COMPUTER ANIMATION OF CLOSED-LOOP PLANAR ROBOTS AND WORKSPACE ANALYSIS OF THE OKLAHOMA CRAWDAD ROBOT

By

BRUCE SUMPTER

Bachelor of Science in Mechanical Engineering

Oklahoma State University

Stillwater, Oklahoma

1983

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1986

COMPUTER ANIMATION OF CLOSED-LOOP PLANAR ROBOTS

AND WORKSPACE ANALYSIS OF THE

OKLAHOMA CRAWDAD ROBOT

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my appreciation to my major adviser, Dr. Atmaram H. Soni, for his guidance and for his interest in the findings of this work. Appreciation is also extended to the other committee members, Dr. Keith Good and Dr. Richard Lowry, for their assistance in the preparation of the final manuscript.

Special thanks are extended to Ron Delahoussaye who provided much assistance including the use of his library of computer graphics subroutines. Thanks are also extended to Professor Tom Cook and CAD/CAM lab assistant, Amir Nassirharand, for the assistance they provided. In addition, appreciation is extended to Ms. Pam Laufer and Mrs. Dottie Alexander for typing this thesis.

Finally, I would like to thank the School of Mechanical and Aerospace Engineering for providing a teaching assistantship which made this work and my graduate study at Oklahoma State University possible.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Today, robots are coming of age. They are now performing some of the tasks that were previously performed by conventional machinery or by humans. Their well-known advantage over machinery is that they can be programmed to perform new jobs (Loeff and Soni, 1975). Thus, for example, when a new part must be welded or a new product assembled, an accompanying new machine to perform these operations will not need to be designed, built, tested, and installed. Rather, an existing robot must just be reprogrammed. Robots have been used instead of humans for jobs that are hazardous, monotonous, or that require precision and/or speed. Robots, especially the Oklahoma Crawdad robot that has the potential for performing high speed work, can make the automation of certain small batch jobs cost effective (Paul, 1983).

Of course robots have not replaced all machinery. In particular, mechanisms in general perform relatively small jobs and certain other jobs better than a robot can. Although mechanisms produce only one specific motion, they will continue to be used for many industrial tasks.

Closed-loop robots could be considered a hybrid of mechanism and the common open-loop industrial robot and possess the most desired mechanical attribute of each:

1. They can be reprogrammed to perform new jobs
2. They can move relatively fast

Most, if not all of the industrial robots used today are of either open-loop or of mixed-loop configuration. The mixed-loop configurations usually contain one closed kinematic chain of one degree of freedom in junction with a major open loop chain (Draganoiu et al., 1982). These industrial robots are similar to the human arm in that their end-effector (hand) is--as its name implies--at the end of a series of kinematic links. The other end of these robots is rigidly attached to the ground. In order to have constrained motion, these robots are driven by actuators at each of the joints or by a system of pulleys or chains running throughout the robot. Stemming from this fact, one can conclude that these robots move relatively slower than mechanisms for basically one reason: the robot must drive more mass through space in order to move to a desired location. Both the mass of the actuators or drive system and the mass of the robot structure required to support their extra load cause these robots to be relatively heavy and slow.

Since mechanisms can and are usually driven by an actuator sitting on the ground, their structure is generally light and their work speed is comparatively fast.

Closed-loop robots can and are well suited to be driven by actuators mounted at ground positions. Thus, closed-loop robots, besides having the capabilities to produce more than one motion and to be programmed to perform new jobs, can move relatively fast.

Since little research has been published in the area of closed-loop robots, this study should be considered an introductory one. Thus, the objectives of this work are as follows:

1. Write a computer program that animates the motion of closed-loop planar robots.

2. Write a computer program to plot the boundary and to compute the area of the workspace for the Oklahoma Crawdad robot and to animate the robot moving around the boundary.

3. Use the Crawdad workspace program to determine the effect of link parameters on the workspace area.

CHAPTER II

EXAMPLES OF CLOSED-LOOP PLANAR ROBOTS INCLUDING THE OKLAHOMA CRAWDAD

This work considers only planar closed-loop robots. This means that all the links of the robot move in the same plane. Of course, in constructing an actual closed-loop robot, connecting links will be somewhat offset. When confined to planar motion, there exist only two natural choices of kinematic joints to consider, revolute and prismatic. Since revolute joints are much more commonly used than prismatic joints, this study did not consider the latter. Finally, the type of kinematic links considered was limited to binary and ternary since these two in combination provide sufficient robot branching.

Figures 1 and 2 show examples of the Oklahoma Crawdad robot. It consists of nine joints and eight links. The ground is considered a ternary link. The remaining seven links are made up of six binary links and one ternary link. In this report, only this second ternary link will be called the ternary link. Using Gruebler's criterion, the number of degrees of freedom for this robot are three in number (Soni, 1974). Thus, the Oklahoma Crawdad robot must be driven by three actuators in order to have constrained motion.

Figures 3 through 6 show examples of other closed-loop planar robots.

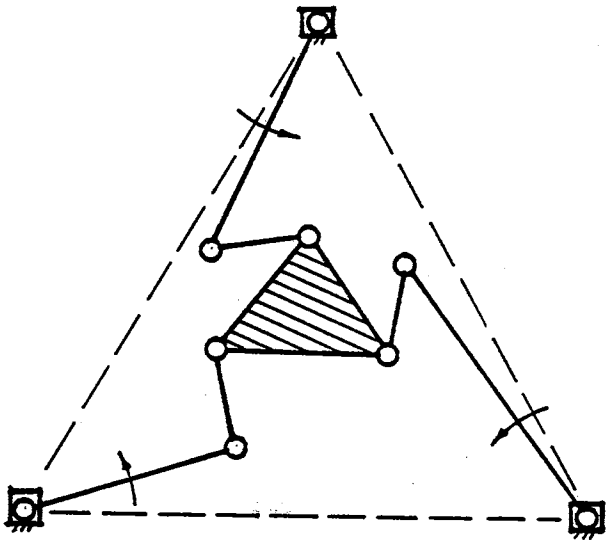


Figure 1. Oklahoma Crawdad Robot with Equilateral Ground Positions

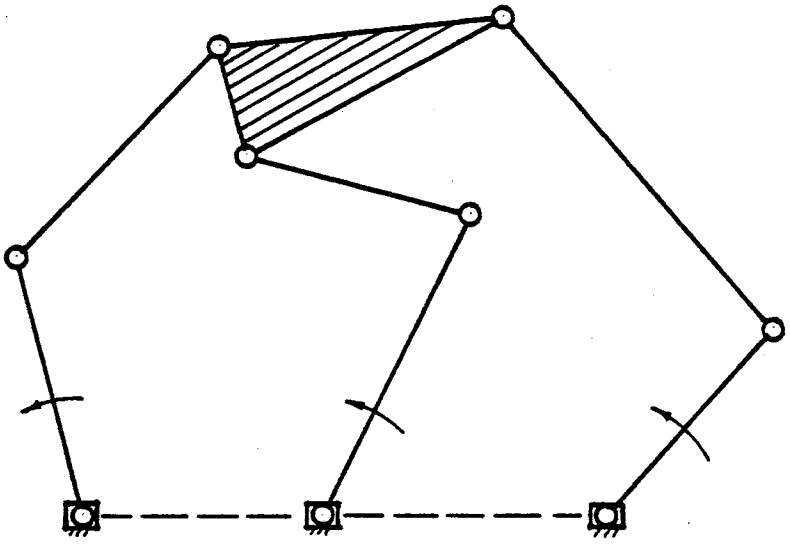


Figure 2. Oklahoma Crawdad Robot with Ground Positions in Line

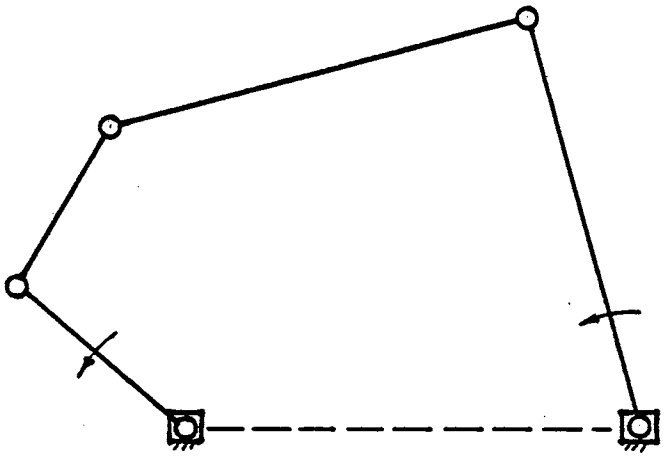


Figure 3. A Five Link Closed-Loop Planar Robot

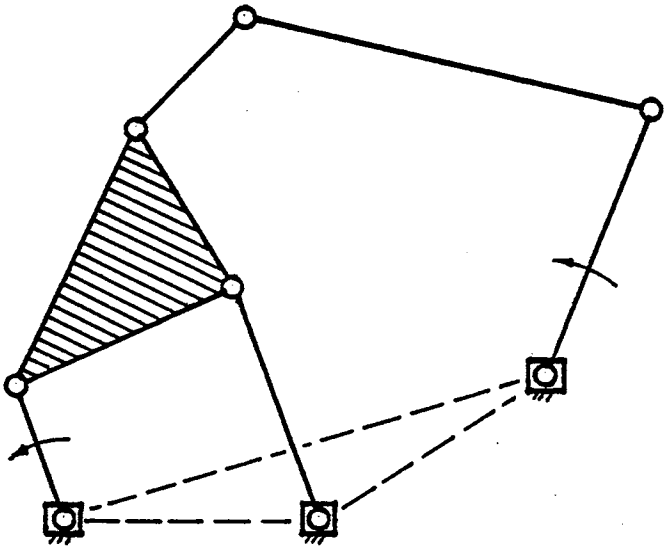


Figure 4. A Seven Link Closed-Loop Planar Robot

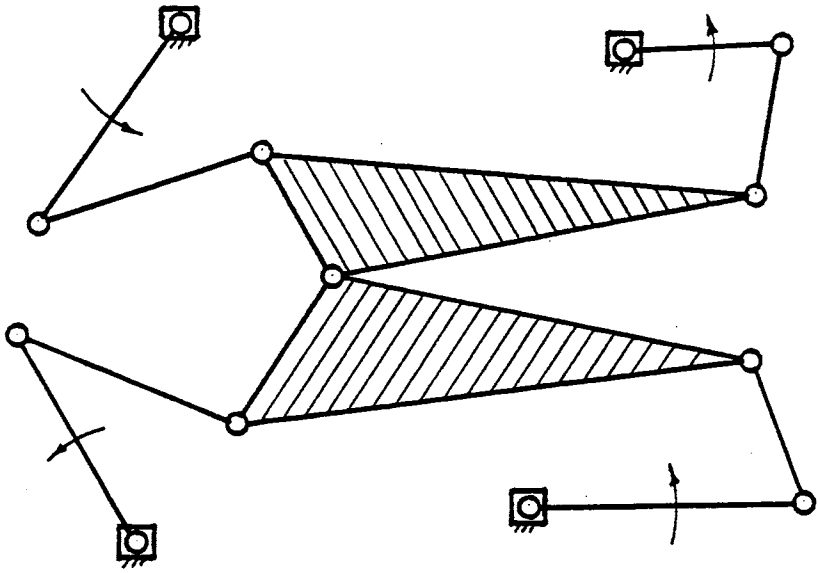


Figure 5. An Eleven Link Closed-Loop Planar Robot

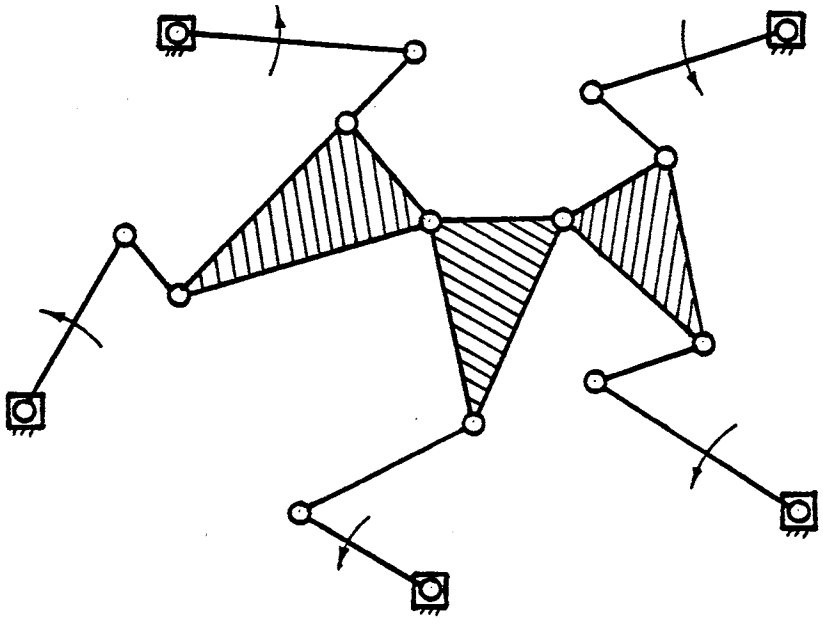


Figure 6. A Fourteen Link Closed-Loop Planar Robot

CHAPTER III

ANIMATION OF CLOSED-LOOP PLANAR ROBOTS

The Kinematic Equations

The Original Coupled Nonlinear Equations

The position of a closed-loop planar robot having only revolute joints is described in terms of the angles formed by each of its links and a common direction. Looking, for example, at the Oklahoma Crawdad robot, the input and solution angles consist of $\Psi_1 - \Psi_3$ and $\theta_1 - \theta_4$ as shown in Figure 7. Letting Q_i, R_i, and T_i be the corresponding link lengths for the links with position described by Ψ_i, θ_i, and $(\theta_i + \alpha_i)$, the following loop closer equations given in complex number notation evolve:

$$x_o+iy_o+Q_1e^{i\Psi_1}+R_4e^{i\theta_4}+T_2e^{i(\theta_2+\alpha_2)}+R_3e^{i\theta_3}+Q_2e^{i\Psi_2}=x_1+iy_1 \qquad (3.1)$$

$$x_o+iy_o+Q_1e^{i\Psi_1}+R_4e^{i\theta_4}+R_2e^{i\theta_2}+R_1e^{i\theta_1}+Q_3e^{i\Psi_3}=x_2+iy_2 \qquad (3.2)$$

Since these equations are coupled and nonlinear for all closed-loop robots, and since the computer program was intended to be as general as possible, a numerical solution technique was chosen over finding countless closed-form solutions.

Since the initial position of the robot is given and since for each new position a nearby previous position is known, the Newton-Raphson predictor-corrector method was used. A first order Taylor's expansion for the example Oklahoma Crawdad robot produce the following equations:

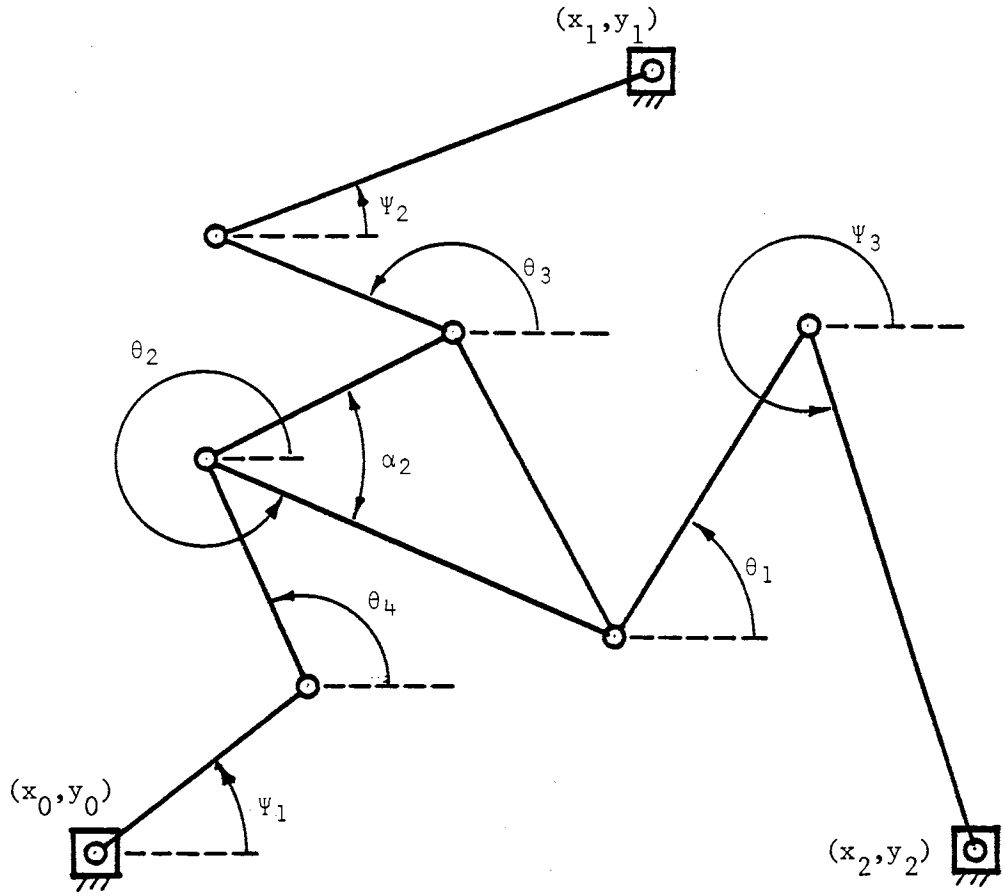


Figure 7. Input and Solution Angles of the Oklahoma Crawdad Robot

$$x_o+iy_o+Q_1e^{i\Psi_1}+R_4e^{i\theta_4}+i\delta_4R_4e^{i\theta_4}+T_2e^{i(\theta_2+\alpha_2)}+i\delta_2T_2e^{i(\theta_2+\alpha_2)}$$
$$+R_3e^{i\theta_3}+i\delta_3R_3e^{i\theta_3}+Q_2e^{i\Psi_2} = x_1+iy_1 \qquad (3.3)$$

$$x_o+iy_o+Q_1e^{i\Psi_1}+R_4e^{i\theta_4}+i\delta_4R_4e^{i\theta_4}+R_2e^{i\theta_2}+i\delta_2R_2e^{i\theta_2}$$
$$+R_1e^{i\theta_1}+i\delta_1R_1e^{i\theta_1}+Q_3e^{i\Psi_3} = x_2+iy_2 \qquad (3.4)$$

The Linearized Matrix Equation

Letting $X_i=x_i-x_o$ and $Y_i=y_i-y_o$ and using trigonometric notation the following matrix equation for animation is produced:

$$\begin{bmatrix} -R_1\sin\theta_1-R_2\sin\theta_2 & 0 & -R_4\sin\theta_4 \\ R_1\cos\theta_1\ R_2\cos\theta_2 & 0 & R_4\cos\theta_4 \\ 0\quad -T_2\sin(\theta_2+\alpha_2) & -R_3\sin\theta_3 & -R_4\sin\theta_4 \\ 0\quad T_2\cos(\theta_2+\alpha_2) & R_3\cos\theta_3 & R_4\cos\theta_4 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix}$$

$$= \begin{bmatrix} X_2 \\ Y_2 \\ X_1 \\ Y_1 \end{bmatrix} - \begin{bmatrix} Q_1\cos\Psi_1+Q_3\cos\Psi_3 \\ Q_1\sin\Psi_1+Q_3\sin\Psi_3 \\ Q_1\cos\Psi_1+Q_2\cos\Psi_2 \\ Q_1\sin\Psi_1+Q_2\sin\Psi_2 \end{bmatrix} - \begin{bmatrix} R_1\cos\theta_1+R_2\cos\theta_2+R_4\cos\theta_4 \\ R_1\sin\theta_1+R_2\sin\theta_2+R_4\sin\theta_4 \\ T_2\cos(\theta_2+\alpha_2)+R_3\cos\theta_3+R_4\cos\theta_4 \\ T_2\sin(\theta_2+\alpha_2)+R_3\sin\theta_3+R_4\sin\theta_4 \end{bmatrix} \qquad (3.5)$$

The above equation can be represented as follows:

$$[A]\underline{\delta} = \underline{G} - \underline{Q} - \underline{R} \qquad (3.6)$$

where [A] is the derivative matrix

$\underline{\delta}$ is the delta theta vector

$\underline{G}$ is the constant ground vector

$\underline{Q}$ is the input vector

and $\underline{R}$ is the previous position vector.

Letting R_i now represent $\begin{bmatrix} -R_i\sin\theta_i \\ R_i\cos\theta_i \end{bmatrix}$ and T_i represent

$\begin{bmatrix} -T_i \sin(\theta_i+\alpha_i) \\ -T_i \sin(\theta_i+\alpha_i) \end{bmatrix}$, each form of the derivative matrix [A] can be easily shown and will represent a different robot. Some examples of these forms and the robots they represent are given in Figure 8. Most complex robots, however, can be represented by several different derivative matrix forms depending upon how one chooses the solution angles.

The Iterative Logic

A flow diagram of the iterative logic utilizing the matrix question is given in Figure 9. With the input functions used, the incremental change for any input link could be at maximum 0.1745 radians. With a convergence criteria of 0.0005 radians, convergence was always found with the second delta theta vector. It was found that if convergence did not occur this quickly, the estimation process would diverge which meant that the robot had exceeded a limit position. In this case, the animation graphics would proceed to this position then stop and give a message to the user.

The Computer Program

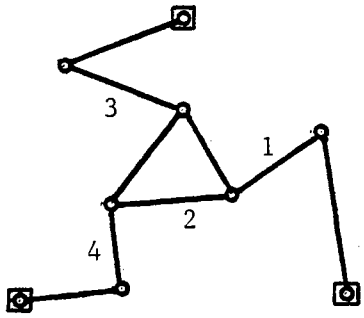
Form of the Required Input Data File

Since the animation program would be used for demonstration purposes, it was deemed more desirable to have a library of input data files describing specific closed-loop robots than to require the user to repeatedly interactively create each robot. In order for users to become familiar with the form of the data file, they may create a single file in the computer disk area named TEST to which the program can access if this option is chosen. After running and modifying this test

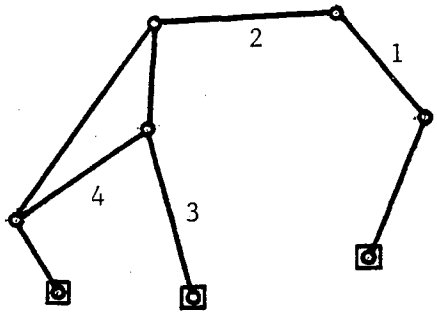
file and producing a desirable animation, the TEST area can then be annexed to the library of data files in the disk area ADATA.

