Title
Human control strategy in operating a telerobot

Author
Takeda, Munehisa,

Publication Date
1988

Peer reviewed|Thesis/dissertation

Human Control Strategy in Operating a Telerobot

by

Munehisa Takeda

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

BIOENGINEERING

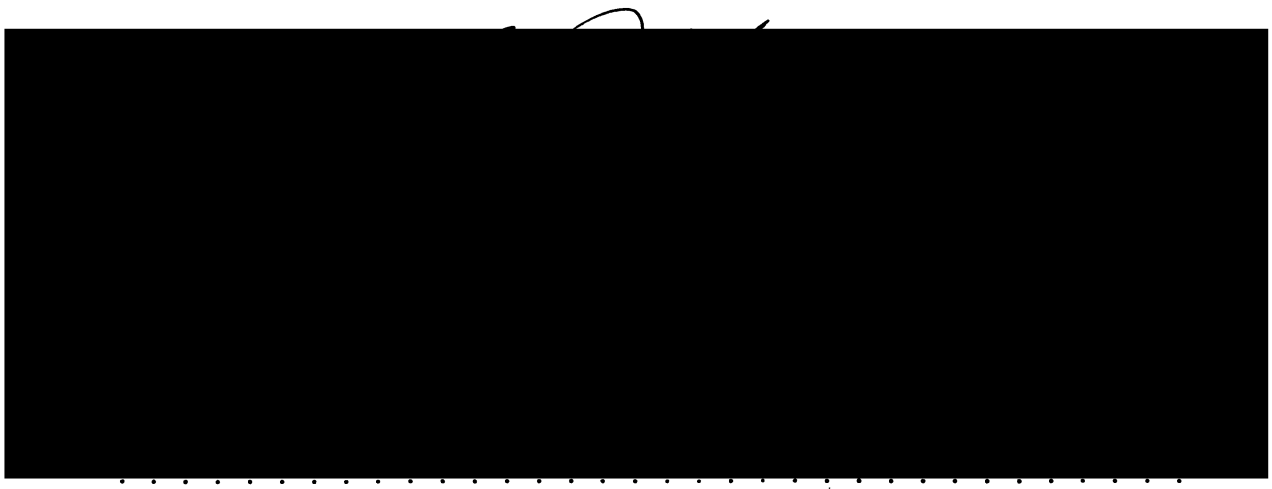
in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA

San Francisco



Date **University Librarian**

Degree Conferred: MAR 27 1988

Acknowledgements

My gratitude and appreciation first goes to Prof. Lawrence Stark for providing me with the opportunity to perform this research. Under his tutelage, I have realized how little I know. His broad knowledge, consideration, enthusiasm, energy, and humor have strongly influenced my developing career. I wish to thank Prof. Steven Lehman for his good advice to decide my curriculum, good lecture, and suggestion for this thesis. Without his guidance, I could not have finished this thesis. I also wish to thank Prof. David M. Auslander for his useful lecture and suggestion for this thesis. His ME 230 course was very beneficial for me to finish up this thesis.

I also thank those who have been in the lab; especially, Dr. Wonsoo Kim, Dr. Katsuya Matsunaga, Dr. Masaru Miyao, and Mr. Frank Tendick. Dr. Kim taught me a lot of things: research attitude, research approach, way of thinking, life, English, etc. Without his help, I could not have finished this thesis. Dr. Matsunaga's suggestion was very beneficial for designing experiments. Dr. Miyao's advice about my research and health encourage me to keep this research. Frank read through this thesis and helped me to revise this thesis.

I also owe a tremendous debt to Mitsubishi Electric Corporation. The company have supported me to study at the University of California, San Francisco and Berkeley. Especially, I wish to thank Dr. Akemi Futakawa, Mr. Kouji Namura, Mr. Takahiro Masuda, and Mr. Kurita for providing me with this wonderful opportunity and supporting me during studying at UCSF/UCB.

Finally, I am very grateful to my parents, sister, and my wife, Kumiko, for their support with deep care and great love.

Human Control Strategy in Operating a Telerobot

Munehisa Takeda

Abstract

Efficient human-robot interaction is necessary in telerobotics because a fully automated robotic system is not yet feasible for performing general purpose telerobotic tasks that require flexibility. The study of human control strategy is important in order to achieve more efficient human-robot interactions.

In this thesis, human coordination and motion control strategies in operating a telerobot were investigated using placement tasks with a joystick-controlled robot. Preliminary experimental results show that human motion control demonstrates two strategies, coarse and fine motion control. Coarse motion is fast, large, and rough motion from the initial position to the vicinity of the target position. Fine motion is slow, small, and accurate motion from the vicinity of the target position to the actual target position. Two experiments are performed to examine these human control strategies. The first experiment compares end effector control (E.E.C.) and joint angle control (J.A.C.) in coarse motion. The second experiment compares E.E.C. and J.A.C. in fine motion. A placement task was chosen as a typical example of telerobotic tasks. For simplicity, the robot motion is confined to a vertical 2-D plane in these experiments. The experimental results indicate that for coarse motion control no significant difference in success rate or completion time was observed between end effector control(E.E.C.) and joint angle control (J.A.C.), whereas for fine motion control E.E.C. was significantly better than J.A.C.

Table of Contents

List of Tables

List of Figures

1. Introduction

Human-robot interaction is becoming an interesting field in telerobotics. Since a fully automated robotic system is not yet feasible for performing general purpose telerobotic tasks that require flexibility, both humans and robots are used in situations where complicated and hazardous conditions exist or the safety of the humans is threatened.[1] The study of human control strategy is thus important in order to achieve more efficient human-robot interactions. Apparently human operators use many different strategies in controlling telerobots. This thesis, in particular, addresses two human control strategies: coordination and motion control strategy.

In a coordination strategy study, Whitney[2] mentioned that end effector control strategy (E.E.C.) would be much easier in controlling a prosthetic arm than joint angle control strategy (J.A.C.), and proposed resolved motion rate control. Recently Hollerbach and Atkeson[3] examined the simplest and most commonly proposed coordination strategy, simple linear interpolation, for both end point variables and joint variables and tried to find the best match of predicted trajectories to experimental human arm movements. They found that this group of planning strategies was inadequate and that a generalization of the coordination strategy to "staggered interpolation" for joint variables best described existing data.[4] This means that humans may use a modified joint angle control strategy in moving their arm. What kinds of strategy does the human use in operating a telerobot? One purpose of this research is to try to answer this question by experiments.

Regarding motion control strategy, Fu, Gonzalez and Lee[5] mentioned in their book that the movement of a robot arm could be performed in two distinct control phases: gross motion control and fine motion control. Brady[6] also mentioned that two modes exist for pick-and-place tasks: fast free motion and slow guarded motion. By observing telerobot motion in a placement task, we can easily see these two modes. We call these two modes coarse motion and fine motion. Another purpose of this research is to figure out the characteristics of these two motion control modes and find the human coordination strategies underlying these motion strategies.

This thesis first describes characteristics of each strategy. Then two experiments are presented. The first experiment compares end effector control (E.E.C.) and joint angle control (J.A.C.) in coarse motion. The second experiment compares E.E.C. and J.A.C. in fine motion. A placement task was chosen as a typical example of telerobotic tasks. For simplicity, the robot motion is confined to a vertical 2-D plane in these experiments. Finally, the control strategies for each motion are discussed.

2. Human control strategy in telemanipulation tasks

2.1 Coordination strategy :

End effector control(E.E.C.) vs. Joint angle control(J.A.C.)[2],[5],[6],[7]

End effector control (E.E.C.) and joint angle control (J.A.C.) are two major human coordination strategies in operating a telerobot. E.E.C. is the strategy which directly controls the end effector (hand) position and orientation in Cartesian coordinates, while J.A.C. is the strategy which controls each joint angle of the telerobot directly.

In E.E.C. it is easy to visualize the correct end effector configurations in Cartesian coordinates and to separate position variables from orientation variables. E.E.C., however, requires an inverse kinematics calculation in order to calculate the joint actuation values. This causes degeneracy problems of the manipulator, multiple solutions of the joint angles, and additional computation time. Furthermore, it is difficult to predict physical constraints such as the torque, force, velocity, and acceleration limits of each joint motor, implying that a straight line movement based on E.E.C. may not be time or energy optimal.

J.A.C. does not require inverse kinematics calculations for the robot . Thus there are no degeneracy problems, and one can obtain unique solutions of joint angles and faster calculations for the robot system. This strategy is very efficient, being limited only by joint accelerations and maximum joint velocities. But the strategy imposes the inverse kinematics calculation on the human operator. Another disadvantage is difficulty in determining locations of the various links and the hand during motion. Two reasons for this difficulty in J.A.C. are that its coordinates are not orthogonal and it does not separate position variables from orientation variables. Furthermore, it is usually required to guarantee obstacle avoidance along trajectory.

Early teleoperator systems found it difficult to achieve Cartesian space straight line motions, since the switches manipulated by the operator controlled individual joint angles. Whitney's resolved motion rate control first demonstrated how to achieve Cartesian space

straight line paths. He focused on the advantages of E.E.C. and disadvantages of J.A.C., but this does not mean that E.E.C. is always more suitable for teleoperation than J.A.C.

J.A.C. also has advantages and E.E.C. has disadvantages. As Hollerbach and Atkeson's study of human arm movement showed, humans may use the J.A.C. strategy in moving their arms. Are there some tasks or modes which are suitable for J.A.C. in operating a telerobot? This research gives some answers to this question by experiments.

2.2 Motion control strategy : Coarse motion control vs. fine motion control[5],[6]

Fu, Gonzalez, and Lee mentioned in their book that the movement of a robot arm is usually performed in two distinct control phases : gross motion control and fine motion control. According to them, the arm moves from an initial location (position/orientation) to the vicinity of the desired target location along a planned trajectory in gross motion. The end effector of the arm dynamically interacts with the object using sensory feedback information from sensors to complete the task in fine motion control. Brady also mentioned that two motions exist : slow guarded motion and fast free motion.

In order to examine these two motions, placement task trajectories were investigated. The placement task trajectories of E.E.C. and J.A.C. with initial and final robot postures are shown in Fig. 2.1 and Fig. 2.2, respectively. In these figures, the trajectories were plotted every sampling time. Therefore, the beginning coarsely spaced dotted curve means that the hand of the robot moved fast, and later densely spaced dotted curve, witch appears to be almost a solid curve, means that the hand of the robot moved slowly. These figures indicate that the trajectories of the placement task can be divided into two different motions in both E.E.C. and J.A.C. We call these two motions coarse motion and fine motion. Coarse motion is fast, large, and rough motion from the initial position to the vicinity of the target position. Fine motion is slow, small, and accurate motion from the vicinity of the target position to the actual target position.

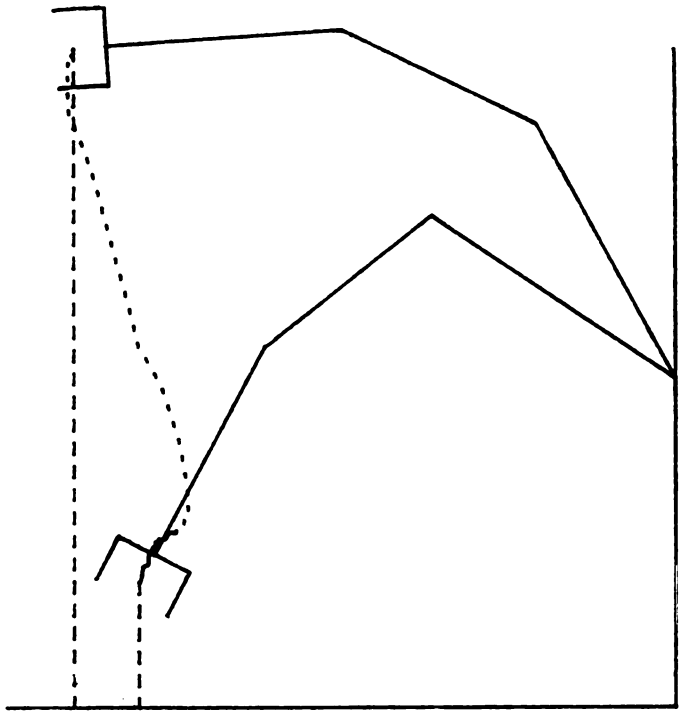


Fig. 2.1 Placement task trajectory for E.E.C.

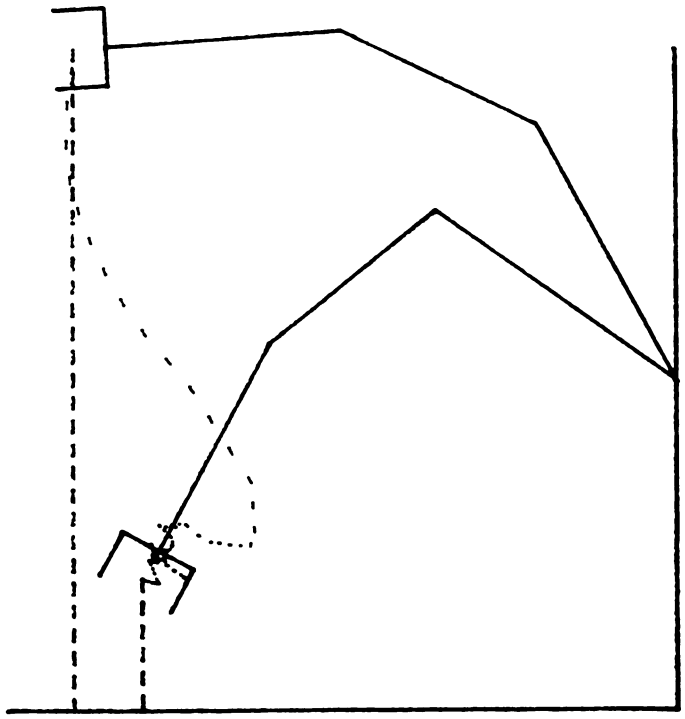


Fig. 2.2 Placement task trajectory for J.A.C.

3. Experimental system

3.1 Hardware

The experimental system consists of a robot system (Mitsubishi Electric Corporation Movemaster II), 2 joysticks (Measurement Systems Model 521z), a computer system (IBM PC-AT), objects and target boxes (Fig. 3.1). The robot is connected to the computer through a Centronics parallel printer port. Two joysticks, which are used as the robot control device for both E.E.C. and J.A.C, are connected to the computer through an 12-bit A/D converter (Metrabyte, DASH 16). The computer system includes a peripheral printer and a plotter.

3.1.1 Robot system

The robot system consists of a robot (RM-501), a drive unit, and a teaching box. The robot has 5 degrees of freedom, but only 3 degrees of freedom (shoulder, elbow, and wrist bend) are used in the experiments. The drive unit controls the 5 joint d.c. motors and hand gripper open/close. Inner position and velocity feedback loops for each d.c. motor of the robot are imbedded in this unit. Two input modes were provided: teaching box control mode and computer control mode. The teaching box is connected to the unit. An operator can control each joint angle and send several commands to the robot through the teaching box. This teaching box was used for initialization or emergency recovery in this experiment. Normally, computer control mode was used in this experiment. The computer sends robot commands (Table 3.1) through the Centronics printer port. The unit receives and interprets the commands and sends the motor actuation output to the robot. The specification of the robot system is shown in Table 3.2. The moving range without hand is shown in Fig. 3.2. It has been found that the workspace of the robot hand is severely limited when the desired orientation of the robot hand is specified. The workspace limitation for the specific pitch angle (0,-30,-60,-90 degrees) is illustrated in Fig. 3.3.

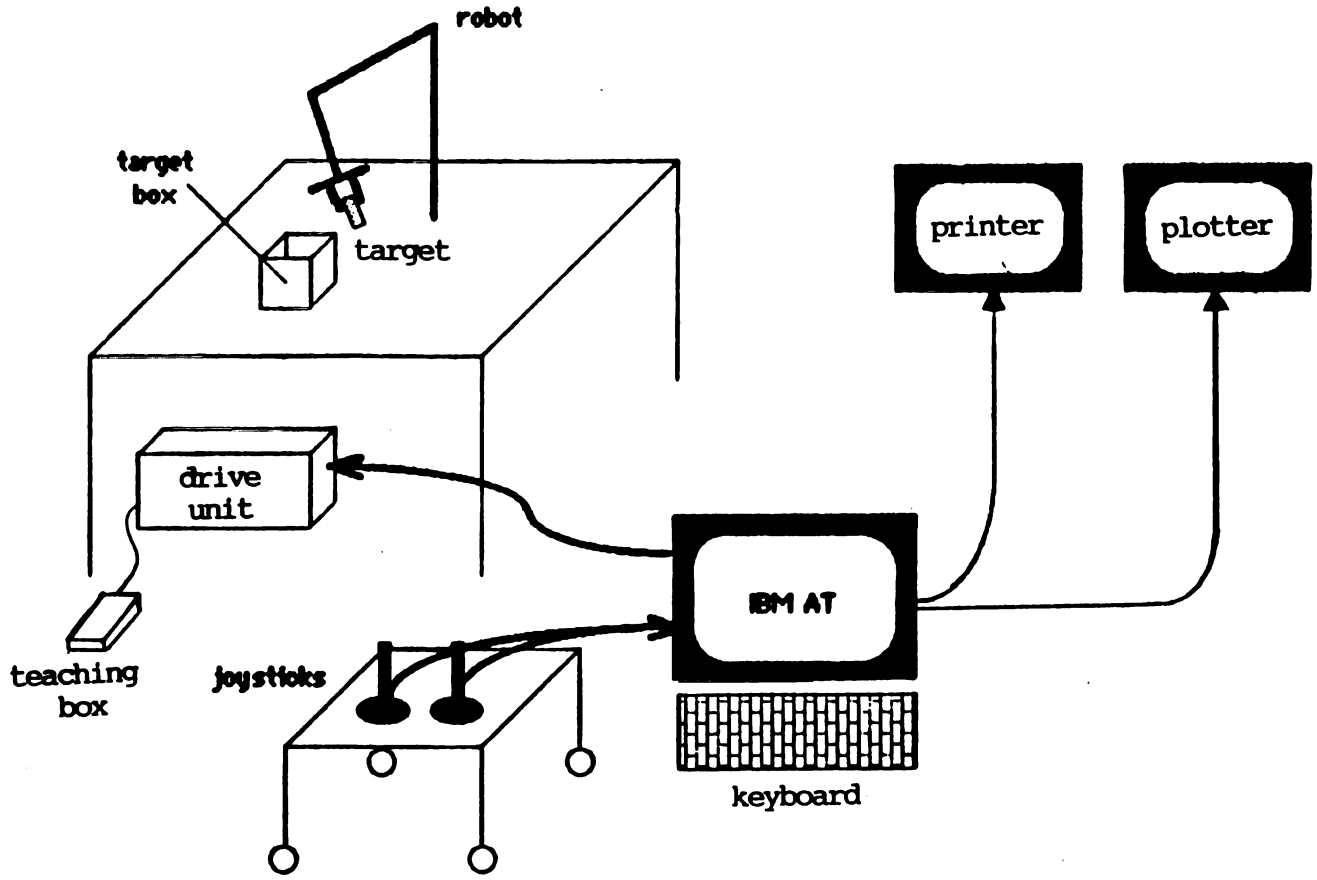


Fig.3.1 Experimental system

Table 3.1 Robot commands

Type	No	Name	Input format	Function	ROM conversion	Remarks
Operation control	1	Nest	NT	Mechanical point of origin output	Yes	
	2	Home	HO	Setting of basic operation position	Yes	
	3	Origin	OG	Return to basic position	Yes	
	4	Move I	MI a_1 a_2 a_3 a_4 a_5 a_6	The individual joints move the number of steps specified by parameters a_1 to a_6	No	Operating range after execution of NEST $-2000 \leqq a_1 \leqq 0$ $-5200 \leqq a_2 \leqq 0$ $0 \leqq a_3 \leqq 3600$ Bending $0 \leqq (a_2+a_3) \leqq 4800$ Rotation $-9600 \leqq (a_4+a_5) \leqq 9600$ a_6 always zero
	5	Move	MO$_a$	The robot moves to the point cataloged in the teaching box or set by the PS command	Yes	$1 \leqq a \leqq 629$ (or $a=0$)
	6	Increment	IP	Move the robot to the position one step ahead of the current position	Yes	
	7	Decrement	DP	Move the robot to the position one step behind the current position	Yes	
	8	Position set	PS a_1 a_2 a_3 a_4 a_5 a_6	Define a position determined by the number of steps from the origin of operation, together with the position number	Yes	$1 \leqq a \leqq 629$ (or $a_1=0$ $a_2 \sim a_6$ same as MI command
	9	Here	HEa	Store a position of operation specified by an MI command, etc. together with the position number	Yes	$1 \leqq a \leqq 629$ (or $a=0$)
	10	Position clear	PC a_1 a_2	Delete position numbers between a_1 and a_2	No	$a_1 \leqq a_2$ $1 \leqq (a_1\ a_2) \leqq 629$ (or $a_1=0$)
Hand control	11	Grip set	GP a_1 a_2 a_3	Increase or decrease the pressure	Yes	$0 \leqq (a_1\ a_2) \leqq 7$ $0 \leqq a_3 \leqq 99$
	12	Grip opened	GO	Open the grip	Yes	
	13	Grip closed	GC	Close the grip	Yes	
	14	Grip flag	GF$_a$	Used when setting grip open/close condition during execution of the PS command	Yes	$a=0$ or 1
	15	Speed	SP$_a$	Set the operation speed (0 low speed 9 high speed)	Yes	$0 \leqq a \leqq 9$
	16	Time	TI$_a$	Stop the operation for the time specified by the parameter (in increments of 0.1 sec)	Yes	$0 \leqq a \leqq 99$

Table 3.2 Specification of the robot system

Item		Specification
Structure		Five degrees of freedom-Vertical multi-joint type
Range of movement	Waist rotation	300°
	Shoulder rotation	130°
	Elbow rotation	90°
	Wrist pitch	±90°
	Wrist roll	±180°
Permissible handling weight		max. 1.2 kg (includes weight of hand)
Maximum synthesis speed		400 mm/sec (wrist tool surface)
Position repeat accuracy		±0.5 mm (wrist tool surface)
Drive system		Electroservo drive by a DC servomotor
Main unit weight		about 27 kg

Note: The permissible handling weight (1.2 kg) is the value at a point 100 mm from the wrist tool surface.

