SHAPE RECOGNITION USING SEGMENTATION OF THE BOUNDARY CURVE

by

YAĞMUR DENİZHAN

B.S. in E.E., Boğaziçi University, 1982

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Boğaziçi University

1984

SHAPE RECOGNITION USING SEGMENTATION

OF THE BOUNDARY CURVE

APPROVED BY

Doç. Dr. ERDAL PANAYIRCI
(Thesis Supervisor)

Doç. Dr. YORGO ISTEFANOPULOS

Yard. Doç. Dr. AVNİ MORGÜL

DATE OF APPROVAL

27.6.1984

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to thank to Nazan Tuğbay for her friendly support in the most urgent days of my study.

I am very grateful to all my research assistant friends in the computer center, without their helps I wouldn't be able to finish my thesis.

And finally would like to thank to Özge Gürgöze for her patient typing.

ABSTRACT

In this project a method is presented that enables a shift, ratation and scale independent representation of a 2-dimentional shape in the computer memory and a criterion is proposed for the comparison of two (or more) such shapes. The boundary curve is extracted and segmented in to such parts that can be represented each by itself with a single valued r(Θ) function. The FFT (Fast Fourier Transform) is of these curve segments are taken and the Fourier coefficients are truncated after certain value for the sake of data reduction.

These truncated Fourier coefficients along with the number of curve segments of the shape and a set of values related with the angular span of each curve segment form a representation vector of the shape.

The "distance" between such vectors forms a basis for the comparison of similarity between different sample shapes.

ÖZETÇE

Bu projede sunulan yöntem 2 boyutlu bir şeklin kayma, dönme ve ölçekten bağımsız olarak bilgisayar belleğinde gösterimlenmesini sağlamaktadır. Ayrıca 2 veya daha fazla şeklin karşılaştırılması için bir ölçüt önerilmiştir. Sınır eğrisi çıkarılıp herbiri tek değerli r(θ) işlevi şeklinde gösterilebilen parçalara ayrılır. Bu eğri parçalarının hızlı fourier dönüşümü (FFT) alınır ve Fourier katsayıları veri indirgemesi için belli bir değerden sonra kesilir.

Bu azaltılmış Fourier katsayıları, eğri parçalarının sayısı ve herbir parçanın açısal uzunluğuyla ilgili bir dizi değerle birlikte şeklin gösterim vektörünü oluşturur.

Bu vektörler arasındaki "uzaklık" değişik şekiller arasındaki benzerliğin ölçülmesi için bir temel oluşturur.

CHAPTER I

INTRODUCTION

Shape recognition is an important branch in the field of the 2-dimentional pattern recognition.

As known, shape is the remaining knowledge about any figure after the information about the position and size as well as the grey-level information of the area within the boundary have been removed. In practical applications like robotics, machine-parts classification, biomedical recognition, hand-printed letter recognition, etc. pattern recognition appears in the form of shape recognition since the silhouette usually supplies sufficient information.

The shape recognition problem can be considered in three steps:

1. Extraction of the Shape-Information of an Object from the Scene Picture:

The output of a digital camera supplies sufficient information about the grey-level of each "pixel" to form the matrix data of the scene. The easiest method to obtain the regions (belonging to the objects) is to put an appropriate grey-level threshold and compare the grey-level of each pixel to that. This method is meaningful as long as the contrast between the object and the background is sufficient, otherwise more sophisticated methods must be applied. Another problem arises when there are several objects in the scene. But these problems are out of the scope of this project.

The result of this step can be expressed either with a binary function of x- and y- coordinates, that assumes the value 1 within the object regions and 0 elsewhere, or with a list of the coordinates of the region points. The later is used in this project.

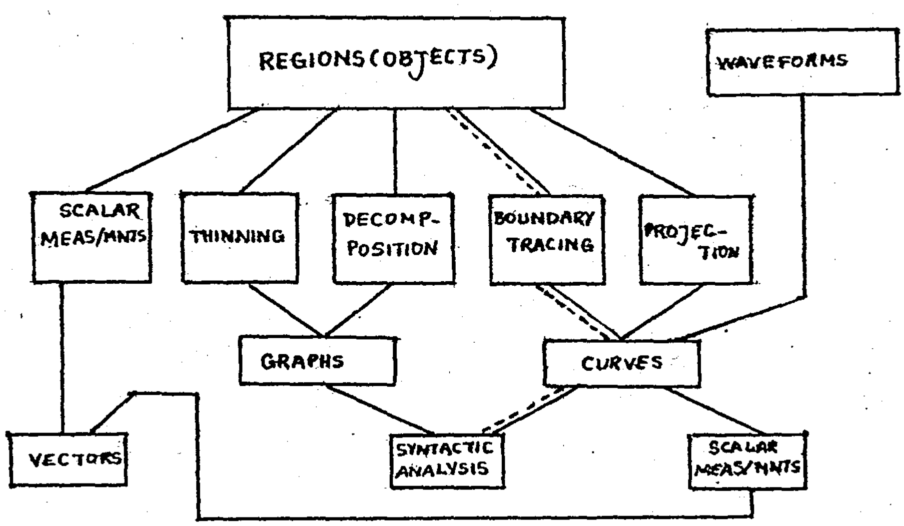
2. Extraction of the Representation Vector:

The shape must somehow be represented in the computer memory, such that this representation will form a basis for comparison of different shapes.

