

OFF-LINE ROBOT VISION SYSTEM PROGRAMMING
USING A COMPUTER AIDED DESIGN SYSTEM

by

S. SRIDARAN

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Industrial Engineering and Operations Research

APPROVED:

Dr. Michael Deisenroth

Dr. K. B. Yu

Dr. Marilyn Jones

October 16, 1985
Blacksburg, Virginia

OFF-LINE ROBOT VISION SYSTEM PROGRAMMING
USING A COMPUTER AIDED DESIGN SYSTEM

by

S. SRIDARAN

Dr. Michael Deisenroth

Industrial Engineering and Operations Research

(ABSTRACT)

Robots with vision capability have been taught to recognize unknown objects by comparing their shape features with those of known objects, which are stored in the vision system as a knowledge base. Traditionally, this knowledge base is created by showing the robot the set of objects that it is likely to come across. This is done with the vision system to be used and must be done in an on-line mode.

An approach to teach the robot in an off-line mode by integrating the robot vision system and an off-line graphic system, has been developed in this research. Instead of showing the objects that the robot is likely to come across, graphic models of the objects were created in an off-line graphic system and a FORTRAN program that processes the models to extract their shape parameters was developed. These shape parameters were passed to the vision system. A program to process an unknown object placed in front of the vision sys-

tem was developed to extract its shape parameters. A program that compares the parameters of the unknown object with those of the known models was also developed.

The vision system was calibrated to measure the the pixel dimensions in inches. In the vision system, shape parameters of the objects were found to vary with different orientations. The range of variation for each parameter was established and this was taken into consideration in the parameter comparison program.

ACKNOWLEDGEMENTS

I take this opportunity to express my sincere thanks to
for his wise and patient steermanship,
encouragement, involvement, and inspiring guidance throughout the course of this research. He accepted the awesome task of editing the manuscript of this thesis, and reduced the literary ineptness of the author to a minimum without altering the meaning of the written word. The author also wishes to express his thanks to all the faculty and staff of the Department of Industrial Engineering and Operations Research for their support and encouragement.

TABLE OF CONTENTS

1.0 INTRODUCTION 1

1.1 Background 1

1.2 Problem Definition 5

1.3 Scope of Research 8

2.0 LITERATURE REVIEW 10

2.1 CAD System and Object Descriptions 10

2.2 Vision System Technology 12

3.0 CAD GEOMETRIC INTERFACE AND PROCESSING 18

3.1 CADAM System 19

 3.1.1 CADAM Geometry Interface 19

 3.1.2 Execution and Linkage Commands 23

3.2 Geometric Parameter Extraction Algorithm 24

 3.2.1 Concatenation Phase 24

 3.2.2 Parameter Extraction Phase 25

3.3 Shape Parameter Extraction Program 34

3.4 Description of Subroutines 40

4.0 VISION SYSTEM 46

4.1 Imaging Technology System Description 46

4.2 Programming the PCVISION frame grabber 50

4.3 Vision Algorithm 53

| | | |
|-------|---|-----|
| 4.3.1 | Binary Processing | 53 |
| 4.3.2 | Connectivity analysis algorithm | 57 |
| 4.3.3 | Parameter Extraction Procedures | 64 |
| 4.4 | Vision system Program | 66 |
| 4.5 | Description of Subroutines | 67 |
| 5.0 | INTEGRATION OF CADAM AND PCVISION SYSTEMS | 69 |
| 5.1 | File Transfer | 69 |
| 5.2 | Calibration of the machine vision system | 71 |
| 5.3 | Vision system parameter ranges | 72 |
| 6.0 | RESULTS AND CONCLUSIONS | 80 |
| | LIST OF REFERENCES | 90 |
| | APPENDIX A. CAD SYSTEM PROGRAM | 92 |
| | APPENDIX B. VISION SYSTEM PROGRAM | 121 |
| | VITA | 130 |

LIST OF ILLUSTRATIONS

Figure 1. An eight connected pixel 16

Figure 2. Chain code of an outline. 17

Figure 3. CADAM system 21

Figure 4. Parametric representation of a point on a
circle 26

Figure 5. Determination of area of a polygon 29

Figure 6. Determination of inward pointing normal 31

Figure 7. Possible cases of arcs 33

Figure 8. Determination of maximum radius 35

Figure 9. Determination of minimum radius 36

Figure 10. Machine Vision System 48

Figure 11. Frame memory organization 49

Figure 12. Display logic 51

Figure 13. Histogram 56

Figure 14. Run-length code 59

Figure 15. Multilabelled blob 60

Figure 16. The 16 possible states of a pixel 62

Figure 17. Variation of area 76

Figure 18. Variation of perimeter 77

Figure 19. Variation of maximum radius 78

Figure 20. Variation of minimum radius 79

Figure 21. Test Model 1 87

Figure 22. Test Model 2 88

Figure 23. Test Model 3 89

LIST OF TABLES

| | | |
|----------|------------------------------------|----|
| Table 1. | Variation of parameters | 75 |
| Table 2. | CAD system parameters | 84 |
| Table 3. | Parameter ranges | 85 |
| Table 4. | Vision system parameters | 86 |

1.0 INTRODUCTION

1.1 BACKGROUND

The task of enabling robots to visualize and recognize objects has been the subject of a significant amount of research. This is because of the potential advantages of a robot which can see, over a sightless robot. A robot which can see may be more effective in carrying out tasks that can also be executed by sightless robots. This may be illustrated by an example (4).

Consider that a sightless robot is used to move objects arriving via an input conveyor to either of two output conveyors where the objects will be carefully channelled and marked by a paint brush placed in a fixed position. These objects might be bonnets and tailgates of toy cars which eventually would be painted blue and green respectively. Consider the technical problems. First, the input conveyor belt would have to contain jigs that present the parts to the manipulator in a predetermined manner. The manipulator controller would have to be told exactly how many parts of each kind are contained in each batch, so that the control program will know when to branch between one part of the program and the other.

Now imagine the presence of a vision camera and a recognition system that can be pointed at the input conveyor belt. Clearly there would be no need for special jigs as the vision system would not only identify the parts but also inform the manipulator as to where they were placed. Then armed with a paint brush, the robot could mark the parts appropriately without having to pick them up. This illustrates the way in which the use of vision transfers the development overheads from mechanical complexity and precision to visual processing which is cost-benefit effective because of the availability of cheaper microprocessors used for vision processing.

A machine vision system uses the information from known objects to recognize the unknown objects presented to it. There are two approaches to building a set of known object patterns. One approach is to treat a pattern generated from a previous encounter as known and compare this with other images. This is called 'teach by showing' and is used in SRI system (1).

Another approach is to generate patterns in an off-line mode with respect to the vision system and pass the data as known patterns to the vision system. This method is being researched by many industries. The inherent disadvantage with 'teach by showing' method is that it is time consuming when the user must physically present to the camera a large number

of objects or a large number of expected configurations (stable states). Another disadvantage is that samples of the finished product must be available. Additionally parameters obtained by 'teach by showing' depend upon the specific configuration of the system and are not transferable to other setups.

Primarily there are two ways of comparing the unknown model with the known dataset. The known object model can be considered as a template and recognition is achieved by moving the template over the unknown until the template becomes completely aligned with the object. In digital system, the template is stored in memory as a two dimensional matrix. This matrix is referred to as the reference. When a frame of video data is loaded into system memory, the system overlays the reference on the upper left hand corner of the video frame and calculates the number of matches. This process, referred to as correlation, continues until the reference has been compared with the video frame throughout the field of view. If there is no acceptable match anywhere in the field of view another reference may be used and the process repeated. This continues until the object is identified. An advantage of this type of system is that it is very tolerant of noise and background information.

The disadvantage with template system is that they cannot easily recognize objects with orientation different from that of the known pattern.

Another approach is called 'matching based on feature extraction'. This is an alternative to template matching, which proceeds at the image level. It abstracts some measurements or features (such as area, perimeter for a two dimensional scene) of the image and then matches these data against the same features of different known patterns. The choice of features is problem dependent.

The features can be described in terms of shape, position and orientation (7). Pot et.al (19) argue that an interesting feature should have the following parameters:

1. It must be stable with respect to the variations of the images of the same object.
2. It must be distinguishing.
3. It must be fast to compute.

They also recommend a number of basic parameters that may be derived from an arbitrary shape to provide valuable classification information.

These include:

1. Area

2. Perimeter
3. Minimum enclosing rectangle
4. Center of area
5. Minimum and maximum radius vectors
6. Holes (number, size, position)
7. Number of corners

1.2 PROBLEM DEFINITION

The problem addressed in this research was to integrate a machine vision system with a computer-aided design (CAD) system operated in an off-line mode with respect to the vision system. The idea of generating patterns in an off-line mode was considered owing to the disadvantages of the 'teach by showing' system, and to the fact that the data in a CAD system, already available in a computer integrated manufacturing facility, could be concurrently used by the vision system. This enhanced the integration of the CAD and CAM systems and improved the utilization of the system by putting the CAD data to an additional use.

The method of feature extraction was considered for recognition here, because this method can help recognize objects differing in orientation with respect to the known pattern. Moreover this was more suitable in the context of the objective mentioned above, since the information required by the

machine vision system is only the features of the known model. This data can be extracted from the models residing in the CAD system and supplied to the machine vision system. The integration of the two systems was effected by passing the feature information.

The CAD system used in this research was the CADAM (R) (Computer Augmented Design And Manufacturing system) system, and the machine vision system used was the Imaging Technology vision system. The different procedures involved in the integration of the machine vision system with the CAD system were:

1. The creation of parametric descriptions of known object models in an off-line mode by using the data from the CADAM graphic system.

To accomplish this:

Wire-frame descriptions of known objects were generated in the CADAM graphic system and added to the CAD database.

An algorithm was developed and implemented on the graphic system to retrieve the wire-frame model of

the object from the CAD database and extract features from the wire-frame representation.

2. In order for object recognition to occur the features extracted from the known object were then compared with the features of the unknown object presented to the machine vision system.

To accomplish this:

The digitized image of the unknown object, available in the memory of the computer connected to the machine vision system, was segmented from the background. Then the segmented image was processed to extract parameters. An algorithm was developed and implemented on the computer connected to the machine-vision system to accomplish the above mentioned tasks.

A decision rule to compare the parameters extracted from the CAD system with those extracted from the machine-vision system was formulated and implemented on the vision system computer.

The units of measurement for the CAD system and those for the machine-vision system were not compatible since the units of

measurement in the CAD system is in inches while those of the machine-vision system are in pixels. Gruver et.al (13) suggested that the vision system be calibrated to measure the machine vision system parameters in terms of CAD system units.

The vision system parameters varied a range of values, since the number of pixels covered by the edges varies depending on the position of the part. Therefore, the range of variation of each parameter was determined experimentally.

1.3 SCOPE OF RESEARCH

In this research only a two dimensional view of an object was considered for recognition. The objects could have polygonal and circular holes in them. The boundary of the object had to have at least one straight edge. All curved edges were approximated as arcs of circles. The view of the object presented to the machine vision system was the same as that stored in the CADAM system, but the former may be different in orientation with respect to the latter.

In this research only two levels of intensity were considered for each pixel. That is, the image was considered to be binary.

The reasons for considering a binary image were:

1. It is evident from previous research (8) that a binary image is adequate for recognition of objects for a variety of robot vision applications.
2. The image can be represented in a compact form.
3. The image analysis algorithms are simpler for binary images because of the structure.
4. Since only one bit of intensity information is required to be stored, an image of size 480 x 512 picture elements can be stored in 30,720 bytes of memory. Thus even a modest microcomputer can store a complete binary image of average spatial resolution and still have memory left over for image analysis software.
5. The algorithms that analyze a binary image are simpler than those which analyze gray scale images. With a binary image, the computations used in such functions as locating the edge of an object are reduced to logical operations involving only a few picture elements (pixels). As a result, binary image analysis programs are generally much faster than comparable programs that process gray scale images.

2.0 LITERATURE REVIEW

2.1 CAD SYSTEM AND OBJECT DESCRIPTIONS

A CAD system is used to model both geometric and nongeometric parts that the user designs. In addition, a CAD database provides the means to model relationships between the components of an assembly in a generic fashion. The designer can describe an object to the graphic system in geometric terms; that is in terms of graphic output primitives such as points, lines, polygons or character strings geometrically oriented in a two or three dimensional world. The data structure in the graphic system stores the object description. It must contain the geometric coordinate data of the graphic primitives, and connectivity relationships and positioning data that defines how the components fit together.

The user creates the geometric description of object and stores it in the data structure through a CAD application program. The application program, by issuing commands to the graphic display system, can make the picture appear on a screen. The graphic system software which consists of a group of plotting subroutines drives the specific devices and causes the device to display the picture.

Three different techniques are used in representing graphical data. The object description method in which two dimensional and three dimensional objects are described in terms of line drawings is called WIRE-FRAME description. Vandam (10) says that wire-frame descriptions are computationally efficient, but they do not give a realistic view of the object. Additionally, they do not allow removal of hidden surfaces from the display, or calculate the weight or volume of an object.

In situations where a realistic representation of an object is required, an object can be defined by combinations of mathematical surfaces such as planes, spheres etc. In this description it is possible to remove hidden surfaces.

The objects can be described in terms of polygonally bound surfaces called polygonal mesh which are easy to manipulate. These provide only an approximate description of curved surfaces. This technique is called SURFACE MODELLING. Another technique of representing an object, called SOLIDS MODELLING, deals directly with solid objects, and uses solids such as cubes, cones, spheres, and cylinders as primitives, which are added and subtracted to form shapes. Though surface and solids modelling techniques give a more realistic description, they are computationally very expensive and are yet to be brought out of the laboratory.

Vandam (10) recommended the use of wire-frame models in an engineering environment, since all engineering drawings are two dimensional line representations of the object. Moreover wire-frame models give an accurate representation of the orthographic projection of a particular view of an object. This is sufficient for many recognition purposes, though protrusions in an orthogonal direction with respect the projected view are not represented well. In many commercially available graphic systems such as CADAM, wire-frame descriptions of the objects are used for representation. Wire-frame drafting gives an inexpensive, reasonably good representation of objects and still provides the rapid responses necessary for interactive system use.

2.2 VISION SYSTEM TECHNOLOGY

A vision system provides part identification by a comparison of the digitized image produced by the video-camera and digitizing logic components of the system, with a stored pattern of parameters. Before comparing the patterns, the digitized image has to be processed to extract the parameters. The image array can either be processed as a gray level image with several levels of intensity, or they can be processed as binary images with only two levels of intensity. Cunningham (8) recommended the use of binary images for robot vision applications, since they are more compact for repre-

sentation, and algorithms can be easily developed and moreover binary images are sufficient for recognition of the object boundaries in a typical manufacturing environment. Cunningham also recommended the use of edge detection techniques using convolution calculations such as connectivity analysis and boundary tracking, since they are easily implementable for a binary images.

Gruver et.al (13) have developed an off-line programmed robot vision system using CAD. Their work is one of the major research efforts in building off-line programmed robot vision system. In their work they have presented a conceptual idea of the operational aspects of a CAD based robot vision system. Their system uses the feature matching method to recognize the unknown objects. A special feature about the CAD system used in their work is that the object models need not be drafted manually. The vision programming module in the CAD system has the provision to generate the object model and to extract features from it automatically. To take a picture of the object, the user has to invoke the PERSPECTIVE function of the CAD graphics system. This produces a 2-D image of the object. This 2-D image is then edited graphically to remove hidden lines, so that profiles of visible features on the object result. Procedures are then invoked from the CAD system to calculate the required object features.

Curtis (9) has developed an off-line programmed robotic inspection system using CAD. The vision system emulates the human inspector. After a part is designed on the CAD system, a translation program translates the CAD information into a database suitable for inspection purposes. Two artificial intelligence modules AI-1, AI-2 act as interface between this database and the vision system. AI-1 module determines and simulates the inspection procedure and outputs general instructions to the AI-2 module. AI-2 module performs the actual actual commands to operate the robot and vision system.

There are many reports found in the literature about extraction of shape parameters from a binary image. Two approaches to extract shape parameters from a binary image have evolved. The first approach called 'connectivity analysis' involves a sequential scanning of each line of the image and the connectivity of each pixel around its neighbors is used to build blob descriptions. The advantage with this method is that the processing is done in one raster scan of the image.

The other approach called 'boundary tracking' follows the boundary of each blob to extract the shape parameters. Prior to applying the algorithm a 'chain code' representation of the image is required. This is a representation of an image in terms of its boundary pixels.

Freeman (11) suggested the idea of representing an image as a chain code. The chain code is a list data structure. Its first entries are the x,y coordinates of the point from which to begin tracing the outline. Subsequent entries are numbers giving the direction between each point in the outline and its neighbor. There are eight possible directions between a point and its neighbour (Figure 1). These eight directions are numbered "0" through "7", counter-clockwise. Figure 2 shows an outline and its chain code.

A.K.Agrawala and A.V.Kularni (2) described a sequential approach to extract shape features from an image. Here only the edge points of each line are used. At any instant only the edge points of the previous scan line and current line are required for the calculations. All of the information about the current scan line is processed in a single pass through the image matrix.

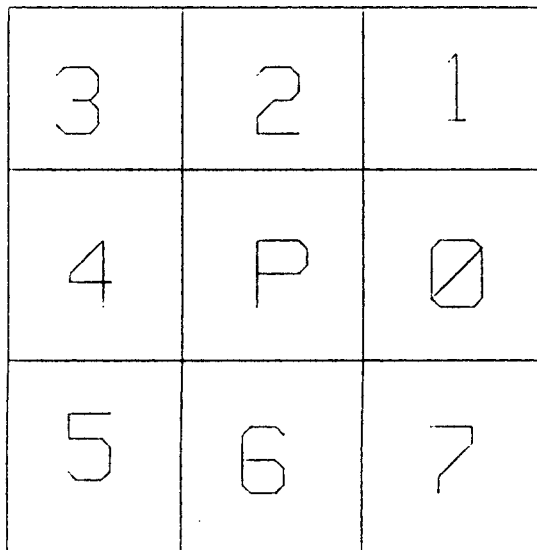


Figure 1. An eight connected pixel

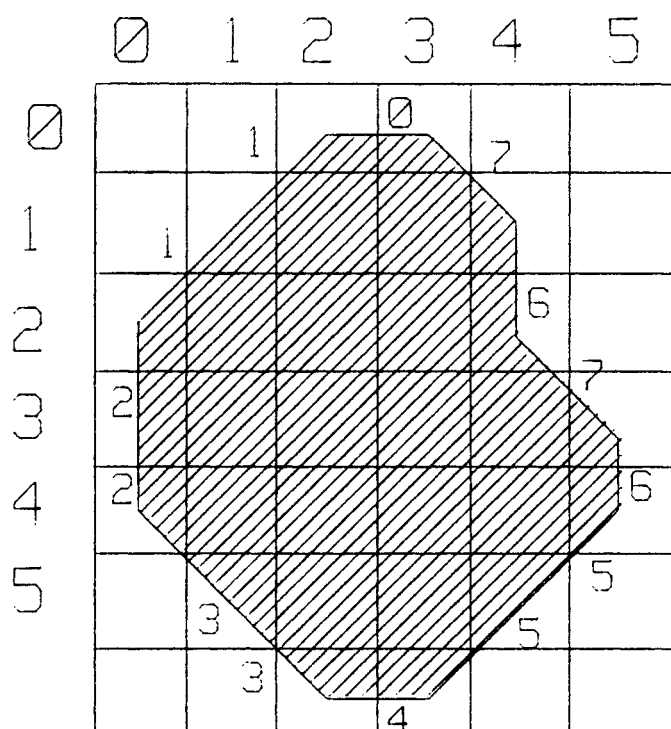


Figure 2. Chain code of an outline.

3.0 CAD GEOMETRIC INTERFACE AND PROCESSING

The first step involved in the processing of CADAM geometric data was to create the geometric models (of objects) in the CADAM database. The CADAM graphic system already available to the user was used for this purpose. The next step was to access the geometric model for the calculation of the desired shape parameters. CADAM geometry interface routines available to the user were used to access the geometric models. Then, an algorithm to extract shape parameters of these graphic models was developed. An application program to implement this algorithm was developed and implemented. This program includes call statements to CADAM geometry interface routines. The shape parameters extracted were passed to the vision system program for comparison with the shape parameters of the unknown object extracted by the vision system program. The details of the available CADAM geometry interface routines are discussed below but the graphic routines used for the construction of graphic models in the CADAM system are not discussed. Also a detailed description of the shape parameter extraction algorithm developed for the CADAM system is given below.

3.1 CADAM SYSTEM

The CADAM system is a set of computer programs that can be used to prepare mechanical drawings on a computer terminal. The CADAM system can be used to construct a wire frame model of an object by providing data about primitive elements such as lines, points, circles etc. Since each graphic entity is associated with a specific view (that has a geometric relationship with other world views), any desired view of an object can be created by graphical construction. This includes isometric view as well as other auxiliary views. Graphic model data are stored in a file for recall and modification. This data is also accessible for manipulation by application programs for external digestion and analysis.

3.1.1 CADAM GEOMETRY INTERFACE

The geometry interface module available in the system provides the ability to access a particular CADAM file in the CADAM database system from an application program instead of from a graphics terminal. The interface modules act as interface between an application program written in a higher level language and the plotting routines and other software routines available in the graphic system. There are four geometry interface functions available in CADAM.

LOFT Takes curve and surface shape definitions and assembles them into a CADAM model containing parametric cubic splines. The model can then be added (as a CADAM drawing) to the CADAM database.

CADCD Provides a collection of subroutines which are driven by FORTRAN CALL statements to produce CADAM elements. The generated elements can then be added as a CADAM model to a drawing file.

CADET Provides a collection of CALL statements which drive passive subroutines which are designed to receive disassembled CADAM elements. The CADAM model is retrieved from a drawing file by the module.

CADMACGM Provides a collection of subroutines which are driven by CALL statements to produce CADAM elements. The generated elements are added to a CADAM model in a foreground mode.

The CADET module was used in this research since the desire was to access the part geometric data. The CADET (CADAM Element Transfer) program combines user written software within an existing program structure. It transfers CADAM model data from a CADAM drawing to user-written routines. An existing

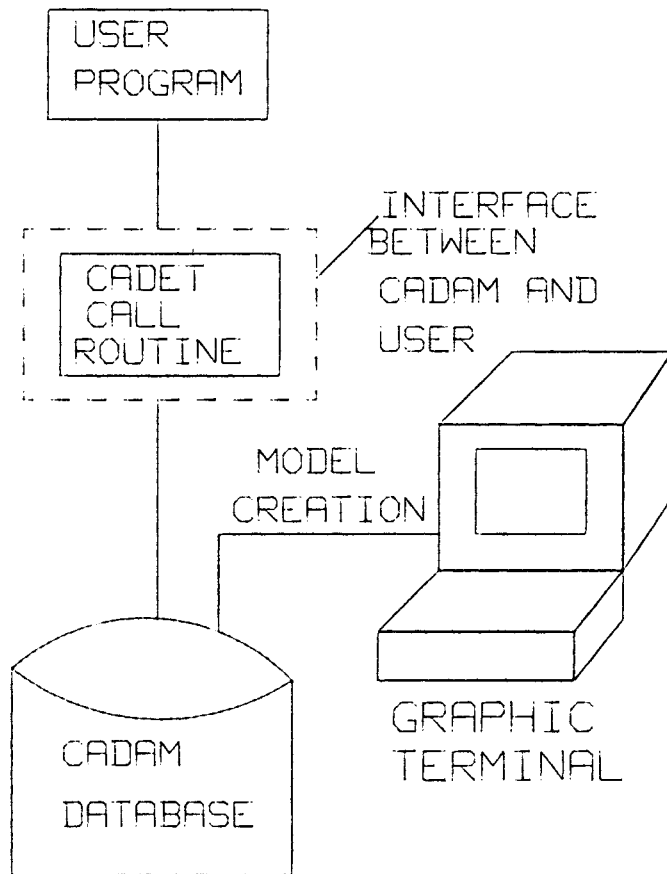


Figure 3. CADAM system

CADAM model is disassembled into elemental pieces (points, lines, text, circles etc) and then each piece is processed by the CADET main program. Each element data is then passed to the appropriate user-written routine. These routines process or store the data as needed and then return to the CADET control program. The processing continues until all CADAM model elements have been transferred. An end of view routine is then called to allow the user additional processing for that view before moving to geometric elements in the next view.