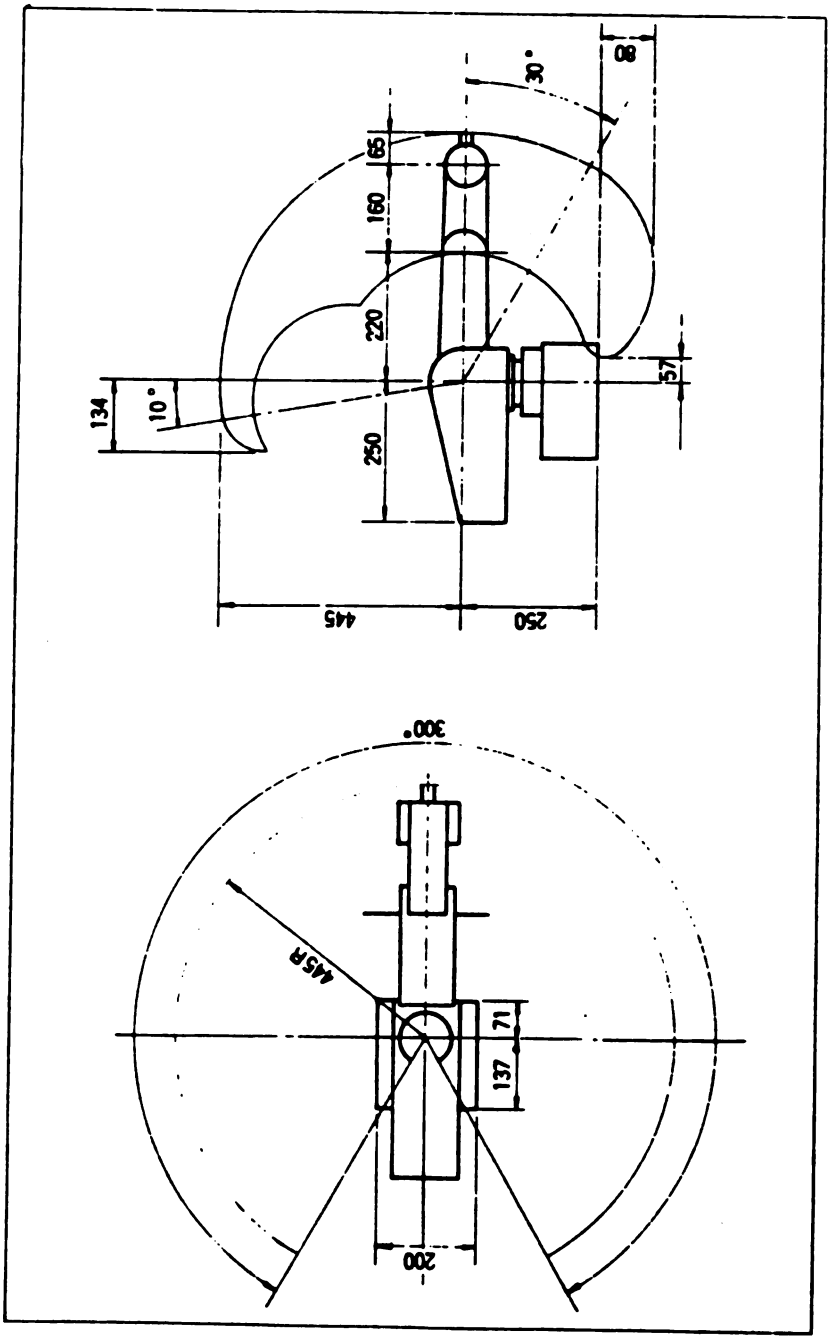


Fig. 3.2 Moving range of the robot

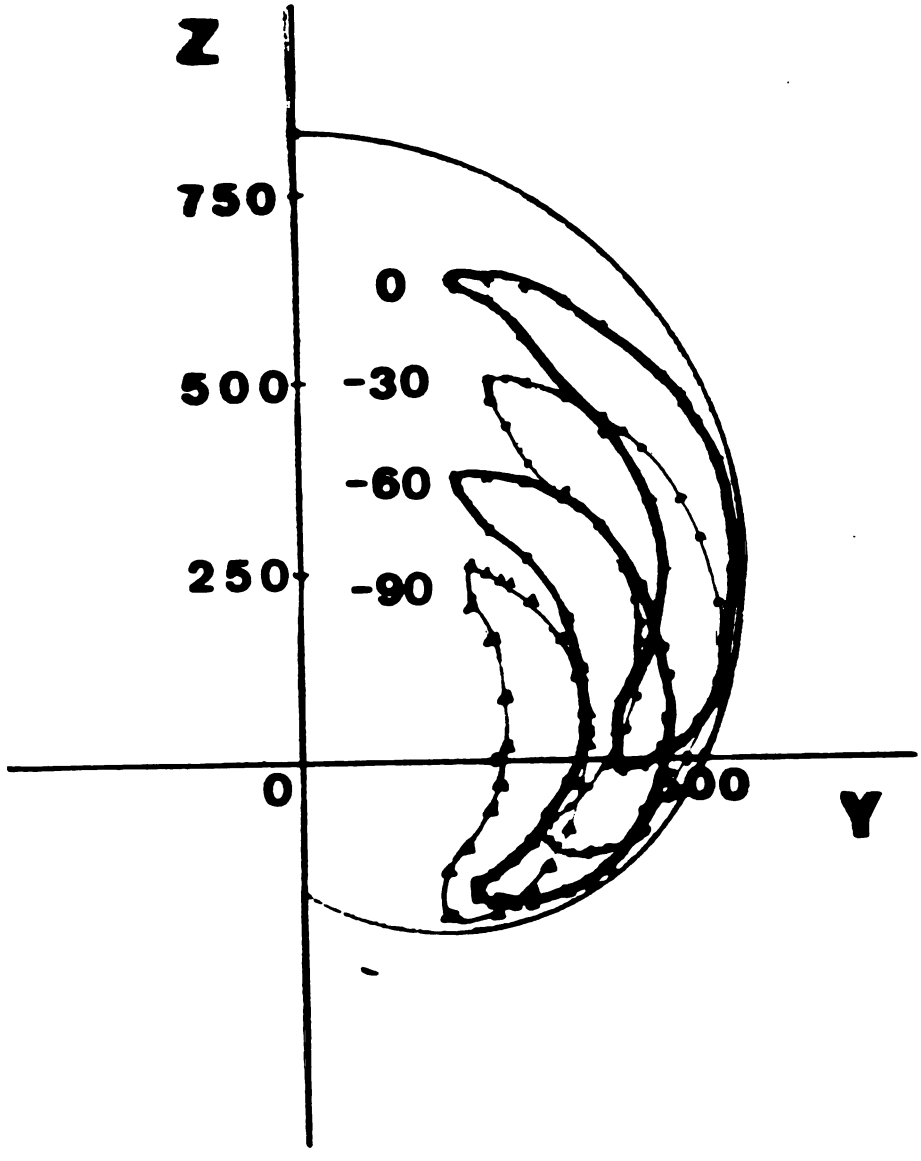


Fig 3.3 2-D workspace of the robot

3.1.2 Joysticks

Two joysticks were used for controlling the rate of the robot. Each joystick had two degrees of freedom. The right side joystick had an additional switch for hand open/close control. Horizontal movement of the stick on the right joystick was assigned as y-axis direction control for E.E.C. or as shoulder angle control for J.A.C. Vertical movement of the stick on the right joystick was assigned as z-axis direction control for E.E.C. or as elbow angle control for J.A.C. Vertical movement of the stick on the left joystick was assigned as pitch angle control for E.E.C. or as wrist angle control for J.A.C. Springs were installed on the bottom of the joystick stick to keep the central position of the stick. +- 15 volts were supplied to the joysticks. +10 volt was sent to a 12-bit A/D converter as the maximum output (rightmost or upmost position of the stick) and 0 volt was sent as the minimum output (leftmost or downmost position of the stick). Center position voltage was adjusted by potentiometers.

3.1.3 Computer system

A PC-AT system is used for developing the program, controlling the robot and analyzing the data. This system consists of CPU, 12-bit A/D converter (DASH-16), parallel printer port, EGA board, 20Mb hard disk, and floppy disk. The system also has a keyboard, a graphic display, a printer, and a plotter. The A/D converter received the signal from the joysticks and sent the 12-bit binary number to the CPU. The CPU calculated the next joint angles from the binary number and sent the robot command to the Centronics printer port. The printer port sent the command to the drive unit of the robot system. The EGA board can generate graphic figures for analyzing the measured data. Plotter outputs are also available.

3.1.4 Objects and target boxes

A micro-switch (Fig. 3.4), which is 20, 15, and 50 millimeters in length, width, and height, respectively, was used as an object. This switch was grasped by the robot hand and placed into two kinds of target boxes. For the coarse motion control experiment, large boxes with a 75

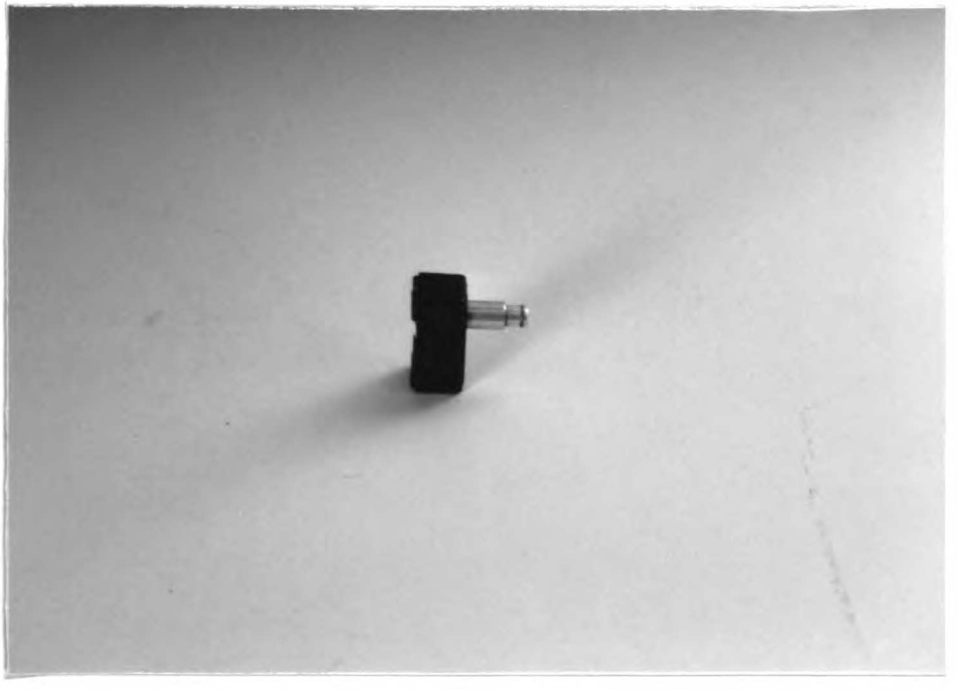


Fig. 3.4 Perspective view of an object

millimeters square entrance hole were chosen as a targets (Fig. 3.5). Three different heights of boxes (75 millimeters, 150 millimeters, and 225 millimeters) were prepared for the experiment. The box was put on the base floor vertically. For fine motion control, a small box was chosen as the target. This small box had approximately 2 millimeters clearance in inserting the object into the box and could be located at any position and orientation by the stand (Fig. 3.6).

3.2 Software

In order to perform the experiments, an end effector control program and a joint angle control program were developed in the C language. Each program included subprograms which acquired the joystick signals and sent the robot command to the drive unit of the robot system. An inverse kinematics calculation subprogram was only embedded in the end effector control program. In this program, only the elbow up solution was chosen in order to avoid the multiple solutions problem of joint angles. The joint angle limitation was informed to the operator by beep.

In order to analyze the experimental data, display and plot routines were also developed. These programs calculated the hand trajectory from the measured data and displayed or plotted the results. The program lists are attached in Appendix A.

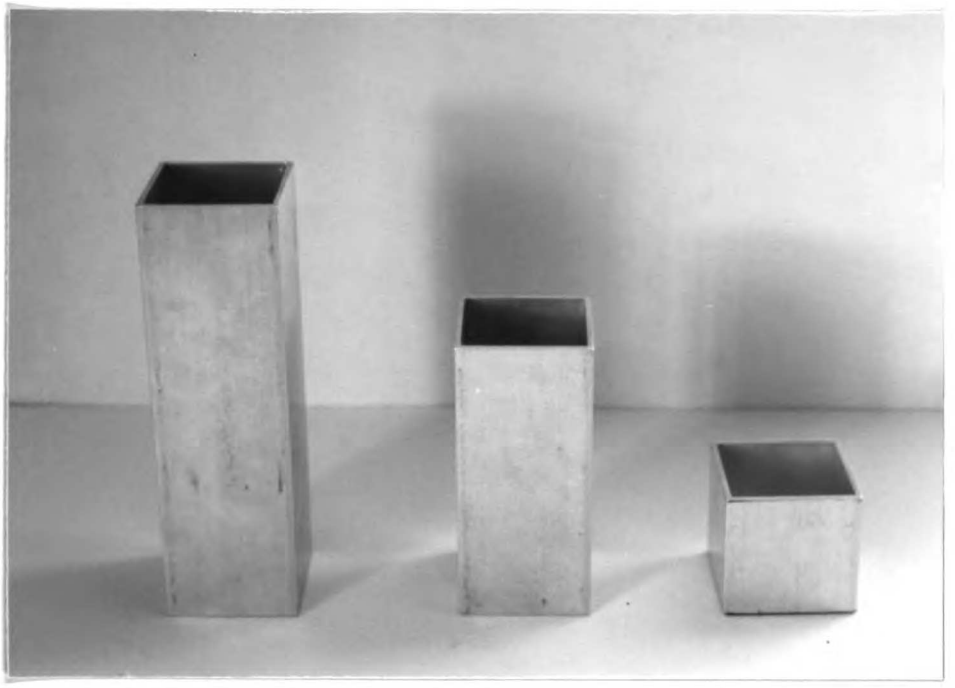


Fig. 3.5 Perspective view of large target boxes

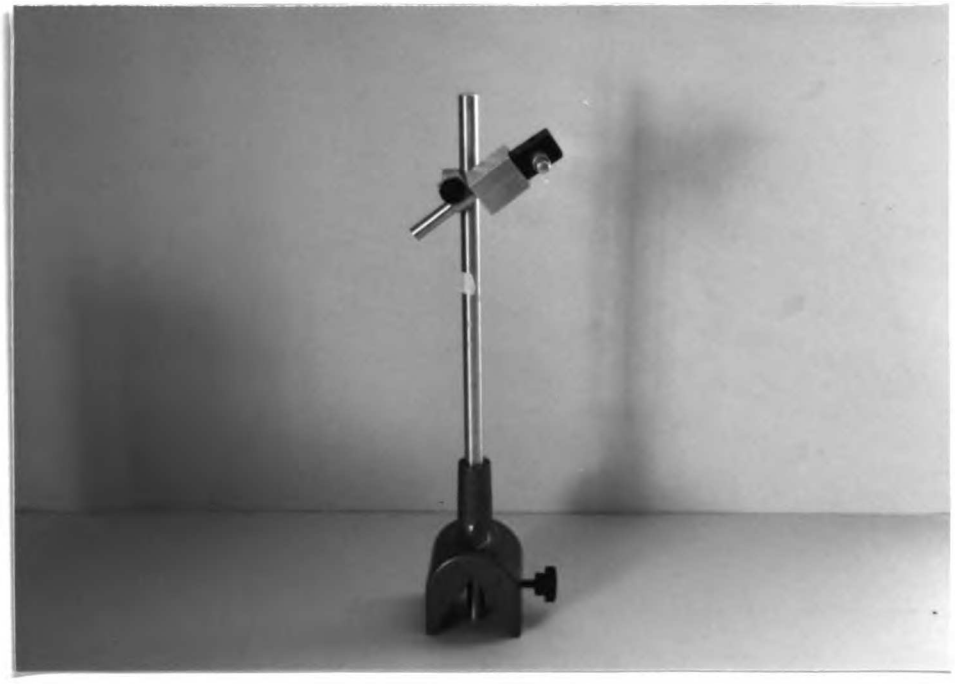


Fig. 3.6 Perspective view of a small target box with an object

4. Experimental methods

4.1 Coarse motion control

In order to investigate coarse motion control, a placement task was performed. Initially, an object was held by the robot hand at some distance from the target box. Then a subject was seated approximately 1 meter away from the robot and was asked to control the robot hand with two joysticks. The task was to move the object from the initial position to right above a large target box and drop it in. Since the box had a large entrance hole, the subject did not need high precision in placing the object into the box. Thus, the task appeared to be suitable to examine human coarse motion control strategy. Each subject performed 11 trials of the placement task with different initial positions and target positions for both E.E.C. and J.A.C. experiments. The eleven sets of different initial positions and target positions (Fig. 4.1 and Table 4.1) were chosen within the allowed workspace (Fig.3.3). The experimental trial order was 1, 7, 3, 10, 5, 11, 6, 8, 2, 9, 4. This order was selected at random but the adjacent positions have much difference in orientation. During the experiment, time and joint angles were stored in a computer file at every sampling time of 500 milliseconds.

Before the experiment, subjects practiced both J.A.C. and E.E.C. operations for more than 30 minutes until they felt comfortable with the operations. Before starting each experiment (J.A.C. or E.E.C.), each subject also performed 2 trials (in different positions from the actual experimental positions) in order to confirm each operation.

Three subjects participated in the experiments. Subjects WK and KM are 32 and 45 year-old males, respectively, and subject KT is a 28 year-old female. All three subjects performed both E.E.C. and J.A.C. experiments. Two subjects (WK and KT) performed the E.E.C. experiment first, whereas the other subject (KM) did the J.A.C. experiment first .

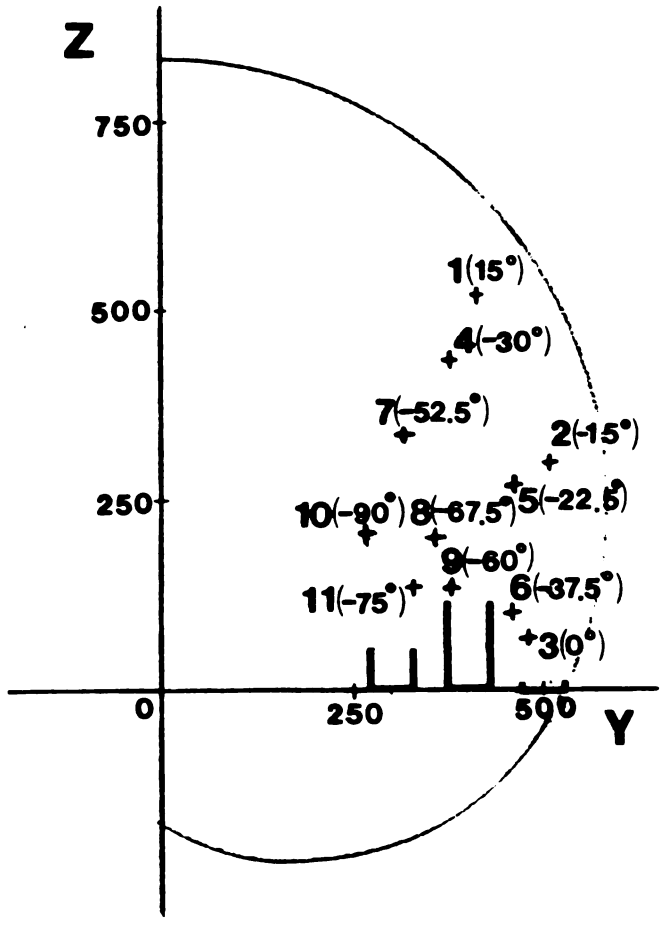


Fig. 4.1 Initial and target positions for coarse motion control

Table 4.1 Initial and target positions for coarse motion experiment

Target position number	initial position			target position	
	y (mm)	z (mm)	pitch (deg)	y (mm)	z (mm)
1	415.4	520	15	300	75 (M)
2	510	300	-15	400	75 (M)
3	480	65.4	0	300	0 (S)
4	380	434.6	-30	400	150 (T)
5	465.4	270	-22.5	300	0 (S)
6	460	100	-37.5	300	150 (T)
7	320	334.6	-52.5	500	150 (T)
8	360	200	-67.5	400	0 (S)
9	380	134.6	-60	500	0 (S)
10	270	204.6	-90	400	75 (M)
11	330	134.6	-75	500	150 (T)

4.2 Fine motion control

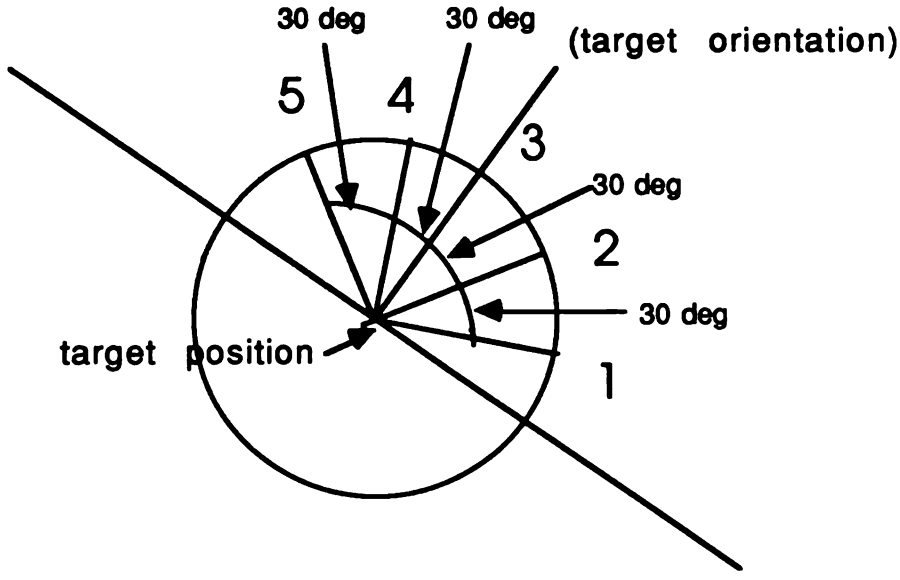
An insertion task was performed to investigate fine motion control. The subject inserted the object into a small box from an initial position in the vicinity of the target position. The initial position and orientation had some relationship to the target position and orientation. These position and orientation deviations are represented

by 1 - 5 numbers shown in Fig. 4.2. As the initial position was near the target position and the small box had only 2 millimeters clearance in inserting the object into the box, this task appeared to be suitable to examine human fine motion control strategy.

In the same way as the coarse motion control experiment, each subject performed 11 trials of the placement task with different initial positions and target positions for both E.E.C. and J.A.C. experiments. The eleven sets of different initial positions and target positions (Fig. 4.3 and Table 4.2) were chosen within the allowed workspace. The experimental trial order was 1, 7, 3, 10, 5, 11, 6, 8, 2, 9, 4. This order was the same as the coarse motion experiment. During the experiment, time and joint angles were stored to the file in the hard disk every sampling time. Each subject performed two trials before starting each experiment (J.A.C. or E.E.C.) in order to confirm the control strategy.

The same subjects as the coarse motion control experiment participated in this experiment. Two subjects (WK and KT) performed the E.E.C. experiment first, whereas the other subject (KM) did the J.A.C. experiment first.

