TOUCH FEEDBACK AND AUTOMATIC CONTROL

J. W. Hill

Summary

A philosophy of automatic control based on both coded and analog information is described. It is shown how a simple arm automaton can be constructed and communicated with in a natural language using both coded and analog information to carry out tasks with a 7-degree-of-freedom manipulator, using three increasingly automatic levels of supervisory control. To a considerable degree, this supervisory control is based on information from touch sensors mounted on the end effector. The use of tactile information in manipulation is studied in two ways. Forst, such information conveyed directly to the hand of the human operator allows him to be more afficient, avoiding drops and fumbles, and allows him to carry out tasks with poor or restricted vision that could not otherwise be carried out. Second, tactile information provided to the computer controller enables the mechanical arm to react with simple automatic reflexes and to act with automatic grasping abilities. On the basis of this research a combined manual-automatic operating system consisting of a battery of automatic subroutines baised on touch sensors is proposed for a prosthetic hand.

Introduction

Commands for movement, which come down the spinal cord, are currently thought to be length commands to individual muscles, which are in a length-control servo loop /1/. Yet it is not known how man, who perceives, thinks, and acts in terms of near, far, left, right, away from, toward, and so on, communicates with his muscles to achieve desired results in these terms. By studying how man does things, and trying to synthesize the sequence of control and sensing actions he takes, we can simulate, experiment with, and perhaps eventually understand his communication with his muscles. Such an approach may allow us to determine what sensory information should be measured and reproduced for the human using a teleoperator or prosthetic device, to make his job more realistic and allow him to project himself into his work. Such information is the key to determing what types of movements and an effector should be capable of making, and what type of sequential organization is necessary for supervisory control and automatic programming.

This research was carried out under Contracts NAS2-5409 and NAS2-6880 from the National Aeronautics and Space Administration.

An example of an analysis of action is illustrated in Figure 1. Here it is assumed that a man has accidentally dropped a coin on the ground, and the figure shows his actions as he reaches down to pick it up. The analysis illustrated in Figure 1 is roughly the following:

(1) Bending over, he puts his hand in the pincher grip position with tumb and index finger opposing. The remaining three fingers are gently clenshed or sometimes fully extended.

(2) Reaching down, he feels unstable; he stabilizese himself with his thumb (resting some weight on it), picking up the coin with the index finger.

(3) His thumb touches the ground--Figure 1 (a).

(4) He continues pushing until his index finger touches the ground--Figure 1 (b).

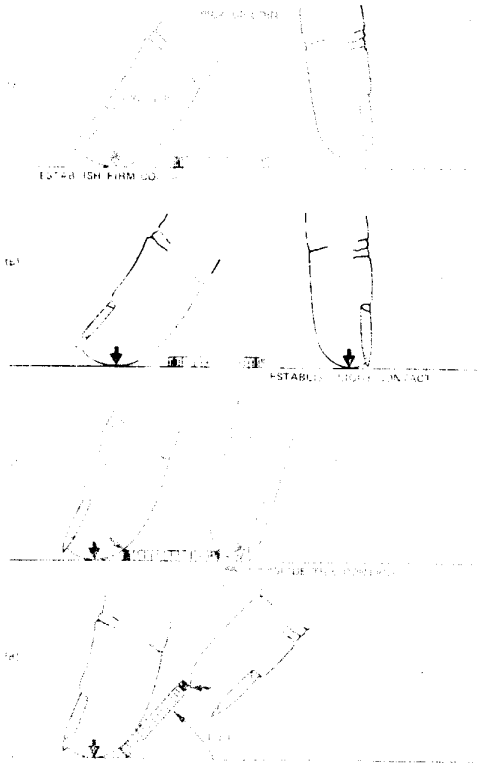

Fig. 1. Analysis of hand picking up coin

(5) He slides his index finger on the ground toward his thumb until the coin is felt against the thumb--Figure 1 (c).

(6) Maintaining a finger-thumb squeeze, he lifts the index finger and rotates the coin--Figure 1 (d).

(7) When the rim of the coin touches his thumb, he applies more closing force and lifts his hand.

This analysis tells us that, to follow this plan with a teleoperator or with automatic control, we need touch information from several areas on the hand. Similarly, an end effector is needed with ability to retract the index finger, as well as the normal ability to close index finger against thumb.

A sequential breakdown sets a series of constraints on an automatic control system designed to carry out the task. The controller must be able to:

(1) Maintain finger tip contact while closing prosthesis fingers.

(2) Close the fingers until the thumb is touched.

(3) Raise and close the index finger around a nutural hinge point.

(4) Detect when an object is safely gripped.

A general purpose arm controller must not only be able to establish a wide variety of control loops on the basis of touch and position information but also be able to carry out a sequence of these control loops by branching from one to the next when certain completion criteria have been met. For the coin pick up example, the flow chart of Figure 2 was prepared to show the sequence of steps an arm automation must follow through.