There are various methods for this, each suitable for different application areas.

At this step usually a data reduction is made without loosing however any information necessary for discrimenation of different shapes.

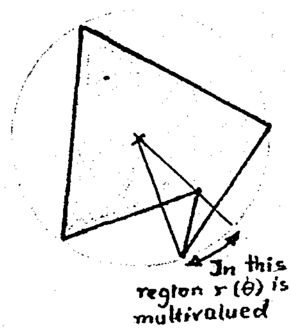
A summary of different methods used is given in the below illustration. The dotted line shows the path followed in this project.



In this project the representation vector extraction goes through the following steps:

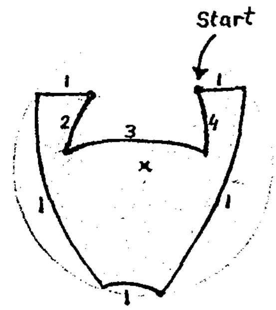
a) The smallest circle encircling the shape and an intrinsic direction of the shape (that will be defined later) are found. The coordinates of the "object-pixels" are transformed into polar coordinates with the origin at the center of the smallest circle and the zero angle in the above mentioned direction. The radius of the smallest circle is used for normalization. Hence the representation is made shift, rotation and scale independent.

b) The boundary curve obtained by curve tracing is to be expressed as $r(\Theta)$. This however may not be possible since the function can be multivalued at some points.



The solution to that proposed in this project is to separate the boundary-curve into such segments, that each by itself can be represented as $r_i(\Theta)$, $i = 1,\ldots,NOS$; NOS being the "number of segments" in the boundary curve of a shape. The following remark about the above mentioned segments should be made:

If we start from a certain point on the boundary and move on it, let us say CCW, and note the angle of center between the point we are on and a certain reference direction, we will see that the derivative of the angle of center with respect to time, as we keep going on the boundary may change sign, i.e. although we move CCW on the boundary we may move CW or CCW with respect to the origin (center of the circle). Each time such a change in the direction of rotation with respect to the origin occurs we can say that a new segment has started.



In this figure it is shown how the boundary curve is segmented. The numbers are the numbers of the segments.

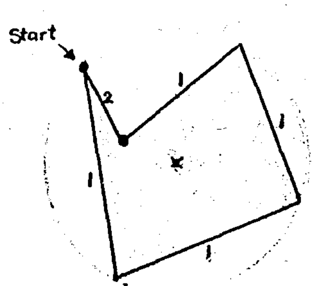
Inspite of this measurement still a segment with multivalued $r(\theta)$ function can occur, if it has a spiral-like behaviour, i.e. its angular span is greater then 360^0.

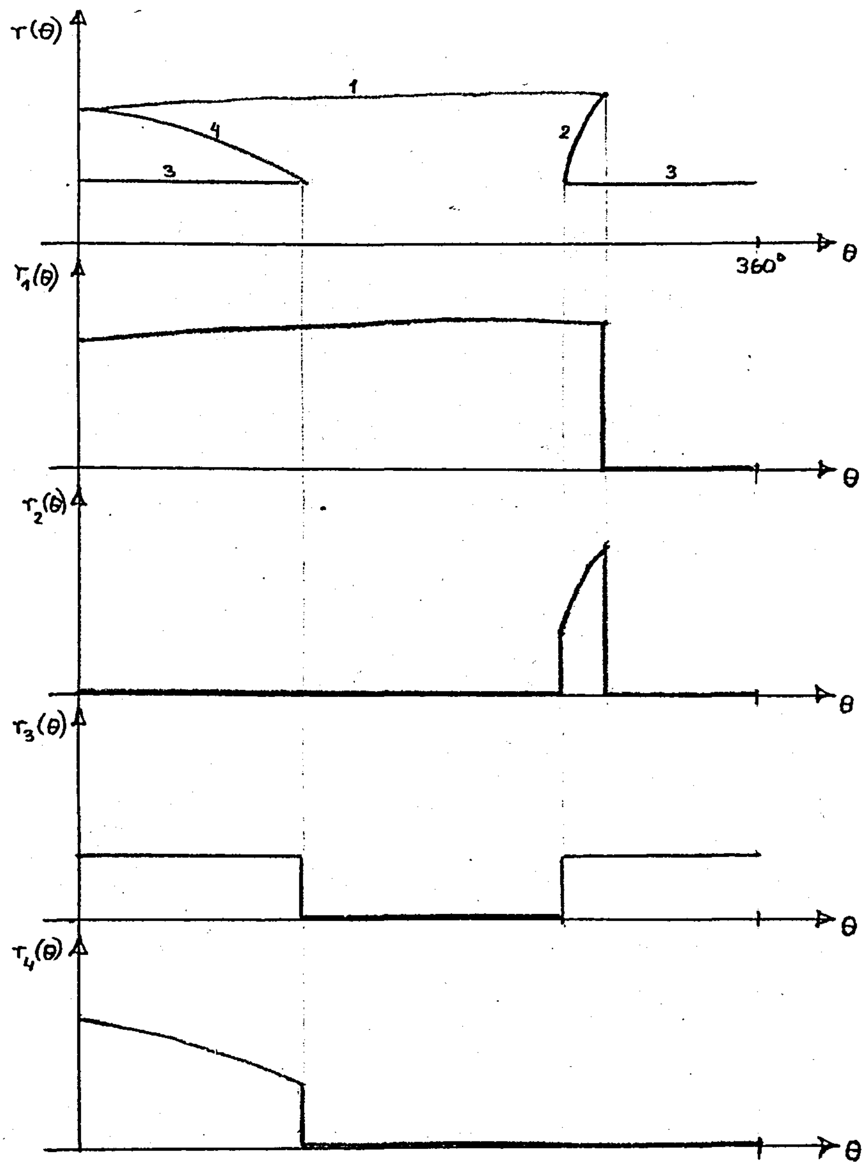
This case is shown in the following figure. Here segment number 1 has an angular span greater than 360^0. This problem is solved treating the $r_1(\theta)$. function of this segment as one with period $2x360^0$. Hence we guarantee that no overlapping in $r_1(\theta)$ itself occurs.