An example of an input data file is given in Figure 10 and a diagram of the robot it describes is given in Figure 11. The program statements that read the data file are of the following format,

$$\begin{bmatrix} R_1 & R_2 & 0 & R_4 \\ 0 & T_2 & R_3 & R_4 \end{bmatrix}$$



$$\begin{bmatrix} R_1 & R_2 & 0 & T_4 \\ 0 & 0 & R_3 & R_4 \end{bmatrix}$$



$$\begin{bmatrix} R_1 & 0 & T_3 & 0 & T_5 & R_6 \\ 0 & R_2 & R_3 & 0 & T_5 & R_6 \\ 0 & 0 & 0 & R_4 & R_5 & R_6 \end{bmatrix}$$

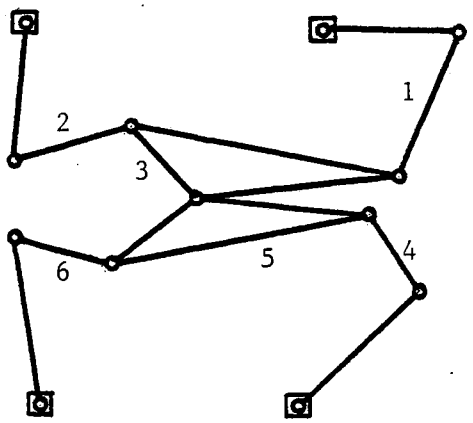


Figure 8. Examples of the Forms of the Derivative Matrix [A] and the Robots they Represent

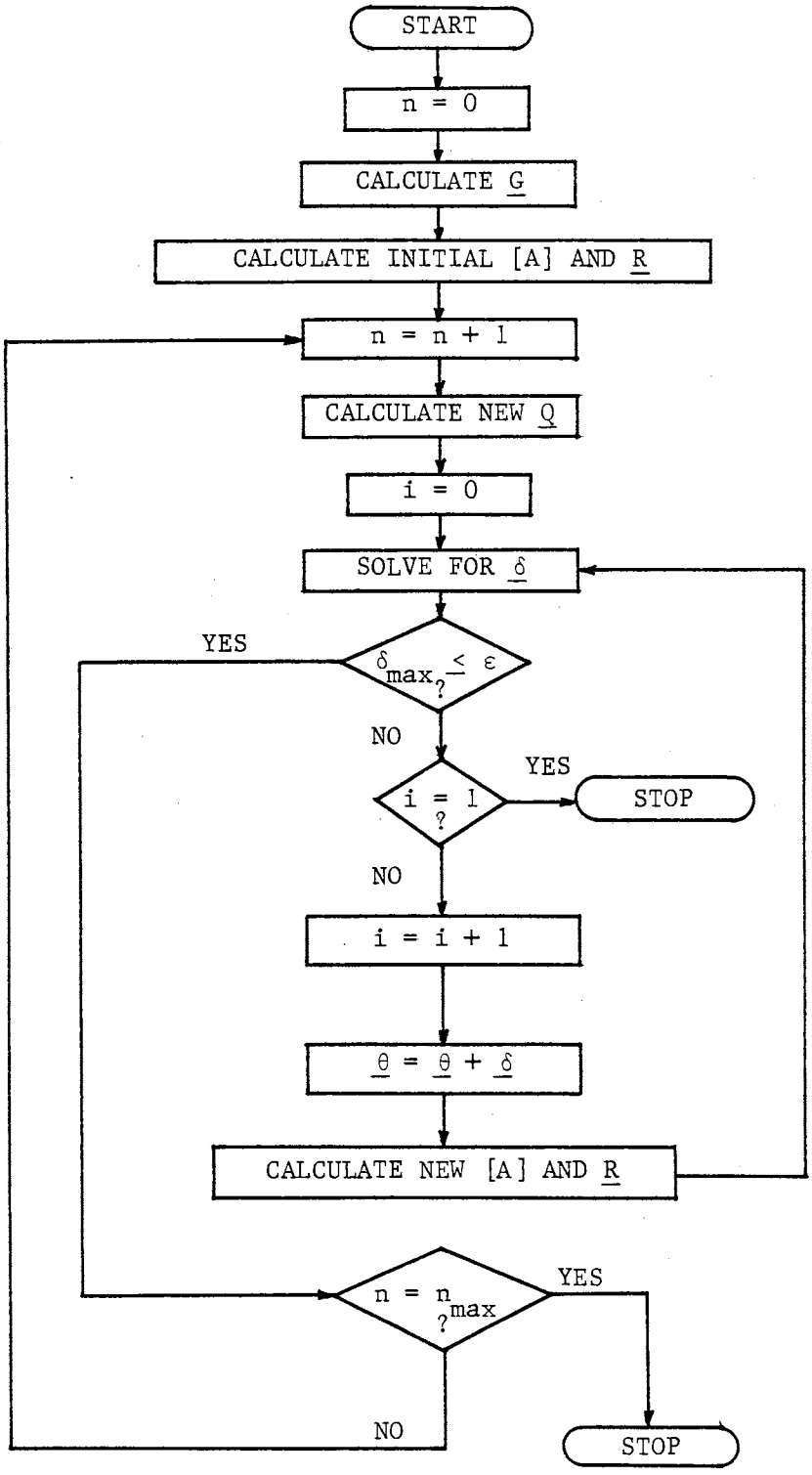


Figure 9. Interative Logic for the Animation Matrix Equation

```
OKC.E                    CRAWDAD W/EE
   1   10    4   11   13   12    6    8   -3    0    0    0    0    0    0    0    0
   2    1   -1    7   -2    4   10    5   -3    0    0    0    0    0    0    0    0
   1    8    6    2    3    4    1    0    0    0    0    0    0    0
   3    4    5    6    8    7    9    0    0    0    0    0    0    0
   0    0    0    0    2    0    0    0    0    0    0    0    0    0
   7    9    5    0    0    7    9    5    0    0
   1    3    2    0    0  -20  -30   10    0    0
   5    4    7
```

Figure 10. An Example Input Data File

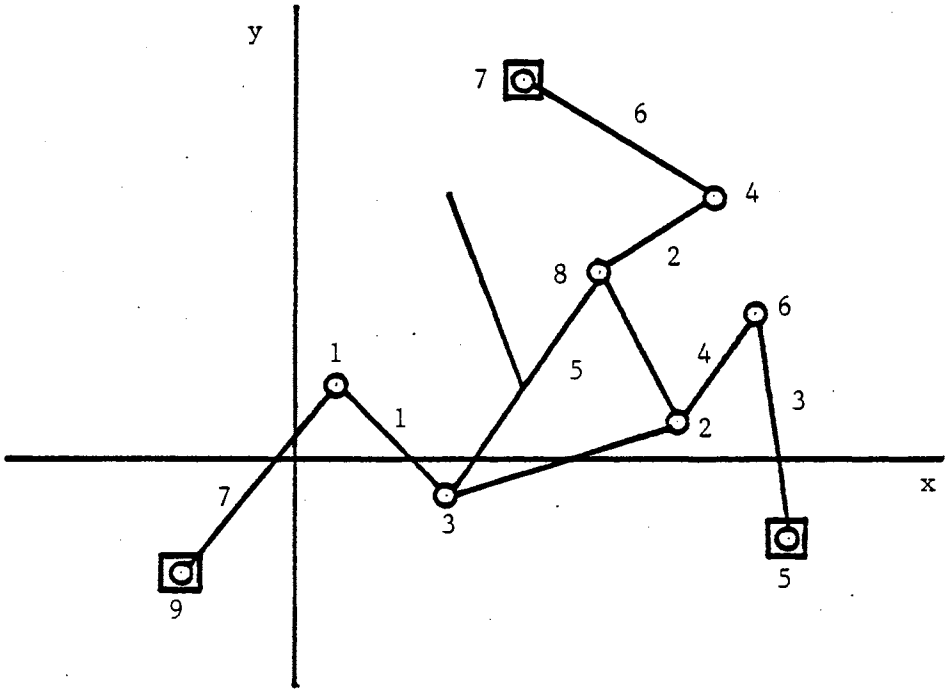


Figure 11. An Example of KP and LINK Numbering Corresponding to the Input Data File of Figure 10

```
READ(NFILE,'(2A20)') DFILE,TITLE
READ(NFILE,*) KP,LINK, GKP,IG,QSP,QMX,NP,P
```

where

```
DFILE is the call name of the data file
TITLE is the title that will be displayed at the graphics
      terminal during animation
KP    are the initial joint coordinates
LINK  are the joints of the links
GKP   are the ground joints
IG    are the input ground joints
QSP   are the relative input function speeds
QMX   are the input function rotations
NP    is the link number of the coupler link
P     are the initial coordinates of the coupler
```

and where the array variables have the following dimensions:

```
INTEGER  LINK(14,3),GKP(5),IG(5),QSP(5)
REAL     KP(17,2),QMX(5),P(2)
```

The input rotation functions were of the following form:

$$\Psi_i = \Psi_{i_o} + \tfrac{1}{2}\,(1-\cos(nQSP_i\Delta t))QMX_i \qquad \|QMX_i\| < 360^\circ \tag{3.7}$$

$$\Psi_i = \Psi_{i_o} + n\Delta t \qquad \|QMX_i\| = 360^\circ \tag{3.8}$$

where $n = 1, n_{max}$

In order to limit the maximum incremental change of the Ψ_i, the maximum allowable $\|QMX_i\|$ for an animation having a maximum QSP of 1 and 2 were set to 90 and 180 degrees respectively. Since the user specifies the initial position of the robot, the Ψ_{i_o} are calculated by the program. The incremental time step and the number of iterations depend on the maximum QSP chosen as shown in the following table:

max QSP	Δt	n_{max}
1	10.0	35
2	5.0	71
3	2.5	143

The program assumes that the user is familiar with the field of kinematics so as to create viable robots (or mechanisms) with viable input parameters. However, if the input rotations specified drive the

robot past a limit position, the animation will continue to this point as earlier mentioned.

Setting Up the Loop Equation Orders

Since the kinematic equations arise from the loop-closer relations, a major portion of the program written is devoted to setting up the loop equation orders (LEO). In order for a robot to be completely described for animation, the position angles of each link must be known. Thus, each link must be represented in at least one loop equation. The method used to properly set up the LEO chose (1) the first ground joint as the common initial ground joint of all of the loops and (2) the remaining ground joints as the final joints of the loops.

After choosing this common ground joint, the program finds the link that shares this joint. Then, all of the other joints of this link become the second joints of different loops. This process is continued until all of the loops have terminated with a ground joint. At this point, the LEO represent an ordering of joint numbers as shown in the example of Figure 12.

These LEO are now converted into link position angle numbers as shown in Figure 13.

Position angles 1, 5, and 8 become input angles, angles 2, 4, 6, and 7 become solution angles, and angle 3 is used to determine the constant ternary angle α_3. The LEO are now used in the iteration process for setting up the matrix equation values.

The Animation Graphics

The basic process of computer animation consists of drawing and

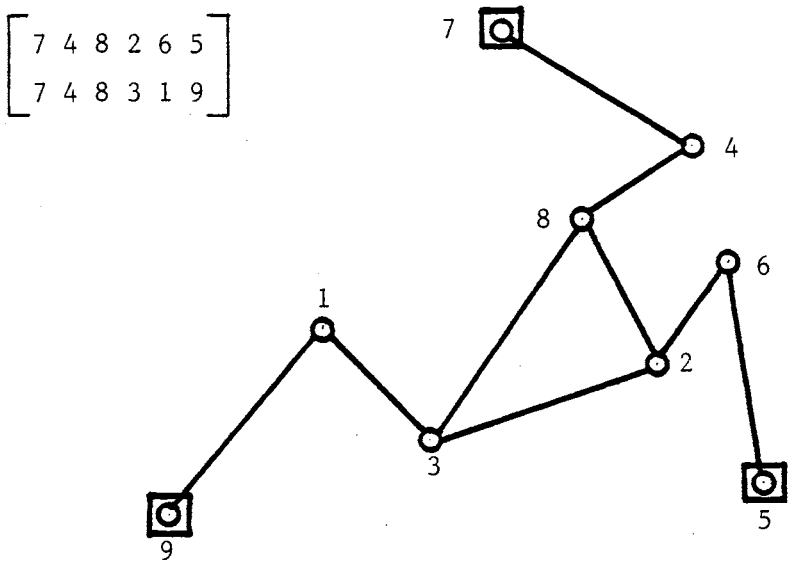


Figure 12. LEO as Ordering of Joint Numbers

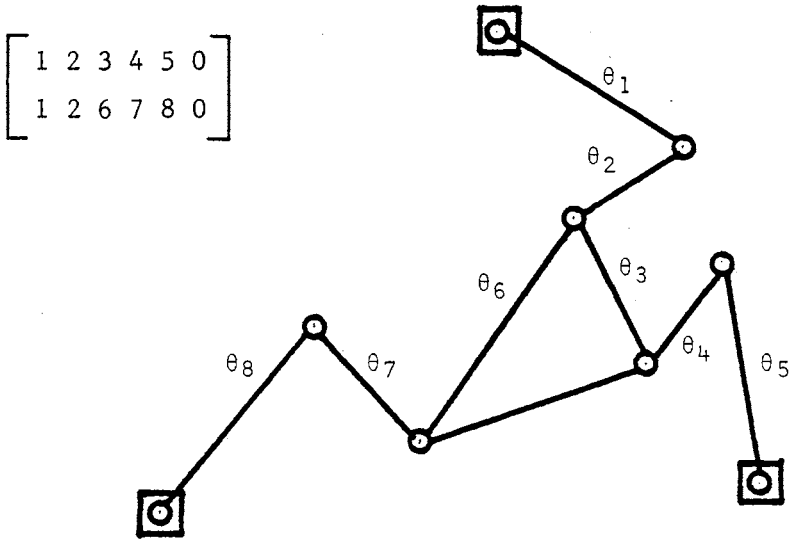


Figure 13. LEO as Ordering of Position Angles

erasing successive frames of the motion of an object. How this process is performed in the way of software and hardware has a great effect on the quality of animation. In addition, the speed at which the host computer communicates with the graphics terminal is another important factor.

The computer used was the Harris 800. This computer is a fast, powerful, virtual memory supermini computer. The terminal used was a Tektronix 4115B. The 4115B is a high resolution, color raster graphics terminal. This combination of computer and terminal was able to communicate at a very fast baudrate of 96200.

The successive positions of each specific robot animation were placed in an array and used at the end of the program during the animation process. The animation was not performed in real time so that the software would cause a minimum of slowdown. For each frame of motion, the program called in particular order graphics library routines MOVE and DRAW using these array elements as the arguments.

In the first attempt made at animation, each frame was drawn and then erased by calling a routine NEWPAG which erased the full terminal screen using the Tektronix hardware. This technique created two major problems:

1. The first links drawn were visible longer than the other links which caused the robot to appear to fade away to different degrees at different locations.

2. Since the full screen was erased for each frame, the titles, borders, and other stationary graphics were continually redrawn which slowed the animation and caused the screen to blink.

Using graphic segments solved both of these problems. A flow

diagram of how they were used in this application is given in Figure 14. In using segments, the Tektronix hardware draws each frame of motion almost instantaneously allowing all links to be visible for almost equal time intervals. In addition, each new frame is displayed an instant after the current frame is erased.

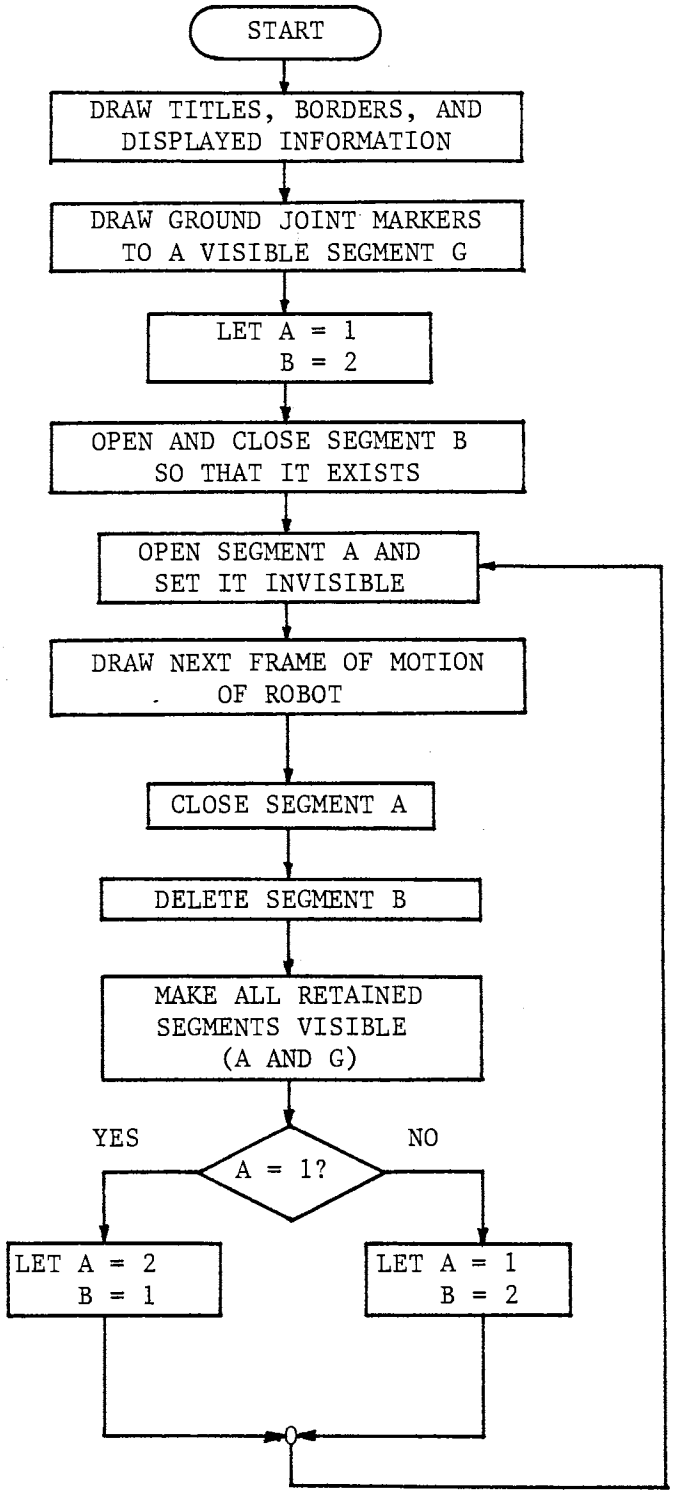


Figure 14. Use of Segments for Animation

CHAPTER IV

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT

Feasibility Determination for Specified End-Effector Position

The workspace of a robot is the region in space that is accessible by the robot's end-effector. Simply put, it is the space in which the robot can perform work. It is one of the most important specifications that a robot designer or user should consider (Tsai, 1981). Thus, a robot is more versatile if it has a large workspace. However, if the applications applied to a robot require a known workspace, there is no need to implement a likely larger, slower, and more expensive robot having an excessive workspace. Nevertheless, one should still take note that most robots work with much diminished efficiency when working near their workspace boundary.

The workspace for a planar robot is a planar region. When implemented in industrial applications, the Oklahoma Crawdad robot will probably serve as the regional positioning structure and have a specialized end-effector attached to the ternary link. As regional positioner, the Oklahoma Crawdad robot will provide the "gross motion" of the end-effector (Milenkovic, 1979). By not considering the orientation of the end-effector, the workspace studied is the primary workspace (Gupta and Roth, 1981). With the addition of an extendable end-effector having a line of action not lying in the plane of the robot, a

planar robot would be well suited for many pick-and-place, drilling, or welding applications.

The Oklahoma Crawdad robot, however, has the ability to move a rigid end-effector to a position within its workspace and provide some degree of orientation. For the rest of this chapter, the term robot will refer to the Oklahoma Crawdad robot.

In order to determine the workspace of the robot, one can specify an end-effector location and then determine whether or not it is attainable. The method used to perform this feasibility test considers the robot to be a 3R robot having two constraining dyads as shown in Figure 15.

With this concept, one first determines if the 3R robot can reach the desired position. If so, one then tests the dyads for compliance given the position and orientation of the ternary link which is the third link of the 3R robot. Thus, if these three conditions hold simultaneously, the specified position is considered to be within the workspace.

Since a 3R robot has three degrees of freedom and a specified planar position provides just two constraints (x and y coordinates), a 3R robot can reach a position within its own workspace in an infinite number of joint rotation combinations. Figure 16 shows the starting and ending θ_1 values for end-effector positions where

$$R_1+R_2+R_e \geq ||\underline{E}|| \geq R_2+R_e-R_1.$$

If $||\underline{E}|| < R_2+R_e-R_1$ then the starting θ_1 value was set to 0 and the ending value to 2π radians.

Dyad compliance would then be checked for incremental θ_1 values starting with $\theta_{1_{start}}$ and incremented by 0.1 radians until either compliance was met or θ_1 became greater than $\theta_{1_{end}}$.

For each θ_1 value not equal to a true θ_1 limit, two solutions for

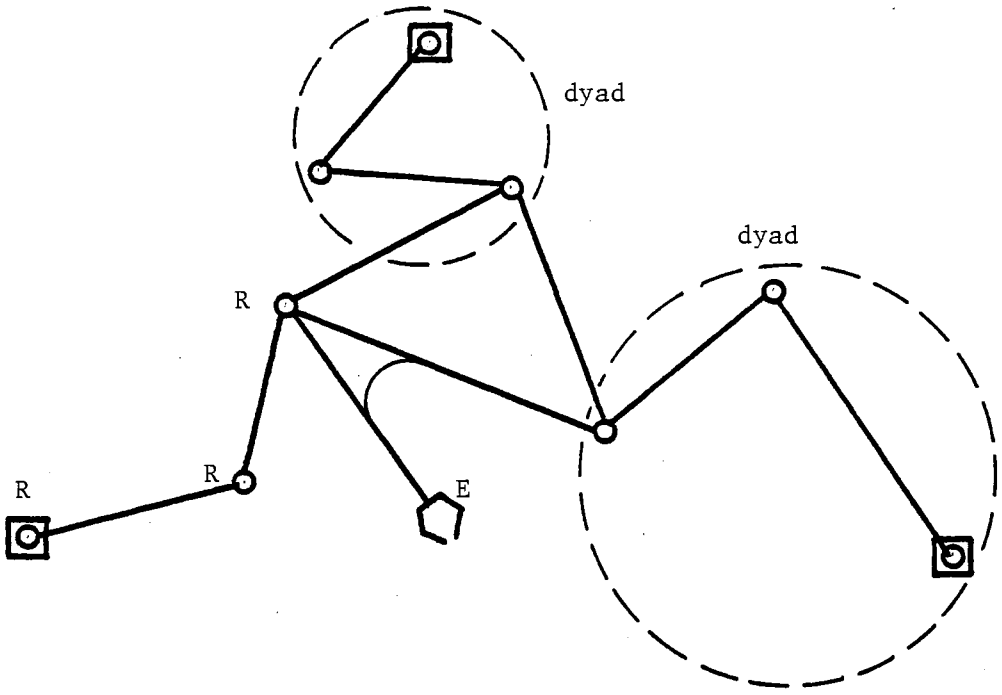


Figure 15. 3R Robot and Constraining Dyads of the Oklahoma Crawdad Robot

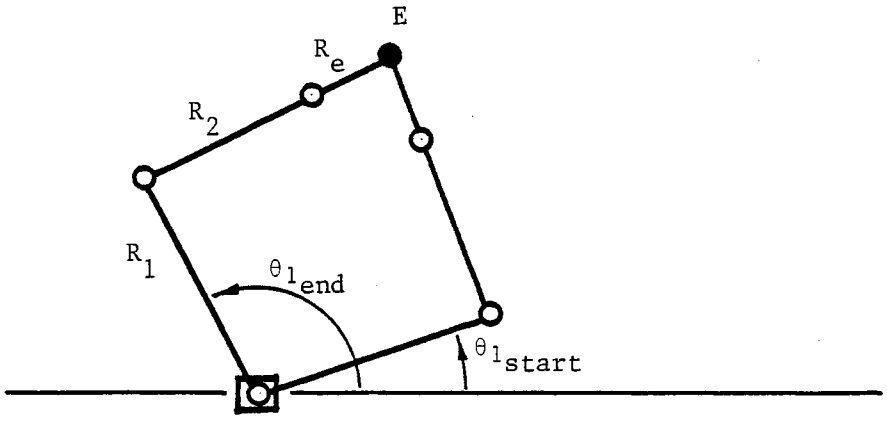


Figure 16. Range of Possible θ_1 Values for a Specified End-Effector Position

θ_2 and θ_e result as shown in Figure 17. The two solutions are given by the following equations.

$$x_{p_1} = R_1\cos\theta_1 \tag{4.1}$$

$$y_{p_1} = R_1\sin\theta_1 \tag{4.2}$$

$$\theta_2 = \tan^{-1}\left[\frac{y_e-y_{p_1}}{x_e-x_{p_1}}\right] \pm \cos^{-1}\left[\frac{R_2^2-R_e^2=(y_e-y_{p_1})^2+(x_e-x_{p_1})^2}{2R_2((y_e-y_{p_1})^2+(x_e-x_{p_1})^2)^{\frac{1}{2}}}\right] \tag{4.3}$$

$$\theta_e = \theta_2 \pm (\pi-\cos^{-1}\left[\frac{R_2^2+R_e^2-((y_e-y_{p_1})^2+(x_e-x_{p_1})^2)}{2R_2R_e}\right]) \tag{4.4}$$

Using one solution at a time, the positions of joints 3 and 6 are calculated by the following equations given in polar form.

$$x_{p_2}+iy_{p_2} = x_{p_1}+iy_{p_1}+R_2e^{i\theta_2} \tag{4.5}$$

$$x_{p_3}+iy_{p_3} = x_{p_2}+iy_{p_2}+R_2e^{i(\theta_e+\alpha_3+\alpha_6)} \tag{4.6}$$

$$x_{p_6}+iy_{p_6} = x_{p_2}+iy_{p_2}+R_6e^{i(\theta_e+\alpha_6)} \tag{4.7}$$

As shown in Figure 18, dyad compliance is then met if all of the following inequalities hold:

$$R_4+R_5 \geq ||\overline{P_3G_5}|| \geq |R_4-R_5|$$

$$R_7+R_8 \geq ||\overline{P_6G_8}|| \geq |R_7-R_8|$$

If the above are satisfied, the specified end-effector location is within the Oklahoma Crawdad robot's workspace.