Position deviation



Orientation deviation

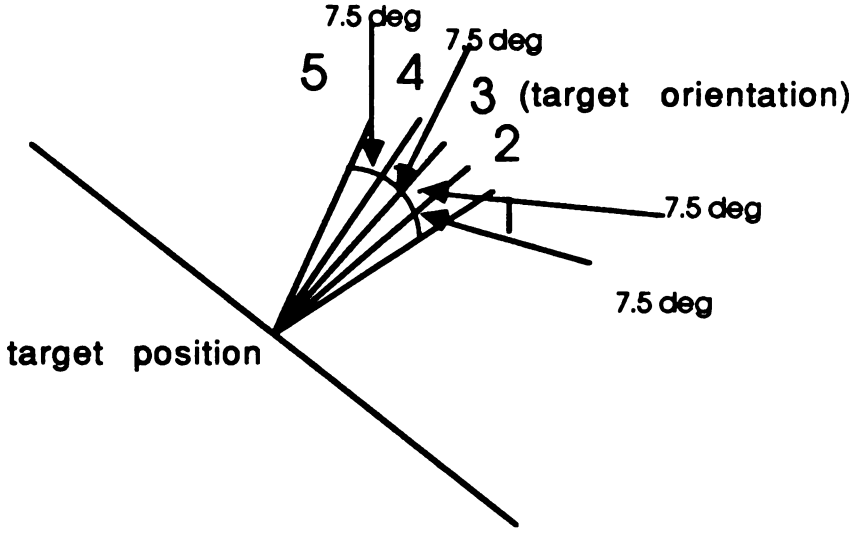


Fig. 4.2 Position and orientation deviation

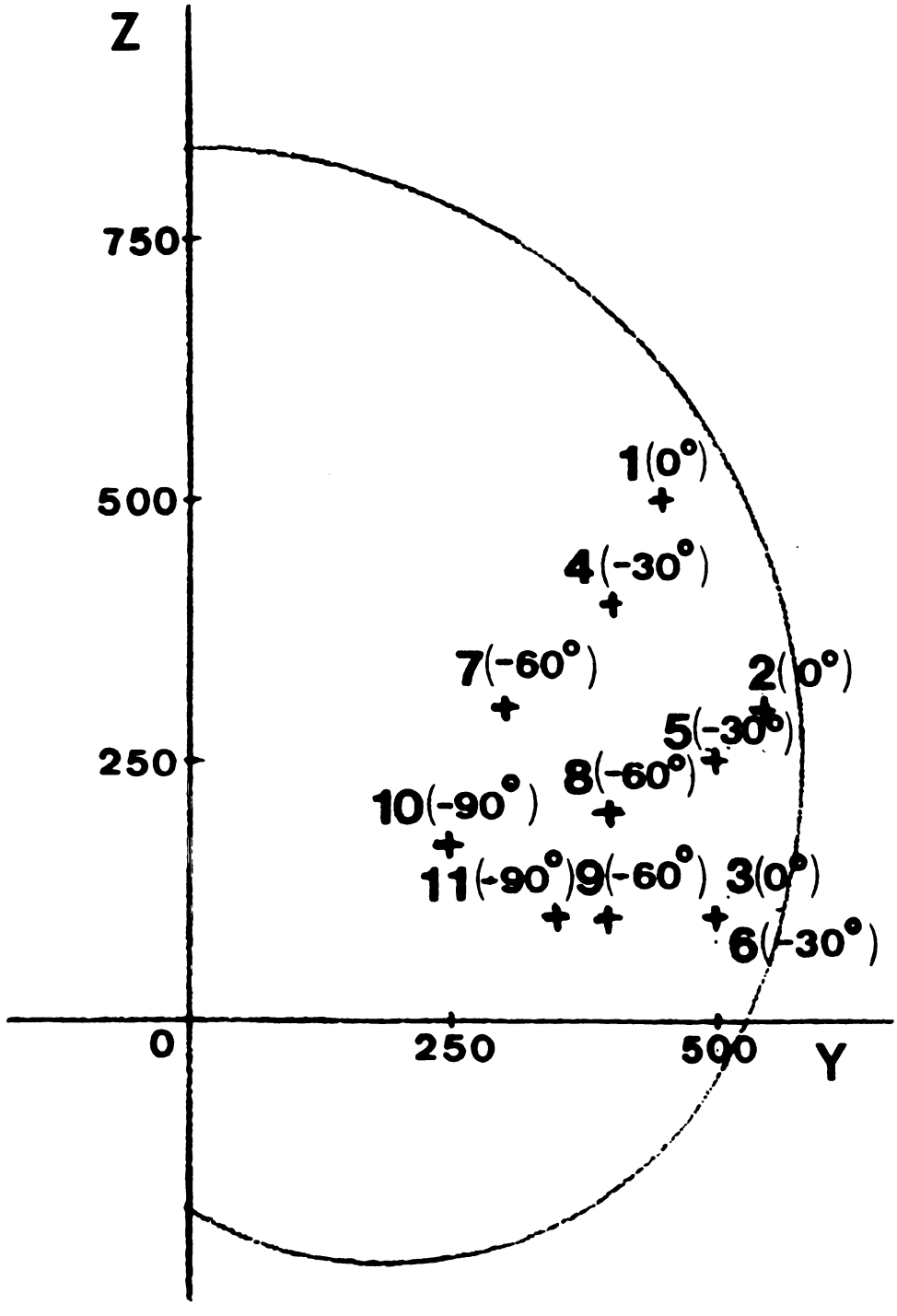


Fig.4.3 Target positions for fine motion control

Table 4.2 Initial and target positions for fine motion experiment

target position number	initial position			target position		
	y (mm)	z (mm)	pitch (deg)	y (mm)	z (mm)	pitch (deg)
1 (4,1)	415.4	520	15	450	500	0
2 (3,5)	510	300	-15	550	300	0
3 (1,3)	480	65.4	0	500	100	0
4 (4,3)	380	434.6	-30	400	400	-30
5 (3,2)	465.4	270	-22.5	500	250	-30
6 (2,1)	460	100	-37.5	500	100	-30
7 (5,2)	320	334.6	-52.5	300	300	-60
8 (1,4)	360	200	-67.5	400	200	-60
9 (3,3)	380	134.6	-60	400	100	-60
10 (4,3)	270	204.6	-90	250	170	-90
11 (2,1)	330	134.6	-75	350	100	-90

5. Experimental results

5.1 Coarse motion control

In the coarse motion control experiment, All three subjects succeeded in the placement tasks 100% for both E.E.C. and J.A.C. The completion time plots of the coarse motion control experiments for three subjects are shown in Fig. 5.1 - Fig. 5.3., respectively. The average completion time plot of these three subjects is also shown in Fig. 5.4. In these figures, the completion times of the placement task are plotted for the initial position numbers. The distance and the height of the target position are written below the initial position numbers. Open marks show the results of E.E.C. and filled marks show that of J.A.C.

These figures indicate that no significant difference in the completion time was observed between E.E.C. and J.A.C. The mean completion times of E.E.C. and J.A.C. are 28.60 seconds and 27.56 seconds, respectively.

These figures also show that in E.E.C. some positions (initial positions 3 and 7) are significantly more difficult to control than the other positions. But in J.A.C. the dependence upon the position is not significant as in E.E.C. The trajectories of position 7 and 8 of subject KT for E.E.C. and J.A.C. are shown in Fig. 5.5 -Fig. 5.8. , respectively.

5.2 Fine motion control

The results of the fine motion control experiment are shown in Table 5.1. In this table, a circle indicates that a subject inserted the object into the target box successfully without moving the box. A triangle is used when a subject moved the target box but managed to insert the object. A 'x' mark is used when a subject failed to insert the object. This table shows that each subject succeeded the task for most cases in E.E.C., but moved the target box for most cases in J.A.C.

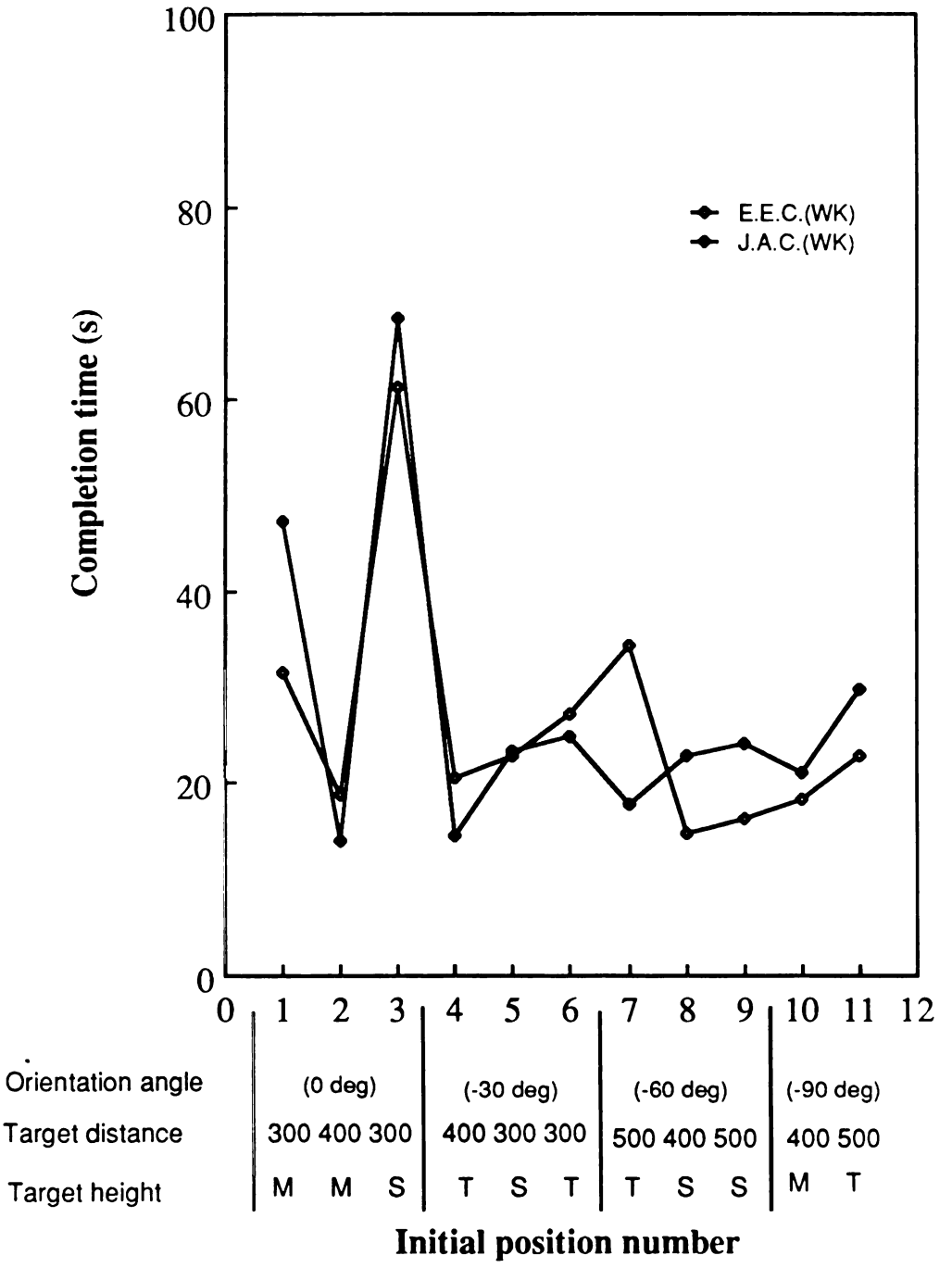


Fig. 5.1 Completion time comparison (coarse motion)

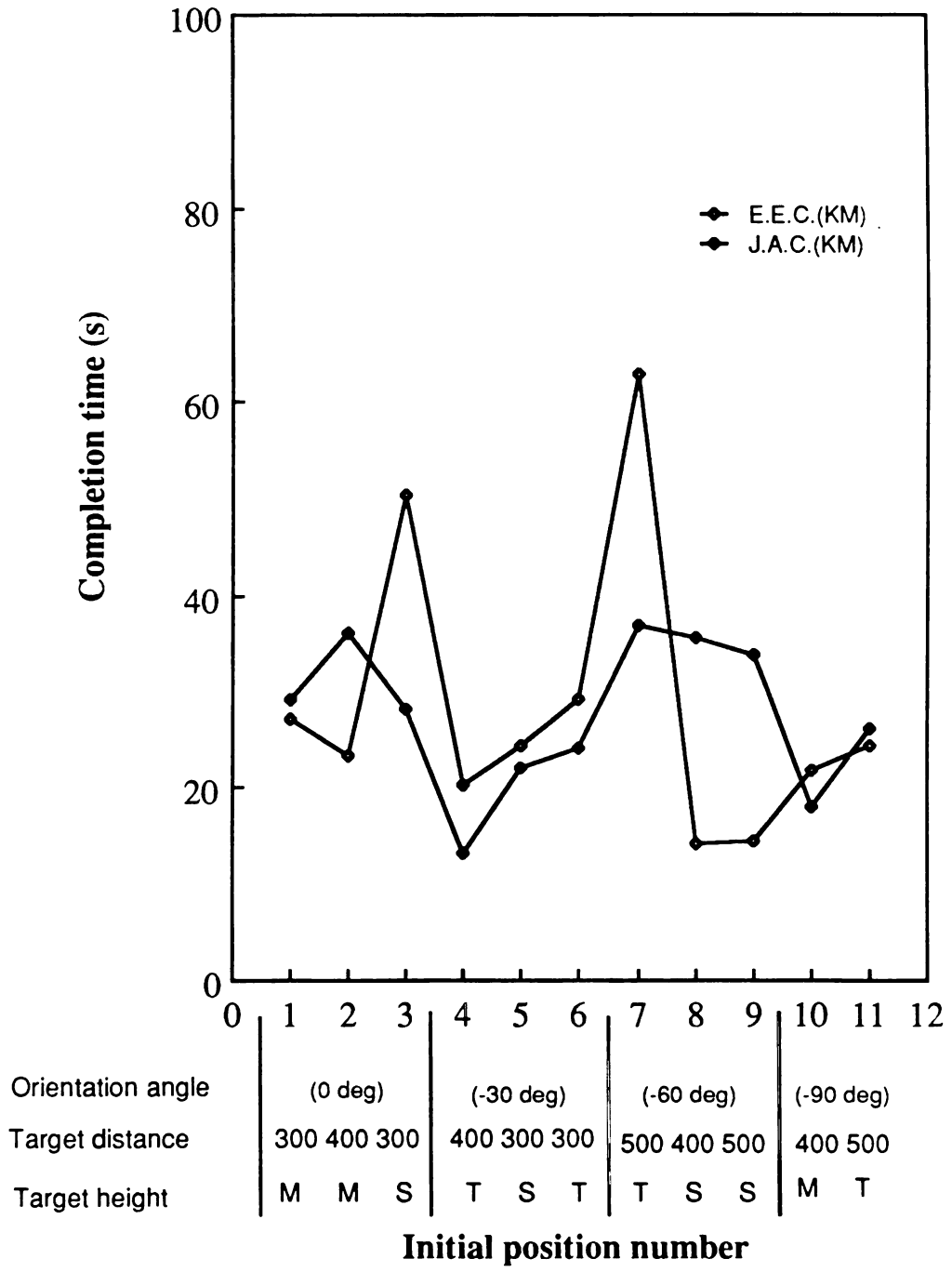


Fig. 5.2 Completion time comparison (coarse motion)

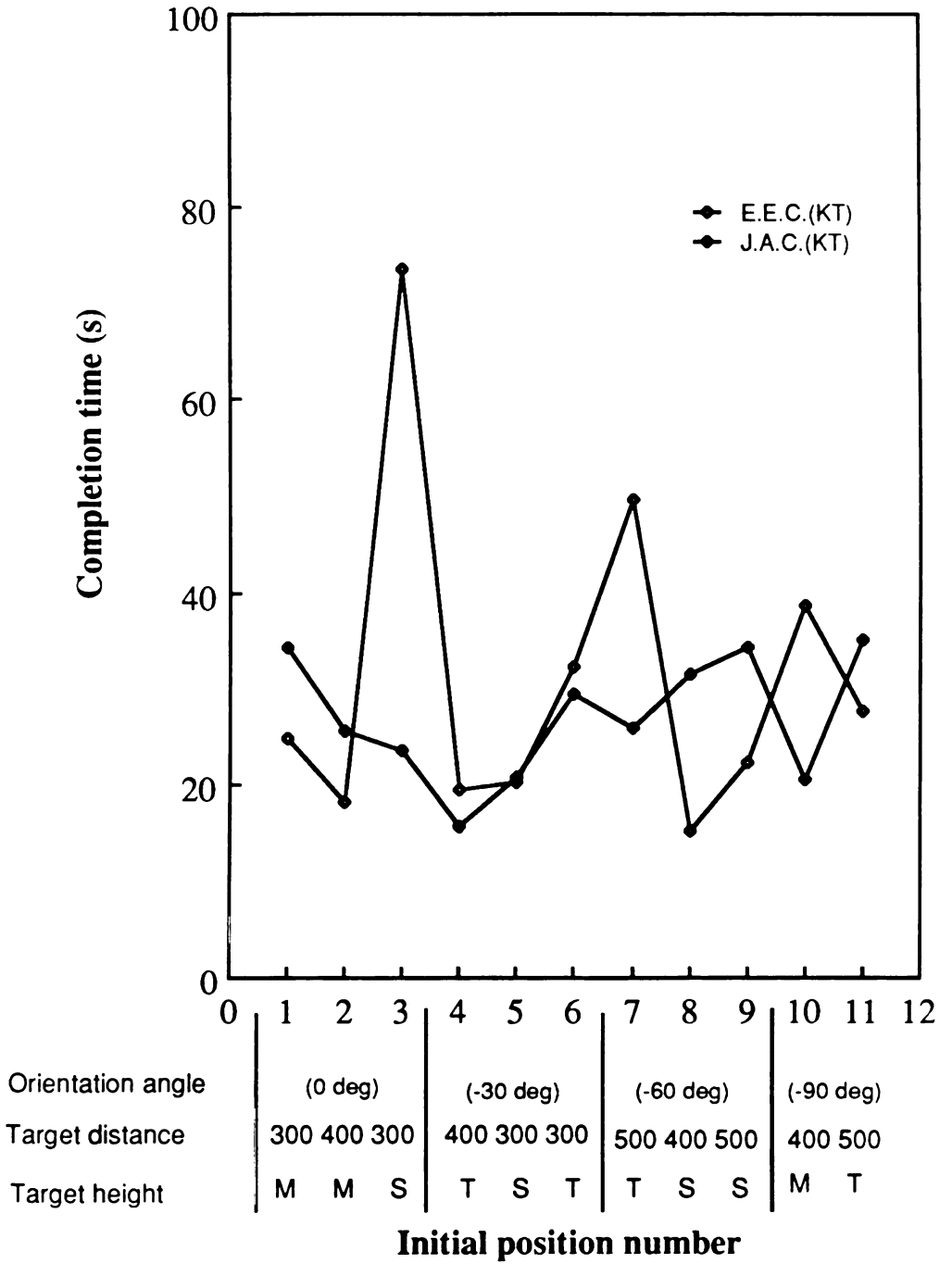


Fig. 5.3 Completion time comparison (coarse motion)

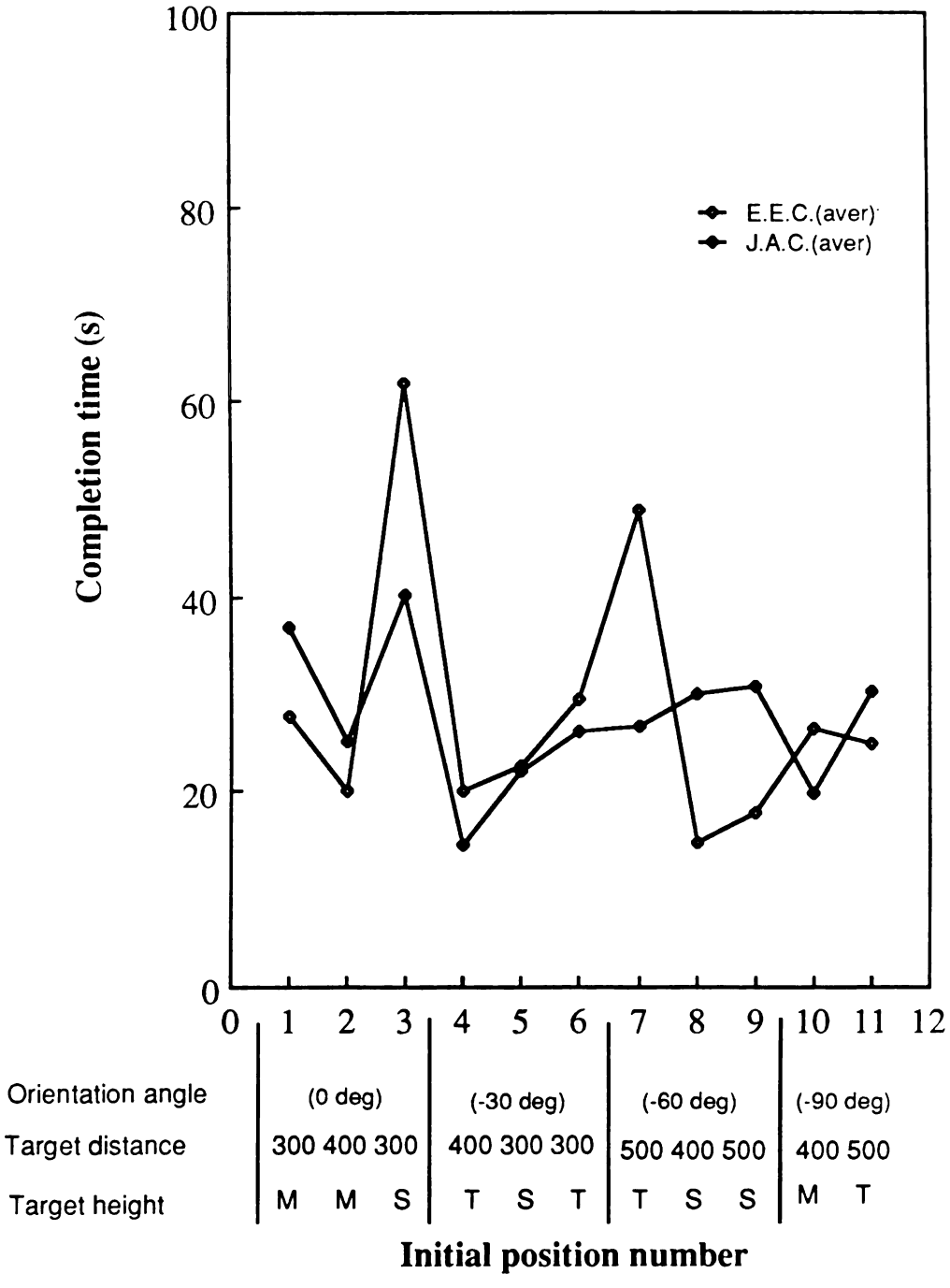


Fig. 5.4 Completion time comparison (coarse motion)

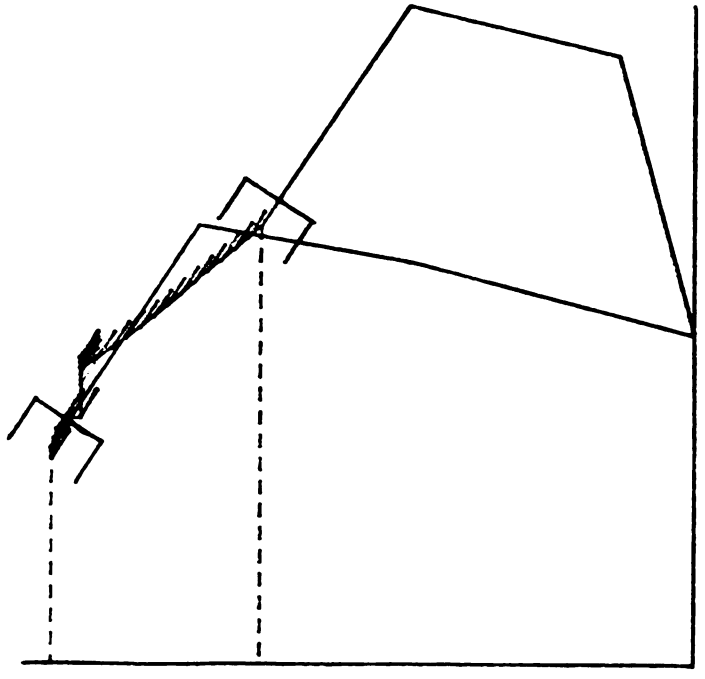


Fig. 5.5 Coarse motion trajectory for E.E.C. (position 7)

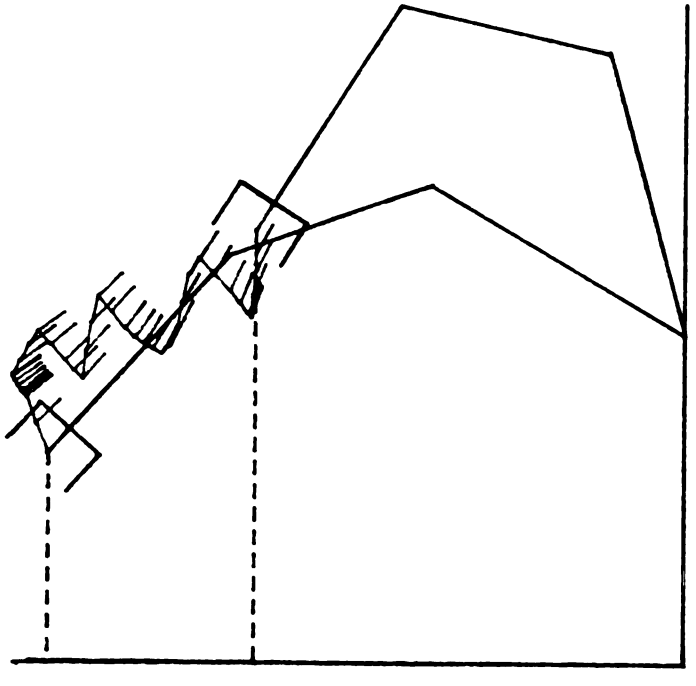


Fig. 5.6 Coarse motion trajectory for J.A.C. (position 7)