In general, if a particular segment(i) rotates n_i times in itself, this number n_i is noted and its $r_i(\theta)$ function is represented within $0 < \theta < n_i x360^0$.

Another remark about $r_i(\theta)$'s must be made:

$r_i(\theta)$ assumes the r value of i'th segment in the θ range where the segment exists and the value 0 everywhere else.





c) $r_i(\theta)$, i = 1,...,NOS are sampled, to obtain 2^m data points for each segment. This treatment is necessary for the FFT that will be used in the next step.

A convenient value for m can be chosen, keeping in mind that increasing m increases the number of resulting Fourier coefficients, the length of the representation vector and hence the acquired computer time, on the other hand decreasing m too much may cause severe loss of information. In the example program given as appendix, m is taken as 7.

d) The FFT's of the sampled $r_i(\theta)$ functions are taken, which results in a set of complex Fourier coefficients for each segment.

At this stage a data reduction can be made considering that the lower order Fourier coefficients carry sufficient global information about the functions.

Choosing an appropriate index of reduction (IOR) only the first $2^{(m-IOR)}$ coefficients are taken (Note: IOR must be less or equal m).

e) The representation vector is constructed with the number of segments (NOS) the numbers n_i indicating how many turns a segment makes (see. 2.6) and the reduced Fourier coefficients.

They must be ordered however in a certain way according to the order number given to each segment. But until now the order number of the segments depended on the starting point of the curve tracing. So at this stage the segment with the greatest angular span is called the 1st segment. (If there are several much segments the one with the greatest r at the first sample point is chosen, if also these are equal second sample points are compared etc.). The next segment that appears when moving on the boundary is then 2nd segment etc.

The representation vector looks like the below one:

NOS	number of segments
n_1	
.	
.	turn number of each segment
.	
n_{NOS}	
Re_{11}	
Im_{11}	real and imaginary parts of the reduced
.	Fourier coefficients of the 1st segment
Re_{1l}	
Im_{1l}	
.	
.	F. coefficients of the NOSth segment
.	

3. COMPARISON:

After obtaining an appropriate representation of the shape the last problem of shape recognition is to find a criterion for similarity between different shapes.

The similarity is measured by the "distance" between the representation vectors of different shapes. This distance can be compared to a threshold, can be minimized or used for clustering (in the case of nonsupervised learning) But it must be defined in an appropriate way depending on the application field, type of the vector etc. A popular definition of distance is for example the Mahalonobis distance $d = [(\underline{v}_1 - \underline{v}_2)^T \underline{\underline{\Sigma}}^{-1} (\underline{v}_1 - \underline{v}_2)]^{1/2}$

The simplest case is where $\underline{\underline{\Sigma}} = \underline{\underline{I}}$, the Euclidean distance.

For the representation vector constructed in 2 e) the following criteria are found suitable for measuring similarity:

i) If the first entries (NOS) of two rep. vectors are different they certainly represent different shape classes.

ii) If they are equal, consider the next NOS terms which indicate the number of turns each segment makes. If they are not identical (i.e. $n_{i1} = n_{i2}$, $\forall\ i = 1,\ldots,NOS$) the vectors again represent different classes.

iii) If first NOS - 1 terms are identical, treat the next of the representation vector separately as a vector, find the Euclidean distance between two such vectors and divide this by NOS. The resulting distance can now be compared to a threshold or used for clustering. The threshold can be determined experimentally. It usually will depend on object size-to-pixel ratio and noise.

This project deals with the extraction of the representation vector only, and suggests a distance criterion for recognition.

The algorithms for finding the smallest circle, boundary tracing, boundary segmentation and interpolation for sampling are original. The FFT program is borrowed as a package program.

CHAPTER II

ALGORITHMS

The input of the whole program consists of a list of object-point-coordinates and some parameters that must be specified according to application area.

A discussion about the determination of parameter values will be given in Chapter III.