After a range of such tasks has been anayzed, it becomes apparent that tactile information and sequenced automatic control play important roles in manipulation. Therefore, the three steps below are suggested as a general strategy for picking up an object.

(1) A man looks at an object, and a sequence of actions and reflex-like responses takes form in his mind. (Included in his plan are a desired initial and final hand--or arm--posture. The plan may even be formed during the first movement toward the object.)

(2) He puts his hand in the desired initial posture.

(3) He initiates a motion toward the object and reflexively carries out his plan of acquistion, using touch, force, and position information as required.

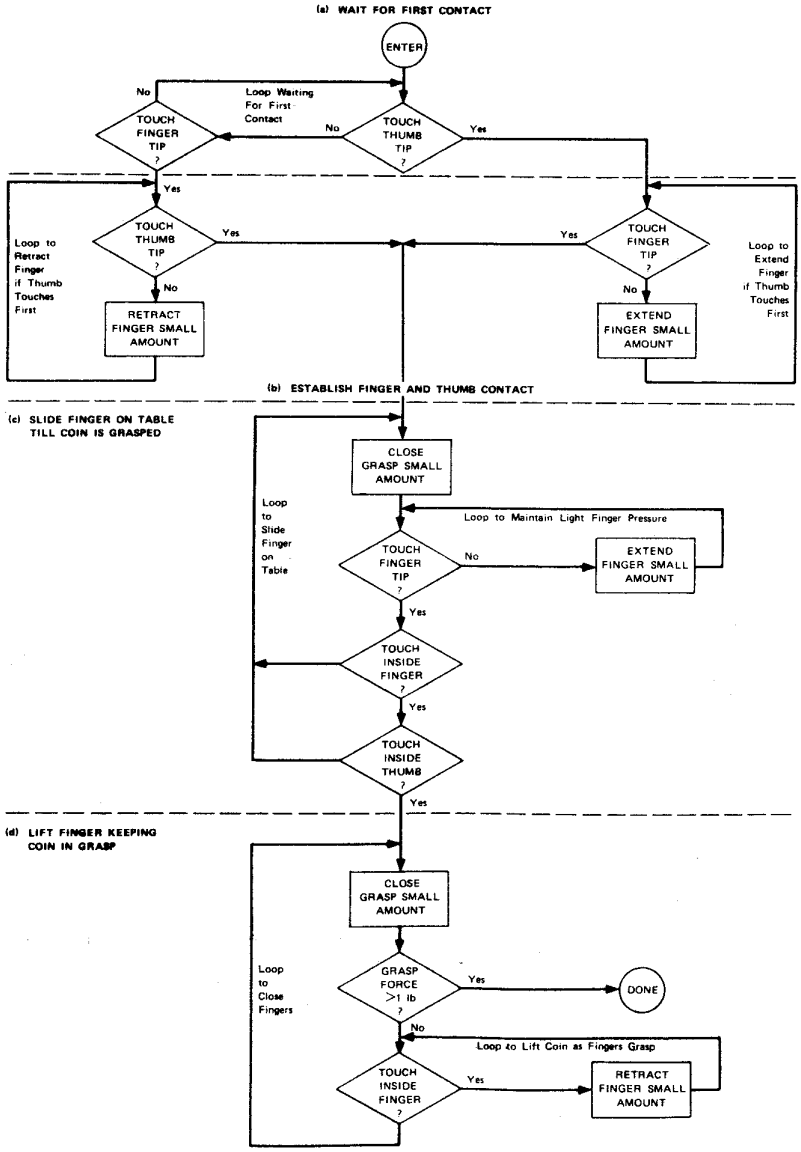

Fig. 2. Flow chart for hypothetical program to pick up coin from tabletop

In this strategy, tactile information is used for alignment, e.g., for positioning an object in the hand or locating exactly where an object is and for limitation of the forces applied to an object. Sequenced automatic control is evident in our every practiced move; we can unscrew a nut from a bolt, knit, or write our signature, all quickly and without looking. The combination of programmed motions and touch sensing permits a wide range of tasks to be carried out automatically /2/.

These two important elements of man's skill and ability to manipulate are not normally supplied by teleoperator control systems. Following the approach suggested at the beginning of this paper, we have experimented (1) with tactile feedback by incorporating a touch sensing and touch diplay system into a teleoperator and (2) with different levels of automatic control by incorporating supervisory control features in teleoperator control system. The concepts on which these experiments have been based are explained in the following sections of the paper.

Touch Sensing and Display

One way of understanding the contribution of touch sensing to man's manipulative skills is to provide a teleoperator with different kinds of feedback touch from slave to master for teleoperator control. Carrying out manipulations with the touch feedback on or off, we can determine what skills the new information has provided. After identifying a skill in this way, we also have an existence proof that the same skill can be obtained from an automatic control system utilizing the same tactile information.

