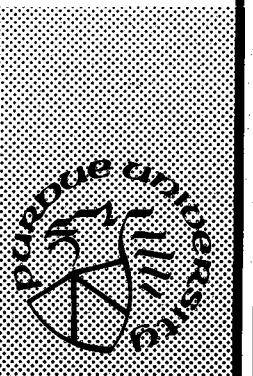
3-1-1987

A Laboratory Experiment on Robot Contouring with Force Feedback

Shaheen Ahmad
Purdue University



A Laboratory Experiment on Robot Contouring with Force Feedback

Shaheen Ahmad

TR-EE 87-8
March 1987

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

A LABORATORY EXPERIMENT ON ROBOT CONTOURING WITH FORCE FEEDBACK

Shaheen Ahmad
School of Electrical Engineering
Purdue University
West Lafayette
Indiana 47906
USA

Abstract

In this paper we describe a laboratory experiment which is part of a laboratory orientated robotics class taken by seniors and first year graduate students.

This experiment is designed to introduce students to real-time robot control system hardware and software. The experiment attempts to fortify material covered in an introductory (non laboratory orientated) class on robotics. The issues covered by this experiments include: kinematics, dynamics, robot drive mechanisms, interfacing of sensors and force control aspects. Students were also required to learn many aspects of real time programming for control applications.

We document this entire experiment so it may be reorganized and repeated. We discuss educational and research value of this experiment.

Introduction

Currently at Purdue, School of Electrical Engineering there are three robotics courses and a variable topics controls class, in which robotics research issues may be covered. The sequence of classes which a student may take is shown in Figure: 1. The laboratory class entitled "Real-time Robot Control Laboratory," is a second course in robotics which enrollment may consist of both seniors and graduate students. In this paper we address an experiment which the students taking this class perform. Section one of this paper describes the equipment, Section two describes the software which the students develop to perform this experiment. Section three describes the contour tracing algorithm used to follow an object utilizing force

feedback information. Section four discusses the results of the experiment and its educational value. Conclusion is presented in Section five.

1. Equipment Organization In The Experiment

The experimental workstation consists of a small (table-top) industrial robot manufactured by Nako-Nihon, a 'Multibus card-cage with an Omnibyte 68000 microprocessor single board computer, a multibus analog board from Burr-Brown, a Lord Corporations force/torque sensor, a programming terminal and a download box with an interface to a VAX 11/780 network.

The organization of the equipment is shown in Figure 2, photograph of the experiment workstation is shown in Figure 16.

The Naka-Nihon Robot Arm

The *Naka-Nihon* (NNK) robot arm is *the* major component of the experiment. It is a small industrial quality robot with five revolute joints: waist, shoulder, elbow, wrist pitch, and wrist roll. It comes with its own controller that primarily functions as a teaching device to teach the robot a number of points which the end-effector must pass, in different sequences. It also contains a serial port to allow an external system to command and inspect the robot joint positions through the controller. Velocity or torque control is not feasible with this controller and thus limits the robots functionality. The controller can only accept joint angles as its set points. It cannot servo through points, it comes to a zero velocity before next joint set point is accepted.

The Arm Configuration: Figure 3 is a top and side view of the robot arm when all angles are in the software defined zero position. Note that this position is a software zero defined in our kinematics program. The actual robot kinematic zero position as defined by the Denavit Hartenberg notation [Paul 81] cannot be reached by the robot. The direction of joint rotations as shown in Figure 3 are defined in 68000 software and may differ from the manufactures definition of the controller's angular direction. All the necessary offset calculations are carried out in the 68000 software. A more detailed description of the robot arm kinematics software will be dealt with in the kinematics section of this paper.

Interfacing The Robot with the 68000 SBC

The interface to the robot is through a RS232 serial port, The robot is hooked basically as a "dumb" slave machine which can accept a limited set of command sequences.

The Omnibyte 68000 SBC

The heart of the contour tracking system is the software residing on the *Omnibyte* 68000 single board computer (SBC) located in the Multibus card cage. This 68000 card contains 16K bytes of EPROM, 128K bytes of RAM, 4 parallel ports (2 PIA's), 2 serial ports (2 ACIA's), and a timer. This SBC is used as the master controller for the entire experiment. Software is downloaded into the SBC from ECN via a monitor residing in the EPROMs. A block diagram of the 68000 SBC is shown in Figure 4. All of the features of this card was used, including both ACIAs and both PIAs.

Lord Force/Torque Sensor

The Lord Corporation Force/Torque sensor is a mechanically rigid device that comes with a controller that will directly output force and torque information as 16-bit fixed point values, from a 16-bit parallel output port [LORD 85]. The force/torque sensor must be initialized through the serial port before it will output force information on the parallel port. This requires the user to disconnect the terminal from the 68000, hook it to the sensor controller, and type "OP<ret>" and "FT<ret>". This is done when the software prompts the user, usually when the contour tracing program is first run. Figure 5 shows the force sensor system connections.

Interconnection Between Devices

Most of the equipment is simply connected via RS-232 cables. The exception to this is a the parallel interface between the 68000 and the force controller.

Below table lists the connections made between the devices:

From	To	Cable
Download box (Host)	ECN VAX 11/780	Standard ECN cable
CRT	Download Box (CRT)	RS-232 cable (Tx-Tx,Rx-Rx)
68000 Serial Port 0	Download Box	RS-232 cable (Tx-Tx,Rx-Rx)
68000 Serial Port 1	Robot RS-232	RS-232 cable (Tx-Rx,Rx-Tx)
Force Serial Port	CRT	Temporary Connection (Tx-Rx,Rx-Tx)
68000 PIA Ports 0 & 1	Force Controller	Special Ribbon Cable
Force Controller	Preprocessor	Special preprocessor cable
Force Sensor	Preprocessor	Special force sensor cable
Robot Controller	Robot Arm	Special cable
Teaching Pendant	Robot Controller	Special pendant cable

TABLE 1: Interconnection Between Devices

2. Software For The Experiment

Before the actual software for the experiment is discussed, the software development system on UNIX will be described first. Most of the software was written in C-language and was compiled for the 68000 using a home grown compiler available on the Purdue's Engineering Computer Networks VAX 11/780 "ed" machine.

68000 SBC Contouring System Software

As stated above, most of the software written was in C-language. Some of the primitive I/O functions and the I/O initialization code was written in assembly language. There are a multitude of routines in the entire package, and most of them are not interesting enough to discuss in this paper. A tabular list of the important files is given below:

Important Files	
File Name	Purpose
Makefile	Contains compiling, assembly, linking and dependency info
head.h	Information common to many files
start.s	I/O initialization routines - must be first in link list
main.c	Contains main command routine
trace.c	This is the tracing algorithm
retrace.c	This is the retracing algorithm
funcs.c	Contains fixed point trigonometric functions
force.c	Contains force table controller I/O functions
check.c	Contains routines to check for robot clearances
kin.c	Contains fixed point kinematic routines
move.c	Contains robot controller I/O functions
printf.c	Redefines printf() to use given I/O routines
ioutils.c	String manipulation and fixed point I/O routines
b.out	Contains object code

Table 2: Contouring System Software

Robot Motion Program

All of the listed files form a program that allows a highly interactive environment for the robot user. A diagram showing the command structure for the program is given in Figure 6. The user can execute any of these commands by simply typing the first letter of the name and then "return". The commands will prompt for more data if necessary. If it is desirable to stop a command while it is executing, a "control-X" will usually abort the command and bring it to the outer level.

Below is a description of each command and how it functions.

The MOVE Command

The MOVE command allows the user to move the robot from its current position to any valid $(x, y, z)^t$ position. If the user enters an invalid position, the program will respond with an error message. There is no guarantee as to the path robot will take to that position since the controller has no trajectory generator.

The HERE Command

The HERE command allows the user to inspect the robot's joint angles and the Cartesian coordinates of the hand. The angles are displayed in degrees and the Cartesian coordinates are displayed in meters. Joint angular position displayed by this command is always valid but the Cartesian position is only valid if the wrist is pointing straight down. See the CORRECT command.

The CORRECT Command

This commands the wrist pitch to point straight down and the wrist roll to be parallel to the base frame y -axis. It does this by reading the robot joint angles, then setting $\Theta_4 = -(\Theta_2 + \Theta_3)$ and $\Theta_5 = -\Theta_1$ and moving the robot to the new position. If the robot cannot move there, an error message is printed.

The FORCE Command

The FORCE command reads the current sensor force and torque values and prints them out as raw data that is scaled as 10 kilo-ounces of force per unit. If the force controller had not been previously initialized, it prompts for this.

The SAFE Command

The SAFE command moves the robot in a generally safe manner to a safe position above the table.

The ZERO Command

The ZERO command first moves the robot to the safe position, then it moves the robot to its software defined zero position. The zero position can be seen in Figure 3.