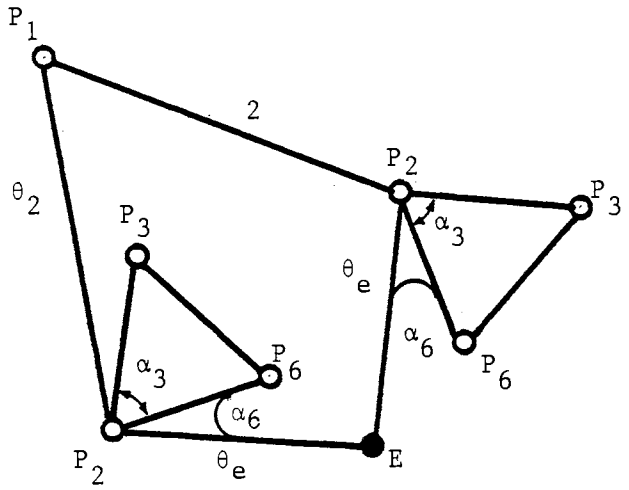


Figure 17. Two Solutions for θ_2 and θ_e for given θ_1

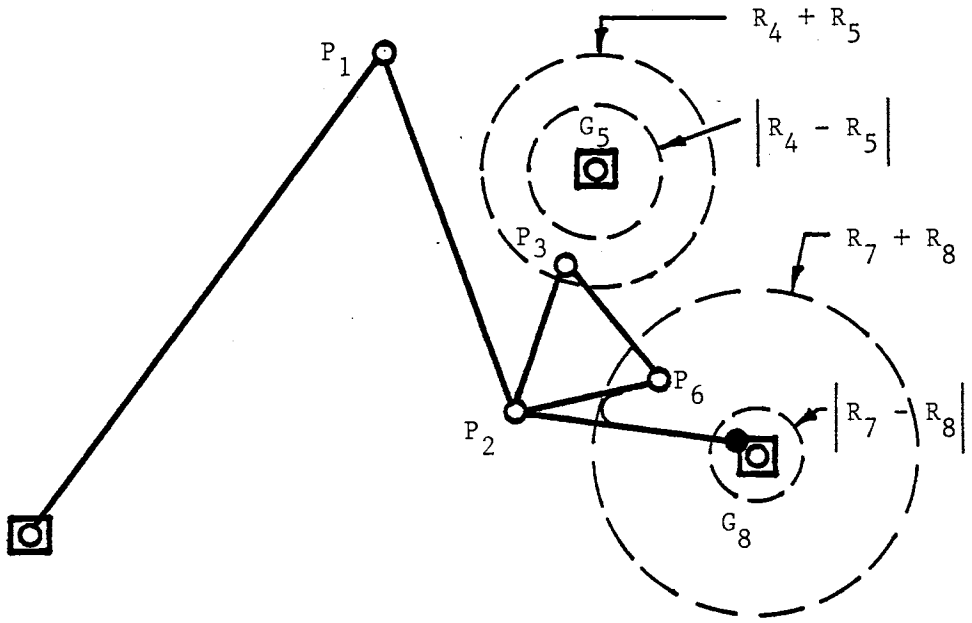


Figure 18. Dyad Compliance

The Computer Program

The Interactive Input

Since this program was written for the purpose of analyzing the workspace of the Oklahoma Crawdad robot, the program was user friendly. During the interactive input, an arbitrary Oklahoma Crawdad is displayed on the terminal screen having most of its links and markers in one color while certain links and markers appear in a contrasting color in order to aid the user to identify the input parameters.

The main interactive questions are listed below:

1. Default Data ?
2. Ground Joint Spacing Equilateral ?
3. Symmetrical Configuration ?
4. RE and THETA (deg) =?
5. Input Rotation Limits ?

The default data describes a symmetrical robot having equilateral ground spacing of 2.5 units, all link lengths equal to 1.0 unit (length of the ternary link is the length of its side), and no input rotation limits. The end-effector is centered on the ternary link.

If the user desires to center the end-effector on a ternary link having side length S, RE and THETA should be $S/\sqrt{3}$ units and 330 degrees respectively.

If the user chooses to specify input rotation limits, the program indicates that the MIN value should be greater than the MAX value if the link can rotate through the zero degree position. For example, if a link is confined to rotation within the cartesian quadrants I and IV, MIN would be 270 degrees and MAX would be 90 degrees. If these values are

reversed, the link will rotate in quadrants II and III.

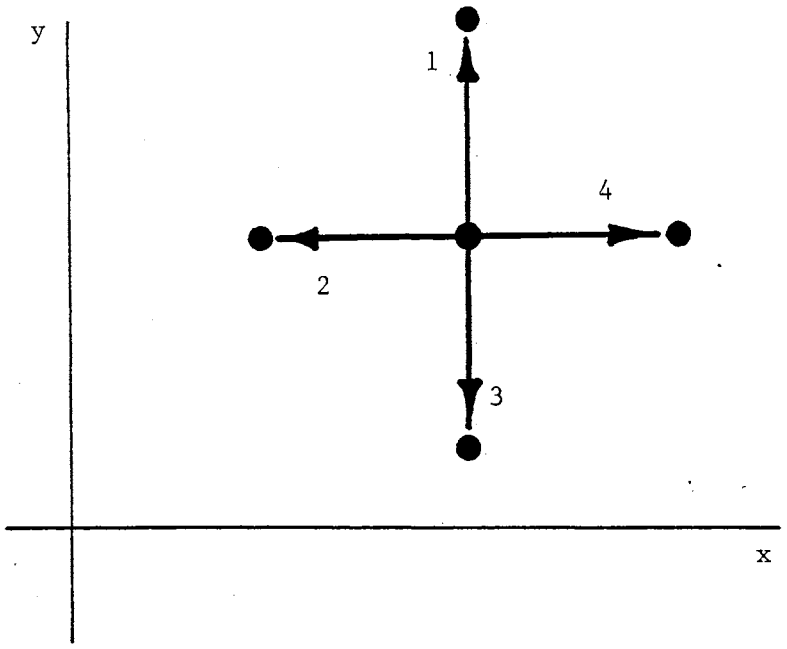
Generation of the Initial End-Effector Position

In order to begin the search for the workspace boundary, an initial position of the robot's end-effector must be known. So that the user would not be required to know or guess at such a point, the workspace program generates an initial position. Since only symmetrical robots having equilateral ground spacing were studied, the generation of this initial position was simple. Since the smallest workspace for these robots is a point at the centroid of the ternary ground link, a point in the workspace can be found on a vertical line passing through the centroid. With robots having a normalized 3R robot length of one unit, the highest y coordinate attainable is also one unit. Thus, the initial end-effector position is found by decrementing y from 1 along this line until a position is found to be feasible by the method described earlier in this chapter.

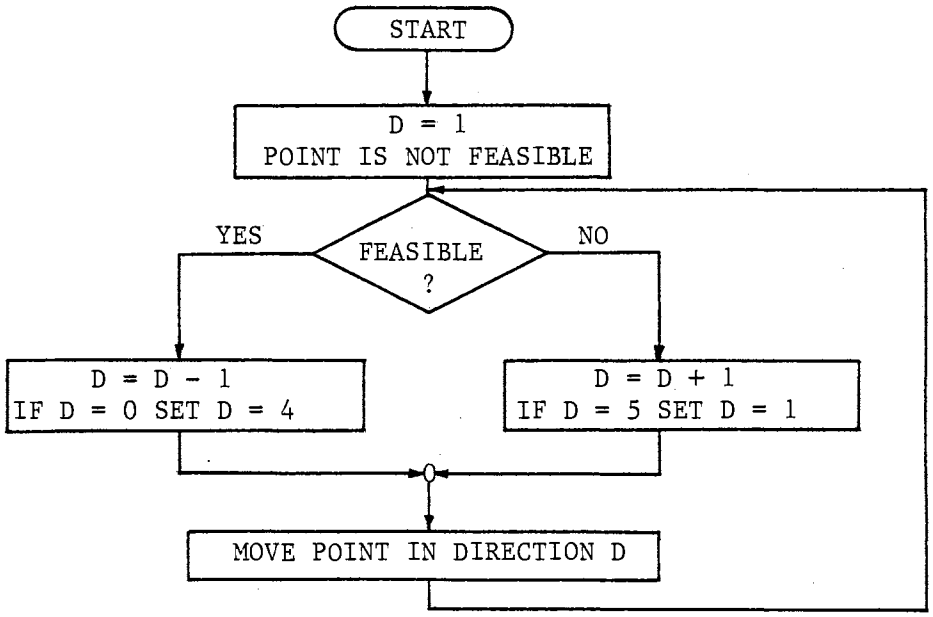
The Logic of Moving to and Along the Workspace Boundaries

Using an algorithm described by the flow diagram of Figure 19, the end-effector was moved around the outer boundary of the workspace in a counter clock-wise manner. This algorithm is similar to Cordray's Contouring Method used by Tsai and Soni (1982). The workspace program saves a portion of these positions of the robot and animates the robot moving along the boundary so that the user can fully visualize the concept of workspace.

For some combinations of link parameters, voïds exist in the workspace of the Oklahoma Crawdad robot just as they exist in 3R



a) Directions for "D"



b) Flow Diagram

Figure 19. Algorithm for Moving Around Outer Workspace Boundary

robots. (Tsai and Soni, 1981) There will exist either one or three voids and in both cases the void(s) will contain the ground positions. In order to determine void boundaries, points rising vertically from one ground point are checked until a feasible position is found. The end-effector is then moved until a feasible position is found. The end-effector is then moved around the void in a counter clock-wise manner using the outer boundary algorithm with reversed YES/NO directions. If this first void does not contain all of the ground points, the voids at the two remaining ground points are found.

The Workspace Area Calculation Method

The algorithm used to move the end-effector around the workspace boundary proved to be very useful in the additional task of determining the area of the workspace. By saving all of the points that were found to be just outside the workspace after attempting a move in the positive or negative y direction, that is D equal to 1 or 3, the workspace was approximated in Riemann sum manner as shown in Figure 20. By taking the appropriate differences of upper and lower boundary points and multiplying their sum by the constant delta value, the workspace area is approximated. If voids exist, their area is calculated in like manner and subtracted from the current area value.

The Effect of Link Parameters on the Workspace Area

With the workspace computer program developed, seventeen parameters exist that influence the workspace area. Twelve of these parameters describe the basic Oklahoma Crawdad robot and represent the link lengths. Two of these parameters describe the position of the end-effector

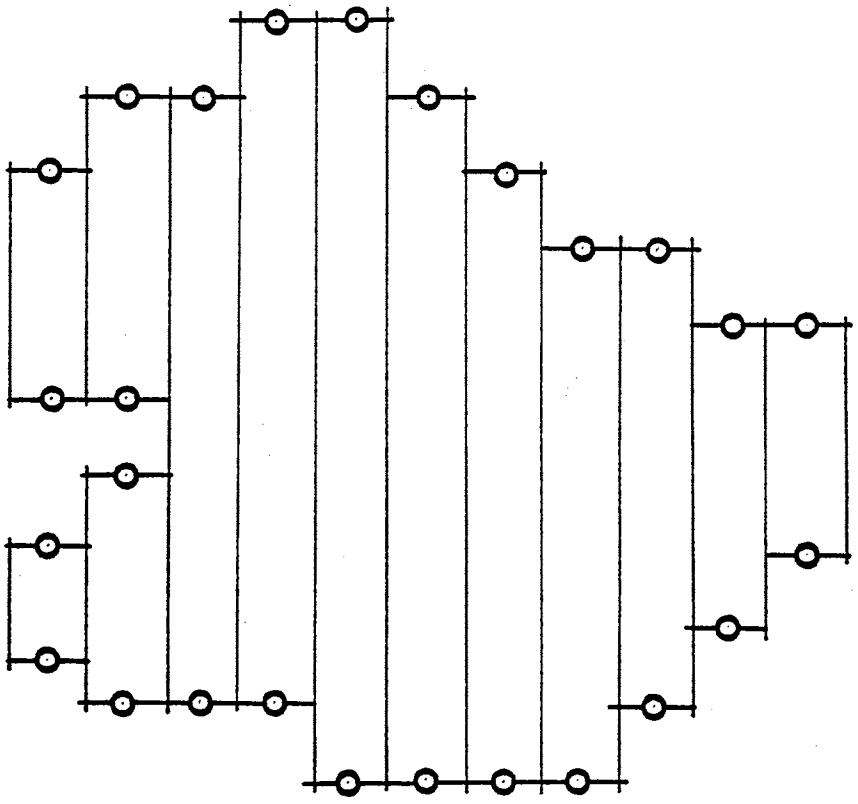


Figure 20. Representation of the Method Used to Calculate the Area Inside a Closed Curve

relative to the non-ground ternary link and the last three parameters represent limits on the rotations of the input links.

This large number of parameters produces a very large range of configuration variations. Only robots having symmetrical configurations were studied in this work. This reduced the number of parameters to four as shown in Figure 21. These parameters describe (1) the length of the input links, R_1, (2) the length of the secondary binary links, R_2, (3) the length of the side of the ternary link, R_3, and (4) the ground spacing, R_x. The end-effector is centered on the ternary link.

In order to compare the workspace areas for robots having different parameter values, the total link length of the imaginary internal 3R robot was set to one unit producing the following normalization:

$$R_1+R_2+R_e = 1$$

where $R_e = \frac{\sqrt{3}}{3} R_3$.

As shown in the example workspace shape chart of Figure 22, several general shapes exist when $\Delta = |R_1-R_2|$ and R_x are varied. The regional boundaries are approximated by the following equations:

$$\Delta_0 \approx R_e$$

$$_1\Delta_3 \approx R_e + \tfrac{1}{2}R_x$$

$$\Delta_d \approx 1+(\sqrt{3}-1)R_e-R_x$$

Although some areas not shown in Figure 22 exist that can be reached by certain robots if the robots are taken apart and reconfigured, they are small and it would be unrealistic to include them.

In the following paragraphs, general parameter effects will be noted and physical constraints will be discussed. These trends and constraints will then be used to determine the parameter values of the optimal Oklahoma Crawdad robot.

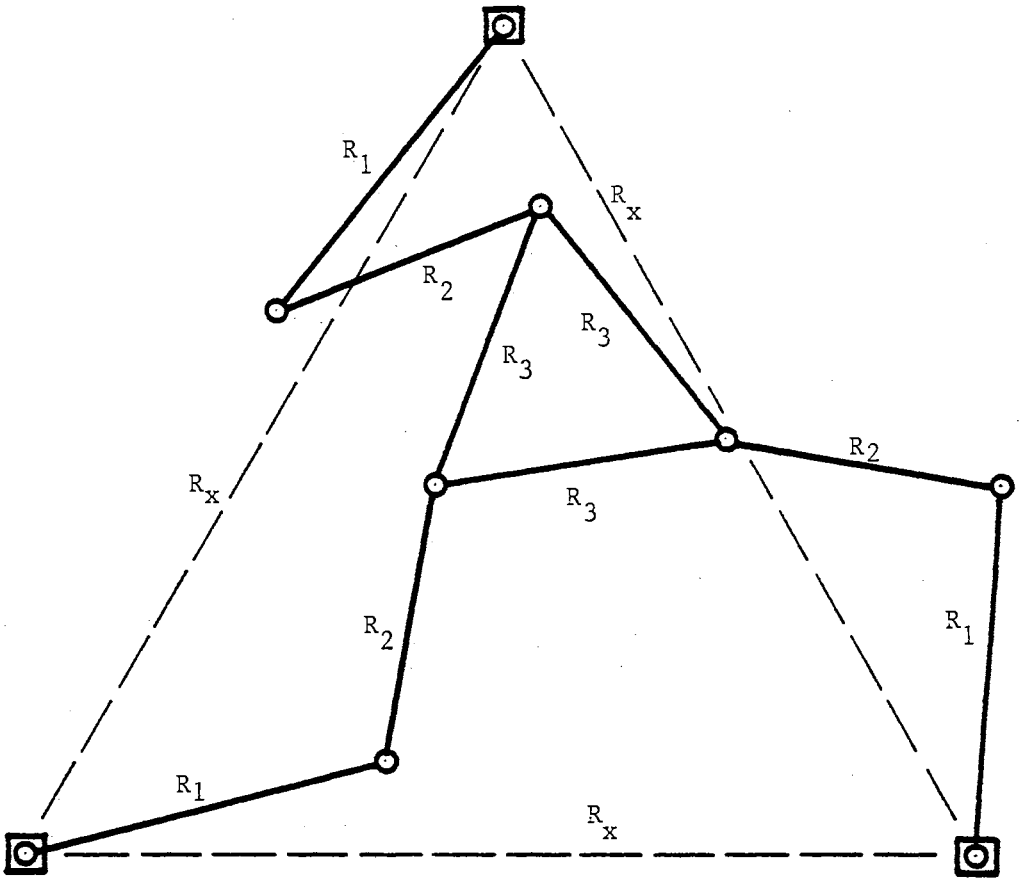


Figure 21. The Four Parameters of Symmetrical Configurations: R_1, R_2, R_3, and R_x.

Effect of Δ and R_x on the Workspace Shape
Of Oklahoma Crawdad Robots with $R_e = \frac{1}{6}$

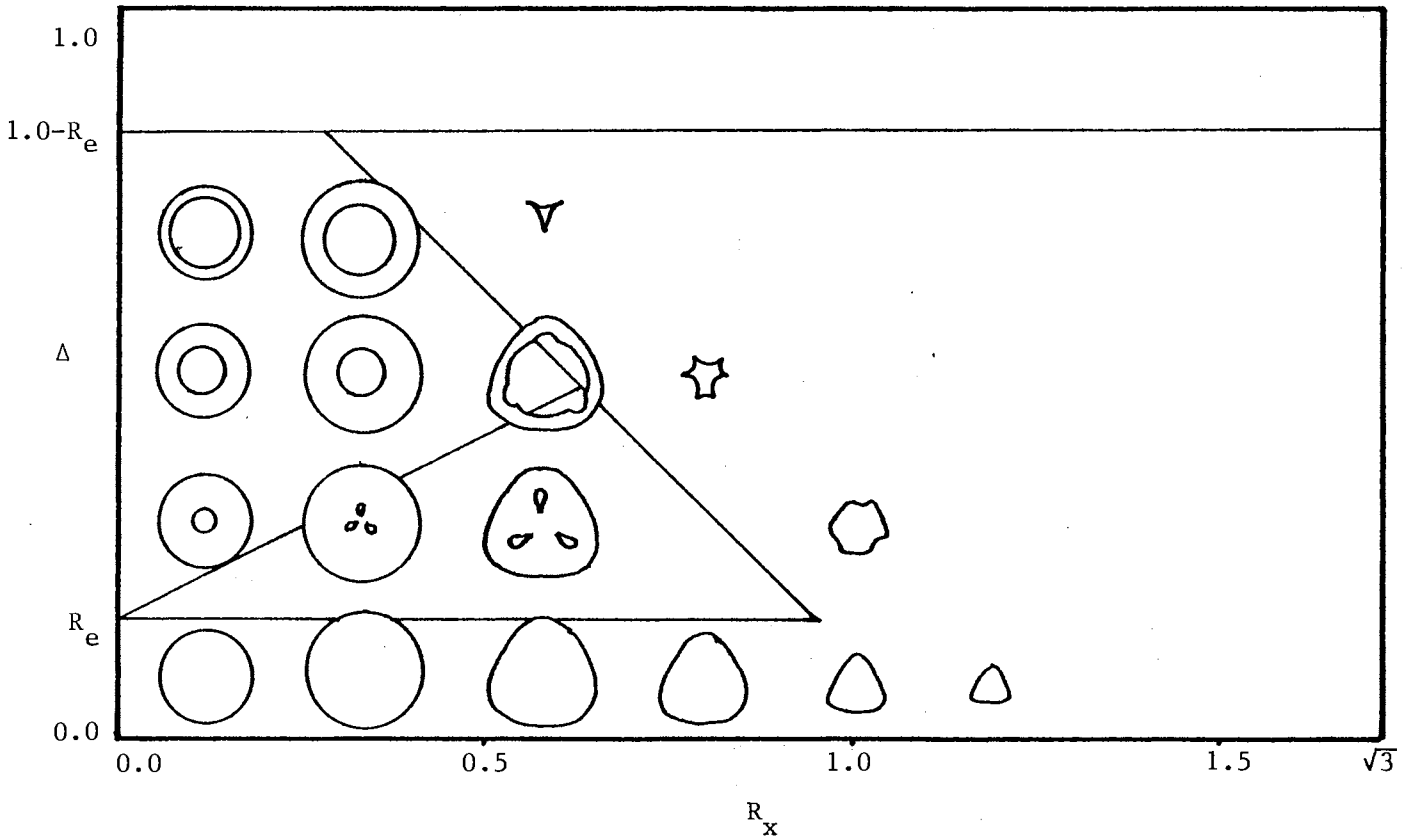


Figure 22. Effect of Δ and R_x on the Workspace Shape of Oklahoma Crawdad Robots with $R_e = \frac{1}{6}$

When Δ is greater than both Δ_0 and Δ_d, the workspace becomes degenerate. These configurations are not further considered.

The outer boundary of the workspace for non-degenerate configurations depends upon parameters R_e and R_x and is not affected by the Δ value. The area inside these outer boundaries increase with decreasing R_e and becomes maximum when $R_x = R_3$ as shown in Figure 23.

One or three voids will exist if Δ is greater than R_e and the total void area increases with increasing Δ. All configurations having Δ less than or equal to R_e will not have voids. Thus, Δ should be chosen less than R_e but not necessarily as small as possible.

When actually constructing the Oklahoma Crawdad robot, physical constraints will arise. The minimum number of these constraints that must be considered is one: noninterference of the input binary links. The geometry of this constraint is depicted in Figure 24 and is represented by the following equation:

$$R_x \geq 2R_1$$

The secondary binary links will not cause interference problems if they are formed at an angle as shown in Figure 25.

Figures 26 and 27 approximate the workspace shape regions for R_e of $\frac{1}{4}$ and $\frac{1}{3}$ respectively. The vertical dashed lines represent configurations having the optimal outer boundary ($R_x = R_3$) and the diagonal dashed lines represent the physical constraint boundary ($R_x = 2R_1$) where $R_1 \leq R_2$. The optimal solution to the parameter values R_1, R_2, and R_x for a given R_e value lies where $R_x = R_3$ or as close as possible to this condition while being both without voids ($\Delta \leq R_e$) and within the physical constraint (to the right of $R_x = 2R_1$). Each of these optimal solutions is indicated by an asterisk.

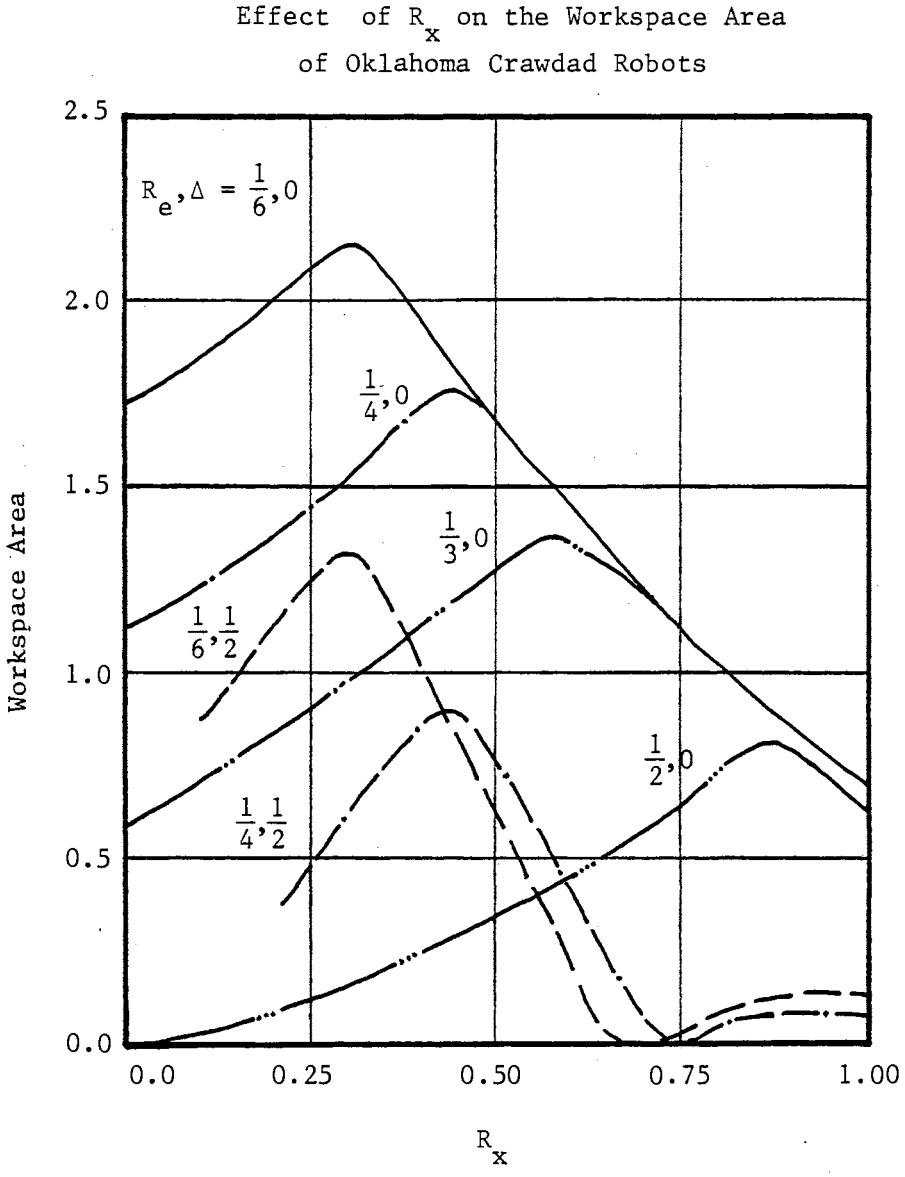


Figure 23. Effect of R_x on the Workspace Area of Oklahoma Crawdad Robots

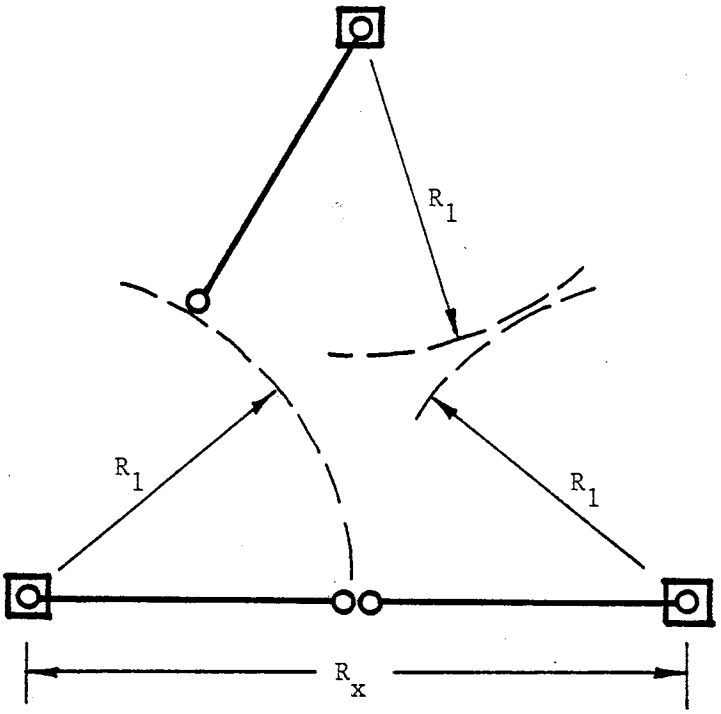


Figure 24. The Physical Constraint $R_x \geq 2R_1$.