Two touch-feedback systems for the teleoperator control system have been constructed for experimental evaluation. Each system consists of a set of sensors mounted on a mechanical arm and a corresponding set of tactile simulators mounted on a control brace. All the sensors are constructed of conducting rubber that is deformed to complete and electrical circuit upon physical contact (see Ref. /3/ for construction details). The construction of most of them depends on etched wiring on printed circuit boards. Individual sensors activate corresponding stimulators in a binary fashion: A stimulator is either full on or full off.

The hand contact system senses and reproduces to the operator contact between the end effector and the object being touched or manipulated. This system consists of a number of conducting rub-

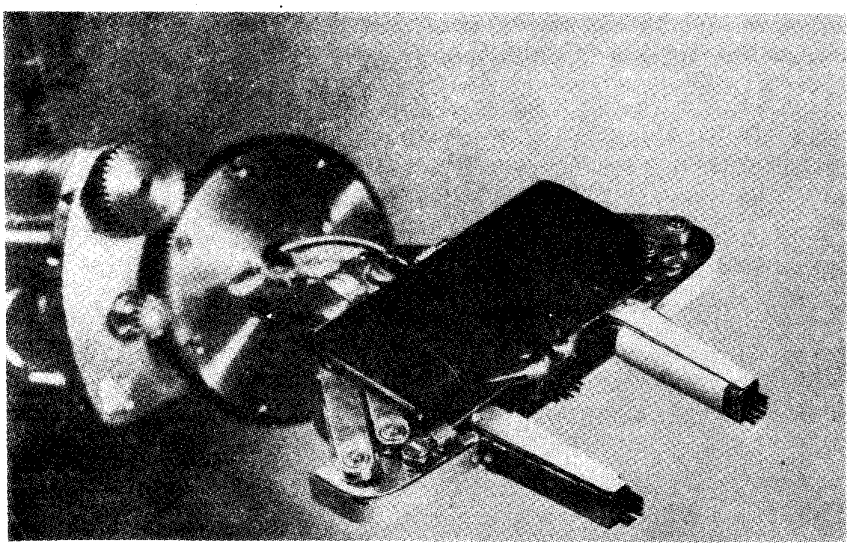

Fig. 3. Construction of touch sensors

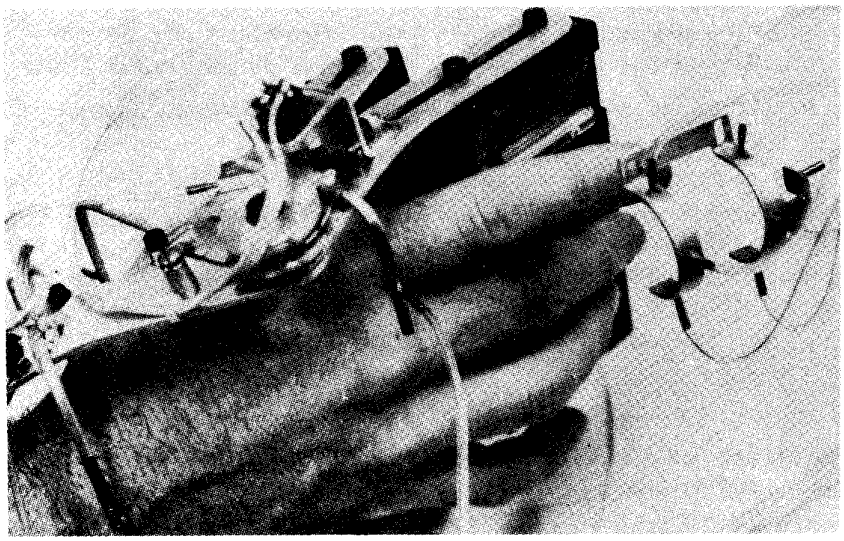

Fig. 4. Tactile simulators on the hand controller

ber sensors mounted on the outside surfaces of the mechanical hand, as shown in Figure 3. The tongs of the hand are completely covered with these sensors (seven sensors per tong), as are the extreme or protruding parts of the upper hand (seven sensors). The sensors are so arranged that any contact of the hand with a flat surface is sensed and that any contact with the tongs is sensed. Each sensor is connected via amplifying and gating circuits to an air-jet tactile stimulator. The air jets are positioned on the control brace to produce touch sensatations on portions of the operator's hand corresponding to the locations of the sensors. Each jet produces an area of pulsating pressure on the skin approximately 3/16 inch in diameter. The arrangement of air-jet stimulators on the control brace is shown in Figure 4. The construction of the air-jet stimulators was described /4/. By using the external sensory feedback system, it is possible to (1) reach into a box without the aid of vision and extract of block from it, (2) locate a visually obscured object to be picked up under the tongs and respond by grasping it, and (3) control wrist rotation so that both tongs rest on a flat surface.