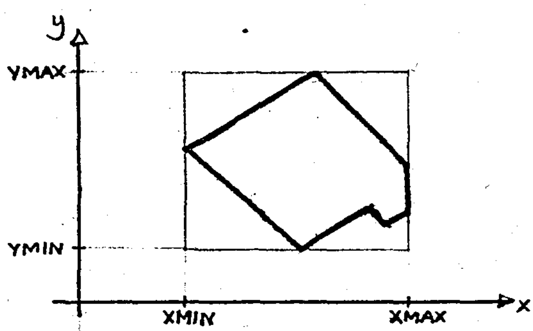
In the algorithms, the points are currently assigned to different groups and are stored in arrays. Storing a point however means storing two coordinates. This unnecessary occupation of memory is avoided giving each point a number. It is arbitrary in which order this assignment is made. Usually as the scene is being scanned the data is sent simultaneously to the memory, so, the number can simply be given according to the order of entrance. This number is the identification number of the point and for example if a point belongs to a certain segment its identification number is simply stored in the array of the segment. Using this number it is always possible to go back and look up the coordinates of the point.

1. Algorithm for Finding the Smallest Circle:

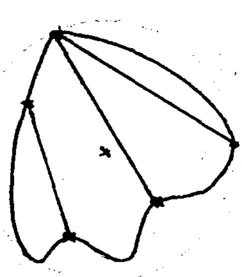
As long as no strong restrictions are put on a shape there exists no analytical method for finding the center of the smallest circle that encloses it. Hence an iterative method is necessary. In fact most of the points of the shape are redundant for the determination of the smallest circle. Taking the iteration into account it is desirable to get rid of as much redundant points as possible.

Reduction of Points:

- First the smallest rectangle enclosing the shape with sides parallel to the particular Cartesian axes is found. For that purpose it is sufficient to search for the maximum and minimum x and y coordinates that appear in the data.

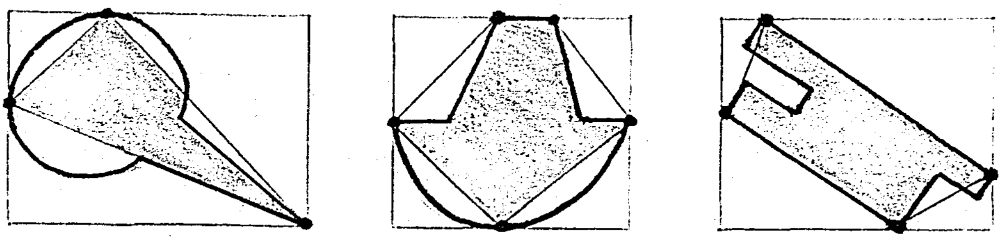


- Theorem: Points that lie on a line segment between any two points of the boundary are redundant for the determination of the smallest circle, because such a line segment is either chord of the smallest circle or lies completely within the circle.



Hence if one has all the boundary points it is possible to get rid of all redundant points, but we don't have them.

Still there are some points that we know that they certainly lie on the boundary, these are the ones touching the sides of the smallest rectangle. Connecting those points one obtains a polygon (minimum 3, maximum 8 corners).



As can be seen in the above examples also all points within that polygon are redundant. On the other hand those outside the polygon may or may not be so. Therefore we keep only those points outside that polygon and the corners of the polygon and use only these points in the iteration.

To determine whether a point lies within or outside the polygon the following analytical method is used:

The vector equation for each side-line is written in the following manner:

For a side with corner coordinates $\begin{pmatrix} x_i \\ y_i \end{pmatrix}$ and $\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}$

the equation looks like:

$$\begin{pmatrix} n_x \\ n_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} n_x \\ n_y \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} = 0$$

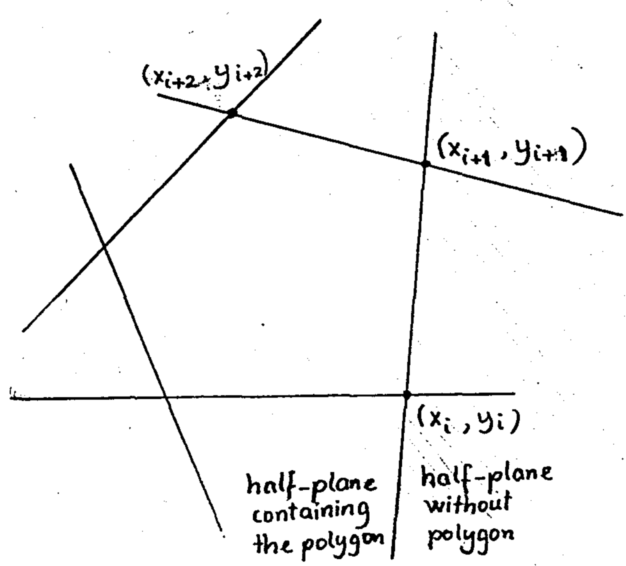
where $\begin{pmatrix} n_x \\ n_y \end{pmatrix}$ is the normal vector of the line

$$\begin{pmatrix} n_x \\ n_y \end{pmatrix} = \begin{pmatrix} y_i - y_{i+1} \\ x_{i+1} - x_i \end{pmatrix}$$

Since each side-line divides the 2-dimentional plane into two half planes one must determine which half plane is the one that containes the polygon. To do that a third corner $\begin{pmatrix} x_{i+2} \\ y_{i+2} \end{pmatrix}$ is placed into the equation of the side (between (x_i , y_i) and (x_{i+1} , y_{i+1})).