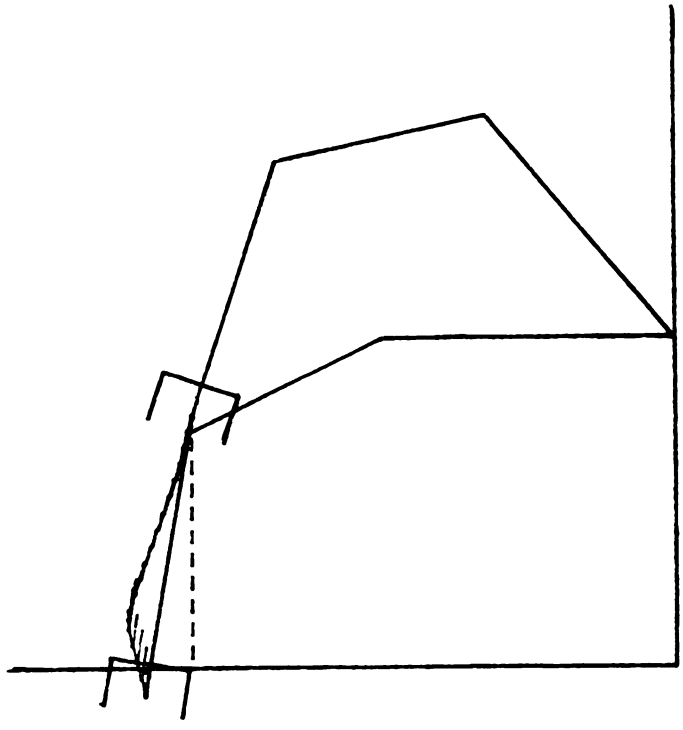


Fig. 5.7 Coarse motion trajectory for E.E.C. (position 8)

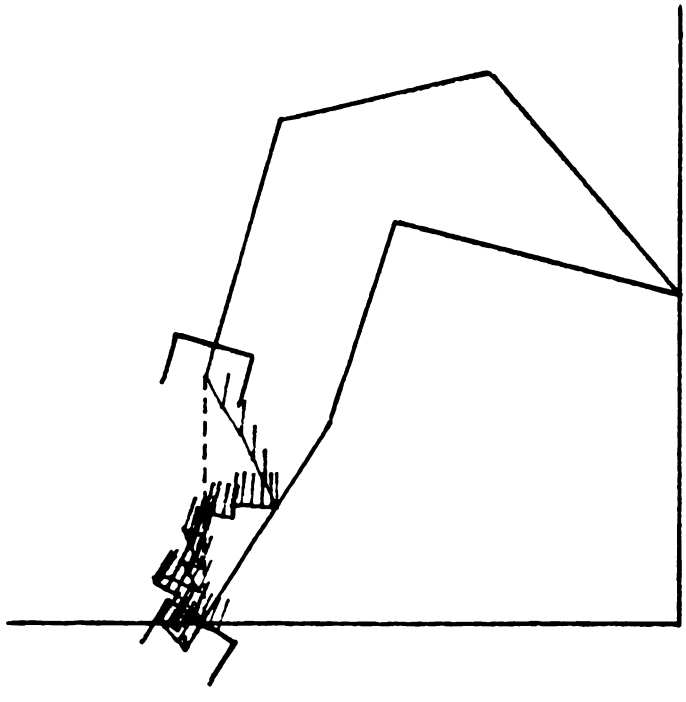


Fig. 5.8 Coarse motion trajectory for J.A.C. (position 8)

Table 5.1 Experimental results for fine motion control

Target position number	subject(WK)		subject(KM)		subject(KT)	
	E.F.C.	J.A.C.	E.F.C.	J.A.C.	E.F.C.	J.A.C.
1	○	○	○	△	○	△
2	○	△	○	×	○	△
3	○	△	○	△	○	△
4	△	△	○	△	○	△
5	○	△	○	△	△	×
6	○	△	○	△	△	△
7	○	△	○	△	○	○
8	△	△	○	△	△	△
9	○	○	△	△	○	△
10	○	△	○	△	○	○
11	○	△	△	△	○	△

○ : success △ : move target box × : failure

The cases in which at least one subject moves the target box in E.E.C. are 4, 5, 6, 8, 9, 11. The typical trajectories of these cases are shown in Fig. 5.9 - 5.10. A typical moved box trajectory in J.A.C. is shown in Fig. 5.11. The cases in which one subject failed the task are 2 and 5. The cases that one subject succeeded the task in J.A.C. are 1, 7, 9, 10. The examples of failed trajectories and succeeded trajectories in J.A.C. are shown in Fig.5.12. and Fig. 5.13.,respectively.

The completion time plots of fine motion control for the three subjects are shown in Fig. 5.14 - Fig. 5.16. In these figures, the abscissa represents 11 experiment conditions. The orientation angles, position deviation numbers and orientation deviation numbers are indicated in the figures. Open marks show the results of E.E.C. and filled marks show those of J.A.C.

These figures show that E.E.C. resulted in shorter completion times than J.A.C. for all positions, orientations and subjects except for target position 7 of subject KM.

Subject WK had trouble at position 3 and took 360 seconds to insert the object for J.A.C. Subject KM and KT had trouble at positions 2 and 5, respectively, and failed to insert the object for J.A.C.

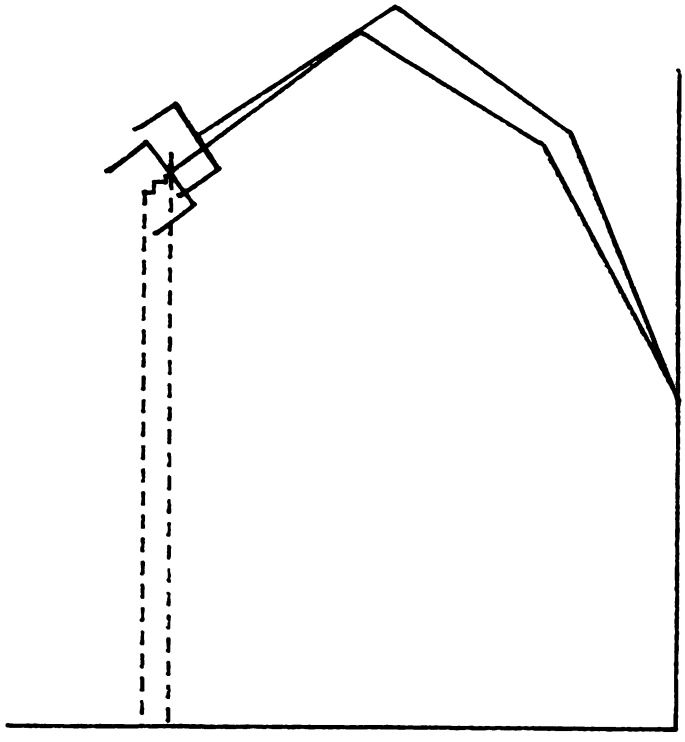


Fig. 5.9 Fine motion trajectory for E.E.C. (position 4)

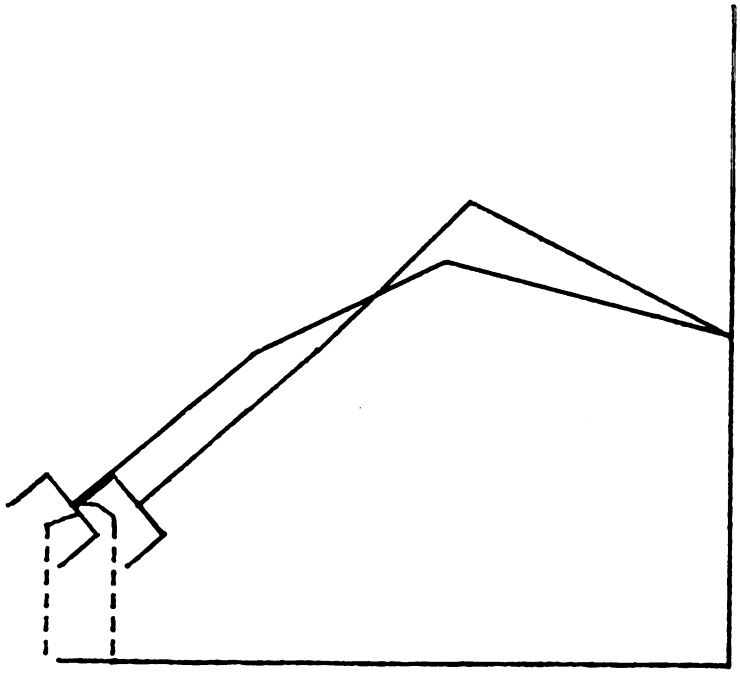


Fig. 5.10 Fine motion trajectory for E.E.C. (position 6)

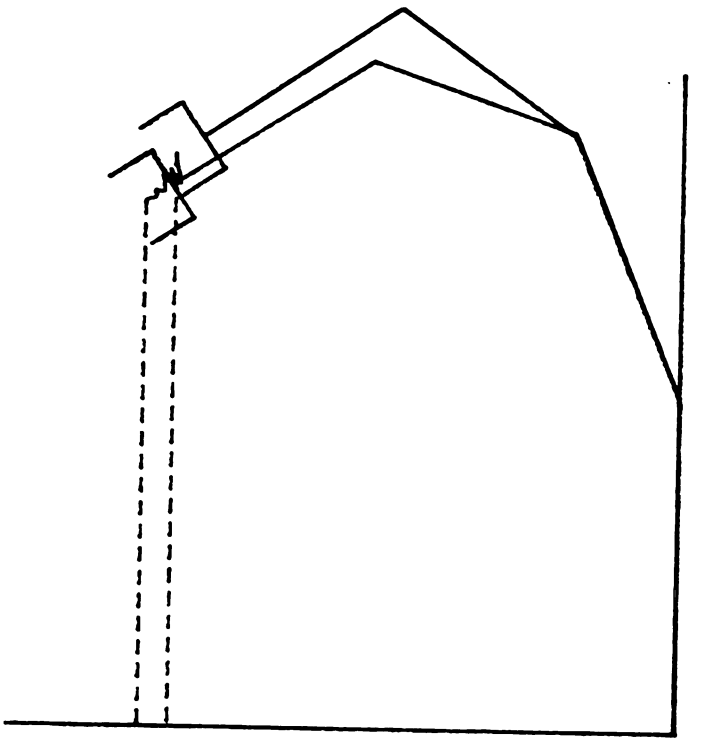


Fig. 5.11 Fine motion trajectory for J.A.C. (position 4)

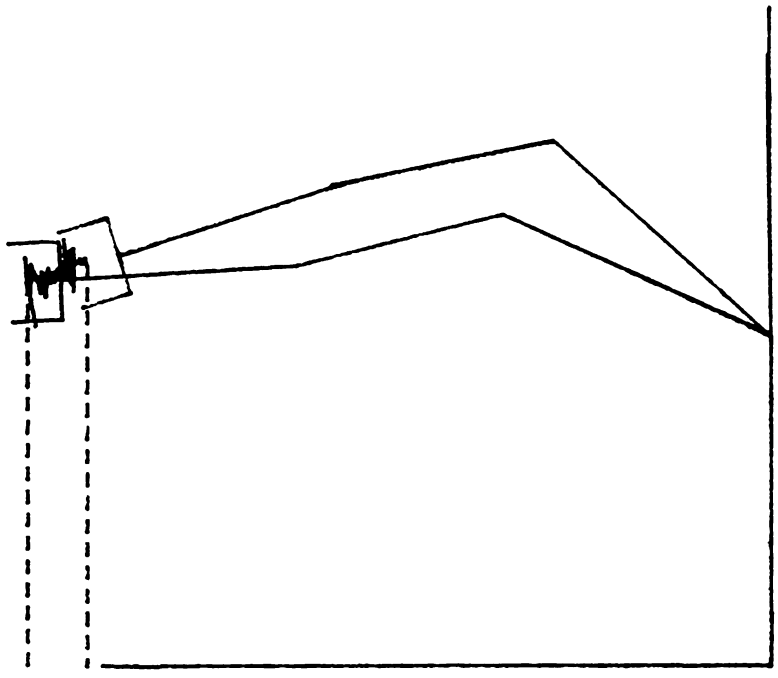


Fig. 5.12 Fine motion trajectory for J.A.C. (position 2)

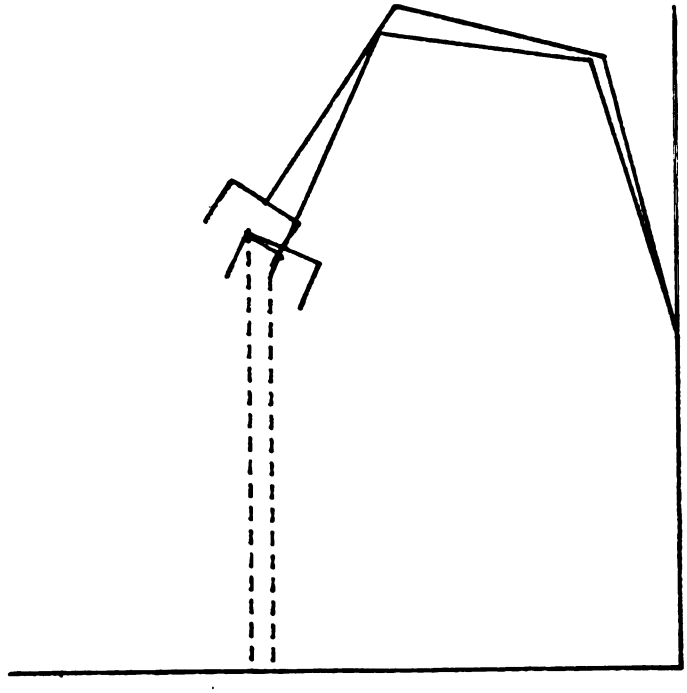


Fig. 5.13 Fine motion trajectory for J.A.C. (position 7)

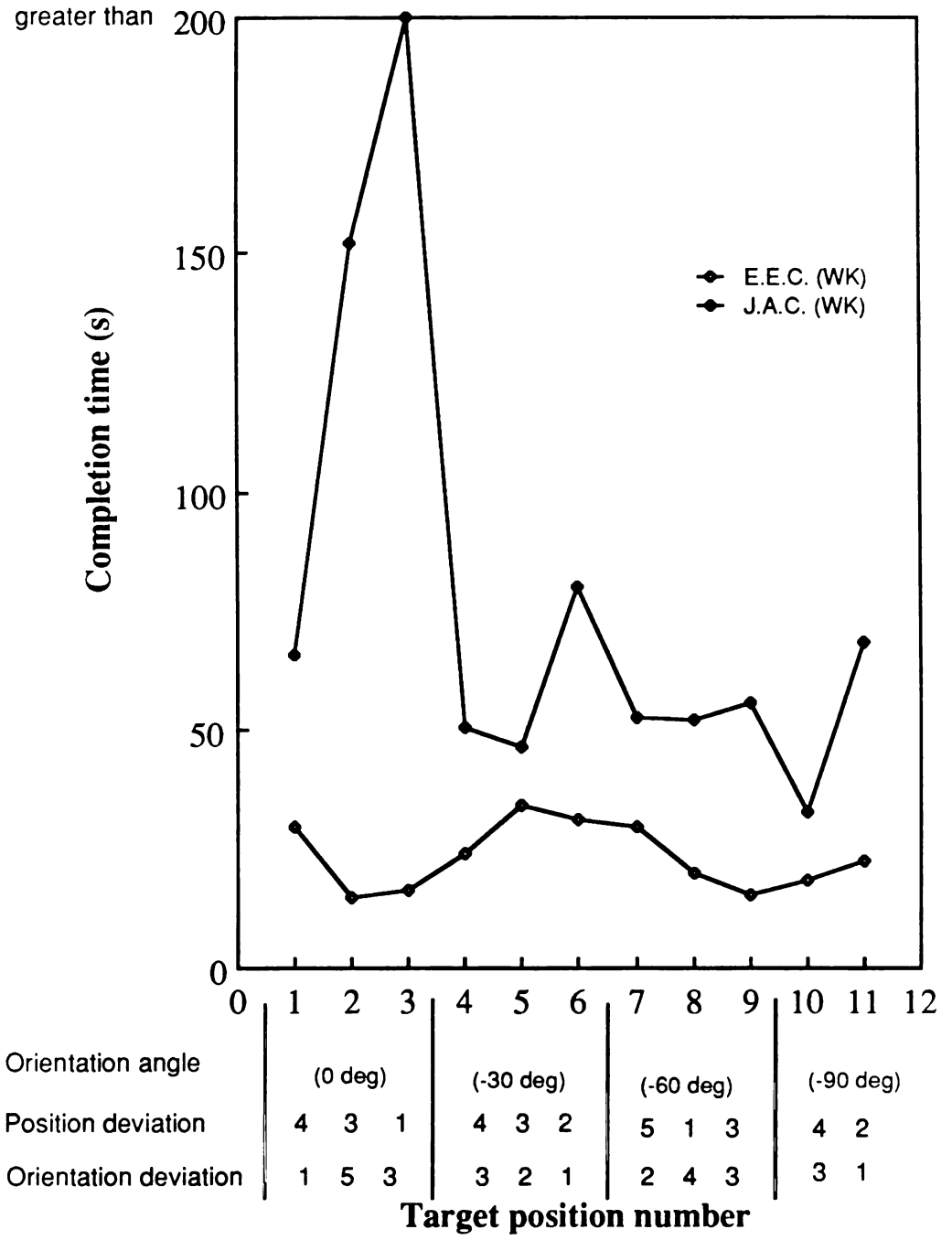


Fig. 5.14 Completion time comparison (fine motion)

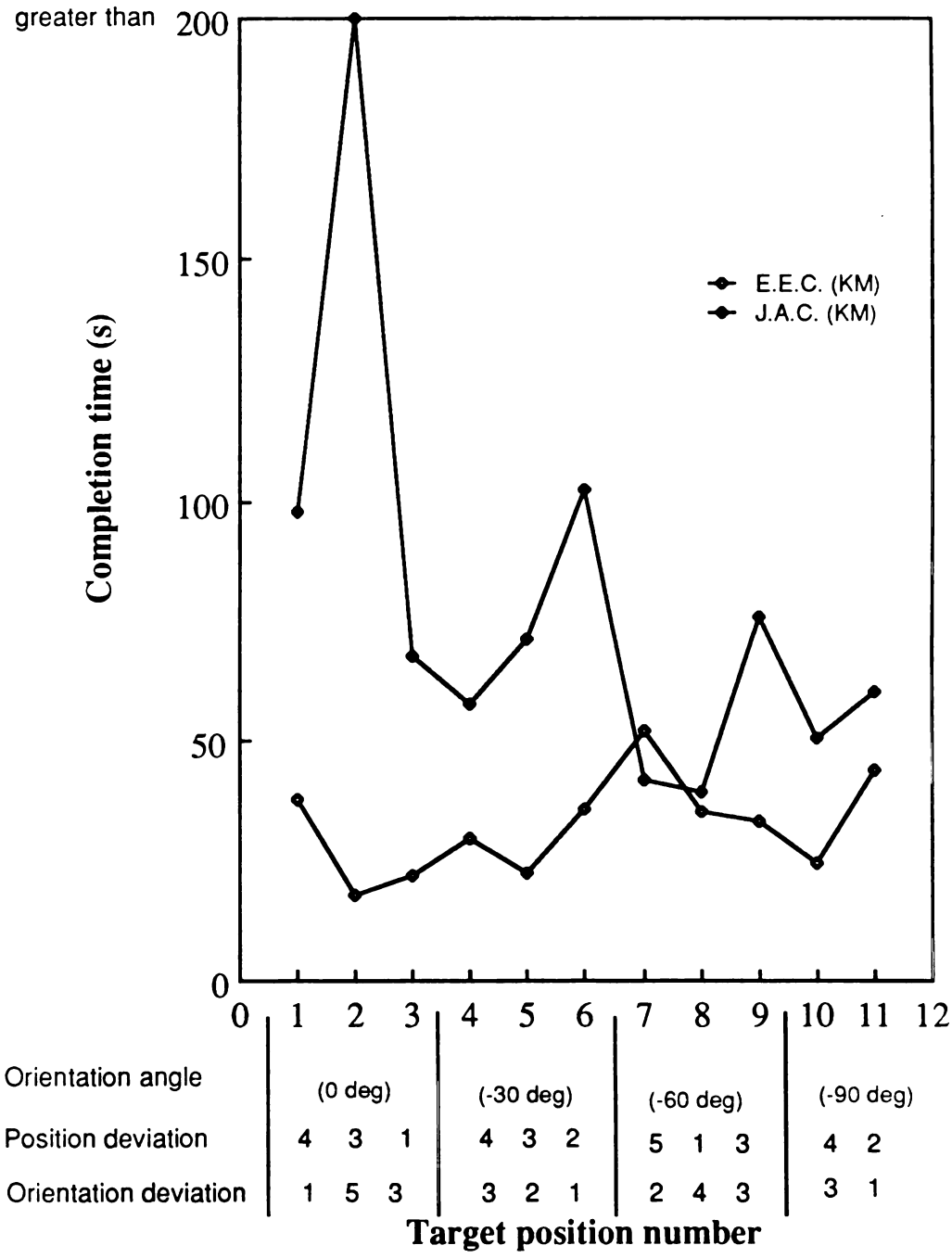


Fig. 5.15 Completion time comparison (fine motion)

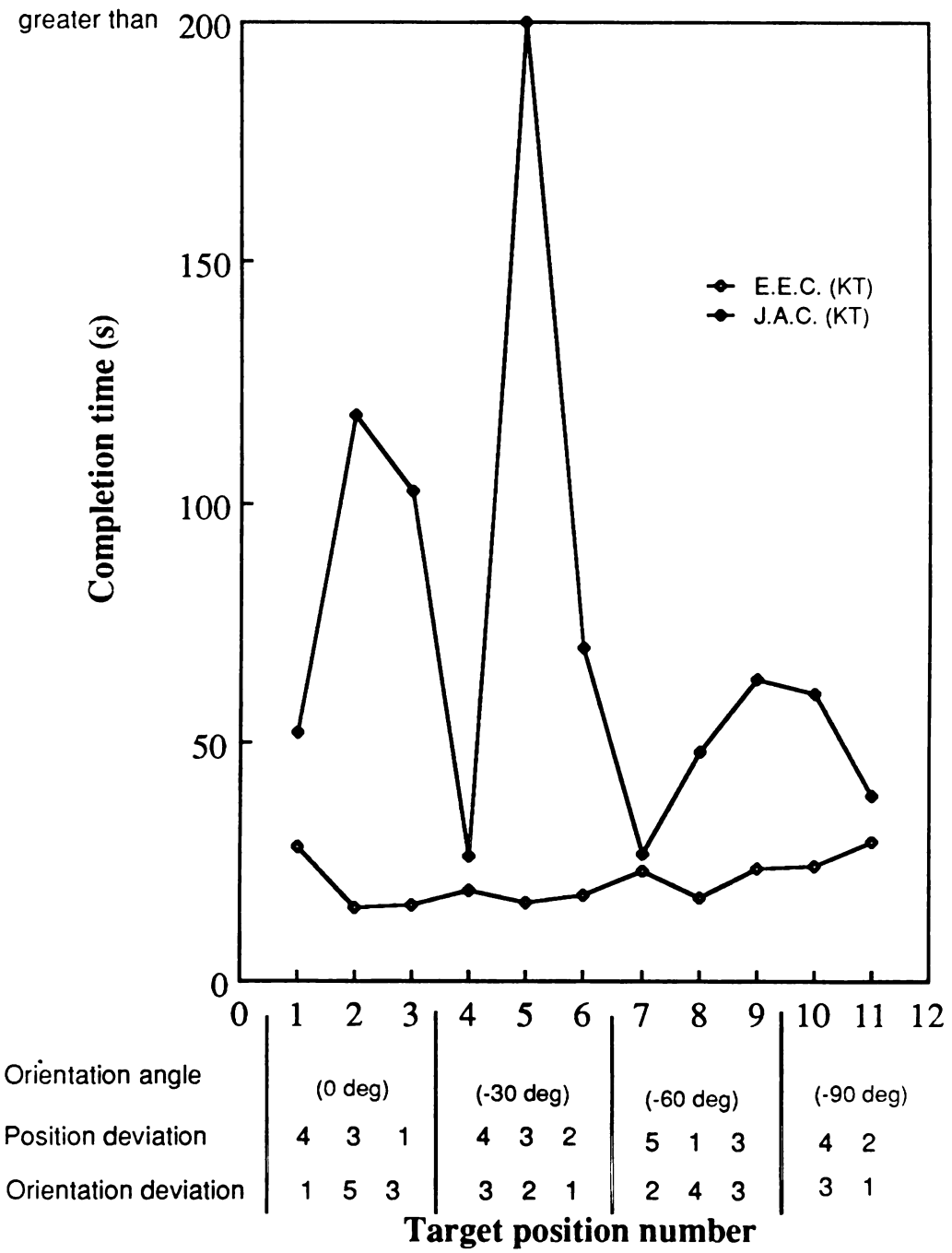


Fig. 5.16 Completion time comparison (fine motion)

6. Discussion

6.1 Control strategy in coarse motion

The experimental results show that no significant difference between E.E.C. and J.A.C. existed for coarse motion. All subjects succeeded all placement tasks in both strategies. E.E.C. took less time than J.A.C. in some conditions, while in the other conditions J.A.C. took less time than E.E.C. The mean completion times of E.E.C. and J.A.C. are almost the same, that is, 28.60 seconds and 27.56 seconds, respectively. Two cases (target positions 7 and 8) which had the largest average difference in completion time between E.E.C. and J.A.C. are considered to find out the human control strategy for coarse motion in this section.

