to decrease with an increase in force, the incompatibility index drops to zero (ICI = 0).

APPENDIX B

TAKEOVER EXPERIMENTS

To investigate the mismatch, cross coupling, and transient problems discussed in the Takeover section (Section 4.6), a number of simple experiments were performed. Due to time constraints only two of the available control modes were used for these experiments—joystick rate and master/slave (These control modes are representative of the two major classes of manual control—rate and position). This study consisted of three parts; (1) a qualitative determination of the offset rate of decrease; (2) a quantitative test of speed and accuracy under a simulated emergency conditions, and (3) a qualitative measurement of accuracy and stability of each control mode. The first part of the experiment consisted of determining the rate of mismatch decrease. This was done by varying the offset rate and allowing the two experimental subjects to comment on the decrease rate compared to previous trials (Due to a time constraint, quantitative offset rate experiments and their effects on operator performance were not done). For the second half of the experiment the gain on the z axis was turned almost completely off and a weight was hung from the arm to hold a pen against a sheet of paper. The operator could control all six degrees of freedom, although the z axis was severely limited by the low gain. Each operator was allowed fifteen minutes of practice time in each mode before the experiment to become familiarized with the equipment. Then each subject was required to observe the slave manipulator traverse a random path with the instructions to take control by pulling on the master if the slave crossed into circle one (Figure B-1). After taking control the subject was required to follow an optimal path around an imaginary wall as accurately and quickly as possible (It was assumed that the weights the subjects would put on accuracy and time for one mode would be the same for the other control mode). At the end of the path the subject was required to pass as closely as possible to the center of circle three, at which time the clock would stop. After passing through circle three, the operator moved the slave to circle four and attempted to hold the pen there for ten seconds. Each subject performed the above procedure three times for each control mode. A quantitative analysis of the mean distance from the optimal path, time to traverse the path, and final positioning accuracy, as well as qualitative evidence for accuracy and stability when attempting to hold a pen in a stationary position can be found on the following pages.

B.1 Offset Decrease Rate Experiments

It has been stated in previous arguments that it is desirable to remove the mismatch so that the operator is in a complete master-slave mode as soon as possible (Section 4.6). The determination of this parameter was largely subjective, being set by comments like, "this feels better than the last offset rate." Through this method of parameter adjustment it was found that the offset rate becomes transparent to the operator if the offsets are zeroed at a rate of 1/3 volt per second (This offset rate corresponds

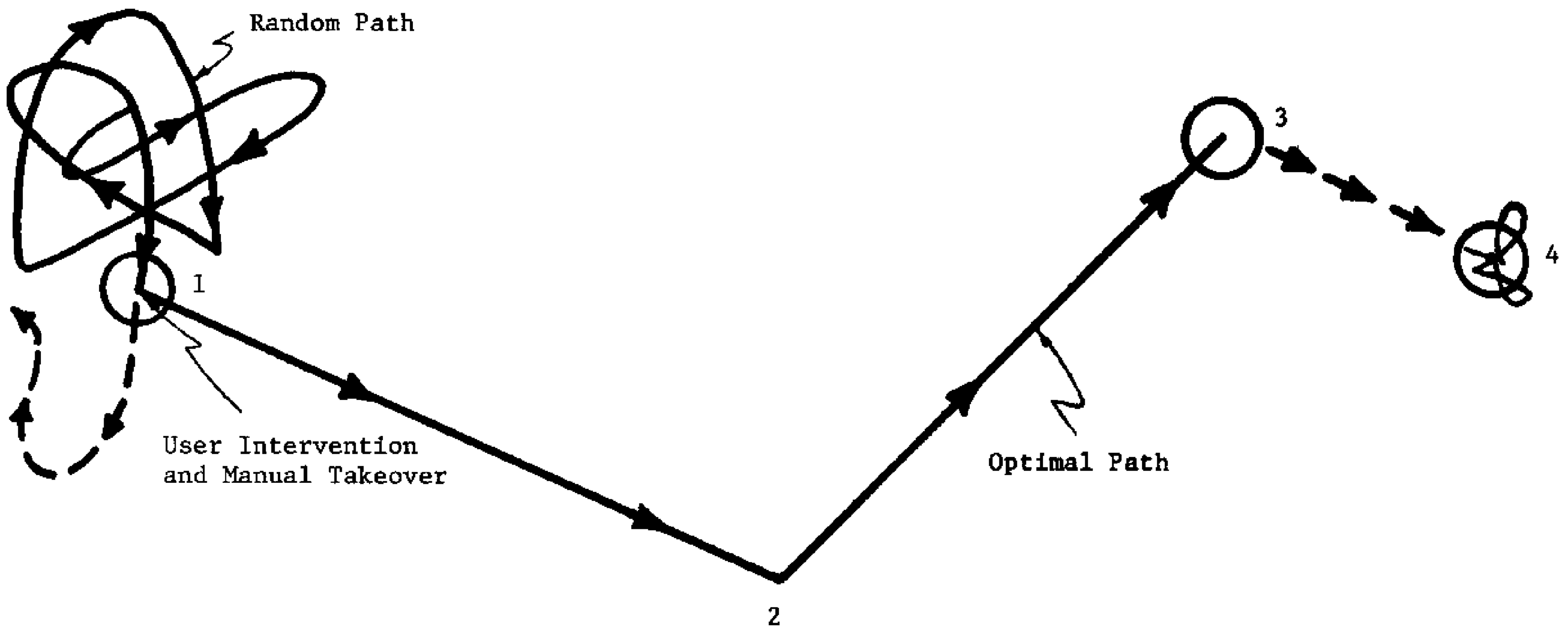


FIGURE B-1: EMERGENCY TAKEOVER PATH

to different angular rates for each degree of freedom since each degree of freedom has a different total angular displacement). Future quantitative experiments with independently adjustable angular rates are clearly needed, but for the purpose of this experiment, both operators felt at ease with the subjectively determined values (Clearly, the offset decrease rates are dependent on the manipulator geometry, and therefore, are manipulator specific).

B.2 Emergency Path Experiments

Figures B-2 and B-3 show the mean distance from the optimal path in a scaled drawing of the actual experiment for both the rate and master/slave control modes(Plots of the mean distance from the optimal path and the standard deviation of the actual paths for both experimental subjects can be found in Figures B-10 through B-14 at the end of this appendix). For both subjects, whether rate or position control, it is immediately apparent that any abrupt change in trajectory not only increases the subjects distance from the optimal path, but also their variability between experiments (i.e., the standard deviation shows increases at exactly the same positions along the curve that the mean path shows increases). For example, note that the second subject's mean distance from the optimal path in rate control (Figure B-12) has a very large initial spike after takeover and after turning the corner. The same effects can be noted in the standard deviations (Figure B-13). These effects cannot be attributed to an electronic or mechanical transient as the 7.2 inches noted for subject two or the 3.5 inches for subject one are too large

Subject #1

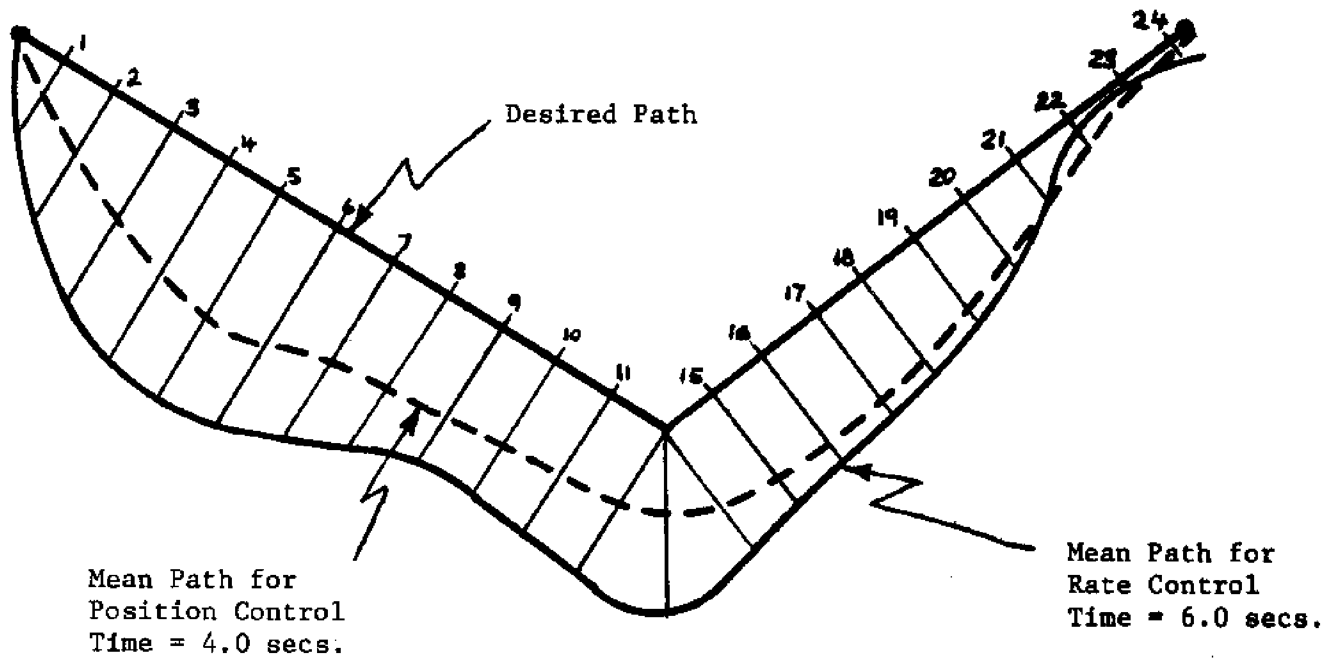


FIGURE B-2: SCALED DISTANCES FROM OPTIMAL PATH FOR SUBJECT #1

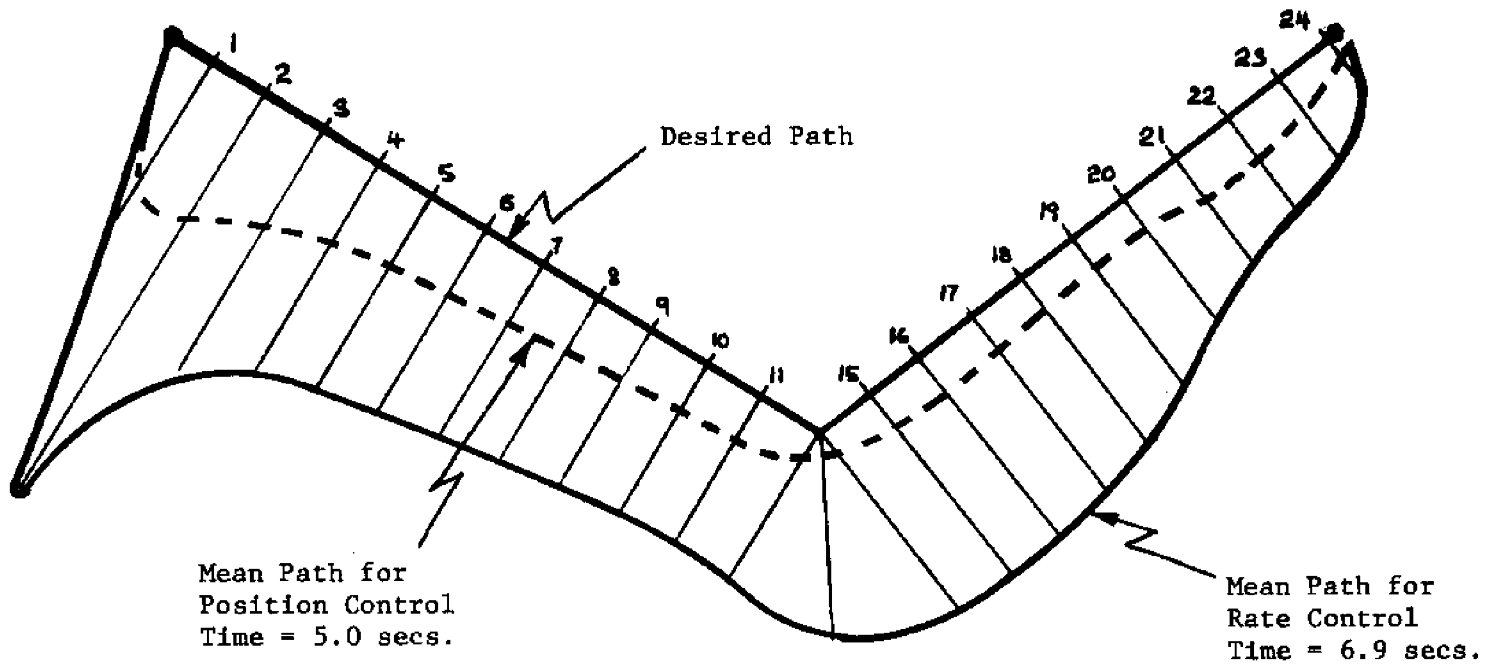

Subject #2
Desired Path
Mean Path for Position Control Time = 5.0 secs.
Mean Path for Rate Control Time = 6.9 secs.
1
2
3
4
5
6
7
8
9
10
11
15
16
17
18
19
20
21
22
23
24

FIGURE B-3: SCALED DISTANCES FROM OPTIMAL PATH FOR SUBJECT #2 to be due to these sources, and therefore, the increases must be partially due to the subjects reaction time.

As mentioned previously, the purpose of the experiment was to simulate emergency takeover, and hence, it is necessary to have time and accuracy constraints if the simulation is to represent real conditions. These constraints were represented by the experimenter's stopwatch and the optimum path, but the relative weights of these constraints were generally left unfixed (i.e., the subjects were only told to get to the end as fast and as accurately as possible). It is interesting to note that the average time for rate control for each subject is two seconds more than the average time for the offset master/slave mode (see Figures B-2 and B-3). Figure B-4 shows the time recorded for the individual experimental runs. For both subjects the time to complete the path with rate control is greater than that with an offset master/slave (note the exception for random path #4 by subject #2).

The final positioning error for the individual runs generally showed the same pattern as the mean distance from the optimal path and the average time did, i.e., the offset master-slave mode has better terminal accuracy compared to the rate mode (Figure B-5).

It should be kept in mind that this experiment does not attempt to determine which of the two, rate or master/slave, is more accurate than the other. Given enough time either one of these modes could follow the optimal path without deviation. What the experiment