A skeleton FORTRAN routine called 'subroutine RESOLV', containing entry points and parameter lists for each type of element, was provided with the CADAM software package. Those routines that deal specifically with the geometric elements of interest were expanded to extract the information needed in an ordered structure and to provide for post view processing for vision parameter computation and analysis.

Since the CADET main program processed each geometric element separately as it is encountered, it was necessary to concatenate the elements to reconstruct the desired contour. The elements of internal holes or pockets were distinguished from those of the outer contours by using the CADAM ATTRIBUTE function. While the drawings were being created, elements lying inside the contour were assigned an attribute value

of 2 and those on the outer contour were assigned a value of 1. When an element is retrieved the attribute value can be checked. A subroutine to concatenate and process the contour was included in the expanded RESOLV subroutine as part of the post view processing routine.

3.1.2 EXECUTION AND LINKAGE COMMANDS

A main program called CADETMN calls the CADET module and was included in the CADAM system. The main program and the modified RESOLV subroutine were compiled and linked using the command 'CADETLNK CADETMN'.

To execute the program module created by CADETLNK, the CADET exec file must be used. The command

```
CADET fn ft group
```

was used where 'fn' and 'ft' were respectively the name and type of the input file for the application program. The 'group' was the name of the CADAM group to which the drawing belongs. The input file contained the user id in columns 2-5, the drawing file name in columns 10-25, and find name starting from column 26. All data was left justified in the appropriate field.

3.2 GEOMETRIC PARAMETER EXTRACTION ALGORITHM

The shape parameter extraction algorithm was divided into two main phases, namely, the concatenation phase and the parameter extraction phase.

3.2.1 CONCATENATION PHASE

In the foregoing discussion it was pointed out that it was essential to distinguish the elements of the outer contour from those of the holes. This was achieved by assigning different attribute values to the elements. The concatenation procedure although the same for both, was carried out separately for the internal and external contours. If there are multiple holes, they are distinguished from one another by assigning a hole number to each hole.

The concatenation procedure started by comparing the first endpoint coordinates of the first element with those of all other elements. If there was a match, then the matching coordinate values were not considered for further comparison, and the other end point of the matching element was compared with all remaining elements. This procedure continued until the contour closes. This occurred when a matching end point

was found for the second end point of the initial geometric element.

3.2.2 PARAMETER EXTRACTION PHASE

Algorithms and appropriate software were developed for the following parameters: perimeter, area, and minimum and maximum radius.

PERIMETER: The perimeter was computed by summing the distance between successive end points of the contour elements. An exception occurred when the element was an arc. In this case, the arc length was computed and added. The coordinates of the arc center and of the arc end points were known. To determine the arc length, a parametric representation of a point on a circle as given in (20) was used. Points on the circumference of the circle were represented by values between 0 and 4.0 as a function of the slope. At any time only one quadrant of the circle was considered. The parameter value of any point on this quadrant was given by

$$T=(\text{SQRT}(1.0+\text{TANT}*\text{TANT})-1.0)/\text{TANT}$$

where

TANT = Y/X if the point was in first or third quadrant

TANT = -Y/X if the point was in second or fourth quadrant

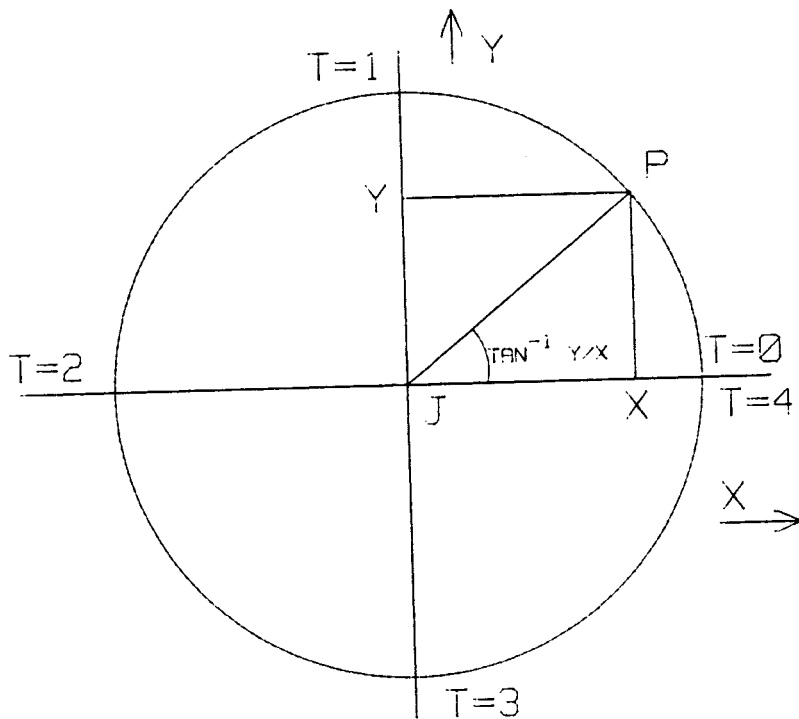


Figure 4. Parametric representation of a point on a circle

X, Y were the differences between the x, y-coordinate values of the center of the circle and that of the concerned point respectively.

The signs of X, and Y were used to determine the quadrant of the circle associated with the point. For example, if X was negative and Y was positive then the concerned point was in the second quadrant. If a point was found to be in second, third or fourth quadrant, the parameter value was incremented by 1, 2, or 3 respectively.

The direction considered for arc definitions was counter-clockwise proceeding from arc point-1 to arc point-2. The parameter difference associated with two arc points was

$$PD = T2 - T1$$

if parameter value of arc point-2 (T2) was greater than parameter value of arc point-1 (T1). If T1 was greater than T2, then the the parameter difference was

$$PD = 4 - (T1 - T2)$$

The length of the arc was given by

$$\text{Arc length} = PD / 4.0 * \text{radius of the arc} * 2 * \pi$$

AREA: Initially the arcs were ignored and the ends of arcs were considered to be connected by straight edges. This reduced the problem to that of determining the area of a polygon. A single vertex was connected to all the vertices of the polygon to form triangles as shown in Figure 5. The

area was then found by summing the areas of the triangles. With this approach the center of area of the polygon can also be determined. The area of a triangle with vertices (XL, YL) , (XK, YK) , (XM, YM) , was given by the vector product

$$\begin{vmatrix} (XL - XK) & (XM - XK) \\ (YL - YK) & (YM - YK) \end{vmatrix}$$

and the center of area was given by

$$XCG = (XK+XL+XM)/3$$

$$YCG = (YK+YL+YM)/3$$

The area of the polygon was the sum of the vector products of all triangles. If the vector product was negative then that triangular area has to be subtracted. Again the convention adapted here was counter-clockwise.

The areas contributed by the arcs had to be determined separately. A problem with arcs was that it was not known whether the arc was convex or concave with respect to the contour of the object. The area contributed by the arc had to be added or subtracted depending on whether the arc was convex or concave, and whether the arc center lay in the positive direction of inward pointing normal at the midpoint of the side or in its negative direction. To determine this, the normal

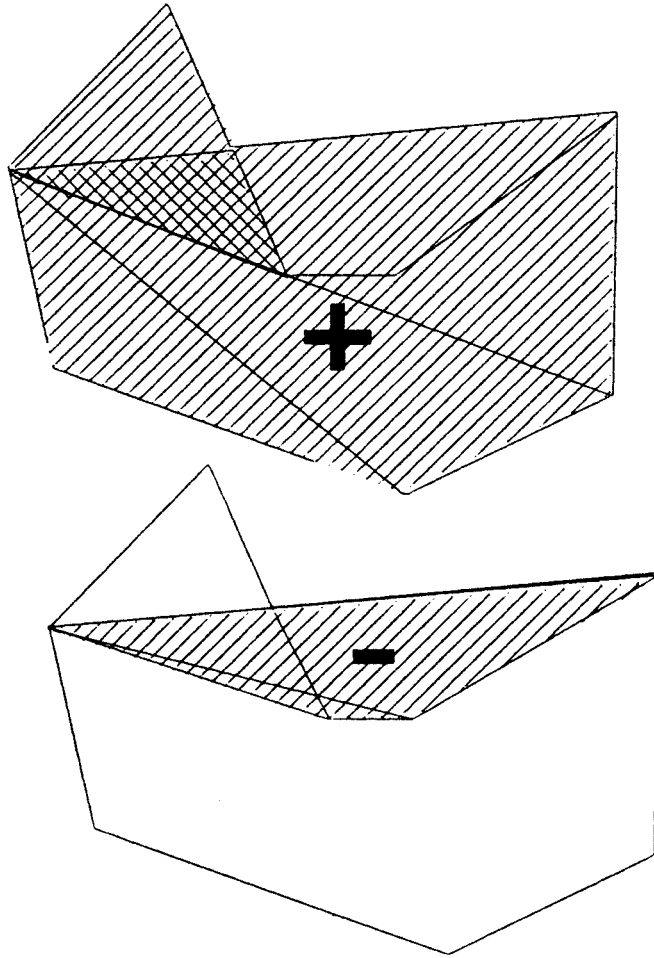


Figure 5. Determination of area of a polygon

pointing into the area from each side of the polygon was to determined.

To determine the direction of an inward pointing normal to a given side, point A, at unit distance from the midpoint of the side is considered (Figure 6). To determine if this point was inside the area, a line was drawn which connects this point to a point B outside the object. If the number of valid intersections of the line segment was odd, the point was inside the object and MA was the normal pointing into the area. The normal for all the other sides was obtained by performing two dimensional transformation on the first normal.

There were three possible locations of the arc center. The arc center could be in the same direction as that of the normal or it may have coincided with the midpoint of the side or it may lie in the direction opposite to that of the normal. For each location of the arc two cases were be considered (Figure 7). These cases depend on whether the arc direction was clockwise or counter clockwise. The areas were then appropriately added or subtracted from the polygonal area. The arc areas for cases 7a, and 7b were calculated by determining the area of the segment of the circle plus the area of the triangle ABC, while the areas for cases 7c, and 7d were calculated by subtracting the area of the triangle ABC from the area of the segment of the circle.

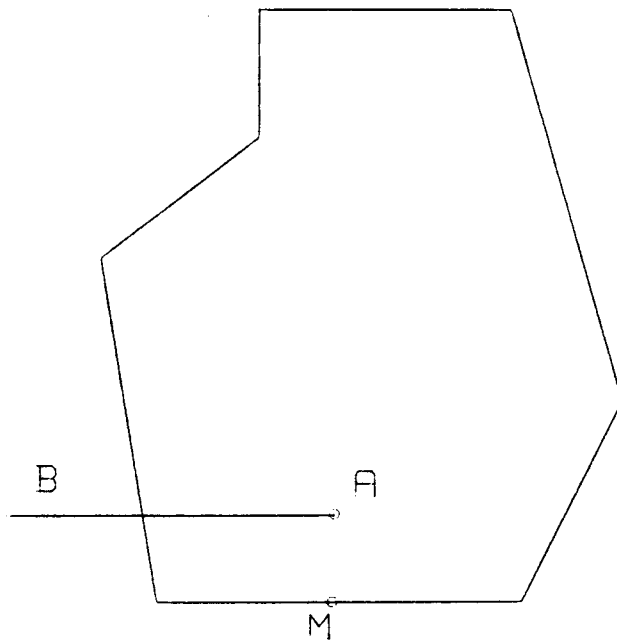


Figure 6. Determination of inward pointing normal

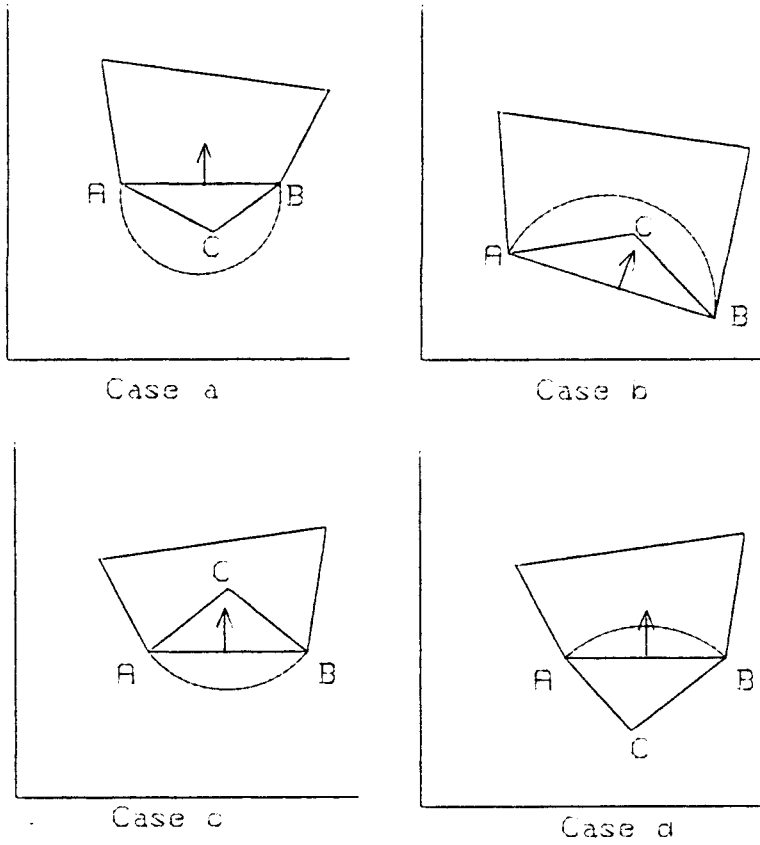


Figure 7. Possible cases of arcs

CENTROID: The centroid was determined as it was needed as a reference point to define the maximum and minimum radius. The centroid of the object was computed by summing the product of the area of a segment of the object and the distance of the center of that segment from the X and Y axes.

$$XCG = \Sigma(XCG(i) * AREA(i))/TOTAL AREA$$

$$YCG = \Sigma(YCG(i) * AREA(i))/TOTAL AREA$$

Where XCG(i) and YCG(i) were the coordinates of the center of area of segment i and AREA(i) was the area of segment i.

If there were holes with the object, the area and centroid of each hole was first determined by considering the contour of the hole as if it were the contour of an object. The hole area and centroid were then multiplied together and subtracted in the calculations of the total object centroid.

MAXIMUM RADIUS: It is known from the theory of linear programming that the maximum point of a convex polygon always lies on one of the vertices. This principle was used in determining the maximum radius of the contour. When there was an arc element, the above principle did not hold. If the arc was convex, then a point on the arc may be farther away from the centroid than the farthest displaced vertex from the centroid. Hence, if there was an arc element, the distance of the farthest point on the arc from the centroid was cal-

culated, and this was compared against the displacements of all vertices and other arcs from the centroid. The maximum of all the displacements is defined as the maximum radius as shown in Figure 8.

MINIMUM RADIUS: The minimum radius was defined as the distance of the nearest point on the outer contour from the centroid. (Figure 9). The minimum distance may have been the distance of a vertex from the centroid or the shortest distance from an edge of the outer contour. The above mentioned cases had to be considered and the minimum of all the distances was defined as the minimum radius.

3.3 SHAPE PARAMETER EXTRACTION PROGRAM

The parameter extraction routines were added to the subroutine RESOLV available in the CADAM system. There were seven main modules in the program which are described below. A listing of the program is included in Appendix-A.

MODULE-1: The module contained the procedure to retrieve the CADAM drawing elements from a drawing. Additionally, entry points CDTST (CADET start processing), CDTBVU (CADET begin view information), and CDTEVU (CADET end view information) were included. Entry points which retrieve elements and in-

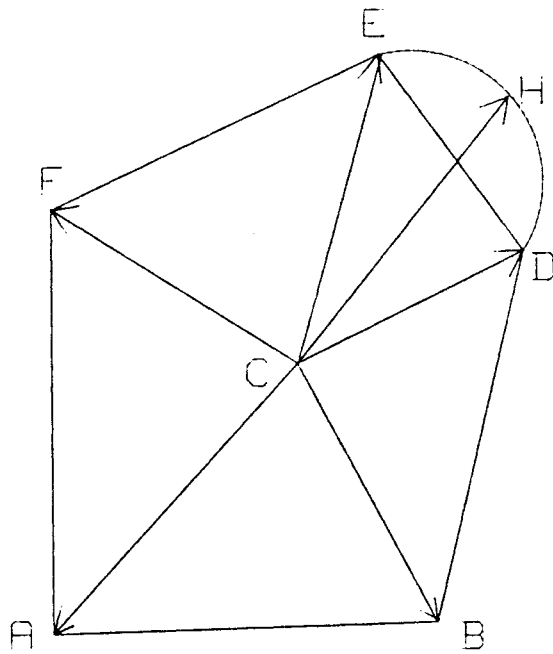


Figure 8. Determination of maximum radius

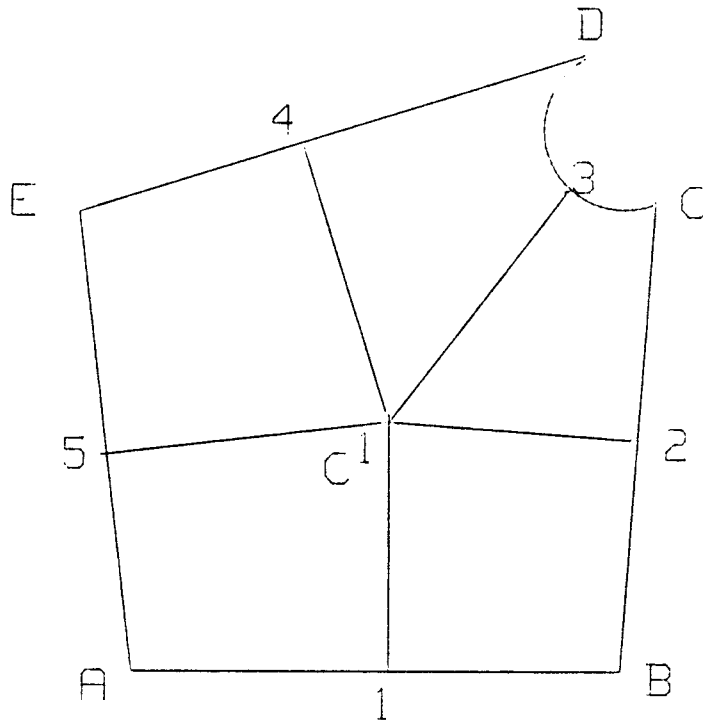


Figure 9. Determination of minimum radius

formation which were not required were included after the parameter extraction module.

For each geometric element of interest, X and Y coordinate values were passed to the procedure developed in the research. These routines saved data for post view processing. The CDTLN routine received data associated with each line element in the C and D arrays given in the argument list: CDTLN(C,D,LT). The CDTARC routine processed arc data as the coordinate center (C) and the end points D and F. This was passed in the argument list CDTARC(C,D,E,LT). Finally, the CDTCIR routine processed the circular element data as the coordinates of the center C, and the radius, RAD. This data was passed via CDTCIR(C,RAD,LT).

The ATTRIB subroutine was called by the CDTLN, CDTARC, and CDTCIR procedures to determine the attribute value of an element. If the attribute value of an element was 1, it formed part of the outer contour and if it was 2, it formed part of a hole.

MODULE-2: This module contained the procedure for the concatenation of the elements of the object. The procedure first linked all of the elements of the outer contour of the object as described earlier. Next the elements of interior holes were chained to form contours appropriately. This

module continued until all elements were accounted for the object contour.

MODULE-3: This module contained the the procedure that computed the perimeter of the object distance by calculating the length of each outer contour element. The procedure compared the value of the variable POLARC(i,1) and POLARC(i,2) for each vertex. If POLARC(i,1) and POLARC(i+1,2) were equal to 1, the element was an arc. Subroutine PARMET was called to find the parameter values of the two arc end points. Then the arc length was computed as described in the algorithm. If an element is a line, the length was computed as the distance between the endpoints. All the element lengths were added to calculate the object perimeter.

MODULE-4: The procedure to compute the area of the outer contour was contained in this module. The subroutines NORM1 and NORM2 were called to determine the normals on each side, that point towards the contour. Then the subroutine PLAREA was called to calculate the area of the contour considering only straight edges, that is circular edges were ignored and considered as straight. Next the subroutine CRAREA was called to determine the area contributed by the circular edges. The algebraic sum of the above two areas gives the effective area of the outer contour. This module also con-

tained the steps to determine the coordinates of the center of area of the outer contour.

MODULE-5: This module contained the procedure to process the holes. The area, and centroid parameters for each hole were calculated in this module.

MODULE-6: This module contained the procedure to determine the center of area of the contour. The areas of holes were taken into consideration in the computation of the effective centroid.

MODULE-7: This module determined the maximum and minimum radii of the outer contour. The effective centroid of the outer contour, that is the centroid calculated after taking into consideration the area of the holes was considered. The subroutines MAXIR and MINIR were called to determine the maximum and minimum radii.

3.4 DESCRIPTION OF SUBROUTINES

SUBROUTINE NORM1: This subroutine determined the normal that points towards the contour for an edge of the contour. The endpoints of the edge were passed as arguments to the subroutine. Initially the edge with end points

(POLYLN(1,1),POLYLN(1,2)) and (POLYLN(2,1),POLYLN(2,2)) was considered. A line segment formed by a point (PINMID(1,1),PINMID(1,2)) located at 0.25" from the midpoint of the edge and the point (PINC3,PINC4) which lies outside the contour was considered. The subroutine INTLN was called to find the intersection point of this line with all the edges. An intersection point with an edge was valid if it fell within the edge. If the number of intersection points was odd then the point was within the contour and the line formed by this point and the midpoint of the edge is the normal pointing towards the object.

SUBROUTINE NORM2: This subroutine determined the inward pointing normals for all the edges other than first one (discussed in Subroutine NORM1). To determine the inward pointing normal segments for all the other edges, a rotational transformation of the point ((PINMID(1,1),PINMID(1,2))) was performed through an angle 'THETA' (where 'THETA' was the angle between successive edges. The subroutine ACANG2 was called to determine the angle 'THETA'.

SUBROUTINE PLAREA: This subroutine calculated the area of the contour considering each element as a straight line. The coordinates of the vertices of the polygon were passed

through the array POLYLN. The area and centroid of the polygon were computed as described in the algorithm.

SUBROUTINE CRAREA: This subroutine calculated the area contributed by the circular portions. The coordinate values of the vertices, the midpoints of the edges, and a point on the normal to each edge which was at a unit distance from the midpoint of the edge were passed to the subroutine. Additionally, the POLARC which indicated if a vertex formed an end point of an arc, and FLPLN, which indicated if the edge was flipped or not, were passed.

If the value POLARC(i,1) and POLARC(i+1,2) were equal to 1, then the vertices i, and i+1 formed the endpoints of an arc element. The arc center location on the normal was determined by treating the normal as a parameterized line represented by

$$XC=XO+T(XN-XO)$$

where

XO was the beginning of the line segment

XN was the end point of the line segment

XC was any point on the line segment

T was the parameter value for the point XC.

The positive direction was from XO to XN. The location of the arc center on the normal was determined by computing the T value of the arc center. The subroutine FINDT was called to compute T. If T was less than 0.02 , then the arc was a semi-circle. The parameter values of the arc end points were computed using the subroutine PARMET, and the arc element was resolved to one of the cases described in the algorithm, and its area was computed. The centroid of the arc segment and the centroid of the composite area were determined as described in the algorithm.

SUBROUTINE MAXIR: This subroutine determined the maximum radius of the contour. The distance between each vertex and the centroid was computed. If there was an arc element, the distance of the farthest point on the arc from the centroid was also considered. The maximum of the computed displacements was found by pairwise comparison of the displacements. The coordinates of the vertices, centroid were passed.

SUBROUTINE MINIR: This subroutine determined the minimum radius point on the outer contour. The distance of each vertex from the centroid was computed, and the shortest distance of the centroid from each side was computed. If a side was composed of an arc element, then the distance of the nearest point on the arc from the centroid was also computed. The

minimum value of all the computed displacements gave the minimum radius of the outer contour.

SUBROUTINE PARMET: This subroutine calculated the value of the parameter for the end point of an arc element. The procedure was a translation of the method to calculate the parameter value that was discussed in the algorithmic procedure that describes the calculation of perimeter. The values of the coordinates of the end point of the arc element, and those of the center of the arc element were passed to the subroutine.