At position 7, J.A.C. took less time than E.E.C. for all subjects. The trajectories of subject KT are examined for comparing E.E.C. and J.A.C. In E.E.C. the subject KT tried to move the robot hand along a straight line, but she could not move the hand to the target position along the straight line because of the joint angle limitation (Fig. 5.5). The subject had to change the pitch angle or direction when she reached the joint angle limitation. The trajectories of the other subjects in E.E.C. were different from this trajectory. Each subject moved the hand in different direction, but all subjects reached the joint angle limitation and changed the pitch angle or direction. The subjects changed their strategies only after reaching the joint angle limitation, because it is difficult to predict the joint angle limitation in E.E.C. This changing strategy during the operation caused additional time in E.E.C. The results revealed one of the disadvantages of E.E.C., difficulty in visualizing the physical constraints.

On the other hand, in J.A.C. the subject KT moved the hand in a zigzag fashion but never reached the joint angle limitation (Fig. 5.6). In J.A.C. the subject controlled the joint angles directly. Therefore, the subject always paid attention to the joint angle limitation. The subject did not need to change her strategy during operation of the robot. The subjects, did, however, have to calculate inverse kinematics. A human operator can calculate inverse kinematics only roughly. This rough inverse kinematics calculation caused the zigzag trajectory in J.A.C. of the subject

KT, but performance was good enough for coarse motion control. The trajectories of the other subjects in J.A.C. were different from this trajectory, but they did not reach to the joint angle limitation in J.A.C. The reason that E.E.C. took longer time than J.A.C. at position 7 is that E.E.C. required thinking time to avoid the joint angle limitation after reaching the limitation.

At position 8, E.E.C. took less time than J.A.C. for all subjects. In this case the subject KT could move the hand to the target position along an almost straight line without reaching the joint angle limitation for E.E.C. (Fig. 5.7). The subject moved the hand in a zigzag fashion in J.A.C., same as that of position 7 (Fig. 5.8), because the subject could not calculate the inverse kinematics accurately. The trajectories of the other subjects had the same tendency, that is, E.E.C. had a straighter trajectory than J.A.C. and J.A.C. had a zigzag or detoured trajectory. The reason that J.A.C. took more time than E.E.C. at position 8 is that J.A.C. had a zigzag or detoured movement because of the rough inverse kinematics of a human operator.

These considerations also answer why some positions were more difficult to control than other positions in E.E.C., but no significant difference between positions were observed in J.A.C.

6.2 Control strategy in fine motion

The experimental results show that E.E.C. had better performance than J.A.C. for fine motion. For E.E.C. the subjects succeeded in 78.8% of the insertion tasks, managed to insert the object but moved the target box in 21.2% of the tasks, and never failed. Using J.A.C., however, the subjects succeeded in only 12.1% of the total tasks, and managed to insert the object but moved the target box in 81.8% of the tasks, and failed 6.1% of the task (Table 5.1). E.E.C. took less time than J.A.C. for all positions, orientations, and subjects except for target position 7 of the subject KM (Fig. 5.14 - Fig. 5.16).

For fine motion control, accurate inverse kinematics calculation is necessary. E.E.C. imposes the inverse kinematics calculation on a computer. Therefore, the human operator does not need to calculate the inverse kinematics. What a human operator has to do is to specify the movement direction with the joysticks. E.E.C. can also separate position control from orientation

control. In this simple 2-D experiment, the only coupling in E.E.C. was the coupling between horizontal and vertical axes. The experimental results show that the cases in witch the subjects moved the target box in E.E.C. were those coupling movements, that is, the cases with pitch angles of -30 or -60 degrees, except for target position 11 of the subject KM, for witch the subject inserted the object into the target box before adjusting the orientation (Table 5.1). Typical examples of the subjects moving the target box were the case in which the subject moved the hand horizontally and vertically by turns (Fig. 5.9) and the case in which the subject moved the hand to a different direction (Fig. 5.10). For a human operator, it is difficult to control more than 2 degrees of freedom at the same time.

On the other hand, J.A.C. imposes the inverse kinematics calculation on the human operator. The human operator can roughly calculate inverse kinematics within a short time, but this rough calculation is not enough for fine motion control. Even in this easy experiment (insertion task with 2 millimeters clearance), the subjects succeeded in only 12.1% of the tasks for J.A.C. Examining the trajectories of fine motion in J.A.C., we can see that the subjects carefully adjusted each joint angle so that the robot's hand followed the line constrained by the target box, but this resulted in zigzag movements because of the rough inverse kinematics calculation of the subjects and the necessity of multiple joint angle control (Fig. 5.11). The subject had to move 3 joint angles simultaneously for the failed case (Fig. 5.12), but the subject could move only two joints for the successful case (Fig. 5.13). Multiple joint angle control is difficult for the subjects.

The completion times of E.E.C. and J.A.C. also show that E.E.C. is easier than J.A.C. for fine motion control (Fig.5.14- Fig. 5.16). For J.A.C. some positions took more time than the other positions. The reasons were that at some positions the inverse kinematics calculation was more difficult than that at the other positions and control of 3 joint angles was necessary.

The reasons that E.E.C. had better performance than J.A.C. for fine motion are that E.E.C. gives an accurate inverse kinematics calculation, which is difficult for the human operator, on a computer and can separate position control from orientation control, whereas J.A.C. imposes

the inverse kinematics calculation and simultaneous multiple joint angle control on a human operator.

In these experiments, the subjects were allowed to practice the operation before the different control strategy experiment. Therefore, the order of the experiment (E.E.C. and J.A.C.) did not affect the results.

7. Conclusions

Human coordination and motion control strategies in operating a telerobot were investigated using placement tasks with a joystick-controlled robot. Preliminary experimental results show that human motion control demonstrates two strategies, coarse and fine motion control. Coarse motion is fast, large, and rough motion from the initial position to the vicinity of the target position. Fine motion is slow, small, and accurate motion from the vicinity of the target position to the actual target position. The experimental results indicate that for coarse motion control no significant difference in success rate or completion time was observed between end effector control(E.E.C.) and joint angle control (J.A.C.), whereas for fine motion control E.E.C. was significantly better than J.A.C.

One of the disadvantages of E.E.C., difficulty in visualizing the physical constraints, sometimes emerged in coarse motion control because of its large motion characteristics. The disadvantage of J.A.C., difficulty in predicting the movement of the hand from joint commands, was not important in coarse motion control because of its rough motion characteristics. Therefore, no significant difference in performance was observed between E.E.C. and J.A.C. for coarse motion.

On the other hand, one of the disadvantages of E.E.C., difficulty in visualizing the physical constraints, was unimportant in fine motion control because of its small motion characteristics. The disadvantage of J.A.C., difficulty in predicting the motion of the hand, always emerged in fine motion control because of its accurate motion characteristics. Therefore, E.E.C. had better performance than J.A.C. for fine motion.

Bibliography

1) A. K. Bejczy, "Sensors, controls, and man-machine interface for advanced teleoperation", Science, vol. 208, no. 4450, pp 1327-1335, 1980

2) D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses", IEEE Tran. Man-Mach. Syst., vol. MMS-10, pp 47-53, 1969

3) J. M. Hollerbach and C. G. Atkeson, "Deducing planning variables from experimental arm trajectories: pitfalls and possibilities", Biol. Cybern., submitted, 1986

4) J. F. Soechting and F. Lacquaniti, "Invariant characteristics of a pointing movement in man", J. Neurosci. 1, pp 710-720, 1981

5) K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, Robotics: control, sensing, vision, and intelligence, McGraw-Hill Book Company, 1987

6) M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, Robot Motion : Planning and control, Cambridge, The MIT Press, 1982

7) R. P. Paul, Robot manipulators: Mathematics, programming, and control, Cambridge, The MIT Press, 1981