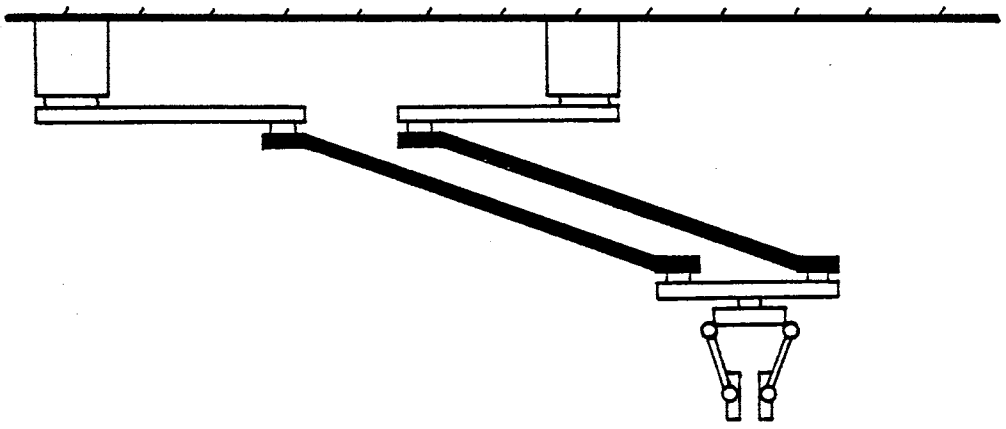


Figure 25. Secondary Binary Links Shown at an Angle from the Horizontal

Workspace Shape Regions for Oklahoma Crawdad Robots with $R_e = \frac{1}{4}$

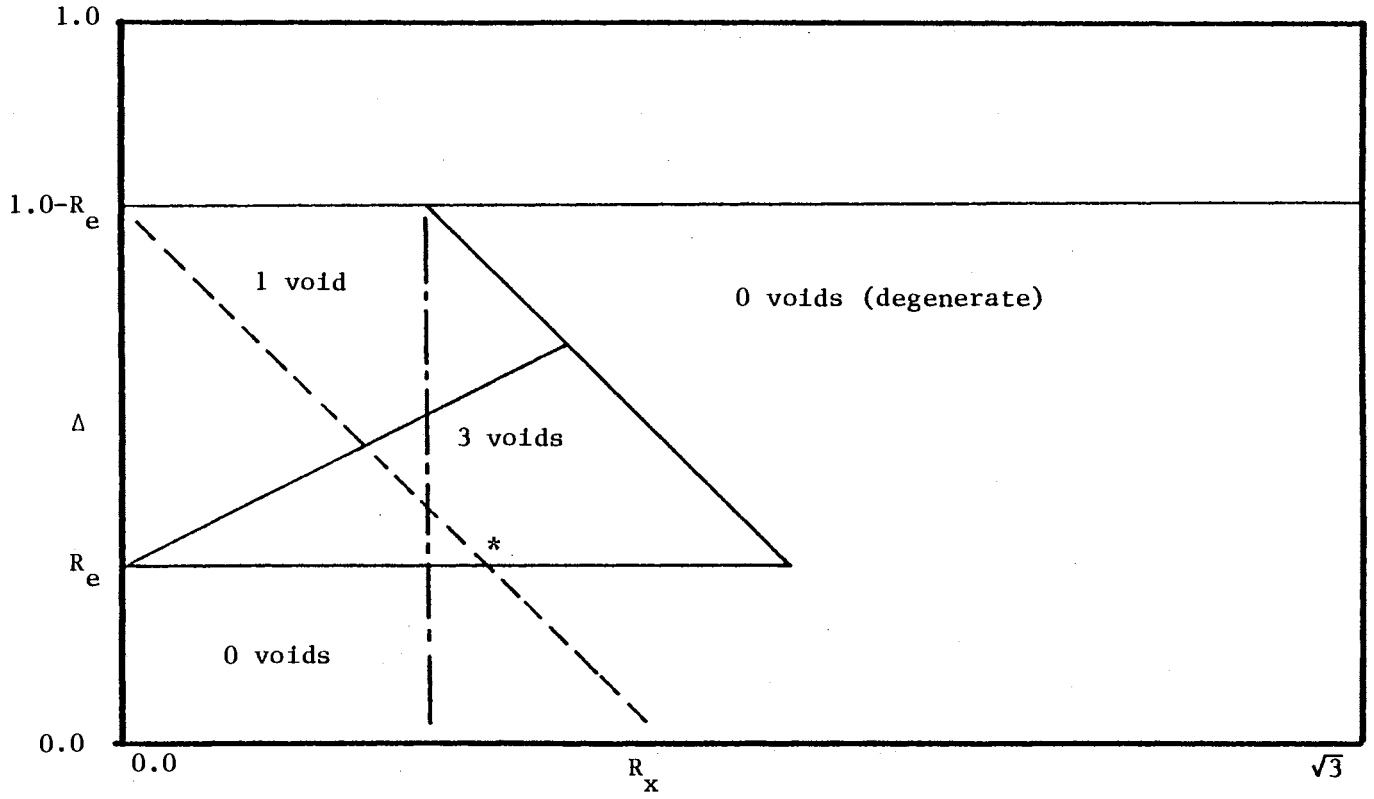


Figure 26. Workspace Shape Regions for Oklahoma Crawdad Robots with $R_e = \frac{1}{4}$. Asterisk indicates optimal design for given R_e.

Workspace Shape Regions for Oklahoma Crawdad Robots with $R_e = \frac{1}{3}$

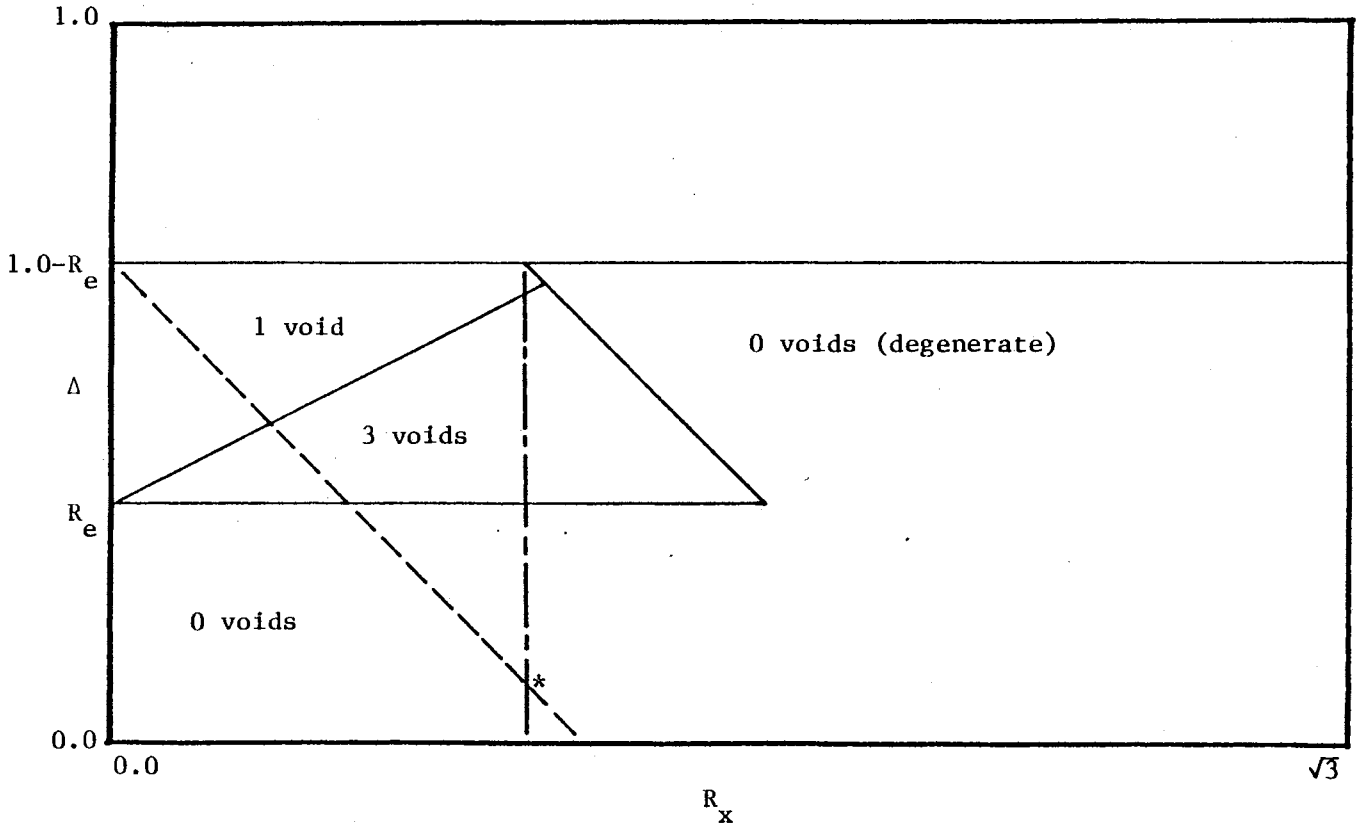


Figure 27. Workspace Shape Regions for Oklahoma Crawdad Robots with $R_e = \frac{1}{3}$. Asterisk indicates optimal design for given R_e.

The Oklahoma Crawdad robot having optimal workspace area is found by forcing the conditions $R_x = R_3$, $R_x = 2R_1$, and $\Delta = R_e$ $(R_1 \leq R_2)$ to occur simultaneously. Using these equations and the normalization equation, the optimal solution was determined and is shown in Figure 28. This robot has the following parameter values:

$R_1 = \frac{1}{2}(2\sqrt{3}-3) = 0.2321$

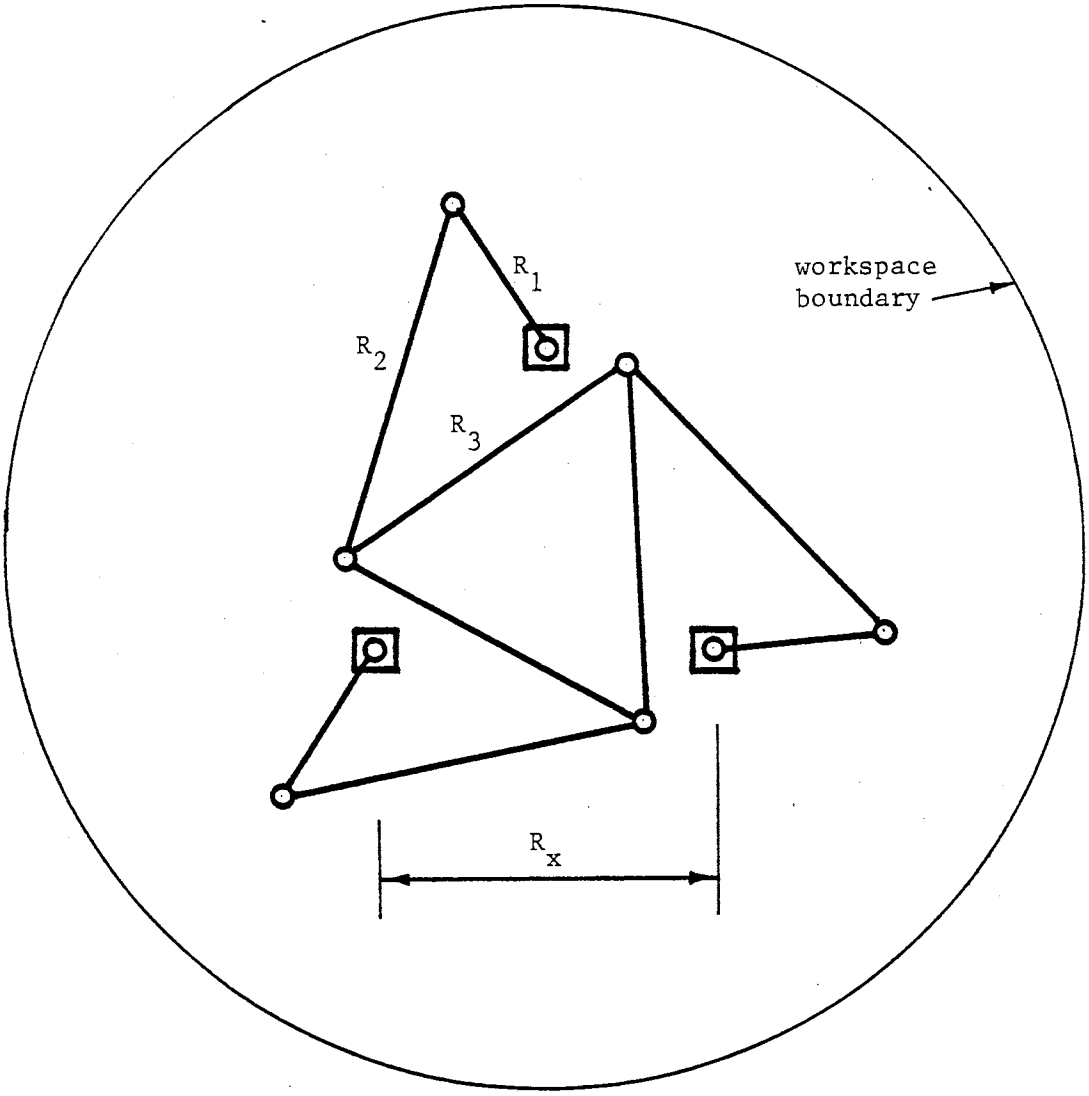
$R_2 = 0.5000$

$R_3 = (2\sqrt{3}-3) = 0.4641$

$R_x = (2\sqrt{3}-3) = 0.4641$

The boundary for symmetrical robots having $R_x = R_3$ is a circle with radius R_1+R_2. Thus, the workspace area for the above optimal robot is $\pi(\sqrt{3}-1)^2$ or 1.684 square units.

The Symmetrical Oklahoma Crawdad Robot
with Optimal Workspace



$$R_1 = \frac{1}{2}(2\sqrt{3}-3)$$

$$R_2 = 0.5000$$

$$R_3 = (2\sqrt{3}-3)$$

$$R_x = (2\sqrt{3}-3)$$

Figure 28. The Symmetrical Oklahoma Crawdad Robot with Optimal Workspace

CHAPTER V

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

This work describes a method that can be used to animate closed-loop planar robots having revolute joints and binary and ternary links. In addition, the effects of link parameters on the shape and area of the workspace of Oklahoma Crawdad robots having symmetrical configurations were studied.

It is recommended that the animation program be extended so that prismatic joints and links of any dimension may be considered. Also, user definable input functions could be implemented. Finally, non-planar robots could be animated.

For the workspace analysis, general link parameter effects are presented in Chapter IV. In studying these effects, it was found that the Oklahoma Crawdad robot having the largest workspace area for a given total 3R link length has parameters where $R_1 = (2\sqrt{3}-3)R_2 = \frac{1}{2}R_3 = \frac{1}{2}R_X$. This optimal robot has a workspace having a circular boundary of radius $(\sqrt{3}-1)$ and no internal void areas. It is noted that this configuration requires ground positions located "above" the working plane.

Due to the fact that this optimal design as $R_1 \neq R_2$, the control of this robot will not be straightforward. It is a recommendation, therefore, of this study that the "inverse problem" of point-path guidance for this robot be researched. The final recommendation of this work is that the workspace of nonsymmetrical configurations of the Oklahoma

Crawdad robot be studied. In particular, "platform" configurations which have colinear ground positions might prove very useful for certain industrial tasks.

REFERENCES CITED

Draganoiu, G. et al., "Computer Method for Setting Dynamical Model of an Industrial Robot with Closed Kinematic Chains," 6th International Conference on Industrial Robot Technology, Paris, France, June 9-11, 1982, pp. 371-375.

Gupta, K. C. and Roth, B., "Design Consideration for Manipulator Workspace," ASME Paper No. 810330, 1981.

Loeff, L. A. and Soni, A. H., "An Algorithm for Computer Guidance of a Manipulator in Between Obstacles," ASME, Journal of Engineering for Industry, 1975, pp. 836-842.

Milenkovic, V., "Computer Synthesis of Continuous Robot Motion," Proccedings 5th World Congress on the Theory of Machines and Mechanisms, Montreal, 1979, Vol. 2, pp. 1332-1335.

Paul, R. P., Robot Manipulators: Mathematics, Programming, and Control, The MIT Press, Cambridge, Massachusetts, 1983.

Soni, A. H., Mechanism Synthesis and Analysis, McGraw-Hill, New York, 1974.

Tsai, Y. C., "Synthesis of Robots/Manipulators for a Prescribed Working Space." (Unpublished Ph.D. dissertation, Oklahoma State University, 1981).

Tsai, Y. C. and Soni, A. H., "The Effect of Link Parameters on the Working Space of General 3R Robot Arms." (Unpublished Ph.D. dissertation, Oklahoma State University, 1981).

Tsai, Y. C. and Soni, A. H., "An Algorithm for the Workspace of a General n-R Robot," ASME Paper No. 82-DET-37. (Presented at the Design and Production Engineering Technical Conference, Washington, D. C., September 12-15, 1982).