The jaw contact system senses and reproduces to the operator the shape and location of the object held in the remote jaws. Two sensing pads are built into the tongs of the mechanical hand (as shown in Figure 3). Each of these opposing pads consists of 144 individual contacts in a 6 by 24 rectangular pattern. Two corresponding 6 by 24 rectangular arrays of bimorphs containing the index finger and thumb are built into the control brace, as shown in Figure 4. Bimorphs produce 250 Hz vibration of the skin, restricted to an area about 1 mm in diameter. Thus the pattern of contact closures is reproduced as a pattern of vibration, enabling the operator to feel on his own thumb and index finger the shape and location of the object held in the remote jaws. A complete description of similar bimorph arrays used in shape recognitation and reading experiments has been given by Bliss /5/ and Bliss et al. /6/. By using the jaw shape sensing system it is possible to (1) pick up an object in the desired part of the tongs, as in the center or at the tip, (2) obtain rotational alingment to a fixed object while gripping it, (3) detect slippage when lifting an object and close the jaws until the slippage has stopped, and (4) pick up an egg without breaking it.

From analyses of such manipulation tasks as that of picking up the coin, it is evident that a great deal of flexibility is required in experimenting with arm control modes. Because of this requirement the entire automatic controller has been simulated, using our laboratory computer (a LINC-8 with 8192 words of memory). The simulation has separate programs for inputting manual control information, performing servo-control functions, and carrying out automatic arm control. These three programs are sequentially serviced 60 times a second. During each 1/60-second epoch, manual control inputs (if any) are accepted, and a new instruction is formed for the automatic controller. Next, the servo calculations are carried out, and, finally, the automatic processor is allowed one sequential operation based on an instruction. The chief advantages of this simulation are the flexibility and speed with which the control structure and parameters can be included or modified through program (software) rather than equipment changes. The 60 Hz sampling rate has been found adequate for preserving human accuracy and smoothenes of operation.

A block diagram of the automatic controller is given in Figure 5. The instruction (two 12-bit words) and the analog joint commands (seven 12-bit words) from the control station are the only inputs. The automanual switch is under program control and can be either closed to accept manual inputs from the operator or open to allow commands generated by programs to move the arm.

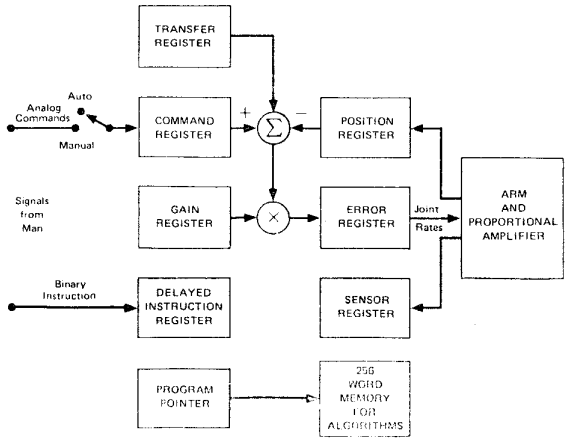

Fig. 5. Components of the computer controller

Arm control is quite conventional, with actual joint positions (obtained by analog-to-digital conversion) subtracted from the command joint positions, and the difference multiplied by the joint gains and then output to the servo motors (via digital-to-analog conversion) to establish angular rates. The transfer register is used to offset the analog commands, so that control can be "transferred" to the human operator smoothly after an automatic operation has been completed.

Discrete instructions transmitted by the human operator are saved in the instruction register. The basic form of the instruction specified a test, an action, and a numerical parameter. The 256-word memory contains sequential lists of instructions for carrying out manipulation tasks, and the program pointer indicates where the next instruction from the memory list will be found.

Not shown in Figure 5, but essential to the operation of the automatic controller, is the instruction processor shown in Figure 6. The processor transfers numbers between registers and carries out sensor and position tests on the basis of individual instructions. These instructions are the building blocks used by both single commands from the control typewriter and multiinstruction programs of manipulation.

A single instruction requests that a specific test be executed and that a specific command be carried out if the test is passed. The first half of the instruction word (6 bits) is used to select one of 64 possible tests by means of a look-up table. If the test is pased, the second half of the instruction (6bits) is similarly used to select one of 64 possible actions. Even though only 19 tests and 25 actions have been implemented, a rich variety of operations is already possible. An example of a single instructio is:

If fingertip sensor closed, then open jaws.

For the computer, this instruction is coded by FO,where the fingerprint sensor closed test is specified by F and the open jaws command is specified by O. Table 1 lists representative tests and actions.