Appendix A

Program lists

1. Control programs
   1.1 End effector control program (2robjme.c)
   1.2 Joint angle control program (2robjae.c)
2. Display program (drob1c.c)
3. Plot program (prob1c.c)

```
/*-----------------------------------------------------------------

File
        2robjme.c  - operate robot using joystick and measure joint
                     angle (2-D version experiment, 10 target position)
Synopsis
        #include <stdio.h>
        #include <math.h>
        #include <conio.h>
        #include "ad.c"

        lcl 2robjme
        linkml 2robjme draw display atime

Description
        This program controls end effector position and orientation
        of a movemaster II and stores time andjoystick angles (robot
        command parameters) in file.
        This program uses a function "jtctran()" which transforms
        joystick signal to cartesian coordinate value.  The joystick
        signal is converted to cartesian coordinate value by multi-
        pling gains.  The joystick signal is connected to a/d converter.
        The right side joystick is used for control y and z coordinates
        movement. The left side joystick (vertical) is used for pitch
        angle control. In addition to the joystick handle, each joystick
        box has a swich on the top face. The switch of the right side
        joystickh is used for hand open and close control (up and middle
        position is assigned to close and down position is assigned to
        open). The switch of the left side joystick is not used in this
        program. The a/d converter configuration is as following:
                a/d channel 0 -- x coordinate movement or roll movement
                                 (right horizontal, red wire, 0-2047)
                a/d channel 1 -- y coordinate movement (right vertical,
                                 green wire, 0-2047)
                a/d channel 2 -- hand open close (right switch, white
                                 wire, open:0 close:>900)
                a/d channel 3 -- z coordinate movement or pitch movement
                                 (left vertical, green wire, 0-2047)
                a/d channel 4 -- position and orientation control change
                                 (left switch, white wire,
                                 position:>1400, orientation:<1400)
        This program uses a function "ttrtran()" which transforms
        joint angles (theta) to robot command parameters(ja).  The re-
        lation between 'theta' and 'ja' is as following:
                ja1 = -40 * theta1                      (-6000<ja1<6000)
                ja2 = 40 * theta2 - 1400                (-2600<ja2<2600)
                ja3 = 40 * theta3 + 1800                (-1800<ja3<1800)
                ja4 = 13.333 * (theta4 - theta5) + 200
                                                  (-2400<ja4-ja5<2400)
                ja5 = 13.333 * (-theta5 - theta4) + 200
                                                  (-4800<ja4+ja5<4800)
        The initial position is as following:
                theta1 = 0 (deg),       ja1 = 0
                theta2 = 35 (deg),      ja2 = 0
```

```
                theta3 = -45 (deg),     ja3 = 0
                theta4 = 0 (deg),       ja4 = +200
                theta5 = 0 (deg),       ja5 = +200

Sub function
                jtctran(phj,ph,roll,pitch,hi)
                caljoy(freq,jmid)
                getjoy(freq,phj,jmid)
                deadband(ia)
                ttrtran(theta,ja)
                mrob(ma)
                inv_kinem(ph,roll,pitch,theta)

Author
        Munehisa Takeda
Date
        Jan. 28, 1988

-------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "ad.c"

#define  LB 250.0       /* length of base */
#define  LS 220.0       /* length of soulder */
#define  LE 160.0       /* length of elbow */
#define  LH 202.0       /* length of hand */
#define  LS2 48400.0    /* square LS */
#define  LE2 25600.0    /* square LE */
#define  LSLE 35200.0   /* multiple LE LS */
#define  LR  1.375      /* LS over LE */
#define  RTD 57.29578   /* scaling parameter (rad to degree) */
#define  DBAND 100.0    /* deadband of joystick reading */
#define  DIS   40.      /* distance from initial to target position */
#define  ANG   0.261799 /*max initial angle from target angle (15 deg)*/

int jmid[3];                    /* joystick mid point */
double kp = 0.01,ka = 0.0025/RTD;                       /* gains */
long t1,t2,t3;                  /* starting & close & open time (system
                                   timer) */
long itime = 0;
long it;                        /* software timer */
int ma5old;                     /* previous hand information */

FILE *fp1;
FILE *printer;

extern double c[];
double tph[11][4];              /* target position & pitch */
double iph[11][4];              /* start position & pitch */

main()
{
```

```
    extern long time();
    extern long decompose();
    static double theta[5] = {0,35./RTD,-45./RTD,0,0};   /*joint angle*/
    static double ph[3] = {0,536.7,63.3};        /* hand position */
    static int phj[6];                    /* joystic digital value */
    double roll = 0.,pitch = -10./RTD;           /* roll and pitch */
    double freq;
    static double otheta[5];
    static double oldph[3];
    double oroll, opitch;
    static int ja[5];                     /* joint angle for robot
                                             command */
    static int oja[5];                    /* old joint angle for robot
                                             command */
    static int ma[6];                     /* robot acctuation parameter */
    char fname[12];                       /* data file name */
    int ij;
    int i;                                /* iteration value */
    int ch;
    int hi = 1;                           /* hand information */
    int viewmode = 0;
    int visualmode = 0;
    double x1, y1, x2, y2;
    char yn[3];

/*   target position setting   */
    tph[0][0] = 0.;
    tph[0][1] = 450.;
    tph[0][2] = 250.;
    tph[0][3] = -4. / RTD;
    tph[1][0] = 0.;
    tph[1][1] = 300.;
    tph[1][2] = 50.;
    tph[1][3] = -64. / RTD;
    tph[2][0] = 0.;
    tph[2][1] = 500.;
    tph[2][2] = -150.;
    tph[2][3] = -4. / RTD;
    tph[3][0] = 0.;
    tph[3][1] = 250.;
    tph[3][2] = -80.;
    tph[3][3] = -94. / RTD;
    tph[4][0] = 0.;
    tph[4][1] = 500.;
    tph[4][2] = 0.;
    tph[4][3] = -34. / RTD;
    tph[5][0] = 0.;
    tph[5][1] = 350.;
    tph[5][2] = -150.;
    tph[5][3] = -94. / RTD;
    tph[6][0] = 0.;
    tph[6][1] = 500.;
    tph[6][2] = -150.;
    tph[6][3] = -34. / RTD;
```

```
    tph[7][0] = 0.;
    tph[7][1] = 400.;
    tph[7][2] = -50.;
    tph[7][3] = -64. / RTD;
    tph[8][0] = 0.;
    tph[8][1] = 550.;
    tph[8][2] = 50.;
    tph[8][3] = -4. / RTD;
    tph[9][0] = 0.;
    tph[9][1] = 400.;
    tph[9][2] = -150.;
    tph[9][3] = -64. / RTD;
    tph[10][0] = 0.;
    tph[10][1] = 400.;
    tph[10][2] = 150.;
    tph[10][3] = -34. / RTD;

/*   initial position setting   */
    iph[0][0] = 0.;
    iph[0][1] = tph[0][1] - (DIS * cos(30. / RTD));
    iph[0][2] = tph[0][2] + (DIS * sin(30. / RTD));
    iph[0][3] = tph[0][3] + ANG;
    iph[1][0] = 0.;
    iph[1][1] = tph[1][1] + (DIS * sin(30. / RTD));
    iph[1][2] = tph[1][2] + (DIS * cos(30. / RTD));
    iph[1][3] = tph[1][3] + (ANG / 2.);
    iph[2][0] = 0.;
    iph[2][1] = tph[2][1] - (DIS * sin(30. / RTD));
    iph[2][2] = tph[2][2] - (DIS * cos(30. / RTD));
    iph[2][3] = tph[2][3];
    iph[3][0] = 0.;
    iph[3][1] = tph[3][1] + (DIS * sin(30. / RTD));
    iph[3][2] = tph[3][2] + (DIS * cos(30. / RTD));
    iph[3][3] = tph[3][3];
    iph[4][0] = 0.;
    iph[4][1] = tph[4][1] - (DIS * cos(30. / RTD));
    iph[4][2] = tph[4][2] + (DIS * sin(30. / RTD));
    iph[4][3] = tph[4][3] + (ANG / 2.);
    iph[5][0] = 0.;
    iph[5][1] = tph[5][1] - (DIS * sin(30. / RTD));
    iph[5][2] = tph[5][2] + (DIS * cos(30. / RTD));
    iph[5][3] = tph[5][3] + ANG;
    iph[6][0] = 0.;
    iph[6][1] = tph[6][1] - DIS;
    iph[6][2] = tph[6][2];
    iph[6][3] = tph[6][3] - (ANG / 2.);
    iph[7][0] = 0.;
    iph[7][1] = tph[7][1] - DIS;
    iph[7][2] = tph[7][2];
    iph[7][3] = tph[7][3] - (ANG / 2.);
    iph[8][0] = 0.;
    iph[8][1] = tph[8][1] - DIS;
    iph[8][2] = tph[8][2];
    iph[8][3] = tph[8][3] - ANG;
```

```
    iph[9][0] = 0.;
    iph[9][1] = tph[9][1] - (DIS * sin(30. / RTD));
    iph[9][2] = tph[9][2] + (DIS * cos(30. / RTD));
    iph[9][3] = tph[9][3];
    iph[10][0] = 0.;
    iph[10][1] = tph[10][1] - (DIS * sin(30. / RTD));
    iph[10][2] = tph[10][2] + (DIS * cos(30. / RTD));
    iph[10][3] = tph[10][3];

printf("direct view or camera view (0 or 1): ");
scanf("%d", &viewmode);
if (viewmode == 1) {
        printf("without or with visual enhancements (0 or 1): ");
        scanf("%d", &visualmode);
}


    printf(" input gains kp and ka (0.01 0.0025) = ");
    scanf("%lf %lf",&kp,&ka);
    ka /= RTD;


    printf(" input sampling frequency = ");
    scanf("%lf",&freq);

    printf(" data file name (j*******.dat) = ");
    scanf("%s", fname);

    printer = fopen("prn","w");  /* printer open for robot output */
    fp1 = fopen(fname, "w");     /* file open for measurement */

    printf(" start joystick calibration\n ");
    caljoy(&freq,jmid);
    printf(" end joystick calibration\n ");

    if(viewmode == 1) {
         init_video();
         sample_picture();
         camcal(c);
    }

    printf("the robot is going to move to the initial position.\n");
    printf("make sure that the work space is clear\n");
    printf("ready? ");
    scanf("%s", &yn);
    printf("are you sure? ");
    scanf("%s", &yn);

    fprintf(printer,"nt\n");              /* move nest position */
    fflush(printer);
    fprintf(printer,"SP 9\n");            /* maximum speed set */
    fflush(printer);
    fprintf(printer,"MI -6000,-2600,1800,1400,-1000,0\n");
                                          /* move to initial position */
```

```
   fflush(printer);
   fprintf(printer,"HO\n");     /* move initial position */
   fflush(printer);

/* timer setting (t1: start time) */
   t1 = decompose(time());
   itime = 0;

 for(ij = 0; ij < 11; ij++)
 {
   /* save values */
   for(i=0; i<3; i++)
        oldph[i] = ph[i];
   for(i=0; i<5; i++)
        otheta[i] = theta[i];
   oroll = roll;
   opitch = pitch;

   ttrtran(otheta,oja); /* change otheta to oja */

   for(i = 0; i < 3; i++)
      ph[i] = tph[ij][i];
   pitch = tph[ij][3];

   if (inv_kinem(ph,&roll,&pitch,theta) == 0) {
        printf("%c",07);                /* sound beep */
        printf(" wrong target position setting !!!\n ");
        goto ex;
   }

        ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
        for(i = 0; i < 5; i++)
           ma[i] = ja[i] - oja[i];
        ma[5] = hi;

        mrob(ma);               /* move robot */

   printf("set the objects and target box. Then remove target for
   moving initial position\n");
   printf("ready? ");
   scanf("%s", &yn);

   /* save values */
   for(i=0; i<3; i++)
        oldph[i] = ph[i];
   for(i=0; i<5; i++)
        otheta[i] = theta[i];
   oroll = roll;
   opitch = pitch;

   ttrtran(otheta,oja); /* change otheta to oja */
```

```
   for(i = 0; i < 3; i++)
      ph[i] = iph[ij][i];
   pitch = iph[ij][3];

   if (inv_kinem(ph,&roll,&pitch,theta) == 0) {
        printf("%c",07);                /* sound beep */
        printf(" wrong start position setting !!!\n ");
        goto ex;
   }

        ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
        for(i = 0; i < 5; i++)
           ma[i] = ja[i] - oja[i];
        ma[5] = hi;

        mrob(ma);               /* move robot */

   printf("set the objects again.\n");
   printf("did you pick the object?\n");
   printf("pick the object by using swich on the joystick.\n");
   printf("after grasping the object , hit the 'p' key.\n");
   printf("ready? ");
   scanf("%s", &yn);
   for(;;)
   {
        ch = bdos(0x06,0x00ff) & 0x00ff;
        switch(ch)
        {
           case 'p':
              goto start;
           default:
              break;
        }
        getjoy(&freq,phj,jmid);

        /* save values */
        for(i=0; i<3; i++)
                oldph[i] = ph[i];
        for(i=0; i<5; i++)
                otheta[i] = theta[i];
        oroll = roll;
        opitch = pitch;

        ttrtran(otheta,oja);    /* change otheta to oja */

        jtctran(phj,ph,&roll,&pitch,&hi);

        if (inv_kinem(ph,&roll,&pitch,theta) == 0) {
                printf("%c",07);                /* sound beep */
                for(i=0; i<3; i++)              /*restore old values*/
                        ph[i] = oldph[i];
                for(i=0; i<5; i++)
```

```
                        theta[i] = otheta[i];
                roll = oroll;
                pitch = opitch;
        }

        ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
        for(i = 0; i < 5; i++)
           ma[i] = ja[i] - oja[i];
        ma[5] = hi;

        mrob(ma);               /* move robot */
   }    /* end of infinite loop */

start:
   printf("start experiment\n");
   printf("ready? ");
   scanf("%s", &yn);

/* timer setting (t1: start time) */
   t1 = decompose(time());
   itime = 0;

for(;;)
   {
        ch = bdos(0x06,0x00ff) & 0x00ff;
        switch(ch)
        {
           case 'e':
              printf("the robot is going to move to the initial
              position.\n");
              printf("make sure that the work space is clear\n");
              printf("ready? ");
              scanf("%s", &yn);
              fprintf(printer,"OG\n");  /* move initial position */
              fflush(printer);
              fprintf(printer,"MI 0,0,0,-500,-500,0\n");
                                        /* move initial position */
              fflush(printer);
              fprintf(printer,"nt\n");          /* move nest position */
              fflush(printer);
              fclose (printer);
              fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],
              ja[2],ja[3],ja[4],1010); /* write to data file */
              fclose(fp1);
              exit(1);
              break;
           case 'n':
              fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],
              ja[2],ja[3],ja[4],1010); /* write to data file */
              fclose(fp1);
              printf(" new data file name (j*******.dat) = ");
              scanf("%s",fname);
```

```
            fp1 = fopen(fname, "w");  /* file open for measuring */
            printf("the robot is going to move to the next target
            position.\n");
            printf("make sure that the work space is clear\n");
            printf("ready? ");
            scanf("%s", &yn);
            goto next;
         default:
            break;
      }

      getjoy(&freq,phj,jmid);

      /* save values */
      for(i=0; i<3; i++)
            oldph[i] = ph[i];
      for(i=0; i<5; i++)
            otheta[i] = theta[i];
      oroll = roll;
      opitch = pitch;

      ttrtran(otheta,oja);    /* change otheta to oja */

      jtctran(phj,ph,&roll,&pitch,&hi);

      if (inv_kinem(ph,&roll,&pitch,theta) == 0) {
            printf("%c",07);                /* sound beep */
            for(i=0; i<3; i++)              /* restore old values */
                  ph[i] = oldph[i];
            for(i=0; i<5; i++)
                  theta[i] = otheta[i];
            roll = oroll;
            pitch = opitch;
      }

      ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
      for(i = 0; i < 5; i++)
         ma[i] = ja[i] - oja[i];
      ma[5] = hi;

      mrob(ma);               /* move robot */

if(viewmode == 1) {
      if (visualmode == 1)
            projection(ph,c,&x1,&y1,&x2,&y2);

      sample_picture();

      if (visualmode == 1)
            line(x1, y1, x2, y2);
}
```

```
        fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],ja[2],
        ja[3],ja[4],hi); /* write to data file */

        itime++;
        it = (long)(itime * 1000 / freq);

    }   /* end of infinite for loop */

next:

 }      /* end of ij for loop */

ex:
  printf("finish the experiment!!!\n");
  printf("the robot is going to move to the initial position.\n");
  printf("make sure that the work space is clear\n");
  printf("ready? ");
  scanf("%s", &yn);
  fprintf(printer,"OG\n");      /* move initial position */
  fflush(printer);
  fprintf(printer,"MI 0,0,0,-500,-500,0\n");/* move initial position */
  fflush(printer);
  fprintf(printer,"nt\n");              /* move nest position */
  fflush(printer);
  fclose (printer);
  fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],ja[2],ja[3],
  ja[4],1010); /* write to data file */
  fclose(fp1);
  exit(1);
}       /* end of main */

/*
timer function (return milisecond)
*/
long decompose(t)
long t;
{
long ms;
   ms = (t >> 24) & 0xFF;
   ms = ms * 60 + ((t >> 16) & 0xFF);
   ms = ms * 60 + ((t >> 8) & 0xFF);
   ms = ms * 100 + (t & 0xFF);
   return(ms * 10);
}


/*
joystick value to cartesian value transformation function
*/

jtctran(phj,ph,roll,pitch,hi)
int phj[6];
double ph[3];
double *roll,*pitch;
```

```
int *hi;
{
     int i;

     for(i=0; i < 3; i++)
          ph[i] += kp * phj[i];

/*   limit range check for position   */
     ph[0] = (ph[0] > 582.) ? 582. : ph[0];
     ph[0] = (ph[0] < -582.) ? -582. : ph[0];
     ph[1] = (ph[1] > 582.) ? 582. : ph[1];
     ph[1] = (ph[1] < -504.) ? -504. : ph[1];
     ph[2] = (ph[2] > 582.) ? 582. : ph[2];
     ph[2] = (ph[2] < -467.) ? -467. : ph[2];

/*   update orientation */
     *roll += ka * phj[3];
     *pitch += ka * phj[4];

/*   limit the range of the orientation */
     *roll = (*roll > 3.14159) ? 3.14159 : *roll;
     *roll = (*roll < -3.14159) ? -3.14159 : *roll;
     *pitch = (*pitch > 3.14159) ? 3.14159 : *pitch;
     *pitch = (*pitch < -3.14159) ? -3.14159 : *pitch;

/*   hand information calculation   */
     if(phj[5] > 850)
        *hi = 0;
     else
        *hi = 1;
}


/*
calibration joystick
*/

caljoy(freq,jmid)
double *freq;
int jmid[3];
{
     settimer(*freq);
     startadc();

        jmid[0] = t_adc(0);     /* x coordinate joystick middle value */
        jmid[1] = adc(1);       /* y coordinate joystick middle value */
        jmid[2] = adc(3);       /* z coordinate joystick middle value */

      stopadc();
}

/*
joystick value get function
*/
```

```
getjoy(freq,phj,jmid)
double *freq;
int jmid[3];
int phj[6];
{
     settimer(*freq);
     startadc();

     if(t_adc(4) > 1400)
     {                                          /* y,z,pitch control*/
        phj[0] = 0;                             /* x coordinate value */
        phj[1] = -deadband(adc(0) - jmid[0]);   /* y coordinate value */
        phj[2] = deadband(adc(1) - jmid[1]);    /* z coordinate value */
        phj[3] = 0;                             /* roll value */
        phj[4] = deadband(adc(3) - jmid[2]);    /* pitch value */
      }
      else
      {                                         /* y,z,pitch control */
        phj[0] = 0;                             /* x coordinate value */
        phj[1] = -deadband(adc(0) - jmid[0]);   /* y coordinate value */
        phj[2] = deadband(adc(1) - jmid[1]);    /* z coordinate value */
        phj[3] = 0;                             /* roll value */
        phj[4] = deadband(adc(3) - jmid[2]);    /* pitch value */
      }
      phj[5] = adc(2);                  /* hand open close */

      stopadc();
}


/*
deadband calculation
*/

deadband(ia)
int ia;
{
   int ix;

   if(ia > DBAND)
      ix = ia - DBAND;
   else if(ia < -DBAND)
      ix = ia + DBAND;
   else
      ix = 0;

   return(ix);
}


/*
theta to robot command parameter transformation function
*/
```

```
ttrtran(theta,ja)
double theta[5];
int ja[5];
{

/* calculate robot command parameter */
   ja[0] = (int)(-40 * RTD * theta[0]);
   ja[1] = (int)(40 * RTD * theta[1]) - 1400;
   ja[2] = (int)(40 * RTD * theta[2]) + 1800;
   ja[3] = (int)(13.333 * RTD * (theta[3] - theta[4])) + 200;
   ja[4] = (int)(13.333 * RTD * (-theta[4] - theta[3])) + 200;

/*   limit range check for robot command parameter   */
   ja[0] = (ja[0] > 6000) ? 6000 : ja[0];
   ja[0] = (ja[0] < -6000) ? -6000 : ja[0];
   ja[1] = (ja[1] > 2600) ? 2600 : ja[1];
   ja[1] = (ja[1] < -2600) ? -2600 : ja[1];
   ja[2] = (ja[2] > 1800) ? 1800 : ja[2];
   ja[2] = (ja[2] < -1800) ? -1800 : ja[2];
   if(((ja[3] - ja[4]) < -2400) ¦¦ ((ja[3] - ja[4]) > 2400))
        exit(1);
   if(((ja[3] + ja[4]) < -4800) ¦¦ ((ja[3] + ja[4]) > 4800))
        exit(1);

}


mrob(ma)
int ma[6];
{
   fprintf(printer, "MI %d,%d,%d,%d,%d,0\n",ma[0],ma[1],ma[2],ma[3],
   ma[4]);
   fflush(printer);
   t2 = decompose(time());
   if(ma[5] != ma5old)
   {
      if(ma[5] == 1)
      {
         fprintf(printer,"go\n");
         fflush(printer);
         t3 = decompose(time());
         printf(" hand open time = %ld %ld\n",(t3-t1),it);
      }
      else
      {
         fprintf(printer,"gc\n");
         fflush(printer);
         t3 = decompose(time());
         printf(" hand close time = %ld %ld\n",(t3-t1),it);
      }
   }
   ma5old = ma[5];
}
```

```
/*
   inverse kinematic function
*/
inv_kinem( ph,roll,pitch,theta)
double ph[3];                    /* hand position */
double *pitch,*roll;             /* pitch & roll */
double theta[5];                 /* joint angle */
{
   double a[3];                  /* hand approach vector */
   double pw[3];                 /* wrist position */
   double a_norm;                /* nomarizing of approach vector */
   double d2,dd;                 /* supplement variable */
   double c1,c3,c23;             /* cos theta value */
   double s1,s3,s23;             /* sin theta value */
   double w11,w13,w22,w32;       /* element of R4R5 matrix */
   double lr;                    /* soulder & elbow length ratio */
   int i;                        /* iteration valiable */

/*   calculate approach vector   */
      a_norm = cos(*pitch) / sqrt(ph[0] * ph[0] + ph[1] * ph[1]);
      a[0] = ph[0] * a_norm;
      a[1] = ph[1] * a_norm;
      a[2] = sin(*pitch);
/*
 printf(" a vector = %f %f %f\n",a[0],a[1],a[2]);
 printf(" pitch roll = %f %f\n",*pitch,*roll);
*/
/*   calculate wrist position   */
   pw[0] = ph[0] - LH * a[0];
   pw[1] = ph[1] - LH * a[1];
   pw[2] = ph[2] - LH * a[2];

/*   calculate theta 1   */
   if (pw[1] < 0.) {
/*      printf(" outside the range of theta 1 \n");
*/
      return(0);
   }
   theta[0] = atan2(-pw[0],pw[1]);      /* pi/2 < theta[0] < pi/2 */
   if (theta[0] > 3.14159265)   /* compensate of calculation error */
      theta[0] = 0.;

/*   calculate theta 3   */
   d2 = pw[0] * pw[0] + pw[1] * pw[1] + pw[2] * pw[2];
   c3 = (d2 - LS2 - LE2) / (2 * LSLE);

   if ( (c3 > -1) && (c3 <= 1) )
   {
      s3 = -sqrt(1.-c3*c3);             /* elbow up */
      theta[2] = atan2(s3,c3);

      if((theta[2] < -1.57079) || (theta[2] > 0.)) {
```

```
  /*       printf(" outside the range of theta3 \n");
*/
         return(0);
      }
   }
   else
   {
/*        printf(" outside the range of c3 \n");
*/
        return(0);
   }

/*   calculate theta 2   */

   s1 = sin(theta[0]);
   c1 = cos(theta[0]);

   dd = -s1 * pw[0] + c1 * pw[1];

   theta[1] = atan2(((c3+LR)*pw[2] - s3*dd),(s3*pw[2] + (c3+LR)*dd));

   if((theta[1] < -0.5236) || (theta[1] > 1.7453))
   {
/*      printf(" outside the range of theta2 \n");
*/
      return(0);
   }

/*   calculate theta 4   */

   c23 = cos(theta[1]+theta[2]);
   s23 = sin(theta[1]+theta[2]);

   w22 = -s1 * c23 * a[0]  +  c1 * c23 * a[1]  +  s23 * a[2];
   w32 = s1 * s23 * a[0]  -  c1 * s23 * a[1]  +  c23 * a[2];

   theta[3] = atan2(w32,w22);

   if((theta[3] < -1.5707) || (theta[3] > 1.5707))
   {
/*      printf(" outside the range of theta4 \n");
*/
      return(0);
   }

/*   calculate theta 5   */

   theta[4] = *roll;

   if((theta[4] < -3.14159) || (theta[4] > 3.14159))
   {
/*      printf(" outside the range of theta5 \n");
*/
      return(0);
```

```
/*-----------------------------------------------------------------

File
        2robjae.c  - operate robot using joystick and measure joint
                     angle for joint angle control (2-D version
                     experiment, 10 target position)

Synopsis
        #include <stdio.h>
        #include <math.h>
        #include <conio.h>
        #include "ad.c"

        lcl 2robjae
        linkml 2robjae draw display atime

Description
        This program controls joint angles (shoulder elbow wrist bend)
        of a movemaster II and stores time and joystick angles (robot
        command parameters) in file every sampling time.
        This program uses a function "jtttran()" which transforms
        joystick signal to joint angles.  The joystick signal is
        converted to joint angle by multipling gains.  The joystick
        signal is connected to a/d converter.
        The right side joystick is used for shoulder and elbow control.
        The left side joystick (vertical) is used for wrist control. In
        addition to the joystick handle, each joystick box has a swich
        on the top face. The switch of the right side joystickh is used
        for hand open and close control (up and middle position is
        assigned to close and down position is assigned to open). The
        switch of the left side joystick is not used in this program.
        The a/d converter configuration is as following:
                a/d channel 0 -- x coordinate movement or roll movement
                                 (right horizontal, red wire, 0-2047)
                a/d channel 1 -- y coordinate movement (right vertical,
                                 green wire, 0-2047)
                a/d channel 2 -- hand open close (right switch, white
                                 wire, open:0 close:>900)
                a/d channel 3 -- z coordinate movement or pitch movement
                                 (left vertical, green wire, 0-2047)
                a/d channel 4 -- position and orientation control change
                                 (left switch, white wire,
                                  position:>1400, orientation:<1400)
        This program uses a function "ttrtran()" which transforms
        joint angles (theta) to robot command parameters(ja).  The re-
        lation between 'theta' and 'ja' is as following:
                ja1 = -40 * theta1                  (-6000<ja1<6000)