APPENDIX A

LISTING OF THE COMPUTER PROGRAMS

```
*************************************************************************
*                                                                       *
*     CLOSED-LOOP PLANAR ROBOT ANIMATION PROGRAM                        *
*                                                                       *
*     1.  Animates closed-loop planar robots having revolute pairs      *
*         and binary and ternary links.                                 *
*     2.  Can display the path traced by a coupler point or             *
*         end-effector.                                                 *
*                                                                       *
*     This program was written by Bruce Sumpter during                  *
*     the school year 1984-1985 at Oklahoma State University.           *
*                                                                       *
*     This was the first program of two standing as the                 *
*     thesis work for the Master of Science Degree in the               *
*     department of Mechanical and Aerospace Engineering.               *
*                                                                       *
*************************************************************************
      CHARACTER*20 FNAME*20,XNAME*20,EOF*20,TITLE*23
      INTEGER B,
     * LEO(6,8),BMX(5),
     * RP,
     * KST(5),BST(5),TACON(16,2),TLINE(3,2),NB(5),
     * IG(5),QSP(5),QIF(5),Q(5),
     * R(12),T(12)
      REAL
     * RL(16),TA(16),
     * A(12,13),
     * P(2),
     * KP(17,2),WINDO(4),
     * QMX(5),G(5,2),Q0(5),
     * AL(12),
     * C(12),
     * G0(2),
     * X(5000),Y(5000)
      COMMON
     * /BLOCK1/ NL,N2,RL,TA
     * /BLOCK2/ A
     * /BLOCK3/ LEO,BMX
     * /BLOCK4/ RP,P,RLP,ALP
     * /BLOCK5/ KP,WINDO,KST,BST,TACON,TLINE,NTL,NB,KPP
     * /BLOCK6/ IG,QSP,QMX,G,QIF,Q,Q0,Q00
     * /BLOCK7/ R,T,AL,NITER,NTA
      DATA
     * EOF/'EOF                  '/
     * BIG/1E+9/
      OPEN(19,FILE='TEST')
      OPEN(20,FILE='ADATA')
      OPEN(21,FILE='PLOT')
      FOR I=1,4
       WRITE(3,*)
      END FOR
      CALL GRSTRT(4113,1)
      CALL HEADING
      WINDO(1)=BIG
      WINDO(2)=-BIG
      WINDO(3)=BIG
      WINDO(4)=-BIG
```

```
      WRITE(3,*)
      WRITE(3,*)
      WRITE(3,*)'CHOOSE DATA FILE:  1-LIBRARY  2-TEST'
      READ(0,*)I
      IF(I.EQ.2) THEN
       NFILE=19
       REWIND(19)
       READ(19,'(20X,A23)')TITLE
       WRITE(3,'(1X,A23)')TITLE
       CALL WAIT(2,2,I)
      ELSE
       NFILE=20
       WRITE(3,*)
       DO
        WRITE(3,*)
        WRITE(3,*)'ENTER LIBRARY FILE NAME'
        READ(0,'(A20)')FNAME
        REWIND(20)
        FOR I=1,10
         READ(20,*)
        END FOR
        DO
         FOR I=1,9
          READ(20,*)
         END FOR
         READ(20,'(A20,A23)')XNAME,TITLE
        UNTIL(XNAME.EQ.FNAME.OR.XNAME.EQ.EOF)
        IF(XNAME.EQ.EOF) WRITE(3,*)'*** FILE NOT FOUND ***'
       UNTIL(XNAME.NE.EOF)
      END IF
      CALL SETUP(NFILE,G0)
      EPSIL=0.0005
      ICMAX=5
      ITER=0
*************************************************************************
*                                                                       *
*     NEW POSITION ITERATION PROCESS                                    *
*                                                                       *
*************************************************************************
      DO ! ITER=1,NITER UNLESS LIMIT POS FOUND
       ITER=ITER+1
       L=NITER-ITER
       WRITE(3,*)L
       CALL NEWKP
       CALL NEWC(ITER,C)
       CALL NEXT(C,ICOUNT,EPSIL,ICMAX)
       EXIT DO IF(ICOUNT.EQ.ICMAX)
      UNTIL (ITER.EQ.NITER) ! DO ABOVE
      I=0
      IF(ICOUNT.EQ.ICMAX) I=1 ! LIMIT POS FOUND
      WRITE(21,*) I,I
      WRITE(21,*) I,I
*************************************************************************
*                                                                       *
*     ANIMATION OF THE ROBOT                                            *
*                                                                       *
*************************************************************************
      FOR I=1,4
       WRITE(3,*)
```

```
      END FOR
      WRITE(3,*)'PLEASE WAIT A MOMENT...'
      NP=0
      IF(RP.NE.0) NP=1
      IMAX=0
      FOR I=1,NL
       IMAX=IMAX+NB(I)+1
      END FOR
      IMAX=IMAX+2*(NTL+NP)
*
      REWIND(21)
      READ(21,*) X(1),Y(1)
      I=1
      DO
       I=I+1
       READ(21,*) X(I),Y(I)
      UNTIL (X(I).EQ.X(I-1).AND.Y(I).EQ.Y(I-1))
      NITER=(I-2)/IMAX
      LIMIT=X(I)
*
      X1=WINDO(1)
      X2=WINDO(2)
      Y1=WINDO(3)
      Y2=WINDO(4)
      DEL=ABS(((X2-X1)-(Y2-Y1))/2.0)
      IF((X2-X1).GT.(Y2-Y1)) THEN
       EDG=(X2-X1)/20.0
       X1=X1-EDG
       X2=X2+EDG
       Y1=Y1-EDG-DEL
       Y2=Y2+EDG+DEL
      ELSE
       EDG=(Y2-Y1)/20.0
       X1=X1-EDG-DEL
       X2=X2+EDG+DEL
       Y1=Y1-EDG
       Y2=Y2+EDG
      END IF
*
      DO
      WRITE(3,*)
      WRITE(3,*)
      WRITE(3,*)'ENTER WHOLE NUMBER < 10 FOR COLOR'
      READ(0,*) LC
      WRITE(3,*)
      WRITE(3,*)
      WRITE(3,*)'POSITION DISPLAY:  1-NORMAL  2-ALL POSITION'
      READ(0,*) IDIS
      WRITE(3,*)
      WRITE(3,*)
      WRITE(3,*)'ENTER NUMBER OF CYCLES'
      READ(0,*) IIEND
      IPATH=2
      IF(NP.EQ.1.AND.IDIS.EQ.1) THEN
       WRITE(3,*)
       WRITE(3,*)
       WRITE(3,*)'COUPLER PATH:  1-VISIBLE  2-NOT VISIBLE'
       READ(0,*) IPATH
      END IF
```

```
      WRITE(3,*)
      WRITE(3,*)
      WRITE(3,*)'SPOOLING TO DISK ?  1-NO  2-YES'
      READ(0,*) ISPOOL
      IF(ISPOOL.EQ.1) THEN
       CALL DELSEG(-1)
       CALL NEWPAG
      ELSE
       WRITE(3,*)
       WRITE(3,*)
       WRITE(3,*)'GIVE SPOOL COMMANDS ; <CR> ; <CR> AFTER SPOOLING'
       READ(0,*)
       CALL GRSTRT(4113,1)
      END IF
      FOR I=1,4
       WRITE(3,*)
      END FOR
      CALL HEADING
*
      CALL WINDOW(0.0,4.0,0.0,1.0)
      CALL VWPORT(87.0,110.0,65.0,70.0)
      CALL TXTCLR(LC)
      CALL MOVE(0.0,0.0)
      CALL TXSIZE(1,0.0,0.0)
      CALL TEXT(23,TITLE)
      CALL WINDOW(X1,X2,Y1,Y2)
      CALL VWPORT(10.0,85.0,15.0,90.0)
      W=EDG/10.0
      CALL BORDER(X1+W,X2-W,Y1+W,Y2-W)
      I=NL+1
      W=EDG/2.0
      G(I,1)=G0(1)
      G(I,2)=G0(2)
      CALL GROUND(I,G,W)
      CALL LINCLR(LC)
      CALL OPNSEG(1)
      CALL CLOSEG(1)
      IF(IDIS.EQ.1) THEN
       J=-1
       JPATH=3
      ELSE
       J=0
      END IF
*
      FOR II=1,IIEND
      I=0
      IF(II.EQ.2) KPATH=JPATH
*
      FOR ITER=1,NITER
       IF(IDIS.EQ.1) THEN
        J=-J
       ELSE
        J=J+1
       END IF
       CALL SETVIS(-2,.FALSE.)
       CALL OPNSEG(2+J)
       FOR L=1,NL
        I=I+1
        CALL MOVE(X(I),Y(I))
```

```
          FOR K=1,NB(L)
           I=I+1
           CALL DRAW(X(I),Y(I))
          END FOR
         END FOR
         FOR K=1,NTL
          I=I+1
          CALL MOVE(X(I),Y(I))
          I=I+1
          CALL DRAW(X(I),Y(I))
         END FOR
         IF(IPATH.EQ.1) THEN
          I=I+1
          CALL MOVE(X(I),Y(I))
          I=I+1
          CALL DRAW(X(I),Y(I))
          CALL CLOSEG(2+J)
          IF(ITER.GT.1.OR.II.GT.1) THEN
           JPATH=JPATH+1
           CALL OPNSEG(JPATH)
           CALL LINCLR(5)
           CALL MOVE(P(1),P(2))
           CALL DRAW(X(I),Y(I))
           CALL CLOSEG(JPATH)
           CALL LINCLR(LC)
          END IF
          P(1)=X(I)
          P(2)=Y(I)
         ELSE
          IF(NP.EQ.1) I=I+2
          CALL CLOSEG(2+J)
         END IF
         IF(IDIS.EQ.1) THEN
          CALL DELSEG(2-J)
          CALL SETVIS(2,.FALSE.)
          CALL SETVIS(2,.TRUE.)
          IF(IPATH.EQ.1.AND.ITER.GT.1) THEN
            CALL SETVIS(JPATH,.TRUE.)
          END IF
         END IF
         CALL SETVIS(2+J,.TRUE.)
         IF(ITER.EQ.1.AND.II.EQ.1.AND.ISPOOL.EQ.0) CALL WAIT(3,2,KREQ)
        END FOR
*
        END FOR
*
        IF(IPATH.EQ.1) THEN
         FOR I=4,KPATH
          CALL SETVIS(I,.FALSE.)
          CALL SETVIS(I,.TRUE.)
         END FOR
        END IF
        IF(ISPOOL.EQ.2) READ(0,*)
        IF(LIMIT.EQ.1) THEN
         WRITE(3,*)'LIMIT POSITION FOUND'
         CALL WAIT(3,2,KREQ) ! 3 SECOND PAUSE
        END IF
        WRITE(3,*)'CHOOSE:  1  CONTINUE WITH SAME ROBOT'
        WRITE(3,*)'         2  CHOOSE NEW ROBOT'
```

```
      WRITE(3,*)'          3  QUIT'
      READ(0,*) I
      UNTIL (I.NE.1)
      IF(I.NE.3) CALL CHAIN(8HANIMAT:X)
      CALL DELSEG(-1)
      CALL NEWPAG
      CALL GRSTOP
      STOP
      END
*
*     END OF MAIN PROGRAM
```

```
*************************************************************************
*                                                                       *
*     SUBROUTINE LIBRARY FOR THE ANIMATION PROGRAM                      *
*                                                                       *
*************************************************************************
*
*     SETUP DETERMINES THE LEO (LOOP EQUATION ORDERS)
*     AND OTHER PARAMETER VALUES
*
      SUBROUTINE  SETUP(NFILE)
      INTEGER B,
     * LEO(6,8),BMX(5),
     * RP,
     * KST(5),BST(5),TACON(16,2),TLINE(3,2),NB(5),
     * IG(5),QSP(5),QIF(5),Q(5),
     * TP,PRT(5),
     * R(12),T(12),RW(12),TW(12),
     * HLN,CHEK,LINK(14,3),GKP(5),LEL(5,7),TERM(5)
      REAL
     * RL(16),TA(16),
     * P(2),
     * KP(17,2),WINDO(4),
     * QMX(5),G(5,2),Q0(5),
     * CW(12),
     * AL(12),ALW(12)
      COMMON
     * /BLOCK1/ NL,N2,RL,TA
     * /BLOCK3/ LEO,BMX
     * /BLOCK4/ RP,P,RLP,ALP
     * /BLOCK5/ KP,WINDO,KST,BST,TACON,TLINE,NTL,NB,KPP
     * /BLOCK6/ IG,QSP,QMX,G,QIF,Q,Q0,Q00
     * /BLOCK7/ R,T,AL,NITER,NTA
      DATA NLK,NGKP,NIG/3*0/
*
      READ(NFILE,*)KP,LINK,GKP,IG,QSP,QMX,RP,P
      DO
       NLK=NLK+1
      UNTIL (NLK.GT.14.OR.LINK(NLK,1).EQ.0)
      NLK=NLK-1
      DO
       NGKP=NGKP+1
       IF(NGKP.LE.5.AND.IG(NGKP).NE.0) NIG=NIG+1
      UNTIL (NGKP.GT.5.OR.GKP(NGKP).EQ.0)
      NGKP=NGKP-1
*
      CHEK=0
      NITER=35
      FOR I=1,NIG
       IF(QSP(I).EQ.1) NITER=143
       IF(QSP(I).EQ.2.AND.NITER.NE.143) NITER=71
       IF(ABS(QMX(I)).GT.360.0.OR.QSP(I).GT.3) CHEK=1
       IF(ABS(QMX(I)).GT.180.0.AND.QSP(I).GT.1) CHEK=1
       IF(ABS(QMX(I)).GT.90.0.AND.QSP(I).GT.2) CHEK=1
      END FOR
      IF(CHEK.EQ.1) THEN
       WRITE(3,*)' IMPROPER INPUT SPEED/MAX FOUND'
       STOP
```

```
      END IF
*
      FOR I=1,NGKP
       CALL WINDOE(KP(GKP(I),1),KP(GKP(I),2),WINDO)
      END FOR
      DELW=WINDO(2)-WINDO(1)
*
      NL=NGKP-1
      N2=2*NL
      NTERM=0
      LEO(1,1)=IG(1)
      KST(1)=1
      BST(1)=1
      B=1
      HLN=1
      DO ! BRANCH
       B=B+1
       NLE=HLN
       FOR N=1,NLE ! LOOP
*
        CHEK=0
        FOR I=1,NTERM
         IF(N.EQ.TERM(I)) CHEK=1
        END FOR
*
        IF(CHEK.EQ.0) THEN ! LOOP NOT TERM
*
        FOR I=1,NLK ! LINK
        FOR J=1,3 ! KP OF LINK
         IF(LEO(N,B-1).EQ.LINK(I,J)) THEN ! CON LINK FOUND
          CHEK=0
          FOR K=1,B-2
           IF(I.EQ.LEL(N,K)) CHEK=1
          END FOR
          IF(CHEK.EQ.0) THEN ! LINK IS NEW
           LEL(N,B-1)=I
           IF(LINK(I,3).EQ.0) THEN ! BINARY
            LEO(N,B)=LINK(I,3-J)
            CHEK=1
           ELSE ! TERNARY
            HLN=HLN+1
            KST(HLN)=LEO(N,B-1)
            BST(HLN)=B-1
            FOR K=1,B-1
             LEO(HLN,K)=LEO(N,K)
             LEL(HLN,K)=LEL(N,K)
            END FOR
            LEO(N,B)=LINK(I,2)
            LEO(HLN,B)=LINK(I,3)
            IF(J.EQ.2) LEO(N,B)=LINK(I,1)
            IF(J.EQ.3) LEO(HLN,B)=LINK(I,1)
            CHEK=2
           END IF
*
           FOR KK=1,CHEK
            NN=N
            IF(KK.EQ.2) NN=HLN
            FOR K=1,NGKP
             IF(LEO(NN,B).EQ.GKP(K)) THEN ! TERM
```

```
              NTERM=NTERM+1
              TERM(NTERM)=NN
              BMX(NN)=B-1
              EXIT DO IF(NTERM.EQ.NL) ! ALL LOOPS FOUND
             END IF
            END FOR
           END FOR
*
          END IF
         END IF
        END FOR ! J
        END FOR ! I
        END IF
       END FOR ! N
      UNTIL (B.GE.10) ! DO ABOVE
*
      NTA=0
      NTL=0
      FOR N=1,NL ! LOOP
       FOR B=1,BST(N)-1 ! BRANCH
        KP1=LEO(N,B)
        KP2=LEO(N,B+1)
        FOR I=1,NTA ! RENUMBER TO TA SUBS
         IF(KP1.EQ.TACON(I,1).AND.KP2.EQ.TACON(I,2)) LEO(N,B)=I
        END FOR
       END FOR
       FOR B=BST(N),BMX(N) ! NEW TA
        NTA=NTA+1
        KP1=LEO(N,B)
        LEO(N,B)=NTA ! RENUMBER
        KP2=LEO(N,B+1)
        TACON(NTA,1)=KP1
        TACON(NTA,2)=KP2
        FOR I=1,NTA-1
         IF(KP1.EQ.TACON(I,1)) THEN ! TERNARY
          NTL=NTL+1
          TLINE(NTL,1)=TACON(I,2)
          TLINE(NTL,2)=KP2
         END IF
        END FOR
        X1=KP(KP1,1)
        Y1=KP(KP1,2)
        X2=KP(KP2,1)
        Y2=KP(KP2,2)
        CALL POLAR(X1,Y1,X2,Y2,RL(NTA),TA(NTA))
       END FOR ! B
       G(N,1)=KP(LEO(N,BMX(N)+1),1)
       G(N,2)=KP(LEO(N,BMX(N)+1),2)
       FOR I=2,NIG
        IF(LEO(N,BMX(N)+1).EQ.IG(I)) THEN ! INPUT
         QIF(N)=I
         Q(N)=LEO(N,BMX(N))
         Q0(N)=TA(Q(N))
        END IF
       END FOR
       LEO(N,BMX(N)+1)=0
      END FOR ! N
      Q00=TA(1)
*
```

```
*     R & T NUMBERING
*
      J=N2+1
      FOR L=1,NL
       J=J-2
       IF(Q(L).NE.0) THEN
        R(J)=LEO(L,BMX(L)-1)
       ELSE
        R(J)=LEO(L,BMX(L))
       END IF
      END FOR
*
      R(N2)=2
      IF(R(N2-1).EQ.4) T(N2-2)=3
*
      B=2
      J=N2
      FOR L=2,NL
       IF(LEO(L,B-1).NE.LEO(L-1,B-1)) B=B-1
       J=J-2
       IF(LEO(L,B).EQ.R(J+2)) B=B+1
       IF(LEO(L,B+1).EQ.R(J-1)) THEN
        R(J)=LEO(L,B)
       ELSE
        IF(T(J).NE.0) THEN
         R(J)=LEO(L,B)
         T(J-2)=LEO(L,B+1)
        ELSE
         K=J+2
         IF(LEO(L-1,B).EQ.R(J+1)) K=J+1
         T(K)=LEO(L,B)
         R(J)=LEO(L,B+1)
         B=B+1
        END IF
       END IF
      END FOR
*
*     ALPHA
*
      FOR J=1,N2
       AL(J)=0.0
       IF(T(J).NE.0) THEN
        AL(J)=TA(T(J))-TA(R(J))
        CALL MAX2PI(AL(J))
       END IF
      END FOR
*
*     COUPLER/END-EFFECTOR
*
      IF(RP.NE.0) THEN
      L=0
      J=0
      DO
       L=L+1
       FOR B=BST(L),BMX(L)
        IF(RP.EQ.LEL(L,B).AND.J.EQ.0) THEN
         J=L
         K=LEO(L,B)
         FOR I=1,N2
```

```
          IF(K.EQ.T(I)) K=I
         END FOR
         RP=R(K) ! RP IS NOW AN R ANGLE
         TP=T(K)
         ALTP=AL(K)
         K=B-1
        END IF
       END FOR
      UNTIL(J.NE.0) ! DO ABOVE
*
      FOR B=1,K
       LEO(NL+1,B)=LEO(J,B) ! P LOOP SET UP
      END FOR
*
      KPP=TACON(RP,1)
      CALL POLAR(KP(KPP,1),KP(KPP,2),P(1),P(2),RLP,ALP)
      ALP=ALP-TA(RP)
      END IF
      FOR L=1,NL
       NB(L)=BMX(L)-BST(L)+1
      END FOR
      RETURN
      END
*
*
*     NEXT DETERMINES THE NEXT DEL THETA VECTOR
*
      SUBROUTINE  NEXT(C,ICOUNT,EPSIL,ICMAX)
      INTEGER B,
     * LEO(6,8),BMX(5),
     * R(12),T(12)
      REAL
     * RL(16),TA(16),
     * A(12,13),
     * AL(12),
     * C(12)
      COMMON
     * /BLOCK1/ NL,N2,RL,TA
     * /BLOCK2/ A
     * /BLOCK3/ LEO,BMX
     * /BLOCK7/ R,T,AL,NITER,NTA
      ICOUNT=0
      DO
       CALL NEWA(C)
       CALL MATRIX(A,N2)
       CALL NEWRTA(DELMX)
       ICOUNT=ICOUNT+1
      UNTIL (DELMX.LT.EPSIL.OR.ICOUNT.EQ.ICMAX) ! DO ABOVE
      CALL NEWTTA
      RETURN
      END
*
      SUBROUTINE  POLAR(X1,Y1,X2,Y2,RL,TA)
      DATA PI/3.14159/
      RL=SQRT(SQR(X1-X2)+SQR(Y1-Y2))
      IF(X1.NE.X2) THEN
       QUAD=0
       IF(X2.LT.X1) QUAD=PI
       TA=ATAN((Y2-Y1)/(X2-X1))+QUAD
```

```
 IF(TA.LT.0.0) CALL MAX2PI(TA)
ELSE
 IF(Y2.GT.Y1) TA=0.5*PI
 IF(Y2.LT.Y1) TA=1.5*PI
END IF
RETURN
END
*
FUNCTION  QFUN(NQ,IT,QSP,QMX)
INTEGER QSP(5)
REAL QMX(5),DEL(3)
DATA DEL/2.5,5.0,10.0/ RAD/57.296/
IF(QMX(NQ).NE.360.0) THEN
 QFUN=((1.0-COS(IT*DEL(QSP(NQ))/RAD))/2.0)*QMX(NQ)/RAD
ELSE
 QFUN=IT*DEL(QSP(NQ))/RAD
END IF
RETURN
END
*
SUBROUTINE  WINDOE(X,Y,WINDO)
REAL WINDO(4)
IF(X.LT.WINDO(1)) WINDO(1)=X
IF(X.GT.WINDO(2)) WINDO(2)=X
IF(Y.LT.WINDO(3)) WINDO(3)=Y
IF(Y.GT.WINDO(4)) WINDO(4)=Y
RETURN
END
*
SUBROUTINE  MAX2PI(ANGLE)
TWOPI=6.28318
DO
 IF(ANGLE.GE.TWOPI) ANGLE=ANGLE-TWOPI
 IF(ANGLE.LT.0.0)   ANGLE=ANGLE+TWOPI
UNTIL (ANGLE.GE.0.0.AND.ANGLE.LT.TWOPI)
RETURN
END
*
FUNCTION  SQR(X)
SQR=X*X
RETURN
END
*
SUBROUTINE  MATRIX(A,N)
REAL A(12,13)
INTEGER R(12),C(12)
FOR K=1,N
 PIV=0.0
 FOR I=1,N
  CHEK=0
  IF(K.GT.1) THEN
   FOR L=1,K-1
    IF(I.EQ.R(L)) CHEK=1
   END FOR
  END IF
  IF(CHEK.EQ.0) THEN
   FOR J=1,N
    IF(ABS(A(I,J)).GT.ABS(PIV)) THEN
     PIV=A(I,J)
```

```
            R(K)=I
            C(K)=J
           END IF
          END FOR
         END IF
        END FOR ! I
        FOR J=1,N+1
         A(R(K),J)=A(R(K),J)/PIV
        END FOR
        FOR I=1,N
         IF(I.NE.R(K)) THEN
          BMULT=A(I,C(K))
          SING=0.0
          FOR J=1,N+1
           A(I,J)=A(I,J)-BMULT*A(R(K),J)
           SING=SING+ABS(A(I,J))
          END FOR
          IF(SING.EQ.0.0) THEN
           WRITE(3,*)'MATRIX IS SINGULAR'
           STOP
          END IF
         END IF
        END FOR
       END FOR ! K
       FOR K=1,N
        A(C(K),1)=A(R(K),N+1)
       END FOR
       FOR I=1,N
        FOR J=2,N+1
         A(I,J)=0.0
        END FOR
       END FOR
       RETURN
       END
*
*
*
       SUBROUTINE  NEWKP
       INTEGER B,
      * LEO(6,8),BMX(5),
      * RP,
      * KST(5),BST(5),TACON(16,2),TLINE(3,2),NB(5)
       REAL
      * RL(16),TA(16),
      * P(2),
      * KP(17,2),WINDO(4)
       COMMON
      * /BLOCK1/ NL,N2,RL,TA
      * /BLOCK3/ LEO,BMX
      * /BLOCK4/ RP,P,RLP,ALP
      * /BLOCK5/ KP,WINDO,KST,BST,TACON,TLINE,NTL,NB,KPP
       FOR L=1,NL
        KP2=KST(L)
        WRITE(21,*) KP(KP2,1),KP(KP2,2)
        FOR B=BST(L),BMX(L)-1
         J=LEO(L,B)
         KP1=KP2
         KP2=TACON(J,2)
         KP(KP2,1)=KP(KP1,1)+RL(J)*COS(TA(J))
```

```
        KP(KP2,2)=KP(KP1,2)+RL(J)*SIN(TA(J))
        WRITE(21,*) KP(KP2,1),KP(KP2,2)
        CALL WINDOE(KP(KP2,1),KP(KP2,2),WINDO)
       END FOR
       KP2=TACON(LEO(L,BMX(L)),2)
       WRITE(21,*) KP(KP2,1),KP(KP2,2)
      END FOR
      FOR N=1,NTL
       KP1=TLINE(N,1)
       KP2=TLINE(N,2)
       WRITE(21,*) KP(KP1,1),KP(KP1,2)
       WRITE(21,*) KP(KP2,1),KP(KP2,2)
      END FOR
      IF(RP.NE.0) THEN
       X=KP(KPP,1)+0.5*RL(RP)*COS(TA(RP))
       Y=KP(KPP,2)+0.5*RL(RP)*SIN(TA(RP))
       WRITE(21,*) X,Y
       CALL WINDOE(X,Y,WINDO)
       X=KP(KPP,1)+RLP*COS(TA(RP)+ALP)
       Y=KP(KPP,2)+RLP*SIN(TA(RP)+ALP)
       WRITE(21,*) X,Y
       CALL WINDOE(X,Y,WINDO)
      END IF
      RETURN
      END
*
      SUBROUTINE  NEWC(ITER,C)
      INTEGER B,
     * IG(5),QSP(5),QIF(5),Q(5)
      REAL
     * RL(16),TA(16),
     * A(12,13),
     * QMX(5),G(5,2),Q0(5),
     * C(12)
      COMMON
     * /BLOCK1/ NL,N2,RL,TA
     * /BLOCK2/ A
     * /BLOCK6/ IG,QSP,QMX,G,QIF,Q,Q0,Q00
      IF(QSP(1).NE.0) TA(1)=Q00+QFUN(IG(1),ITER,QSP,QMX)
      X1=RL(1)*COS(TA(1))
      Y1=RL(1)*SIN(TA(1))
      FOR L=1,NL
       J=QIF(L)
       IF(Q(L)*QSP(J).NE.0) TA(Q(L))=Q0(L)+QFUN(J,ITER,QSP,QMX)
      END FOR
      L=NL+1
      FOR I=1,N2-1,2
       L=L-1
       C(I  )=G(L,1)-X1
       C(I+1)=G(L,2)-Y1
       IF(Q(L).NE.0) THEN
        C(I  )=C(I  )-RL(Q(L))*COS(TA(Q(L)))
        C(I+1)=C(I+1)-RL(Q(L))*SIN(TA(Q(L)))
       END IF
      END FOR
      RETURN
      END
*
      SUBROUTINE  NEWA(C)
```

```
      INTEGER B,
     * LEO(6,8),BMX(5),
     * R(12),T(12)
      REAL
     * RL(16),TA(16),
     * A(12,13),
     * AL(12),
     * C(12),
     * SN(12),CS(12)
      COMMON
     * /BLOCK1/ NL,N2,RL,TA
     * /BLOCK2/ A
     * /BLOCK3/ LEO,BMX
     * /BLOCK7/ R,T,AL,NITER,NTA
      FOR J=1,N2
       IF(ABS(TA(R(J))).GT.10.0) CALL MAX2PI(TA(R(J)))
       SN(J)=-RL(R(J))*SIN(TA(R(J)))
       CS(J)= RL(R(J))*COS(TA(R(J)))
      END FOR
      FOR L=NL,1,-1
       I=2*(NL-L)+1
       FOR B=1,BMX(L)
        FOR J=1,N2
         IF(LEO(L,B).EQ.R(J)) THEN
          A(I  ,J)=SN(J)
          A(I+1,J)=CS(J)
         ELSE IF(LEO(L,B).EQ.T(J)) THEN
          A(I  ,J)=-RL(T(J))* SIN( TA(R(J))+AL(J) )  ! YES TA(R(J))
          A(I+1,J)= RL(T(J))* COS( TA(R(J))+AL(J) )
         END IF
        END FOR
       END FOR
      END FOR
      K=N2+1
      FOR I=1,N2-1,2
       A(I  ,K)=C(I  )
       A(I+1,K)=C(I+1)
       FOR J=1,N2
        A(I  ,K)=A(I  ,K)-A(I+1,J)
        A(I+1,K)=A(I+1,K)+A(I  ,J)
       END FOR
      END FOR
      RETURN
      END
*
      SUBROUTINE  NEWRTA(DELMX)
      INTEGER B,
     * R(12)
      REAL
     * RL(16),TA(16),
     * A(12,13)
      COMMON
     * /BLOCK1/ NL,N2,RL,TA
     * /BLOCK2/ A
     * /BLOCK7/ R,T,AL,NITER,NTA
      DELMX=0.0
      FOR J=1,N2
       DEL=A(J,1)
       A(J,1)=0.0
```

```
        TA(R(J))=TA(R(J))+DEL
        IF(ABS(DEL).GT.DELMX) DELMX=ABS(DEL)
       END FOR
       RETURN
       END
*
       SUBROUTINE  NEWTTA
       INTEGER B,
      * R(12),T(12)
       REAL
      * RL(16),TA(16),
      * AL(12)
       COMMON
      * /BLOCK1/ NL,N2,RL,TA
      * /BLOCK7/ R,T,AL,NITER,NTA
       FOR J=1,N2
        IF(T(J).NE.0)
         TA(T(J))=TA(R(J))+AL(J)
         CALL MAX2PI(TA(T(J)))
        END IF
       END FOR
       RETURN
       END
*
       SUBROUTINE HEADING
       CHARACTER*38 CHAR1
       CHARACTER*51 CHAR2
       CALL WINDOW(0.0,13.0,0.0,1.0)
       CALL VWPORT(0.0,130.0,90.0,100.0)
       CALL MOVE(0.0,0.75)
       CALL TXTCLR(7)
       CALL TXSIZE(2,0.0,0.0)
       CHAR1='ANIMATION OF CLOSED-LOOP PLANAR ROBOTS'
       CALL TEXT(38,CHAR1)
       CALL MOVE(1.5,0.50)
       CALL TXTCLR(5)
       CALL TXSIZE(1,0.0,0.0)
       CHAR2='HAVING REVOLUTE JOINTS AND BINARY AND TERNARY LINKS'
       CALL TEXT(51,CHAR2)
       RETURN
       END
*
       SUBROUTINE BORDER(X1,X2,Y1,Y2)
       CALL LINCLR(2)
       CALL MOVE(X1,Y1)
       CALL DRAW(X2,Y1)
       CALL DRAW(X2,Y2)
       CALL DRAW(X1,Y2)
       CALL DRAW(X1,Y1)
       RETURN
       END
*
       SUBROUTINE GROUND(NG,G,R)
       REAL G(5,2)
       CALL OPNSEG(2)
       CALL LINCLR(4)
       DEL=0.628318
       FOR I=1,NG
        X=G(I,1)
```

```
      Y=G(I,2)
      CALL MOVE(X+R,Y)
      FOR J=1,10
       CALL DRAW(X+R*COS(J*DEL),Y+R*SIN(J*DEL))
      END FOR
      S=1.2*R
      CALL MOVE(X+S,Y+S)
      CALL DRAW(X-S,Y+S)
      CALL DRAW(X-S,Y-S)
      CALL DRAW(X+S,Y-S)
      CALL DRAW(X+S,Y+S)
     END FOR
     CALL CLOSEG(2)
     CALL SETVIS(2,.TRUE.)
     RETURN
     END
*
*    END OF SUBROUTINE LIBRARY
```

```
************************************************************************
*                                                                      *
*     OKLAHOMA CRAWDAD ROBOT WORKSPACE PROGRAM                         *
*                                                                      *
*     1.  Determines outer boundary and boundary of voids.             *
*     2.  Calculates area of workspace.                                *
*     3.  Animates robot moving around the outer boundary.             *
*                                                                      *
*     This program was written by Bruce Sumpter during                 *
*     the 1984-1985 school year at Oklahoma State University.          *
*                                                                      *
*     This was the second program of two standing as the               *
*     thesis work for the Master of Science Degree in the              *
*     department of Mechanical and Aerospace Engineering.              *
*                                                                      *
************************************************************************
      INTEGER
     * NVOID(3)
      REAL
     * G0(2),G5(2),G8(2),
     * T1(2),T2(2),T3(2),T4(2),T5(2),T6(2),T7(2),T8(2),T9(2),
     * P1(2),P2(2),P3(2),P4(2),P5(2),P6(2),P7(2),E(2),EG(2),
     * T1R(2),T2R(2),RTRI(3),
     * WN(4),ALIM(500,2,2),ROT(3,2),
     * POS(500,2,14),TE(500),
     * VOID(3,500,2)
      COMMON
     * /BLOCK1/ ALIM
     * /BLOCK2/ G0,G5,G8,R1,R2,R3,R4,R5,R6,R7,R8,R9
     *          ,R29,R1LONG,T63,T96,IROT,ROT,IE,PH,WN
     *          ,P1,P2,P3,P4,P6,P7,T9
     * /BLOCK3/ VOID
      DATA
     * G0/0.0,0.0/G8(2)/0.0/ BIG/1E+9/
     * PI,TWOPI/3.141592654,6.283185308/
     * G5/1.25,2.165/G8(1)/2.5/ IROT/0/ ROT/0,0,0,6.28,6.28,6.28/
     * R1,R2,R3,R4,R5,R6,R63,R7,R8,R9/1,1,1,1,1,1,1,1,1,0.577/
     * R1LONG/1.01/
     * T63,T96/1.047,5.76/
     * ISYM/1/
************************************************************************
*                                                                      *
*     THE INTERACTIVE INPUT USING GRAPHICS AS AIDS                     *
*                                                                      *
************************************************************************
    1 CALL GRSTRT(4113,1)
      CALL HEADING
      CALL WRITE2
      CALL WRITE2
      CALL DELSEG(-1)
      CALL WINDOW(0.,25.,0.,25.)
      CALL VWPORT(0.,85.,15.,100.)
      CALL ALLSEG
      CALL WRITE2
      WRITE(3,*)'DEFAULT DATA ?  1-YES  2-NO'
      READ(0,*)I
      IF(I.EQ.1) GO TO 2
      CALL TXTCLR(7)
```

```
CALL WRITE2
WRITE(3,*)'GROUND POSITIONS:  1-EQUILATERAL  2-SPECIFIC'
READ(0,*) I
CALL SEG1(7)
CALL WRITE2
IF(I.EQ.1) THEN
 WRITE(3,*)'RX =?'
 READ(0,*) G8(1)