The TRACE Command

The TRACE command runs the trace algorithm which is described in more detail in this report. It basically will trace around the outside of a object attached to the force table and record its contour in the base coordinate xy -plane.

The RETRACE Command

The RETRACE command will move the robot arm around a previously recorded object contour.

Fixed Point Arithmetic For Forward and Inverse Kinematics

As mentioned earlier, one of the more interesting aspects of the software was the fixed-point arithmetic used to do the kinematics. The need to use fixed-point came about because of the lack of reliable floating point software available to us. The increased speed of program execution was another reason for using fixed point arithmetic. In a highly optimal configuration it is possible that a fixed-point operation could perform faster than the equivalent floating point operation, if properly coded in assembly language. The routines used in this experiment were coded in C-language to reduce development time.

The basic philosophy of fixed point arithmetic is to scale every number as an integer and just manipulate the scaled integers. Fractions only exist in the eyes of the user because the formatted input/output routines know where to print the decimal point to visually achieve a scaling action. This generally works if the values have a small range of magnitudes. For example, distances for the experiment are specified in meters, but they are stored as tenths of millimeters. The input routine knows that the decimal point will come 4 digits before the end of the number. When printing out numbers, the output routine knows to print a decimal point before printing the last 4 digits. If an input number was smaller than a tenth of a millimeter, the input routine would truncate the number to zero. The effect of the input and output routines on an internal number X is as follows:

$$X = input * \text{SCALE}$$

$$output = X / \text{SCALE}$$

To retain precision in fixed-point, several precautions must be taken to keep the operations from truncating the numbers. Addition and subtraction are straightforward and can be done as expected, including the cascading of operations:

$$X = Y + Z$$

$$V = W - X$$

$$V = W - (Y + Z)$$

For multiplication and division, one must be more careful. As each number has been multiplied by SCALE, it is important to cancel SCALE values when making further multiplications, and to SCALE further when making divisions. For example, in floating point the following could be done:

$$X=Y*Z$$

$$V=W/X$$

$$V=W/(Y*Z)$$

But, in fixed-point arithmetic the following must be done:

$$X=(Y*Z)/\text{SCALE}$$

$$V=(W*\text{SCALE})/X$$

$$V=(W*\text{SCALE})/((Y*Z)/\text{SCALE})$$

The introduction of SCALE keeps the output value in range for the next operation. It should be noted that the multiplication and division operations must occur in double precision integer arithmetic since the value of SCALE is generally the maximum size of a single precision integer so as to retain as much accuracy as possible. It should also be noted that the order of these operations must be observed and that the factors of SCALE should not be canceled or combined or else significant truncation or overflow errors will occur. For example, if there was a fixed point routine to square a number X and the user input the value 0.1 to the routine, the formatted input function would assign the value of 1000 to X (assuming SCALE = 10000). The squaring equation would be as follows:

$$Y=(X*X)/\text{SCALE}$$

The value of X would be squared producing 1000000, this number would then be divided by 10000 leaving Y to equal 100. The formatted output routine would return a 0.01 because it functionally divides by SCALE during the output. Which is the correct answer.

Robot Kinematic Equations

The kinematic equations for the robot arm were developed without the use of Homogeneous transformations as developed in Paul's book [Paul 81]. They were developed using fundamental techniques in solid geometry . The application of contour tracking allows several simplifications to made about the robot configuration in which the tracing is carried out. The robot is always assumed to be in a shoulder-up configuration with the tool approach vector always pointing down, parallel to the z-axis, and the wrist roll was forced to be parallel to the y-axis. This forced a unique inverse kinematic solution to the five degree-of-freedom robot.

Robot Parametric Definitions

Figure 2 shows the robot arm and the definitions of the parameters as given in Table 3. The zero position of the joints of the robot is also shown in Figure 2. The length of the robot links were measured using a steel ruler and thus are subject to error. Likewise, the zero position of the robot arm was also inaccurately measured and probably is part of the reason that the robot cannot be commanded exactly where to go. The ratio of robot angular units to degrees was taken for granted from the operator's manual and is also subject to error. All of these problems lead to parametric errors that cause the inverse kinematics program to be in error also.

Robot Parameter Definitions	
θ_1	Joint 1 angle (Waist)
θ_2	Joint 2 angle (Shoulder)
θ_3	Joint 3 angle (Elbow)
θ_4	Joint 4 angle (Wrist Pitch)
θ_5	Joint 5 angle (Wrist Roll)
l_1	Height of Shoulder from Table
l_2	Length from Shoulder to Elbow
l_3	Length from Elbow to Wrist
l_4	Length from Wrist to Tool along normal to tool
l_5	Length from Wrist to Tool along tool axis
x	Linear distance in front of robot in Cartesian space
y	Linear distance to left side of robot in Cartesian space
z	Linear distance above table top in Cartesian space
R	The radial distance from the base to the tool in cylindrical space
H	Linear distance above table top to the tool in cylindrical space

Table 3: Robot Arm Parameters

The Forward Kinematic Equations

The discussion below outlines the development of the forward kinematic equations using the parameters described above. The robot position is first converted from the joint angles to cylindrical coordinates. Then the cylindrical coordinates are used to calculate Cartesian coordinates of the hand.

First define the sines and cosines as follows:

$$C_i = \cos\theta_i \tag{1}$$

$$S_i = \sin\theta_i \tag{2}$$

$$C_{ij} = \cos\theta_{ij} \tag{3}$$

$$S_{ij} = \sin\theta_{ij} \tag{4}$$

$$\theta_{ij} = \theta_i + \theta_j \tag{5}$$

The constraint of the wrist pitch pointing down parallel to the base from z-axis requires that:

$$\theta_{234} = 0 \tag{7}$$

Likewise, the wrist roll plane is always assumed to be parallel to the y-axis which requires:

$$\theta_{15} = 0 \tag{9}$$

Now the radial reach (R) of the robot base to the tool in the xy-plane is:

$$R = l_2 C_2 + l_3 C_{23} + l_4 \tag{11}$$

The height (H) from the table top to the tool tip is:

$$H = l_1 + l_2 S_2 + l_3 S_{23} - l_5 \tag{12}$$

The cylindrical coordinates (R, H, θ_1) of the end effector can be used to calculate its end-effector coordinates $(x, y, z)^t$.

$$x = RC_1 \tag{13}$$

$$y = RS_1 \tag{14}$$

$$z = H \tag{15}$$

The Inverse Kinematic Equations

The joint angles $(\theta_1 \ldots \theta_5)^t$ can be easily calculated for the above assumptions given tool position $(x,y,z)^t$ as follows: First the cylindrical coordinates are derived:

$$R=\sqrt{x^2+y^2} \tag{16}$$

$$H=z \tag{17}$$

$$\theta_1=atan2(y,x) \tag{18}$$

Next two intermediate variables are computed:

$$K_1=R-l_4 \tag{19}$$

$$K_2=H-l_1+l_5 \tag{20}$$

Then the elbow joint is solved for by first finding C_3:

$$C_3=\frac{K_1^2+K_2^2-l_2^2-l_3^2}{2l_2l_3} \tag{29}$$

If the value of $|C_3|>1$ then the robot cannot reach this point, as Θ_3 is solved as follows:

$$S_3=-\sqrt{1-C_3^2} \tag{30}$$

$$\theta_3=atan2(S_3,C_3) \tag{31}$$

If $S_3=0$ then the following is used to find Θ_2:

$$\theta_2=atan2(K_2,K_1) \tag{67}$$

If $S_3\neq 0$ then the following procedure is used to find Θ_2. First, find an intermediate variable:

$$K_3=\frac{l_2+l_3C_3}{l_3S_3} \tag{46}$$

Then compute θ_2 as:

$$\theta_2=atan2[(K_1-K_2K_3)\ ,\ (-K_2-K_1K_3)] \tag{60b}$$

The last two angles are very simply computed as:

$$\theta_4 = -(\theta_2 + \theta_3) \tag{68}$$

$$\theta_5 = -\theta_1 \tag{69}$$

3. The Contour Tracking Algorithm

This section describes the contour tracking algorithm in detail. The basic purpose of the algorithm is to guide the robot around an irregularly shaped object so as to determine its contour. Once the shape information is obtained, it can be stored for use at a later time. This could be useful for such tasks as removing flashing from a die cast part. A human could remove the flashing off of one part, have the robot learn the part's shape, then use the stored shape information to have the robot remove the flashing off any similar part at any time.

The algorithm is rather simple conceptually. In a sentence, the robot just moves at a right angle to the contact force. Problems arise in that the robot must somehow approach the object without knowing exactly where it is. It must not loose contact until it has completely traced around the object, and it must not apply too much force to the object. It must also determine when the tracing around the object is complete.