SUBROUTINE INTLIN: This subroutine determined the intersection point between two line segments. The values of the coordinates of the end points of both the lines were passed to the subroutine. The procedure checked if the intersection point were within the line segments or not and passed this information out.

SUBROUTINE ACANG2: This subroutine determined the angle subtended by an arc at its center. The values of the coordinates of the arc center, and those of the end points of the arc were passed to the subroutine and the value of the arc angle was passed out.

SUBROUTINE INTCIR: This subroutine determined the intersection points of a line segment and an arc element. The values of the coordinates of the end points of the line segment, arc end points and the arc center were passed to the subroutine. The coordinates of the points of intersection were passed out.

SUBROUTINE ARCENG: This subroutine determined the coordinates of the center of area of a chord. The values of the coordinates of the end points of the chord, center of the arc that forms the chord, and the radius of the arc were passed to the subroutine. The coordinates of the centroid of the chord were passed out.

SUBROUTINE SLPLN: This subroutine calculated the slope of a line segment. The coordinates of the end points of the line segment were passed to the subroutine, and the slope of the line segment was passed out.

SUBROUTINE FINDT: This subroutine took the values of the coordinates of end points of a line and converted it to a parametric representation of the form

$$X = X_0 + t(X_1 - X_0)$$

$$Y = Y_0 + t(Y_1 - Y_0)$$

Where

X and Y were the coordinates of any point on the

infinite line

X_0, Y_0 were the coordinates of the first point
or the origin of the line segment

X_1, Y_1 were the coordinates of the second or
end of the line segment

t was the parameter value of a point lying
on the infinite line with coordinates X, Y .

If the t was between 0 and 1, the point with coordinates X, Y was within the line segment formed by (X_0, Y_0) and (X_1, Y_1) , otherwise the point with coordinates X, Y was outside the line segment formed by (X_0, Y_0) and (X_1, Y_1)

In this subroutine the coordinates of the endpoints of the line segment and the point for which the parameter value was sought were passed to the subroutine. The value of the parameter was passed out.

4.0 VISION SYSTEM

In this chapter, a brief description of the vision system used in this research is discussed. The ASSEMBLY language subroutines that were written to access and modify the the picture elements are described. A detailed discussion of the processing of the binary image of an object placed in front of the vision system is also presented.

4.1 IMAGING TECHNOLOGY SYSTEM DESCRIPTION

The machine vision system used in this research was comprised of the components as shown in figure 13. The PCVISION system was hooked to the IBM-PC/AT to process the image of an object grabbed by the frame grabber of the PCVISION system. The major components of the PCVISION system were the frame grabber, the video camera, and the external monitor. There were three major components in the frame grabber. They were: the digitizer, the display logic, and the frame memory. The image of the object was captured in the form of an analog signal by the video camera. The analog signal was RS-170 compatible and needed to be transformed to a digital form to be able to be stored in the frame memory of frame grabber.

The incoming video signal was digitized to 8 bits of resolution at a rate of 30 frames per second. The frame memory had 512 rows and 512 columns, but only 480 rows and 512 columns were actually used. So, the total number of pixels or picture elements in the stored image was 480 x 512. Since each pixel was represented by 8 data bits, 256 possible shades of gray were possible. The stored image was displayed on the video monitor by the display logic, which converted the stored digital image back to analog form.

The PCVISION frame grabber was directly compatible with the bus structure found in the IBM personal computer. This interface was the communication path between the IBM PC and the PCVISION frame grabber. The frame memory interface enabled the IBM-PC to access the frame memory. The PCVISION frame grabber mapped 64 kbytes of its frame memory into the memory space of the IBM PC. Since there was a total of 256 kbytes of frame memory on the PCVISION frame grabber, only one fourth of the frame memory was accessible from the IBM PC Bus at any time. But FORTRAN callable routines written in ASSEMBLY language provided access to the frame memory and mask this hardware memory segmentation.

The frame memory subsystem was capable of simultaneously acquiring and displaying an image. This was accomplished with a read/modify/write cycle on each frame memory access. The

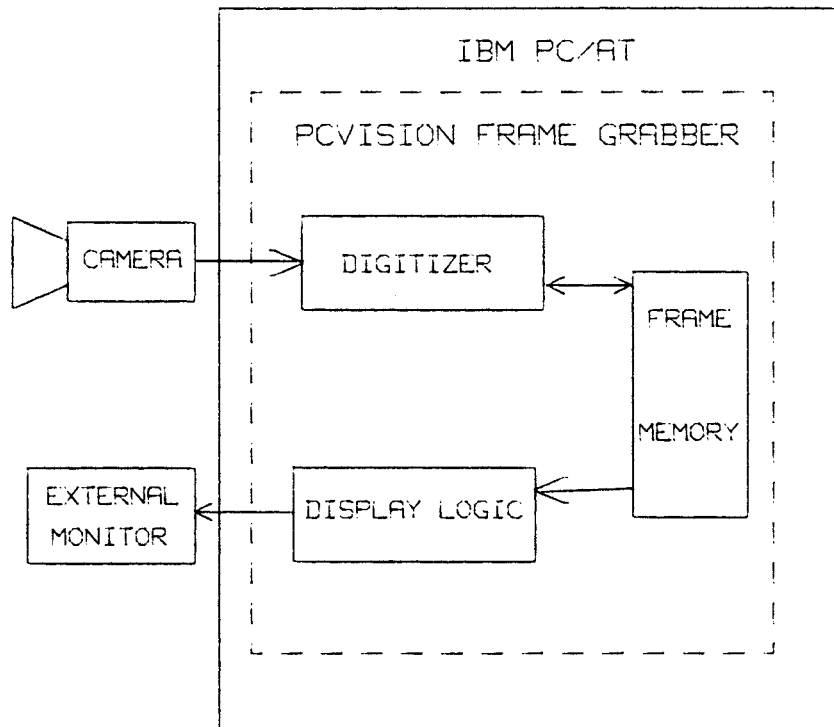


Figure 10. Machine Vision System

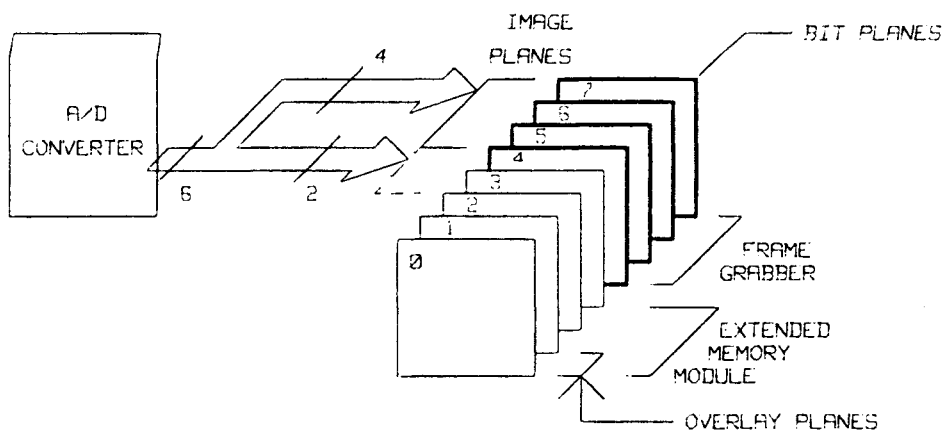


Figure 11. Frame memory organization

pixel which was read was transmitted to the display logic for digital-to-analog conversion, and the new pixel from the digitizing logic was written into the memory. Therefore, a one frame lag existed (1/30th second) when the PCVISION frame grabber was simultaneously acquiring and displaying an image.

The PCVISION frame grabber display logic contained 4 output lookup tables (LUT). Each look-up table was 256 bytes deep and is programmable through the IBM PC bus interface. Look-up tables were used to transform the pixel intensities prior to display. This permitted threshold viewing of grey scale images that were converted to binary. Different LUT's could be used to visualize effects of possible threshold values.

4.2 PROGRAMMING THE PCVISION FRAME GRABBER

FORTRAN callable ASSEMBLY language routines were written to assist the processing of the the image stored in the frame memory. Each subroutine is discussed briefly below.

SUBROUTINE VINIT: This subroutine initialized the PCVISION frame grabber, look up tables etc. It was called prior to any other vision system subroutine call.

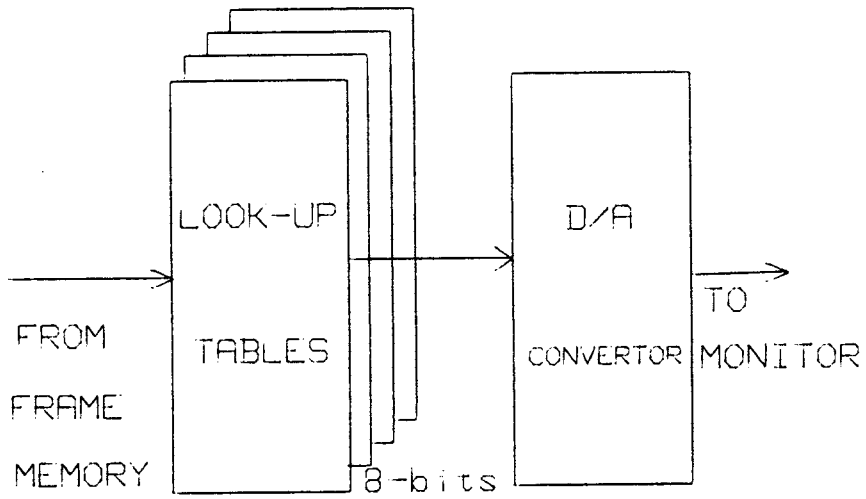


Figure 12. Display logic

SUBROUTINE FREEZE: This subroutine grabbed the image of the object, and stored it in the frame memory. This was called before the processing of the image.

SUBROUTINE RAMP: This subroutine was called to program the lookup table with a ramp function. This was done after calling the VINIT and FREEZE routines but before prior to the processing of the image.

SUBROUTINE THOLD: This was called to set the output lookup table for binary thresholding on the image. This permitted the use to visually see the results of a specific threshold value before using it in the image processing algorithm.

SUBROUTINE VRDROW: This subroutine allowed reading of a single row of pixel data from the frame memory. The intensity returned was a gray scale value between 0 and 255 and was processed by the vision program.

SUBROUTINE VWRROW: This subroutine allowed writing of single row of pixels to the frame memory. The original intensity values of those pixels which were written were modified to the new values.

The above subroutines enable the user to convert a grey scale image to a binary image and allow the user to read from and write to an individual pixel.

4.3 VISION ALGORITHM

The first step involved in the grabbing and processing the image of an object placed in front of the PCVISION system was to call the routines that initialize the vision system and grab the image. Subroutine calls VINIT, FREEZE, and RAMP accomplished this. The grabbed image had to be converted to a binary image since, only a binary image was considered in this research. In order to achieve this, an appropriate threshold value had to be selected by calling the routine THOLD. The intensity values of the pixels which fell below the threshold value had to be made zero and the intensity of the pixels which fell above the threshold value had to be made 255. To achieve this, a call to the routine VWRROW was made. Then, the image processing program was executed on the modified image. The details of the above mentioned steps are discussed below.

4.3.1 BINARY PROCESSING

In this research only a binary image of the object was considered for parameter extraction. The image of the object

stored in the frame memory initially existed in different intensity levels. So to segment the object from the background a 'thresholding' operation was performed on the image. In its most general form, thresholding may be defined by the relation

$$g(x,y) = k \text{ if } T_k \leq c(x,y) < T_{k+1}$$

$$k = 1, 2, \dots, m-1$$

where

(x,y) = pixel coordinates.

$c(x,y)$ = characteristic feature function
(e.g. intensity)

$g(x,y)$ = segmented function

T_i = i th threshold value

m = number of distinct threshold levels
assigned to the image

If we let $p(x,y)$ denote some local property (e.g., average intensity) of the point (x,y) , then T_i may be viewed as a function of the form

$$T_i = [x,y,p(x,y),c(x,y)]$$

If T_i depends only on $c(x,y)$, it is called a global threshold; if it depends on $p(x,y)$ and $c(x,y)$, it is called a local threshold; finally, if it depends on $p(x,y)$ and $c(x,y)$, and the coordinates (x,y) , it is called a dynamic threshold.

Global thresholding is suitable in applications where the objects to be extracted vary markedly from the background in some characteristic. Local thresholding is useful in situations involving gray-scale images where the difference of objects and background is not as clearly defined. Dynamic thresholding is appropriate when the contrast between different parts of the object has to be enhanced in order to gain more information about the object. For this research, global thresholding was appropriate since the objects used were simply two dimensional patterns or silhouettes of the views stored in the CAD database. In global thresholding the brightness of each pixel is compared against a common threshold level (e.g., background intensity) and a value of 1 is assigned wherever the threshold is exceeded and 0 is assigned if otherwise. The result is a binary image.

A histogram of the pixel data and the dynamic thresholding capability of the PCVISION system were used to determine the common threshold level. As can be seen in Figure 13 the histogram showed the number of pixels with a given gray level. The presence of two strong peaks and a valley of zero reading permitted easy thresholding to separate the frame into object pixels and background pixels. Additionally, the THOLD subroutine was also used to view the results of selecting a specific value for the threshold. This allowed alteration of the value for specific lighting conditions.

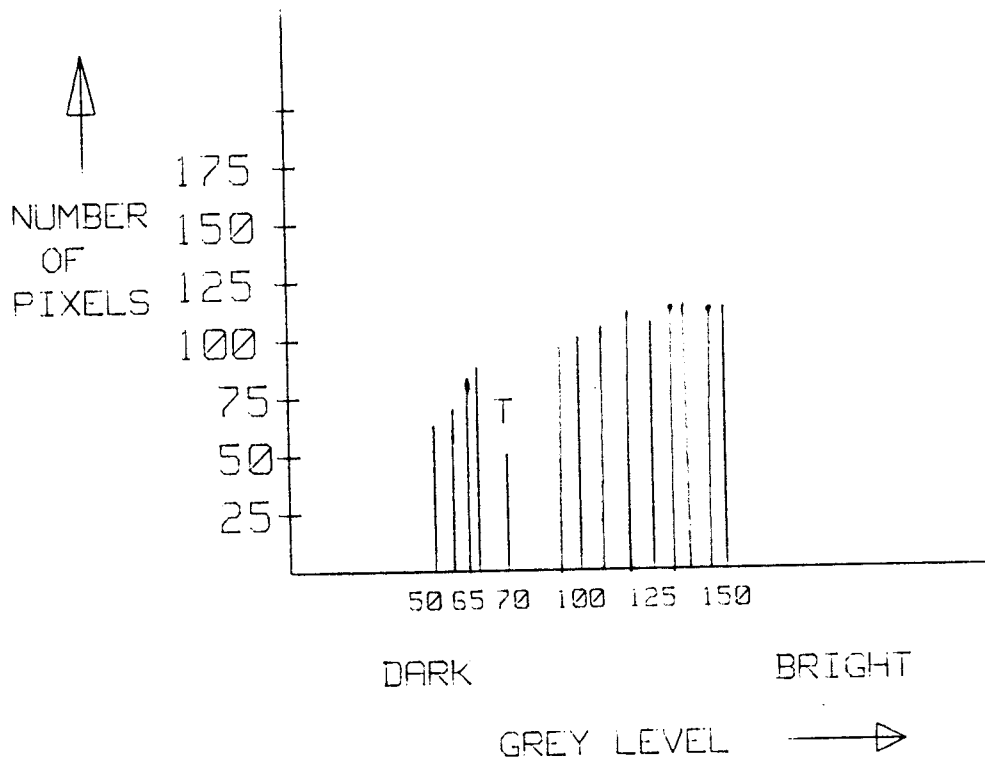


Figure 13. Histogram

Connectivity analysis and boundary following are two widely followed approaches to extract shape parameters from a binary image of the unknown object placed before the camera. These approaches were discussed in the literature search in terms of their processing strategies. There are also other considerations such as hardware restrictions and number of memory accesses required and ease of processing in future. Connectivity analysis is the more elegant approach of the two in that all processing is done in a single raster scan in the image. The boundary tracker alternates between a raster scan search for new blobs and following the boundary, which requires random access to the image.

4.3.2 CONNECTIVITY ANALYSIS ALGORITHM

The algorithm scans the image line by line, top to bottom and processes each pixel. The contribution of each pixel to area, perimeter, and other shape parameters was added to the blob or region in which it was located. A blob here is defined as a region comprised of pixels of the same color and is bounded by another of region of a different color. A run-length encoding scheme was used to compress the information on each row of data. In the coding scheme, used by the connectivity analysis algorithm, each unbroken run of 0's or 1's in a raster line of the image was described by a record that tells its starting (leftmost) column, its length, and

which blob the run belongs to. Each line of the picture was then described by a list of these run-length records.

The run-length array data structure for the connectivity analysis algorithm is shown in Figure 14. The background is assumed to be blob-1 and the object, blob-2. The four numbers in each record are interpreted as the start of a run, the number of pixels in the run, the number of the blob these pixels belong to, and the color of the run. An array was maintained to keep all the parameters of a blob.

At the end of each run of 0's or 1's, the run-length array was updated, and blob statistics are accumulated. If any pixel of the run just completed is connected to a pixel of the same color on the previous line, an existing blob was extended to include this run. Otherwise, a new blob was created. Each blob was assigned a unique number. Pixels belonging to the blob were labeled with this number in the run-length array. A blob may initially have more than one label associated with it, as shown in Figure 15. It is not until the scan reaches point A that the two components labeled 1 and 2 are recognized as being part of the same blob. At this point they are merged - the record for blob-2 is released after combining statistics of blobs 1 and 2.

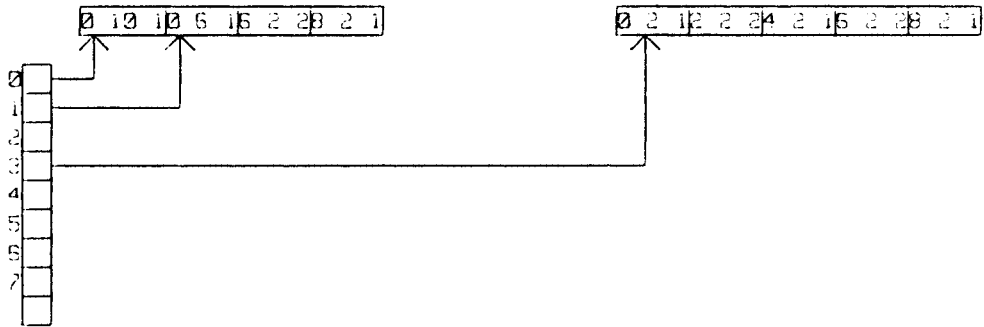
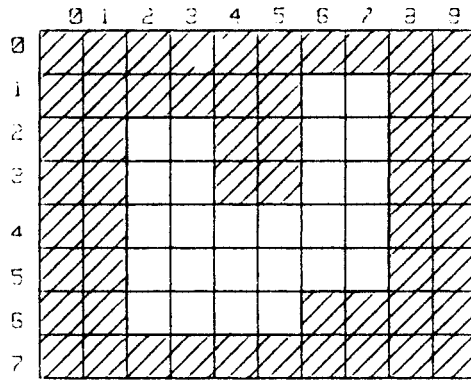


Figure 14. Run-length code

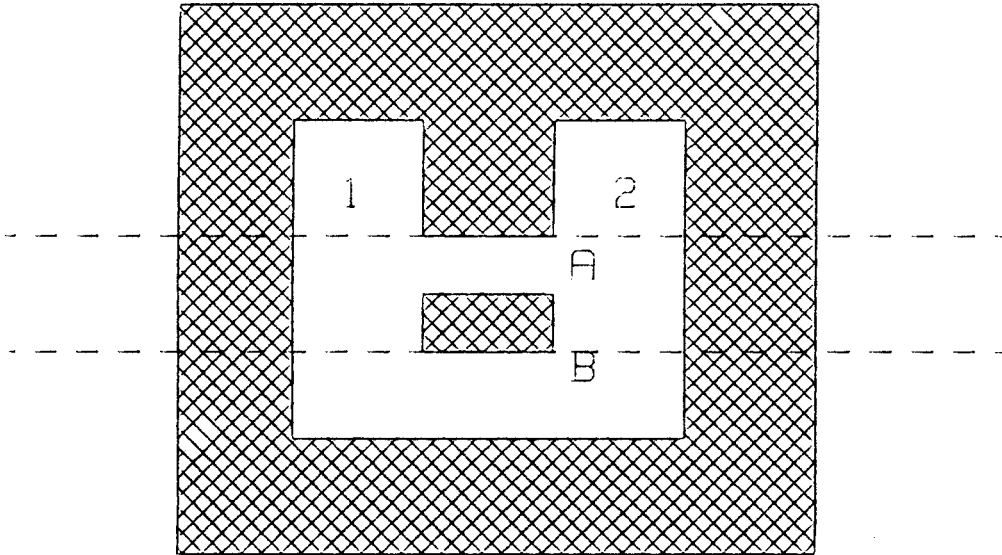


Figure 15. Multilabelled blob

Whenever there is a change in color, it indicates the end of a run and hence the blob statistics need to be updated. To detect a transition, the image is scanned with what amounts to a 2 by 2 pixel window into the image whose lower right corner is always the current pixel. Thus, the previous pixel on the same line and the two corresponding pixels on the previous line make up the other elements in the window. Since each pixel can be colored either black or white, there are 16 possible patterns that can appear in the 2 by 2 window. This is shown in Figure 16. These window states are numbered by treating the values of the pixels in the window as a four bit binary number, according to the formula shown in the Figure 16. Only 8 state pairs need be considered since the transitions 0, 1, 2, 3, 4, 5, 6, 7 are reciprocal of states 15, 14, 13, 12, 11, 10, 9, 8.

The results of the analysis of each state of interest is discussed below.

STATES(7,8): It is possible that this state indicates the start of a new blob. Hence a pointer, NEWBLOB, is set to indicate this. It is also the beginning of a new horizontal boundary segment (which will contribute to the perimeter) and a new run. The current column is saved for future update. The parameters are updated for the run just completed.

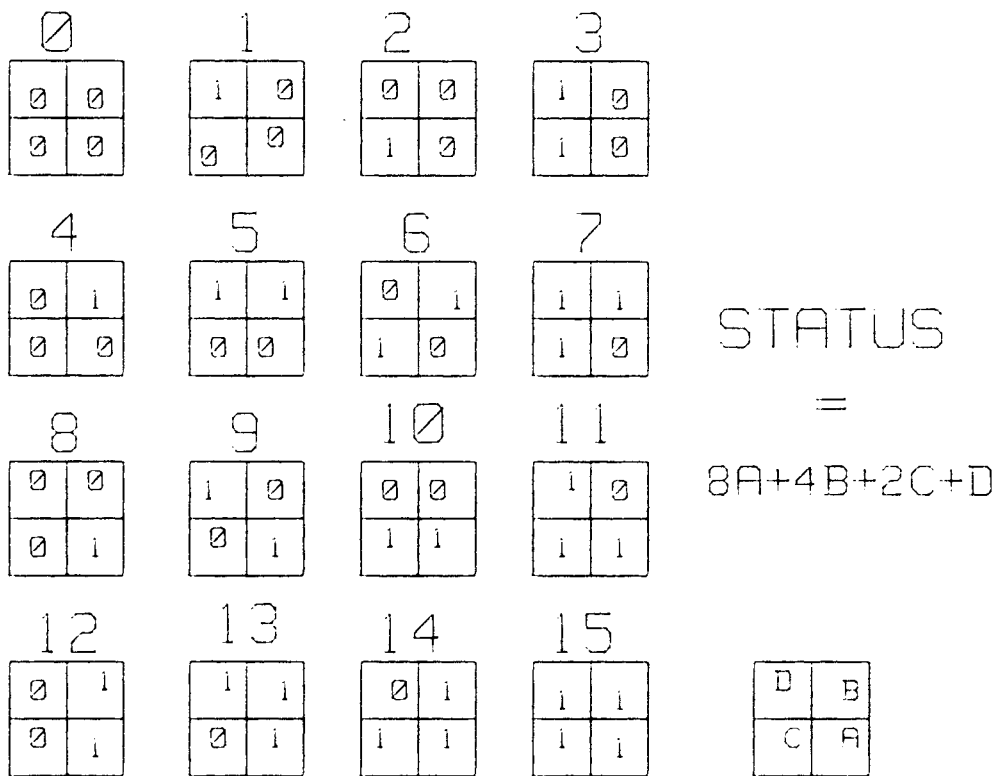


Figure 16. The 16 possible states of a pixel

STATES(4,11): This marks the beginning of a new horizontal boundary segment. So the current column is saved. The pointer BLBPTD is set to the blob of pixel B which is obtained by looking it up in the run-length array.