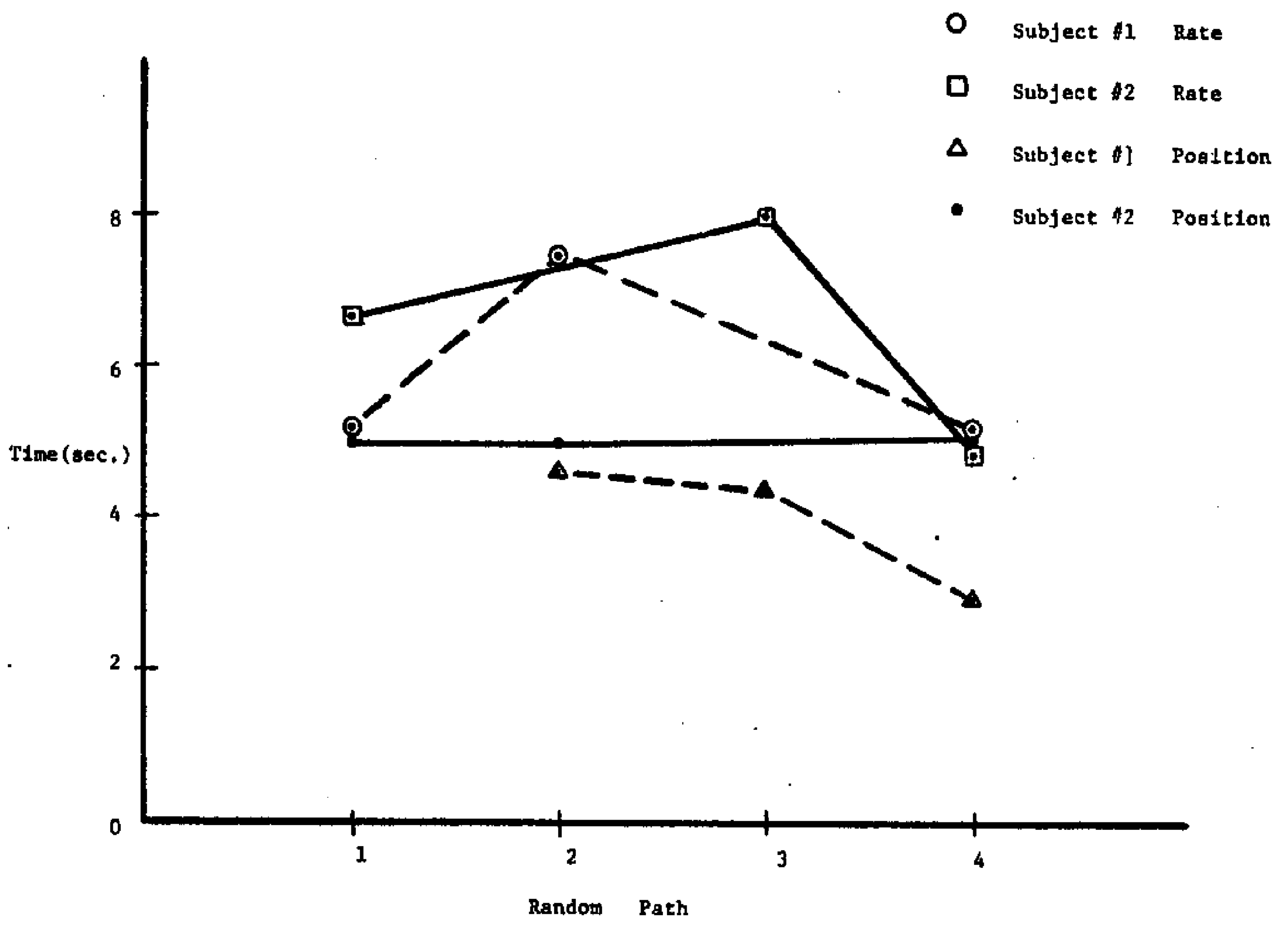


FIGURE B-4: PATH COMPLETION TIME FOR INDIVIDUAL SUBJECTS AND RANDOM PATHS

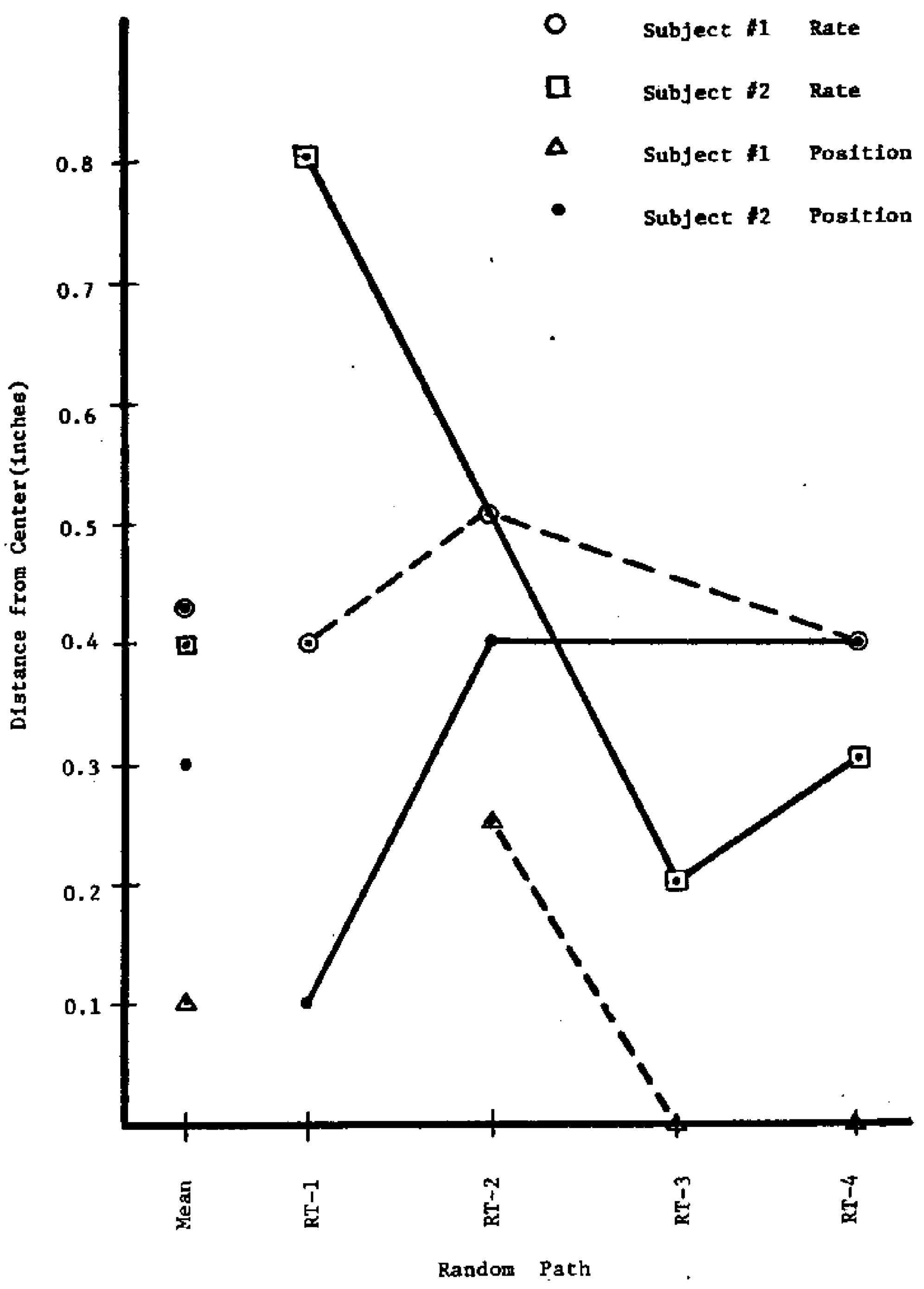


FIGURE B-5: TERMINAL PATH ACCURACY FOR INDIVIDUAL SUBJECTS AND PATHS does say is that under emergency conditions, when time and accuracy could be decisive factors, master/slave manipulation will be faster and more accurate than rate (It should be obvious that a switch controlled rate manipulator would be even less effective than the joystick controlled rate manipulator used for this experiment).

B.3 Stationary Task

Figures B-6 through B-9 show the actual recorded position of the pen for the stationary task. The results indicate that the offset master/slave for this particular experiment was more stable and accurate for both subjects (It should be noted that due to the results of this experiment, a dead zone has been added to the rate control with the result that some of the instability in Figures B-6 and B-8 has dissappeared).

B.4 Takeover Experiment Conclusions

In conclusion, a number of important findings resulted from these experiments:

1) Operators generally prefer the positional control over the rate for "accurate takeover and ease of use".

2) Operators feel that both modes should be available for use at the operator's discretion. There should also be the ability to change from one mode to another immediately, so that a combination of the two modes can be used. This combinational mode would work well for instances when there was a large disparity between the master and slave,

and hence, the affects of cross coupling would be noticed most. When this situation arises the possibility of cross coupling could be circumvented by using rate control until the orientation of the two arms were sufficiently similar that position control could be used.

3) Rate control should have a variable scale and the joystick should have a deadzone to prevent drift. The original experiment only allowed the subject to vary the slave's rate through the joystick, so that the rate was continuously variable from 0 to 20 inches per second for a movement of 0 to 5 inches of the joystick. From this experiment it was discovered that a variable rate-scale setting, which allows the maximum rate for 5 inches of joystick travel to be set independently of the joystick, will increase the accuracy at a small expense in rate. This allows the operator to move toward a target at rates up to 20 inches per second, but as the target approaches, the operator can adjust the rate-scale so that the sensitivity of the joystick is greater but at the cost of the maximum rate.

4) The question of offset decrease rate is still unanswered, but subjective results appear to be adequate

for preliminary studies. Further investigation would seem to be warranted.

The built in capability for the human operator to take control directly and immediately from the computer appears to enhance the total system and is felt to be justified for any computer controlled manipulator.

FIGURE B-6: TEN SECOND STATIONARY TASK FOR SUBJECT #1 WITH RATE CONTROL

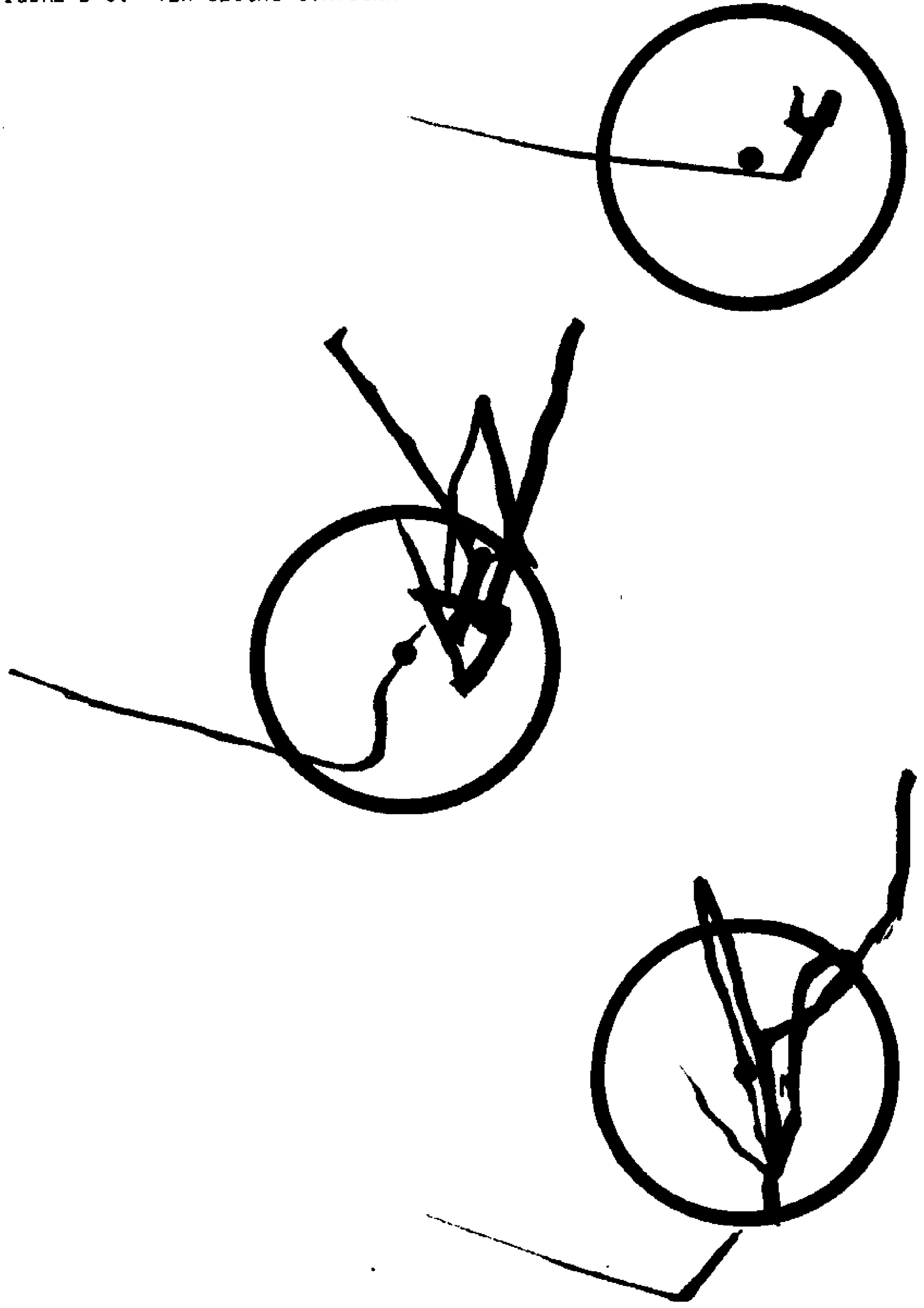


FIGURE B-7: TEN SECOND STATIONARY TASK FOR SUBJECT #1 WITH POSITION CONTROL

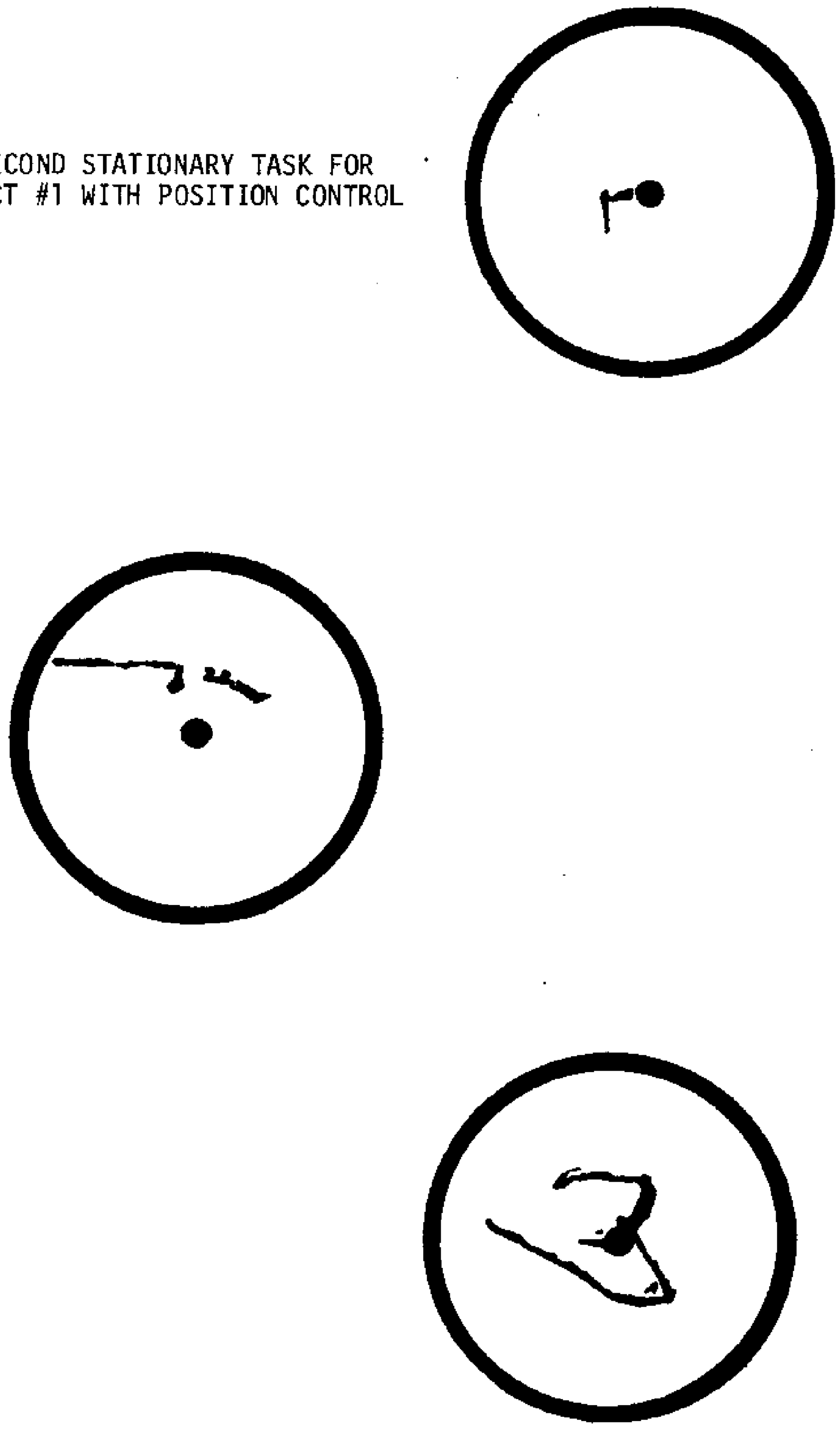
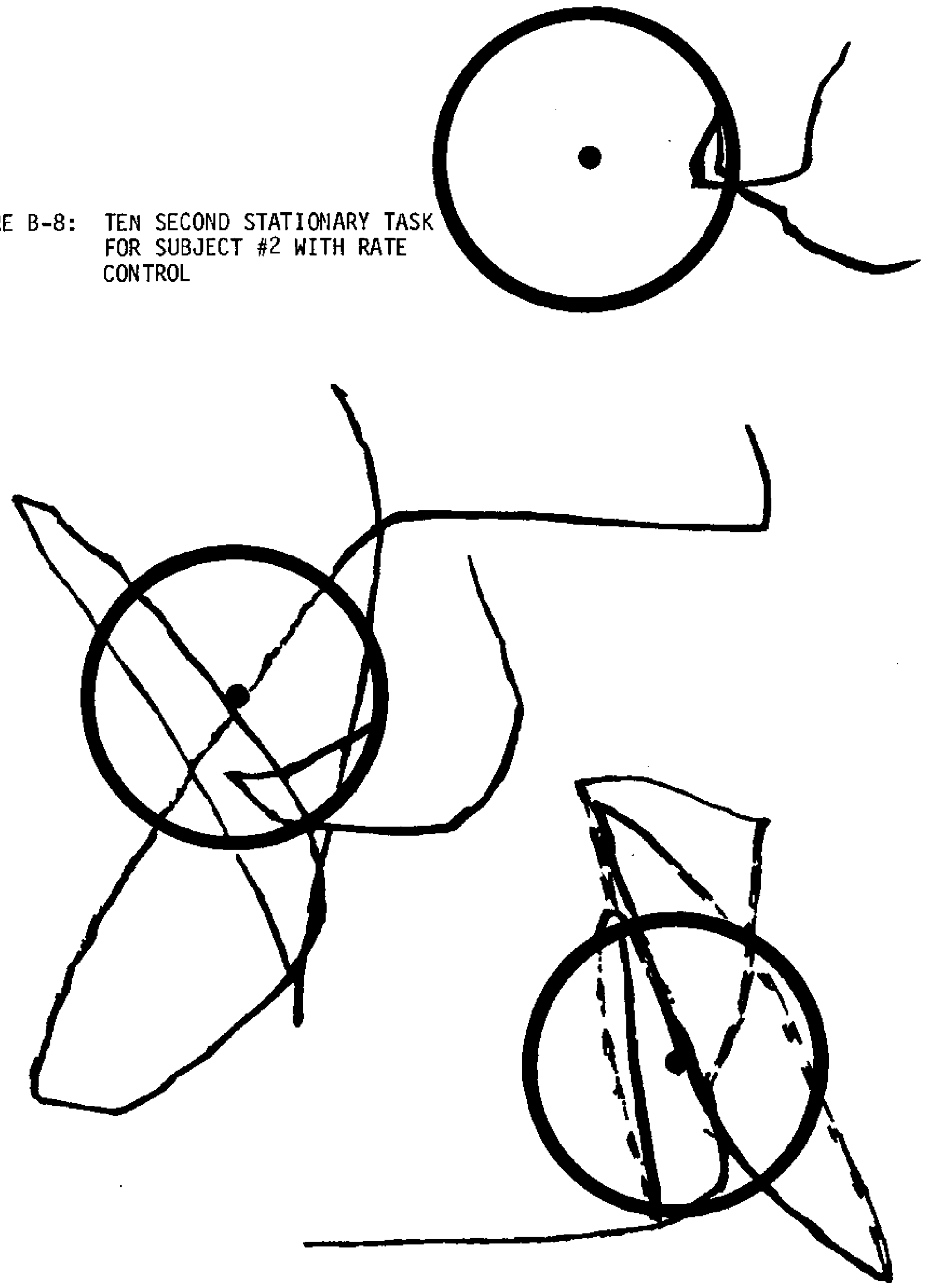