Algorithmic Language for Remote Manipulation (ARM)

If request for the automatic operations described in the preceding section are taken from a list, the list can be considered a program of motions (an algorithm) to carry out a manipulation

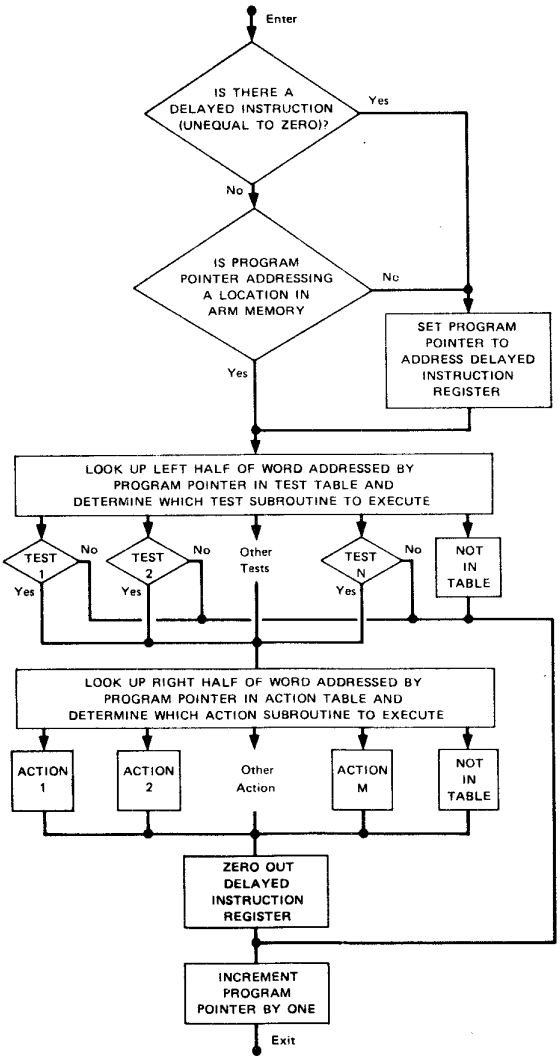

Fig. 6. Instruction processor
This subroutine carries out single instructions and programs of instructions. It can be seen that, if instructions are being taken from a program in arm memory, an instruction sent from the control station will cause the program to be stopped and the instruction to be carried out.

task. The effective utilization of such a program, however, requires a means for writing it in an easy-to-use language and a means for assembling (or generating) a list of arm operations from the statements in the language. Under the constraint of a small computer system, we simultaneously developed the separate concepts of the ARM language, the assembler, and the instruction set for the automation controller previously described.

ARM is an extension of the MHI or THI language developed by Ernst /7/ and of MANTRAN developed by Barber /8/, in that manual inputs from the operator can be used in addition to teletype input. Thus, the operator can move his control brace and reqest that the arm move to "this" position or move "this" joint "this" much. Each "this" is the preceding sentence is a manually specified quantity that is difficult to verbalize, much less to quantify as joint vector for typewriter input.

An example of a program (writen in ARM), the algorithm for the actions of picking up a coin previously described, is given in Table 2. An entire program is given to an assembler for conversion to a list of numbers (instructions) for execution by the remote computer. Compiling is quite straightforward: Values for the various symbols on each line are simply added together to form the instruction.

Even after the pickup program of Table 2 has been loaded into the controller's memory and started, the grasping sequence is not begun until the index finger or thumb of the end effector has been brought into contact with the object. After the sequence of moves has been finished, the program returns smoothly to the manual control mode by the transfer command. If for any reason the operator wishes to stop in mid-task, he need only transmit any one instruction to the controller.

As can be seen from the coin pickup program (Table 2), the language is quite simple and powerful, needing only 42 12-bit instructions and two command vectors for carrying out the coin pickup. The longest ARM program written to date uses 60 instructions and 42 storage locations to direct a 7-degrees-of-freedom manipulator to unscrew a nut from a bolt and deposit it in a receptacle /3/. The length of this program could have been halved if inclusion of special procedures had not been necessary to compensate for inprecision in the manipulator and control system used.

Table 1

REPRESENTATIVE TESTS AND ACTIONS

Tests

Finger Bottom-- Either of the sensors on the bottom of the index finger is on.

Anysensor--One or more of the external contact sensors on the end effector are on.

Grab--More than "Sthresh" sensors of the 288 sensors on the finger pads are on. Sthresh is changed by the instruction "Set Sthresh; X"

Done--All the components of the error vector are less than "Epsilon" Epsilon may be changed by the instruction "Set Epsilon;X". The done test indicates whether the arm has finished moving to a new command position.

Jaw--The jaw opening is greater than "Jsize," which is changed by the instruction "Set Jsize; X." Different actions can be carried out, depending on the size of the grasped object.