Additional problems arise as the robot controller used has only positional control. This means that the robot cannot servo to a force when it is in contact. Also, it is necessary to move the robot in small but appreciable distances to increase or decrease the force. This increases the risk of losing contact with the object, or applying too much force. The application of too much force can cause the robot actuators to saturate and the arm to jam and hence fail to reach its end position, and this causes the present controller to "lock up" and not respond to any more commands from the 68000. The algorithm developed addresses all these problems and a flow chart of this algorithm is shown in Figure 7.

The Contour Tracking Flowchart Description

As can be seen in Figure 7, the algorithm is divided into six major sections. These sections are described below:

Initialization

Before the robot can even begin to approach the object, it must initialize a variety of variables. First, it moves the arm to a safe position. It then initializes the force table and any existing bias forces sensed at the sensor.

Approach Phase

Once at the starting position, the robot then approaches the object by moving small incremental distances (currently 0.6mm) and then checks the force sensor. The robot continues in this fashion until a threshold force is exceeded at that point the to ensure contact. The *contact* force threshold ($F_{contact\_min}$) and its value is experimentally set. Currently it is at 5 ounces which was chosen because it was sufficiently high enough to keep force sensor noise from falsely causing a valid contact condition. The contact relation is shown below:

$$\text{if } \left(\left(F_{contact} = \sqrt{(F_x^2 + F_y^2)} \right) > F_{contact\_min} \right) \text{ then } contact\_established; \quad (70)$$

where F_x and F_y are the sensed contact forces along the x and y axis. Once contact has been made, the algorithm enters into the tracing phase.

The Contact Threshold Force

The robot must now be moved to a new position while maintaining contact. The contact force must be appropriately selected such that excessive force is not exerted which may result in saturation of the robot actuators. As this would cause the robot to jam as it is position controlled and will not accept a new set point until the position error is zero. Let us define this force as F_{jam_max}, then the contact force has to be maintained within the following range:

$$\frac{1}{2}\left(F_{jam} + F_{contact\_min} \right) < \sqrt{(F_x^2 + F_y^2)} < F_{jam} \quad (71)$$

where $F_{jam} < F_{jam\_max}$.

The desired contact force is experimentally found to be 10 ounces. This depends on the material of the tool tip and the workpiece. The jamming force F_{jam} is set at considerably less than the actually jamming force F_{jam_max}. As the robot must able to move from one contact point to the next contact point without jamming up (and requiring human intervention). It was experimentally determined for a given set of contours and contact materials this force to be 15 ounces.

If the contact threshold and the jamming threshold are too close together, the robot may not be able to position the force in between them due to its minimum travel distance. This can remedied by increasing the spread between the values, or by reducing the incremental robot moves to smaller distances.

This thresholding action is illustrated in Figure 8. As the position of the probe moves from $\mathbf{P}_{i-1}$ to $\mathbf{P}_i$ the robot tries to keep the contact force value between $\mathbf{F}_{min}$ and $\mathbf{F}_{max}$ before it tries to move at a right angle to the force.

Record Position

When the robot has completed an incremental move while keeping in contact with the object its new probe position is recorded. This position must be compensated for as the robot and its tool flexibility alters actual location of the object contour, as sensed at the joint position sensor.

Conditions for Terminating the Contour Tracking Motion

The starting position on the contour is defined as the first point at which contact was made. The robot is continually moved to the right while maintaining a contact. A stopping region is defined as being the circle with the center as the starting point. When the tool enters this circle the contour tracking operation is terminated. This is shown in Figure 9.

Moving Around the Contour

The probe must be moved from position $\mathbf{P}_{i-1}$ to $\mathbf{P}_i$. In this experiment the current force information is used to compute the incremental move to point $\mathbf{P}_i$. Given the radial step size for the movement is Δr, then

$$dP_{x,i} = \Delta r \frac{F_x}{\sqrt{(F_x^2 + F_y^2)}} \tag{72}$$

$$dP_{y,i} = -\Delta r \frac{F_y}{\sqrt{(F_x^2 + F_y^2)}} \tag{73}$$

where F_x and F_y are forces monitored from the current contact. Then,

$$\mathbf{P}_i = \mathbf{P}_{i-1} + d\mathbf{P}_i \tag{74}$$

where

$$d\mathbf{P}_i = (dP_{x,i}, dP_{y,i})^t \tag{75}$$

Once the move is completed the contact force is checked, if the contact force is below the minimum contact force a new move $d\mathbf{P}_i'$ is computed. The new incremental move bisects the angle between the initial direction of the incremental movement and the direction of the maximum force. This is shown in Figure 10. If the move $d\mathbf{P}_i'$ does not ensure a minimum contact force, the angle between $d\mathbf{P}_i'$ and the direction of the maximum force is further halved until contact is made.

Effects of Probe and Robot Compliance

As mentioned earlier the robot used in this experiment is a position controlled device as a result compliance is essential for fine force resolution. If K_x is the cartesian stiffness of the end-effector and $dx_{\min}$ is the minimum cartesian movement, then,

$$F_{\min} = K_x dx_{\min} \tag{76}$$

Additional compliance can be added by a compliant probe (K_{probe}) as:

$$K = \frac{K_x K_{probe}}{K_x + K_{probe}} \cong K_{probe} \quad \text{if } K_x < K_{probe} \tag{77}$$

where K is the altered stiffness of the robot and the tool as seen at the tip of the probe. Figure 11, shows the effects on positional error with a compliant probe and a stiff probe.

4. Results of the Experiment and Its Education Value

The algorithm that was developed to trace the path of the object on the force table worked quite well. The probe maintained contact with the object and never became lost. The process of tracing the entire object was fairly slow due to the nature of the robot communication as it took long for the robot to execute the move commands. When the object was retraced with previously calculated and stored joint positions, the process of tracing the perimeter of the object was cut to roughly one half of the original time.

Plots of Object Outline

Four different objects were traced and the of their outlines was plotted. Figure 12 shows three of the objects. Figures 13, 14 and 15 show the outlines of the objects from the stored data. These outlines are a by-product of the algorithm, since the algorithm's purpose is to teach the robot how to trace around an object. Although the Cartesian points plotted may have errors in them, the angular positions stored are still correct because they truely represent the position of the contact with the object, at that specified force.

Analysis Of Results

There are three kind of errors most notable in the outlines of the plotted objects. The first error is the shrinking of the object, primarily in the x dimension. The shrinking of the dimensions of an object is generally caused by the compliance of the probe and the robot arm joints. It may also be caused by backlash in the gears,

creating the effect of the robot having to move the joint further to achieve the same pressure. However, the dimensional distortion due to backlash would be small. The reason the object shrinks more severly in the x dimension is because this axis is aligned with the radial dimension of the arm. There are three joint along the arm in this direction, causing much more compliance in x than along the y-axis.

The second type of error is a broad edge distortion along the sides. This type of error is contributed mostly to parametric errors in the robot. The length information of the arm is approximate, as is the joint angle zero positions and as is the arm gear ratios. Along with the eccentricity of the gears this will lead to nonlinear distortion in the radial direction of the robot. This will occur more noticeably in the x direction, once again because it is in the radial direction of the arm, the plane in which most of the robot joints operate.

The third type of error is jagged lines along all of the object edges. There are two major considerations for this error. The first and more noticeable was due to the thresholding of the force. One point may have had the maximum pressure against it when its neighbor may have had the minimum force. Since there is compliance, this force difference may cause small random fluctuations in distance. Another source for this error is due to the resolution of the plots themselves. Many jagged edges appear that way because the plot has moved over one pixel distance.

Educational Achievements of This Experiment

Objective of this experiment was to introduce graduate students to real-time robot control aspects, these included: kinematics, robot dynamics, drive train mechanisms, interfacing of sensors, real-time system programming and force control.

This was a very ambitious objective, most our students accomplished this, as they already had taken an introductory course in robotics (taught out of *"Robot Manipulators"* by R. P. Paul [Paul 81]).

In the first two weeks of this course student were familiarized with the 68000 microprocessor assembler programming and hardware operations. Integer arithmetic abilities of this device was also discussed. This enabled the students to begin the software development for the inverse and forward kinematics and the interfaces.

One of the prerequisites for this class was the knowledge of the C-language. Therefore, the students were encouraged to develop most of their software in C-language to reduce debugging time. Most of the input/output driver were written in assembler as discussed in the earlier part of this paper.

Once the programs were running, student immediately became aware of number problems.

(i) *Inherent inaccuracy of the manipulator* due to drive mechanism nonlinearities and imprecise Denavit-Hartenberg parameters (see [Paul 81] [Ahmad 87] [Wu 84] [Hayati 84]) assumed in the inverse kinematics program. They notice this problem as they are unable to command the manipulator to go to an exact cartesian position.