FIGURE B-8: TEN SECOND STATIONARY TASK FOR SUBJECT #2 WITH RATE CONTROL



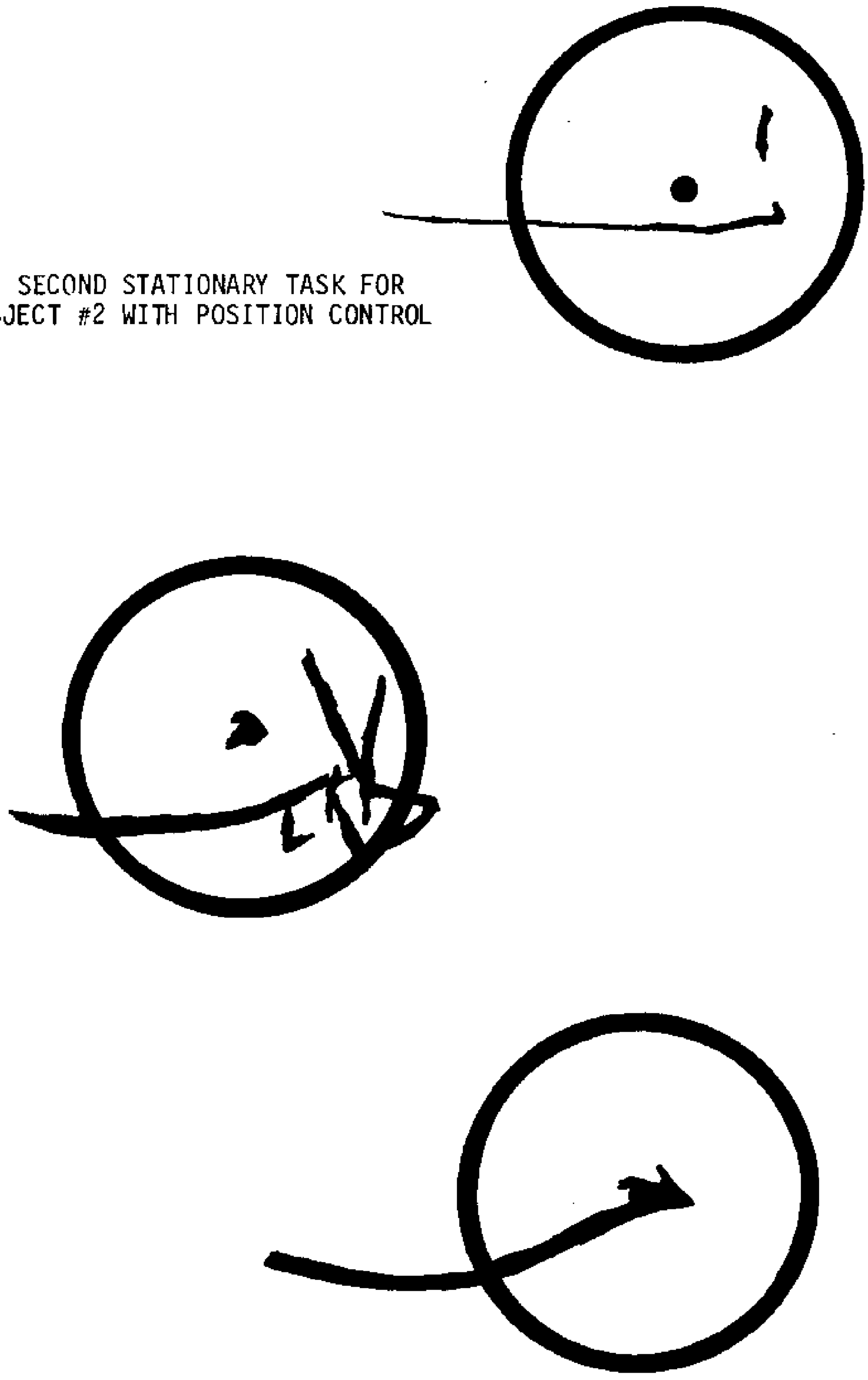
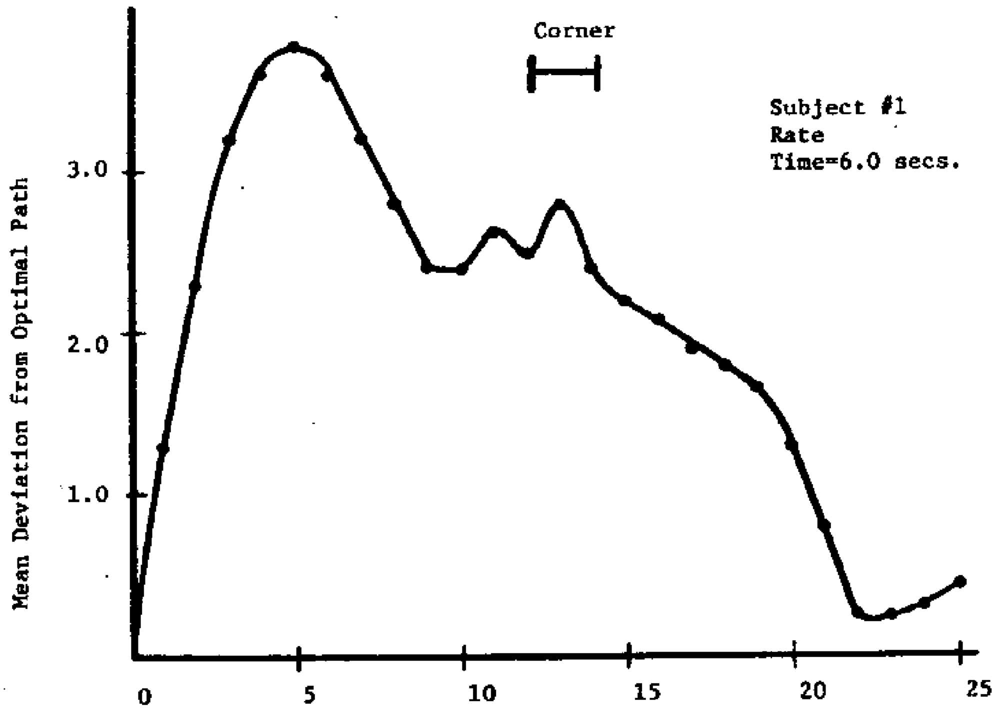