Simple Actions

Auto--Causes the analog commands switch to open so that the command register can be used for automatic operations.

Close--Causes the jaws to close by placing the closed jaw angle in the proper entry of the command register.

Stop--Causes the contents of the position register to be placed in the command register, stopping the arm.

Transfer--Subtracts the contents of the commands register from incoming analog commands, places the result in the transfer register, and closes the auto-manual switch. This allows control to be taken over externally with no transient motion.

Set Timer; X -- Loads the timer with the contents of the next memory cell (timer set to X). The number in the timer, which is reduced by one every 60 seconds, can be tested to limit the length of a move.

Go To; X--loads the program pointer with X so that instruction X will be the next one executed.

Vector Actions

Define; X--Contents of the position register are stored in memory, starting at location X. This stores the set of joint angles (a posture, or, equivalently, a position of the end effector in space) to be returned to later.

Move to; X-- The contents of the position register are replaced by the contents of memory starting at location X, thus causing the arm to assume the new joint angles.

Increment by; X--The contents of memory starting at location X are added to the contents of the command register.

Set gains from; X--The contents of the gain register are replaced by the contents of memory starting at location X. Reducing a particular gain makes the joint more "spongy"; setting a gain to zero makes the joint free to move or comply with external forces.

Table 2

COIN PICKUP PROGRAM

```
BEGIN

WAIT:     IF THUMBTIP GO TO; EXTEND          /Loop Waiting for First Contact
          IF FINGERTIP GO TO; RETRACT
          GO TO; WAIT

EXTEND:   IF FINGERTIP GO TO:  SLIDE         /Loop to Extend Finger if Thumb
          INCREMENT BY; DX                   /Touches First
          GO TO; EXTEND

RETRACT:  IF THUMBTIP GO TO; SLIDE           /Loop to Retract Finger if Finger
          DECREMENT BY; DX                   /Touches First
          GO TO; RETRACT

SLIDE:    DECREMENT BY:  DOPEN               /Slide Finger on Table Till

S1:         IF FINGERTIP GO TO; S2           /Coin is Grasped
            INCREMENT BY; DX
            GO TO S1

S2:       IF INSIDE FINGER SKIP 2
          GO TO; SLIDE
          IF INSIDE THUMB GO TO; LIFT
          GO TO; SLIDE

LIFT:     DECREMENT BY; DOPEN                /Lift Finger Keeping Coin in Grasp
          IF JAWCLOSED TRANSFER              /Return Control to Man

L1:         IF INSIDE FINGER SKIP 2
            GO TO:  LIFT
            DECREMENT BY; DX
            GO TO; L1

DX:       1; -1; 0; 0/Incremental Command Vector to Extend Index

DOPEN:    0, 0, 0; -1/Incremental Command Vector to Open Fingers

END
```

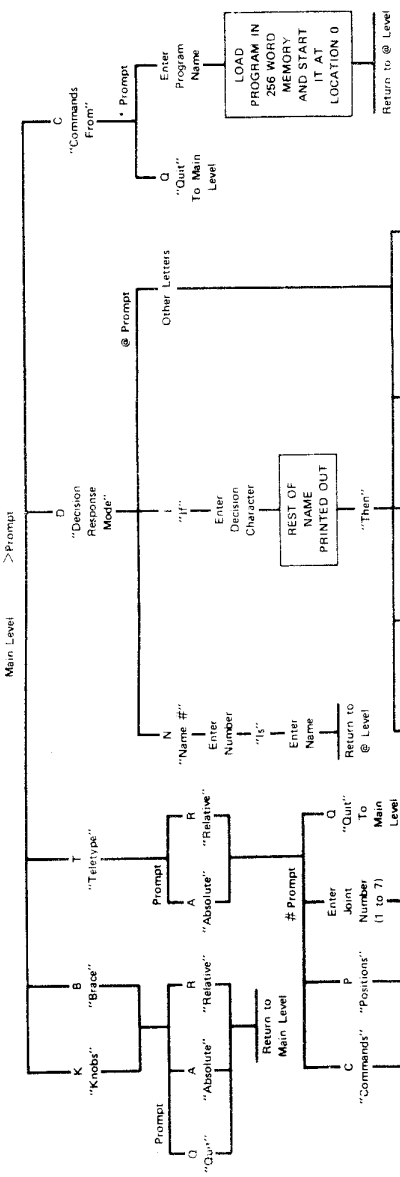

Fig. 7. Simplified command tree for the supervisory control station

Supervisory Control

Our goal is to design a control scheme that optimizes performance in carrying out remote tasks by combining the best attributes of man and computer. Therefore, man's ability to interpret scenes, estimate distances, and project motion with a multicoordinate control brace is combined with the computer's ability to save and accurately duplicate arm positions, remember sequences of motions, carry out tests based on arm position, and interpret touch sensors. General background material on such supervisory control has been given by Johnson and Corliss /9/, /10/.