The right handside of the equation becomes different than zero. The sign of this value at the right handside supplies the knowledge about the half-plane that contains the polygon.

Any point that gives a result with the same sign is in the half-plane that contains the polygon.



If a point lies in the half-plane containing the polygon for each side-line then it lies within the polygon.

All points are checked and only those outside the polygon are used in the iteration.

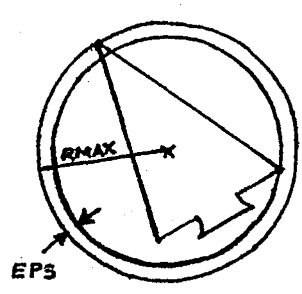
Iteration for Finding the Center:

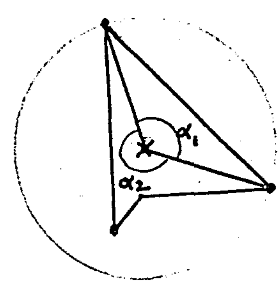
The method used in the iteration resembles the intuitive behaviour of a human trying to encircle an object with a ring with adjustable radius. He first ensures that the object lies within the circle, than he reduces the radius to the smallest possible value, given the particular center. After doing so he checkes the touching points. He tries to shift the ring in the direction of the touching points, such that they don't touch anymore and hence the ring-radius can be reduced. If there are however other touching points in the "opposite direction" of the shift, such a shift is of course not justified. When no shift and no reduction of radius is possible the smallest circle is found.

This intuitive method is applied as follows:

- Initiate the center coordinates of the circle with those of the smallest rectangle.

- Find the furthest data point(s) from this center and note the angles of center between these points. The distance of the furthest point is RMAX for the time being. A tolerance of iteration EPS (epsylon) is chosen, and also those points with a distance from the center greater or equal to (RMAX - EPS) are considered as having maximum distance. This corresponds to considering the circle as a ring of with EPS. (More information about EPS is given in the "Comments")





- Call the so formed angles od center α_i's. There will be as many α's as the number of points touching the ring with the above determined center and radius RMAX.

. If there is a single such point shift the center coordinates in the direction of this touching point for a distance STEP. (STEP must be chosen by the programmer)

. If there are several such points check if there is any α greater than 180^0. If there is such an α_i shift the center of the circle in the direction of the bisector of α_i for a distance = STEP .

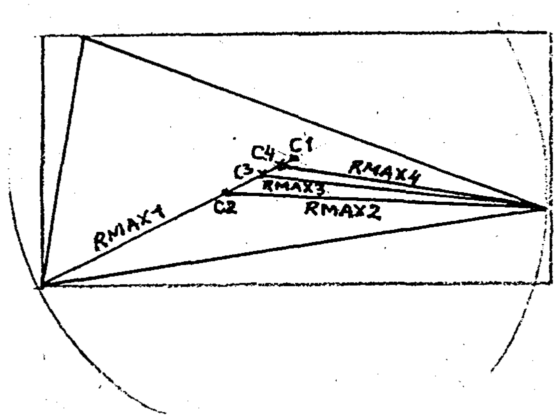
. If there are more than one touching points but no α_i greater than 180^0 the iteration finishes, the last coordinate values of the center are the correct ones.

. If a shift has occured find the furthest points from the new center and note its distance.

. If this distance is greater than the previous value of RMAX this is an unjustified shift, hence go back in the same direction to half of the distance from the old center. Find again the furthest point and repeat this step until you obtain a radius less than the previous RMAX.

. If the distance of the furthest point is less than RMAX this is a justified shift. The new value of RMAX is the distance of the furthest from the new center.

As stated above the iteration is terminated when no α_i greater than 180^0 is found. The below figure shows different steps of the iteration.



The coordinates of the data points (including those within the polygon) are transferred into polar coordinates with the origin at the center of the circle, i.e. two new arrays are generated where a list of r and θ values are stored according to the identification number of each point. The last value of RMAX, which is equal to the radius of the smallest circle is stored. Later the Fourier coefficients are divided by that value for the sake of scale normalization.

2. Algorithm for Boundary Tracing and Segmentation:

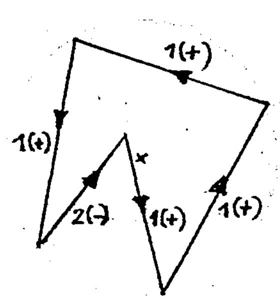
In the following algorithm both the boundary extraction from the region data and segmentation of the boundary are made by simultaneously.

Before explaining the algorithm let us first define some concepts:

We are going to trace the boundary starting from a point that touches the smallest ciecle, because such a point certainly belongs to the boundary. We will move on the boundary CCW, this however doesn't mean that the angle of center with respect to the origin (center of the smallest circle) is always going to increase as we move. This is anyway how different segments of the boundary are distinguished from each other.