FIGURE B-9: TEN SECOND STATIONARY TASK FOR SUBJECT #2 WITH POSITION CONTROL



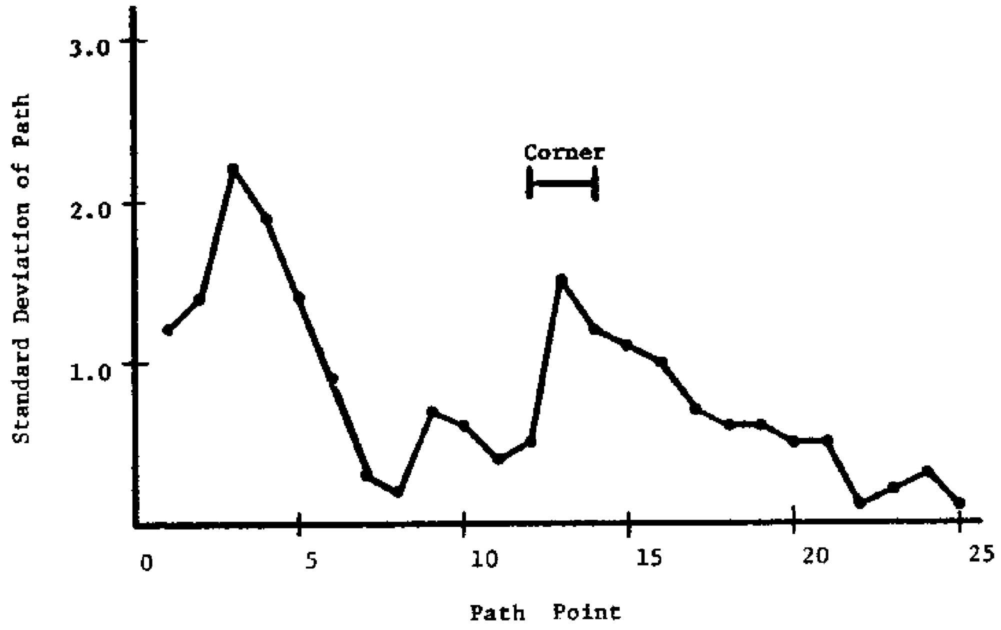


FIGURE B-10: MEAN DISTANCE FROM OPTIMAL PATH AND STANDARD DEVIATION OF MEAN PATH FOR SUBJECT #1 UNDER RATE CONTROL

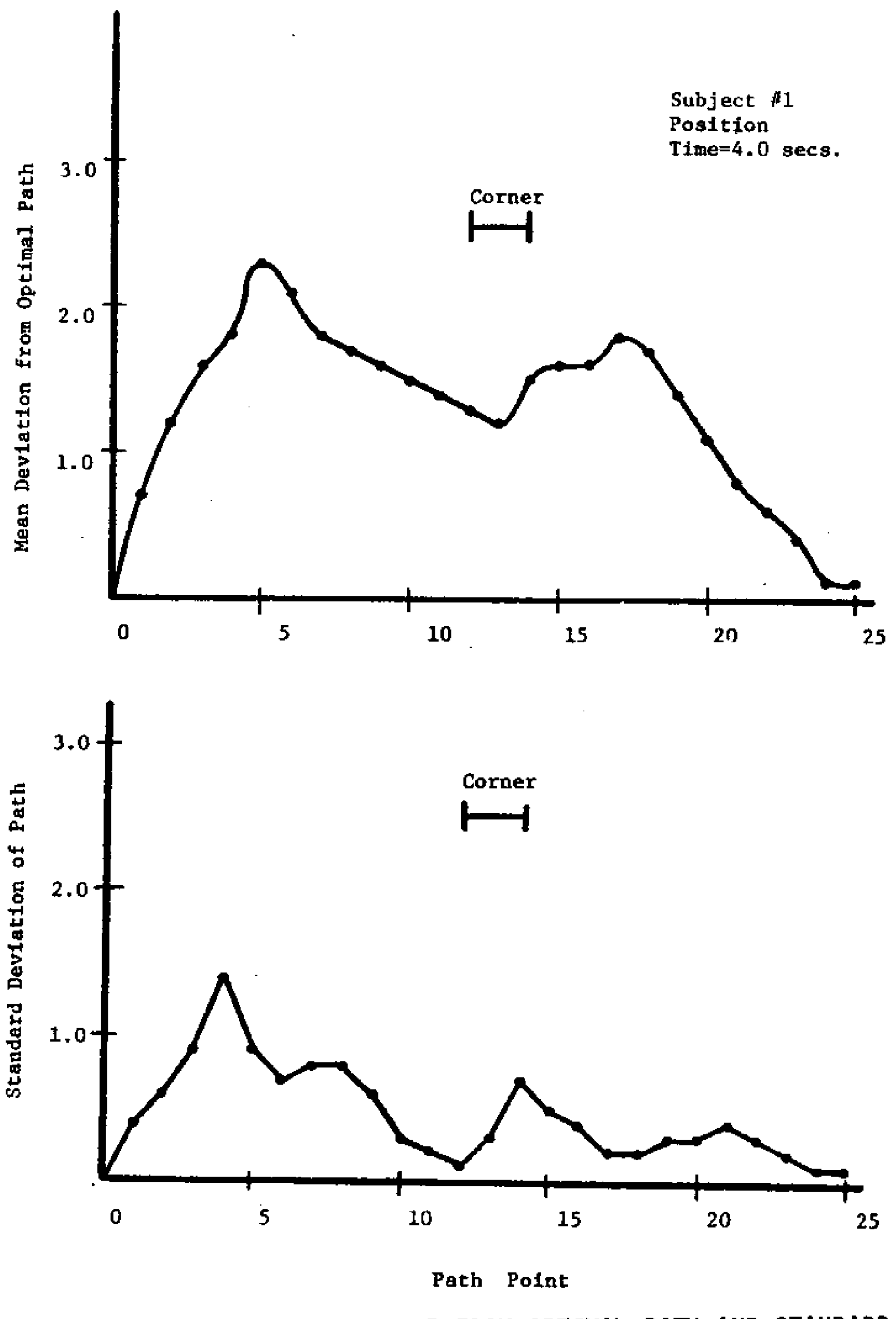


FIGURE B-11: MEAN DISTANCE FROM OPTIMAL PATH AND STANDARD DEVIATION OF MEAN PATH FOR SUBJECT #1 UNDER POSITION CONTROL

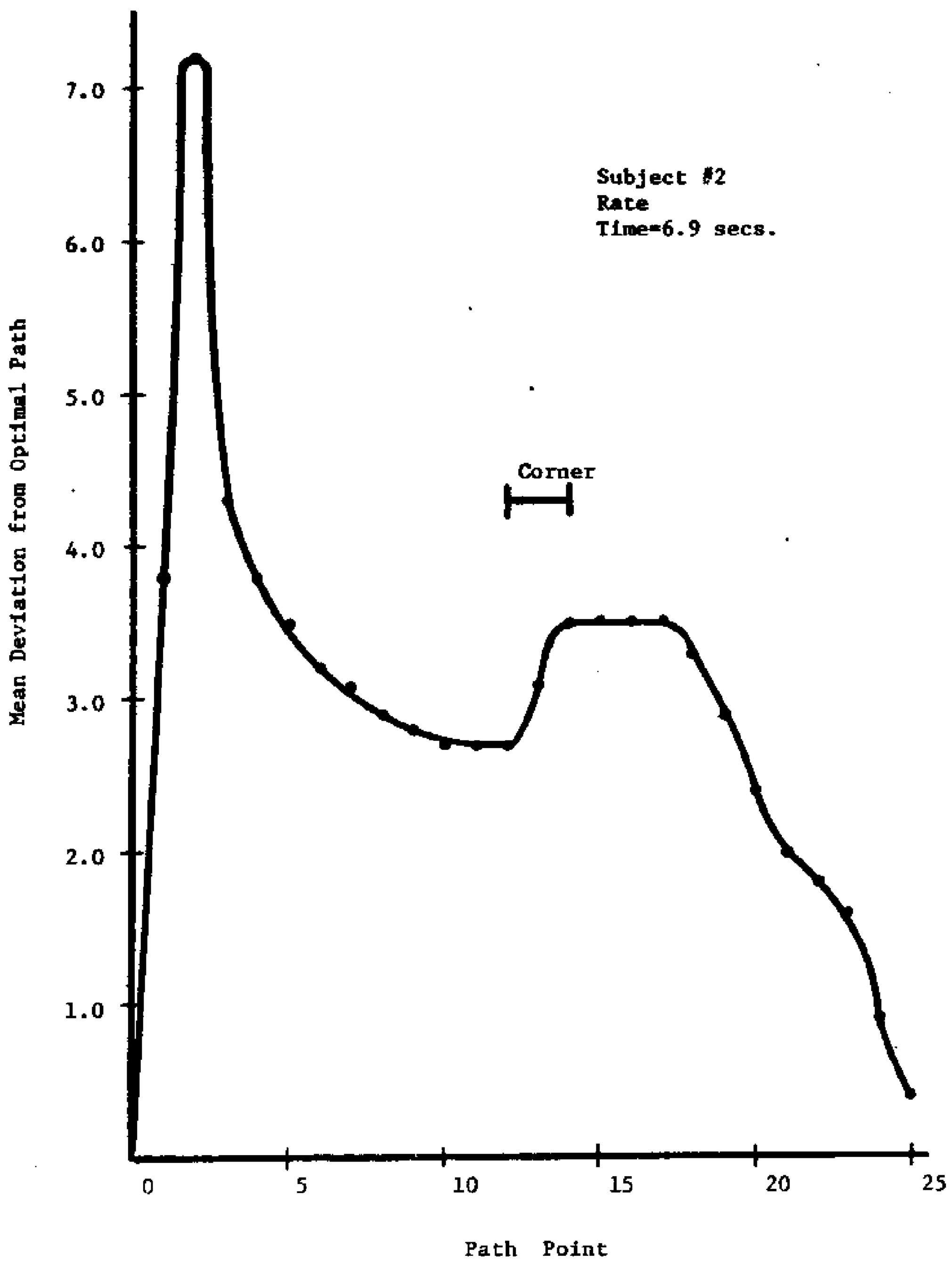


FIGURE B-12: MEAN DISTANCE FROM OPTIMAL PATH FOR SUBJECT #2 UNDER RATE CONTROL

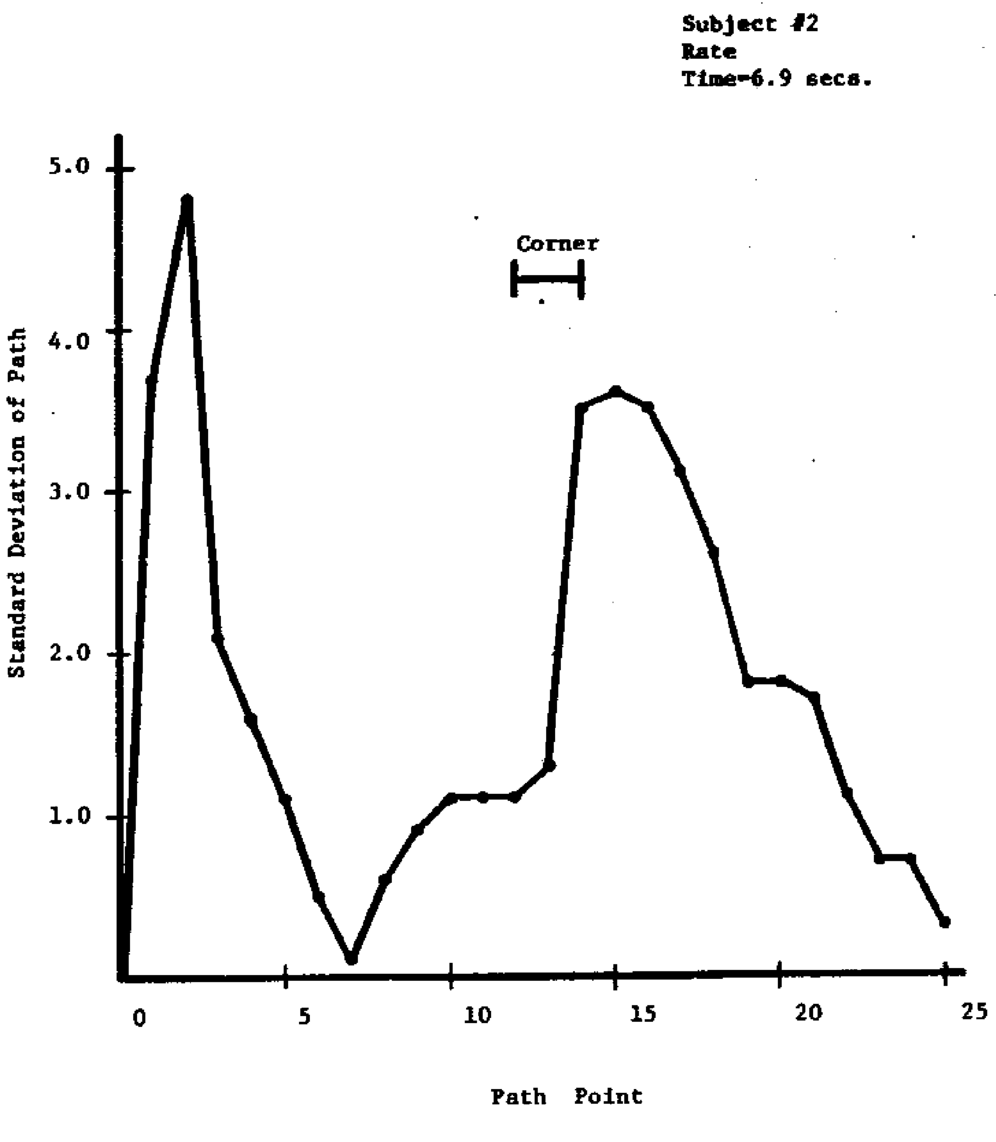
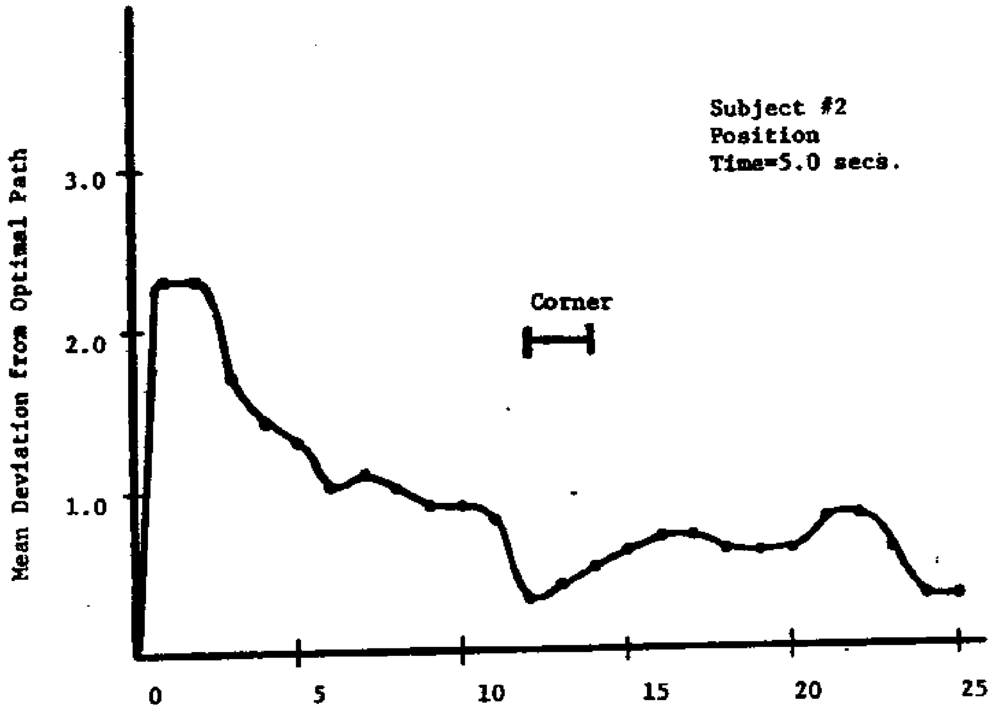


FIGURE B-13: STANDARD DEVIATION OF MEAN PATH FOR SUBJECT #2 UNDER RATE CONTROL



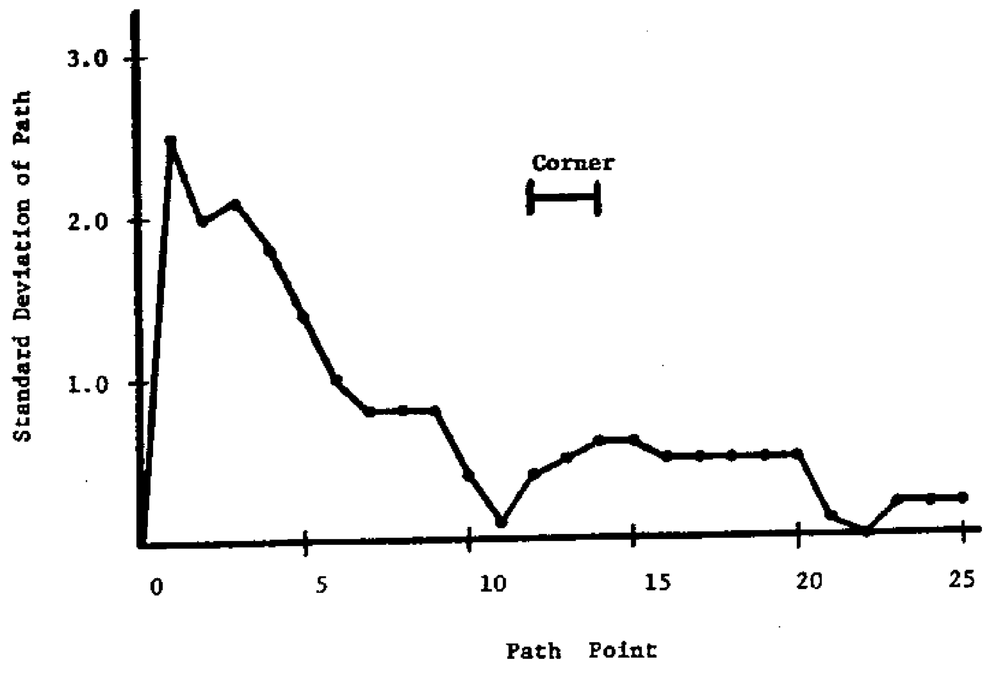


FIGURE B-14: MEAN DISTANCE FROM OPTIMAL PATH AND STANDARD DEVIATION OF MEAN PATH FOR SUBJECT #2 UNDER POSITION CONTROL

APPENDIX C

TRANSFORMATION MATRICES FOR THE E2-MANIPULATOR

The E2-manipulators at the Man-Machine Systems Laboratory have six degrees of freedom, excluding the gripping action. The assignment of coordinate systems to each of the degrees of freedom is shown in Figure C-1. Frame 0 is defined at the manipulator base and is fixed to the vehicle. Each joint of the arm is assigned a coordinate system, starting with frame 1 at the base out to the hand which is designated as frame 6. The joint angles θ_k signify the rotation of the kth frame with respect to the previous frame (k-1). With the notation of Figure C-1 and the general transformation matrix of Section 2.2, the required transformations between congruent frames have been obtained and are tabulated in Table C-1. (The $\sin\theta_k$ and the $\cos\theta_k$ are represented symbolically as Sk and Ck, respectively).

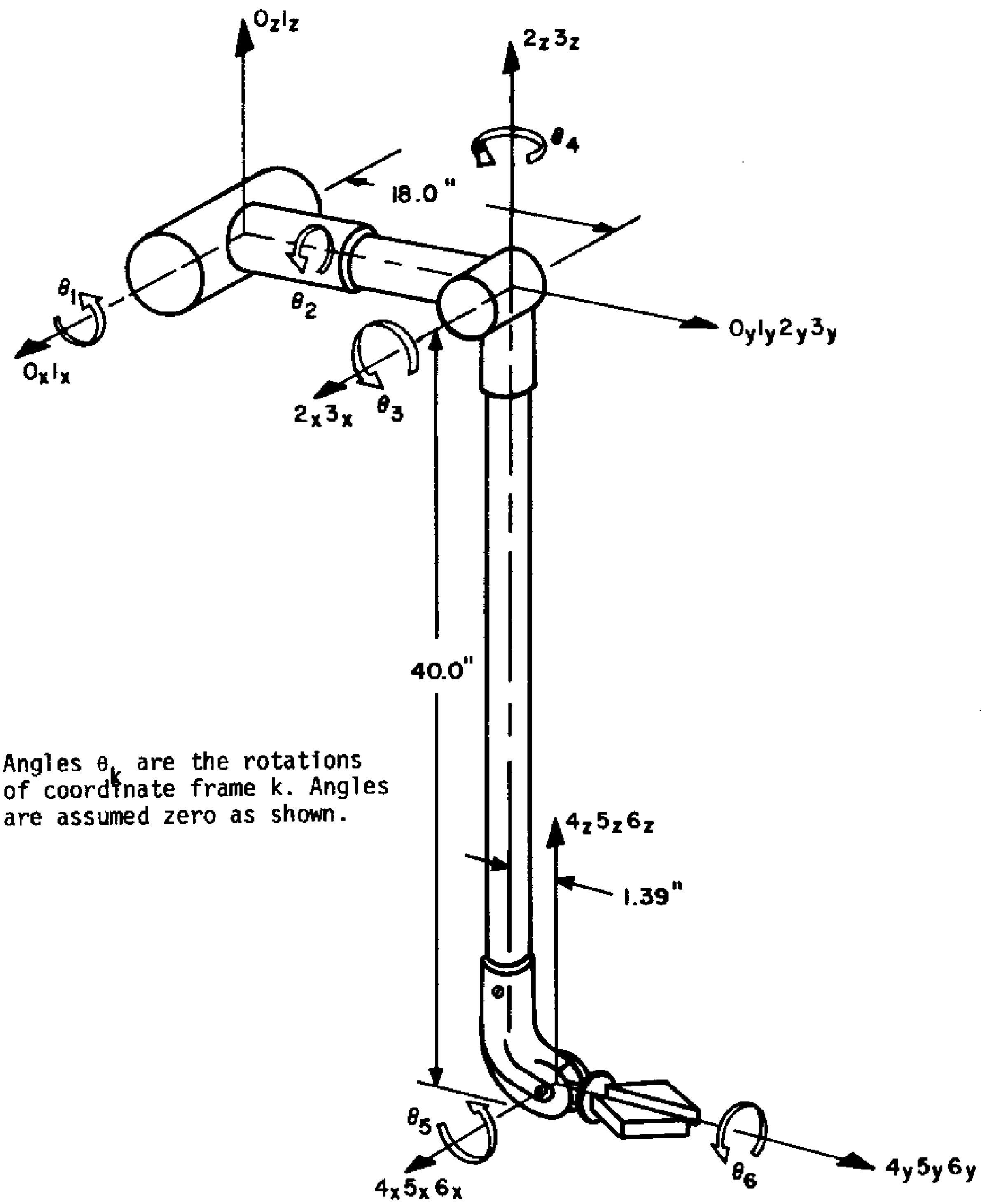
The transformation from the hand frame to the vehicle frame, as stated in Section 2.3, is given as

$$^0\underline{A}_6 = {}^0\underline{A}_1\,{}^1\underline{A}_2\,{}^2\underline{A}_3\,{}^3\underline{A}_4\,{}^4\underline{A}_5\,{}^5\underline{A}_6$$

and the transformation from the vehicle to the hand coordinate system is given by

$$^6\underline{A}_0 = {}^6\underline{A}_5\,{}^5\underline{A}_4\,{}^4\underline{A}_3\,{}^3\underline{A}_2\,{}^2\underline{A}_1\,{}^1\underline{A}_0$$

Substituting the congruent frame transformations of Table C-1 into the above equations, the transformation from vehicle to hand, and



Angles θ_k are the rotations of coordinate frame k. Angles are assumed zero as shown.

Figure C-1: Definition of Coordinate Systems and Rotation Angles (Modified from Groome[12]) vice versa, have been obtained and are given in Tables C-2 and C-3 (a_{ij} and a'_{ij} represent elements in matrices ${}^{6}\underline{A}_{0}$ and ${}^{0}\underline{A}_{6}$ respectively).

Table C-1: E2 Congruent Frame Transformations

$$^{1}A_{0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C1 & S1 & 0 \\ 0 & -S1 & C1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{0}A_{1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C1 & -S1 & 0 \\ 0 & S1 & C1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}A_{1} = \begin{bmatrix} C2 & 0 & -S2 & 0 \\ 0 & 1 & 0 & -18 \\ S2 & 0 & C2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{1}A_{2} = \begin{bmatrix} C2 & 0 & S2 & 0 \\ 0 & 1 & 0 & 18 \\ -S2 & 0 & C2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}A_{2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C3 & S3 & 0 \\ 0 & -S3 & C3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{2}A_{3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C3 & -S3 & 0 \\ 0 & S3 & C3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}A_{3} = \begin{bmatrix} C4 & S4 & 0 & 0 \\ -S4 & C4 & 0 & -1.39 \\ 0 & 0 & 1 & 40 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{3}A_{4} = \begin{bmatrix} C4 & -S4 & 0 & -1.39S4 \\ S4 & C4 & 0 & 1.39C4 \\ 0 & 0 & 1 & -40 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{5}A_{4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C5 & S5 & 0 \\ 0 & -S5 & C5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{4}A_{5} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C5 & -S5 & 0 \\ 0 & S5 & C5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{6}A_{5} = \begin{bmatrix} C6 & 0 & -S6 & 0 \\ 0 & 1 & 0 & 0 \\ S6 & 0 & C6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{5}A_{6} = \begin{bmatrix} C6 & 0 & S6 & 0 \\ 0 & 1 & 0 & 0 \\ -S6 & 0 & C6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Table C-2: E2 Transformation from Vehicle to Hand

$$
{}^6A_0 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

a_{11} = (C2C4+S2S3S4)C6 + (S2S3C4S5-C2S4S5-S2C3C5)S6

a_{12} = (C1C3C4-S1C2S3C4-S1S2S4)S5S6 + (C1S3+S1C2C3)C5S6 +(C1C3-S1C2S3)S4C6+S1S2C4C6

a_{13} = (C1S2S4+S1C3C4+C1C2S3C4)S5S6 + (S1S3-C1C2C3)C5S6 +(C1C2S3+S1C3)S4C6 - C1S2C4C6

a_{14} = -18(C3S4C6+C3C4S5S6+S3C5S6) -40 C5S6-1.39 S5S6

a_{21} = (S2S3C4-C2S4)C5 + S2C3S5

a_{22} = (C1C3C4-S1S2S4-S1C2S3C4)C5 - (C1S3+S1C2C3)S5

a_{23} = (C1S2S4+S1C3C4+C1C2S3C4)C5 + (C1C2C3-S1S3)S5

a_{24} = 18(S3S5-C3C4C5)-1.39 C5 + 40S5

a_{31} = (C2S4S5-S2S3C4S5+S2C3C5)C6 + (C2C4 + S2S3S4)S6

a_{32} = (S1S2S4-C1C3C4+S1C2S3C4)S5C6 - (C1S3+S1C2C3)C5C6 +(C1C3-S1C2S3)S4S6 + S1S2C4S6

=

a_{33} = -(C1S2S4+S1C3C4 + C1C2S3C4)S5C6 + (S1C3+C1C2S3)S4S6 +(C1C2C3-S1S3)C5C6 - C1S2C4S6

a_{34} = 40 C5C6 + 18(C3C4S5C6-C3S4S6 + S3C5C6) + 1.39 S5C6

Table C-3: E2 Transformation form Hand to Vehicle

$$^{0}A_{6} = \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} \\ a'_{21} & a'_{22} & a'_{23} & a'_{24} \\ a'_{31} & a'_{32} & a'_{33} & a'_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$a'_{11} = a_{11}$

$a'_{12} = a_{21}$

$a'_{13} = a_{31}$

$a'_{14} = 1.39(S2S3C4-C2S4)-40(S2C3)$

$a'_{21} = a_{12}$

$a'_{22} = a_{22}$

$a'_{23} = a_{32}$

$a'_{24} = 1.39(C1C3C4-S1C2S3C4-S1S2S4) + 40(C1S3+S1C2C3)+18C1$

$a'_{31} = a_{13}$

$a'_{32} = a_{23}$

$a'_{33} = a_{33}$

$a'_{34} = 1.39(S1C3C4+C1C2S3C4+C1S2S4)+40(S1S3-C1C2C3)+18S1$

APPENDIX D

MANIPULATOR MODIFICATIONS

As mentioned in Chapter IV, to achieve better performance of the overall system the manipulator was modified. The modifications to the arm consisted of mechanical and electronic alterations which were the direct result of a change from syncro/resolvers to potentiometers for position feedback. The mechanical modifications primarily involved changes in gearing to limit potentiometer rotation to less than 360 degrees (some of the syncrho/resolvers moved through three revolutions for 90 degrees of joint movement). The remainder of this appendix will deal with the design of the servo control circuits and electronic modifications. The following description of the ANL E2 manipulators has been taken from Mullen[51] (Mullen's original explanation has been changed by the author to account for the manipulator modifications).