                ja2 = 40 * theta2 - 1400            (-2600<ja2<2600)
                ja3 = 40 * theta3 + 1800            (-1800<ja3<1800)
                ja4 = 13.333 * (theta4 - theta5) + 200
                                                  (-2400<ja4-ja5<2400)
                ja5 = 13.333 * (-theta5 - theta4) + 200
                                                  (-4800<ja4+ja5<4800)
        The initial position is as following:
```

```
                theta1 = 0 (deg),       ja1 = 0
                theta2 = 35 (deg),      ja2 = 0
                theta3 = -45 (deg),     ja3 = 0
                theta4 = 0 (deg),       ja4 = +200
                theta5 = 0 (deg),       ja5 = +200

Sub function
                jtttran(phj,theta,&hi);
                caljoy(freq,jmid)
                getjoy(freq,phj,jmid)
                deadband(ia)
                ttrtran(theta,ja)
                mrob(ma)
                inv_kinem(ph,roll,pitch,theta)--for initial &
                                                target setting

Author
        Munehisa Takeda
Date
        Jan. 28, 1988

-----------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "ad.c"

#define  LB 250.0       /* length of base */
#define  LS 220.0       /* length of soulder */
#define  LE 160.0       /* length of elbow */
#define  LH 202.0       /* length of hand */
#define  LS2 48400.0    /* square LS */
#define  LE2 25600.0    /* square LE */
#define  LSLE 35200.0   /* multiple LE LS */
#define  LR  1.375      /* LS over LE */
#define  RTD 57.29578   /* scaling parameter (rad to degree) */
#define  DBAND 100.0    /* deadband of joystick reading */
#define  DIS   40.      /* distance from initial to target position */
#define  ANG   0.261799 /* max initial angle from target angle
                           (15 deg) */

int jmid[3];                    /* joystick mid point */
double kp = 0.01,ka = 0.005/RTD;                        /* gains */
long t1,t2,t3;          /*starting & close & open time (system timer)*/
long itime = 0;
long it;                        /* software timer */
int ma5old;                     /* previous hand information */

FILE *fp1;
FILE *printer;

extern double c[];
double tph[11][4];              /* target position & pitch */
double iph[11][4];              /* start position & pitch */
```

```
main()
{
   extern long time();
   extern long decompose();
   static double theta[5] = {0,35./RTD,-45./RTD,0,0};   /*joint angle*/
   static double ph[3] = {0,536.7,63.3};          /* hand position */
   static int phj[6];                    /* joystic digital value */
   double roll = 0.,pitch = -10./RTD;             /* roll and pitch */
   double freq;
   static double otheta[5];
   static double oldph[3];
   double oroll, opitch;
   static int ja[5];             /* joint angle for robot command */
   static int oja[5];            /* old joint angle for robot command */
   static int ma[6];             /* robot acctuation parameter */
   char fname[12];               /* data file name */
   int i;                        /* iteration value */
   int ij;
   int ch;
   int hi = 1;                           /* hand information */
   int viewmode = 0;
   int visualmode = 0;
   double x1, y1, x2, y2;
   char yn[3];

/*   target position setting   */
   tph[0][0] = 0.;
   tph[0][1] = 450.;
   tph[0][2] = 250.;
   tph[0][3] = -4. / RTD;
   tph[1][0] = 0.;
   tph[1][1] = 300.;
   tph[1][2] = 50.;
   tph[1][3] = -64. / RTD;
   tph[2][0] = 0.;
   tph[2][1] = 500.;
   tph[2][2] = -150.;
   tph[2][3] = -4. / RTD;
   tph[3][0] = 0.;
   tph[3][1] = 250.;
   tph[3][2] = -80.;
   tph[3][3] = -94. / RTD;
   tph[4][0] = 0.;
   tph[4][1] = 500.;
   tph[4][2] = 0.;
   tph[4][3] = -34. / RTD;
   tph[5][0] = 0.;
   tph[5][1] = 350.;
   tph[5][2] = -150.;
   tph[5][3] = -94. / RTD;
   tph[6][0] = 0.;
   tph[6][1] = 500.;
```

```
    tph[6][2] = -150.;
    tph[6][3] = -34. / RTD;
    tph[7][0] = 0.;
    tph[7][1] = 400.;
    tph[7][2] = -50.;
    tph[7][3] = -64. / RTD;
    tph[8][0] = 0.;
    tph[8][1] = 550.;
    tph[8][2] = 50.;
    tph[8][3] = -4. / RTD;
    tph[9][0] = 0.;
    tph[9][1] = 400.;
    tph[9][2] = -150.;
    tph[9][3] = -64. / RTD;
    tph[10][0] = 0.;
    tph[10][1] = 400.;
    tph[10][2] = 150.;
    tph[10][3] = -34. / RTD;

/*   initial position setting   */
    iph[0][0] = 0.;
    iph[0][1] = tph[0][1] - (DIS * cos(30. / RTD));
    iph[0][2] = tph[0][2] + (DIS * sin(30. / RTD));
    iph[0][3] = tph[0][3] + ANG;
    iph[1][0] = 0.;
    iph[1][1] = tph[1][1] + (DIS * sin(30. / RTD));
    iph[1][2] = tph[1][2] + (DIS * cos(30. / RTD));
    iph[1][3] = tph[1][3] + (ANG / 2.);
    iph[2][0] = 0.;
    iph[2][1] = tph[2][1] - (DIS * sin(30. / RTD));
    iph[2][2] = tph[2][2] - (DIS * cos(30. / RTD));
    iph[2][3] = tph[2][3];
    iph[3][0] = 0.;
    iph[3][1] = tph[3][1] + (DIS * sin(30. / RTD));
    iph[3][2] = tph[3][2] + (DIS * cos(30. / RTD));
    iph[3][3] = tph[3][3];
    iph[4][0] = 0.;
    iph[4][1] = tph[4][1] - (DIS * cos(30. / RTD));
    iph[4][2] = tph[4][2] + (DIS * sin(30. / RTD));
    iph[4][3] = tph[4][3] + (ANG / 2.);
    iph[5][0] = 0.;
    iph[5][1] = tph[5][1] - (DIS * sin(30. / RTD));
    iph[5][2] = tph[5][2] + (DIS * cos(30. / RTD));
    iph[5][3] = tph[5][3] + ANG;
    iph[6][0] = 0.;
    iph[6][1] = tph[6][1] - DIS;
    iph[6][2] = tph[6][2];
    iph[6][3] = tph[6][3] - (ANG / 2.);
    iph[7][0] = 0.;
    iph[7][1] = tph[7][1] - DIS;
    iph[7][2] = tph[7][2];
    iph[7][3] = tph[7][3] - (ANG / 2.);
    iph[8][0] = 0.;
    iph[8][1] = tph[8][1] - DIS;
```

```
    iph[8][2] = tph[8][2];
    iph[8][3] = tph[8][3] - ANG;
    iph[9][0] = 0.;
    iph[9][1] = tph[9][1] - (DIS * sin(30. / RTD));
    iph[9][2] = tph[9][2] + (DIS * cos(30. / RTD));
    iph[9][3] = tph[9][3];
    iph[10][0] = 0.;
    iph[10][1] = tph[10][1] - (DIS * sin(30. / RTD));
    iph[10][2] = tph[10][2] + (DIS * cos(30. / RTD));
    iph[10][3] = tph[10][3];

printf("direct view or camera view (0 or 1): ");
scanf("%d", &viewmode);
if (viewmode == 1) {
        printf("without or with visual enhancements (0 or 1): ");
        scanf("%d", &visualmode);
}

    printf(" input gain ka (0.005) = ");
    scanf("%lf",&ka);
    ka /= RTD;

    printf(" input sampling frequency = ");
    scanf("%lf",&freq);

/*   printf(" input hand position = ");
    scanf("%lf %lf %lf",&ph[0],&ph[1],&ph[2]);
*/
    printf(" data file name (j*******.dat) = ");
    scanf("%s", fname);

    printer = fopen("prn","w");  /* printer open for robot output */
    fp1 = fopen(fname, "w");     /* file open for measurement */

    printf(" start joystick calibration\n ");
    caljoy(&freq,jmid);
    printf(" end joystick calibration\n ");

    if(viewmode == 1) {
         init_video();
         sample_picture();
         camcal(c);
    }

    printf("the robot is going to move to the initial position.\n");
    printf("make sure that the work space is clear\n");
    printf("ready? ");
    scanf("%s", &yn);
    printf("are you sure? ");
    scanf("%s", &yn);

    fprintf(printer,"nt\n");             /* move nest position */
    fflush(printer);
    fprintf(printer,"SP 9\n");           /* maximum speed set */
```

```
   fflush(printer);
   fprintf(printer,"MI -6000,-2600,1800,1400,-1000,0\n");
                                        /* move initial position */
   fflush(printer);
   fprintf(printer,"HO\n");     /* move initial position */
   fflush(printer);

/* timer setting (t1: start time) */
   t1 = decompose(time());
   itime = 0;

 for(ij = 0; ij < 11; ij++)
 {
   /* save values */
   for(i=0; i<3; i++)
        oldph[i] = ph[i];
   for(i=0; i<5; i++)
        otheta[i] = theta[i];
   oroll = roll;
   opitch = pitch;

   ttrtran(otheta,oja); /* change otheta to oja */

   for(i = 0; i < 3; i++)
      ph[i] = tph[ij][i];
   pitch = tph[ij][3];

   if (inv_kinem(ph,&roll,&pitch,theta) == 0) {
        printf("%c",07);                /* sound beep */
        printf(" wrong target position setting !!!\n ");
        goto ex;
   }

        ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
        for(i = 0; i < 5; i++)
           ma[i] = ja[i] - oja[i];
        ma[5] = hi;

        mrob(ma);               /* move robot */

   printf("set the objects and target box. Then remove target for
   moving initial position\n");
   printf("ready? ");
   scanf("%s", &yn);

   /* save values */
   for(i=0; i<3; i++)
        oldph[i] = ph[i];
   for(i=0; i<5; i++)
        otheta[i] = theta[i];
   oroll = roll;
   opitch = pitch;
```

```
   ttrtran(otheta,oja); /* change otheta to oja */

   for(i = 0; i < 3; i++)
      ph[i] = iph[ij][i];
   pitch = iph[ij][3];

   if (inv_kinem(ph,&roll,&pitch,theta) == 0) {
        printf("%c",07);                /* sound beep */
        printf(" wrong start position setting !!!\n ");
        goto ex;
   }

        ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
        for(i = 0; i < 5; i++)
           ma[i] = ja[i] - oja[i];
        ma[5] = hi;

        mrob(ma);               /* move robot */

   printf("set the objects again.\n");
   printf("did you pick the object?\n");
   printf("pick the object by using swich on the joystick.\n");
   printf("after grasping the object , hit the 'p' key.\n");
   printf("ready? ");
   scanf("%s", &yn);
   for(;;)
   {
        ch = bdos(0x06,0x00ff) & 0x00ff;
        switch(ch)
        {
           case 'p':
              goto start;
           default:
              break;
        }
        getjoy(&freq,phj,jmid);

        /* save values */
        for(i=0; i<5; i++)
                otheta[i] = theta[i];

        ttrtran(otheta,oja);    /* change otheta to oja */

        jtttran(phj,theta,&hi);

        ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
        for(i = 0; i < 5; i++)
           ma[i] = ja[i] - oja[i];
        ma[5] = hi;
```

```
        mrob(ma);                /* move robot */
    }   /* end of infinite loop */

start:
    printf("start experiment\n");
    printf("ready? ");
    scanf("%s", &yn);

/* timer setting (t1: start time) */
    t1 = decompose(time());
    itime = 0;

for(;;)
    {
        ch = bdos(0x06,0x00ff) & 0x00ff;
        switch(ch)
        {
            case 'e':
                printf("the robot is going to move to the initial
                position.\n");
                printf("make sure that the work space is clear\n");
                printf("ready? ");
                scanf("%s", &yn);
                fprintf(printer,"OG\n");  /* move initial position */
                fflush(printer);
                fprintf(printer,"MI 0,0,0,-500,-500,0\n");
                                          /* move initial position */
                fflush(printer);
                fprintf(printer,"nt\n");  /* move nest position */
                fflush(printer);
                fclose (printer);
                fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],
                ja[2],ja[3],ja[4],1010); /* write to data file */
                fclose(fp1);
                exit(1);
                break;
            case 'n':
                fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],
                ja[2],ja[3],ja[4],1010); /* write to data file */
                fclose(fp1);
                printf(" new data file name (j*******.dat) = ");
                scanf("%s",fname);
                fp1 = fopen(fname, "w");  /* file open for measuring */
                printf("the robot is going to move to the next target
                position.\n");
                printf("make sure that the work space is clear\n");
                printf("ready? ");
                scanf("%s", &yn);
                goto next;
            default:
                break;
        }
```

```
        getjoy(&freq,phj,jmid);

        /* save values */
        for(i=0; i<5; i++)
                otheta[i] = theta[i];

        ttrtran(otheta,oja);    /* change otheta to oja */

        jtttran(phj,theta,&hi);

        ttrtran(theta,ja);      /* change theta to ja */

/*   calculate robot acctuation parameter   */
        for(i = 0; i < 5; i++)
           ma[i] = ja[i] - oja[i];
        ma[5] = hi;

        mrob(ma);               /* move robot */

if(viewmode == 1) {
        if (visualmode == 1)
                projection(ph,c,&x1,&y1,&x2,&y2);

        sample_picture();

        if (visualmode == 1)
                line(x1, y1, x2, y2);
}

        fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],
        ja[2],ja[3],ja[4],hi); /* write to data file */

        itime++;
        it = (long)(itime * 1000 / freq);

    }   /* end of infinite for loop */

next:

 }      /* end of ij for loop */

ex:
  printf("finish the experiment!!!\n");
  printf("the robot is going to move to the initial position.\n");
  printf("make sure that the work space is clear\n");
  printf("ready? ");
  scanf("%s", &yn);
  fprintf(printer,"OG\n");      /* move initial position */
  fflush(printer);
  fprintf(printer,"MI 0,0,0,-500,-500,0\n");
                                /* move initial position */
  fflush(printer);
  fprintf(printer,"nt\n");              /* move nest position */
  fflush(printer);
```

```
  fclose (printer);
  fprintf(fp1,"%ld %d %d %d %d %d %d\n",(t2-t1),ja[0],ja[1],ja[2],
  ja[3],ja[4],1010); /* write to data file */
  fclose(fp1);
  exit(1);
}       /* end of main */

/*
timer function (return milisecond)
*/
long decompose(t)
long t;
{
long ms;
   ms = (t >> 24) & 0xFF;
   ms = ms * 60 + ((t >> 16) & 0xFF);
   ms = ms * 60 + ((t >> 8) & 0xFF);
   ms = ms * 100 + (t & 0xFF);
   return(ms * 10);
}


/*
joystick value to cartesian value transformation function
*/

jtttran(phj,theta,hi)
int phj[6];
double theta[5];
int *hi;
{
     int i;

     for(i=0; i < 5; i++)
          theta[i] += ka * phj[i];

/*   limit range check for position   */
     theta[0] = (theta[0] > (90. / RTD)) ? (90. / RTD) : theta[0];
     theta[0] = (theta[0] < -(90. / RTD)) ? -(90. / RTD) : theta[0];
     theta[1] = (theta[1] > (100. / RTD)) ? (100. / RTD) : theta[1];
     theta[1] = (theta[1] < -(30. / RTD)) ? -(30. / RTD) : theta[1];
     theta[2] = (theta[2] > (0. / RTD)) ? (0. / RTD) : theta[2];
     theta[2] = (theta[2] < -(90. / RTD)) ? -(90. / RTD) : theta[2];
     theta[3] = (theta[3] > (90. / RTD)) ? (90. / RTD) : theta[3];
     theta[3] = (theta[3] < -(90. / RTD)) ? -(90. / RTD) : theta[3];
     theta[4] = (theta[4] > (195. / RTD)) ? (195. / RTD) : theta[4];
     theta[4] = (theta[4] < -(165. / RTD)) ? -(165. / RTD) : theta[4];

/*   hand information calculation   */
     if(phj[5] > 850)
        *hi = 0;
     else
        *hi = 1;
}
```

```
/*
calibration joystick
*/

caljoy(freq,jmid)
double *freq;
int jmid[3];
{
     settimer(*freq);
     startadc();

        jmid[0] = t_adc(0);     /* x coordinate joystick middle value */
        jmid[1] = adc(1);       /* y coordinate joystick middle value */
        jmid[2] = adc(3);       /* z coordinate joystick middle value */

      stopadc();
}

/*
joystick value get function
*/

getjoy(freq,phj,jmid)
double *freq;
int jmid[3];
int phj[6];
{
     settimer(*freq);
     startadc();

     if(t_adc(4) > 1400)
     {                                          /* j2,3,4 control*/
        phj[0] = 0;                             /* j1 value */
        phj[1] = deadband(adc(0) - jmid[0]);    /* j2 value */
        phj[2] = deadband(adc(1) - jmid[1]);    /* j3 value */
        phj[3] = deadband(adc(3) - jmid[2]);    /* j4 value */
        phj[4] = 0;                             /* j5 value */
      }
      else
      {                                         /* j2,3,4 control */
        phj[0] = 0;                             /* j1 value */
        phj[1] = deadband(adc(0) - jmid[0]);    /* j2 value */
        phj[2] = deadband(adc(1) - jmid[1]);    /* j3 value */
        phj[3] = deadband(adc(3) - jmid[2]);    /* j4 value */
        phj[4] = 0;                             /* j5 value */
      }
      phj[5] = adc(2);                  /* hand open close */

      stopadc();
}
```

```
/*
deadband calculation
*/

deadband(ia)
int ia;
{
   int ix;

   if(ia > DBAND)
      ix = ia - DBAND;
   else if(ia < -DBAND)
      ix = ia + DBAND;
   else
      ix = 0;

   return(ix);
}


/*
theta to robot command parameter transformation function
*/

ttrtran(theta,ja)
double theta[5];
int ja[5];
{

/* calculate robot command parameter */
   ja[0] = (int)(-40 * RTD * theta[0]);
   ja[1] = (int)(40 * RTD * theta[1]) - 1400;
   ja[2] = (int)(40 * RTD * theta[2]) + 1800;
   ja[3] = (int)(13.333 * RTD * (theta[3] - theta[4])) + 200;
   ja[4] = (int)(13.333 * RTD * (-theta[4] - theta[3])) + 200;

/*   limit range check for robot command parameter   */
   ja[0] = (ja[0] > 6000) ? 6000 : ja[0];
   ja[0] = (ja[0] < -6000) ? -6000 : ja[0];
   ja[1] = (ja[1] > 2600) ? 2600 : ja[1];
   ja[1] = (ja[1] < -2600) ? -2600 : ja[1];
   ja[2] = (ja[2] > 1800) ? 1800 : ja[2];
   ja[2] = (ja[2] < -1800) ? -1800 : ja[2];
   if(((ja[3] - ja[4]) < -2400) || ((ja[3] - ja[4]) > 2400))
        exit(1);
   if(((ja[3] + ja[4]) < -4800) || ((ja[3] + ja[4]) > 4800))
        exit(1);

}


mrob(ma)
int ma[6];
{
```

```
   fprintf(printer, "MI %d,%d,%d,%d,%d,0\n",ma[0],ma[1],ma[2],ma[3],
   ma[4]);
   fflush(printer);
   t2 = decompose(time());
   if(ma[5] != ma5old)
   {
      if(ma[5] == 1)
      {
         fprintf(printer,"go\n");
         fflush(printer);
         t3 = decompose(time());
         printf(" hand open time = %ld %ld\n",(t3-t1),it);
      }
      else
      {
         fprintf(printer,"gc\n");
         fflush(printer);
         t3 = decompose(time());
         printf(" hand close time = %ld %ld\n",(t3-t1),it);
      }
   }
   ma5old = ma[5];
}


/*
   inverse kinematic function
*/
inv_kinem( ph,roll,pitch,theta)
double ph[3];                    /* hand position */
double *pitch,*roll;             /* pitch & roll */
double theta[5];                 /* joint angle */
{
   double a[3];                  /* hand approach vector */
   double pw[3];                 /* wrist position */
   double a_norm;                /* nomarizing of approach vector */
   double d2,dd;                 /* supplement variable */
   double c1,c3,c23;             /* cos theta value */
   double s1,s3,s23;             /* sin theta value */
   double w11,w13,w22,w32;       /* element of R4R5 matrix */
   double lr;                    /* soulder & elbow length ratio */
   int i;                        /* iteration valiable */

/*   calculate approach vector   */
      a_norm = cos(*pitch) / sqrt(ph[0] * ph[0] + ph[1] * ph[1]);
      a[0] = ph[0] * a_norm;
      a[1] = ph[1] * a_norm;
      a[2] = sin(*pitch);
/*
 printf(" a vector = %f %f %f\n",a[0],a[1],a[2]);
 printf(" pitch roll = %f %f\n",*pitch,*roll);
*/
/*   calculate wrist position   */
   pw[0] = ph[0] - LH * a[0];
```

```
   pw[1] = ph[1] - LH * a[1];
   pw[2] = ph[2] - LH * a[2];

/*   calculate theta 1   */
   if (pw[1] < 0.) {
/*      printf(" outside the range of theta 1 \n");
*/
      return(0);
   }
   theta[0] = atan2(-pw[0],pw[1]);      /* pi/2 < theta[0] < pi/2 */
   if (theta[0] > 3.14159265)           /* compensate of calculation
                                           error */
      theta[0] = 0.;

/*   calculate theta 3   */
   d2 = pw[0] * pw[0] + pw[1] * pw[1] + pw[2] * pw[2];
   c3 = (d2 - LS2 - LE2) / (2 * LSLE);

   if ( (c3 > -1) && (c3 <= 1) )
   {
      s3 = -sqrt(1.-c3*c3);             /* elbow up */
      theta[2] = atan2(s3,c3);

      if((theta[2] < -1.57079) || (theta[2] > 0.)) {
  /*       printf(" outside the range of theta3 \n");
*/
         return(0);
      }
   }
   else
   {
/*        printf(" outside the range of c3 \n");
*/
        return(0);
   }

/*   calculate theta 2   */

   s1 = sin(theta[0]);
   c1 = cos(theta[0]);

   dd = -s1 * pw[0] + c1 * pw[1];

   theta[1] = atan2(((c3+LR)*pw[2] - s3*dd),(s3*pw[2] + (c3+LR)*dd));

   if((theta[1] < -0.5236) || (theta[1] > 1.7453))
   {
/*      printf(" outside the range of theta2 \n");
*/
      return(0);
   }

/*   calculate theta 4   */
```

```
   c23 = cos(theta[1]+theta[2]);
   s23 = sin(theta[1]+theta[2]);

   w22 = -s1 * c23 * a[0]  +  c1 * c23 * a[1]  +  s23 * a[2];
   w32 = s1 * s23 * a[0]  -  c1 * s23 * a[1]  +  c23 * a[2];

   theta[3] = atan2(w32,w22);

   if((theta[3] < -1.5707) || (theta[3] > 1.5707))
   {
/*      printf(" outside the range of theta4 \n");
*/
      return(0);
   }

/*  calculate theta 5   */

   theta[4] = *roll;

   if((theta[4] < -3.14159) || (theta[4] > 3.14159))
   {
/*      printf(" outside the range of theta5 \n");
*/
      return(0);
   }
   return(1);
}
```

```
/*--------------------------------------------------------------------

File
        drob1c.c - draw robot figure from measured data file
                   with coordinates
Synopsis
        #include <stdio.h>
        #include <math.h>
        #include "robintc.c"

        lcl drob1c
        linkvdi drob1c

Description
        This program draws the robot hand trajectory, pitch angle tick,
        initial & final robot figures,and coordinates from the data file
        of the pick and place task.  Perspective view point and zoom is
        changeable.  The end of file is represented by hi = 1010.

Author
        Munehisa Takeda

Date
        Jan. 28, 1988

--------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
   int hi;
#include "robintc.c"

#define RTD 57.29578    /* scaling parameter (rad to degree) */


FILE * fp1;

main()
{
   static double theta[5];
   char fname[12];
   int ja[5];
   long t;
   int i;
   char ch;

   printf(" data file name (********.dat) = ");
   scanf("%s",fname);
   printf(" viewing distance (dispz:-1000) = ");
   scanf("%lf",&dispz);
   printf(" viewing angle (vax:-45,vay:0,vaz:-45) = ");
   scanf("%lf %lf %lf",&vax,&vay,&vaz);
   printf(" zoom factor (zoom:15) = ");
   scanf("%lf",&zoom);
```

```
   vax /= RTD;
   vay /= RTD;
   vaz /= RTD;

   fp1 = fopen(fname, "r");     /* file open for reading */
   v_opnwk(parameter,&screen,savary);

start:
   do
   {

      fscanf(fp1,"%ld %d %d %d %d %d %d",&t,&ja[0],&ja[1],&ja[2],&ja[3],
      &ja[4],&hi);

      rtttran(ja,theta);

      for(i = 0; i < 5; i++)
         theta[i] *= RTD;

      theta[4] -= 90.;

      robot1(theta);

/*
      printf(" t,ja hi = %ld %d %d %d %d %d %d\n",t,ja[0],ja[1],ja[2],
      ja[3],ja[4],hi);
      printf(" theta = %lf %lf %lf %lf %lf\n",theta[0],theta[1],theta[2],
      theta[3],theta[4]);
*/

   } while (hi != 1010);
/*
   printf(" finish task !!\n ");
   printf(" hit 'r' key for repeating.\n ");
   printf(" hit 'f' key for changing data file.\n ");
   printf(" hit 'd' key for changing viewing distance.\n ");
   printf(" hit 'a' key for changing viewing angle.\n ");
   printf(" hit 'z' key for changing zoom factor.\n ");
   printf(" hit 'c' key for changing all data.\n ");
   printf(" hit 'e' key for ending.\n ");
*/
for(;;)
   {
        ch = bdos(0x06,0x00ff) & 0x00ff;
        switch(ch)
        {
          case 'r':     /* repeat */
              fclose(fp1);
              v_clrwk(screen);
              ct = 0;
              fp1 = fopen(fname, "r");  /* file open for reading */
              goto start;
              break;
          case 'f':     /* change data file */
```

```
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" data file name (********.dat) = ");
        scanf("%s",fname);
        v_clrwk(screen);
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'd':     /* change view distance */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" viewing distance (dispz:-1000) = ");
        scanf("%lf",&dispz);
        v_clrwk(screen);
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'a':     /* change view angle */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" viewing angle (vax:-45,vay:0,vaz:-45) = ");
        scanf("%lf %lf %lf",&vax,&vay,&vaz);
        v_clrwk(screen);
        vax /= RTD;
        vay /= RTD;
        vaz /= RTD;
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'z':     /* change zoom */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" zoom factor (zoom:15) = ");
        scanf("%lf",&zoom);
        v_clrwk(screen);
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'c':     /* change all data */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" data file name (********.dat) = ");
        scanf("%s",fname);
        printf(" viewing distance (dispz:-1000) = ");
        scanf("%lf",&dispz);
        printf(" viewing angle (vax:-45,vay:0,vaz:-45) = ");
        scanf("%lf %lf %lf",&vax,&vay,&vaz);
        printf(" zoom factor (zoom:15) = ");
        scanf("%lf",&zoom);
```

```
                v_clrwk(screen);
                vax /= RTD;
                vay /= RTD;
                vaz /= RTD;
                fp1 = fopen(fname, "r");  /* file open for reading */
                goto start;
                break;
            case 'e':     /* exit */
                fclose(fp1);
                v_clrwk(screen);
                v_clswk(screen);
                exit(1);
                break;
             default:
                break;
          }       /* end of switch */
      }           /* end of infinit loop */

}

/*
theta to robot command parameter transformation function
*/

rtttran(ja,theta)
int ja[];
double theta[];
{

   int i;

/* calculate robot command parameter */
   theta[0] = -0.025 * ja[0];
   theta[1] = 0.025 * (ja[1] + 1400);
   theta[2] = 0.025 * (ja[2] - 1800);
   theta[3] = 0.0375 * (ja[3] - ja[4]);
   theta[4] = -0.0375 * (ja[3] + ja[4] - 400);

/*   limit range check for robot command parameter   */
   theta[0] = (theta[0] > 90) ? 90 : theta[0];
   theta[0] = (theta[0] < -90) ? -90 : theta[0];
   theta[1] = (theta[1] > 100) ? 100 : theta[1];
   theta[1] = (theta[1] < -30) ? -30 : theta[1];
   theta[2] = (theta[2] > 0) ? 0 : theta[2];
   theta[2] = (theta[2] < -90) ? -90 : theta[2];
   theta[3] = (theta[3] > 90) ? 90 : theta[3];
   theta[3] = (theta[3] < -90) ? -90 : theta[3];
   theta[4] = (theta[4] > 195) ? 195 : theta[4];
   theta[4] = (theta[4] < -165) ? -165 : theta[4];

   for(i = 0; i < 5; i++)
      theta[i] /= RTD;

}
```

```
/*  robintc.c  */
/*  Dec. 12, 1987  */

#define MAXPOINTS 11
#define X 1
#define Y 2
#define Z 3
#define JX 1
#define JY 2
#define JZ 3
#define SX 36          /* scaling factors */
#define SY 25
#define TX 320
#define TY 175
#define DEGTORAD .017453
#define L0 250.          /* base length                          */