(ii) *Dynamical Effects of the Arm Motion.* The NNK robot is a step input independent joint controlled device. The effects of the nonlinear dynamics are apparent as the effective time constants of each joint change. Joint angles corresponding to each cartesian position is sent to the NNK manipulator. The NNK controller treats this as a step input to each joint. The joints try to reach the terminal position independent of each other, and each joint reach the terminal joint angle at a different time. This causes the end effector path to deviate from a straight line trajectory. At a different end point, each joint exhibits a different step response. This illustrates that the loading an each joint changes with each trajectory, and therefore the dynamic characteristics of the arm change with the arm configurations. From an earlier class, Introduction to Robotics EE569, they derived the dynamics of a two degree of freedom robot. From this experiment they can relate to the dynamic effects which deterriorate the transient response of the arm. Such as the joint time constants which is proportional to the effective joint inertia.

(iii) *Force Control and Manipulator Compliance*
This experiment did not concentrate on force control. However, it did require the implementation of a kinematic force control algorithm e.g. making contact with an object with a specified force. Student were initially asked to perform the tracing with a hard probe, this resulted in instability or chattering. This is a problem in force control which is being currently addressed [Kazerooni et al. 86] [Whitney 85]). This problem arises because the manipulator is being position controlled as opposed to being torque controlled. Chattering occurs when the desired force resolution is smaller than the product of the effective robot and tool stiffness and the cartesian motion resolution.

5. Conclusion

In this paper we described an experiment which is performed by our seniors and graduate student for a class in Real-time Robotics. This is an ambitions experiment designed to introduce the students to many aspects of current day robotics research. Students were able to complete this experiment in one semester as they had previously taken an introductory class in robotics. The complete description of the experiment and its results have been addressed in this paper. We found this experiment

challenged graduate students and helped them progress in their research areas.

Acknowledgements

Jim Gallo and Correy Ustanik were the first students to perform this experiment, the results presented in this paper are reprinted from their laboratory reports.

References

[Ahmad 87]
Ahmad Shaheen, "Analysis of Robot Drive Train Errors Their Static Effects and Their Compensations," TR-EE-87-4, Purdue University, West Lafayette, Indiana 47907.

[Hayati 83]
Hyati, S., "Robot Form Geometric Link Parameter Estimation," Proceedings of the 22nd IEEE Conference on Decision and Control, December 1983.

[Kazerooni et al. 86]
Kazerooni, H., Sheridan, T. B., Houpt, P. K., "Robot Compliant Motion for Manipulators," Part I: "The Fundamental Concepts of Compliant Motion," Part II: "Design Method," IEEE Journal of Robotics and Automation, June 1986.

[LORD 85]
LORD Corporation, "F/T Sensor Installation and Operations Manual," LORD Corporation, Cary, NC.

[Paul 81]
R. P. Paul, "Robot Manipulator," MIT Press, 1981.

[Whitney 85]
Whitney, D. E., "Historical Perspective and State of the Art in Robot Force Control," Proceedings of the 1985 IEEE Conference on Robotics and Automation, March 1985, St. Louis, MO.

[Wu 83]
Wu, C. H., "The Kinematic Error Model for the Design of Robot Manipulators," Proceedings of the 1983 Amercian Controls Conference, San Francisco, June 1983.

*EE*569: Introduction To Robotics (Seniors and Graduate Students)

*EE*595*A*:Real-Time Robot Systems and Control Laboratory (Seniors and Graduate Students)

*EE*686: Variable Topics in Control (Graduate Students)

*EE*695*D*:Robotic Controls (Graduate Students)

Figure 1: Sequence in which Robotics Classes may be taken at the School of Electrical Engineering