The ANL E2 arm used in this project is a master/slave device with all electrical connections between master and slave. Master and slave arms are identical except that the master has a grip which fits the human hand. The slave has a gripper similar to a pair of tongs. On both master and slave there are six rotating joints (see Figure C-1 in Appendix C). The lower three turn joints ($\theta_4,\theta_5,\theta_6$) in Figure C-1 are connected to their motors by means of small cables and pulleys. The upper three turn joints ($\theta_1,\theta_2,\theta_3$) are connected to their motors by means of gears. The arm is well-balanced and mechanically stabe. The motors used to drive the joints are 10 watt, 115 volt, 60 cycle A.C. motors. The input from the master, which is controlled by the human operator, is a mechanical angle θ_m for each axis on the manipulator. This mechanical angle is converted to an electrical signal by a potentiometer and line driving amplifier. On the slave end of the device is an identical potentiometer circuit whose output is the slave joint angle. These two signals are compared to produce a difference signal. The difference signal is then modulated and used to drive the motors on the master and slave. By driving both master and slave, the unit is made bilateral, giving the device force feedback. The amount of force feedback can be altered by varying the strength of the signal to the motors on the master. To improve stability, two other loops are present, a tachometer feed-forward and a tachometer feedback loop.[51]

Figure D-1 is a generalized block diagram for one joint of the E2 system in the master/slave mode. The solid lines in the figure are electrical connections and the dashed lines are mechanical connections. The AC signal sense is indicated by the small graphs of alternating current on the solid lines. The feeling of force feedback is obtained through the reversed signal on the master which causes a torque in the opposite direction of the torque exerted by the slave. To prevent the operator from feeling a torque when the slave is moving with a velocity, a tach feedforward loop from the slave cancels the master driving signal so that a reverse torque is

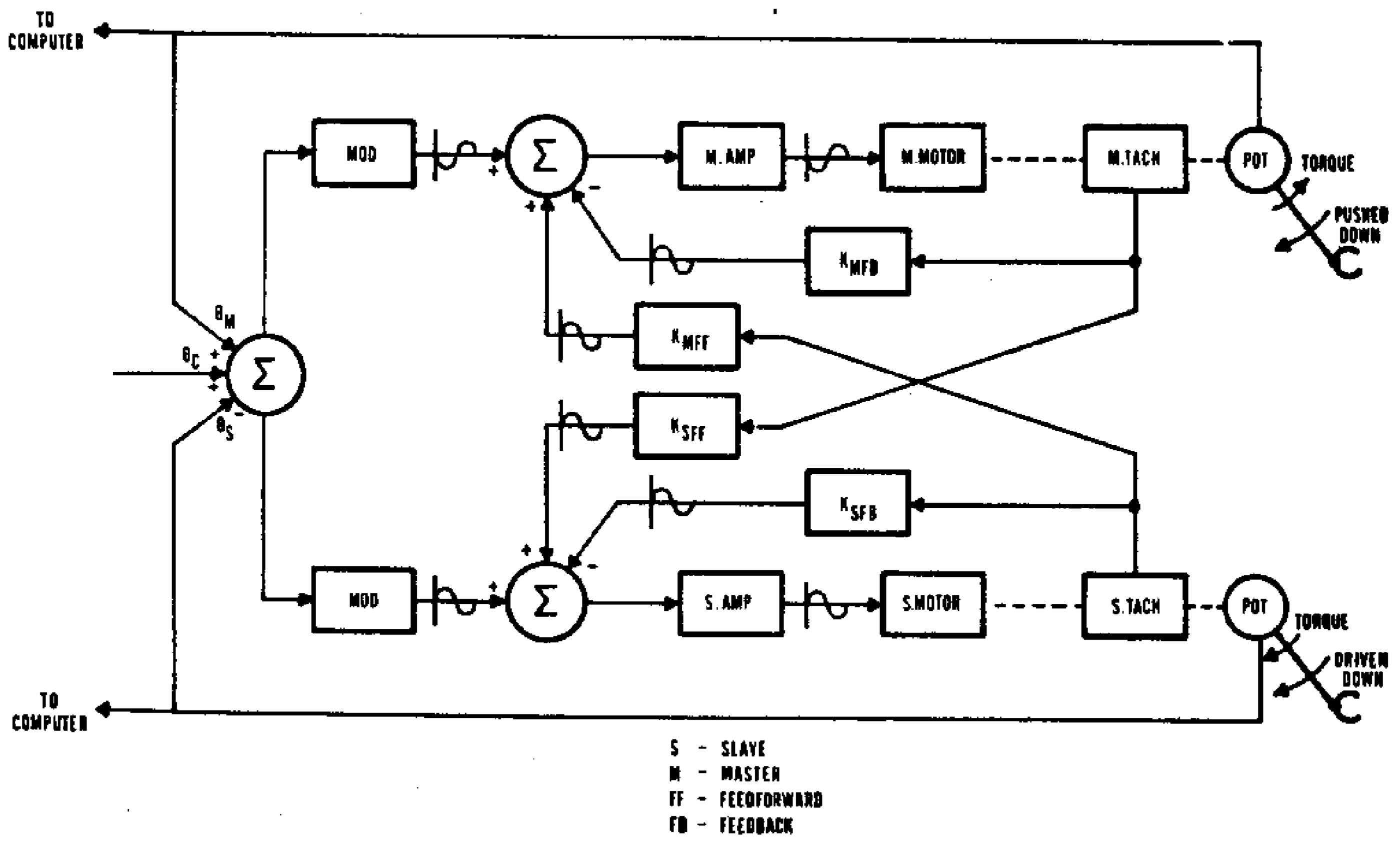


Figure D-1: Generalized Block Diagram of Servo Control Loop in Master/Slave Mode not felt at the master. Through θ_c, the computer can input a signal to maintain a mismatch between the two arms (see Section 4.6).

Figure D-2 is a generalized block diagram of the E2 system under computer control. The computer or operator can switch between the master/slave and computer configuration through the use of relays which are closed by the digital output ports or manual override switches on the servo rack. Each relay is independently closed so that any combination of computer and master/slave control can be used. Under computer control the feedforward loop is disconnected,and position signals are input directly from the computer (θ_c) into the individual comparator circuits. All manual control signals, except master/slave, are generated through the computer (e.g., under switch rate the computer reads the switches and generates the output signals).

Figures D-3 through D-8 are schematics of the servo electronic circuits. The comparator circuit shown in Figure D-5 is a proportional controller with an averaging filter which attenuates frequencies higher than 35 Hz. Proportional control was chosen so that the difference in commanded position and the actual position would represent the equivalent joint torque. If an integral controller had been used in the local servo loop, the system would continue to increase the joint torque until the actual and commanded positions where equal. Hence, if proprioceptive feedback (joint rotation) is

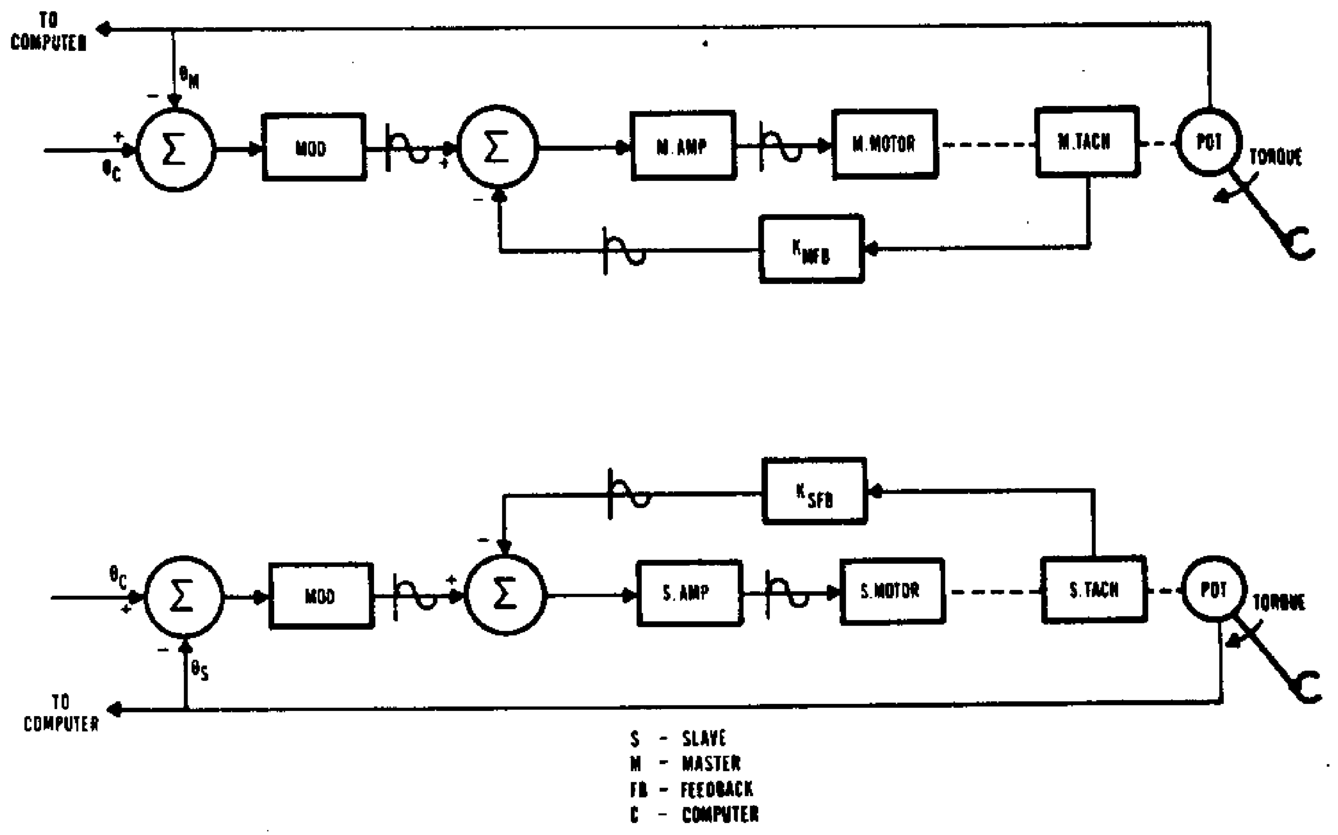


Figure D-2: Generalized Block Diagram of Servo Control Loop in Computer Mode used to determine the end effector force the local servo control loop cannot be integrated. Integral control must be done by the control algorithm when desired.