#define L1 220.          /* shoulder to elbow                    */
#define L2 160.          /* elbow to wrist                       */
#define L3 177.6         /* wrist to gripper                     */
#define L4 30.           /* one half of maximum gripper width    */
#define L5 37.           /* gripper length                       */
#define L6 24.4          /* gripper base to pad center           */

struct vector3
       {
        double x;
        double y;
        double z;
       };
struct vector3 nrobot[11];
struct vector3 robot[11] = { 0.,  0., 0.,    /* point 0 = base */
                             0.,  0., 0.,
                             0.,  0., 0.,
                             0.,  0., 0.,
                             0.,  0., 0.,    /* wrist/gripper joint  */
                             L4, L5,  0.,    /* ¦                    */
                             L4,  0., 0.,    /* ¦  4 points defining */
                             -L4, 0., 0.,    /* ¦     gripper        */
                             -L4, L5, 0.,    /* ¦                    */
                             0.,  L6, 0.,    /* center of gripper    */
                             0.,  L6, 0.   /* base of reference line */
                            };

double jac1,jac2,jac3,jac4,jac5;    /* joint angles cosines */
double jas1,jas2,jas3,jas4,jas5;    /* joint angles sines */
double ja1,ja2,ja3,ja4,ja5;         /* joint angles */
double vax=-45*DEGTORAD, vay=0, vaz=-45*DEGTORAD;
double dispz=-1000;
double zoom=15;
double vacx,vacy,vacz;
double vasx,vasy,vasz;
double point[3];
double arrow[3];                    /* orientation of the end effector*/
double xy[2];
```

```
int uv[12][2];
int cuv[4][2];

long ct = 0;

int screen,gdms_err,savary[66];
int parameter[19] = {2,1,1,3,1,1,1,0,0,1,1,
                     'D','I','S','P','L','A','Y',' '};

int i;
int quit;
int again;

robot1(ja)
double ja[5];
{
   /* obtain user inputs ja1, ja2, ja3, ja4, ja5 , D, and d */

/*   GetAngles();  */
/*   Getview();    */


   /* computation of sins and cosines */

      ja1 = DEGTORAD * ja[0];
      ja2 = DEGTORAD * ja[1];
      ja3 = DEGTORAD * ja[2];
      ja4 = DEGTORAD * ja[3];
      ja5 = DEGTORAD * ja[4];

/*      vax *= DEGTORAD;
      vay *= DEGTORAD;
      vaz *= DEGTORAD;  */

      jac1 = cos(ja1);
      jac2 = cos(ja2);
      jac3 = cos(ja3);
      jac4 = cos(ja4);
      jac5 = cos(ja5);
      jas1 = sin(ja1);
      jas2 = sin(ja2);
      jas3 = sin(ja3);
      jas4 = sin(ja4);
      jas5 = sin(ja5);

      vacx = cos(vax);
      vacy = cos(vay);
      vacz = cos(vaz);
      vasx = sin(vax);
      vasy = sin(vay);
      vasz = sin(vaz);


for (i=0; i < MAXPOINTS; i++)
```

```
{
      point[0] = robot[i].x;
      point[1] = robot[i].y;
      point[2] = robot[i].z;
      switch (i)
      {
        case 0:
                break;
        case 1:
                translate(0.,0.,L0,point);
/* printf("\n rotate points %f %f %f",point[0],point[1],point[2]);*/
                break;

        case 2:
                translate(0.,L1,0.,point);       /* might consider
                                                    writing this */
                rotate(JX,jac2,jas2,point);    /* block as function */
                rotate(JZ,jac1,jas1,point);
                translate(0.,0.,L0,point);
                break;

        case 3:
                translate(0.,L2,0.,point);
                rotate(JX,jac3,jas3,point);
                translate(0.,L1,0.,point);
                rotate(JX,jac2,jas2,point);
                rotate(JZ,jac1,jas1,point);
                translate(0.,0.,L0,point);
                break;

        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
                translate(0.,L3,0.,point);
                rotate(JY,jac5,jas5,point);
                rotate(JX,jac4,jas4,point);
                translate(0.,L2,0.,point);
                rotate(JX,jac3,jas3,point);
                translate(0.,L1,0.,point);
                rotate(JX,jac2,jas2,point);
                rotate(JZ,jac1,jas1,point);
                translate(0.,0.,L0,point);
                break;

         }
 nrobot[i].x = point[0];
 nrobot[i].y = point[1];
 nrobot[i].z = point[2];

 nrobot[10].z = 0.;                    /* reference line base */
```

```
orientation(nrobot[4].x,nrobot[4].y,nrobot[4].z,nrobot[9].x,nrobot[9].y,
nrobot[9].z,arrow);


}

     for (i=0; i<MAXPOINTS; i++)
     {
       point[0] = nrobot[i].x;
       point[1] = nrobot[i].y;
       point[2] = nrobot[i].z;

       rotate(Z,vacz,vasz,point);
       rotate(Y,vacy,vasy,point);
       rotate(X,vacx,vasx,point);

       translate(0.,0.,dispz,point);
       perspective(point,zoom,xy);
       screen_map(xy,&uv[i][0]);

     }

       if(ct == 0)
       {
          uv[11][0] = uv[9][0];
          uv[11][1] = uv[9][1];
          draw_cood();
       }

/*     v_opnwk(parameter,&screen,savary);   */
/*     v_clrwk(screen);                     */
       if((ct == 0) || (hi == 1010))
          draw_robot();
       draw_tick();
       draw_traject();

       uv[11][0] = uv[9][0];
       uv[11][1] = uv[9][1];
       ct++;

/*     v_clswk(screen);                     */
/*printf("\n center of end effector = %f %f %f",nrobot[9].x,nrobot[9].y,
nrobot[9].z); */
/*printf("\n wrist joint position = %f %f %f", nrobot[4].x,nrobot[4].y,
nrobot[4].z);
printf("\n orientation of the end effector is (%f %f %f)",arrow[0],
arrow[1], arrow[2]); */
/* printf("\n run program again (y/n) >>   ");
scanf("%s",&again);
if(again != 'y')
       {
              quit = 1;
       }
```

```
    }
    while(!quit);     */
}                                                   /* end of main */


GetAngles()                                        /* currently not used */
{
      printf("\n input joint angles 1, 2, 3, 4, 5 : ");
      scanf("%lf %lf %lf %lf %lf",&ja1,&ja2,&ja3,&ja4,&ja5);
      printf("joint angles are = %lf %lf %lf %lf %lf \n",ja1,ja2,ja3,
      ja4,ja5);
}

Getview()
{
      printf("\n input angle of X rotation (view):  ");
      scanf("%lf",&vax);
      printf("x view angle = %lf \n",vax);

      printf("\n input angle of Y rotation (view):  ");
      scanf("%lf",&vay);
      printf("y view angle = %lf \n",vay);

      printf("\n input angle of Z rotation (view):  ");
      scanf("%lf",&vaz);
      printf("z view angle = %lf \n",vaz);

      printf("\n input D:");
      scanf("%lf",&dispz);
      printf("dispz = %lf \n",dispz);

      printf("\n input zoom distance: ");
      scanf("%lf",&zoom);
      printf("zoom = %lf \n",zoom);
}

rotate(axis,cosdeg,sindeg,pt)
int axis;
double cosdeg,sindeg;
double pt[];
{
static double temp[3];

switch (axis)
       {
        case 1:
                temp[0] = pt[0];
                temp[1] = cosdeg * pt[1] - sindeg * pt[2];
                temp[2] = sindeg * pt[1] + cosdeg * pt[2];
                break;

        case 2:
                temp[0] = cosdeg * pt[0] + sindeg * pt[2];
                temp[1] = pt[1];
```

```
                temp[2] = sindeg * pt[0] - cosdeg * pt[2];
                break;

        case 3:
                temp[0] = cosdeg * pt[0] - sindeg * pt[1];
                temp[1] = sindeg * pt[0] + cosdeg * pt[1];
                temp[2] = pt[2];
                break;
        }

pt[0] = temp[0];
pt[1] = temp[1];
pt[2] = temp[2];
/* printf("\n rotate points %lf %lf %lf",pt[0],pt[1],pt[2]);      */

}

orientation(pt4x,pt4y,pt4z,pt9x,pt9y,pt9z,arow)
double pt4x, pt4y, pt4z;
double pt9x, pt9y, pt9z;
double arow[];
{
        arow[0] = pt9x - pt4x;
        arow[1] = pt9y - pt4y;
        arow[2] = pt9z - pt4z;
}

translate(tx,ty,tz,pt)
double tx,ty,tz;
double pt[];
{
        pt[0] += tx;
        pt[1] += ty;
        pt[2] += tz;
}

perspective(pt,zoom_d,xy)
double pt[3];
double zoom_d;
double xy[2];
{
        xy[0] = zoom_d * pt[0]/pt[2];
        xy[1] = zoom_d * pt[1]/pt[2];
}

screen_map(xy,uv)
double xy[2];
int uv[2];
{
        uv[0] = SX * xy[0] + TX;
        uv[1] = SY * xy[1] + TY;
}

draw_robot()
```

```
{
        vsl_color(screen,4);
        vsl_type(screen,1);
        draw_line(0,1);
        draw_line(1,2);
        draw_line(2,3);
        draw_line(3,4);
        draw_line(4,6);
        draw_line(4,7);
        draw_line(7,8);
        draw_line(6,5);
        vsl_color(screen,2);
        vsl_type(screen,5);
        draw_line(9,10);
}


draw_tick()
{
        vsl_color(screen,5);    /* set yellow */
        vsl_type(screen,1);     /* set solid line */
        draw_line(4,9);
}


draw_traject()
{
        vsl_color(screen,1);    /* set white */
        vsl_type(screen,1);     /* set solid line */
        draw_line(9,11);
}


draw_line(m,n)
int m,n;
{
static int xy[4];
int count;
count = 2;
xy[0] = uv[m][0];
xy[1] = uv[m][1];
xy[2] = uv[n][0];
xy[3] = uv[n][1];
v_pline(screen,count,xy);
}


draw_cood()
{
        cuv[0][0] = uv[0][0];
        cuv[0][1] = uv[0][1];

        point[0] = 500.;
        point[1] = 0.;
```

```
        point[2] = 0.;

        rotate(Z,vacz,vasz,point);
        rotate(Y,vacy,vasy,point);
        rotate(X,vacx,vasx,point);

        translate(0.,0.,dispz,point);
        perspective(point,zoom,xy);
        screen_map(xy,&cuv[1][0]);

        point[0] = 0.;
        point[1] = 500.;
        point[2] = 0.;

        rotate(Z,vacz,vasz,point);
        rotate(Y,vacy,vasy,point);
        rotate(X,vacx,vasx,point);

        translate(0.,0.,dispz,point);
        perspective(point,zoom,xy);
        screen_map(xy,&cuv[2][0]);

        point[0] = 0.;
        point[1] = 0.;
        point[2] = 500.;

        rotate(Z,vacz,vasz,point);
        rotate(Y,vacy,vasy,point);
        rotate(X,vacx,vasx,point);

        translate(0.,0.,dispz,point);
        perspective(point,zoom,xy);
        screen_map(xy,&cuv[3][0]);

        vsl_type(screen,1);     /* set solid line */
        vsl_color(screen,3);    /* set x axis as green */
        draw_cline(0,1);
        vsl_color(screen,6);    /* set y axis as cyan */
        draw_cline(0,2);
        vsl_color(screen,7);    /* set zaxis as magenta */
        draw_cline(0,3);
}


draw_cline(m,n)
int m,n;
{
static int cxy[4];
int count;
count = 2;
cxy[0] = cuv[m][0];
cxy[1] = cuv[m][1];
cxy[2] = cuv[n][0];
cxy[3] = cuv[n][1];
```

```
/*--------------------------------------------------------------------

File
        prob1c.c - plot robot figure from measured data file
                   with coordinates, pitch tick
Synopsis
        #include <stdio.h>
        #include <math.h>
        #include "probintc.c"

        lcl prob1c
        linkvdi prob1c

Description
        This program plots the robot hand trajectory, pitch angle tick,
        initial and final figures, and coordinates from the data file
        of the pick and place task. Perspective view and zoom aare
        changeable.  The end of file is represented by hi = 1010.

Author
        Munehisa Takeda

Date
        Dec. 12, 1987

--------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
   int hi;
#include "probintc.c"

#define RTD 57.29578    /* scaling parameter (rad to degree) */


FILE * fp1;

main()
{
   static double theta[5];
   char fname[12];
   int ja[5];
   long t;
   int i;
   char ch;

   printf(" data file name (********.dat) = ");
   scanf("%s",fname);
   printf(" viewing distance (dispz:-1000) = ");
   scanf("%lf",&dispz);
   printf(" viewing angle (vax:-45,vay:0,vaz:-45) = ");
   scanf("%lf %lf %lf",&vax,&vay,&vaz);
   printf(" zoom factor (zoom:15) = ");
   scanf("%lf",&zoom);
   printf(" plotter scaling (SX,SY,TX,TY) = ");
```

```
   scanf("%lf %lf %lf %lf",&SX,&SY,&TX,&TY);

   vax /= RTD;
   vay /= RTD;
   vaz /= RTD;

   fp1 = fopen(fname, "r");     /* file open for reading */
   v_opnwk(parameter,&screen,savary);

printf(" hor ver %d %d = \n",savary[51],savary[52]);
start:
   do
   {

      fscanf(fp1,"%ld %d %d %d %d %d %d",&t,&ja[0],&ja[1],&ja[2],&ja[3],
      &ja[4],&hi);

      rtttran(ja,theta);

      for(i = 0; i < 5; i++)
         theta[i] *= RTD;

      theta[4] -= 90.;

      robot1(theta);

/*
      printf(" t,ja hi = %ld %d %d %d %d %d %d\n",t,ja[0],ja[1],ja[2],
      ja[3],ja[4],hi);
      printf(" theta = %lf %lf %lf %lf %lf\n",theta[0],theta[1],
      theta[2],theta[3],theta[4]);
*/

   } while (hi != 1010);
/*
   printf(" finish task !!\n ");
   printf(" hit 'r' key for repeating.\n ");
   printf(" hit 'f' key for changing data file.\n ");
   printf(" hit 'd' key for changing viewing distance.\n ");
   printf(" hit 'a' key for changing viewing angle.\n ");
   printf(" hit 'z' key for changing zoom factor.\n ");
   printf(" hit 'c' key for changing all data.\n ");
   printf(" hit 'e' key for ending.\n ");
*/
for(;;)
   {
        ch = bdos(0x06,0x00ff) & 0x00ff;
        switch(ch)
        {
          case 'r':     /* repeat */
              fclose(fp1);
              v_clrwk(screen);
              ct = 0;
              fp1 = fopen(fname, "r");  /* file open for reading */
```

```
        goto start;
        break;
    case 'f':     /* change data file */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" data file name (********.dat) = ");
        scanf("%s",fname);
        v_clrwk(screen);
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'd':     /* change view distance */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" viewing distance (dispz:-1000) = ");
        scanf("%lf",&dispz);
        v_clrwk(screen);
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'a':     /* change view angle */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" viewing angle (vax:-45,vay:0,vaz:-45) = ");
        scanf("%lf %lf %lf",&vax,&vay,&vaz);
        v_clrwk(screen);
        vax /= RTD;
        vay /= RTD;
        vaz /= RTD;
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'z':     /* change zoom */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" zoom factor (zoom:15) = ");
        scanf("%lf",&zoom);
        v_clrwk(screen);
        fp1 = fopen(fname, "r");  /* file open for reading */
        goto start;
        break;
    case 'c':     /* change all data */
        fclose(fp1);
        v_clrwk(screen);
        ct = 0;
        printf(" data file name (********.dat) = ");
        scanf("%s",fname);
        printf(" viewing distance (dispz:-1000) = ");
        scanf("%lf",&dispz);
        printf(" viewing angle (vax:-45,vay:0,vaz:-45) = ");
```

```
                scanf("%lf %lf %lf",&vax,&vay,&vaz);
                printf(" zoom factor (zoom:15) = ");
                scanf("%lf",&zoom);
                v_clrwk(screen);
                vax /= RTD;
                vay /= RTD;
                vaz /= RTD;
                fp1 = fopen(fname, "r");  /* file open for reading */
                goto start;
                break;
            case 'e':     /* exit */
                fclose(fp1);
                v_clrwk(screen);
                v_clswk(screen);
                exit(1);
                break;
             default:
                break;
        }       /* end of switch */
    }           /* end of infinit loop */

}

/*
theta to robot command parameter transformation function
*/

rtttran(ja,theta)
int ja[];
double theta[];
{

   int i;

/* calculate robot command parameter */
   theta[0] = -0.025 * ja[0];
   theta[1] = 0.025 * (ja[1] + 1400);
   theta[2] = 0.025 * (ja[2] - 1800);
   theta[3] = 0.0375 * (ja[3] - ja[4]);
   theta[4] = -0.0375 * (ja[3] + ja[4] - 400);

/*   limit range check for robot command parameter   */
   theta[0] = (theta[0] > 90) ? 90 : theta[0];
   theta[0] = (theta[0] < -90) ? -90 : theta[0];
   theta[1] = (theta[1] > 100) ? 100 : theta[1];
   theta[1] = (theta[1] < -30) ? -30 : theta[1];
   theta[2] = (theta[2] > 0) ? 0 : theta[2];
   theta[2] = (theta[2] < -90) ? -90 : theta[2];
   theta[3] = (theta[3] > 90) ? 90 : theta[3];
   theta[3] = (theta[3] < -90) ? -90 : theta[3];
   theta[4] = (theta[4] > 195) ? 195 : theta[4];
   theta[4] = (theta[4] < -165) ? -165 : theta[4];

   for(i = 0; i < 5; i++)
```

```
/*  probintc.c  */
/*  Dec. 14, 1987  */

#define MAXPOINTS 11
#define X 1
#define Y 2
#define Z 3
#define JX 1
#define JY 2
#define JZ 3
double SX = 540;         /* scaling factors */
double SY = 375;
double TX = 160;
double TY = 80;
#define DEGTORAD .017453
#define L0 250.         /* base length                          */
#define L1 220.         /* shoulder to elbow                    */
#define L2 160.         /* elbow to wrist                       */
#define L3 177.6        /* wrist to gripper                     */
#define L4 30.          /* one half of maximum gripper width    */
#define L5 37.          /* gripper length                       */
#define L6 24.4         /* gripper base to pad center           */

struct vector3
       {
        double x;
        double y;
        double z;
       };
struct vector3 nrobot[11];
struct vector3 robot[11] = { 0.,  0., 0.,        /* point 0 = base */
                             0.,  0., 0.,
                             0.,  0., 0.,
                             0.,  0., 0.,
                             0.,  0., 0.,    /* wrist/gripper joint  */
                             L4, L5,  0.,    /* ¦                    */
                             L4,  0., 0.,    /* ¦  4 points defining */
                             -L4, 0., 0.,    /* ¦     gripper        */
                             -L4, L5, 0.,    /* ¦                    */
                             0.,  L6, 0.,    /* center of gripper    */
                             0.,  L6, 0.    /* base of reference line */
                            };

double jac1,jac2,jac3,jac4,jac5;    /* joint angles cosines */
double jas1,jas2,jas3,jas4,jas5;    /* joint angles sines */
double ja1,ja2,ja3,ja4,ja5;         /* joint angles */
double vax=-45*DEGTORAD, vay=0, vaz=-45*DEGTORAD;
double dispz=-1000;
double zoom=15;
double vacx,vacy,vacz;
double vasx,vasy,vasz;
double point[3];
double arrow[3];                    /* orientation of the end effector*/
double xy[2];
```

```
int uv[12][2];
int cuv[4][2];

long ct = 0;

int screen,gdms_err,savary[66];
int parameter[19] = {2,1,1,3,1,1,1,0,0,1,1,
                     'P','L','O','T','T','E','R',' '};

int i;
int quit;
int again;

robot1(ja)
double ja[5];
{
   /* obtain user inputs ja1, ja2, ja3, ja4, ja5 , D, and d */

/*   GetAngles();  */
/*   Getview();    */


   /* computation of sins and cosines */

      ja1 = DEGTORAD * ja[0];
      ja2 = DEGTORAD * ja[1];
      ja3 = DEGTORAD * ja[2];
      ja4 = DEGTORAD * ja[3];
      ja5 = DEGTORAD * ja[4];

/*      vax *= DEGTORAD;
      vay *= DEGTORAD;
      vaz *= DEGTORAD;  */

      jac1 = cos(ja1);
      jac2 = cos(ja2);
      jac3 = cos(ja3);
      jac4 = cos(ja4);
      jac5 = cos(ja5);
      jas1 = sin(ja1);
      jas2 = sin(ja2);
      jas3 = sin(ja3);
      jas4 = sin(ja4);
      jas5 = sin(ja5);

      vacx = cos(vax);
      vacy = cos(vay);
      vacz = cos(vaz);
      vasx = sin(vax);
      vasy = sin(vay);
      vasz = sin(vaz);


for (i=0; i < MAXPOINTS; i++)
```

```
{
      point[0] = robot[i].x;
      point[1] = robot[i].y;
      point[2] = robot[i].z;
      switch (i)
      {
        case 0:
                break;
        case 1:
                translate(0.,0.,L0,point);
/* printf("\n rotate points %f %f %f",point[0],point[1],point[2]);*/
                break;

        case 2:
                translate(0.,L1,0.,point);       /* might consider
                                                    writing this */
                rotate(JX,jac2,jas2,point);    /* block as function */
                rotate(JZ,jac1,jas1,point);
                translate(0.,0.,L0,point);
                break;

        case 3:
                translate(0.,L2,0.,point);
                rotate(JX,jac3,jas3,point);
                translate(0.,L1,0.,point);
                rotate(JX,jac2,jas2,point);
                rotate(JZ,jac1,jas1,point);
                translate(0.,0.,L0,point);
                break;

        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
                translate(0.,L3,0.,point);
                rotate(JY,jac5,jas5,point);
                rotate(JX,jac4,jas4,point);
                translate(0.,L2,0.,point);
                rotate(JX,jac3,jas3,point);
                translate(0.,L1,0.,point);
                rotate(JX,jac2,jas2,point);
                rotate(JZ,jac1,jas1,point);
                translate(0.,0.,L0,point);
                break;

         }
 nrobot[i].x = point[0];
 nrobot[i].y = point[1];
 nrobot[i].z = point[2];

 nrobot[10].z = 0.;                    /* reference line base */
```

```
orientation(nrobot[4].x,nrobot[4].y,nrobot[4].z,nrobot[9].x,nrobot[9].y,
nrobot[9].z,arrow);


}

     for (i=0; i<MAXPOINTS; i++)
     {
       point[0] = nrobot[i].x;
       point[1] = nrobot[i].y;
       point[2] = nrobot[i].z;

       rotate(Z,vacz,vasz,point);
       rotate(Y,vacy,vasy,point);
       rotate(X,vacx,vasx,point);

       translate(0.,0.,dispz,point);
       perspective(point,zoom,xy);
       screen_map(xy,&uv[i][0]);

     }

       if(ct == 0)
       {
          uv[11][0] = uv[9][0];
          uv[11][1] = uv[9][1];
          draw_cood();
       }

/*     v_opnwk(parameter,&screen,savary);   */
/*     v_clrwk(screen);                     */
       if((ct == 0) || (hi == 1010))
          draw_robot();
       draw_tick();
       draw_traject();

       uv[11][0] = uv[9][0];
       uv[11][1] = uv[9][1];
       ct++;

/*     v_clswk(screen);                     */
/*printf("\n center of end effector = %f %f %f",nrobot[9].x,nrobot[9].y,
nrobot[9].z); */
/*printf("\n wrist joint position = %f %f %f", nrobot[4].x,nrobot[4].y,
nrobot[4].z);
printf("\n orientation of the end effector is (%f %f %f)",arrow[0],
arrow[1], arrow[2]); */
/* printf("\n run program again (y/n) >>   ");
scanf("%s",&again);
if(again != 'y')
       {
              quit = 1;
       }
```

```
    }
    while(!quit);     */
}                                                   /* end of main */


GetAngles()                                        /* currently not used */
{
      printf("\n input joint angles 1, 2, 3, 4, 5 : ");
      scanf("%lf %lf %lf %lf %lf",&ja1,&ja2,&ja3,&ja4,&ja5);
      printf("joint angles are = %lf %lf %lf %lf %lf \n",ja1,ja2,ja3,
      ja4,ja5);
}

Getview()
{
      printf("\n input angle of X rotation (view):  ");
      scanf("%lf",&vax);
      printf("x view angle = %lf \n",vax);

      printf("\n input angle of Y rotation (view):  ");
      scanf("%lf",&vay);
      printf("y view angle = %lf \n",vay);

      printf("\n input angle of Z rotation (view):  ");
      scanf("%lf",&vaz);
      printf("z view angle = %lf \n",vaz);

      printf("\n input D:");
      scanf("%lf",&dispz);
      printf("dispz = %lf \n",dispz);

      printf("\n input zoom distance: ");
      scanf("%lf",&zoom);
      printf("zoom = %lf \n",zoom);
}

rotate(axis,cosdeg,sindeg,pt)
int axis;
double cosdeg,sindeg;
double pt[];
{
static double temp[3];

switch (axis)
       {
        case 1:
                temp[0] = pt[0];
                temp[1] = cosdeg * pt[1] - sindeg * pt[2];
                temp[2] = sindeg * pt[1] + cosdeg * pt[2];
                break;

        case 2:
                temp[0] = cosdeg * pt[0] + sindeg * pt[2];
                temp[1] = pt[1];
```

```
                temp[2] = sindeg * pt[0] - cosdeg * pt[2];
                break;

        case 3:
                temp[0] = cosdeg * pt[0] - sindeg * pt[1];
                temp[1] = sindeg * pt[0] + cosdeg * pt[1];
                temp[2] = pt[2];
                break;
        }

pt[0] = temp[0];
pt[1] = temp[1];
pt[2] = temp[2];
/* printf("\n rotate points %lf %lf %lf",pt[0],pt[1],pt[2]);      */

}

orientation(pt4x,pt4y,pt4z,pt9x,pt9y,pt9z,arow)
double pt4x, pt4y, pt4z;
double pt9x, pt9y, pt9z;
double arow[];
{
        arow[0] = pt9x - pt4x;
        arow[1] = pt9y - pt4y;
        arow[2] = pt9z - pt4z;
}

translate(tx,ty,tz,pt)
double tx,ty,tz;
double pt[];
{
        pt[0] += tx;
        pt[1] += ty;
        pt[2] += tz;
}

perspective(pt,zoom_d,xy)
double pt[3];
double zoom_d;
double xy[2];
{
        xy[0] = zoom_d * pt[0]/pt[2];
        xy[1] = zoom_d * pt[1]/pt[2];
}

screen_map(xy,uv)
double xy[2];
int uv[2];
{
        uv[0] = SX * xy[0] + TX;
        uv[1] = SY * xy[1] + TY;
}

draw_robot()
```

```
{
        vsl_color(screen,4);
        vsl_type(screen,1);
        draw_line(0,1);
        draw_line(1,2);
        draw_line(2,3);
        draw_line(3,4);
        draw_line(4,6);
        draw_line(4,7);
        draw_line(7,8);
        draw_line(6,5);
        vsl_color(screen,2);
        vsl_type(screen,5);
        draw_line(9,10);
}


draw_tick()
{
        vsl_color(screen,5);    /* set yellow */
        vsl_type(screen,1);     /* set solid line */
        draw_line(4,9);
}


draw_traject()
{
        vsl_color(screen,1);    /* set white */
        vsl_type(screen,1);     /* set solid line */
        draw_line(9,11);
}


draw_line(m,n)
int m,n;
{
static int xy[4];
int count;
count = 2;
xy[0] = uv[m][0];
xy[1] = uv[m][1];
xy[2] = uv[n][0];
xy[3] = uv[n][1];
v_pline(screen,count,xy);
}


draw_cood()
{
        cuv[0][0] = uv[0][0];
        cuv[0][1] = uv[0][1];

        point[0] = 500.;
        point[1] = 0.;
```

```
        point[2] = 0.;

        rotate(Z,vacz,vasz,point);
        rotate(Y,vacy,vasy,point);
        rotate(X,vacx,vasx,point);

        translate(0.,0.,dispz,point);
        perspective(point,zoom,xy);
        screen_map(xy,&cuv[1][0]);

        point[0] = 0.;
        point[1] = 500.;
        point[2] = 0.;

        rotate(Z,vacz,vasz,point);
        rotate(Y,vacy,vasy,point);
        rotate(X,vacx,vasx,point);

        translate(0.,0.,dispz,point);
        perspective(point,zoom,xy);
        screen_map(xy,&cuv[2][0]);

        point[0] = 0.;
        point[1] = 0.;
        point[2] = 500.;

        rotate(Z,vacz,vasz,point);
        rotate(Y,vacy,vasy,point);
        rotate(X,vacx,vasx,point);

        translate(0.,0.,dispz,point);
        perspective(point,zoom,xy);
        screen_map(xy,&cuv[3][0]);

        vsl_type(screen,1);     /* set solid line */
        vsl_color(screen,3);    /* set x axis as green */
        draw_cline(0,1);
        vsl_color(screen,6);    /* set y axis as cyan */
        draw_cline(0,2);
        vsl_color(screen,7);    /* set zaxis as magenta */
        draw_cline(0,3);
}


draw_cline(m,n)
int m,n;
{
static int cxy[4];
int count;
count = 2;
cxy[0] = cuv[m][0];
cxy[1] = cuv[m][1];
cxy[2] = cuv[n][0];
cxy[3] = cuv[n][1];
```