 CALL POINT(G0,G8(1),1.047,G5)
ELSE
 ISYM=0
 CALL SEG2(7)
 WRITE(3,*)'RX TX TY =?'
 READ(0,*) G8(1),G5
 CALL SEG2(5)
END IF
CALL SEG1(5)
WN(1)=BIG
WN(2)=-BIG
WN(3)=BIG
WN(4)=-BIG
CALL WNDO(G0,WN)
CALL WNDO(G5,WN)
CALL WNDO(G8,WN)
CALL WRITE2
WRITE(3,*)'SYMMETRIC CONFIGURATION ?  1-YES  2-NO'
READ(0,*) I
CALL WRITE2
IF(I.EQ.1) THEN
 CALL SEG3(7)
 CALL TEXT1(1)
 WRITE(3,*)'R =?'
 READ(0,*) R1
 CALL DELSEG(7)
 CALL SEG3(5)
 R5=R1
 R8=R1
 CALL SEG4(7)
 CALL TEXT1(2)
 CALL WRITE2
 WRITE(3,*)'R =?'
 READ(0,*) R2
 CALL DELSEG(7)
 CALL SEG4(5)
 R4=R2
 R7=R2
 CALL SEG5(7)
 CALL TEXT1(3)
 CALL WRITE2
 WRITE(3,*)'R =?'
 READ(0,*) R3
 CALL DELSEG(7)
 CALL SEG5(5)
 R6=R3
 R63=R3
ELSE
 CALL SEG3(7)
 CALL SEG4(7)
 CALL SEG5(7)
```

```
        CALL TEXT2
        WRITE(3,*)'R1 THRU R5 =?'
        READ(0,*) R1,R2,R3,R4,R5
        CALL WRITE2
        WRITE(3,*)'R6 THRU R9 =?'
        READ(0,*) R6,R63,R7,R8
        CALL DELSEG(8)
        CALL SEG3(5)
        CALL SEG4(5)
        CALL SEG5(5)
        ISYM=0
       END IF
       T63=TRIAN(R3,R6,R63)
       R1LONG=1.01*R1
       CALL SEG6(7)
       CALL WRITE2
       WRITE(3,*)'RE THETA(DEG) =?'
       READ(0,*) R9,T96
       CALL SEG6(5)
       T96=T96*(PI/180.0)
       IF(T96.LT.0.0) T96=T96+TWOPI
     2 CALL WRITE2
       WRITE(3,*)'INPUT ROTATION LIMITS ?  1-NO  2-YES'
       READ(0,*) I
       IF(I.EQ.2) THEN
        FOR I=1,3
         CALL SEG7(I)
         WRITE(3,*)
         WRITE(3,*)'MIN (0-360) MAX (0-360) =?'
         WRITE(3,*)'NOTE:  IF ROTATION CROSSES 0 DEG, MAX < MIN'
         READ(0,*) ROT(I,1),ROT(I,2)
         ROT(I,1)=ROT(I,1)*PI/180.0
         ROT(I,2)=ROT(I,2)*PI/180.0
        END FOR
        CALL SEG7(0)
        IROT=1
       ELSE
        IROT=0
        FOR I=1,3
         ROT(I,1)=0.0
         ROT(I,2)=6.28
        END FOR
       END IF
       R29=R2+R9
*************************************************************************
*                                                                       *
*     GENERATION OF THE INITIAL END-EFFECTOR POSITION                   *
*                                                                       *
*************************************************************************
       IF(ISYM.EQ.1) THEN
        ABLE=0
        E(1)=0.5*G8(1)
        W1=ABS(R1-R2)-R9
        IF(W1.GT.0.0) THEN
         E(2)=0.866*G8(1)+1.01*W1
         CALL SOLVE(E,ABLE,W1)
        END IF
        IF(ABLE.EQ.0) THEN
         E(2)=1.0
```

```
   DO
    E(2)=E(2)-0.001
    CALL SOLVE(E,ABLE,W1)
   UNTIL(ABLE.EQ.1.OR.E(2).LT.0.0)
   IF(ABLE.EQ.0) THEN
    WRITE(3,*)'COULD NOT GENERATE E'
    STOP
   END IF
  END IF
 ELSE
 P2(1)=(G5(1)+G8(1))/3.0
 P2(2)=G5(2)/3.0
 W1=ATANQ(G0,P2)
 T1(1)=W1
 T1(2)=T1(1)+TWOPI
 T5(2)=ATANQ(G5,P2)
 T8(2)=ATANQ(G8,P2)
 IF(IROT.EQ.1) THEN
  T1(1)=ROT(1,1)
  T1(2)=ROT(1,2)
  IF(T1(2).LT.T1(1)) T1(2)=T1(2)+TWOPI
  T5(2)=T5(2)+PI
  T8(2)=T8(2)+PI
  DO
   CALL ROTLIM(2,T5(2),ROT,OK)
   IF(OK.EQ.0) T5(2)=T5(2)+0.017
  UNTIL(OK.EQ.1)
  DO
   CALL ROTLIM(3,T8(2),ROT,OK)
   IF(OK.EQ.0) T8(2)=T8(2)+0.017
  UNTIL(OK.EQ.1)
  T5(2)=T5(2)+PI
  T8(2)=T8(2)+PI
 END IF
 CALL POINT(G5,R5,T5(2),P4)
 CALL POINT(G8,R8,T8(2),P7)
 T1PRE=999
 DO
  CALL POINT(G0,R1,T1(1),P1)
  T2(1)=W1
  T2(2)=T2(1)+TWOPI
  DO
   CALL POINT(P1,R2,T2(1),P2)
   CALL DIAD(P2,P4,R3,R4,T3,T4,OK)
   IF(OK.EQ.1) THEN
    FOR I=1,2
     T6(I)=T3(I)-T63
     CALL POINT(P2,R6,T6(I),P6)
     D=(R7-DIST(P6,P7))/R7
     IF(D.LT.0.1.AND.D.GE.0.0) THEN
      T9(I)=T6(I)-T96
      CALL POINT(P2,R9,T9(I),E)
      T1PRE=T1(1)
      CALL MAX2PI(T1PRE)
      EXIT DO
     END IF
    END FOR
   END IF
   T2(1)=T2(1)+0.1
```

```
        UNTIL (T2(1).GT.T2(2))
        EXIT DO IF(T1PRE.NE.999)
        T1(1)=T1(1)+0.1
       UNTIL (T1(1).GT.T1(2))
       IF(T1PRE.EQ.999) THEN
        WRITE(3,*)'UNABLE TO GENERATE INITIAL END-EFFECTOR POSITION'
        WRITE(3,*)
        WRITE(3,*)'PLEASE SPECIFY INITIAL E(X,Y)'
        READ(0,*) E
       END IF
       END IF
       E2MAX=E(2)
       CALL WRITE2
       CALL WRITE2
       PH=1
       DIR=1
       IE=10
       NLIM1=0
       NLIM2=0
       EG(1)=E(1)
       EG(2)=E(2)
       RG=R1+R2+R3+R4+R5+R6+R7+R8+R9+R63
       DE=RG/1000.0
       RG=RG/90.0
       E2MIN=BIG
       OPEN(22,FILE='BBB')
*************************************************************************
*                                                                       *
*     ITERATION PROCESS OF MOVING TO AND ALONG THE OUTER BOUNDARY       *
*                                                                       *
*************************************************************************
      DO
       IF(PH.EQ.1) THEN
        E(2)=E(2)+DE
       ELSE
        CALL EMOVE(1,DIR,DE,ABLE,E)
       END IF
       CALL SOLVE(E,ABLE,W1)
       IF(ABLE.EQ.1.AND.IE.EQ.0) THEN
        WRITE(22,*)G0,P1,P2,P3,P4,G5,P2,P6,P7,G8,P3,P6,P2,E,W1
       END IF
       IF(ABLE.EQ.0) THEN
        IF(DIR.EQ.1) THEN
         NLIM1=NLIM1+1
         ALIM(NLIM1,1,1)=E(1)
         ALIM(NLIM1,1,2)=E(2)
        ELSE IF(DIR.EQ.3)
         NLIM2=NLIM2+1
         ALIM(NLIM2,2,1)=E(1)
         ALIM(NLIM2,2,2)=E(2)
        END IF
       ELSE IF(E(2).LT.E2MIN)
        E2MIN=E(2)
       END IF
       FOR I=1,2
        IF(ABLE.EQ.0) E(I)=EG(I)
        IF(ABLE.EQ.1) EG(I)=E(I)
       END FOR
       IF(PH.EQ.1.AND.ABLE.EQ.0) THEN
```

```
          PH=2
          E1=E(1)
          E2=E(2)-DE
         ELSE IF(PH.EQ.2.AND.E(1).GT.E1.AND.E(2).LT.E2)
          PH=3
          E2=E(2)
         END IF
        UNTIL (PH.EQ.3.AND.E(1).LT.E1.AND.E(2).GT.E2)
        CALL WRITE2
        WRITE(3,*)
        WRITE(3,*)'PLEASE WAIT A FEW MOMENTS'
        WORKSP=AREA(NLIM2,DE)
        WRITE(22,*)'999 0 0 0 0 0 0 0 0 0 0 0 0 0 0'
        WRITE(22,*)'    0 0 0 0 0 0 0 0 0 0 0 0 0 0'
        CLOSE(22)
*
        R12=R1+R2
        R45=R4+R5
        R78=R7+R8
        RTRI(1)=R9
        RTRI(2)=(R3**2+R9**2-2.0*R3*R9*COS(T96+T63))**0.5
        RTRI(3)=(R6**2+R9**2-2.0*R6*R9*COS(T96))**0.5
        OPEN(23,FILE='CCC')
        PH=1
        I=0
        DO
         CALL ENODES(WN,DELN,I,J,E)
         CALL SOLVE(E,ABLE,W1)
         IF(ABLE.EQ.1) THEN
          CALL ROTNOD(E,G0,G5,G8,R12,R45,R78,RTRI,W1)
          WRITE(23,*)E
          WRITE(23,'(F3.1)')W1
         END IF
        UNTIL(I.EQ.0)
        W1=99.0
        WRITE(23,*)W1,W1
        CLOSE(23)
*************************************************************************
*                                                                       *
*     MOVING TO AND ALONG THE VOID(S) IF PRESENT                        *
*                                                                       *
*************************************************************************
        IF(ABS(R1-R2).GT.R9.AND.E2MAX.GT.G5(2)) THEN
        NV=0
        W2=0
        DO
        IF(NV.EQ.0) THEN
         WRITE(3,*)'VOID 1'
         E(1)=0.0
         E(2)=0.0
         NLIM1=0
         NLIM2=0
         FOR I=1,500
          FOR J=1,2
           FOR K=1,2
            ALIM(I,J,K)=0.0
           END FOR
          END FOR
         END FOR
```

```
 NV=1
ELSE IF(NV.EQ.1)
 WRITE(3,*)'VOID 2'
 E(1)=G5(1)
 E(2)=G5(2)
 NV=2
ELSE
 WRITE(3,*)'VOID 3'
 E(1)=G8(1)
 E(2)=0.0
 NV=3
END IF
PH=1
DIR=1
NVOID(NV)=0
IE=10
DO
 IF(PH.EQ.1) THEN
  E(2)=E(2)+DE
 ELSE
  CALL EMOVE(-1,DIR,DE,ABLE,E)
 END IF
 CALL SOLVE(E,ABLE,W1)
 IF(PH.NE.1) THEN
  IF(ABLE.EQ.0) THEN
   IF(NV.EQ.1) THEN
    IF(DIR.EQ.3) THEN
     NLIM1=NLIM1+1
     ALIM(NLIM1,1,1)=E(1)
     ALIM(NLIM1,1,2)=E(2)
    ELSE IF(DIR.EQ.1)
     NLIM2=NLIM2+1
     ALIM(NLIM2,2,1)=E(1)
     ALIM(NLIM2,2,2)=E(2)
    ELSE IF(E(1).GT.G8(1))
     W2=1
    END IF
   END IF
  ELSE
   IF(IE.EQ.10) THEN
    IE=0
    NVOID(NV)=NVOID(NV)+1
    VOID(NV,NVOID(NV),1)=E(1)
    VOID(NV,NVOID(NV),2)=E(2)
   ELSE
    IE=IE+1
   END IF
  END IF
  FOR I=1,2
   IF(ABLE.EQ.0) E(I)=EG(I)
   IF(ABLE.EQ.1) EG(I)=E(I)
  END FOR
 END IF
 IF(PH.EQ.1.AND.ABLE.EQ.1) THEN
  PH=2
  EG(1)=E(1)
  EG(2)=E(2)
  E1=E(1)
  E2=E(2)
```

```
       ELSE IF(PH.EQ.2.AND.E(1).GT.E1.AND.E(2).LE.E2)
        PH=3
        E2=E(2)
       END IF
      UNTIL (PH.EQ.3.AND.E(1).LE.E1.AND.E(2).GT.E2)
      UNTIL(NV.EQ.3.OR.W2.EQ.1)
      WRITE(3,*)
      WRITE(3,*)WORKSP
      IF(NVOID(1).GE.2) WORKSP=WORKSP-NV*AREA(NLIM2,DE)
      END IF
      WRITE(3,*)
      WRITE(3,*)WORKSP
*************************************************************************
*                                                                       *
*     ANIMATION OF ROBOT MOVING ALONG THE OUTER BOUNDARY                *
*     AND GRAPHICS DISPLAYING THE WORKSPACE INCLUDING VOID(S)           *
*     IF PRESENT                                                        *
*                                                                       *
*************************************************************************
      XMIN=WN(1)
      XMAX=WN(2)
      YMIN=WN(3)
      YMAX=WN(4)
      OPEN(22,FILE='BBB')
      I=0
      DO
       I=I+1
       READ(22,*) ((POS(I,J,K),J=1,2),K=1,14),TE(I)
       EXIT DO IF(POS(I,1,1).EQ.999)
      UNTIL (I.EQ.500)
      CLOSE(22)
      IMAX=I-1
      GDEL=(IMAX/9)+1
      GQUIT=IMAX-(GDEL/2)
    3 G=1
      IG=4
      X1=XMIN
      X2=XMAX
      Y1=YMIN
      Y2=YMAX
      W1=ABS(((X2-X1)-(Y2-Y1))/2.0)
      IF((X2-X1).GT.(Y2-Y1)) THEN
       EDG=(X2-X1)/20.0
       X1=X1-EDG
       X2=X2+EDG
       Y1=Y1-EDG-W1
       Y2=Y2+EDG+W1
      ELSE
       EDG=(Y2-Y1)/20.0
       X1=X1-EDG-W1
       X2=X2+EDG+W1
       Y1=Y1-EDG
       Y2=Y2+EDG
      END IF
      CALL WRITE2
      WRITE(3,*)'OUTPUT:  1-SCREEN  2-DISK  3-HARD COPY'
      READ(0,*) IOUT
      CALL WRITE2
      WRITE(3,*)'INCLUDE ROTATION AT NODES ?  1-YES  2-NO'
```

```
READ(0,*) INOD
IF(IOUT.EQ.2) THEN
 CALL WRITE2
 WRITE(3,*)'GIVE SPOOL COMMANDS ; <CR> ; <CR> AFTER SPOOLING'
 READ(0,*)
END IF
CALL WRITE2
CALL WRITE2
CALL GRSTRT(4113,1)
IF(IOUT.EQ.3) CALL GRSTRT(4113,2)
CALL DELSEG(-1)
CALL HEADING
CALL SPECS(G8(1),G5(1),G5(2),R1,R2,R3,R4,R5,R6,R63,R7,R8,R9,T96
*          ,WORKSP,ROT)
CALL BORDER(X1,X2,Y1,Y2)
CALL BRDLIM(X1,X2,Y1,Y2)
CALL WINDOW(X1,X2,Y1,Y2)
CALL VWPORT(10.,85.,20.,95.)
CALL OPNSEG(3)
W1=EDG/2.0
CALL LINCLR(4)
CALL GROUND(0.,0.,W1)
CALL GROUND(G5(1),G5(2),W1)
CALL GROUND(G8(1),G8(2),W1)
CALL CLOSEG(3)
CALL LINCLR(7)
CALL OPNSEG(2)
CALL CLOSEG(2)
J1=2
LASTI=IMAX
IF(IOUT.EQ.3) LASTI=1
FOR I=1,LASTI
 IF(J1.EQ.2) THEN
  J1=1
  J2=2
 ELSE
  J1=2
  J2=1
 END IF
 CALL SETVIS(-2,.FALSE.)
 CALL OPNSEG(J1)
 CALL LINCLR(7)
 CALL MOVE(POS(I,1,1),POS(I,2,1))
 FOR J=2,6
  CALL DRAW(POS(I,1,J),POS(I,2,J))
 END FOR
 CALL MOVE(POS(I,1,7),POS(I,2,7))
 FOR J=8,10
  CALL DRAW(POS(I,1,J),POS(I,2,J))
 END FOR
 CALL MOVE(POS(I,1,11),POS(I,2,11))
 CALL DRAW(POS(I,1,12),POS(I,2,12))
 CALL MOVE(POS(I,1,13),POS(I,2,13))
 CALL DRAW(POS(I,1,14),POS(I,2,14))
 IF(I.NE.G.OR.I.GE.GQUIT) THEN
  CALL GRIP(POS(I,1,14),POS(I,2,14),TE(I),RG,0,0)
  CALL CLOSEG(J1)
 ELSE
  CALL CLOSEG(J1)
```

```
      CALL GRIP(POS(I,1,14),POS(I,2,14),TE(I),RG,1,IG)
      IG=IG+1
      G=G+GDEL
     END IF
     CALL DELSEG(J2)
     CALL SETVIS(-1,.FALSE.)
     CALL SETVIS(-1,.TRUE.)
     IF(I.GT.1) THEN
      CALL LINCLR(3)
      CALL MOVE(POS(I-1,1,14),POS(I-1,2,14))
      CALL DRAW(POS(I,1,14),POS(I,2,14))
     END IF
    END FOR
    CALL LINCLR(3)
    CALL MOVE(POS(1,1,14),POS(1,2,14))
    FOR I=2,IMAX
     CALL DRAW(POS(I,1,14),POS(I,2,14))
    END FOR
    CALL DRAW(POS(1,1,14),POS(1,2,14))
    IF(ISYM.EQ.1) THEN
     FOR I=1,NV
      CALL MOVE(VOID(I,1,1),VOID(I,1,2))
      FOR J=1,NVOID(I)
       CALL DRAW(VOID(I,J,1),VOID(I,J,2))
      END FOR
      CALL DRAW(VOID(I,1,1),VOID(I,1,2))
     END FOR
    END IF
    IF(INOD.EQ.1) CALL ROTVAL(X1,X2)
    IF(IOUT.EQ.3) THEN
     CALL GRSTOP
     WRITE(3,*)'TYPE:  VPLOT07'
     STOP
    END IF
    IF(IOUT.EQ.2) READ(0,*)
    CALL WRITE2
    WRITE(3,*)'1-REPLAY  2-CHANGE WINDOW  3-NEW ROBOT  0-QUIT'
    READ(0,*) I
    IF(I.EQ.0) STOP
    IF(I.EQ.1) GO TO 3
    IF(I.EQ.2) THEN
     DO
      WRITE(3,*)'ENTER CHOICE AND NEW VALUE:      1-YMAX'
      WRITE(3,*)'                                2-XMIN   3-XMAX'
      WRITE(3,*)'ENTER 0 0 TO QUIT                4-YMIN'
      READ(0,*) I,W1
      IF(I.EQ.1) YMAX=W1
      IF(I.EQ.2) XMIN=W1
      IF(I.EQ.3) XMAX=W1
      IF(I.EQ.4) YMIN=W1
     UNTIL(I.EQ.0)
     GO TO 3
    ELSE
     GO TO 1
    END IF
    CALL GRSTOP
    WRITE(3,*)'GOOD DAY'
    END
*
```

```
*************************************************************************
*                                                                       *
*     THIS IS THE SUBROUTINE LIBRARY FOR THE WORKSPACE PROGRAM          *
*                                                                       *
*************************************************************************
*
*     EMOVE CONTAINS THE LOGIC FOR THE MOVING OF THE END-EFFECTOR
*     AROUND THE WORKSPACE BOUNDARIES
*
      SUBROUTINE EMOVE(J,DIR,DE,ABLE,E)
      REAL E(2)
      DATA D1,D2,D3/1,1,1/
      I=+J
      IF(ABLE.EQ.1) I=-J
      DIR=DIR+I
      IF(DIR.EQ.0) DIR=4
      IF(DIR.EQ.5) DIR=1
      IF(DIR.EQ.1) E(2)=E(2)+DE
      IF(DIR.EQ.2) E(1)=E(1)-DE
      IF(DIR.EQ.3) E(2)=E(2)-DE
      IF(DIR.EQ.4) E(1)=E(1)+DE
      IF(D1.EQ.1.AND.D2.EQ.2.AND.D3.EQ.3.AND.D4.EQ.4.AND.DIR.EQ.1) THEN
       WRITE(3,*)'THIS POINT NOT IN WORKSPACE: ',E
       STOP
      ELSE IF(D1.EQ.4.AND.D2.EQ.3.AND.D3.EQ.2.AND.D4.EQ.1.AND.DIR.EQ.4)
       WRITE(3,*)'NUMERICAL PROBLEM EXISTS'
       STOP
      END IF
      D1=D2
      D2=D3
      D3=D4
      D4=DIR
      RETURN
      END
*
*
*     SOLVE DETERMINES IF A SPECIFIED END-EFFECTOR POSITION
*     IS FEASIBLE
*
      SUBROUTINE SOLVE(E,ABLE,T9ANG)
      REAL E(2),G0(2),G5(2),G8(2),ROT(3,2),WN(4)
     *    ,T1(2),T2(2),T3(2),T4(2),T5(2),T6(2),T7(2),T8(2),T9(2)
     *    ,T1R(2),T2R(2),P1(2),P2(2),P3(2),P4(2),P6(2),P7(2)
      COMMON /BLOCK2/ G0,G5,G8,R1,R2,R3,R4,R5,R6,R7,R8,R9
     *               ,R29,R1LONG,T63,T96,IROT,ROT,IE,PH,WN
     *               ,P1,P2,P3,P4,P6,P7,T9
      DATA TWOPI/6.283185308/
      CALL DIAD(G0,E,R1,R29,T1,T2,OK) ! CHECK E-LOOP DIAD
      IF(OK.EQ.0) THEN
       IF(R29.GE.ABS(R1-DIST(G0,E))) THEN
        OK=1
        T1(1)=ATANQ(E,G0)
        T1(2)=T1(1)+TWOPI
       END IF
      END IF
      ABLE=0
      IF(OK.EQ.1) THEN
```

```
      DO
       CALL POINT(G0,R1,T1(1),P1)      ! FIND P1
       CALL DIAD(P1,E,R2,R9,T2,T9,OK) ! FIND THE TWO SOLS
       IF(OK.EQ.1) THEN
        FOR I=1,2
         CALL POINT(P1,R2,T2(I),P2) ! FIND P2
         IF(IROT.EQ.1) THEN
          CALL DIAD(G0,P2,R1LONG,R2,T1R,T2R,OK)
          IF(OK.EQ.1) CALL ROTLIM(1,T1R(1),ROT,OK)
         END IF
         IF(IROT.EQ.0.OR.OK.EQ.1) THEN
          T3(I)=T9(I)+T96+T63
          CALL POINT(P2,R3,T3(I),P3)       ! FIND P3
          CALL DIAD(P3,G5,R4,R5,T4,T5,OK) ! CHECK FIRST DIAD
         END IF
         IF(IROT.EQ.1.AND.OK.EQ.1) CALL ROTLIM(2,T5(2),ROT,OK)
         IF(OK.EQ.1) THEN
          T6(I)=T9(I)+T96
          CALL POINT(P2,R6,T6(I),P6)       ! FIND P6
          CALL DIAD(P6,G8,R7,R8,T7,T8,OK) ! CHECK SECOND DIAD
          IF(IROT.EQ.1.AND.OK.EQ.1) CALL ROTLIM(3,T8(2),ROT,OK)
          IF(OK.EQ.1) THEN
           ABLE=1
           IF(PH.NE.1) THEN
            IF(IE.EQ.10) THEN
             WRITE(3,*)E
             T9ANG=T9(I)
             CALL DIAD(G0,P2,R1LONG,R2,T1,T2,OK)
             CALL POINT(G0,R1,T1(1),P1)
             CALL POINT(P3,R4,T4(2),P4)
             CALL POINT(P6,R7,T7(2),P7)
             CALL WNDO(P1,WN)
             CALL WNDO(P2,WN)
             CALL WNDO(P3,WN)
             CALL WNDO(P4,WN)
             CALL WNDO(P6,WN)
             CALL WNDO(P7,WN)
             CALL WNDO(E,WN)
             IE=0
            ELSE
             IE=IE+1
            END IF
           END IF
           EXIT DO
          END IF
         END IF
        END FOR
       END IF
       EXIT DO IF(T1(1).GE.T1(2))
       T1(1)=T1(1)+0.1
      UNTIL (ABLE.EQ.1)
     END IF
     RETURN
     END
*
*
*    AREA CALCULATES THE AREA OF A BOUNDED REGION AND IS USED
*    TO DETERMINE THE AREA OF THE WORKSPACE
*
```

```
      FUNCTION AREA(NLIM,DE)
      REAL ALIM(500,2,2)
      COMMON /BLOCK1/ ALIM
      DDE=0.01*DE
      AREA=0.0
      FOR I=1,NLIM
       ADD=1E+9
       X=ALIM(I,1,1)
       Y=ALIM(I,1,2)
       FOR J=1,NLIM
        IF(ABS(ALIM(J,2,1)-X).LE.DDE) THEN
         DIF=Y-ALIM(J,2,2)
         IF(DIF.GT.0.0.AND.DIF.LT.ADD) ADD=DIF
        END IF
       END FOR
       IF(ADD.LT.1E+9) AREA=AREA+ADD
      END FOR
      AREA=DE*AREA
      RETURN
      END
*
      SUBROUTINE DIAD(P1,P2,R1,R2,T1,T2,OK)
      REAL P1(2),P2(2),T1(2),T2(2)
      DATA PI/3.14159/ D/0.000001/
      OK=0
      R3=DIST(P1,P2)
      IF(R1+D.GT.R2+R3.OR.R2+D.GT.R1+R3.OR.R3+D.GT.R1+R2) RETURN
      T3=ATANQ(P1,P2)
      T13=TRIAN(R1,R3,R2)
      T12=TRIAN(R1,R2,R3)
      IF(T13.LT.0.05.OR.T12.LT.0.1) RETURN
      OK=1
      T1(1)=T3-T13
      T1(2)=T3+T13
      T2(1)=T1(1)+(PI-T12)
      T2(2)=T1(2)-(PI-T12)
      RETURN
      END
*
      FUNCTION TRIAN(A,B,C)
      TRIAN=ACOS((A**2+B**2-C**2)/(2.0*A*B))
      RETURN
      END
*
      SUBROUTINE POINT(P0,R1,T1,P1)
      REAL P0(2),P1(2)
      P1(1)=P0(1)+R1*COS(T1)
      P1(2)=P0(2)+R1*SIN(T1)
      RETURN
      END
*
      FUNCTION ATANQ(P1,P2)
      REAL P1(2),P2(2)
      DATA PI/3.14159/
      IF(P2(1).EQ.P1(1)) THEN
       IF(P2(2).GT.P1(2)) THEN
        ATANQ=0.5*PI
       ELSE
        ATANQ=1.5*PI
```

```
      END IF
     ELSE
      SLOPE=((P2(2)-P1(2))/(P2(1)-P1(1)))
      ATANQ=ATAN(SLOPE)
      IF(P2(1).LT.P1(1)) ATANQ=ATANQ+PI
     END IF
     RETURN
     END
*
     FUNCTION DIST(P1,P2)
     REAL P1(2),P2(2)
     DIST=((P2(2)-P1(2))**2+(P2(1)-P1(1))**2)**0.5
     RETURN
     END
*
     SUBROUTINE MAX2PI(ANGLE)
     DATA TWOPI/6.283185308/
     DO
      IF(ANGLE.GE.TWOPI) ANGLE=ANGLE-TWOPI
      IF(ANGLE.LT.0.0) ANGLE=ANGLE+TWOPI
     UNTIL(ANGLE.GE.0.0.AND.ANGLE.LT.TWOPI)
     RETURN
     END
*
     SUBROUTINE WNDO(P,WN)
     REAL P(2),WN(4)
     IF(P(1).LT.WN(1)) WN(1)=P(1)
     IF(P(1).GT.WN(2)) WN(2)=P(1)
     IF(P(2).LT.WN(3)) WN(3)=P(2)
     IF(P(2).GT.WN(4)) WN(4)=P(2)
     RETURN
     END
*
     SUBROUTINE HEADING
     CHARACTER*39 CHAR1
     CALL WINDOW(0.0,13.0,0.0,1.0)
     CALL VWPORT(0.0,130.0,90.0,100.0)
     CALL MOVE(0.0,0.75)
     CALL TXTCLR(5)
     CALL TXSIZE(2,0.0,0.0)
     CHAR1='WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT'
     CALL TEXT(39,CHAR1)
     CALL TXSIZE(1,0.0,0.0)
     RETURN
     END
*
     SUBROUTINE BORDER(XL,XR,YB,YT)
     CALL WINDOW(XL,XR,YB,YT)
     CALL VWPORT(10.0,85.0,20.0,95.0)
     W=(XR-XL)/100.0
     X1=XL+W
     X2=XR-W
     Y1=YB+W
     Y2=YT-W
     CALL LINCLR(2)
     CALL MOVE(X1,Y1)
     CALL DRAW(X2,Y1)
     CALL DRAW(X2,Y2)
     CALL DRAW(X1,Y2)
```

```
      CALL DRAW(X1,Y1)
      W1=(X2-X1)/100.0
      W2=X1+W1
      IY=Y1-1
      DO
       IY=IY+1
       Y=IY
       CALL MOVE(X1,Y)
       CALL DRAW(W2,Y)
      UNTIL(IY.GE.Y2-1)
      W2=Y1+W1
      IX=X1-1
      DO
       IX=IX+1
       X=IX
       CALL MOVE(X,Y1)
       CALL DRAW(X,W2)
      UNTIL(IX.GE.X2-1)
      RETURN
      END
*
      SUBROUTINE BRDLIM(X1,X2,Y1,Y2)
      CHARACTER*6 XL,XR,YB,YT
      OPEN(21,FILE='AAA')
      WRITE(21,'(4F6.2)') X1,X2,Y1,Y2
      REWIND(21)
      READ(21,'(4A6)') XL,XR,YB,YT
      CLOSE(21)
      CALL WINDOW(0.0,8.25,0.0,0.25)
      CALL VWPORT(10.0,85.0,17.5,20.0)
      CALL TXSIZE(1,0.0,0.0)
      CALL TXTCLR(2)
      CALL MOVE(0.0,0.0)
      CALL TEXT(6,XL)
      CALL MOVE(7.25,0.0)
      CALL TEXT(6,XR)
      CALL WINDOW(0.0,1.0,0.0,8.25)
      CALL VWPORT(0.0,9.0,21.0,95.0)
      CALL MOVE(0.0,0.0)
      CALL TEXT(6,YB)
      CALL MOVE(0.0,8.0)
      CALL TEXT(6,YT)
      RETURN
      END
*
      SUBROUTINE SEG1(I)
      CHARACTER*2 RX
      CALL DOL(1,I)
      IF(I.EQ.7) THEN
       CALL MOVE(13.0,4.1)
       RX='RX'
       CALL TEXT(2,RX)
       CALL DIMEN(5.0,4.0,22.0,4.0)
      END IF
      CALL GROUND(5.0,10.0,0.4)
      CALL GROUND(22.0,10.0,0.4)
      CALL CLOSEG(1)
      RETURN
      END
```

```
*
      SUBROUTINE SEG2(I)
      CHARACTER*2 TX,TY
      CALL DOL(2,I)
      IF(I.EQ.7) THEN
       CALL MOVE(7.0,22.1)
       TX='TX'
       CALL TEXT(2,TX)
       CALL MOVE(2.0,16.0)
       TY='TY'
       CALL TEXT(2,TY)
       CALL DIMEN(3.0,10.0,3.0,20.0)
       CALL DIMEN(5.0,22.0,10.0,22.0)
      END IF
      CALL GROUND(10.0,20.0,0.4)
      CALL CLOSEG(2)
      RETURN
      END
*
      SUBROUTINE SEG3(I)
      CALL DOL(3,I)
      CALL DIMEN(5.0,10.0,7.0,15.0)
      CALL DIMEN(15.0,18.0,10.0,20.0)
      CALL DIMEN(18.0,14.0,22.0,10.0)
      CALL CLOSEG(3)
      RETURN
      END
*
      SUBROUTINE SEG4(I)
      CALL DOL(4,I)
      CALL DIMEN(7.0,15.0,9.0,12.0)
      CALL DIMEN(12.0,16.0,15.0,18.0)
      CALL DIMEN(15.0,12.0,18.0,14.0)
      CALL CLOSEG(4)
      RETURN
      END
*
      SUBROUTINE SEG5(I)
      CALL DOL(5,I)
      CALL DIMEN(9.0,12.0,12.0,16.0)
      CALL DRAW(15.0,12.0)
      CALL DRAW(9.0,12.0)
      CALL CLOSEG(5)
      RETURN
      END
*
      SUBROUTINE SEG6(I)
      CHARACTER RE *2,THETA *5
      DATA X,Y,R,TH,DTH/9.0,12.0,5.0,0.0,-0.077/
      CALL DOL(6,I)
      CALL DIMEN(9.0,12.0,15.0,7.0)
      CALL GRIP(15.0,7.0,5.59,0.5,0,0)
      IF(I.EQ.7) THEN
       CALL MOVE(11.0,8.0)
       RE='RE'
       CALL TEXT(2,RE)
       CALL MOVE(15.0,9.0)
       THETA='THETA'
       CALL TEXT(5,THETA)
```

```
      CALL LINCLR(2)
      CALL MOVE(14.0,12.0)
      FOR J=1,9
       CALL DRAW(X+R*COS(TH+J*DTH),Y+R*SIN(TH+J*DTH))
      END FOR
     END IF
     CALL CLOSEG(6)
     RETURN
     END
*
     SUBROUTINE DOL(N,I)
     CALL DELSEG(N)
     CALL OPNSEG(N)
     CALL LINCLR(I)
     RETURN
     END
*
     SUBROUTINE DIMEN(X1,Y1,X2,Y2)
     CALL MOVE(X1,Y1)
     CALL DRAW(X2,Y2)
     X=0.
     Y=0.
     IF(X1.EQ.X2) X=0.5
     IF(Y1.EQ.Y2) Y=0.5
     CALL MOVE(X1-X,Y1-Y)
     CALL DRAW(X1+X,Y1+Y)
     CALL MOVE(X2-X,Y2-Y)
     CALL DRAW(X2+X,Y2+Y)
     RETURN
     END
*
     SUBROUTINE GROUND(X,Y,R)
     DATA DEL/0.628318/
     CALL MOVE(X+R,Y)
     FOR J=1,10
      CALL DRAW(X+R*COS(J*DEL),Y+R*SIN(J*DEL))
     END FOR
     S=1.2*R
     CALL MOVE(X+S,Y+S)
     CALL DRAW(X-S,Y+S)
     CALL DRAW(X-S,Y-S)
     CALL DRAW(X+S,Y-S)
     CALL DRAW(X+S,Y+S)
     RETURN
     END
*
     SUBROUTINE GRIP(X1,Y1,TE1,RG,I,IG)
     X=X1
     Y=Y1
     TE=TE1
     IF(I.EQ.1) THEN
      CALL OPNSEG(IG)
      CALL LINCLR(3)
     END IF
     X=X+1.8*RG*COS(TE-0.281)
     Y=Y+1.8*RG*SIN(TE-0.281)
     CALL MOVE(X,Y)
     TE=TE-1.571
     FOR J=1,5
```

```
      TE=TE-1.047
      X=X+RG*COS(TE)
      Y=Y+RG*SIN(TE)
      CALL DRAW(X,Y)
     END FOR
     IF(I.EQ.1) THEN
      CALL CLOSEG(IG)
      CALL LINCLR(7)
     END IF
     RETURN
     END
*
     SUBROUTINE TEXT1(I)
     CHARACTER*1 R
     REAL P(2,3)
     DATA P/5.7,13.0,8.0,14.0,10.0,14./
     CALL OPNSEG(7)
     CALL MOVE(P(1,I),P(2,I))
     R='R'
     CALL TEXT(1,R)
     CALL CLOSEG(7)
     RETURN
     END
*
     SUBROUTINE TEXT2
     REAL P(2,9)
     CHARACTER*2 R(9)
     DATA R/'R1','R2','R3','R4','R5','R6','R7','R8','R9'/
    *     P/5.,13., 8.,14., 9.5,14., 14.,16.5, 13.,19., 11.,12.1,
    *       14.,14., 16.5,12., 20.2,12./
     CALL OPNSEG(8)
     FOR I=1,9
      CALL MOVE(P(1,I),P(2,I))
      CALL TEXT(2,R(I))
     END FOR
     CALL CLOSEG(8)
     RETURN
     END
*
     SUBROUTINE ALLSEG
     CALL SEG1(5)
     CALL SEG2(5)
     CALL SEG3(5)
     CALL SEG4(5)
     CALL SEG5(5)
     CALL SEG6(5)
     RETURN
     END
*
     SUBROUTINE WRITE2
     WRITE(3,*)
     WRITE(3,*)
     RETURN
     END
*
     SUBROUTINE SPECS(RX,TX,TY,R1,R2,R3,R4,R5,R6,R7,R8,R9,RE,THTA,AREA
    *                 ,ROT)
     REAL ROT(3,2)
     CHARACTER*6 NAME(18),VAL(21)
```

```
      DATA
     * NAME/'RX   ','TX   ','TY   '
     *      ,'R1   ','R2   ','R3   ','R4   ','R5   ','R6   '
     *      ,'R7   ','R8   ','R9   ','RE   ','THETA'
     *      ,'ROT1 ','ROT2 ','ROT3 ','AREA '/
      OPEN(21,FILE='AAA')
    1 FORMAT(20F6.2,/,F6.2)
      WRITE(21,1) RX,TX,TY,R1,R2,R3,R4,R5,R6,R7
     *           ,R8,R9,RE,THTA,ROT(1,1),ROT(2,1),ROT(3,1)
     *           ,AREA,ROT(1,2),ROT(2,2),ROT(3,2)
      REWIND(21)
      READ(21,'(20A6,/,A6)') VAL
      CLOSE(21)
      CALL WINDOW(0.0,4.0,0.0,8.0)
      CALL VWPORT(85.0,130.0,0.0,90.0)
      CALL TXSIZE(1,0.0,0.0)
      X1=0.5
      X2=1.5
      X3=2.5
      Y=7.5
      DY=-0.30
      CALL TXTCLR(5)
      J=0
      FOR I=1,18
       IF(I.EQ.4.OR.I.EQ.15.OR.I.EQ.18) Y=Y+DY
       IF(I.GT.3) CALL TXTCLR(7)
       IF(I.GE.15) CALL TXTCLR(2)
       IF(I.EQ.18) CALL TXTCLR(3)
       CALL MOVE(X1,Y)
       CALL TEXT(6,NAME(I))
       CALL MOVE(X2,Y)
       CALL TEXT(6,VAL(I))
       IF(I.GE.15.AND.I.NE.18) THEN
        J=J+1
        CALL MOVE(X3,Y)
        CALL TEXT(6,VAL(18+J))
       END IF
       Y=Y+DY
      END FOR
      RETURN
      END
*
      SUBROUTINE ROTLIM(I,ANG,ROT,OK)
      REAL ROT(3,2),LO
      DATA PI/3.14159/
      T=ANG
      IF(I.NE.1) T=T+PI
      OK=1
      LO=ROT(I,1)
      HI=ROT(I,2)
      CALL MAX2PI(T)
      IF(LO.LE.HI) THEN
       IF(T.LT.LO.OR.T.GT.HI) OK=0
      ELSE
       IF(T.GT.HI.AND.T.LT.LO) OK=0
      END IF
      RETURN
      END
*
```