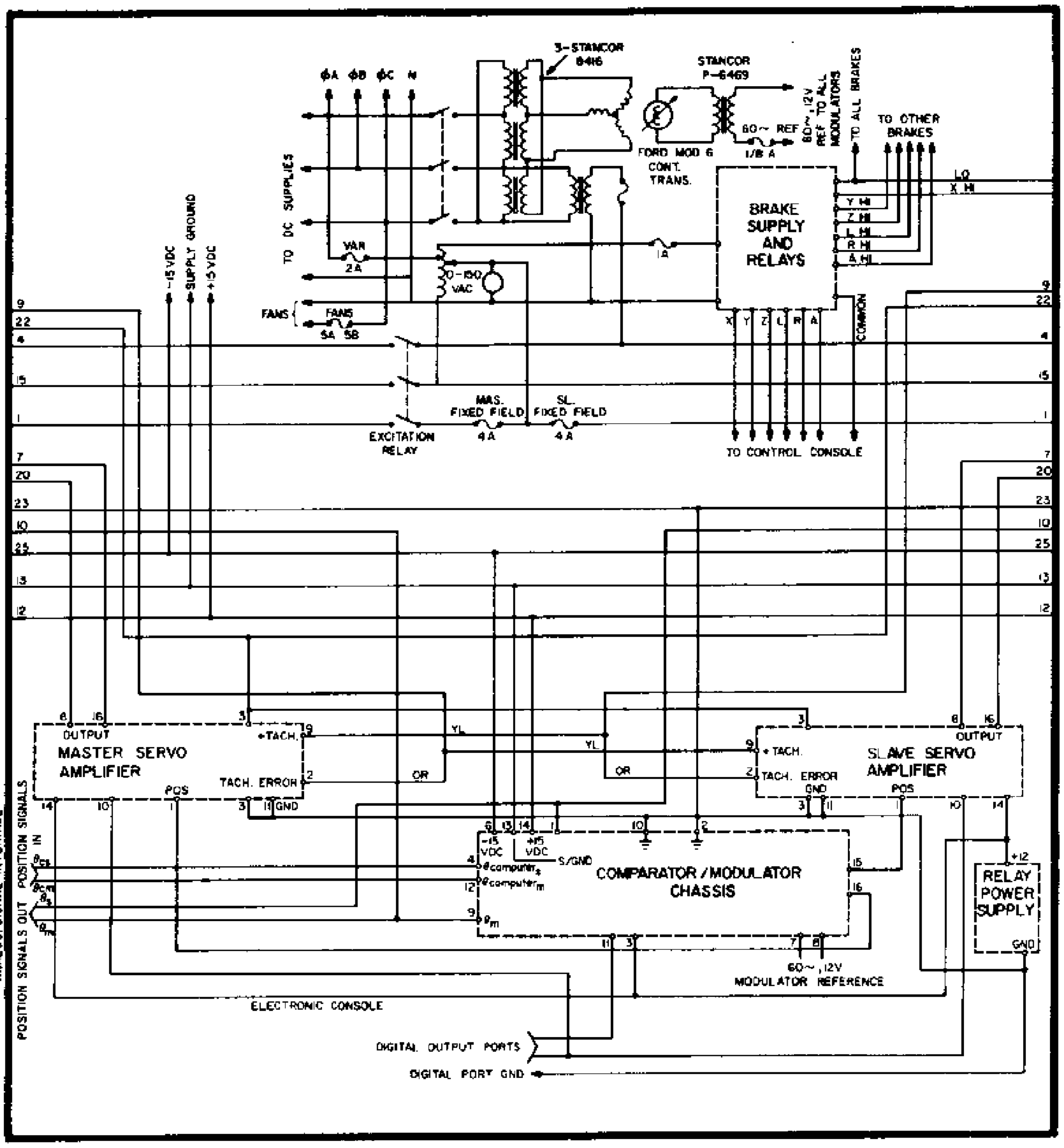


Figure D-3: Servo Control Rack Schematic

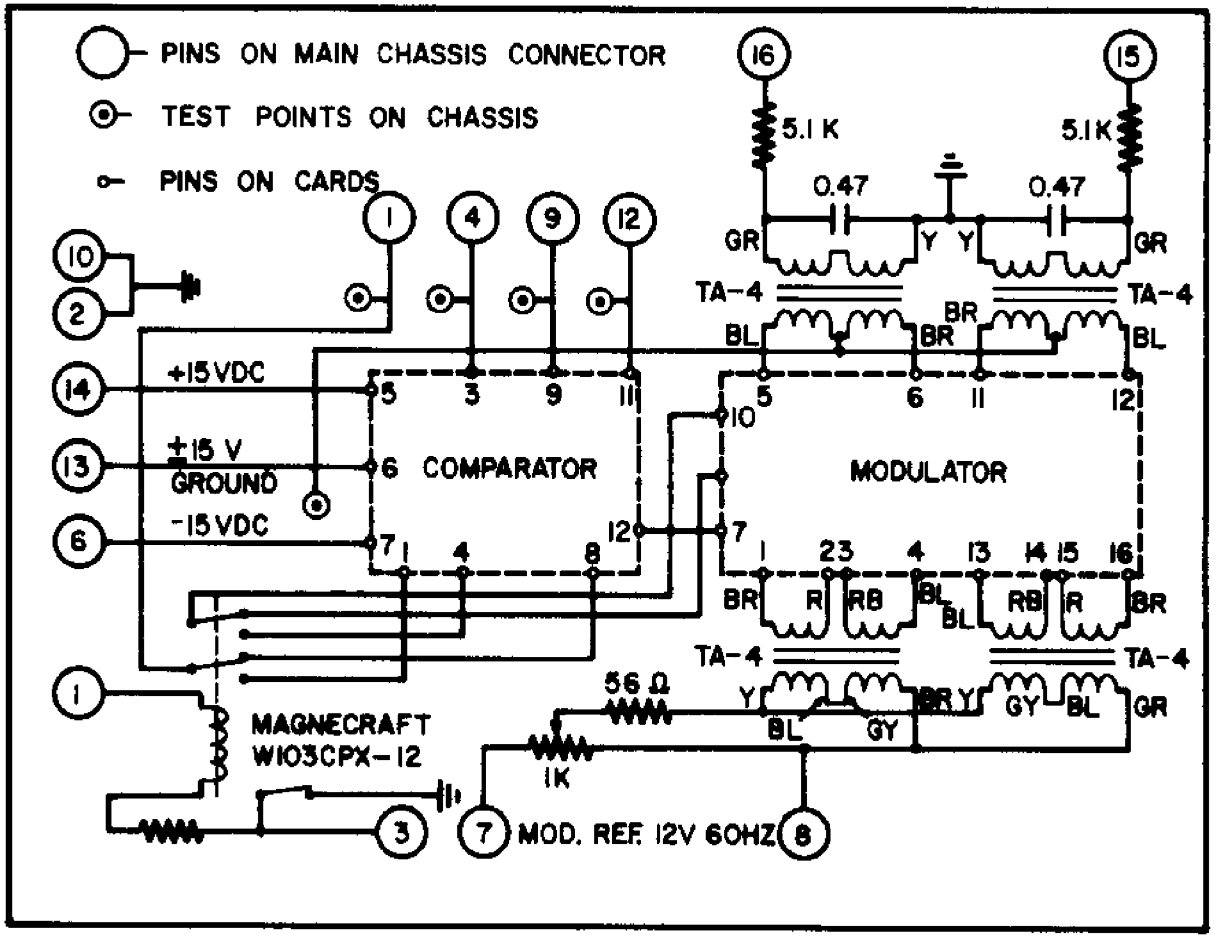


Figure D-4: Comparator/Modulator Chassis Schematic

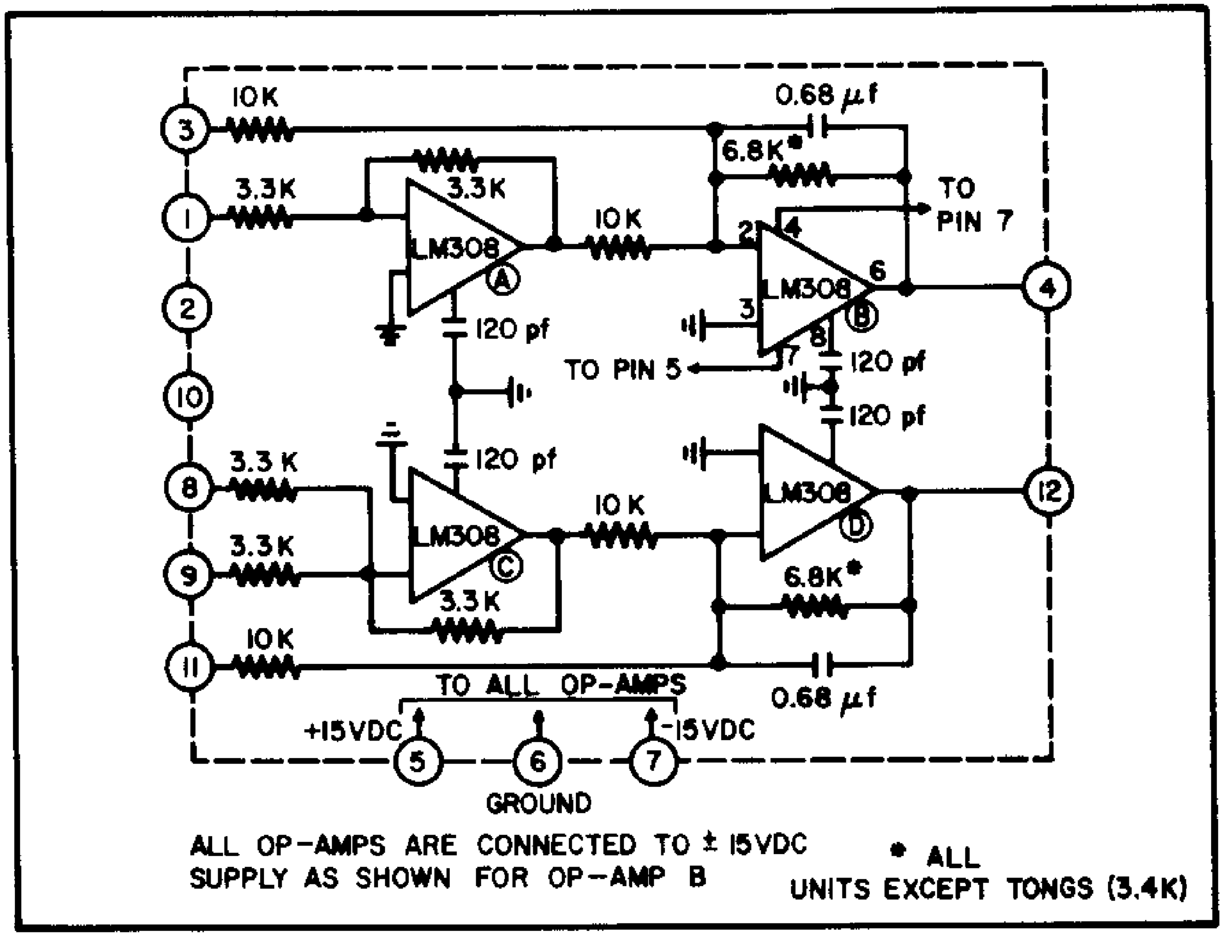


Figure D-5: Comparator Circuit Schematic

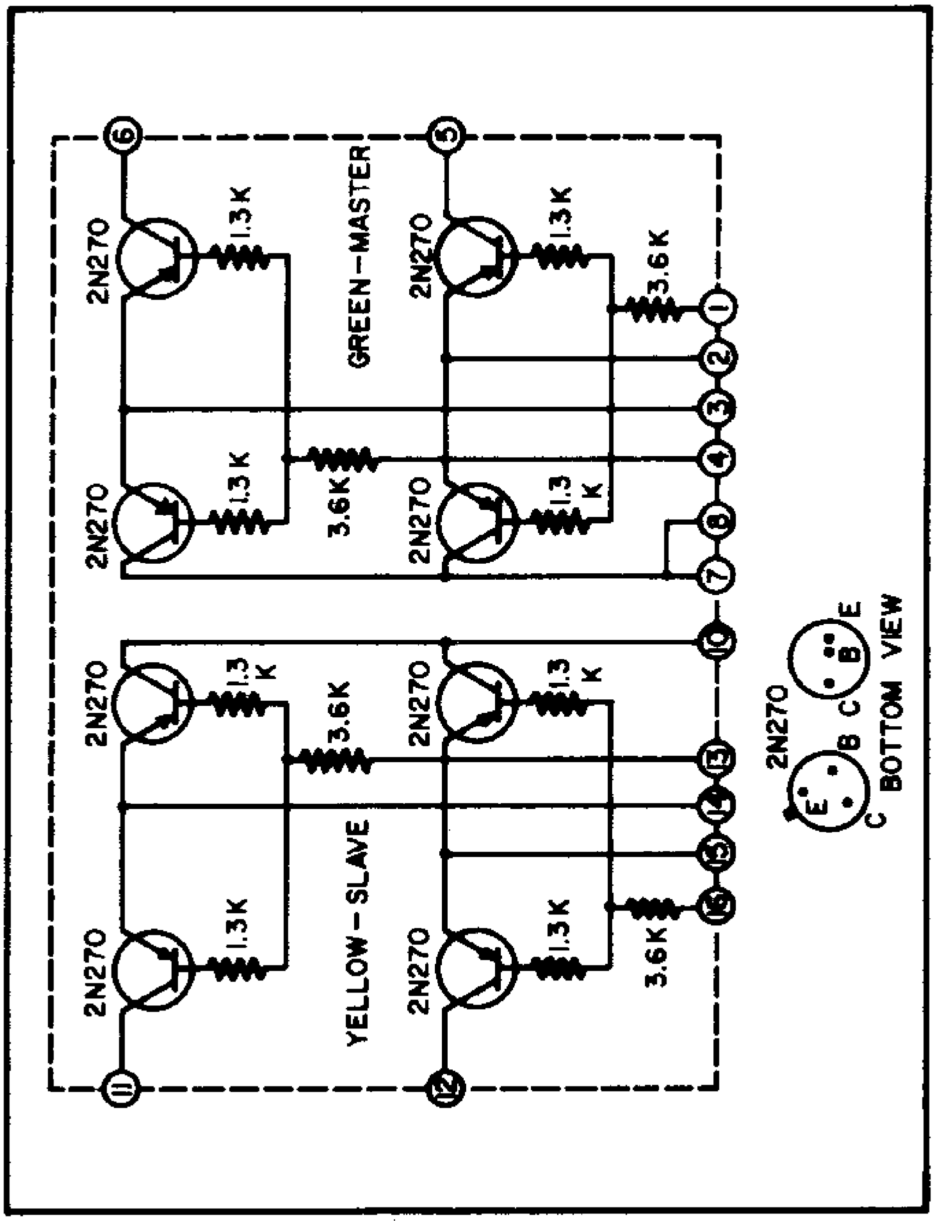


Figure D-6: Modulator Circuit Schematic

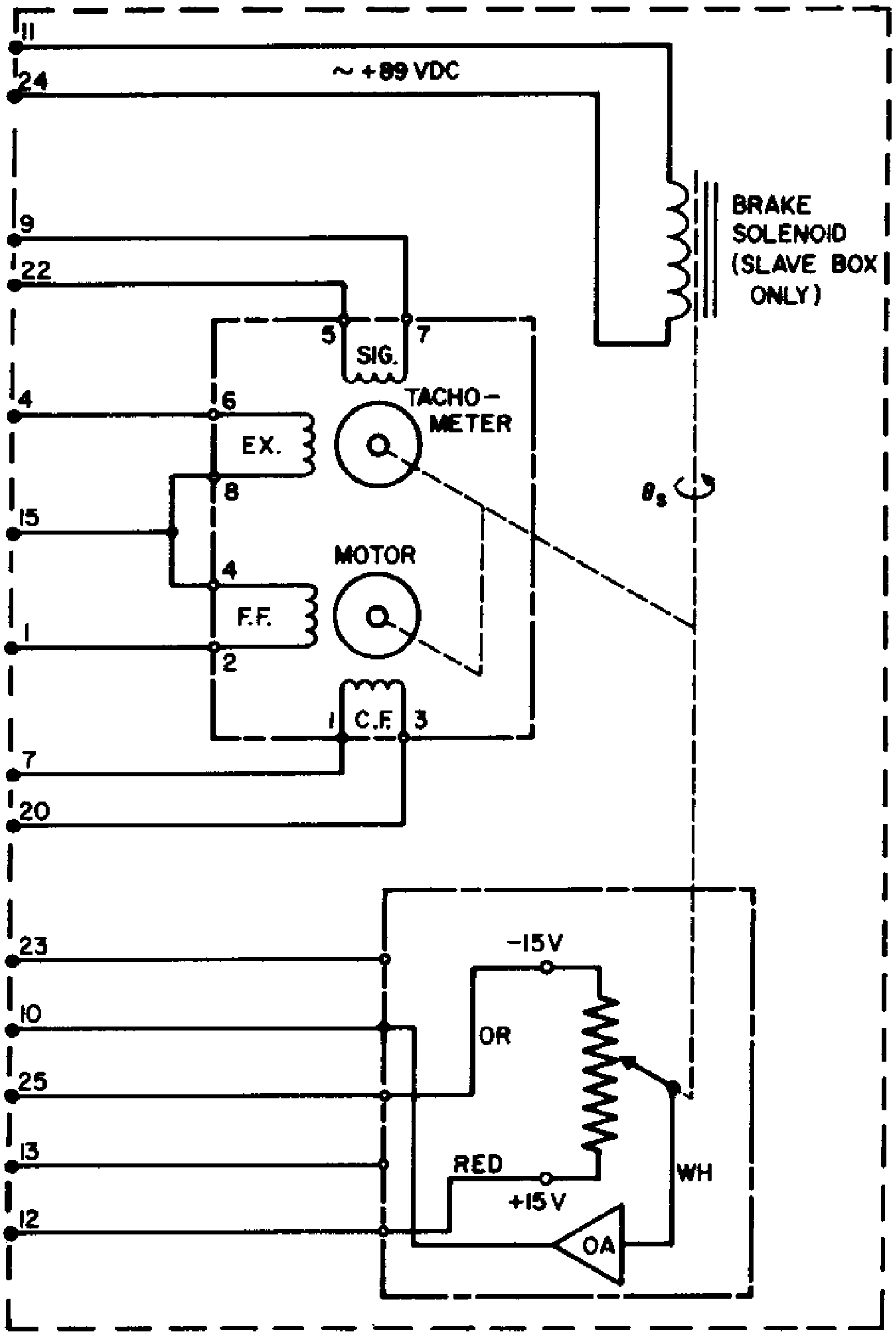


Figure D-7: Gearbox Schematic

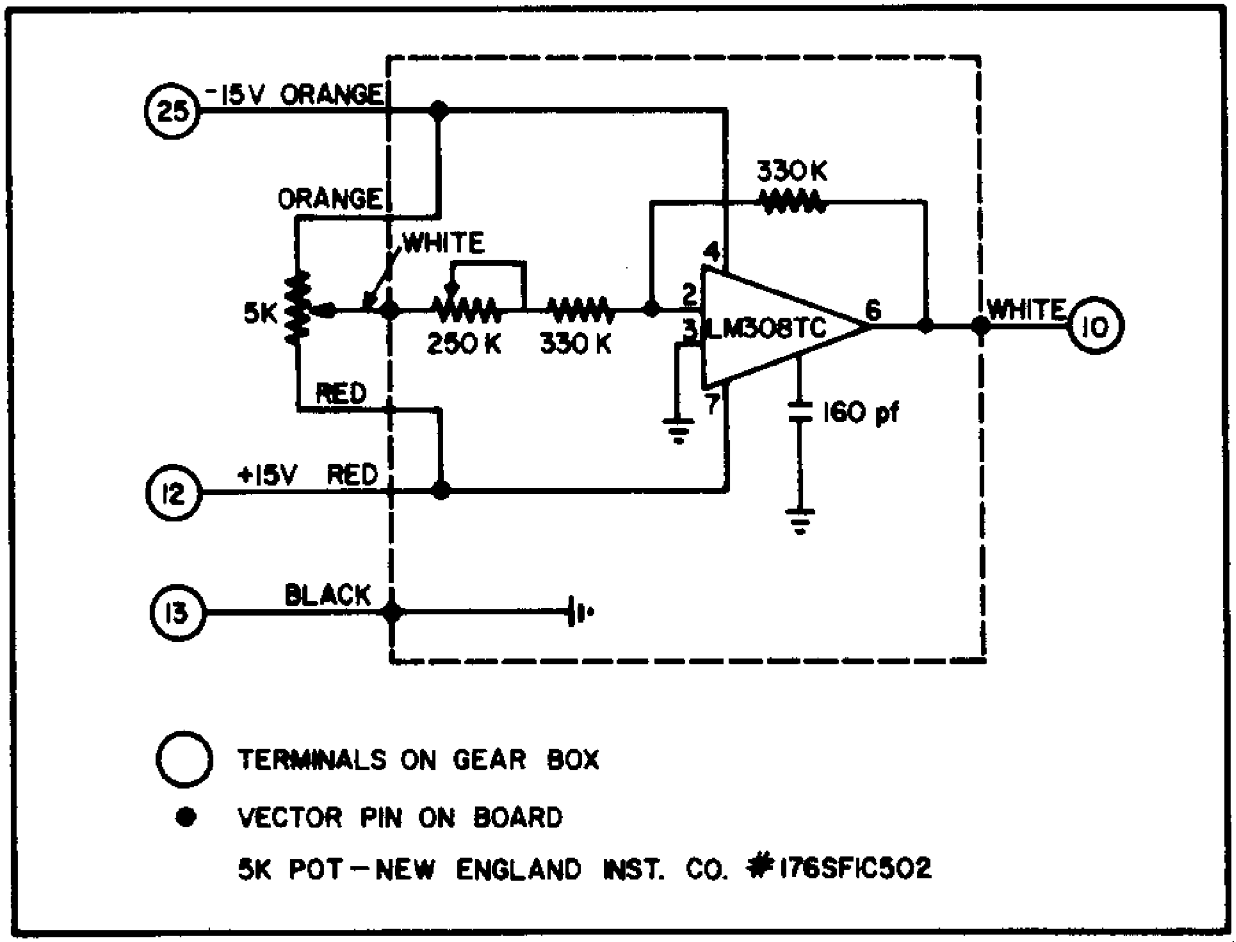


Figure D-8: Gearbox Potentiometer and Line Driver Schematic

APPENDIX E

CAMERA SLAVED TO END EFFECTOR POSITION

Assume it is desired to slave the video camera to the end effector so that the human operator will not have to continuously adjust the camera position. This can be easily achieved through the use of the transformation matrices and an analytic solution for the two degree of freedom pan and tilt mechanism. The procedure is as follows.