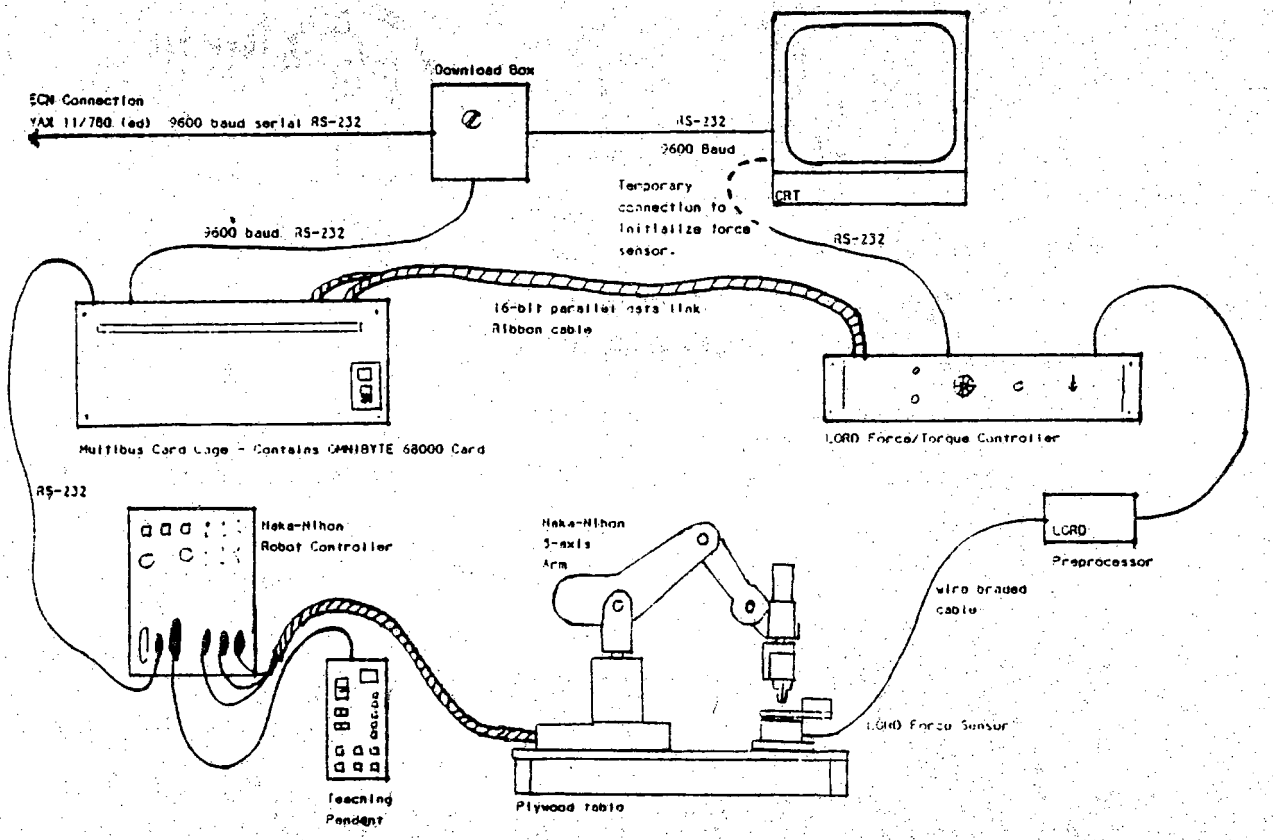


Figure 2:
Equipment Setup

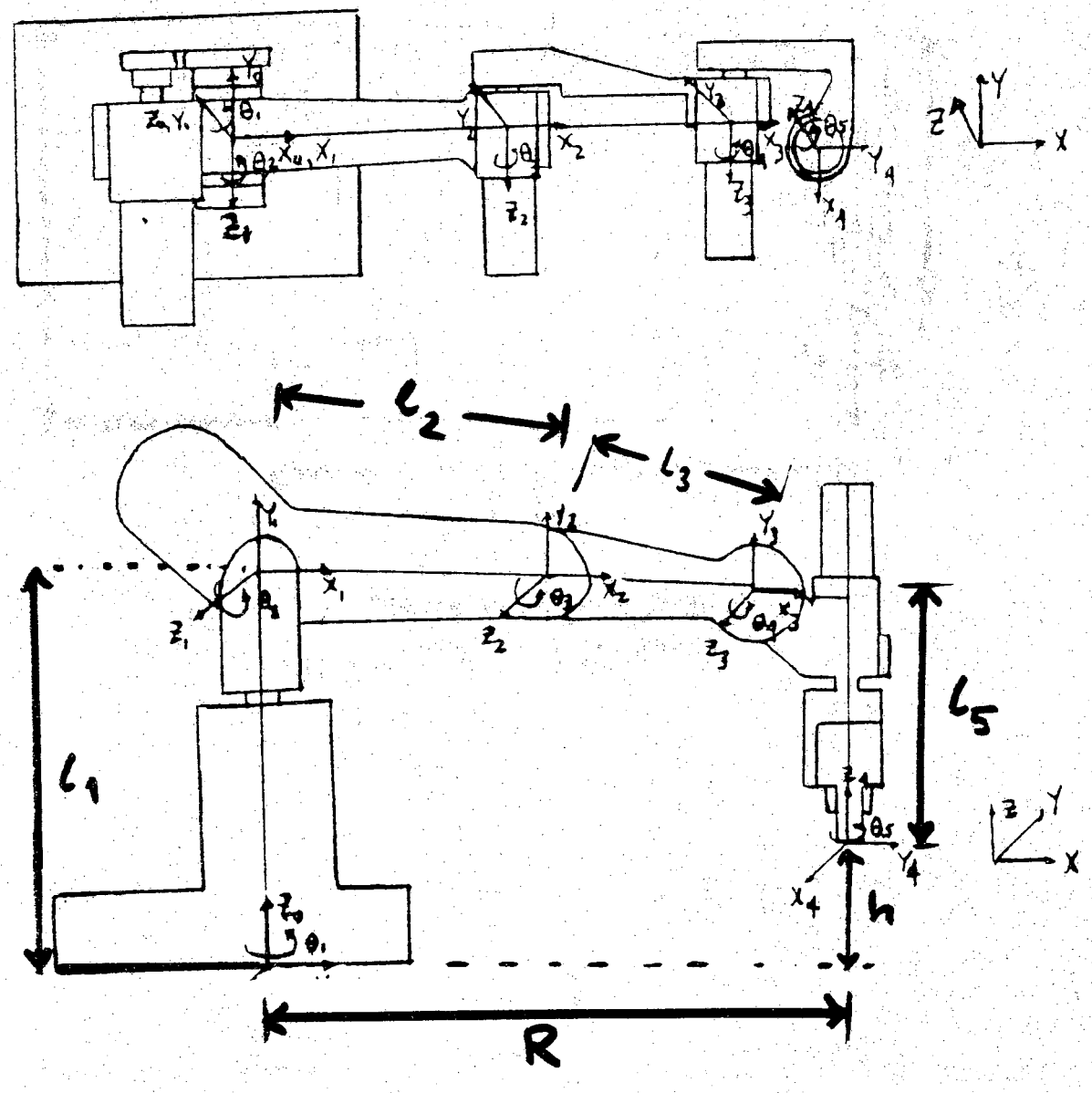


Figure 3: Software Defined
Robot Arm Zero Position.