STATES(3,12): The parameters for the run just ended, and the run-length array are updated. BLBPTC and BLBPTD are set to the blob of pixel B.

STATES(2,13): If NEWBLB is true, a new blob is created for the run just completed. The NEWBLB is set to false and BLBPTC is set to the number of the new blob. Then the parameters are updated including the perimeter which is the length of the run just completed. The BLBPTC is set to the value of BLBPTD.

STATES(6,9): If NEWBLB is true then a new blob is created as in the previous case. BLBPTC is set to the value of BLBPTD

STATE 1,14: Three possibilities have to be considered. If NEWBLB is true then what was originally assumed to be a new blob has merged with an existing blob. If NEWBLB is false then either two distinct blobs have merged or a blob has closed on itself surrounding the blob containing pixel D. The variable HOLE is set to the value of BLBPTD and the value of BLBPTD is set to the blob of pixel B. If NEWBLB is true then the value of BLBPTC is set to the value of

BLBPTD and NEWBLB is set to false. This will include the new run in the old blob.

If NEWBLB is false and BLBPTC is not equal to BLBPTD, then the two blobs are merged by releasing BLBPTD, deleting it from the blob list and combining both sets of statistics into the blob BLBPTC. If however, NEWBLB is false and BLBPTC is equal to BLBPTD then this indicates the presence of a hole. If there is no hole the perimeter of the assumed hole is subtracted from that of the blob of BLBPTC

4.3.3 PARAMETER EXTRACTION PROCEDURES

The parameters that were considered in this research are:

Area

Perimeter

Number of holes

Maximum radius

Minimum radius

CALCULATION OF MOMENTS: The parameters area, and the first moment of area are determined by calculating the moments of the given region. The procedure is discussed below. For any region, we can generate an infinite series of moments. But for our purpose, the first three terms of the series suffice.

The first moment gives the area of the area of the region, the second and third moments give the first moment of the region about X and Y axes respectively. Let x and y be the coordinates of an arbitrary point (pixel) in the image. Let x_L and y_L be the value of x and y at a point L in the run. For any $L \geq 1$, let

$$\delta X_L = x_L - x_{L-1}$$

$$\delta Y_L = y_L - y_{L-1}$$

The moments for a particular run-length containing 'n' pixels are given by:

Let

$$A_L = x_L * \delta y_L - y_L * \delta x_L$$

Then the first six moments are given by:

$$M_{00} = 1/2 * \Sigma A_L$$

$$M_{10} = 1/3 * \Sigma A_L (y_L - 1/2 * \delta y_L)$$

$$M_{01} = 1/3 * \Sigma A_L (x_L - 1/2 * \delta x_L)$$

Perimeter is determined by adding the run-lengths and to account for the vertical length, the height of the pixel is added. The coordinates of the center of area were found by dividing second and third moments by the area. In order to

determine the maximum radius, the distance of the end pixel of each run-length was computed and that distance which was maximum was the maximum radius. To determine the minimum radius, as discussed above the distance of the end pixel of each run-length from the centroid was computed and besides this the shortest distance of each edge from the centroid was also considered. The minimum of the above distances was the minimum radius. The procedure for the determination of the minimum radius is valid only for objects without holes.

4.4 VISION SYSTEM PROGRAM

A listing of the program can be found in Appendix B. It was assumed that at least one pixel of background color borders the object. Initially the assembly subroutines VINIT, GRAB, FREEZE, RAMP were called to initialize the vision system and freeze the image of the object. The subroutine THOLD was called to threshold the image. The program was written to enable the user to give a threshold intensity value interactively so that the user can give an appropriate value depending on the gray level of the object. Then the subroutine VRDROW was called to access a row of pixels. The status of each pixel in a row was calculated to one of the fifteen states and depending on the status an appropriate subroutine is called. The color, start, end columns of each run are stored in an array. In addition to the image processing pro-

cedure, the program also contains the procedure to recognize an object placed in front of the vision system as one of the known objects. This procedure is discussed in Chapter 6.

4.5 DESCRIPTION OF SUBROUTINES

SUBROUTINE ALLOC: This subroutine allocated a new region or blob and called the subroutine REGINI to initialize the parameters of the newly allocated blob. This subroutine was called when the status of the current pixel was (2, 13,) or (6, 9).

SUBROUTINE REGINI: This subroutine was called to initialize all the parameters for a newly allocated blob. This was called once at the at the beginning of the program and was called by the subroutine ALLOC.

SUBROUTINE UPDATE: This subroutine updated the parameters at the end of each run. The procedure for the calculation of the parameters area, perimeter, and moments, were included in this subroutine. A run-length array was allocated for the run just ended. The run-length array contained the starting column, ending column, and the color of the run.

SUBROUTINE LOOKUP: This subroutine was called to identify the blob number of a particular run. This was useful to determine the question whether a blob was valid or one that had been merged with another blob.

SUBROUTINE RADCAL: This subroutine was called to calculate the distance of end pixels of each run of black color from the center of area of the blob. The distances were used to determine the maximum radius, that is the distance of the farthest pixel from the center of area of the blob.

5.0 INTEGRATION OF CADAM AND PCVISION SYSTEMS

The steps involved in the integration of the computer-aided design system and the machine vision systems are discussed below. The procedure to calibrate the vision system is discussed. A detailed discussion about the determination of the variation of the shape parameters such as area, perimeter, maximum radius, and minimum radius in the vision system is also given.

5.1 FILE TRANSFER

Initially, graphic models of the components that were to be recognized by the machine vision system were created in the CADAM system. The FORTRAN program that processed the graphic models to extract shape parameters was executed and the output was written to a separate CMS file. This file was then transferred to the IBM PC/AT that contained the machine vision system. The YTERM software package, that allows a user to connect a PC to the main frame computer, was used for this purpose. The execution of the transfer of data required a three step process:

First the program was initialized by entering the following commands.

```
CD\YTERM
```

```
X
```

```
T 96 B
```

Next the user logged onto the main computer by entering these main frame computer command:

```
LOG <user-id>
```

Then, the user entered his main frame computer password.

Finally, the user issued the following command to transfer the CADAM output file to the PC vision system.

```
PCTRANS DOWN <pc file-name> <CADAM output>
```

The data available in this file was used to compare against the shape parameters of the object presented to the machine vision system generated by executing the vision system program. The format used was F6.2 for the parameters area, perimeter, maximum radius, and minimum radius. The data start from column-1 on each row and the sequence of the parameters is area, perimeter, maximum radius, and minimum radius. An example of the format used is given below:

11.83

15.81

2.83

1.56

5.2 CALIBRATION OF THE MACHINE VISION SYSTEM

The field of vision of the machine vision system is 480 rows by 512 columns. This means that the length of each pixel is $1/512$ part of the length of the field of vision and the height is $1/480$ part of the height of the field of vision. Since machine component dimensions were expressed in inches, it was decided to adopt the same unit of measure to the length and height or width of the pixel.

To calibrate the system, three square paper blocks of size 1", 2", 3" were selected as test objects. A FORTRAN program was then written which calls the vision system subroutines VINIT, GRAB, FREEZE, RAMP, THOLD, and VRDROW, to capture and threshold the image, and to read the pixel intensity values. Each sample block was placed in the field of view of the vision system and a sample frame evaluated for part alignment. The block was then adjusted until the top edge of the block was represented by a single row of pixels for the entire width of the image.

It was found that the one inch square occupied 42 pixels along the length and 56 pixels along the height; that is one square inch occupied 42 columns and 56 rows. The test program was replicated on the two and three inch squares and there was no variation in the ratio of the number of pixels along the length of the object and the number of pixels along the height/width of the object. Hence a unit pixel pixel was assumed to have a length of 0.02381 inches and a height of 0.01786 inches.

5.3 VISION SYSTEM PARAMETER RANGES

It was imperative to test a vision system processing algorithm for accuracy in computation and for variability of parameter values with respect to different orientations. The range of values could then be used to modify the parameters obtained from the CAD system in order to establish an object recognition system. The above necessity arises from the facts that inaccuracies may arise due to selecting inappropriate threshold values, and the variation in the number of pixels occupied with respect to orientation. Additionally, type of connectivity analysis used or the algorithms implemented may also affect the accuracy of measurement of specific parameters.

Three test objects of known dimensions and shape parameters were used to test the vision system program. One object was a square of side 2 inches, the other was a trapezium of sides 4, 3 inches and height 2 inches and the third object was a rectangle of dimension 4 x 3 inches. The parameters considered were area, perimeter, and the minimum and the maximum radii. The vision system program was executed for each test object kept under three different orientations. The range of variation of each parameter for each object is given in Table 1.

Graphs between the actual value and the observed value of each parameter were plotted to find the range of variation of that parameter. These graphs are given in Figure 17, Figure 18, Figure 19, and Figure 20. The line drawn at 45 degrees is used as the line of reference since, the actual and observed values are the same on this line.

AREA: The area parameter was found to have a variation of 7.2 percent on the positive side and there was no variation on the negative side (Figure 17). The reason for this could be that the threshold value selected could have been improper and this could have added more pixels than were actually covered by the object. Additionally, in the case of the edge pixels, the entire area of a pixel is added instead of adding

only a fraction of the pixel that is actually covered by the object.

PERIMETER: This parameter was observed to have a wider variation than area. This variation of 17 percent, was also on the positive side (Figure 18). This could be attributed to the type of connectivity used and the type of approximation used in the in the calculation of the distance between diagonally placed pixels. Additionally, the increased area mentioned above contributed to an increase in the perimeter of the image.

MAXIMUM RADIUS: The maximum radius had a variation of 4.1 percent on the positive side (Figure 19). The variation of area has a direct bearing on the variation of the maximum radius. If the variation of area is on the positive side then, this means that more area is being accounted for than the actual area. Consequently, the distance of the farthest pixel will also be more than the actual value.

MINIMUM RADIUS: The minimum radius had a variation of 6 percent on the positive side (Figure 20). The reason for this is the same as that which explains the variation of the maximum radius.

Table 1. Variation of parameters

| OBJECT | AREA VARIATION | PERIMETER VARIATION | MAX.RAD VARIATION | MIN.RAD VARIATION |
|--------------------|-------------------|---------------------|-------------------|-------------------|
| 2"x2" square | 4.15 to 4.21 | 8.38 to 9.0 | 1.43 to 1.45 | 1.01 to 1.06 |
| 4"x3" rectangle | 12.91 to 12.98 | 15.0 to 15.11 | 2.59 to 2.62 | 1.5 to 1.56 |
| 4"x3"x2" trapezoid | 7.42 to 7.33 | 12.12 to 13.5 | 2.23 to 2.30 | 1.0 to 1.05 |

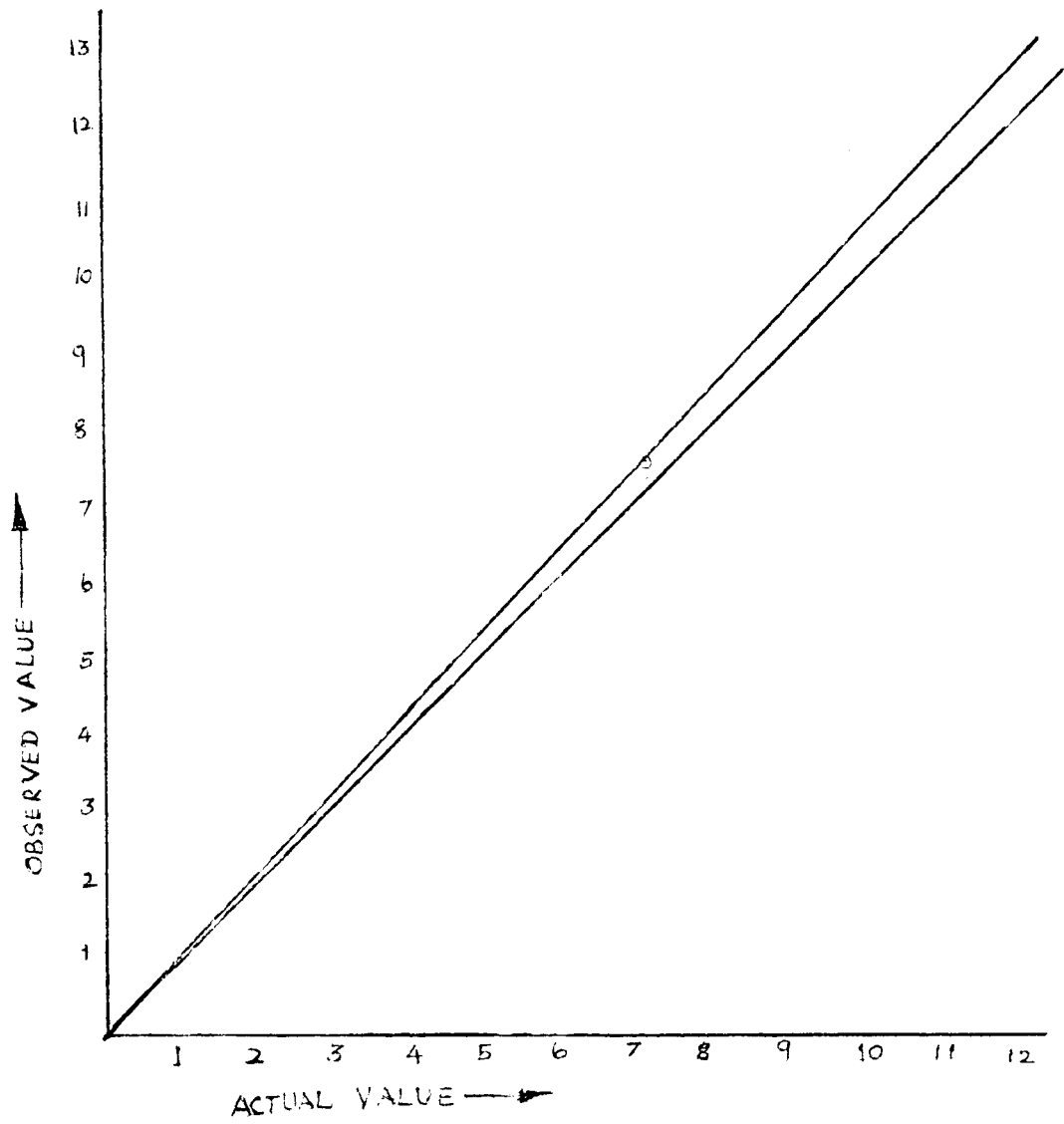


Figure 17. Variation of area

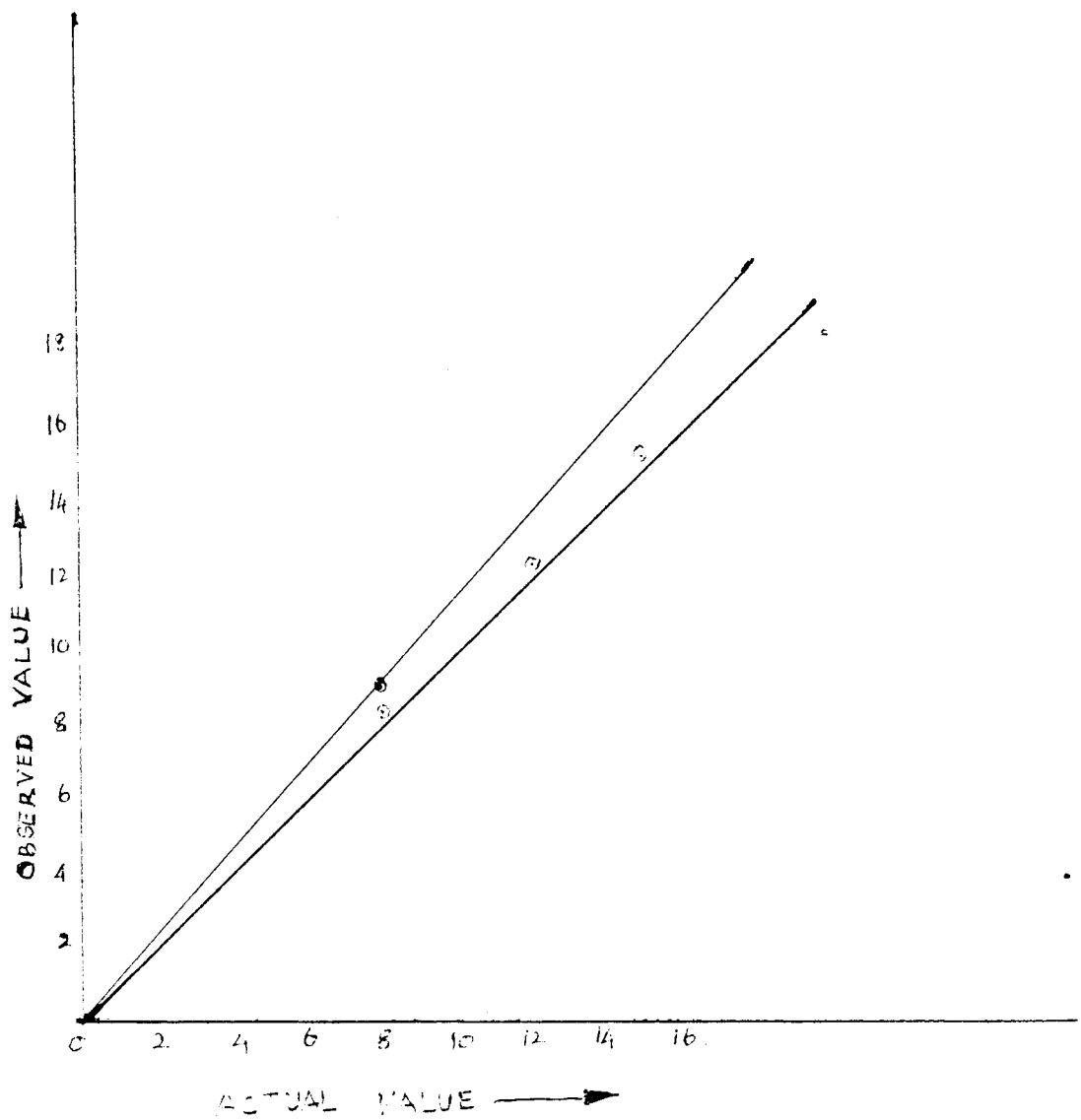


Figure 18. Variation of perimeter

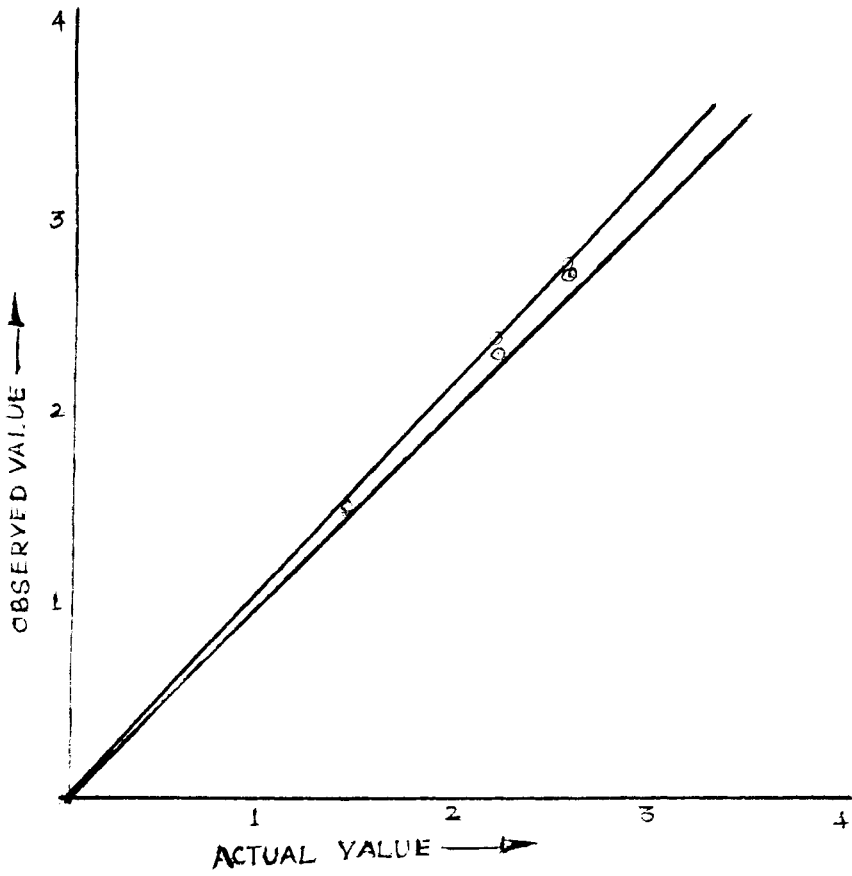


Figure 19. Variation of maximum radius

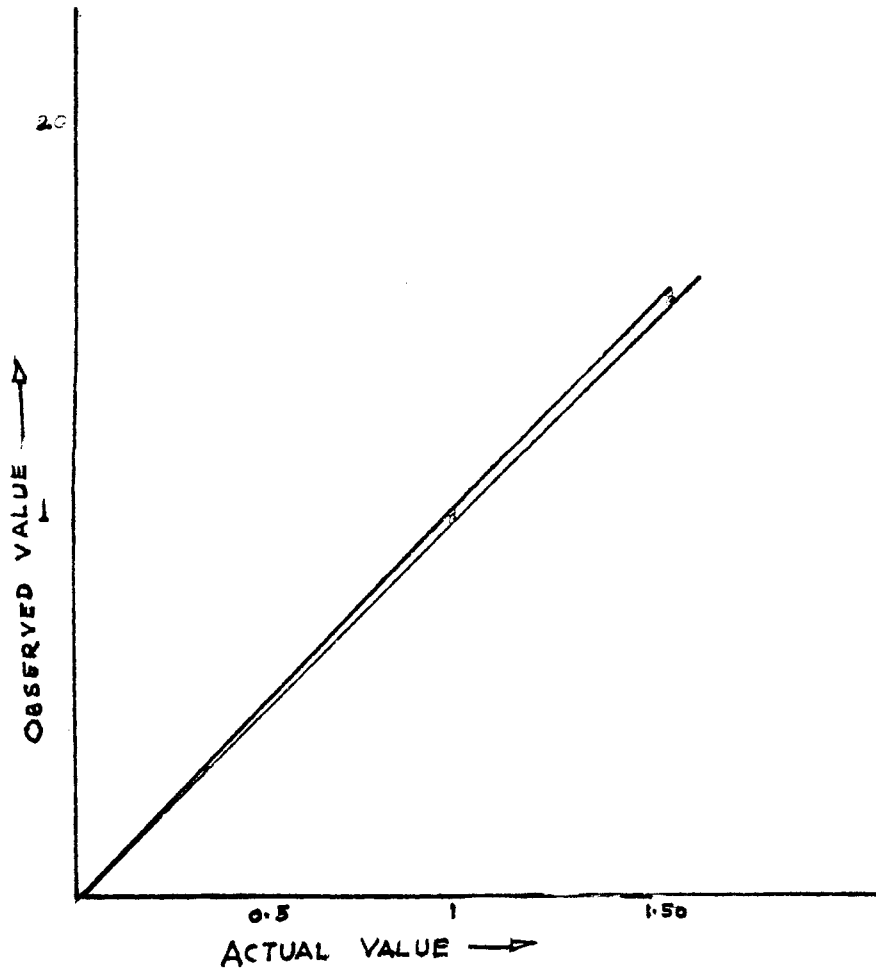


Figure 20. Variation of minimum radius

6.0 RESULTS AND CONCLUSIONS

It has been shown in Chapter 3, that CAD based geometric part data can be processed to obtain parametric data describing a two dimensional silhouette of an object of interest. Chapter 4 indicated a method of extracting the same parameters from a frame of data gathered from a computer vision system. Finally, Chapter 5 indicated a means of calibrating the vision system to establish a relationship between the two data systems. Further more, it outlined a means of establishing a range of variability associated with each parameter based on actual value.

The final proof of the integration of the two systems was to test the use of CAD based data files to identify different objects. Initially, the shape parameters of three graphic models created in the CAD database were passed to the vision system following the method discussed in the previous chapter. The percentage of variation of each parameter was then applied to establish ranges of acceptability. Then, each shape parameter of the unknown object processed by the vision system program was checked to see if it was within the range of acceptable values. This process was repeated for all parameters of each known models. If all the shape parameters

of the unknown object lie within their respective range, then the unknown object is recognized as that object.