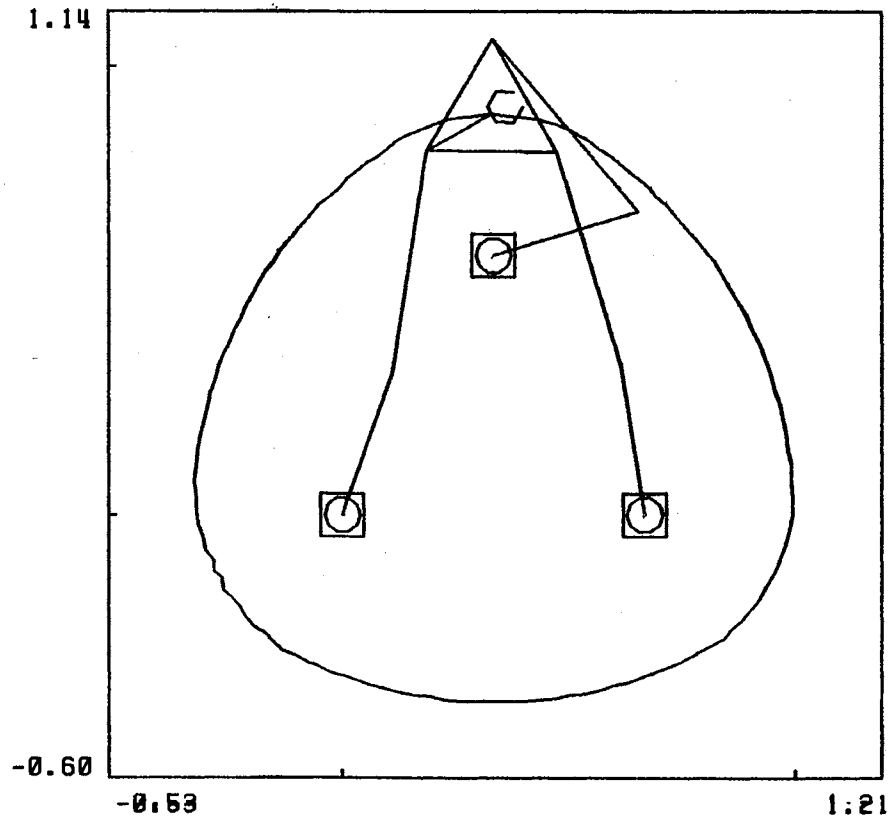
```
      SUBROUTINE SEG7(I)
      CHARACTER*3 MIN,MAX
      DATA MIN,MAX/'MIN','MAX'/
      CALL TXTCLR(7)
      IF(I.EQ.1) THEN
       CALL OPNSEG(7)
       CALL LINCLR(2)
       CALL ARC(5.0,10.0,2.0,5.8,8.4,2)
       CALL MOVE(6.0,8.0)
       CALL TEXT(3,MIN)
       CALL MOVE(2.0,11.0)
       CALL TEXT(3,MAX)
      ELSE IF(I.EQ.2) THEN
       CALL DELSEG(7)
       CALL SEG3(5)
       CALL OPNSEG(7)
       CALL ARC(10.0,20.0,2.0,4.5,6.8,2)
       CALL MOVE(8.0,18.0)
       CALL TEXT(3,MIN)
       CALL MOVE(11.0,21.5)
       CALL TEXT(3,MAX)
      ELSE IF(I.EQ.3) THEN
       CALL DELSEG(7)
       CALL SEG3(5)
       CALL OPNSEG(7)
       CALL ARC(22.0,10.0,2.0,1.0,3.6,2)
       CALL MOVE(23.5,10.5)
       CALL TEXT(3,MIN)
       CALL MOVE(20.0,8.0)
       CALL TEXT(3,MAX)
      END IF
      IF(I.NE.0) THEN
       CALL CLOSEG(7)
      ELSE
       CALL DELSEG(7)
       CALL SEG3(5)
      END IF
      RETURN
      END
*
      SUBROUTINE ARC(X,Y,R,T1,T2,ICL)
      CALL LINCLR(ICL)
      CALL MOVE(X+R*COS(T1),Y+R*SIN(T1))
      DO
       T1=T1+0.1745
       CALL DRAW(X+R*COS(T1),Y+R*SIN(T1))
      UNTIL(T1.GE.T2)
      RETURN
      END
*
      SUBROUTINE ENODES(WN,DELN,I,J,E)
      REAL WN(4),E(2)
      IF(I.EQ.0) THEN
       I=1
       DELN=(WN(2)-WN(1))/8.0
       E(1)=WN(1)+DELN
       E(2)=WN(3)+DELN
      ELSE
       IF(E(1).LT.WN(2)) THEN
```

```
        E(1)=E(1)+DELN
       ELSE
        E(1)=WN(1)+DELN
        E(2)=E(2)+DELN
       END IF
       IF(E(2).GE.WN(4)) I=0
      END IF
      RETURN
      END
*
      SUBROUTINE ROTNOD(E,G0,G5,G8,R12,R45,R78,RTRI,D)
      REAL E(2),G0(2),G5(2),G8(2),RTRI(3),T1(2),T2(2)
      DATA TWOPI/6.28319/
      CALL DIAD(G0,E,R12,RTRI(1),T1,T2,OK)
      CALL MAX2PI(T2(1))
      CALL MAX2PI(T2(2))
      D1=T2(1)-T2(2)
      IF(D1.LT.0.0) D1=D1+TWOPI
      CALL DIAD(G5,E,R45,RTRI(2),T1,T2,OK)
      CALL MAX2PI(T2(1))
      CALL MAX2PI(T2(2))
      D2=T2(1)-T2(2)
      IF(D2.LT.0.0) D2=D2+TWOPI
      CALL DIAD(G8,E,R78,RTRI(3),T1,T2,OK)
      CALL MAX2PI(T2(1))
      CALL MAX2PI(T2(2))
      D3=T2(1)-T2(2)
      IF(D3.LT.0.0) D3=D3+TWOPI
      D=D1
      IF(D2.LT.D) D=D2
      IF(D3.LT.D) D=D3
      RETURN
      END
*
      SUBROUTINE ROTVAL(X1,X2)
      CHARACTER*3 R
      DX=0.2*(X2-X1)/7.0
      DY=0.125*(X2-X1)/7.0
      OPEN(23,FILE='CCC')
      CALL TXTCLR(3)
      DO
       READ(23,*)X,Y
       EXIT DO IF(X.EQ.99.)
       READ(23,'(A3)')R
       CALL MOVE(X-DX,Y-DY)
       CALL TEXT(3,R)
      UNTIL(X.EQ.99.)
      CLOSE(23)
      RETURN
      END
*
*     END OF SUBROUTINE LIBRARY
```