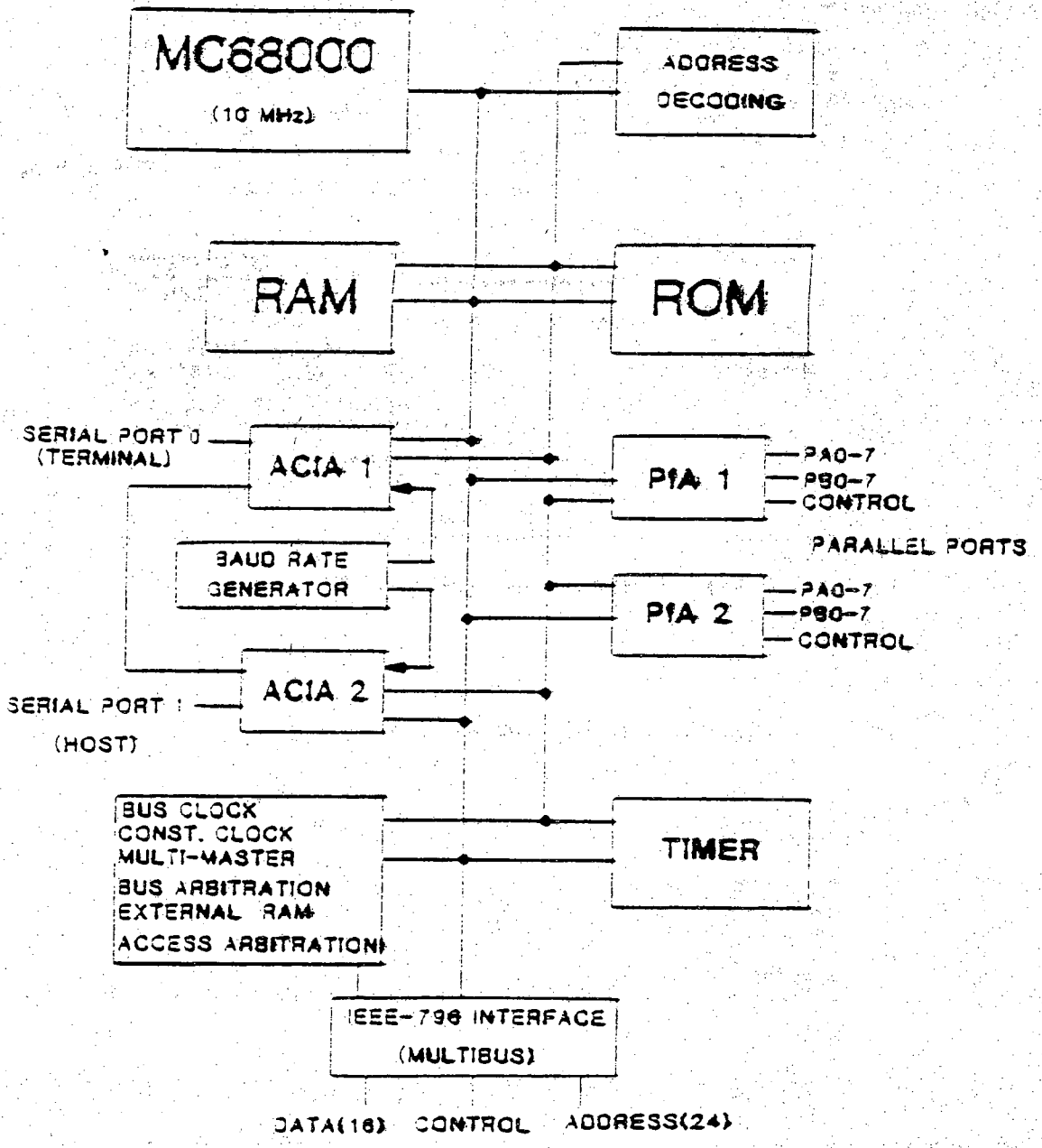


Figure 4
Omnibyte 68000 SBC Block Diagram

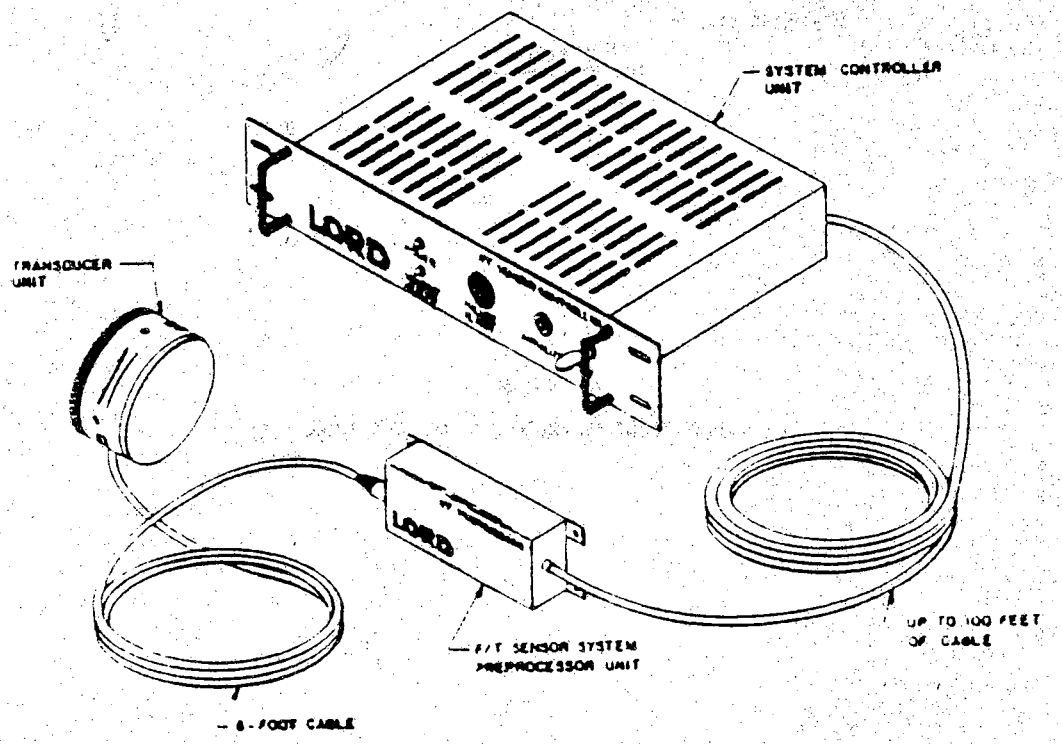


Figure 5
Lord Force/Torque Sensor System

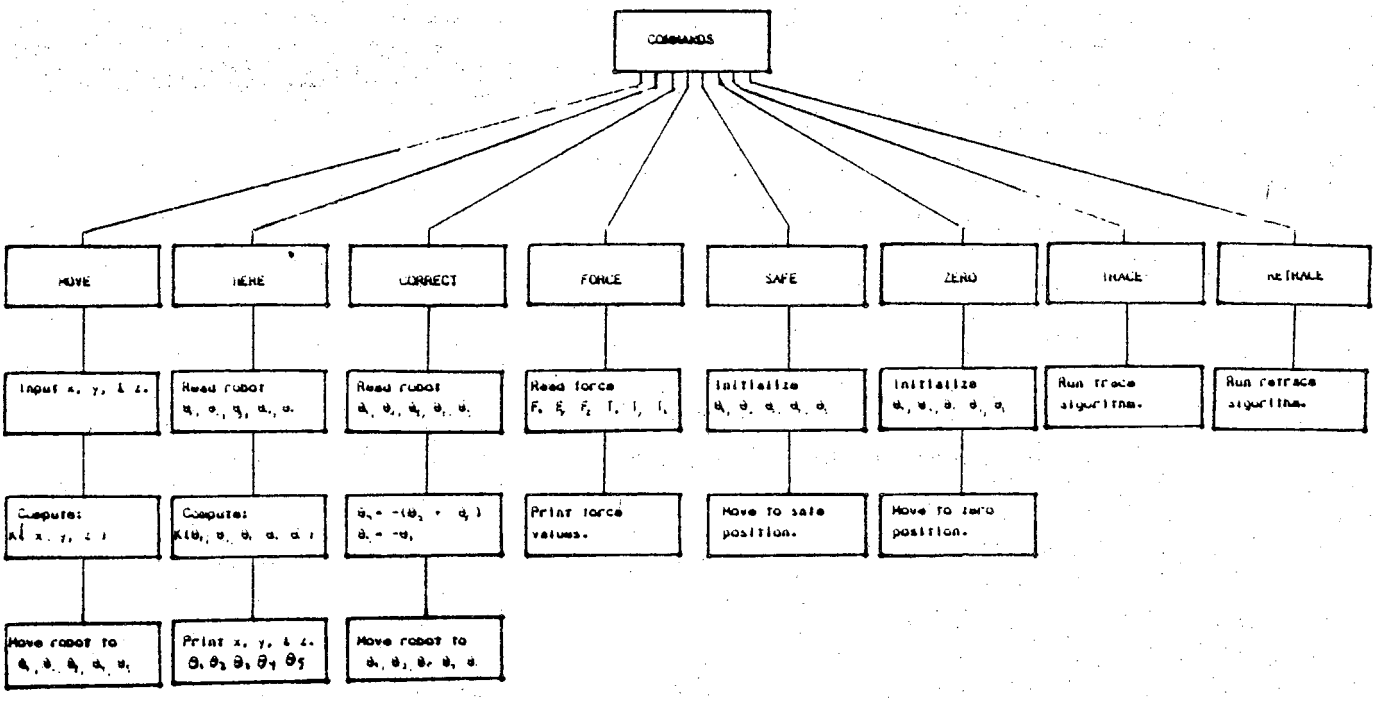


Figure 6
Robot Contouring System Command Structure

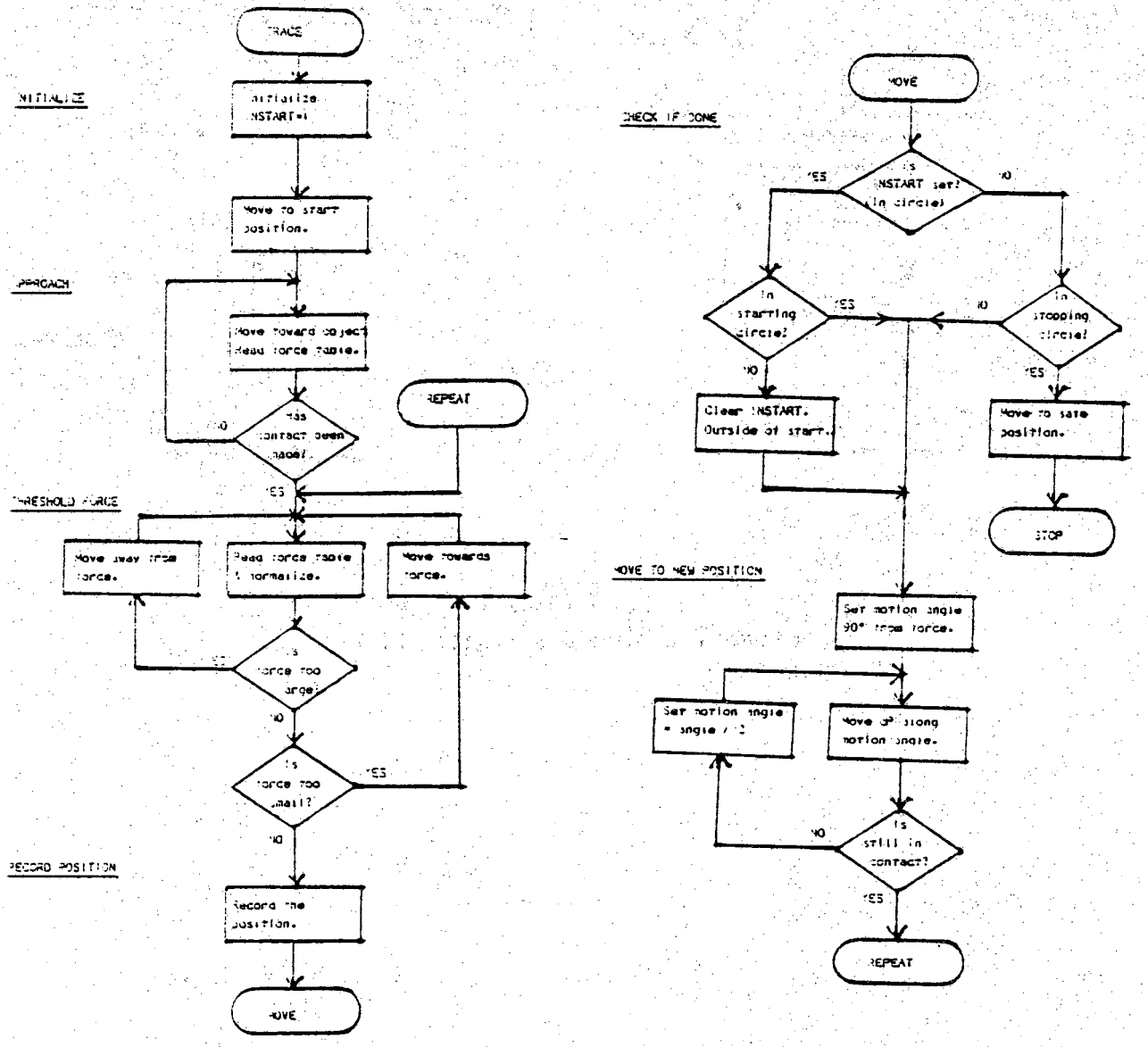


Figure 7
The Contour Tracking Algorithm Flowchart