Table-2 shows the values of each parameter extracted by the CAD data processing program for each of the known model. Table-3 shows the range of values, each parameter can have for each model. Table-4 shows the values of each parameter extracted by the vision data processing program for each known model. It may be noticed, that the shape parameters extracted by the vision system program lie within their respective ranges. The figures of the three models used are shown in Figure 21, Figure 22, and Figure 23.

The above procedure indicated the feasibility of integrating an off-line CAD system and a machine vision system. This enables the manufacturing engineer to teach the robot whichever object he desires the robot to recognize, by creating a graphic model of the object in the CAD system instead of showing the object to the vision system. This also enhances the integration of the different components of a computer integrated manufacturing system.

While the research has demonstrated the feasibility of using CAD based data as an input for vision system models, certain restrictions were imposed on the model and the types of ob-

jects to be considered. Only two dimensional silhouettes of objects were considered and all vision processing was done on a binary image. Additionally, since only four parameters were considered, families of geometric shape with similar parametric ranges would be difficult to accommodate. Extensions to the present work would enhance applicability of the basic concepts.

First the number of parameters being analyzed could be increased. Since an unknown part must fall within acceptable limits on all parameters, this would increase the ability of the system to discriminate between similar parts. The added parameters must be chosen wisely. First they must not vary with part orientation. Next a means must be available to compute the parameter from the list of geometric elements given in the CAD system. Finally, a means must be created to evaluate the parameter from vision system data. Candidate parameters for addition to the research would include polar moment of inertia, product of inertia, minimum and maximum moments of inertia, number of corners, perimeter of enclosing convex hull, etc.

A major area of expansion of the work would include the use of three dimensional geometric data and gray scale vision systems. Extensions of work in this area would accommodate the geometric perspectives and camera position consider-

ations. Surfaces, contours as well as internal edge and surface features might also be considered. Expansion of the thesis in this direction would involve a significant amount of work both on the CAD system and within the vision processing algorithms.

Table 2. CAD system parameters

| OBJECT | AREA | PERIMETER | MAX. RADIUS | MIN. RADIUS |
|---------|-------|-----------|----------------|----------------|
| Model A | 3.41 | 7.4 | 1.41 | 0.808 |
| Model B | 11.84 | 15.81 | 2.83 | 1.56 |
| Model C | 14.79 | 19.82 | 3.09 | 1.54 |

Table 3. Parameter ranges

| OBJECT | AREA | PERIMETER | MAX. RADIUS | MIN. RADIUS |
|---------|-------------------|-------------------|------------------|------------------|
| Model A | 3.17 to 3.65 | 7.4 to 8.58 | 1.41 to 1.456 | 0.81 to 0.856 |
| Model B | 11.01 to 12.66 | 15.81 to 18.33 | 2.83 to 2.94 | 1.56 to 1.653 |
| Model C | 14.79 to 15.82 | 19.82 to 22.97 | 3.09 to 3.21 | 1.54 to 1.63 |

Table 4. Vision system parameters

| OBJECT | AREA | PERIMETER | MAX. RADIUS | MIN. RADIUS |
|---------|-------|-----------|----------------|----------------|
| Model A | 3.53 | 7.98 | 1.42 | 0.805 |
| Model B | 12.45 | 16.02 | 2.87 | 1.56 |
| Model C | 15.2 | 21.42 | 3.10 | 1.54 |

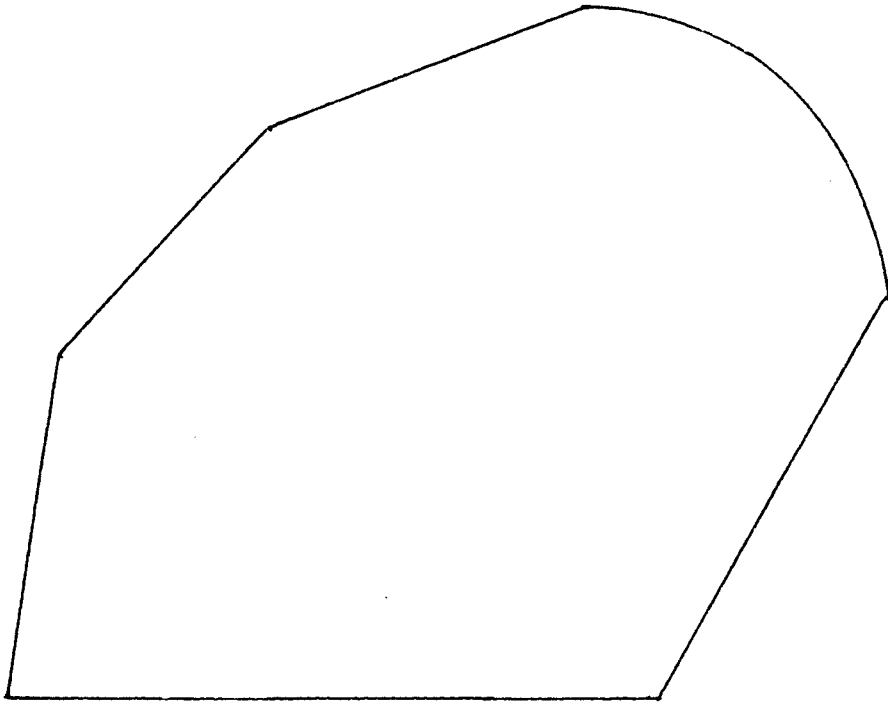


Figure 21. Test Model 1

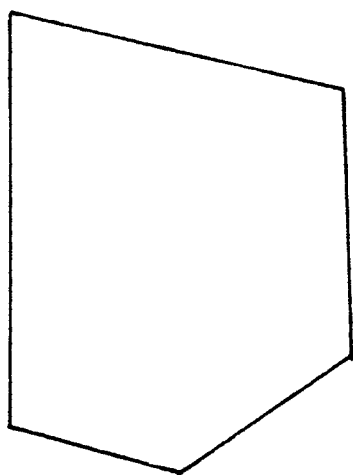


Figure 22. Test Model 2

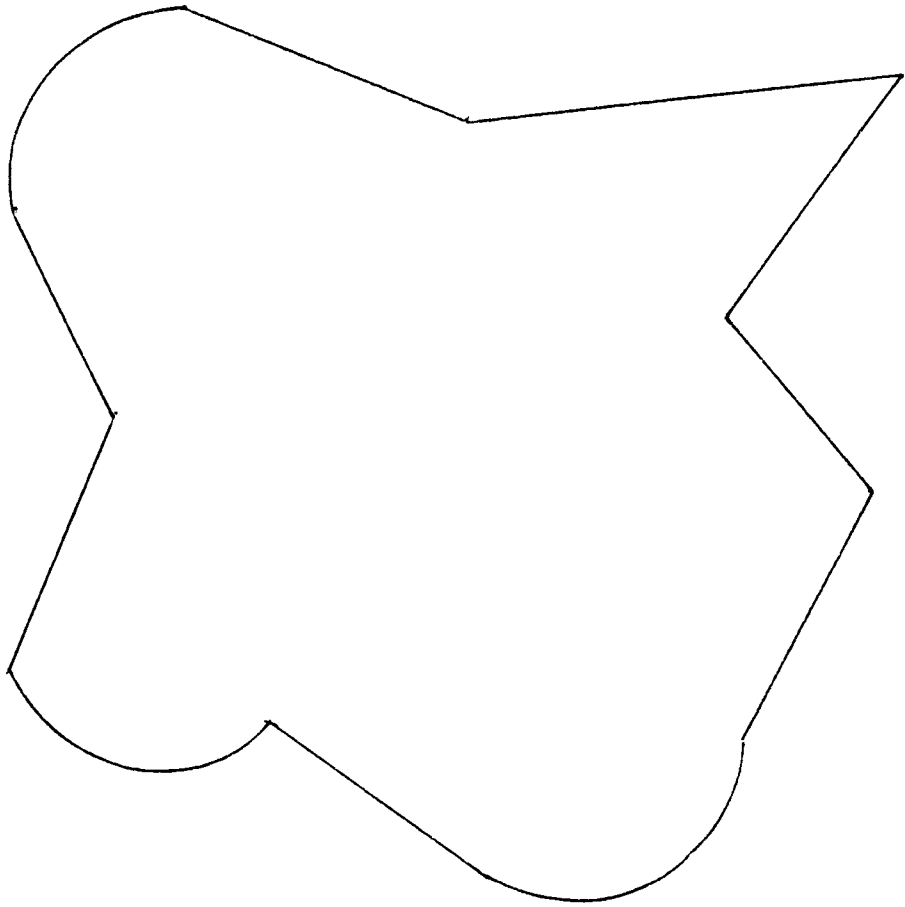


Figure 23. Test Model 3

LIST OF REFERENCES

1. Agin, G. J., and Duda, G. O., "SRI Vision Research for Advanced Industrial Application", Proc. 2nd USA-Japan Computer Conference, Tokyo, (1980).
2. Agrawala, A. K., "A Sequential Approach to Extraction of Shape Features", Computer Graphics and Image Processing, Vol.6,(1977).
3. Alexander, I., Artificial Vision for Robots, Chapman and Hall, New York, (1983).
4. Brice, R. O., and Hart, P. E., Pattern Classification and Scene Analysis, Wiley, New York,(1973).
5. Cederberg, R., "Chain-link Coding and Segmentation for Raster Scan Devices", Computer Graphics and Image Processing, Vol. 10, (1979).
6. Chen, C. H.,(ed) Pattern Recognition and Artificial Intelligence, Academic Press,New York, (1976).
7. Cunningham, R., "Segmenting Binary Images", Robotics Age, Vol. 3, (1981).
8. Curtis, S. B., "Computer Vision Inspection from a CAD Database", Robots 8 Proceedings, (1981).
9. Foley, J. D, and Vandam, A., Fundamentals of Interactive Computer Graphics, Addison Wesley Publishing Company, New York, (1984).
10. Freeman, H., "Computer Processing of Line Drawing Images", Computing Surveys, Vol. 6, No.1, March, (1974).
11. Gleason, G. J., and Agin, G. J., "A Modular Vision System for Sensor-controlled Manipulation and Inspection", Proc. of 9th Int'l Symposium on Industrial robots, Washington,D.C, (1979).
12. Gruver, W. A, "Off-line Robot Vision System Programming by Computer Aided Design," Robots 8 Proceedings, (1981).
13. Horowitz, S. L., and Pavlidis, T., "Picture Segmentation by Directed Split and Merge Procedure", Proc.2nd

Int'l Joint Conf. on Pattern Recognition., (Aug. 13-15,1974, Copenhagen), (1975).

14. Jarvis, R. A., "Application-oriented Robotic Vision- A Review", Robotica, Vol. 1, Jan. (1984).
15. Kitchin, P. W., Processing of Binary Images in Robot Vision, IFS Publications Ltd, London, (1983).
16. Kothari, T., "The Role of Vision Systems in Robotic Applications", Robots 8 Proceedings, (1981).
17. Nevatia, R., Machine Perception, Prentice-Hall Inc, New York, (1983).
18. Pot, J., "Comparison of Five Methods for the Recognition of Industrial Parts", in Artificial Vision for Robots, Chapman and Hall, New York, (1983).
19. Pratt, M. R., and Faux, I. J., Computational Geometry for Design and Manufacture, Ellis Horwood Publishers, London, (1981).
20. Pugh, A., (ed), Robot Vision, IFS Publications Ltd, London, (1983).
21. Rogers, D., and Adams, A., Mathematical Elements for Computer Graphics, Academic Press, New York, (1976).
22. Rosenfeld, A., and Thurston, M., "Edge and Curve Detection for Visual Scene Analysis", IEEE Trans. on Computers, C-20, No.5, PP567-569 (May, (1971).
23. Rosenfeld, A., and Kak, A., Digital Picture Processing, Academic Press, New York, (1976).
24. Sobel, I., "Neighborhood Coding of Binary Images for Fast Contour Following and General Binary Array Processing", Computer Graphics and Image Processing, Vol. 9, (1978).

APPENDIX A. CAD SYSTEM PROGRAM

```
SUBROUTINE RESOLV
DIMENSION A(2),B(2),C(3),ARRAY(100),ARRAE(100),
3 D(3),E(3)
COMMON /CDTCOM/ LETGO,NOREAD,CDTSHO,ELSHOW,
4 IPRIN,MODSIZ,MDLLET,HAWC(20,2),
*FILDAT(2),MDLTX(17),CIRPTS(20,4),NUMLIN,
*RAWC(20,4),PTLIN(20,4),NUMCIR,LINELE,IHOLIN,
*PTLNH(20,4),HOLPTS(20,4),NMHCIR,HOLS(10,3),LNPOHE
DIMENSION PT1(3),PT2(3),TEXT(64),DIR(3),START(3)
DIMENSION EMPNO(2),PART(5),TXT(19),FULL(18)
DIMENSION IWORD(18),NSTDTO(10),FDERIV(NOPTS)
DIMENSION XYZ(3,NOPTS),ABC(3,NOPTS),FANGLE(NOPTS)
DIMENSION S(NOPTS),TWIST(3,4)
DIMENSION HOMOG(4,1),UVEC(3,4),WVEC(3,4)
DIMENSION SL(6),CIR(15),E1(15),E2(15),DLTRAD(5)
DIMENSION HBP(4,16),ABC2(3,1)
INTEGER*2 LOOKB(200),LOOKR(200),NUMB(200)
INTEGER*2 ISEQ(200),KINDEL(200)
REAL*8 DOUBLE(9)
REAL POLYLN(20,2),HOLYLN(10,20,2)
LOGICAL*1 BYTE(72),SETNME(20)
LOGICAL*1 SPACTB(776)
LOGICAL CLC
INTEGER REDUNT,SPAN,ELSHOW
INTEGER*2 ID
REAL LINEWT
REAL POLYC(20,2),HOLYC(10,20,2),PINMID(10,2),
2 PTMID(10,2),PX(10),PY(10),ARCAR(20),
*CENARX(20),CENARY(20),XA(20),YA(20),XB(20),
*YB(20),MR(20),NEGP,MAXRAD,MINRAD,
*HTMID(20),
*SOLYLN(20,2),SOLYC(20,2),HINMID(10,2)
INTEGER TURN,SCALE,VERT,HORIZ,JUST,FONT,IFSUPR
INTEGER*2 IHALF(36)
INTEGER FLPLN(20),POLARC(20,2),HOLARC(10,20,2),
*IB(10),IE(10),ISLPLN(20),IPOARC(20,2),HLPLN(10,20)
CHARACTER*3 END
DATA IOTYP/1/
DATA IUNIT /6/
DATA NUMHOL /0/
```

C

C DESCRIPTION OF VARIABLES

C POLYLN(I,1)... X-COORDINATE OF A VETEX POINT OF THE
C..... CONTOUR

C POLYLN(I,2)...-COORDINATE OF A VERTEX POINTOF THE

```

C .....CONTOUR
C CIRPTS(I,1)...X-COORDINATE OF END-1 OF AN ARC ELEMENT
C CIRPTS(I,2)...Y-COORDINATE OF END-1 OF AN ARC ELEMENT
C CIRPTS(I,3)..X-COORDINATE OF END-2 OF AN ARC ELEMENT
C CIRPTS(I,4)..Y-COORDINATE OF END-2 OF AN ARC ELEMENT
C ARC(I).....VALUE INDICATING WHETHER AN ARC ELEMENT
C PTLIN(I,1)...X-COORDINATE OF END-1 OF AN LINE ELEMENT
C PTLIN(I,2)...Y-COORDINATE OF END-1 OF AN LINE ELEMENT
C PTLIN(I,3)...X-COORDINATE OF END-2 OF AN ARC ELEMENT
C PTLIN(I,4)..Y-COORDINATE OF END-2 OF AN ARC ELEMENT
C POLARC(I,J).. VALUE INDICATING WHETHER A VERTEX IS AN
C..... END POINT OF AN ARC OR NOT
C NUMPTS..... NUMBER OF VERTICES
C NUMLIN ..... NUMBER OF ELEMENTS
C POLYC(I,1).. X-COORDINATE OF THE CENTER POINT OF AN
C ..... ARC OF WHICH VERTEX I IS AN END POINT
C POLYC(I,2)... Y-COORDINATE OF THE CENTER POINT OF AN ARC
C..... OF WHICH VERTEX I IS AN END POINT
C*****
C
C          MODULE-1 (BEGIN)
C THIS MODULE RETRIEVES THE CADAM DRAWING ELEMENTS
C*****
C
C          START CADET PROCESSING - GET USER AND DRAWING NAME
          NUMLIN = 0
          NUMCIR=0
          LINELE=0
          NMHOLC=0
          IHOLIN=0
          LNPOHE=0
          NMHOLP=0
          CHAR=0
          CXARCG=0
          CYARCG=0
          CINERX=0
          CINERY=0
          CHAINA=0
          CHXRCG=0
          CHYRCG=0
          CHQPGX=0
          CHQPGY=0
          CHAPGX=0
          CHAPGY=0
          END = 'END'
C*****
C THE ROUTINE CDTST MARKS THE BEGINNING OF THE CADET
C PROCESSING
C*****
          ENTRY CDTST (EMPNO,PART,IPRINT,IEOF)
C EMPNO- USERID A6 RETURNED
C PART- DRAWING NAME A20 RETURNED

```

```

C      IPRINT-   OUTPUT UNIT NUMBER INTEGER RETURNED
C      IEOF  -   1 - NO MORE DRAWINGS OR 1 RETURNED
C              0 - DO MORE DRAWINGS
5      FORMAT (' ',A4,A2,2X,5A4)
      READ (5,5,END=10) EMPNO,PART
      IEOF = 0
      GO TO 15
10     IEOF = 1
15     CONTINUE
      RETURN

C
C      END OF CADET PROCESSING - PRINT NUMBER OF VIEWS
C      AND RECOPY OUTPUT
      ENTRY CDTEND (EMPNO,PART)
C          EMPNO -   USERID                A6 PASSED
C          PART  -   DRAWING NAME          A20 PASSED
      IPRINT = 6
C      WRITE (6,20) PART, FILDAT,NUMVU
C20    FORMAT(' DRAWING NAME = ',5A4,9X,2A4,
C        4'NUM OF VIEWS ',I2)
40     RETURN
C      FOUND A LINE IN THE DRAWING, SAVE IT IF PART OF
C      PERIMETER
      ENTRY CDTLN (C,D,LT)
C          C -   COORDINATES OF END ONE    3 REAL PASSED
C          D -   COODINATES OF END TWO    1X3 REAL PASSED
C          LT -   LINE TYPE                 INTEGER PASSED
C
      CALL ELNAME(NAME,IGROUP,NSTDTO,LEVEL)
      CALL ATTRIB(NAME,NXTNME,NUMATT,IATYPE,LEN,FULL,
* IHALF,BYTE,IWORD,DOUBLE,LEVEL)
      IF (NUMATT.NE.1) GO TO 50
      NUMLIN = NUMLIN+1
      LINELE=LINELE+1
C      ROUND DATA TO 3 PLACES
      CP1=C(1)*1000.0+.5
      CP2=C(2)*1000.0+.5
      DP1=D(1)*1000.0+.5
      DP2=D(2)*1000.0+.5
      PTLIN(LINELE,1) = AINT(CP1)/1000
      PTLIN(LINELE,2) = AINT(CP2)/1000
      PTLIN(LINELE,3) = AINT(DP1)/1000
      PTLIN(LINELE,4) = AINT(DP2)/1000
50     IF (NUMATT.NE.2) GO TO 53
      IHOLIN = IHOLIN+1
      LNPOHE=LNPOHE+1
C      ROUND DATA TO 3 PLACES
      CP1=C(1)*1000.0+.5
      CP2=C(2)*1000.0+.5
      DP1=D(1)*1000.0+.5
      DP2=D(2)*1000.0+.5

```

```

        PTLNH(LNPOHE,1) = AINT(CP1)/1000
        PTLNH(LNPOHE,2) = AINT(CP2)/1000
        PTLNH(LNPOHE,3) = AINT(DP1)/1000
        PTLNH(LNPOHE,4) = AINT(DP2)/1000
53      RETURN
C
C      FOUND AN ARC - SAVE IT IF PART OF PERIMETER
        ENTRY CDTARC(C,D,E,LT)
C      C = AN ARRAY CONTAINING THE X,Y, (Z) COORDINATES OF
C      THE ARC CENTER (FLOATING POINT)>
C
C      D = AN ARRAY CONTAINING THE X,Y,(Z) COORDINATES
C      OF END 1 OF THE ARC (FLOATING POINT).
C
C      NOTE: AN ARC IS ALWAYS DEVELOPED IN THE COUNTER
C      CLOCKWISE DIRECTION.
C      E = AN ARRAY CONTAINING THE X,Y,(Z), COORDINATES
C      OF END 2 OF THE ARC (FLOATING POINT)>
C
C      LT = AN INTEGER VALUE SPECIFYING THE LINE TYPE.
C      THE VALUES RANGE FROM 0 - 7.
        CALL ELNAME(NAME,IGROUP,NSTDTO,LEVEL)
        CALL ATTRIB(NAME,NXTNME,NUMATT,ITYPE,LEN,FULL,
        *IHALF,BYTE,IWORD,DOUBLE,LEVEL)
        IF (NUMATT.NE.1) GO TO 52
        NUMLIN = NUMLIN + 1
        NUMCIR=NUMCIR+1
C.....ROUND DATA TO 3 PLACES.
        CK1=C(1)*1000.0+.5
        CK2=C(2)*1000.0+.5
        EK1=E(1)*1000.0+.5
        EK2=E(2)*1000.0+.5
        DK1=D(1)*1000.0+.5
        DK2=D(2)*1000.0+.5
        CIRPTS(NUMCIR,1) = AINT(DK1)/1000
        CIRPTS(NUMCIR,2) = AINT(DK2)/1000.0
        CIRPTS(NUMCIR,3) = AINT(EK1)/1000.0
        CIRPTS(NUMCIR,4) = AINT(EK2)/1000.0
        RAWC(NUMCIR,1) = AINT(CK1)/1000.0
        RAWC(NUMCIR,2) = AINT(CK2)/1000.0
52      IF (NUMATT.NE.2) GO TO 48
        IHOLIN=IHOLIN+1
        NMHCIR=NMHCIR+1
C.....ROUND DATA TO 3 PLACES.
        CK1=C(1)*1000.0+.5
        CK2=C(2)*1000.0+.5
        EK1=E(1)*1000.0+.5
        EK2=E(2)*1000.0+.5
        DK1=D(1)*1000.0+.5
        DK2=D(2)*1000.0+.5
        HOLPTS(NMHCIR,1) = AINT(DK1)/1000

```

```

HOLPTS(NMHCIR,2) = AINT(DK2)/1000.0
HOLPTS(NMHCIR,3) = AINT(EK1)/1000.0
HOLPTS(NMHCIR,4) = AINT(EK2)/1000.0
HAWC(NMHCIR,1) = AINT(CK1)/1000.0
HAWC(NMHCIR,2) = AINT(CK2)/1000.0
48 RETURN
C
C
C FOUND A CIRCULAR HOLE INSIDE THE CONTOUR
ENTRY CDTTCIR(C,RAD,LT)
C C - CENTER POINT 1X3 REAL PASSED
C RAD -- RADIUS REAL PASSED
C LT - LINETYPE INTEGER PASSED
CALL ELNAME(NAME,IGROUP,NSTDTO,LEVEL)
CALL ATTRIB(NAME,NXTNME,NUMATT,ITYPE,LEN,FULL,
*IHAF, BYTE, IWORD, DOUBLE, LEVEL)
IF (NUMATT.NE.2) GO TO 60
NMHOLC=NMHOLC+1
CM1=C(1)
CM2=C(2)
HOLS(NMHOLC,1)=AINT(CM1)/1000
HOLS(NMHOLC,2)=AINT(CM2)/1000
HOLS(NMHOLC,3)=AINT(RAD)/1000
60 RETURN
C
C BEGIN NEW VIEW
ENTRY CDTBVU (IVU, ID, I, J, ARRAY, C, A, B, SCAL)
C IVU - VIEW NUMBER INTEGER PASSED
C ID - VIEW ID A2 PASSED
C IOTYPE - COORDINATE 1 - PAPER 1,2,3 RETURNED
C 1 - VIEW 3 -ABSOLUTE
C IR - 0 -PROCESS VIEW 0,1RETURNED
C ARRAY - ROTATION MATRIX 3*3 REAL PASSED
C C - TRANSLATION MATRIX 1*3 REAL PASSED
C A - AXIS TRANSLATION 1*2 REAL PASSED
C B - AXIS ROTATION 1*2 REAL PASSED
C SCAL - SCALE-LARGER = BIGGER REAL
I=IOTYPE
J=0
NUMVU = IVU+1
C WRITE (6,70) NUMVU
C70 FORMAT(' ',I2)
75 RETURN
C
C..... RECEIVES END VIEW INFORMATION AND DOES
C..... CALCULATIONS ON THAT VIEW
C
ENTRY CDTEVU(I)
C
C I - VIEW NUMBER INTEGER PASSED
C