Segments that yield increasing angle of center as we keep moving on the boundary in CCW sense will be assigned a positive "turn" (ITURN = +1 in the program), those that yield decreasing angle of center a negative turn (ITURN = -1).

As we keep moving on the boundary a + turn segment will be followed by a -turn segment and vice versa.

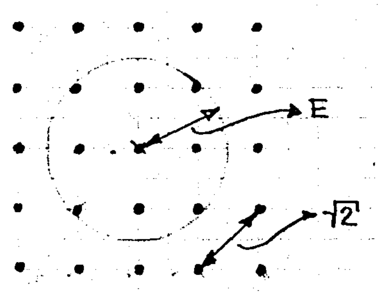


In the above figure the arrows show the direction of the motion. In this particular case segment 1 has a +turn and 2 a -turn.

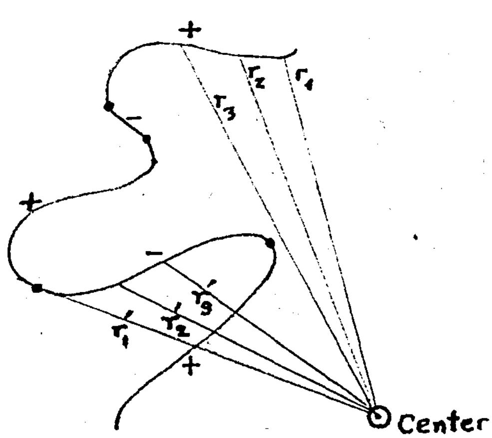
<u>Tracing</u>:

- Tracing starts as mentioned above from a point that touches the smallest circle.

- The nearest points to the starting point are found checking the distances of all data points from the starting point. The nearest neighbours are those within a neighbourhood with radius E = 1.01 x $\sqrt{2}$, this value allows maximum 8 nearest neighbours. (E is discussed in further detail in the "Comments" part).



- The nearest neighbours are checked to find a next point that will give a +turn with respect to the first point. If there are several such points the one with greatest r value is chosen since as we are moving on a +turn segment we always want to have the outmost points. The point found is noted as the next point of the first segment and of the boundary. If there is no nearest neighbour with +turn then the nearest neighbour with -turn and smallest r is chosen, since this means that we are starting with a -turn segment. The nearest neighbour with the smallest r is chosen as the next point in this case, because a -turn segment on aCCW boundary means "beginning of a concavity", hence we want the in-most point.



- Once the next point is determined, now this point is treated as the center of a neighbourhood with radius = E. The nearest neighbours are checked to find a point that yields the same turn (better to say not the opposite turn, since points on the same radius of the enclosing circle, i.e. those with 0-turn-can be considered still being on the same segment) as the turn of the particular segment we are on. If there is no such point within E , increase E for $1.01x\sqrt{2}$ and search for the nearest neighbours yielding the same segment-turn within this radius, if still no such point exists keep on increasing E up to a certain limit, which is conveniently can be taken as three times the original value of E.(The significance of this limit is discussed in the "Comments"). If there still no next point that yield the same turn as the one of the segment, this means that a new segment is starting, which of course has the opposite turn.

In this case go back to the neighbourhood with radius $E = 1.01x\sqrt{2}$ and choose the point with greatest or smallest r depending on whether the new segment has a + or - turn respectively.

When a point is determined as the next point and placed in a segment, put a check on this point, such that it won't be considered afterwards.

- The tracing is finished when the first starting point is found as the next point of a point, hence the loop is finished.

- The turns of the first and last segments are compared, if they are equal they are mended and number of segments is decreased by 1.

(All along this algorithm the distance calculations between points are made using the Cartesian coordinates, since

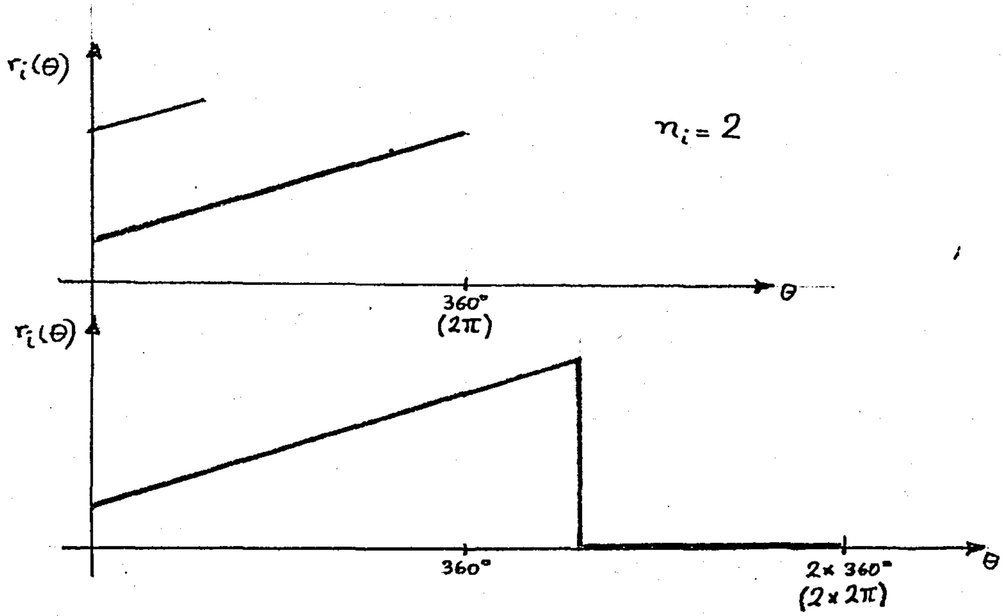
i) Distances don't change with coordinate transformation

ii) Cartesian coordinates are more suitable and yield less calculation error).