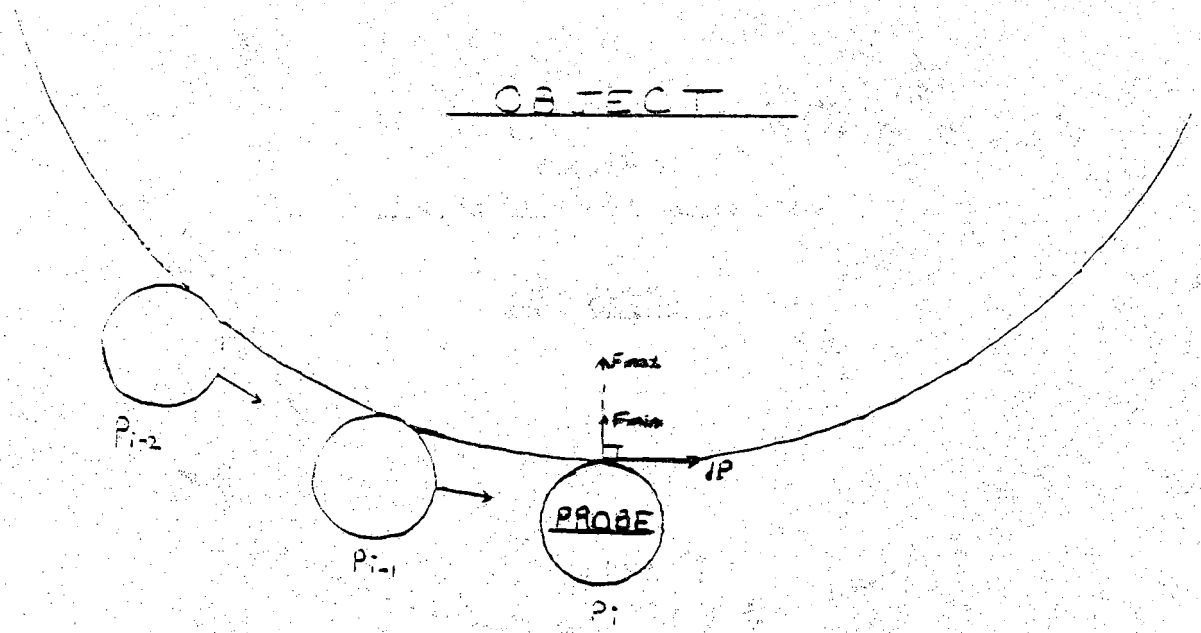


Figure 8
Thresholding the Contact Force

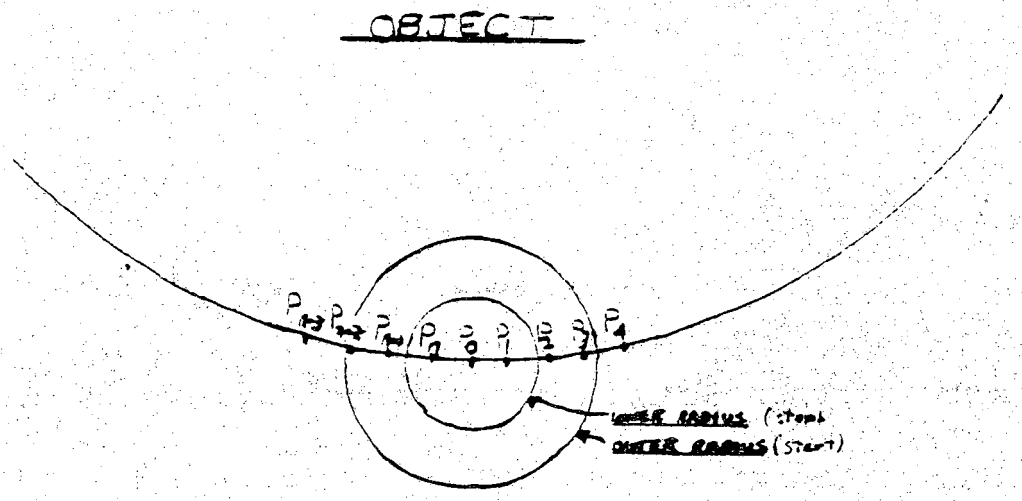


Figure 9
Starting and Stopping Regions

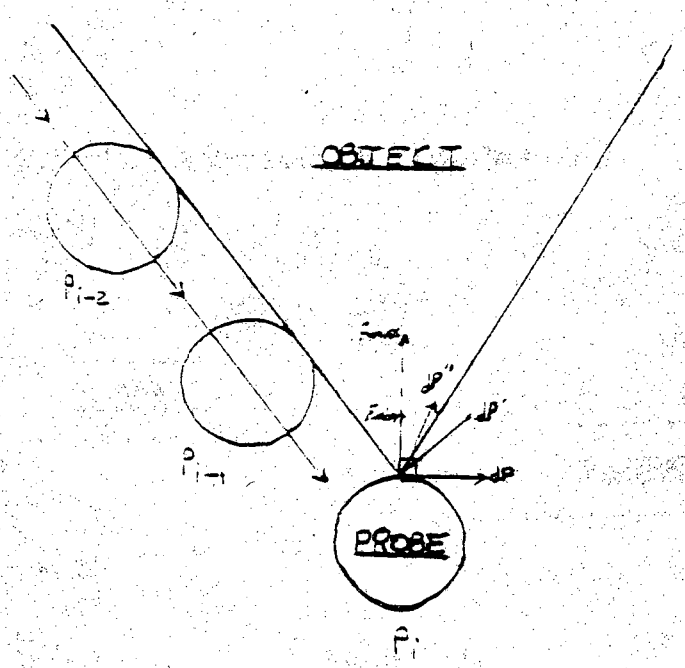


Figure 10
Movement to New Position

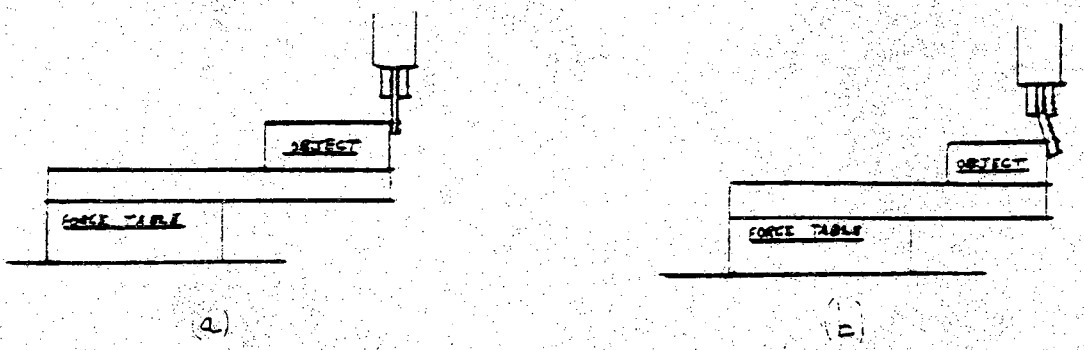


Figure 11
Positional Error With A Compliant Probe

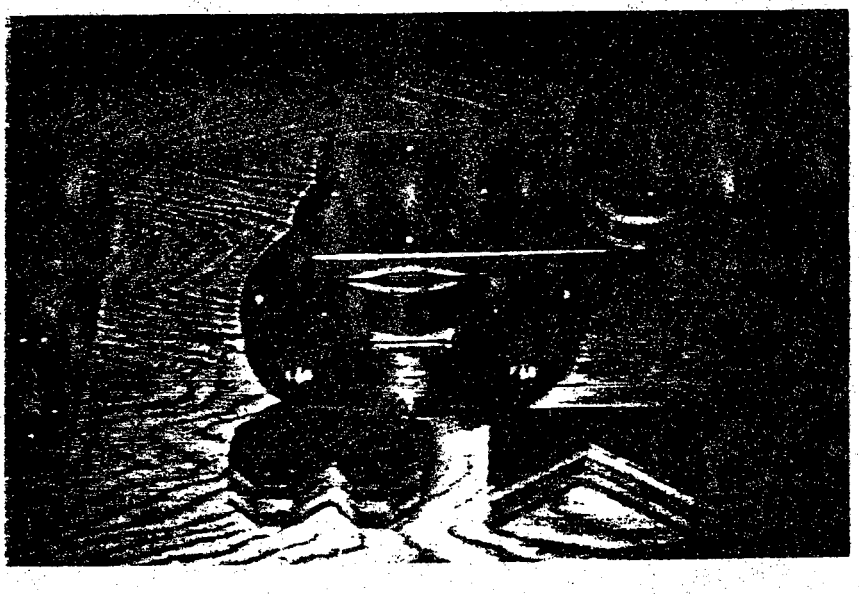


Figure 12