```

```

C*****
C          MODULE-1 (END)
C*****
C          MODULE-2 (BEGIN)
C THIS MODULE CONCATENATES THE CADAM ELEMENTS
C OF THE OUTER CONTOUR TO FORM A CLOSED CONTOUR AND ALSO
C ALSO THE ELEMENTS OF EACH HOLE IF ANY ARE CONCATENATED.
C*****
C   DONE COLLECTING DATA, ORDER POINTS INTO A POLYLINE
      NUMPTS = NUMLIN+1
      IDONE=2
      IMATCH=0
      ACCY=1.0E-03
      POLYLN(1,1)=PTLIN(1,1)
      POLYLN(1,2)=PTLIN(1,2)
      POLYLN(2,1)=PTLIN(1,3)
      POLYLN(2,2)=PTLIN(1,4)
      POLARC(1,1)=0
      POLARC(NUMLIN+1,1)=0
      POLARC(2,2)=0
      DO 95 J=2,NUMLIN
      DO 85 I=2,LINELE
      IF (PTLIN(I,1) .NE. POLYLN(J,1)) THEN
        GO TO 80
      ENDIF
      IF (PTLIN(I,2) .NE. POLYLN(J,2)) THEN
        GO TO 80
      ENDIF
      IMATCH=1
      POLYLN(J+1,1)=PTLIN(I,3)
      POLYLN(J+1,2)=PTLIN(I,4)
      POLARC(J,1)=0
      POLARC(J+1,2)=0
      PTLIN(I,1)=9999
      PTLIN(I,2)=9999
      PTLIN(I,3)=9999
      PTLIN(I,4)=9999
      GO TO 95
80  IF (PTLIN(I,3) .NE. POLYLN(J,1)) THEN
      GO TO 85
      ENDIF
      IF (PTLIN(I,4) .NE. POLYLN(J,2)) THEN
      GO TO 85
      ENDIF
      IMATCH=1
      POLYLN(J+1,1)=PTLIN(I,1)
      POLYLN(J+1,2)=PTLIN(I,2)
      POLARC(J,1)=0
      POLARC(J+1,2)=0
      PTLIN(I,1)=9999

```

```

        PTLIN(I,2)=9999
        PTLIN(I,3)=9999
        PTLIN(I,4)=9999
        GO TO 95
85    CONTINUE
        IF(IMATCH.EQ.1) THEN
            GO TO 95
        ENDIF
90    DO 94 I=1,NUMCIR
        IF(CIRPTS(I,1).NE.POLYLN(J,1)) THEN
            GO TO 93
        ENDIF
        IF(CIRPTS(I,2).NE.POLYLN(J,2)) THEN
            GO TO 93
        ENDIF
        IMATCH=1
        FLPLN(J)=0
        POLYLN(J+1,1)=CIRPTS(I,3)
        POLYLN(J+1,2)=CIRPTS(I,4)
        POLYC(J,1)=RAWC(I,1)
        POLYC(J,2)=RAWC(I,2)
        POLARC(J,1)=1
        POLARC(J+1,2)=1
        CIRPTS(I,1)=9999
        CIRPTS(I,2)=9999
        CIRPTS(I,3)=9999
        CIRPTS(I,4)=9999
93    IF (CIRPTS(I,3).NE.POLYLN(J,1)) THEN
            GO TO 94
        ENDIF
        IF(CIRPTS(I,4).NE.POLYLN(J,2)) THEN
            GO TO 94
        ENDIF
        IMATCH=1
        FLPLN(J)=1
        POLYLN(J+1,1)=CIRPTS(I,1)
        POLYLN(J+1,2)=CIRPTS(I,2)
        POLYC(J,1)=RAWC(I,1)
        POLYC(J,2)=RAWC(I,2)
        POLARC(J,1)=1
        POLARC(J+1,2)=1
        CIRPTS(I,1)=9999
        CIRPTS(I,2)=9999
        CIRPTS(I,3)=9999
        CIRPTS(I,4)=9999
94    CONTINUE
        IF (IMATCH .EQ. 0) THEN
            GO TO 100
        ENDIF
95    IMATCH=0
        DO 21 I=1,NUMLIN

```

```

21      CONTINUE
      IF (POLYLN(1,1) .NE. POLYLN(NUMLIN+1,1)) THEN
        POLDI1=ABS(POLYLN(1,1)-POLYLN(NUMLIN+1,1))
      ENDIF
      IF (POLDI1 .GT. 0.009) THEN
        GO TO 100
      ENDIF
      IF (POLYLN(1,2) .NE. POLYLN(NUMLIN+1,2)) THEN
        POLDI2=ABS(POLYLN(1,2)-POLYLN(NUMLIN+1,2))
      ENDIF
      IF (POLDI2 .GT. 0.009) THEN
        GO TO 100
      ENDIF

```

C
C
C

```

      CONCATENATION OF POLYGONAL HOLES
      IF(IHOLIN.EQ.0) THEN
        GO TO 1098
      ENDIF
      ICHN=1
      HOLYLN(ICHN,1,1)=PTLNH(1,1)
      HOLYLN(ICHN,1,2)=PTLNH(1,2)
      HOLYLN(ICHN,2,1)=PTLNH(1,3)
      HOLYLN(ICHN,2,2)=PTLNH(1,4)
      HOLARC(ICHN,1,1)=0
      HOLARC(ICHN,2,2)=0

```

C

```

      IK=1
      IB(ICHN)=1
      J=2
1089  IF(((IHOLIN .GT. 0) .AND. (J.GT.IHOLIN))
      *.OR.(IHOLIN.EQ.0)) THEN
        GO TO 1098
      ENDIF
      DO 1085 I=2, LNPOHE
      IF (PTLNH(I,1) .NE. HOLYLN(ICHN,J,1)) THEN
        GO TO 1080
      ENDIF
      IF (PTLNH(I,2) .NE. HOLYLN(ICHN,J,2)) THEN
        GO TO 1080
      ENDIF
      MATCH=1
      HOLYLN(ICHN,J+1,1)=PTLNH(I,3)
      HOLYLN(ICHN,J+1,2)=PTLNH(I,4)
      HOLARC(ICHN,J,1)=0
      HOLARC(ICHN,J+1,2)=0
      PTLNH(I,1)=9999
      PTLNH(I,2)=9999
      PTLNH(I,3)=9999
      PTLNH(I,4)=9999
      GO TO 1090

```

```

1080   IF (PTLNH(I,3) .NE. HOLYLN(ICHN,J,1)) THEN
      GO TO 1085
    ENDIF
    IF (PTLNH(I,4) .NE. HOLYLN(ICHN,J,2)) THEN
      GO TO 1085
    ENDIF
    MATCH=1
    HOLYLN(ICHN,J+1,1)=PTLNH(I,1)
    HOLYLN(ICHN,J+1,2)=PTLNH(I,2)
    HOLARC(ICHN,J,1)=0
    HOLARC(ICHN,J+1,2)=0
      PTLNH(I,1)=9999
      PTLNH(I,2)=9999
      PTLNH(I,3)=9999
      PTLNH(I,4)=9999
      GO TO 1090
1085   CONTINUE
1090   IF(MATCH.EQ.1) THEN
      GO TO 1095
    ENDIF
      DO 1094 I=1,NMHCIR
    IF(HOLPTS(I,1).NE.HOLYLN(ICHN,J,1)) THEN
      GO TO 1093
    ENDIF
    IF(HOLPTS(I,2).NE.HOLYLN(ICHN,J,2)) THEN
      GO TO 1093
    ENDIF
    MATCH=1
    HLPLN(ICHN,J)=0
    HOLYLN(ICHN,J+1,1)=HOLPTS(I,3)
    HOLYLN(ICHN,J+1,2)=HOLPTS(I,4)
    HOLYC(ICHN,J,1)=HAWC(I,1)
    HOLYC(ICHN,J,2)=HAWC(I,2)
    HOLARC(ICHN,J,1)=1
    HOLARC(ICHN,J+1,2)=1
    HOLPTS(I,1)=9999
      HOLPTS(I,2)=9999
      HOLPTS(I,3)=9999
      HOLPTS(I,4)=9999
      GO TO 1095
1093   IF (HOLPTS(I,3).NE.HOLYLN(ICHN,J,1)) THEN
      GO TO 1094
    ENDIF
    IF(HOLPTS(I,4).NE.HOLYLN(ICHN,J,2)) THEN
      GO TO 1094
    ENDIF
    MATCH=1
    HLPLN(ICHN,J)=1
    HOLYLN(ICHN,J+1,1)=HOLPTS(I,1)
    HOLYLN(ICHN,J+1,2)=HOLPTS(I,2)
    HOLYC(ICHN,J,1)=HAWC(I,1)

```

```

HOLYC(ICHN,J,2)=HAWC(I,2)
HOLARC(ICHN,J,1)=1
HOLARC(ICHN,J+1,2)=1
HOLPTS(I,1)=9999
  HOLPTS(I,2)=9999
  HOLPTS(I,3)=9999
  HOLPTS(I,4)=9999
  GO TO 1095
1094   CONTINUE
1095   IF (MATCH.EQ.0) THEN
      GO TO 101
    ENDIF
    MATCH=0
    LK=J+1
    HOLDI1=ABS((HOLYLN(ICHN,IB(ICHN),1))
*-HOLYLN(ICHN,J+1,1))
    HOLDI2=ABS((HOLYLN(ICHN,IB(ICHN),2))
*-HOLYLN(ICHN,J+1,2))
    IF (HOLDI1.LT.0.009) THEN
      IF (HOLDI2.LT.0.009) THEN
        IE(ICHN)=J
        HOLARC(ICHN,J+1,2)=0
        IF (J.LT.IHOLIN) THEN
          DO 1123 I=2, LNPOHE
            IF (PTLNH(I,1) .LT.9999) THEN
              IF (PTLNH(I,2) .LT.9999) THEN
                IF (PTLNH(I,3) .LT.9999) THEN
                  IF (PTLNH(I,4) .LT.9999) THEN
                    ICHN=ICHN+1
                    IB(ICHN)=J+1
                    J=J+1
                    HOLYLN(ICHN,J,1)=PTLNH(I,1)
                    HOLYLN(ICHN,J,2)=PTLNH(I,2)
                    HOLYLN(ICHN,J+1,1)=PTLNH(I,3)
                    HOLYLN(ICHN,J+1,2)=PTLNH(I,4)
                    GO TO 1124
                  ENDIF
                ENDIF
              ENDIF
            ENDIF
          CONTINUE
        ENDIF
      ENDIF
    ENDIF
1123   CONTINUE
      ENDIF
    ENDIF
1124   J=LK
      GO TO 1089
1098   IF ((IHOLIN.EQ.0).AND. (NMHOLC.EQ.0)) THEN
      GO TO 1549
    ELSE
      NUMHOL= NMHOLC+ICHN
      WRITE(6,27) NUMHOL
    
```

```

27      FORMAT(I2)
      ENDIF
C*****
C          MODULE-2 (END)
C*****
C          MODULE-3 (BEGIN)
C*****
C
C  PROCEDURE TO DETERMINE THE PERIMETER OF THE CONTOUR
1549  PERIM1=0
      PERIM2=0
      DO 350 I=1,NUMLIN
      IF((POLARC(I,1).EQ.1).AND.(POLARC(I+1,2).EQ.1)) THEN
          GO TO 315
      ELSE
          TERM1=(POLYLN(I,1)-POLYLN(I+1,1))*2
          TERM2=TERM1+(POLYLN(I,2)-POLYLN(I+1,2))*2
          TERM=TERM1+TERM2
          PERIM1=PERIM1+SQRT(TERM)
          WRITE (6,*) ' PERIM1 IS',PERIM1
          GO TO 350
      ENDIF
315  IF(FLPLN(I).EQ.1) THEN
          POLY1=POLYLN(I+1,1)
          POLY2=POLYLN(I+1,2)
          POLY3=POLYLN(I,1)
          POLY4=POLYLN(I,2)
      ELSE
          POLY1=POLYLN(I,1)
          POLY2=POLYLN(I,2)
          POLY3=POLYLN(I+1,1)
          POLY4=POLYLN(I+1,2)
      ENDIF
      POLC1=POLYC(I,1)
      POLC2=POLYC(I,2)
      CALL PARMET(POLC1,POLC2,POLY1,POLY2,PARAM1,ACCY,RA)
      CALL PARMET(POLC1,POLC2,POLY3,POLY4,PARAM2,ACCY,RA)
      IF (PARAM2.GT.PARAM1) THEN
          PARMT=(PARAM2-PARAM1)/4.0
      ELSE
          IF (PARAM1.GT.PARAM2) THEN
              PARMT=(4.0-(PARAM1-PARAM2))/4.0
          ENDIF
      ENDIF
      PERIM2=PERIM2+PARMT*6.28*(RA)
350  CONTINUE
      PERIM=PERIM1+PERIM2
      WRITE(6,*) ' PERIMETER= ',PERIM
C*****
C          MODULE-3 (END)
C*****

```

```

C*****
C
C           MODULE-4 (BEGIN)
C
C THIS MODULE CALCULATES THE AREA OF THE OUTER CONTOUR
C*****
C STEPS TO FIND THE EQUATION OF THE NORMAL THAT
C POINTS TOWARDS THE AREA FOR ELEMENT-1.
C           CALL NORM1(POLYLN, PINMID, PTMID, NUMLIN, ACCY)
C
C.....DETERMINATION OF INWARD POINTING NORMALS FOR
C.....ELEMENTS FROM 2 TO N.
C
C           CALL NORM2(PTMID, POLYLN, PINMID, NUMLIN, ACCY)
C
C.....DETERMINATION OF THE AREA OF THE CONTOUR
C.....CONSIDERING ONLY THE POLYGONAL AREAS. ARC AREAS
C.....ARE TO BE DETERMINED SEPARATELY AND
C APPROPRIATELY ADJUSTED
C           CALL PLAREA(POLYLN, NUMLIN, AREA1, ARXCG1,
* ARYCG1, XCG, YCG)
C
C
C.....DETERMINATION OF NET AREA AFTER ADJUSTING FOR
C THE CIRCULAR PARTS
C           CALL CRAREA(POLYLN, FLPLN, NUMLIN, PTMID, PINMID,
* POLARC, ARXCG2, ARYCG2, AREARC, POLYC, ACCY, ARCAR)
C           AREA=AREA1
C           AREA=AREA1+AREARC
C           TOTAR=AREA
C           AREAXC=(ARXCG1+ARXCG2)/AREA
C           AREAYC=(ARYCG1+ARYCG2)/AREA
C*****
C           MODULE-4 (END)
C*****
C
C*****
C           MODULE-5 (BEGIN)
C*****
C PROCESSING OF THE POLYGONAL HOLES TO EXTRACT
C PARAMETERS
C           IF(IHOLIN.EQ.0) GO TO 4005
C           DO 4000 M=1, ICHN
C               ICB=IB(M)
C               ICE=IE(M)
C           MUMLIN=1
C           DO 3100 K=ICB, ICE
C               SOLYLN(K, 1)=HOLYLN(M, K, 1)
C               SOLYLN(K, 2)=HOLYLN(M, K, 2)
C               ISLPLN(K)=HLPLN(M, K)
C               IPOARC(K, 1)=HOLARC(M, K, 1)

```

```

        IPOARC(K,2)=HOLARC(M,K,2)
        SOLYC(K,1)=HOLYC(M,K,1)
        SOLYC(K,2)=HOLYC(M,K,2)
        MUMLIN=MUMLIN+1
3100  CONTINUE
        MUMLIN=MUMLIN-1
        CALL NORM1(SOLYLN,HINMID,HTMID,MUMLIN,ACCY)
        CALL NORM2(HTMID,SOLYLN,HINMID,MUMLIN,ACCY)
        CALL PLAREA(SOLYLN,MUMLIN,AREAH1,HRXCG1,HRYCG1,
        3HXCG,HYCG)
        CALL CRAREA(SOLYLN,ISLPLN,MUMLIN,HTMID,HINMID,
        *IPOARC,HRXCG2,HRYCG2,HREARC,SOLYC,ACCY,HRCAR)
        CHAINA=AREAH1+HREARC
        CHXRCG=CHXRCG+(HRXCG1+HRXCG2)
        CHYRCG=CHYRCG+(HRYCG1+HRYCG2)
4000  CONTINUE
C*****
C
C      EXTRACTION OF PARAMETERS OF CIRCULAR HOLES
C
C
4005  IF (NMHOLC.GT.0) THEN
        DO 29  J=1,NMHOLC
            CAR=3.14*((HOLS(J,3))**2)
            CHAR=CHAR+CAR
            CXARCG=CXARCG+CAR*HOLS(J,1)
            CYARCG=CYARCG+CAR*HOLS(J,2)
29      CONTINUE
        ENDIF
C
C*****
C      MODULE-5 (END)
C*****
C
C *****
C      MODULE-6 (BEGIN)
C *****
C      DETERMINATION OF THE CENTER OF AREA OF THE CONTOUR
        TOTAR=AREA-CHAR-CHAINA
        AREAXC=((AREAXC*AREA)-CXARCG-CHXRCG)/TOTAR
        AREAYC=((AREAYC*AREA)-CYARCG-CHYRCG)/TOTAR
C*****
C      MODULE-6 (END)
C*****
C
C *****
C      MODULE-7 (BEGIN)
C*****
C...DETERMINATION OF THE MAXIMUM RADIUS.

```

```

C
C
      CALL MAXIR(POLYLN,AREAXC,AREAYC,POLARC,FLPLN,PTMID,
*PINMID,ACCY,MAXRAD,POLYC,NUMLIN)
C
C      STEPS TO DETERMINE MINIMUM RADIUS
      CALL MINIR(POLYLN,AREAXC,AREAYC,POLARC,FLPLN,PTMID,
*PINMID,ACCY,MINRAD,POLYC,NUMLIN)
C
C*****
C                      MODULE-7 (END)
C*****
C
C
      GO TO 1700
101  WRITE (6,*) ' DEGENERATE HOLE'
C    1700 MARKS THE END OF THE PROGRAM
      GO TO 1700
100  WRITE(6,*) 'DEGENERATE PERIMETER'
      GO TO 1700
C 1700 MARKS THE EXIT OF THE PROGRAM
1700  RETURN
      ENTRY CDTDTO(C,D,IDET,SCAL,IFMIRR,IFXPND,LT)
      RETURN
      ENTRY CDTPT(C,LT)
      RETURN
      ENTRY CDTBGD(IVU,I)
      RETURN
      ENTRY CDTEDT(P,Q,R,T,I)
      RETURN
      ENTRY CDTEPS(C,D,E,AMAJ,AMIN,A,LT)
      RETURN
      ENTRY CDTDOT(C,R,LT)
      RETURN
      ENTRY CDTTRI(C,D,E,LT)
      RETURN
      ENTRY CDTREC(C,HH,HW,LT,DIR)
      RETURN
      ENTRY CDTRVT(C,D,DIST)
      RETURN
      ENTRY CDTSP(L,NOPTS,KIND,XYZ,ABC,S,FANGLE,FDERIV,
*OFFSET,JMIN,JMAX,LT,IPARNT)
      RETURN
      ENTRY CDTDIM(ISW,IRET,LT,IFSPEC)
      RETURN
      ENTRY CDTBRK(PT1,PT2,DIST,WGLHI,WGLWID,NBRKS)
      RETURN
      ENTRY CDTARO (PT1,PT2,NCHAR,TEXT,START,C,TXTHI,
4TXTWID)
      RETURN
      ENTRY CDTSC (PT1,PT2,C,D,E)

```

```

RETURN
ENTRY CDTNTE (PT1,NCHAR,TEXT,NTHT,TXTHI,TXTWID)
RETURN
ENTRY CDT CUT (PT1,DIR,TLRAD,CRNRAD)
RETURN
ENTRY CDTDLT(PT1,NCHAR,TEXT,PT2,TXTHI,TXTWID,SL,
4CIR,DLTRAD,E1,E2)
RETURN
ENTRY CDTBLN (PT1,NCHAR,TEXT,RAD,PT2,TXTHI,TXTWID)
RETURN
ENTRY CDTTXT (START,ANG,NCHAR,CHHIGH,CHWIDE,TEXT)
RETURN
ENTRY CDTTXL (PT1,START,PT2,NCHAR,ISVERT,TEXT,
4TXTHI,TXTWID,C,D)
RETURN
ENTRY CDTSUP (START,ANG,NCHAR,CHHIGH,CHWIDE,TEXT,
*FONT,SCALE,TURN,JUST,HORIZ,VERT,
2SLANT,SPAN,HSPACE,VSPACE,LINWT,IFSUPR)
RETURN
ENTRY CDTPLT (D,ANG,SCAL,DUPI,CHRHI,A,DHI,DWD,
*IOVPLT,IFMTNO,FU1,IFU2)
RETURN
ENTRY CDTGSN (SYMTB,ISYMNO,IR)
RETURN
ENTRY CDTESE (SYMTB,ISYMNO,IR)
RETURN
ENTRY CDTGFN (SYMTB,ISYMNO,IR,SPACTB)
RETURN
ENTRY CDTEFE (SYMTB,ISYMNO,IR)
RETURN
ENTRY CDTMPL (XYZ,NOPTS,OFFSET,LT,IPARNT,ITYP)
RETURN
ENTRY CDTBCB (XYZ,UVEC,WVEC,TWIST,BCBID,LT)
RETURN
ENTRY CDTREV (REVID,NOPTS,NPS,NC,NCALLS,XYZ,ABC,
*ABC2,HOMOG,NUMSPL,ARRAY,ANG,LT,NPT)
RETURN
ENTRY CDTRUL(NOPTS,NPS,NC,NCALLS,XYZ,ABC,ABC2,
*HOMOG,LT,N2PTS,NP2,NPT,N2T)
RETURN
ENTRY CDT3DC (C,D,E,P,Q,LT,IFCIR,PT1,PT2,AMAJ,AMIN)
RETURN
ENTRY CDTPLN (E1)
RETURN
ENTRY CDT3DP (PT1,J)
RETURN
ENTRY CDT3DD (PT1,RAD,LT)
RETURN
ENTRY CDT3DL (PT1,PT2,LT)
RETURN
ENTRY CDT3DA (PT1,PT2,LT)

```

```

RETURN
ENTRY CDTHBP (J,HBP)
RETURN
ENTRY CDTSYM (PT1,DIR,SCAL,MIRR,ISTN,ISN,IFFILL,
3 IEXPND)
RETURN
ENTRY CDT3DS (NOPTS,NPS,NC,NCALLS,XYZ,ABC,ABC2,
2 HOMOG,LT,NPT)
RETURN
END
SUBROUTINE NORM1(POLYLN,PINMID,PTMID,N,ACCY)
REAL POLYLN(20,2),PINMID(10,2),PTMID(10,2),PX(10)
REAL PY(10)
PTMID(1,1)=(POLYLN(1,1)+POLYLN(2,1))/2.0
PTMID(1,2)=(POLYLN(1,2)+POLYLN(2,2))/2.0
IF (ABS(POLYLN(2,2)-POLYLN(1,2)).LE.ACCY) THEN
    SLP MID=0.25
    DELX=0.0
ELSE IF (ABS(POLYLN(2,1)-POLYLN(1,1)).LE.ACCY) THEN
    SLP MID=0.0
    DELX=0.25
ELSE
    SLNR= (POLYLN(2,2)-POLYLN(1,2))/
4    (POLYLN(2,1)-POLYLN(1,1))
    DELX=0.25
    SLP MID=-0.25/SLNR
ENDIF
PINMID(1,1)=PTMID(1,1)+DELX
PINMID(1,2)=PTMID(1,2)+SLP MID
XMIN=POLYLN(1,1)
DO 400 I=2,N-1
IF (XMIN.GT.POLYLN(I+1,1)) THEN
    XMIN=POLYLN(I+1,1)
ENDIF
400 CONTINUE
INTPS=0
PINC1=PINMID(1,1)
PINC2=PINMID(1,2)
PINC3=XMIN-1
PINC4=PINMID(1,2)
DO 450 I=1,N
POLY1=POLYLN(I,1)
POLY2=POLYLN(I,2)
POLY3=POLYLN(I+1,1)
POLY4=POLYLN(I+1,2)
CALL INTLIN(POLY1,POLY2,POLY3,POLY4,PINC1,PINC2,
3PINC3,PINC4,TINT,XI,YI,ACCY)
PX(I)=XI
PY(I)=YI
IF(TINT.EQ.1.0) THEN
    IF((PX(I).NE.PX(I-1)).AND.(PY(I).NE.PY(I-1))) THEN