APPENDIX B

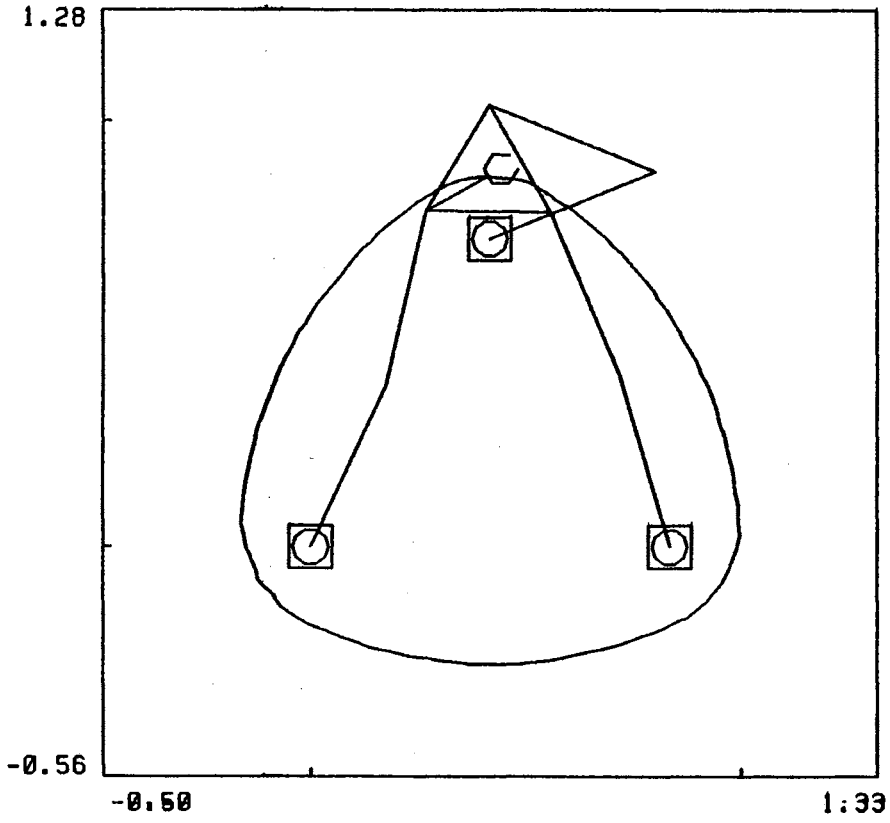
SAMPLE WORKSPACE SHAPES OF THE OKLAHOMA CRAWDAD ROBOT

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



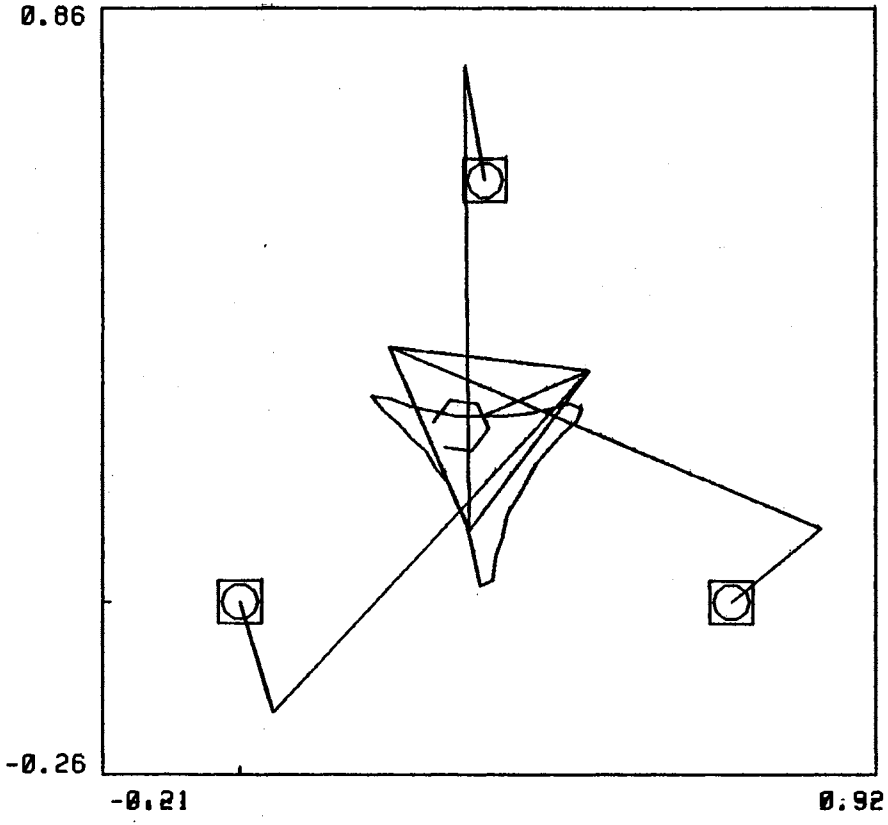
RX	0.67	
TX	0.33	
TY	0.58	
R1	0.33	
R2	0.50	
R3	0.29	
R4	0.50	
R5	0.33	
R6	0.29	
R7	0.29	
R8	0.50	
R9	0.33	
RE	0.17	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	1.34	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



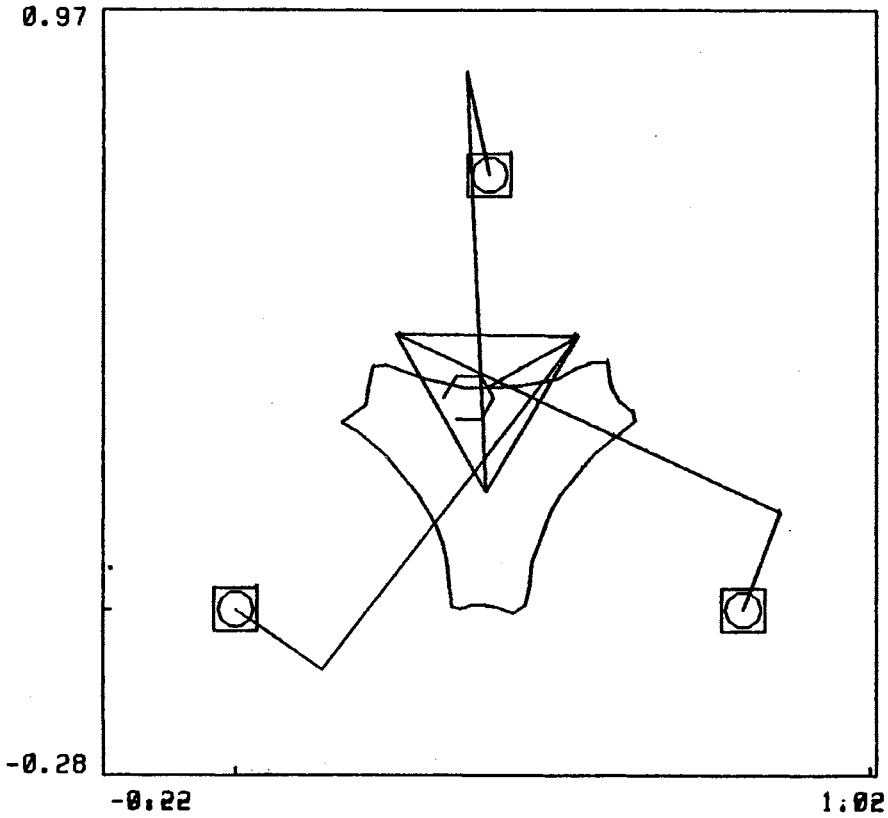
RX	0.83	
TX	0.42	
TY	0.72	
R1	0.42	
R2	0.42	
R3	0.29	
R4	0.42	
R5	0.42	
R6	0.29	
R7	0.29	
R8	0.42	
R9	0.42	
RE	0.17	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	1.00	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



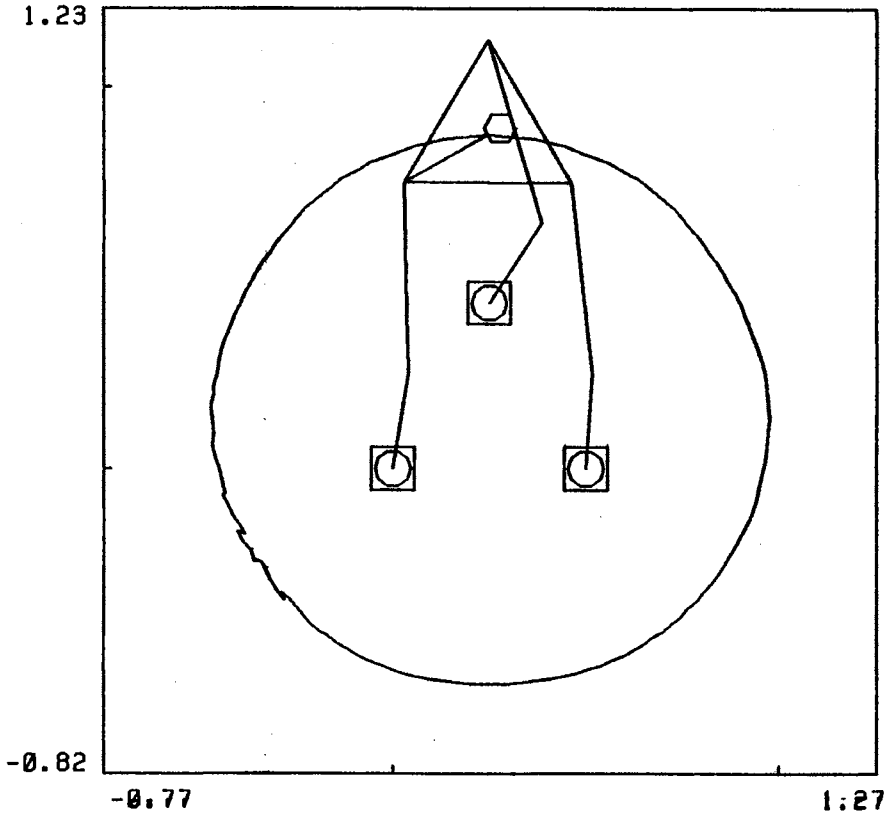
RX	0.70	
TX	0.35	
TY	0.61	
R1	0.17	
R2	0.67	
R3	0.29	
R4	0.67	
R5	0.17	
R6	0.29	
R7	0.29	
R8	0.67	
R9	0.17	
RE	0.17	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	0.03	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



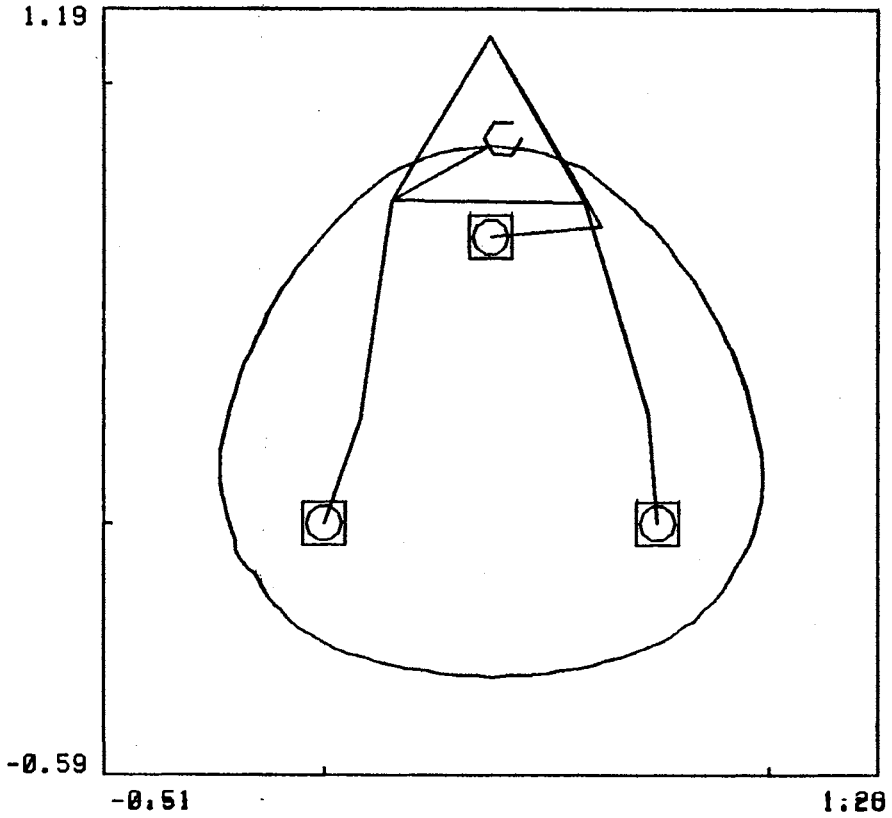
RX	0.80	
TX	0.40	
TY	0.69	
R1	0.17	
R2	0.67	
R3	0.29	
R4	0.67	
R5	0.17	
R6	0.29	
R7	0.29	
R8	0.67	
R9	0.17	
RE	0.17	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	0.10	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



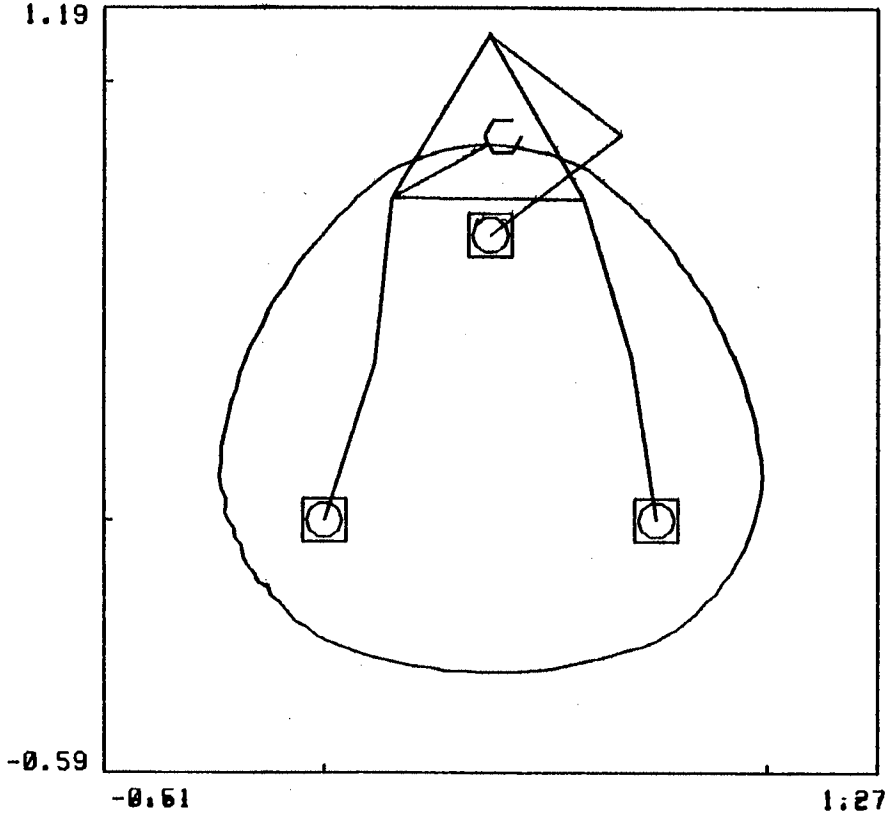
RX	0.50	
TX	0.25	
TY	0.43	
R1	0.25	
R2	0.50	
R3	0.43	
R4	0.50	
R5	0.25	
R6	0.43	
R7	0.43	
R8	0.50	
R9	0.25	
RE	0.25	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	1.64	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



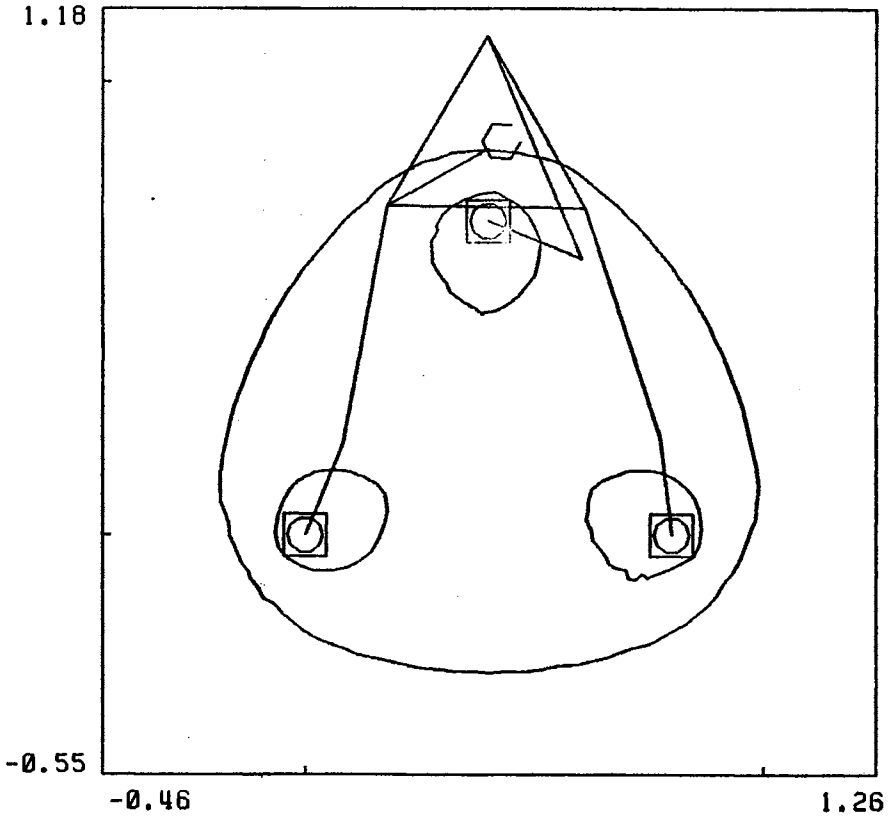
RX	0.75	
TX	0.38	
TY	0.65	
R1	0.25	
R2	0.50	
R3	0.43	
R4	0.50	
R5	0.25	
R6	0.43	
R7	0.43	
R8	0.50	
R9	0.25	
RE	0.25	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	1.14	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



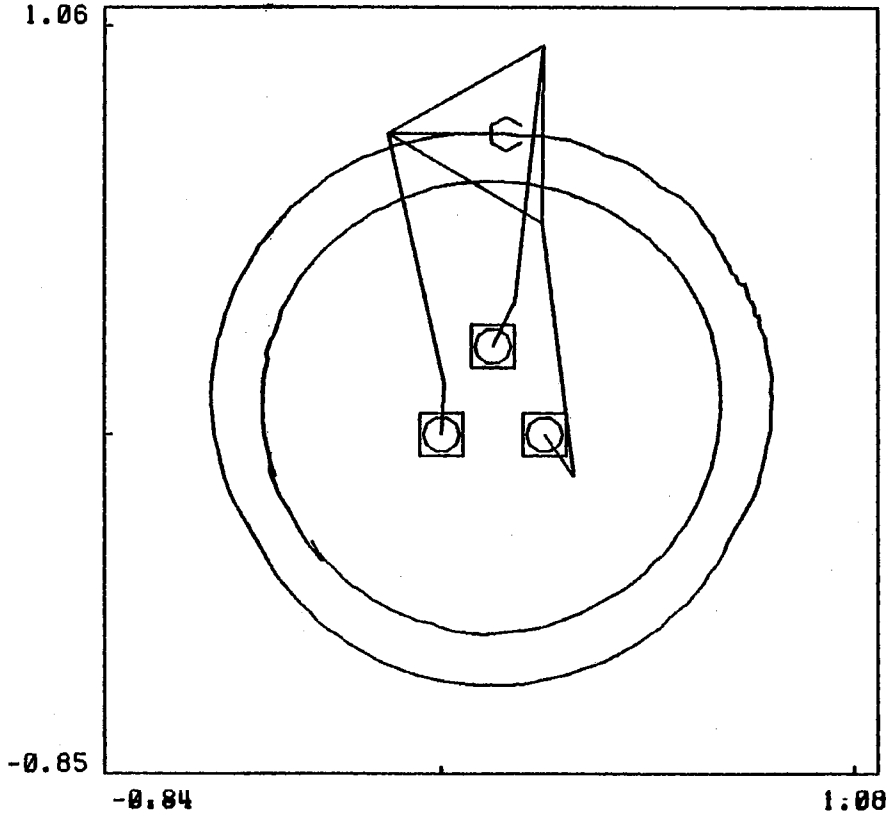
RX	0.75	
TX	0.38	
TY	0.65	
R1	0.38	
R2	0.38	
R3	0.43	
R4	0.38	
R5	0.38	
R6	0.43	
R7	0.43	
R8	0.38	
R9	0.38	
RE	0.25	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	1.15	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



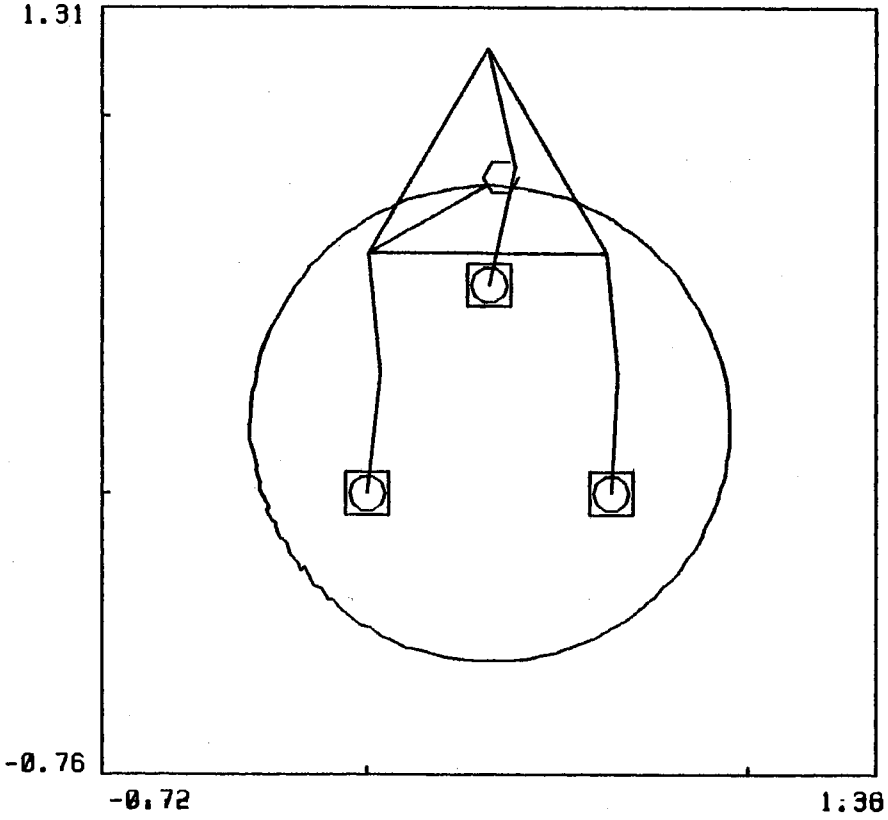
RX	0.80	
TX	0.40	
TY	0.69	
R1	0.22	
R2	0.53	
R3	0.43	
R4	0.53	
R5	0.22	
R6	0.43	
R7	0.43	
R8	0.53	
R9	0.22	
RE	0.25	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	0.92	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



RX	0.25	
TX	0.13	
TY	0.22	
R1	0.12	
R2	0.62	
R3	0.43	
R4	0.62	
R5	0.12	
R6	0.43	
R7	0.43	
R8	0.62	
R9	0.12	
RE	0.25	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	0.50	

WORKSPACE OF THE OKLAHOMA CRAWDAD ROBOT



RX	0.64	
TX	0.32	
TY	0.55	
R1	0.32	
R2	0.32	
R3	0.62	
R4	0.32	
R5	0.32	
R6	0.62	
R7	0.62	
R8	0.32	
R9	0.32	
RE	0.36	
THETA	5.76	
ROT1	0.00	6.28
ROT2	0.00	6.28
ROT3	0.00	6.28
AREA	1.26	

VITA

Bruce Sumpter

Candidate for the Degree of

Master of Science

Thesis: COMPUTER ANIMATION OF CLOSED-LOOP PLANAR ROBOTS AND WORKSPACE ANALYSIS OF THE OKLAHOMA CRAWDAD ROBOT

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Minneapolis, Minnesota, April 29, 1961, the son of Dennis F. and Carolyn R. Sumpter.

Education: Graduated from Ponca City High School, Ponca City, Oklahoma, in May, 1979; received Bachelor of Science Degree in Mechanical Engineering from Oklahoma State University in May, 1983; completed requirements for the Master of Science degree at Oklahoma State University in May, 1986.

Professional Experience: Teaching Assistant, Department of Mechanical and Aerospace Engineering, Oklahoma State University, August, 1983, to May, 1985.