Photograph of the Objects

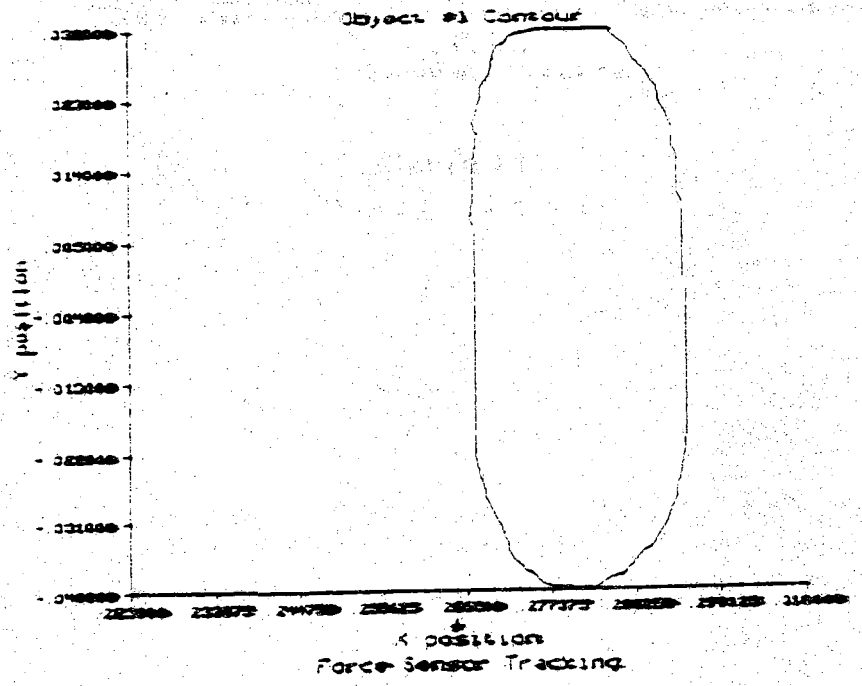


Figure 13
Object #1 Contour

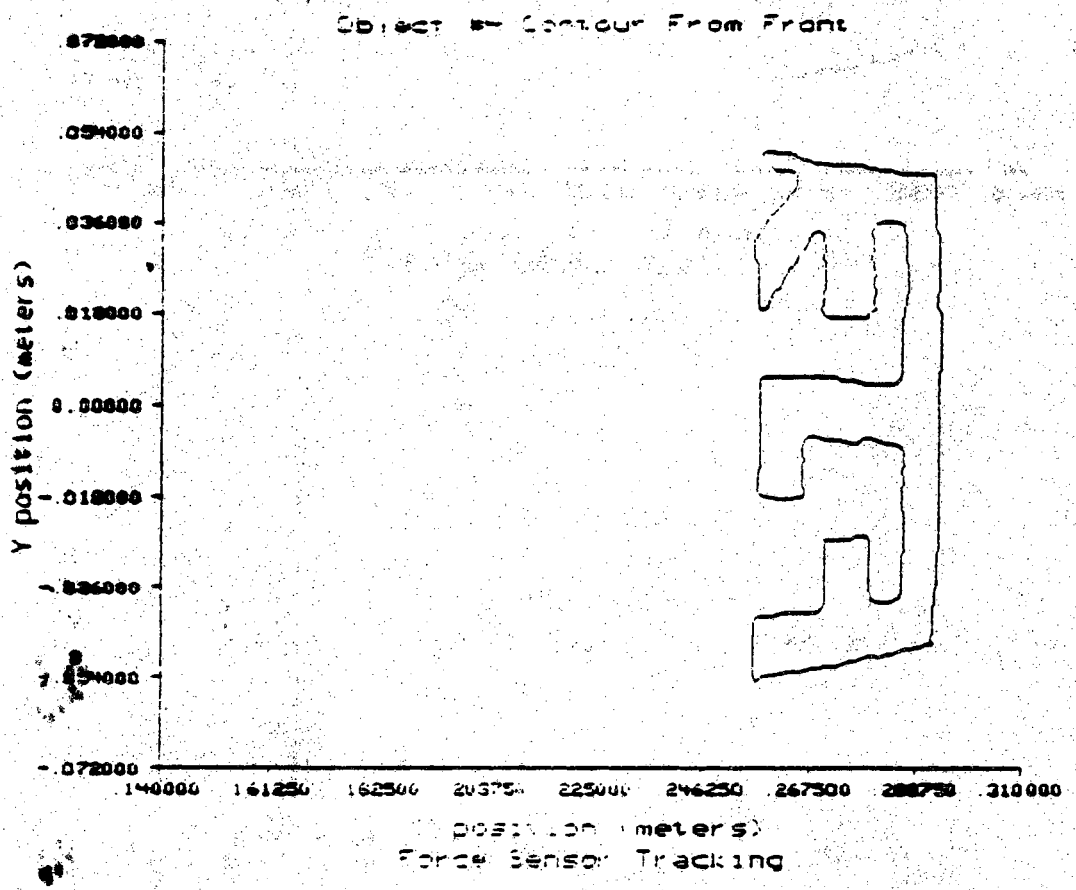


Figure 14
Object #2 Contour (Traced From Front & Back)

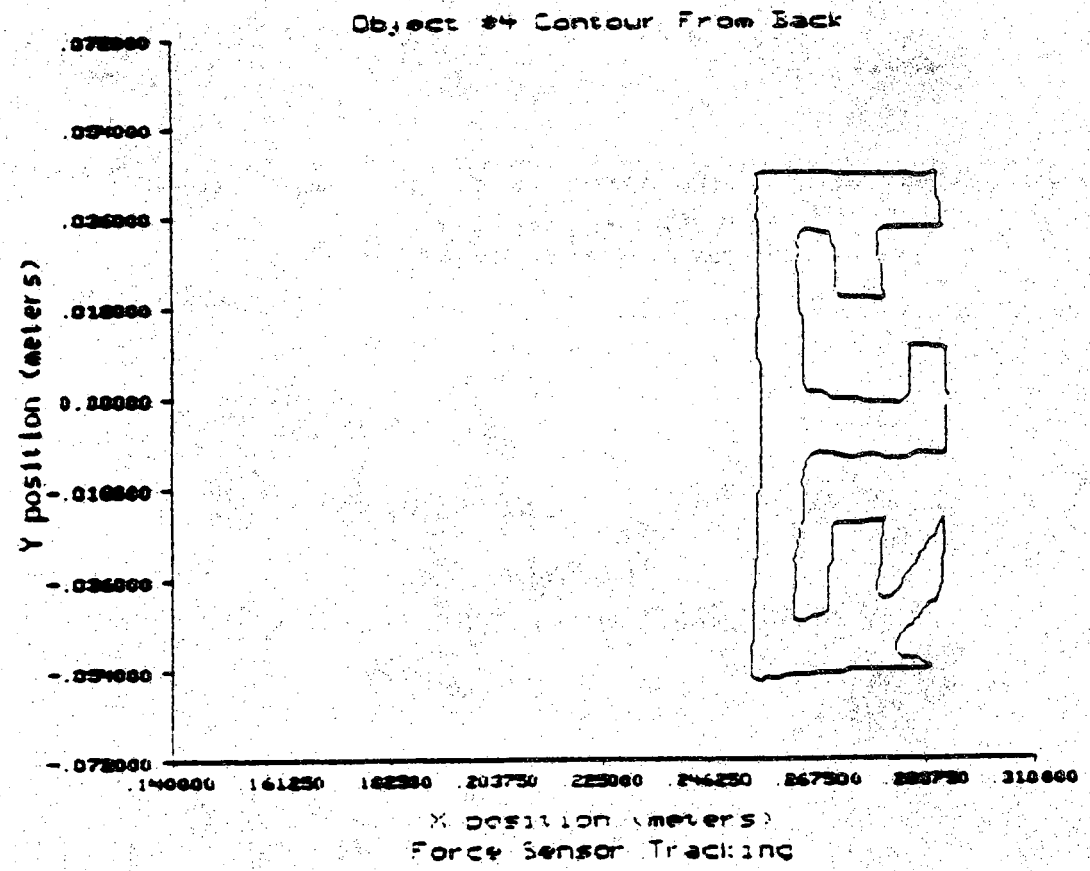


Figure 15
Object #3 Contour (Traced From Front & Back)

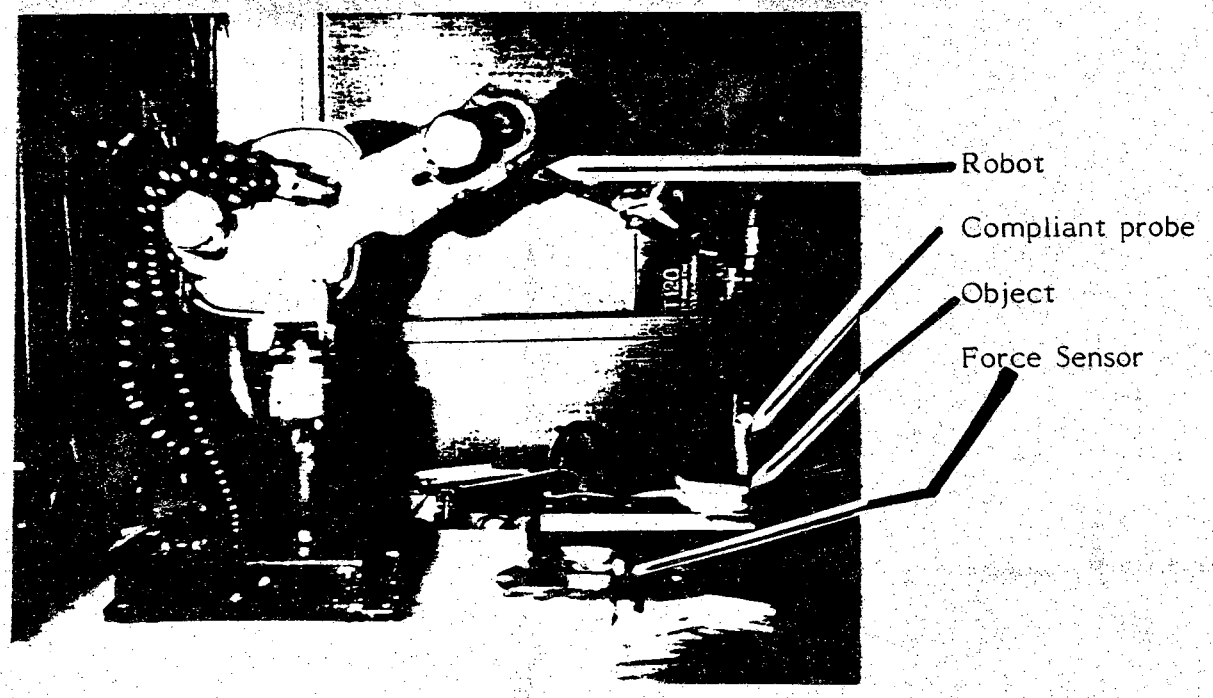


Figure 16
Photograph of Experiment Workstation