The supervisory control scheme for interacting with the previously described automatic controller is governed by the command tree shown in Figure 7. Moving from branch to branch by taping single letters, numbers, or short names, the human operator can specify communication options from three increasingly automatic levels of control. After the operator has typed a character on a keyboard, specifying which control branch he wishes to take, the computer first prints out one of the short massages indicated in Figure 7 by quotation marks; it then prints, on a new line, the prompting symbol for the new control branch. This approach limits the amount of coded information required to carry out a task, permitting use of a typewriter keyboard with one hand and the control brace or joystick for manual control with the other. A simple five-finger binary keyboard or even a telegraph key might be more satisfactory than a typewriter keyboard.

The first, or manual, mode of control is obtained by typing K (for knobs, a bank of potentiometers), B (for a 7-degrees-of-freedom Rancho control brace worn on the right arm), or T (for teletype), followed by A (for absolute) or R (for relative). Absolute control causes joint positions to be read directly from these devices and to be transmitted to a remote station. Relative control specifies that joint commands from the control source take up where the previous joint commands left off. Thus, after a relative transfer, the new control source continues where the old one left off, and there is no transient motion artifact.

The second, or decision-response, mode of control is obtained by typing D; it permits instructions to be transmitted to the arm automation. If I is typed, the computer prints IF and is prepared for a character identifying an action. For example, Tests T is "thumb sensor closed", and Action O is "open jaws". There-

fore, as a sequence I, T, O is typed, the instruction "If Thumb then Open" is printed out, coded, and transmitted to the automation (manually entered letters are underlined here). In conjuction with the vector actions (Define, Move to, +Increment by, and -Increment by), it is possible to use names previously entered that correspond to addresses of joint-angle vectors in memory. Thus, after entering the name BOX in the catalog, it is possible to enter the commands:

Define BOX

Move to BOX

If thumb then move to BOX

All of the 19 tests and 26 actions built into the automatic controller are thus executable by typing a few characters under decision-response control.

The third, and most automatic, mode of supervisory control is obtained by typing C (Commands from*) followed by the name of a program previously constructed by using ARM language. This causes the program of tests and actions to be loaded into the remote computer's 256-word memory. The supervisory controller then switches the operator to the decision response mode for further interaction with the program. Position of objects the program needs to know about are input by using the Define command. This command causes the seven joint angles of the arm to be stored at the program location associated with the name in the catalog. These named vectors are stored in a region outside the 256-word memory and are accessible to any program run.

Currently, a series of ARM programs are being developed to perform a variety of useful tasks based on remote information from touch sensors and joint angles. These programs control the arm to unscrew a nut from a bolt, search for an object on a tabletop, align the jaws with a fixed object, center and object in the jaws horizontally and vertically, hold an object lightly without letting it slip, and yield to an external force. When a working library of these programs becomes avilable, a fourth, and more useful, mode of automatic control can be constructed, to link ARM programs as subroutines.

Applications to Prosthetic Control

Although the work described in this paper has been concerned with control of a mechanical arm, the advantages of sensory feedback and automatic control are also applicable to the design of prosthesis devices. An example, consider the problem of controlling a prosthetic hand with the 24 degrees of freedom of the human hand. This end effector would have 24 motors pulling on 24 cables connected to the points on the model hand corresponding to the points where the tendions of the human hand anchor. Obviously, it is impossible to control the 24 degrees of freedom directly by using separate myoelectric potantials from the amputee's stump, where there may only be one such analog signal avilable.

Instead of using one-to-one analog control on 24 channels, a better way to solve this control problem would be to use one analog channel and one channel for coded instructions. The analog channel would be conventional, with myoelectric potentials used to provide a variable control signal. The instruction channel would convey information to the prosthesis controller governing

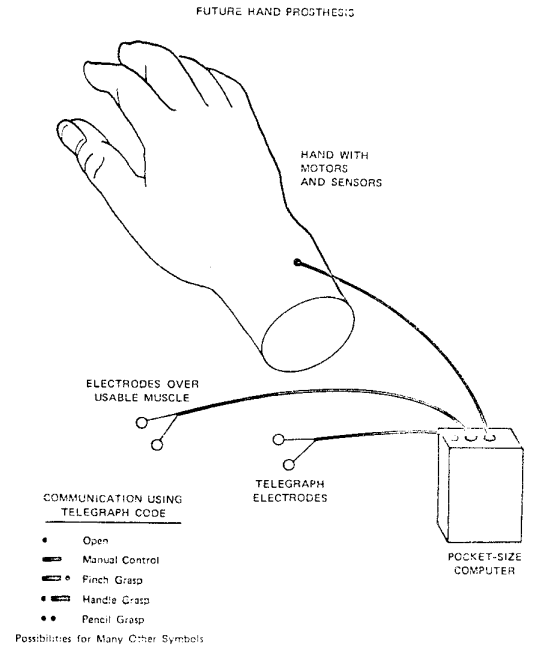

Fig. 3. Suggested code-controlled hand prosthesis

the use of the analog channel. The source of the discrete, coded information could be a single free muscle in the body that could communicate information such as characters of the international telegraph code. Through a series of dots and dashes, individual letters and numbers could be transmitted to a prosthesis control computer. Such a hand-prosthesis supervisory control system is shown in Figure 8.