Assign coordinate frames to the camera and manipulator bases as shown in Figure E-1. The vector $\underline{X}_{EE}$ is given by the transformation from the end effector to the manipulator base (Equation 2-4);

$$^{0}\underline{X}_{EE} = {^{0}\underline{A}_{1}}\ {^{1}\underline{A}_{2}}\ {^{2}\underline{A}_{3}}\ {^{3}\underline{A}_{4}}\ {^{4}\underline{A}_{5}}\ {^{5}\underline{A}_{6}}\ {^{6}\underline{X}_{EE}} \qquad \text{(E-1)}$$

The vector from the camera base to the end effector $^{c}\underline{X}_{EE}$ is then given by,

$$^{c}\underline{X}_{EE} = {^{c}\underline{A}_{0}}\ {^{0}\underline{A}_{6}}\ {^{6}\underline{X}_{EE}} \qquad \text{(E-2)}$$

where,

$^{c}\underline{A}_{0}$ - is the transformation from the manipulator base to the camera base.

$^{0}\underline{A}_{6}$ - is the transformation from the end effector to the manipulator base.

Now that the position of the end effector is known in the camera base coordinates it will be necessary to obtain the solution

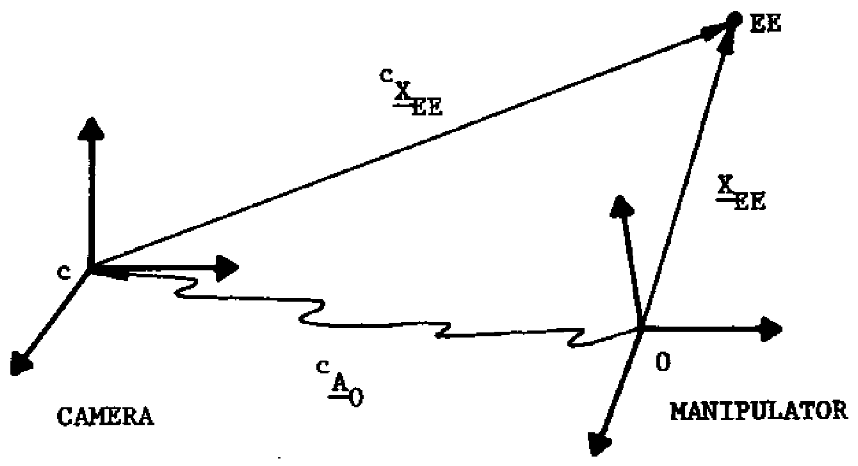


Figure E-1: Coordinate Frame Assignments for Camera and Manipulator of the camera joint space (i.e., the pan and tilt rotations required to point the camera at the end effector). The tilting rotation will be denoted by ψ and the panning rotation will be denoted by θ. It will be assumed that the camera pans first and then tilts (This assumption is only a mathematical simplification and places no restrictions on the actual mechanism). The vector ${}^{c}\underline{X}_{EE}$ points from the camera base to the end effector — the exact direction that it is desired for the camera to point.

Assume a coordinate frame p is fixed to the camera with the Y axis pointing out of the camera through the lens and that frame p has been panned through the angle θ as in Figure E-2.

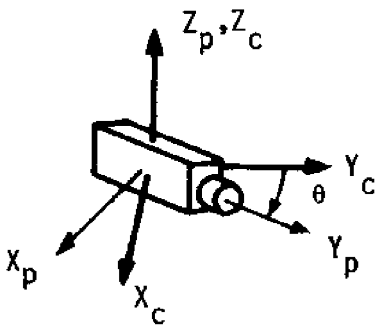


Figure E-2: Camera Panned Through Angle θ

The transformation from coordinate frame p (pan) to coordinate frame c (camera) is

$$
{}^{c}\underline{A}_{p} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(E-3)}
$$

Now that the camera has been panned through θ degrees, imagine that another coordinate frame t is fixed to the camera with the Y axis pointing out of the camera through the lens. Also, assume that the coordinate frame p is now fixed and that the camera tilts through the angle ψ as shown in Figure E-3.

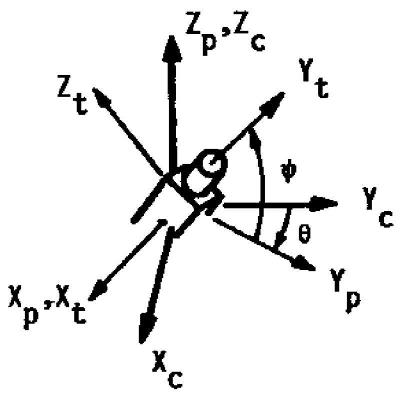


Figure E-3: Camera Tilted Through Angle ψ

The transformation from coordinate frame t (tilt) to coordinate frame p (pan) is simply

$$
{}^{p}\underline{A}_{t} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(E-4)}
$$

By Equation 2-3 the transformation from the tilt frame t to the camera frame c is given by,

$$ {}^{c}\underline{A}_t = {}^{c}\underline{A}_p \, {}^{p}\underline{A}_t = \begin{bmatrix} \cos\theta & \cos\psi\sin\theta & -\sin\psi\sin\theta & 0 \\ -\sin\theta & \cos\psi\cos\theta & -\sin\psi\cos\theta & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(E-5)} $$

Now since the Y_t axis points in the direction of the end effector, the vector ${}^{c}\underline{X}_{EE}$ is defined in the tilt frame by

$$ {}^{t}\underline{X}_{EE} = |{}^{c}\underline{X}_{EE}| \, \underline{Y}_t = \begin{bmatrix} 0 \\ d \\ 0 \\ 1 \end{bmatrix}_t \quad \text{(E-6)} $$

where

$\underline{Y}_t$ is the unit vector in the Y_t direction

$d = |{}^{c}\underline{X}_{EE}| = [x_c^2 + y_c^2 + z_c^2]^{1/2}$ (The magnitude of the vector from the camera base frame to the end effector).

Combining equations E-5 and E-6, gives

$$ {}^{c}\underline{X}_{EE} = {}^{c}\underline{A}_t \, {}^{t}\underline{X}_{EE} = {}^{c}\underline{A}_t \begin{bmatrix} 0 \\ d \\ 0 \\ 1 \end{bmatrix}_t $$

Performing the matrix multiplications gives three equations in two unknowns,

$$ x_c = d\cos\psi\sin\theta $$
$$ y_c = d\cos\psi\cos\theta $$

$$z_c = d\sin\psi$$

which gives,

$$\sin\psi = \frac{z_c}{d}$$

$$\tan\theta = \frac{x_c}{y_c}$$

the desired result.

APPENDIX F

SUPERMAN TASK PROGRAM LISTINGS

Get-Tool Task

```
1     ABSOLUTE
2     THROUGH PATH
3     FIXED VELOCITY - 31% MAXV
4     DISCRETE PATH
5     DISCRETE PATH
6     ADAPT A R L Z Y
7     GRASP WITH FORCE 199
8     TERMINATE FIXED VELOCITY
9     THROUGH PATH
10    THROUGH PATH
12    DPATH TO RETURN CONTROL
13    END
```

Return-Tool Task

```
1     ABSOLUTE
2     THROUGH PATH
3     FIXED VELOCITY - 31% MAXV
4     DISCRETE PATH
5     DISCRETE PATH
6     ADAPT A R L Z Y
7     RELEASE
8     TERMINATE FIXED VELOCITY
9     THROUGH PATH
10    THROUGH PATH
12    DPATH TO RETURN CONTROL
13    END
```

Nut-Off Task

```
 1    RELATIVE
 2    DISCRETE PATH
 3    LABEL  3
 4    GRASP WITH FORCE 199
 5    DISCRETE PATH
 6    INCREMENT Y   590
 7    IF Y-FORCE.GT. 35 EXECUTE NEXT COMMAND
 8    GO TO  2
 9    GO TO  1
10    LABEL  2
11    RELEASE
12    DISCRETE PATH
13    GO TO  3
14    LABEL  1
15    ABSOLUTE
16    THROUGH PATH
17    THROUGH PATH
18    DISCRETE PATH
19    RELEASE
20    THROUGH PATH
22    DPATH TO RETURN CONTROL
23    END
```

Sampler Task

```
 1     ABSOLUTE
 2     GRASP WITH FORCE 199
 3     FIXED VELOCITY - 9Ø% MAXV
 4     THROUGH PATH
 5     THROUGH PATH
 6     RELEASE
 7     THROUGH PATH
 9     DPATH TO RETURN CONTROL
1Ø     END
```

Digger Task

```
 1     RELATIVE
 2     GRASP WITH FORCE 197
 3     FIXED VELOCITY -1ØØ% MAXV
 4     DISCRETE PATH
 5     THROUGH PATH
 6     FIXED VELOCITY - 66% MAXV
 7     THROUGH PATH
 8     ABSOLUTE
 9     THROUGH PATH
1Ø     THROUGH PATH
11     FIXED VELOCITY -1ØØ% MAXV
12     DISCRETE PATH
13     THROUGH PATH
15     DPATH TO RETURN CONTROL
16     END
```

Bolt-On Task

```
 1     RELATIVE
 2     GRASP WITH FORCE 199
 3     DISCRETE PATH
 4     DISCRETE PATH
 5     DISCRETE PATH
 6     IF R-FORCE.GT.  2 EXECUTE NEXT COMMAND
 7     IF L-FORCE.GT.  2 EXECUTE NEXT COMMAND
 8     GO TO  3
 9     INCREMENT Z  -960
10     IF Z-FORCE.GT. 50 EXECUTE NEXT COMMAND
11     GO TO  1
12     GO TO  4
13     LABEL  1
14     RELEASE
15     DISCRETE PATH
16     GRASP WITH FORCE 199
17     DISCRETE PATH
18     IF R-FORCE.GT.  2 EXECUTE NEXT COMMAND
19     IF L-FORCE.GT.  2 EXECUTE NEXT COMMAND
20     GO TO  2
21     GO TO  1
22     LABEL  3
23     MESSAGE  1
24     ALERT
25     GO TO  5
26     LABEL  4
27     MESSAGE  2
28     ALERT
29     GO TO  5
30     LABEL  2
31     RELEASE
32     LABEL  5
34     DPATH TO RETURN CONTROL
35     END
```

Valve Task

```
RELATIVE
LABEL 1
DISCRETE PATH
GRASP WITH FORCE 199
DISCRETE PATH
IF R-FORCE.GT.1Ø4 EXECUTE NEXT COMMAND
IF L-FORCE.GT.1Ø4 EXECUTE NEXT COMMAND
GO TO  2
RELEASE
GO TO  1
LABEL  2
MESSAGE  1
RELEASE
DPATH TO RETURN CONTROL
END
```

APPENDIX G

DATA

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL (MEAN/SD)	3.8/0.3	4.4/0.5	4.2/0.4	5.4/1.4	14.5/3.5	14.7/2.0	49.6/5.5	53.7/13.5	BJ
	8.5/1.4	6.2/0.8	11.9/3.5	7.8/0.9	*/*	*/*	70.4/20.0	73.6/20.7	MT
	4.4/0.6	4.3/0.5	6.0/2.0	5.0/0.5	10.5/2.5	11.0/1.1	57.2/12.9	56.8/10.2	SB
SUPERVISORY CONTROL (MEAN/SD)	4.6/0.0 (Absolute Task Without Operator Travel Results in Constant Time)								SB
									MT
									SB

Table G-1: Mean Times and Standard Deviations for Tool-Retrieval Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL EXPECTED # ERRORS	0.2HNS	NO ERRORS	0.2C	0.2C 0.6HNS	0.4HNS	0.6HNS	0.2HNS	0.6C 0.2HNS	BJ
	0.2HNS	0.4HNS	0.2C 0.2D 0.6HNS	0.2C 0.6HNS	*	*	0.6C 0.4HNS	0.2C 0.2HNS	MT
	NO ERRORS	NO ERRORS	NO ERRORS	0.2C	NO ERRORS	0.2C 0.2HNS	0.4C 0.2HNS	0.2C 0.6HNS	SB
SUPERVISORY CONTROL EXPECTED # ERRORS	NO ERRORS (Task Only Requires Button Push)								BJ
									MT
									SB

CODE: C - COLLISION HNS - HANDLE NOT SEATED PROPERLY D-DROPPED TOOL

Table G-2: Expected Errors for Tool-Retrieval Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VAR.ABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL (MEAN/SD)	4.6/0.6	4.5/1.0	5.1/0.8	5.2/0.4	16.4/1.0	15.5/2.6	48.8/8.8	54.8/13.7	BJ
	9.7/1.8	8.7/3.8	13.4/5.5	12.3/4.8	*/*	*/*	87.5/39.6	84.5/21.1	MT
	5.4/1.1	4.4/0.3	7.8/2.9	5.8/1.3	14.7/2.1	12.1/2.4	66.3/6.8	58.4/5.8	SB
SUPERVISORY CONTROL (Mean/SD)	5.2/0.0 (Absolute Task Without Operator Travel Results in Constant Time)								BJ
									MT
									SB

Table G-3: Mean Times and Standard Deviations for Tool-Return Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL EXPECTED # ERRORS	NO ERRORS	NO ERRORS	NO ERRORS	NO ERRORS	0.2C	NO ERRORS	0.2D 0.2NSR	0.2C	BJ
	NO ERRORS	0.4NSR	0.6C 0.2D 0.2NSR	0.8C 0.2NSR	*	*	0.4C 0.4D 0.4NSR	0.6C	MT
	NO ERRORS	NO ERRORS	NO ERRORS	NO ERRORS	0.6C	0.4C	0.4C	NO ERRORS	SB
SUPERVISORY CONTROL EXPECTED # ERRORS	NO ERRORS (TASK ONLY REQUIRES BUTTON PUSH)								BJ
									MT
									SB

CODE: C-COLLISION NSR-TOOL NOT SEATED ON RACK D-DROPPED TOOL

Table G-4: Expected Errors for Tool-Return Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL (MEAN/SD)	31.0/5.4	22.3/3.1	39.9/5.3	30.3/5.1	*/*	*/*	120.0/44.8	95.9/13.8	BD
	20.5/3.6	17.1/3.3	24.3/4.8	23.0/2.3	46.0/6.0	33.4/9.1	89.0/13.5	75.5/10.1	AB
	18.8/3.0	15.0/1.9	27.0/7.2	18.3/3.4	41.3/5.5	28.3/3.0	63.0/7.8	68.0/2.5	SB
SUPERVISORY CONTROL (MEAN/SD)	20.5/1.0	19.7/0.4	21.5/1.4	20.2/0.7	*/*	*/*	58.4/15.1	49.0/7.3	BD
	20.5/0.6	20.4/1.0	20.6/0.6	21.4/1.0	29.5/4.8	27.4/1.9	44.8/17.2	36.2/5.3	AB
	20.2/0.5	19.3/0.3	22.7/3.1	20.1/0.6	28.5/2.3	24.6/1.6	43.0/4.4	36.3/3.6	SB

Table G-5: Mean Times and Standard Deviations for Nut-Off Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VAI ABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL EXPECTED # ERRORS	0.2C 0.6D	0.2C	0.8D	0.2C 0.4D	*	*	0.2C 0.4D	0.6C 0.8D	BD
	0.6C 0.4D	0.2C 0.2D	0.2C 0.4D	0.2C	0.6C 0.6D	0.2C 0.6D	0.4C 0.4D	0.4C 0.2D	AB
	0.2D	0.4D	0.4D	0.2D	0.6C 0.6D	0.4C 0.6D	0.2C	0.4C	SB
SUPERVISORY CONTROL EXPECTED # ERRORS	NO ERRORS	NO ERRORS	0.2C	NO ERRORS	*	*	0.2C	0.2C	BD
	NO ERRORS	NO ERRORS	0.2C	NO ERRORS	0.2C	NO ERRORS	0.2C	0.2C	AB
	NO ERRORS	NO ERRORS	NO ERRORS	NO ERRORS	0.2C	0.2C	0.2C	NO ERRORS	SB

CODE: C - COLLISIONS D - NUT DROPPED

Table G-6: Expected Errors for Nut-Off Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL (MEAN /SD)	49.1/3.9	50.9/5.8	54.8/3.9	57.8/4.5	123.5/7.5	136.3/7.9	252.8/25.1	223.6/22.6	KL
	40.8/2.1	47.3/4.4	42.5/3.8	48.2/3.2	112.4/6.1	118.2/2.7	218.0/26.2	239.0/15.2	JA
	49.1/3.7	50.5/1.3	50.2/2.9	55.3/4.9	120.2/9.6	132.0/9.4	230.0/9.3	248.7/20.9	SB
SUPERVISORY CONTROL (MEAN/SD)	68.3/2.2	66.1/2.7	77.2/5.2	74.4/4.3	101.8/7.2	107.0/8.4	139.2/15.8	118.2/11.6	KL
	66.2/1.1	67.7/4.6	74.8/3.3	73.8/5.6	93.8/3.9	91.2/5.6	128.6/12.3	133.8/8.2	JA
	67.6/3.4	67.5/3.7	75.7/3.9	76.5/3.1	100.4/6.5	102.0/4.1	141.6/9.2	149.0/11.3	SB