```

```

        INTPS=INTPS+1
        ENDIF
    ENDIF
450  CONTINUE
    IF ((MOD(INTPS,2).EQ.0).OR.(INTPS.EQ.0)) THEN
        PINMID(1,1)=PTMID(1,1)-DELX
        PINMID(1,2)=PTMID(1,2)-SLPMID
    ENDIF
    RETURN
    END

C
C
C  SUBROUTINE TO DETERMINE THE INWARD POINTING NORMALS
C  FOR ELEMENTS FROM 2 TO N.
    SUBROUTINE NORM2(PTMID,POLYLN,PINMID,N,ACCY)
    REAL POLYLN(20,2),PINMID(10,2),PTMID(10,2)
    DO 500 I=1,N-1
        PTMID(I+1,1)=(POLYLN(I+1,1)+POLYLN(I+2,1))/2.0
        PTMID(I+1,2)=(POLYLN(I+1,2)+POLYLN(I+2,2))/2.0
        POL11=POLYLN(I,1)
        POL12=POLYLN(I,2)
        POL21=POLYLN(I+2,1)
        POL22=POLYLN(I+2,2)
        POLC1=POLYLN(I+1,1)
        POLC2=POLYLN(I+1,2)
        CALL ACANG2(POL11,POL12,POL21,POL22,POLC1,POLC2,
3  THEETA)
        PLK=POLYLN(I+1,1)-POLYLN(I,1)
        PMK=POLYLN(I+2,1)-POLYLN(I,1)
        QLK=POLYLN(I+1,2)-POLYLN(I,2)
        QMK=POLYLN(I+2,2)-POLYLN(I,2)
        DET=0.5*((PLK*QMK)-(PMK*QLK))
        IF (DET.LT.0.0) THEN
            THEETA=180+THEETA
        ELSE
            THEETA=180-THEETA
        ENDIF
        THEETA=THEETA/57.3
        PINMID(I+1,1)=PINMID(I,1)*COS(THEETA)-PINMID(I,2)*
2SIN(THEETA)+PTMID(I,2)*SIN(THEETA)-
4PTMID(I,1)*COS(THEETA)+PTMID(I+1,1)
        WRITE(6,*) ' PINMID1 IS ',PINMID(I+1,1)
        PINMID(I+1,2)=PINMID(I,1)*SIN(THEETA)+PINMID(I,2)*
1COS(THEETA)-PTMID(I,1)*SIN(THEETA)-
4PTMID(I,2)*COS(THEETA)+PTMID(I+1,2)
        WRITE(6,*) ' PINMID2 IS ',PINMID(I+1,2)
500  CONTINUE
        RETURN
    END

C
C

```

C
C
C

```
SUBROUTINE TO DETERMINE THE POLYGONAL AREA.  
SUBROUTINE PLAREA(POLYLN,N,AREA1,ARXCG1,ARYCG1,  
4 XCG,YCG)  
REAL POLYLN(20,2)  
XCG=0.0  
YCG=0.0  
ARESUM=0.0  
XCOM=POLYLN(N,1)  
YCOM=POLYLN(N,2)  
XOLD=POLYLN(1,1)  
YOLD=POLYLN(1,2)  
DO 550 I=2,N-1  
X=POLYLN(I,1)  
Y=POLYLN(I,2)  
ARETRI=(XCOM-X)*(YOLD-YCOM)+(XOLD-XCOM)*(Y-YCOM)  
XCG=XCG+ARETRI*(X+XOLD)  
YCG=YCG+ARETRI*(Y+YOLD)  
WRITE(6,*) ' ARETRI IS ',ARETRI  
ARESUM=ARESUM+ARETRI  
XOLD=X  
YOLD=Y  
550 CONTINUE  
ARESUM=ABS(ARESUM)  
AREA1=ARESUM/2.0  
WRITE(6,*) ' AREA OF THE POLYGONAL CONTOUR IS = ',  
3 AREA1  
AREINV=1.0/ARESUM  
XCG=(XCG*AREINV+XCOM)*0.33333  
YCG=(YCG*AREINV+YCOM)*0.33333  
WRITE(6,*) ' XCG= ',XCG,' YCG= ',YCG  
ARXCG1=AREA1*XCG  
ARYCG1=AREA1*YCG  
RETURN  
END
```

C
C
C
C
C

```
SUBROUTINE CRAREA(POLYLN,FLPLN,N,PTMID,PINMID,  
*POLARC,ARXCG2,ARYCG2,AREARC,POLYC,ACCY,ARCAR)  
REAL POLYLN(20,2),PINMID(10,2),PTMID(10,2)  
REAL CENARX(20),CENARY(20),NEGP,ARCAR(20)  
REAL POYC(20,2)  
INTEGER FLPLN(20),POLARC(20,2)  
AREARC=0.0  
ARXCG2=0.0  
ARYCG2=0.0  
DO 600 I=1,N
```

```

IF (FLPLN(I).EQ.1) THEN
  POLY1=POLYLN(I+1,1)
  POLY2=POLYLN(I+1,2)
  POLY3=POLYLN(I,1)
  POLY4=POLYLN(I,2)
ELSE
  POLY1=POLYLN(I,1)
  POLY2=POLYLN(I,2)
  POLY3=POLYLN(I+1,1)
  POLY4=POLYLN(I+1,2)
ENDIF
IF ((POLARC(I,1).EQ.1).AND.(POLARC(I+1,2).EQ.1))
  THEN
    PTMID1=PTMID(I,1)
    PTMID2=PTMID(I,2)
    POLYC1=POLYC(I,1)
    POLYC2=POLYC(I,2)
    PINPT1=PINMID(I,1)
    PINPT2=PINMID(I,2)
    R1=(POLY1-POLYC1)**2
    R2=(POLY2-POLYC2)**2
    RAD1=SQRT(R1+R2)
    WRITE(6,*) ' RADIUS IS ',RAD1
    RADU=R1+R2
  CALL PARAMET(POLYC1,POLYC2,POLY1,POLY2,
3    PA1,ACCY,RA)
  WRITE(6,*) ' ARCPOINT1 PARAMETER IS ',PA1
  CALL PARAMET(POLYC1,POLYC2,POLY3,POLY4,PA2,
3    ACCY,RA)
  WRITE(6,*) ' ARCPOINT2 PARAMETER IS ',PA2
  CALL FINDT(PTMID1,PTMID2,PINPT1,PINPT2,POLYC1,
3    POLYC2,TC)
  WRITE(6,*) ' TC IS ',TC
  IF (ABS(TC).LE.0.02) THEN
    CALL INTCIR(PTMID1,PTMID2,PINPT1,PINPT2,
3    POLY1,POLY2,POLY3,POLY4,
4    POLYC1,POLYC2,TCIR1,TCIR2,XCIR1,YCIR1,
4    XCIR2,YCIR2,NOINT,TTAN,ACCY)
    CALL FINDT(PTMID1,PTMID2,PINPT1,PINPT2,XCIR1,
4    YCIR1,TC1)
    CALL FINDT(PTMID1,PTMID2,PINPT1,PINPT2,XCIR2,
3    YCIR2,TC2)
    IF (TC1.GT.TC2) THEN
      X1=XCIR1
      X2=XCIR2
      Y1=YCIR1
      Y2=YCIR2
    ELSE
      X1=XCIR2
      Y1=YCIR2
      X2=XCIR1

```

```

        Y2=YCIR1
    ENDIF
    CALL PARMET(POLYC1,POLYC2,X1,Y1,POSP,ACCY,RA)
    WRITE(6,*) ' POSITIVE PARAMETER IS ',POSP
    CALL PARMET(POLYC1,POLYC2,X2,Y2,NEGP,ACCY,RA)
    WRITE(6,*) ' NEGATIVE PARAMETER IS ',NEGP
    IF (PA1.GT.PA2) THEN
        IF((POSP.GT.PA2).AND.(POSP.LT.PA1)) THEN
            AREA2=0.5*3.14*(RADU)
            TVAL=(4*RADI/(9.42))/RADI
            XARCG=POLYC1+TVAL*(X2-POLYC1)
            YARCG=POLYC2+TVAL*(Y2-POLYC2)
        ELSE
            AREA2=-0.5*3.14*(RADU)
            TVAL=(4*RADI/(9.42))/RADI
            XARCG=POLYC1+TVAL*(X1-POLYC1)
            YARCG=POLYC2+TVAL*(Y1-POLYC2)
        ENDIF
    ELSE
        IF((NEGP.GT.PA1).AND.(NEGP.LT.PA2)) THEN
            AREA2=0.5*3.14*(RADU)
            TVAL=(4*RADI/(9.42))/RADI
            XARCG=POLYC1+TVAL*(X2-POLYC1)
            YARCG=POLYC2+TVAL*(Y2-POLYC2)
        ELSE
            AREA2=-0.5*3.14*(RADU)
            TVAL=(4*RADI/(9.42))/RADI
            XARCG=POLYC1+TVAL*(X1-POLYC1)
            YARCG=POLYC2+TVAL*(Y1-POLYC2)
        ENDIF
    ENDIF
ELSE
    IF(PA2.GT.PA1) THEN
        PA=PA2-PA1
    ELSE
        PA=4.0-(PA1-PA2)
    ENDIF
    PLK=POLYC1-POLYLN(I,1)
    PMK=POLYLN(I+1,1)-POLYLN(I,1)
    QLK=POLYC2-POLYLN(I,2)
    QMK=POLYLN(I+1,2)-POLYLN(I,2)
    DET=0.5*((PLK*QMK)-(PMK*QLK))
    IF(TC.LT.0.0) THEN
        IF (PA.LT.2.0) THEN
            AREA2=-((PA/4.0)*3.14*(RADU)-ABS(DET))
            ALPHA=((PA/8.0)*360)/57.3
            CALL ARCENG(PTMID1,PTMID2,POLYC1,POLYC2,ALPHA,
                RADI,W,XARCG,YARCG)
        ELSE
            AREA2=(PA/4.0)*3.14*(RADU)+ABS(DET)
            PDEL=4.0-PA
        ENDIF
    ENDIF

```

3

```

        ALPHA=(PDEL/8.0)*360/57.3
        CALL ARCENG(PTMID1,PTMID2,POLYC1,POLYC2,
3ALPHA,RADI,W,XARCG,YARCG)
        AR1=3.14*RADU
        AR2=(3.14*RADU)-ABS(AREA2)
        XARCG=((AR1*POLYC1)-(AR2*XARCG))/(AR1+AR2)
        YARCG=((AR1*POLYC2)-(AR2*YARCG))/(AR1+AR2)
    ENDIF
ELSE
    IF(PA.LT.2.0) THEN
        AREA2=(PA/4.0)*3.14*(RADU)-ABS(DET)
        ALPHA=(PA/8.0)*360/57.3
        CALL ARCENG(PTMID1,PTMID2,POLYC1,POLYC2,
3ALPHA,RADI,W,XARCG,YARCG)
    ELSE
        AREA2=-((PA/4.0)*3.14*(RADU)+ABS(DET))
        PDEL=4.0-PA
        ALPHA=(PDEL/8.0)*360/57.3
        CALL ARCENG(PTMID1,PTMID2,POLYC1,POLYC2,
3ALPHA,RADI,W,XARCG,YARCG)
        AR1=3.14*RADU
        AR2=(3.14*RADU)-ABS(AREA2)
        XARCG=((AR1*POLYC1)-(AR2*XARCG))/(AR1+AR2)
        YARCG=((AR1*POLYC2)-(AR2*YARCG))/(AR1+AR2)
    ENDIF
ENDIF
ENDIF
ARCAR(I)=ABS(AREA2)
CENARX(I)=XARCG
CENARY(I)=YARCG
AREARC=AREARC+AREA2
WRITE(6,*) ' AREA OF AREARC IS ',AREARC
ARXCG2=ARXCG2+AREA2*XARCG
ARYCG2=ARYCG2+AREA2*YARCG
ENDIF
600 CONTINUE
    RETURN
    END
C
C
C
C
C
C
C
C
SUBROUTINE TO DETERMINE THE MAXIMUM RADIUS
SUBROUTINE MAXIR(POLYLN,AREAXC,AREAYC,POLARC,
*FLPLN,PTMID,PINMID,ACCY,MAXRAD,POLYC,N)
    REAL POLYLN(20,2),PINMID(10,2),PTMID(10,2),
    REAL NEGPA,MR(20),MAXRAD,POLYC(20,2)
    INTEGER POLARC(20,2),FLPLN(20)
    J=1

```

```

DO 700 I=1,N
  DIS1=(POLYLN(I,1)-AREAXC)**2
  DIS2=(POLYLN(I,2)-AREAYC)**2
  MR(J)=SQRT(DIS1+DIS2)
  IF (POLARC(I,1).EQ.1.AND.POLARC(I+1,2).EQ.1)
3    THEN
      J=J+1
      IF (FLPLN(I).EQ.1) THEN
          POLY1=POLYLN(I+1,1)
          POLY2=POLYLN(I+1,2)
          POLY3=POLYLN(I,1)
          POLY4=POLYLN(I,2)
      ELSE
          POLY1=POLYLN(I,1)
          POLY2=POLYLN(I,2)
          POLY3=POLYLN(I+1,1)
          POLY4=POLYLN(I+1,2)
      ENDIF
      PTMID1=PTMID(I,1)
      PTMID2=PTMID(I,2)
      POLYC1=POLYC(I,1)
      POLYC2=POLYC(I,2)
      PINPT1=PINMID(I,1)
      PINPT2=PINMID(I,2)
      R1=(POLY1-POLYC1)**2
      R2=(POLY2-POLYC2)**2
      RAD1=SQRT(R1+R2)
      RADU=R1+R2
      CALL PARMET(POLYC1,POLYC2,POLY1,POLY2,
3        PA1,ACCY,RA)
      CALL PARMET(POLYC1,POLYC2,POLY3,POLY4,
3        PA2,ACCY,RA)
      CALL INTCIR(PTMID1,PTMID2,PINPT1,PINPT2,
4        POLY1,POLY2,POLY3,POLY4,
4Y4,POLYC1,POLYC2,TCIR1,TCIR2,XCIR1,YCIR1,XCIR2,
5YCIR2,NOINT,TTAN,ACCY)
      CALL FINDT(PTMID1,PTMID2,PINPT1,PINPT2,XCIR1,
3        YCIR1,TC1)
      CALL FINDT(PTMID1,PTMID2,PINPT1,PINPT2,XCIR2,
3        YCIR2,TC2)
      IF (TC1.GT.TC2) THEN
          X1=XCIR1
          X2=XCIR2
          Y1=YCIR1
          Y2=YCIR2
      ELSE
          X1=XCIR2
          Y1=YCIR2
          X2=XCIR1
          Y2=YCIR1
      ENDIF
  ENDIF

```

```

CALL PARMET(POLYC1,POLYC2,X1,Y1,POSP,ACCY,RA)
CALL PARMET(POLYC1,POLYC2,X2,Y2,NEGP,ACCY,RA)
IF (PA1.GT.PA2) THEN
  IF((POSP.GT.PA2).AND.(POSP.LT.PA1)) THEN
    DIS1=(X2-AREAXC)**2
    DIS2=(Y2-AREAYC)**2
    MR(J)=SQRT(DIS1+DIS2)
  ELSE
    DIS1=(X1-AREAXC)**2
    DIS2=(Y1-AREAYC)**2
    MR(J)=SQRT(DIS1+DIS2)
  ENDIF
ELSE
  IF((NEGP.GT.PA1).AND.(NEGP.LT.PA2)) THEN
    DIS1=(X2-AREAXC)**2
    DIS2=(Y2-AREAYC)**2
    MR(J)=SQRT(DIS1+DIS2)
  ELSE
    DIS1=(X1-AREAXC)**2
    DIS2=(Y1-AREAYC)**2
    MR(J)=SQRT(DIS1+DIS2)
  ENDIF
ENDIF
ENDIF
ENDIF
J=J+1
700 CONTINUE
NJ=J
MAXRAD=MR(1)
DO 750 J=1,NJ-1
  IF(MAXRAD.LE.MR(J+1)) THEN
    MAXRAD=MR(J+1)
  ENDIF
750 CONTINUE
WRITE(6,*) MAXRAD
RETURN
END

```

C
C
C
C

```

SUBROUTINE TO DETERMINE MINIMUM RADIUS
SUBROUTINE MINIR(POLYLN,AREAXC,AREAYC,POLARC,
*FLPLN,PTMID,PINMID,ACCY,MINRAD,POLYC,N)
  REAL POLYLN(20,2),PTMID(10,2),PINMID(10,2),
  REAL MINRAD,MR(20),NEGP,POLYC(20,2)
  INTEGER POLARC(20,2),FLPLN(20)
  J=0
  DO 800 I=1,N
    J=J+1
    XJO=AREAXC-POLYLN(I,1)
    YJO=AREAYC-POLYLN(I,2)
    DS1=XJO**2

```

```

DS2=YJO**2
MR(J)=SQRT(DS1+DS2)
WRITE(6,*) ' MRJ IS',MR(J)
F=POLYLN(I+1,1)-POLYLN(I,1)
G=POLYLN(I+1,2)-POLYLN(I,2)
NR=F*XJO+G*YJO
DR=F*F+G*G
TVALUE=NR/DR
IF(TVALUE.GE.0.0.AND.TVALUE.LE.1.0) THEN
    J=J+1
    FYGX=F*YJO-G*XJO
    V1=G*FYGX
    V2=-F*FYGX
    IF(POLARC(I,1).EQ.1.AND.POLARC(I+1,2).EQ.1) THEN
        IF(FLPLN(I).EQ.1) THEN
            POLY1=POLYLN(I+1,1)
            POLY2=POLYLN(I+1,2)
            POLY3=POLYLN(I,1)
            POLY4=POLYLN(I,2)
        ELSE
            POLY1=POLYLN(I,1)
            POLY2=POLYLN(I,2)
            POLY3=POLYLN(I+1,1)
            POLY4=POLYLN(I+1,2)
        ENDIF
        PTMID1=PTMID(I,1)
        PTMID2=PTMID(I,2)
        POLYC1=POLYC(I,1)
        POLYC2=POLYC(I,2)
        PINPT1=PINMID(I,1)
        PINPT2=PINMID(I,2)
        R1=(POLY1-POLYC1)**2
        R2=(POLY2-POLYC2)**2
        RAD1=SQRT(R1+R2)
        RADU=R1+R2
        CALL PARMET(POLYC1,POLYC2,POLY1,POLY2,
3          PA1,ACCY,RA)
        CALL PARMET(POLYC1,POLYC2,POLY3,POLY4,
3          PA2,ACCY,RA)
        CALL PARMET(POLYC1,POLYC2,AREAXC,AREAYC,
4          PARCG,ACCY,RA)
        IF(PA2.LT.PA1) THEN
            IF(PARCG.GT.PA1.OR.PARCG.LT.PA2) THEN
                DIS1=(AREAXC-POLYC1)**2
                DIS2=(AREAYC-POLYC2)**2
                MR(J)=SQRT(DIS1+DIS2)-RAD1
            ELSE
                DIS1=(AREAXC-POLY1)**2
                DIS2=(AREAYC-POLY2)**2
                DIST1=SQRT(DIS1+DIS2)
                DIS1=(AREAXC-POLY3)**2

```

```

        DIS2=(AREAYC-POLY4)**2
        DIST2=SQRT(DIS1+DIS2)
        MR(J)=MIN(DIST1,DIST2)
    ENDIF
ELSE
    IF(PARCG.GT.PA1.AND.PARCG.LT.PA2) THEN
        DIS1=(AREAXC-POLYC1)**2
        DIS2=(AREAYC-POLYC2)**2
        MR(J)=SQRT(DIS1+DIS2)-RADI
    ELSE
        DIS1=(AREAXC-POLY1)**2
        DIS2=(AREAYC-POLY2)**2
        DIST1=SQRT(DIS1+DIS2)
        DIS1=(AREAXC-POLY3)**2
        DIS2=(AREAYC-POLY4)**2
        DIST2=SQRT(DIS1+DIS2)
        MR(J)=MIN(DIST1,DIST2)
    ENDIF
ENDIF
ELSE
    MR(J)=SQRT(V1*V1+V2*V2)/DR
ENDIF
ENDIF
CONTINUE
NJ=J
MINRAD=MR(1)
DO 850 J=1,NJ-1
    IF(MR(J+1).LE.MINRAD) THEN
        MINRAD=MR(J+1)
    ENDIF
CONTINUE
WRITE(6,*) MINRAD
RETURN
END

```

C
C
C

```

SUBROUTINE  PARMET(POC1,POC2,PO1,PO2,PAR,ACCY,RA)
    XKJ=PO1-POC1
    YKJ=PO2-POC2
    XK=XKJ*XKJ
    YK=YKJ*YKJ
    RA=SQRT(XK+YK)
    IF(XKJ.GE.O.O.AND.YKJ.GE.O.O) THEN
        IF (XKJ.GT.ACCY) THEN
            TNT=YKJ/XKJ
            PAR=(SQRT(1.0+TNT*TNT)-1.0)/TNT
        ELSE
            PAR=0.0
        ENDIF
    ELSE IF (XKJ.LE.O.O.AND.YKJ.GE.O.O) THEN

```

```

        IF (YKJ.GT.ACCY) THEN
            TNT=-XKJ/YKJ
            PAR=1.0+(SQRT(1.0+TNT*TNT)-1.0)/TNT
        ELSE
            PAR=1.0
        ENDIF
    ELSE IF (XKJ.LE.0.0.AND.YKJ.LE.0.0) THEN
        IF(XKJ.LE.-ACCY) THEN
            TNT=YKJ/XKJ
            PAR=2.0+(SQRT(1.0+TNT*TNT)-1.0)/TNT
        ELSE
            PAR=2.0
        ENDIF
    ELSE
        IF (YKJ.LT.-ACCY) THEN
            TNT=-XKJ/YKJ
            PAR=3.0+(SQRT(1.0+TNT*TNT)-1.0)/TNT
        ELSE
            PAR=3.0
        ENDIF
    ENDIF
    RETURN
END

```

```

C.....SUBROUTINE TO DETERMINE THE INTERSECTION
C.....POINT OF THE LINE SEGMENT FORMED BY THE
C.....A POINT AT UNIT DISTANCE FROM THE MID POINT
C.....OF A LINE SEGMENT AND THE POINT WITH X-VALUE
C...EQUAL TO XMIN-1, WITH ANY LINE SEGMENT IN THE CONTOUR
SUBROUTINE INTLIN(POLY1,POLY2,POLY3,POLY4,PINC1,
3PINC2,PINC3,PINC4,TINT,XI,YI,ACCY)
    XLK=PINC3-PINC1
    YLK=PINC4-PINC2
    XNM=POLY3-POLY1
    YNM=POLY4-POLY2
    XMK=POLY1-PINC1
    YMK=POLY2-PINC2
    DET=XNM*YLK-YNM*XLK
    IF (ABS(DET).GT.ACCY) THEN
        DETINV=1.0/DET
        S=(XNM*YMK-YNM*XMK)*DETIINV
        T=(XLK*YMK-YLK*XMK)*DETIINV
        IF(S.LT.0.0.OR.S.GT.1.0.OR.T.LT.0.0.OR.T.GT.1.0)
            THEN
                TINT=0.0
            ELSE
                TINT=1.0
                XI=PINC1+XLK*S
                YI=PINC2+YLK*S
            ENDIF
        ENDIF
    RETURN