An series of examples will most easely explain this approach. The code "dash-dot" could signify that the prosthesis controller should give commands to extend the index finger and thumb and commands to curl up the remaining fingers. Additionally, the computer would interpret the single analog signal as the control source to move the thumb and index finger in opposition to form a pincher grip. We would identify this dash-dot symbol as the pincher grip instruction. Other similar control modes for grasping a handle, pencil, ball, and so forth could be set up upon receipt of particular instructions. One format for governing these gripping procedures is:

- Establish an initial posture.
- Choose an incremental control vector that, when added to the 24-element command vector, causes the end effector to produce the correct movement.
- Set the gains register to fix or free (whichever is appropriate) the degrees of freedom that do not change.
- Use the analog signal to control repetitive addition or subtraction of the new incremental vector and the control vector.

A parallel can be drawn between manual control with the supervisory arm control system previously described (knobs, brace, and so on) and manual control with a prosthetic hand. By extending this relationship, it would be possible to adapt the previous suprevisory control approach to implement a"hand"decision-response mode and the "hand" programming mode. Both would be based on signals from the 24 cable-pulling servos in the hand and a network of touch sensors distributed on its surface. Because of the close correspondence of the arm and hand control systems, the implications for further development of automatic hand control will be left to the reader.

Practically, though, what about the possibility of constructing a small, pocket-sized computer controller for this prosthesis? Estimates of the size required, based on the previously described control of a mechanical arm, suggested that approximately 1,000 10-bit words of memory would be needed to implement a reasonable range of automatic operations. This is roughly the complexity (35,000 transistors) of a pocket-sized electronic slide-rule calculator currently being marketed for $ 400.

The problem in building this realistic hand rests, therefore, not with the automatic computer controller or with limited manual inputs available for control, but with the mechanical design problem of building 24 servomotors of the desired power into the space available. With this supervisory control approach, the bottleneck in designing a lifelike hand prosthesis is returned to the mechanical designer.

If such a mechanical design and supervisory controller were put together, we could conceive of a man sending telegraph codes to the device and it responding by moving a pencil to print the letter corresponding to the code character sent. A single code symbol could generate a signature! This brings us back to where we began, with the problem of how man controls muscles: What kinds and what levels of commands does he use to produce the motions we see?

References

/1/ Merton, P.A., "How We Control the Contraction of Muscles", *Scientific American* Vol. 226, 30-37, May 1972.

/2/ Tomovic, R., "Outline of a Control Thecry of Prosthesis", *Proc of the Second Congress of International Federation of Automatic Control*, Basle, V. Broida, ed., London; Butterworths, 1964.

/3/ Hill, J.W. and Bliss, J.C., *Tactile Perception Studies Related to Teleoperator Systems*, Final report 2, Contract NAS2-5409, Stanford Research Institute, Menlo Park, Ca., 1971.

/4/ Bliss, J.C. and Crane, H.D., *Experiments in Tactual Perception*, Final report, Contract NAS2-1679, Stanford Research Institute, Menlo Park, Ca., 1965. Appendix B

/5/ Bliss, J.C., "A Relatively High-Resolution Reading Aid for the Blind", *IEEE Trans.*, Vol. MMS-10, 1-9, 1969.

/6/ Bliss, J.C. et al., "Optical-to-Tactile Image Conversion for the Blind", *IEEE Trans.*, Vol. MMS-11, 58-64, 1970.

/7/ Ernst, H.A., *A Computer Operated Mechanical Hand*, Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1961.

/8/ Barber, D.J., *MANTRAN: A Symbolic Language for Supervisory Control of an Intelligent Remote Manipulator*, DSR 70283-2 Engineering Projects Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., 1967.

/9/ Johnson, E.G. and Corliss, W.R., *Teleoperators and Human Augmentation*, NASA-SP-5047, Superintendent of Documents, U.S. Goverment Printing Office, Washington D.C., 1967., 69-74.

/10/ Corliss, W.R. and Johnson, E.G., *Teleoperator Controls*, NASA-SP-5070, Clearinghouse for Federal and Technical Information, Springfield, Wa., 1969., 7-124.