Table G-7: Mean Times and Standard Deviations for Sampler Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
EXPECTED NUMBER OF ERRORS	0.4C 0.2R	0.4C 0.4M 0.2R	1.2C 0.4M	1.2C 0.6M	3.0C 1.0M 0.4R	3.2C 1.8M	1.6C 0.2M 0.2R	2.6C 0.4M	KL
	0.6C	0.4C 0.4M	0.6C 0.2M	1.4C 0.4M	1.0C 1.4M	2.0C 0.6M	1.8C 0.2M	1.4C 0.6M 0.2R	JA
	0.8C 0.2R	0.2M	0.6C	0.8C 0.2M 0.2R	1.8C 1.0M	3.2C 0.2M	1.8C 0.6M	1.6C 0.4M	SB
EXPECTED NUMBER OF ERRORS	0.2M 0.4R 0.2WB	0.2WB	0.2R 0.4WB	0.2R 0.6WB	0.2WB	0.2C 0.2M	0.4C 0.2R 0.2WB	0.2C 0.4R 0.2WB	KL
	0.2M 0.2WB	0.2M 0.2WB	0.6WB	0.8WB	0.4WB	0.2WB	0.2M 0.4WB	0.2M	JA
	0.2WB	0.4WB	0.2C 0.8WB	0.2WB	0.2C 0.2R 0.2WB	0.4WB	0.4C 0.2M 0.4WB	0.4WB	SB

CODE: C - COLLISION M-MISSED BUCKET R-PUSHED OUT OF REACH WB - WRONG BUTTON PRESSED

Table G-8: Expected Errors for Sampler Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VA.IABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL (MEAN/SD)	*	*	*	*	*	*	*	*	
	30.1/4.5	23.6/3.0	35.4/6.5	26.8/3.2	41.9/5.0	38.5/2.7	107.8/21.2	93.1/7.2	SB
	*	*	*	*	*	*	*	*	
SUPERVISORY CONTROL (MEAN/SD)	*	*	*	*	*	*	*	*	
	27.8/0.6	27.9/0.3	28.2/0.7	28.3/0.3	36.2/1.1	35.3/2.8	53.8/6.2	46.9/4.9	SB
	*	*	*	*	*	*	*	*	

Table G-9: Mean Times and Standard Deviations for Valve Operation Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL EXPECTED # ERRORS	*	*	*	*	*	*	*	*	
	NO ERRORS	0.2C	1.6C	1.8C	1.4C	1.0C	0.8C	NO ERRORS	SB
	*	*	*	*	*	*	*	*	
SUPERVISORY CONTROL EXPECTED # ERRORS	*	*	*	*	*	*	*	*	
	NO ERRORS	NO ERRORS	NO ERRORS	NO ERRORS	0.6C	0.2C	0.4C	NO ERRORS	SB
	*	*	*	*	*	*	*	*	

CODE: C - COLLISION

Table G-10: Expected Errors for Valve Operation Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL (MEAN/SD)	*	*	*	*	*	*	*	*	
	3.9/0.5	3.6/0.3	4.1/0.4	4.3/0.4	14.2/0.5	14.2/0.8	38.0/4.3	39.1/2.7	SB
	*	*	*	*	*	*	*	*	
SUPERVISORY CONTROL (ONE RUN ONLY)	*	*	*	*	*	*	*	*	
	7.0	6.9	7.0	7.0	7.4	7.5	7.9	7.9	SB
	*	*	*	*	*	*	*	*	

Table G-11: Mean Times and Standard Deviations for Digging Task

	MASTER/SLAVE FORCE FEEDBACK		MASTER/SLAVE NO FORCE FEEDBACK		VARIABLE RATE JOYSTICK		VARIABLE RATE SWITCH		
	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	1-VIEW	2-VIEW	
MANUAL CONTROL EXPECTED # ERRORS	*	*	*	*	*	*	*	*	
	NO ERRORS	NO ERRORS	0.2C	0.4C	2.2C	0.8C	1.4C	2.6C	SB
	*	*	*	*	*	*	*	*	
SUPERVISORY CONTROL EXPECTED # ERRORS	*	*	*	*	*	*	*	*	
	NO ERRORS	NO ERRORS	NO ERRORS	NO ERRORS	NO ERRORS	NO ERRROS	NO ERRORS	NO ERRORS	SB
	*	*	*	*	*	*	*	*	

CODE: C - COLLISION

Table G-12: Expected Errors for Digging Task

APPENDIX H

SUPERMAN SOURCE PROGRAMS

AVAILABLE UPON REQUEST

FROM

MAN-MACHINE SYSTEMS LAB

M.I.T.

REFERENCES

1. Talkington, H., "Manned and Remotely Operator Submersible Systems: A Comparison", NVC TP511, Naval Undersea Center, June, 1976.

2. Vadus, J.R., "International Status and Utilization of Undersea Vehicles 1976", Inter Ocean '76 Conference, June 1976.

3. Ferrel, W.R., Sheridan, T.B., "Supervisory Control of Remote Manipulation", IEEE Spectrum, Vol. 4, No. 10, October 1967, pp. 81-88.

4. Sheridan, T.B. and Verplank, W.L., "Human and Computer Control of Undersea Teleoperators", Man-Machine Systems Laboratory Report, Massachusetts Institute of Technology, July 1978.

5. Sheridan, T.B., "Modelling Supervisory Control of Robots", Theory and Practice of Robots and Manipulators, 2nd International CISM-IFT.MM Symposium, September 1976.

6. Meyers, J.J., Holm, C.H., and McAllister, R.F., Handbook of Ocean and Underwater Engineering, McGraw Hill, 1969.

7. Busby, F.R., Manned Submersibles, Office of the Oceanographer of the Navy, 1976.

8. Pesch, A.J., Hill, R.G., and Klepser, W.F., "Performance Comparisons of Scuba Divers vs. Submersible Manipulator Controllers in Undersea Work", Offshore Technology Conference, Houston, TX., 1971.

9. Moore, A.P., "Metals Joining in the Deep Ocean", Master's Thesis, Department of Ocean Engineering, M.I.T., May 1975.

10. Pieper, D.I. and Roth, B., "The Kinematics of Manipulators Under Computer Control", Ph.D. Thesis, Stanford University, Department of Mechanical Engineering, October 1968.

11. Roth, Bernard, "Performance Evaluation of Manipulators from a Kinematic Viewpoint", Performance Evaluation of Programmable Robots and Manipulators, NBS Special Publication #459, October 1976.

12. Groome, R.C., "Force Feedback Steering of a Teleoperator System", Master's Thesis, Massachusetts Institute of Technology, August 1972.

13. Nevins, J.L., Whitney, D.E., and Simunovic, S.N., "System Architecture for Assembly Machines", Report # R-764, Charles Stark Draper Laboratory, Inc., Cambridge, MA., November 1973.

14. Roth, B., "Position Paper on Manipulation Geometry", NSF Workshop on Impact on the Academic Community of Required Research Activity for Generalized Robotic Manipulators, Gainesville, Fla., February 1978.

15. Vertut, J., "Experience and Remarks on a Manipulator Evaluation", Performance Evaluation of Programmable Robots and Manipulators, NBS Special Publication #459, October 1975.

16. Duffy, J. and Tesar, D., Proposal for the Conceptual and Analytical Design of Manipulator Based Systems, University of Florida, 1977.

17. Tesar, D., "A Review of Modelling, Control, and Design of Manipulators in Terms of the Planar 3 DOF System", NSF Workshop on the Impact on the Academic Community of Required Research Activity for Generalized Robotic Manipulators, Gainesville, FLA., February 1978.

18. Uicker, J.J., Jr., "Some Unanswered Questions on the Kinematic Analysis of Robots and Manipulators", NSF Workshop on the Impact on the Academic Community of Required Research Activity for Generalized Robotic Manipulators, Gainesville, FLA., February 1978.

19. Corwin, M., "A Computer Controlled Robot for Automatic Manufacturing", International Symposium on Automotive Technology and Automation, Wolfsburg, West Germany, September 1977.

20. Bejczy, A.K., "Issues in Advanced Automation for Manipulator Control", Joint Automatic Control Conference, Purdue University, Indiana, July 1976.

21. Winston, P.H., Artificial Intelligence, Addison-Wesley Publishing Company, Reading, MA, 1977.

22. Freedy, A., Hull, F., and Lyman, J., "A Computer Based Learning System for Inhibitory Teleoperator Control", Proceedings of the Sixth Annual Conference on Manual Control, Wright Patterson AFB, Ohio, April 1979.

23. Park, W.T., "Mini-Computer Software Organization for Control of Industrial Robots", Proceedings Joint Automatic Control Conference San Francisco, June 1977.

24. Crooks, W.H., Shaket, E., and Alperowitch, Y., "Man-Machine Communication in Computer-Aided Remote Manipulation", Technical Report # PATR-1034-78-3, Perceptronics, Woodland Hills, CA., March 1978.

25. Albus, J.S., "The Control of a Manipulator by a Computer Model of the Cerebellum", Remotely Manned Systems, California Institute of Technology, 1973.

26. Bejczy, A.K., "Distribution of Control Decisions in Remote Manipulators", IEEE Conference on Decision and Control, Houston, TX, December 1975.

27. Sword, A.J. and Park, W.T., "Adaptive Control Study for Industrial Robots", Final Report, Stanford Research Institute, California, October, 1974.

28. Fu, K.S., "Learning Control Design and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control", IEEE Transaction on Automatic Control, Vol. AC-16, No. 1, February, 1971.

29. Whitney, D.E., "Resolved Motion Rate Control of Resolved Manipulators and Human Prostheses", Fifth Annual NASA-University Conference on Manual Control, MIT, Cambridge, MA., March 1969.

30. Whitney, D.E., "The Mathematics of Coordinated Control of Prostheses and Manipulators", Eighth Annual Conference on Manual Control, University of Michigan, May 1972.

31. Paul, R.P., "Modelling Trajectory Calculation and Servicing of a Computer Controlled Arm", Ph.D. Dissertation, Stanford University, August, 1972.

32. Shimaro, B.E., "The Kinematic Design and Force Control of Computer Controlled Manipulators", Ph.D. Thesis, Stanford University, March 1978.

33. Rosen, C., et. al., "Machine Intelligence Research Applied to Industrial Automation", Report, Stanford Research Institute, August 1977.

34. Johnson, E.G., and Corliss, W.R., Human Factors in Teleoperator Design and Operation, John Wiley & Son, 1971.

35. Weltman, G. and Freedy, A., "Control Automation in Undersea Manipulator System", Second Conference on Remotely Manned Systems, Los Angeles, California, June 1975.

36. Shaket, E. and Crooks, W.H., "Man-Machine Communication in Computer Aided Manipulation", Status Report, Perceptronics, Woodland Hills, California, May 1977.

37. Verplank, W.L., "Symbolic and Analogic Command Handware for Computer-Aided Manipulation", Master's Thesis, M.I.T., 1966.

38. Morgan, C.T., et.al., (eds.), Human Engineering Guide to Equipment Design, New York, McGraw-Hill, 1963.

39. Sheridan, T.B. and Ferrell, W.K., Man-Machine Systems, M.I.T. Press, Cambridge, MA., 1974.

40. Hill, J.W. and Sword, A.J., "Manipulators Based on Sensor-Directed Control : An Integrated End Effector and Touch Sensing System", 17th Annual Human Factors Society Convention, Washington, D.C., October 1973.

41. Grossman, D.D., Taylor, R.H., "Interactive Generation of Object Models with a Manipulator", IEEE Transaction on Systems, Man, and Cybernetics, Vol. SMC-8, No. 9, September 1978.

42. Unimation, Inc., "User's Guide to VAL", User's Manual, Unimation, Inc., Danbury, Conn., November 1978.

43. Hohn, R.E., "Application Flexibility of a Computer-Controlled Industrial Robot", Technical Paper #MR76-603, Society of Mechanical Engineers, 1976.

44. Ferrell, W.R., "Command Language for Supervisory Control of Remote Manipulation", Remotely Manned Systems, California Instit-tute of Technology, 1973.

45. Rechritzer, A.B. and Sutter, W., "Naval Applications of Remote Manipulation", Remotely Manned Systems Exploration and Operation in Space, California Institute of Technology, 1973, pp. 493-502.

46. Schneider, M.R., "Task Analysis for Undersea Manipulators", Master's Thesis, Man-Machine Systems Laboratory, M.I.T., March 1977.

47. Winget, C.L., "Hand Tools and Mechanical Accessories for a Deep Submersible", Remotely Manned Systems Exploration and Operation in Space, California Institute of Technology, 1973, pp. 505-523.

48. Bejczy, A.K., "Performance Evaluation of Computer-Aided Manipulator Control", 1976 IEEE International Conference on Systems, Man and Cybernetics, Washington, D.C., November 1976.

49. Black, J.H., "Factoral Study of Remote Manipulation with Transmission Time Delay", Master's Thesis, M.I.T., December 1970.

50. Hill, J.W., "Two Measures of Performance in a Peg-in-Hole Manipulation Task with Force Feedback", Thirteenth Annual Conference on Manual Control, Cambridge, MA, June 1977.

51. Mullen, D.P., "An Evaluation of Resolved Motion Rate Control for Remote Manipulators", Master's Thesis, M.I.T., January 1973.

52. Pesch, A.J., Bertsche, W.R., "Performance Measurement for Undersea Systems", Performance Evaluation of Programmable Robots and Manipulation, NBS Special Publication #459, October 1976.

53. Pieper, D.I. and Roth, B., "The Kinematics of Manipulator Under Computer Control", Proceedings 2nd International Congress on the Theory of Machines and Mechanisms, 1969, Vol. 2, pp. 159-68,

54. Berson, B.L., Crooks, W.H., Shaket, E., and Weltman, G., "Man-Machine Communication in Computer-Aided Remote Manipulation", Technical Report, Perceptronics, Woodland Hills, California, March 1977.

55. Park, W.T., "Robotics Research Trends", Technical Note #160 SRI International, Menlo Park, Ca.

56. Thompson, D.A., "The Development of a Six Degree-of-Constraint Robot Performance Evaluation List", Thirteenth Annual Conference on Manual Control, Cambridge, MA., June 1977.

57. Rogers, D.F., Adams, J.A., Mathematical Elements for Computer Graphics, McGraw-Hill Inc., 1976.

58. Bejczy, A.K., "Displays for Supervisory Control of Manipulators", Thirteenth Annual Conference on Manual Control, M.I.T., June 1977.

59. Sheridan, T.B., Verplank, W.L., and Brooks, T.L., "Human/Computer Control of Undersea Teleoperators", Fourteenth Annual Conference on Manual Control, Los Angeles, Ca., April 1978.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SUPERMAN: A SYSTEM FOR SUPERVISORY MANIPULATION AND THE STUDY OF HUMAN/COMPUTER INTERACTIONS		5. TYPE OF REPORT & PERIOD COVERED Technical Report 15 June 1978 - 30 June 1979
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) THURSTON L. BROOKS		8. CONTRACT OR GRANT NUMBER(s) N00014-77-C-0256 04-7-158-44079
9. PERFORMING ORGANIZATION NAME AND ADDRESS Massachusetts Institute of Technology Cambridge,MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 196-152
11. CONTROLLING OFFICE NAME AND ADDRESS Engineering Psychology Programs Office of Naval Research (Code 455) Arlington, VA 22217		12. REPORT DATE May 11, 1979
		13. NUMBER OF PAGES 280
~~14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)~~ and MIT Sea Grant Program Office of Sea Grant,Nat'l Ocean.& Atmos. Admin. U.S. Dept. of Commerce,Washington, D.C.		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Supervisory Control	Robotics
Computer Control	Human Performance
Remote Manipulators	Manual Control
Teleoperators	Man-Machine Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis considers the need for supervisory control of remote teleoperator vehicles in the ocean environment, and shows that computer controlled systems can increase the effectiveness of remote manipulation.

A distinction is made between absolute tasks (tasks which have a known spatial relationship to the manipulator base prior to execution) and relative tasks (tasks which cannot be spatially defined prior to execution).

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.
A second distinction is made between fixed tasks (tasks which remain fixed with respect to the manipulator base during execution) and moving tasks (tasks which continuously move with respect to the manipulator base during execution). Four distinct combinations can be made from this 2x2 array: (1) fixed-absolute tasks, (2) fixed-relative tasks, (3) moving-absolute tasks, and (4) moving-relative tasks. Mathematical principles are developed to deal with each of these four possibilities.

A unified theoretical framework of supervisory manipulation is considered to give the designer an overview of: (1) manipulator and processor selection factors, (2) interface design considerations, (3) control language attributes and implementation factors, and (4) control philosophies.

Based on the mathematical and theoretical foundations described in this thesis, a supervisory system was developed and demonstrated.

The major conclusion derived from this study is that even under the "best" control conditions (i.e., no time delays, no frame rate problems, high visibility, etc.) supervisory control can improve system performance for all forms of manual control except master/slave with force feedback.