```

```

        END
    SUBROUTINE ACANG2(POL11,POL12,POL21,POL22,POLC1,
4POLC2,THEETA)
    D11=(POL11-POLC1)**2
    D12=(POL12-POLC2)**2
    D1=D11+D12
    D1=SQRT(D1)
    D21=(POL21-POLC1)**2
    D22=(POL22-POLC2)**2
    D2=D21+D22
    D2=SQRT(D2)
    XA=POL11-POLC1
    XB=POL21-POLC1
    YA=POL12-POLC2
    YB=POL22-POLC2
    CSTETA=((XA*XB)+(YA*YB))/(D1*D2)
    IF((CSTETA.GE.(-0.02)) .AND.(CSTETA.LE.(0.02)))
3    THEN
        THEETA=89.0
    ELSE IF (CSTETA.LE.-0.9939) THEN
        THEETA=178.0
    ELSE
        THEETA=57.3*ACOS(CSTETA)
    ENDIF
    RETURN
    END

```

C.....
C.....

```

    SUBROUTINE INTCIR(PINC1,PINC2,PINC3,PINC4,POLY1,
3POLY2,POLY3,POLY4,POLYC1,POLYC2,TI1,TI2,XI1,
3YI1,XI2,YI2,NOINT,TTAN,ACCY)
    F=PINC3-PINC1
    G=PINC4-PINC2
    FSQ=F*F
    GSQ=G*G
    FGSQ=FSQ+GSQ
    SM1=(POLYC1-POLY1)**2
    SM2=(POLYC2-POLY2)**2
    SM=SM1+SM2
    RJ=SQRT(SM)
    XJO=POLYC1-PINC1
    YJO=POLYC2-PINC2
    FYGX=F*YJO-G*XJO
    ROOT=RJ*RJ*FGSQ-FYGX*FYGX
    IF(ROOT.LT.-ACCY) THEN
        NOINT=1
    ELSE
        FXGY=F*XJO+G*YJO
        IF (ROOT.LT.ACCY) THEN
            TTAN=1
            TI1=FXGY/FGSQ

```

```

        XI1=PINC1+F*TI1
        YI1=PINC2+F*TI1
    ELSE
        ROOT=SQRT(ROOT)
        FGINV=1.0/FGSQ
        TI1=(FXGY-ROOT)*FGINV
        TI2=(FXGY+ROOT)*FGINV
        XI1=PINC1+F*TI1
        YI1=PINC2+G*TI1
        XI2=PINC1+F*TI2
        YI2=PINC2+G*TI2
    ENDIF
ENDIF
RETURN
END
C.....SUBROUTINE TO DETERMINE THE COORDINATES OF THE
C.....CENTER OF AREA OF THE CHORD.
    SUBROUTINE ARCENG(P1,P2,POLYC1,POLYC2,ALPHA,
2   RADI,W,XARCG,YARCG)
    RN=1.333*RADI*((SIN(ALPHA))**3)
    TOALPA=2.0*ALPHA
    DR=TOALPA-SIN(TOALPA)
    W=RN/DR
    D1X=(P1-POLYC1)**2
    D1Y=(P2-POLYC2)**2
    D1XY=SQRT(D1X+D1Y)
    TVAL=W/D1XY
    XARCG=POLYC1+TVAL*(P1-POLYC1)
    YARCG=POLYC2+TVAL*(P2-POLYC2)
    RETURN
    END
C.....SUBROUTINE TO TRANSFORM THE AREA MOMENT OF
C.....INERTIA OF INCLINED CIRCULAR SEGMENTS
    SUBROUTINE ERTIA(POLY1,POLY2,POLY3,POLY4,
3   CIU,CIV,AIX,AIY,ACCY)
    IF(ABS(POLY1-POLY3).LE.ACCY) THEN
        AIX=CIU
        AIY=CIV
    ELSE IF(ABS(POLY2-POLY4).LE.ACCY) THEN
        AIX=CIV
        AIY=CIU
    ELSE
        SLNR=(POLY4-POLY2)/(POLY3-POLY1)
        TTA=ATAN(SLNR)*57.3
        WRITE(6,*) ' TTA IS ',TTA
        TTA=(180-TTA)/57.3
        AIX=AIX*(COS(TTA)*COS(TTA))+AIY*
3        (SIN(TTA)*SIN(TTA))
        AIY=AIX
    ENDIF
    RETURN

```

```

                END
C   SUBROUTINE TO DETERMINE THE SLOPE
C
        SUBROUTINE SLPLN(POLY1,POLY2,POLY3,POLY4,TNTA)
                IF((ABS(POLY1-POLY3).LE.ACCY)) THEN
                        TNTA=90.0
                ELSE IF((ABS(POLY2-POLY4).LE.ACCY)) THEN
                        TNTA=0.0
                ELSE
                        TATA=(POLY4-POLY2)/(POLY3-POLY1)
                        TNTA=ATAN(TATA)
                ENDIF
                RETURN
        END
C...SUBROUTINE TO FIND THE PARAMETER VALUE OF THE CENTER
C...POINT ON THE LINE JOINED BY THE MID-POINT OF THE
C...AND THE POINT AT A DISTANCE OF ONE UNIT
C...FROM THE MIDPOINT AND PERPENDICULAR TO THE POLYLINE
        SUBROUTINE FINDT(PMDPT1,PMDPT2,PNCPT1,PNCPT2,
4 POLYC1,POLYC2,TC)
        TC=-((PMDPT1-POLYC1)*(PNCPT1-PMDPT1)+(PMDPT2-
3 POLYC2)*(PNCPT2-PMD
3PT2))/((PNCPT1-PMDPT1)**2+(PNCPT2-PMDPT2)**2)
        RETURN
        END

```

APPENDIX B. VISION SYSTEM PROGRAM

```
C      THIS PROGRAMS PROCESSES THE GIVEN BINARY IMAGE
C      TO EXTRACT SHAPE PARAMETERS
C      DESCRIPTION OF VARIABLES
C      B(I,1) = LEVEL0; B(I,2) = LEVEL2; B(I,3) = LEVEL3;
C      B(I,4) = COLOR;
C      FMOY = FIRST MOMENT ABOUT Y AXIS; SMOX = SECOND
C      MOMENT ABOUT X AXIS;
C      SMOY = SECOND MOMENT ABOUT Y AXIS;
C      B(I,6) = PERIMETER; B(I,9) = REGION NUMBER;
C      B(I,10) = NUMBER OF HOLES
C      INITIALIZATION OF VARIABLES
C      INTEGER BN,VACUM, BOOK, B(20,10), R(1000,4),BP(25)
C      INTEGER*2 IER,IARY(512),CROW(512),PROW(512),
C      RP(10),CBLOB,RBLOB,HOLREG,INNP(10,10),ND
C      REAL FMOX(10),FMOY(10),SMOX(10),SMOY(10),
C      AREA(10),PERI(10)
C      INITIALIZATION OF VARIABLES
C
C      INTEGER EL,PJOG,CJOG,STATUS,TKK,TKK1
C      OPEN(5,FILE='CADIN1',STATUS='UNKNOWN')
C      OPEN(7,FILE='CADIN2',STATUS='UNKNOWN')
C      OPEN(8,FILE='CADIN3',STATUS='UNKNOWN')
C      READ(5,8) AR1,PER1,RMX1,RMI1
C      READ(7,8) AR2,PER2,RMX2,RMI2
C      READ(8,8) AR3,PER3,RMX3,RMI3
8      FORMAT(F6.2,/,F6.2,/,F6.2,/,F6.2)
C
C      THE FOLLOWING PROCEDURE INITIALIZES THE PCVISION
C      SYSTEM AND THEN GRABS AND FREEZES THE IMAGE IN
C      THE FIELD OF VIEW
C      CALL VINIT(IER)
C      CALL GRAB(IER)
C      CALL FREEZE(IER)
C      CALL RAMP(IER)
11     WRITE(*,*) 'ENTER THE THRESHOLD VALUE'
C      READ(*,*) ITH
C      IF(ITH.LT.0).OR.(ITH.GT.255) THEN
C          GOTO 11
C      ELSE
C          CALL THOLD(ITH,IER)
C          WRITE(*,*) 'IF THRESHOLD SATISFACTORY ENTER 1'
C          READ(*,*) ISAT
C          IF(ISAT.EQ.1) THEN
C              GOTO 12
C          ELSE
C              GOTO 11
C          ENDIF
C      ENDIF
C      ENDIF
```

```

C THE FOLLOWING PROCEDURE PROCESSES THE THRESHOLDED
C IMAGE
12 DO 18 IA=0,479
    CALL VRDROW(IA,IARY,IER)
    DO 17 IC=1,512
        IF(IARY(IC).GT.ITH) GO TO 16
        IARY(IC)=0
        GOTO 17
16 IARY(IC)=255
17 CONTINUE
    CALL VWRROW(IA,IARY,IER)
18 CONTINUE
C
    VACUM=2
    BN = 1
    BOOK = -1
    DO 20 I=1,512
20 PROW(I) = 1
    N=1
    CROW(512)=1
    CALL REGINI(N,IR,BN,BP,B,CROW,FMOX,FMOY,SMOX,SMOY,
4AREA,PERI)
    B(1,7)=1
    B(1,8) = 1
    RP(JL) =1
    R(1,1)=1
    R(1,2)=512
    R(1,3)=1
    R(1,4)=1
    PJOG=1
    R(2,1)=1
    CJOG=2
    DO 200 J=1,479
        NEWFLG=0
        STATUS=15
        RP(J)=CJOG
C GET THE LINE CORRESPONDING TO J
    CALL VRDROW(J,IARY,IER)
    DO 40 K=1,512
        IF(IARY(K).EQ.255) THEN
            IARY(K)=1
        ENDIF
40 CROW(K)=INNP(J,K)
    ND = R(PJOG,3)
    CALL LOOKUP(ND,BP,LKK)
    RBLOB=LKK
    CBLOB=RBLOB
    DO 100 I=1,512
        STATUS= STATUS/4.0+4*PROW(I)+8*CROW(I)
        PROW(I)=CROW(I)
        IF((STATUS.EQ.7) .OR. (STATUS.EQ.8)) THEN DO

```

```

        EL=0
        CALL UPDATE(B,R,CJOG,CBLOB,I,J,PJOG,EL,IL,FMOX,
3 FMOY,SMOX,SMOY,AREA,PERI)
        LFTEND=I
        NEWFLG=1
        GO TO 100
    ENDIF
    IF ((STATUS.EQ.4) .OR. (STATUS.EQ.11)) THEN DO
        LFTEND=I
        PJOG=PJOG+1
        ND=R(PJOG,3)
        CALL LOOKUP(ND,BP,LKK)
        RBLOB=LKK
        GO TO 100
    ENDIF
    IF ((STATUS.EQ.3) .OR. (STATUS.EQ.12)) THEN DO
        EL=0
        CALL UPDATE(B,R,CJOG,CBLOB,I,J,PJOG,EL,IL,FMOX,
4 FMOY,SMOX,SMOY,AREA,PERI)
        PJOG=PJOG+1
        ND=R(PJOG,3)
        CALL LOOKUP(ND,BP,LKK)
        RBLOB=LKK
        CBLOB=RBLOB
        GO TO 100
    ENDIF
    IF((STATUS.EQ.2) .OR. (STATUS.EQ.13)) THEN DO
        IF (NEWFLG.NE.0) THEN DO
            CALL ALLOC(NEWFLG,BOOK,VACUM,CBLOB,B,BN,BP,CROW,
*I,J,R,RBLOB,FMOX,FMOY,SMOX,SMOY,AREA,PERI)
        ENDIF
        EL=0
        CALL UPDATE(B,R,CJOG,CBLOB,I,J,PJOG,EL,IL,FMOX,
3 FMOY,SMOX,SMOY,AREA,PERI)
        PERI(CBLOB)=PERI(CBLOB)+(I-LFTEND)*0.02381
        PERI(RBLOB)=PERI(RBLOB)+(I-LFTEND)*0.02381
        CBLOB=RBLOB
        GO TO 100
    ENDIF
    IF((STATUS.EQ.6) .OR. (STATUS.EQ.9)) THEN DO
        IF (NEWFLG.NE.0) THEN DO
            CALL ALLOC(NEWFLG,BOOK,VACUM,CBLOB,B,BN,BP,
*CROW,I,J,R,RBLOB,FMOX,FMOY,SMOX,SMOY,AREA,PERI)
        ENDIF
        EL=0
        CALL UPDATE(B,R,CJOG,CBLOB,I,J,PJOG,EL,IL,FMOX,
3 FMOY,SMOX,SMOY,AREA,PERI)
        PERI(CBLOB)=PERI(CBLOB)+(I-LFTEND)*0.02381
        PERI(RBLOB)=PERI(RBLOB)+(I-LFTEND)*0.02381
        CBLOB=RBLOB
        PJOG=PJOG+1

```

```

ND=R(PJOG,3)
CALL LOOKUP(ND,BP,LKK)
RBLOB=LKK
LFTEND=I
GO TO 100
ENDIF
IF((STATUS.EQ.1) .OR. (STATUS.EQ.14)) THEN
HOLREG=RBLOB
PJOG=PJOG+1
ND=R(PJOG,3)
CALL LOOKUP(ND,BP,LKK)
RBLOB=LKK
IF(NEWFLG.NE.0) THEN
NEWFLG=0
CBLOB=RBLOB
PERI(CBLOB)=PERI(CBLOB)+(I-LFTEND)*0.02381
ELSE
IF(CBLOB.EQ.RBLOB) THEN
TKK=B(CBLOB,2)
B(HOLREG,3)=TKK
B(HOLREG,1)=CBLOB
B(CBLOB,2)=HOLREG
B(CBLOB,10)=B(CBLOB,10)+1
PERI(CBLOB)=PERI(CBLOB)-B(HOLREG,6)
ELSE
PERI(CBLOB)=PERI(CBLOB)+PERI(RBLOB)+
(I-LFTEND)*0.02381
AREA(CBLOB)=AREA(CBLOB)+AREA(RBLOB)
FMOX(CBLOB)=FMOX(CBLOB)+FMOX(RBLOB)
FMOY(CBLOB)=FMOY(CBLOB)+FMOY(RBLOB)
SMOX(CBLOB)=SMOX(CBLOB)+SMOX(RBLOB)
SMOY(CBLOB)=SMOY(CBLOB)+SMOY(RBLOB)
IF(B(CBLOB,10).NE.0) THEN DO
B(CBLOB,10)=B(CBLOB,10)+B(RBLOB,10)
TKK=B(RBLOB,2)
IF(TKK.GT.0) THEN DO
TKK1=TKK
TKK=B(TKK1,3)
ENDIF
B(TKK1,3)=B(CBLOB,2)
B(CBLOB,2)=B(RBLOB,2)
ENDIF
B(B(RBLOB,8),7)=B(RBLOB,7)
B(B(RBLOB,7),8)=B(RBLOB,8)
BP(B(RBLOB,9))=-B(CBLOB,9)
B(RBLOB,7)=BOOK
BOOK=RBLOB
RBLOB=CBLOB
ENDIF
ENDIF
B(HOLREG,6)=B(HOLREG,6)+I-LFTEND
ENDIF

```

```

100     CONTINUE
        EL=1
        CALL UPDATE(B,R,CJOG,CBLOB,I,J,PJOG,EL,IL,
4       FMOX,FMOY,SMOX,SMOY,AREA,PERI)
200     CONTINUE
C
C     THIS PROCEDURE DETERMINES THE MAXIMUM RADIUS
C     OF THE BLACK BLOB WITHRESPECT TO ITS OUTER
C     CONTOUR, THAT IS IGNORING THE PRESENCE OF HOLES
        XBAR=FMOX(2)/AREA(2)
        YBAR=FMOY(2)/AREA(2)
        DMAX=0.0
        DMIN=99999.0
        M1=1
        M2=479
        LCR=1
        DO 700 JJ=M1,M2
            K=RP(JJ)
            NK=RP(JJ+1)
            MK=NK-1
            DO 611 N=K,MK
                I1=R(N,1)
                I2=R(N,1)+R(N,2)-1
                IF((XBAR.GE.I1).AND.(XBAR.LE.I2)) THEN
                    IF(R(N,4).NE.LCR) THEN
                        D=ABS(JJ-YBAR)*0.01786
                        IF(D.LE.DMIN) THEN
                            DMIN=D
                        ENDIF
                    ELSE
                        LCR=R(N,4)
                    ENDIF
                IF(R(N,4).EQ.0) THEN
                    XA=I1
                    YA=JJ
                    CALL RADCAL(XBAR,YBAR,XA,YA,D)
                    IF(D.GE.DMAX) THEN
                        DMAX=D
                    ENDIF
                    IF(D.LE.DMIN) THEN
                        DMIN=D
                    ENDIF
                    XA=I2
                    YA=JJ
                    CALL RADCAL(XBAR,YBAR,XA,YA,D)
                    IF(D.GE.DMAX) THEN
                        DMAX=D
                    ENDIF
                    IF(D.LE.DMIN) THEN
                        DMIN=D
                    ENDIF
                ENDIF
            END DO
        END DO

```

```

        ENDIF
        IF((R(N,1).LE.XBAR).AND.(R(N,2).GE.XBAR)) THEN
            D=ABS(YBAR-JJ)
            IF(D.LE.DMIN) THEN
                DMIN=D
            ENDIF
        ENDIF
611    CONTINUE
700    CONTINUE
        WRITE(*,*) 'DMAX IS',DMAX
        WRITE(*,*) 'DMIN IS',DMIN
C
C THE FOLLOWING PROCEDURE RECOGNIZES THE OBJECT
C PRESENTED TO THE VISION SYSTEM AS ONE OF THE
C KNOWN OBJECTS BY CARRYING OUT PARAMETER
C COMPARISON. IF THE PARAMETER COMPARISON TEST
C FAILS FOR ALL THE KNOWN MODELS. IT MEANS
C THAT THE OBJECT IS NOT KNOWN TO THE SYSTEM
C
        AR11=1.072*AR1
        AR21=1.072*AR2
        AR31=1.072*AR3
        PER11=1.17*PER1
        PER21=1.17*PER2
        PER31=1.17*PER3
        RMX11=1.041*RMX1
        RMX21=1.041*RMX2
        RMX31=1.041*RMX3
        RMI11=1.06*RMI1
        RMI21=1.06*RMI2
        RMI31=1.06*RMI3
        IRECG=0
        IF((AREA(2).GE.AR1).AND.(AREA(2).LE.AR11)) THEN
            IF((PERI(2).GE.PER1).AND.(PERI(2).LE.PER11)) THEN
                IF((DMAX.GE.RMX1).AND.(DMAX.LE.RMX11)) THEN
                    IF((DMIN.GE.RMI1).AND.(DMIN.LE.RMI11)) THEN
                        WRITE(*,*) ' THE OBJECT IS MODEL-A'
                        IRECG=1
                        GOTO 119
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
        IF((AREA(2).GE.AR2).AND.(AREA(2).LE.AR21)) THEN
            IF((PERI(2).GE.PER2).AND.(PERI(2).LE.PER21)) THEN
                IF((DMAX.GE.RMX2).AND.(DMAX.LE.RMX21)) THEN
                    IF((DMIN.GE.RMI2).AND.(DMIN.LE.RMI21)) THEN
                        WRITE(*,*) ' THE OBJECT IS MODEL-B'
                        IRECG=1
                        GOTO 119
                    ENDIF
                ENDIF
            ENDIF
        ENDIF

```

```

        ENDIF
    ENDIF
ENDIF
IF((AREA(2).GE.AR3).AND.(AREA(2).LE.AR31)) THEN
    IF((PERI(2).GE.PER3).AND.(PERI(2).LE.PER31)) THEN
        IF((DMAX.GE.RMX3).AND.(DMAX.LE.RMX31)) THEN
            IF((DMIN.GE.RMI3).AND.(DMIN.LE.RMI31)) THEN
                WRITE(*,*) ' THE OBJECT IS MODEL-C'
                IRECG=1
                GOTO 119
            ENDIF
        ENDIF
    ENDIF
ENDIF
IF(IRECG.EQ.0) THEN
    WRITE(*,*) ' THE OBJECT IS NOT ANY ONE OF THE
* MODELS'
C
119  STOP
    END
C
    SUBROUTINE TO ALLOCATE THE REGION
    SUBROUTINE ALLOC(NEWFLG,BOOK,VACUM,CBLOB,B,BN,BP,
5 CROW,I,J,R,RBLOB,FMOX,FMOY,SMOX,SMOY,AREA,PERI)
    INTEGER B(20,10),R(1000,4)
    INTEGER BP(25),BOOK,VACUM,TMP,RBLOB,BN,CBLOB
    INTEGER*2 CROW(512)
    REAL FMOX(10),FMOY(10),SMOX(10),SMOY(10),AREA(10),
*PERI(10)
    NEWFLG=0
    IF(BOOK.LT.0) THEN DO
        CBLOB=VACUM
        VACUM=VACUM+1
    ELSE DO
        CBLOB=BOOK
        BOOK=B(BOOK,7)
    ENDIF
    CALL REGINI(CBLOB,I-1,BN,BP,B,CROW,FMOX,
3FMOY,SMOX,SMOY,AREA,PERI)
    TMP=B(RBLOB,7)
    B(RBLOB,7)=CBLOB
    B(CBLOB,7)=TMP
    B(TMP,8)=CBLOB
    B(CBLOB,8)=RBLOB
    RETURN
    END
C
C
    SUBROUTINE TO INITIALIZE THE REGION PARAMETERS
    SUBROUTINE REGINI(N,MCOL,BN,BP,B,CROW,FMOX,
3FMOY,SMOX,SMOY,AREA,PERI)
    INTEGER BP(25),B(20,10),BN
    INTEGER*2 CROW(512)

```

```

REAL FMOX(10), FMOY(10), SMOX(10), SMOY(10), AREA(10),
3PERI(10)
BP(BN)=N
B(N,9)=BN
  B(BN,1)=-1
  B(BN,2)=-1
  B(BN,3)=-1
  B(BN,4)=CROW(MCOL)
  B(BN,5)=0
  B(BN,6)=0
  B(BN,10)=0
  FMOX(BN)=0
  FMOY(BN)=0
  SMOX(BN)=0
  SMOY(BN)=0
  BN=BN+1
RETURN
END

```

C

C

```

SUBROUTINE TO UPDATE THE PARAMETERS
SUBROUTINE UPDATE(B,R,CJOG,CBLOB,I,J,PJOG,EL,IL,
3 FMOX,FMOY,SMOX,SMOY,AREA,PERI)
  INTEGER B(20,10),R(1000,4),EL
  INTEGER CJOG,CBLOB,PJOG
  REAL FMOX(10),FMOY(10),SMOX(10),SMOY(10)
  REAL AREA(10),PERI(10)
  L=I-R(CJOG,1)
  R(CJOG,2)=I-R(CJOG,1)
  R(CJOG,3)=B(CBLOB,9)
  R(CJOG,4)=B(CBLOB,4)
  K=R(CJOG,1)
  AREA(CBLOB)= AREA(CBLOB)+L*0.02381*0.01786
  PERI(CBLOB)=PERI(CBLOB)+2*0.01786
  FMOX(CBLOB)=FMOX(CBLOB)+L*((L-1)/2 +K)
  FMOY(CBLOB)=FMOY(CBLOB)+L*J
  SMOX(CBLOB)=SMOX(CBLOB)+L*((L-1)*(K+(2*L-1)/6)+K*K)
  SMOY(CBLOB)=SMOY(CBLOB)+L*J*J
  CJOG=CJOG+1
  IF(EL.EQ.0) THEN DO
    R(CJOG,1)=I
  ELSE DO
    R(CJOG,1)=IL
    PJOG=PJOG+1
  ENDIF
RETURN
END

```

C

C

```

C SUBROUTINE TO DETERMINE THE BLOB NUMBER
SUBROUTINE LOOKUP(ND,BP,LKK)
  INTEGER BP(25)

```

```

C      IF(BP(ND) .LT. 0) THEN DO
          ND=-BP(ND)
      ENDIF
          LKK=ND
          RETURN
          END
C
C  SUBROUTINE TO CALCULATE DISTANCE BETWEEN TWO POINTS
      SUBROUTINE RADCAL(X1,Y1,X2,Y2,D)
          X2=X2*0.02381
          Y2=Y2*0.01786
          D1=(X1-X2)*(X1-X2)
          D2=(Y1-X2)*(Y1-Y2)
          D=SQRT(D1+D2)
      RETURN
      END

```

**The vita has been removed from
the scanned document**