3. Turn Correction, Span Expansion, Determination of n_i's and Sampling: The following processes are applied to each segment.

- If the segment has a -turn, the order of the segment points is reversed.

- Checking the angles of the segment points the angular span and n_i are determined. For those segments with $n_i>1$ the definition interval of $r_i(\theta)$ is expanded to $n_i x 2\pi$.



- $r_i(\theta)$ is sampled with a grid separation DELTA where DELTA $\equiv n_i 2\pi/(2^m - 1)$. Such a sampling yields 2^m points.

In the sampling process for a sampling point that lies between two data points simple interpolation is used, i.e. <u>given</u> r_i, θ_i and r_{i+1}, θ_{i+1}

$\theta_i < (k-1)\text{DELTA} < \theta_{i+1}$

the r value for k^{th} sampling point

$r_k = r_i + |(k-1)\text{DELTA} - \theta_i|(r_{i+1} - r_i)/(\theta_{i+1} - \theta_i)$

4. Determination of the Zero Angle:

To make the representation rotation-independent an intrinsic direction of the shape must be defined. This is defined as the starting angle of the longest segment.

If there are several such, the one starting with greatest r value (etc., as described in II.e) is chosen.

5. Fourier Coefficients and Normalizations:

The FFT of each segment is taken using the 2^m sampled points. The obtained coefficients are truncated with IOR and are combined in magnitude and phase form. IOR will be discussed in further detail in the "Comments". The magnitude is divided by RMAX for the sake of scale normalization.

The phase is shifted for θ_0/n_i, where θ_0 is starting angle of the longest segment - which will start from 0^0 from now on (by definition) -.

The reason for dividing θ_0 by n_i is the fact that such a shift means really a shift of θ_0 degree if n_i is 1, i.e. $r_i(\theta)$ is defined in $[0, 2\pi]$, but for $n_i > 1$ a shift of θ_0 degree in with respect to the absolute coordinates, will mean only a shift of $\underline{\theta_0}$ degree of phase shift, since $r_i(\theta)$ has a period of $2\pi x n_i$.

After the normalizations the coefficients are again converted into real-and-imaginary form.

CHAPTER III

COMMENTS

1. The input of the algorithms consists of the coordinate data and the values assigned to some parameters. The choice of these values is left to the programmer. Let us consider these parameters one-by-one:

a) EPS = This is the tolerance of iteration used as the ring width in the algorithm for finding the smallest circle.

A small ring width will yield a more exact result, but will also increase the number of iterations applied until this result is reached. On the other hand the choice of a large EPS can cause a large deviation of the center coordinates from the true value. (The possible deviation is approximately equal to EPS). Such a deviation is especially dangerous for the next stages of the method presented. The segmentation of the boundary curve is made according to a change of sense of rotation with respect to the center. Hence a deviation in the center coordinates can result in finding a wrong (actually non-existing) segment or in skipping a segment that does exist.

In view of this danger and the fact that the iteration used here doesn't acquire too much time anyway, a small value for EPS should be preferred. In the program given in the appendix it is taken as 1, a rather convenient value that reduces the possible deviation to a pixel size.

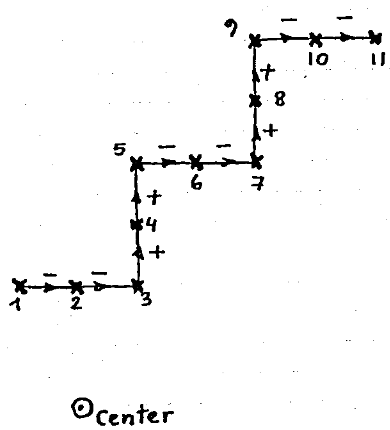
b) STEP = This value is the step size used in the determination of the smallest circle. Although appearently a large STEP size will decrease the number of iterations, this is not true, since if a too large STEP is used, the center will be shifted through the right location and beyond it.

So in the next iteration a shift-back will occur. In view of this the choice of STEP seems to be rather arbitrary, usually a fraction of the diagonal length of the smallest rectangle is a convenient value.

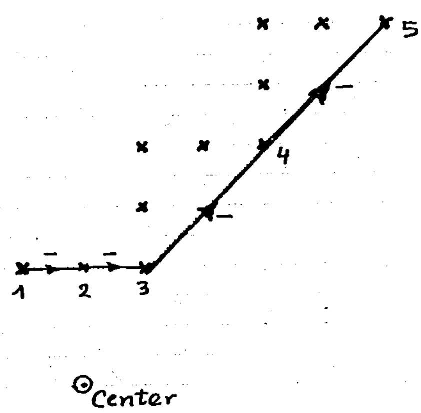
c) E and upper limit of E = E is the radius of nearest neighbourhood used in boundary curve tracing.

When we try to trace the boundary curve what we do is in fact to make an interpretation about the continuity of a curve given a discrete data. The number of the nearest neighbours of a pixel is 8. So a radius slightly larger than the distance between the center pixel and a diagonal neighbour ($\simeq\sqrt{2}$) is the smallest value that can be chosen for E.

As described in Chapter II, the neighbours within a radius E are checked to find a point that will be next member of the last segment, i.e. a point that yields the same "turn" as the segment turn is sought after. When such a point is not found E is increased step by step up to a certain limit. Let us clarify the meaning of this procedure with the help of the following figures:



without increasing E

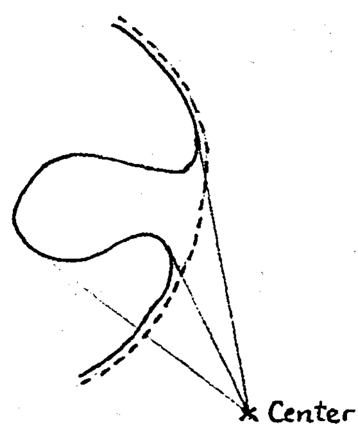


increasing E

The left figure shows what happens if the radius of nearest neighbourhood is not increased: Each change in turn is accepted as a new segment, resulting in a series of segments each consisting of twopoints only, this is however a serious mistake because such a data like shown in the figures would usually result when a straight line appears with an angle of 45^0 to the particular coordinate system. Hence a line that would be represented by a single segment can look like consisting of several segments when rotated for 45^0.

The right figure shows the case when E is increased to find a point that will keep the segment going on. This is actually nothing but "smooting". The upper limit put on E determines the degree of smoothing.

If the upper limit is chosen too large some of the segments will simply the skipped.



The solid like shows the true boundary whereas the dotted line represents the result of a high upper limit of E.

Choosing the starting value (usually taken $\sim 1.01x\sqrt{2}$) of E to high has practically the same result as a high upper limit, some segments will be automatically skipped.

An upper limit of $2x1.01\sqrt{2}$ takes care of segmentation mistakes due to rotation with respect to the coordinate system and any mistake up to approximately 2 pixel size.

An upper limit of $3x1.01\sqrt{2}$ is rather convenient, it means a smoothing of the order of 3-pixel size, a reasonable tolerance in view of possible noise and EPS (resulting in a tolerance of center location).

d) m and IOR = (In the program m appears as NB2 = m of sampling points with basis 2)

Each $r_i(\theta)$ sampled to obtained 2^m data points, high m values ensure a better sampling, but increase the number of the resulting Fourier coefficients.

It should be noted that $2\pi x n_i/2^{(m-1)}$ determines the sampling grid spacing, this spacing is however in terms of angle. As a result of this observation one could say that for boundary parts passing near the origin a low m will be sufficient since the angular spacing between neighbouring pixels near the origin are much larger than that between pixels far from the origin. But a unique m value must be chosen that will take care also of the worst conditions. So the choice of a high m value is meaningful especially for those parts of the boundary that are far from the origin.

IOR shows the extend of truncation of the Fourier coefficients, namely the original number of coefficients is divided by 2^{IOR}.

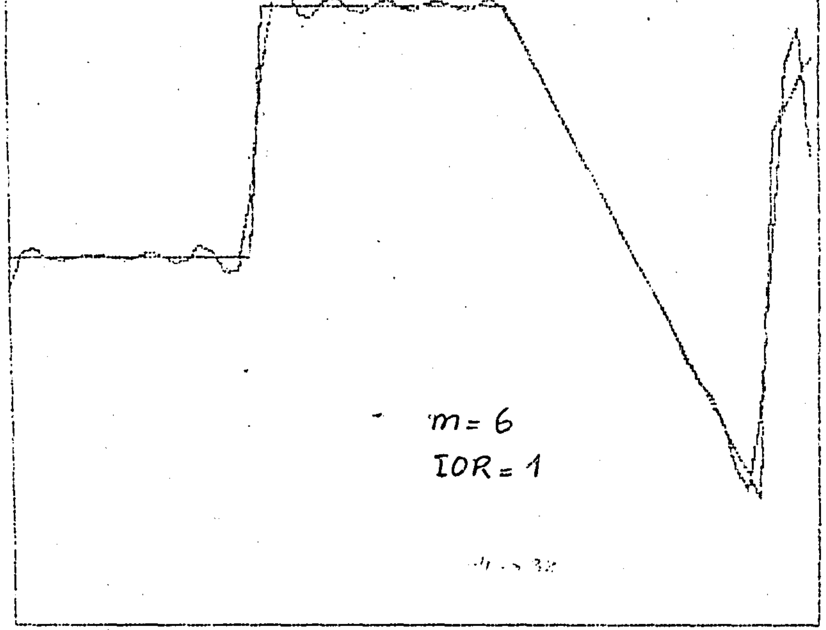
The figures in the next two pages show the effects of different IOR values and justify the truncation of the Fourier coefficients.

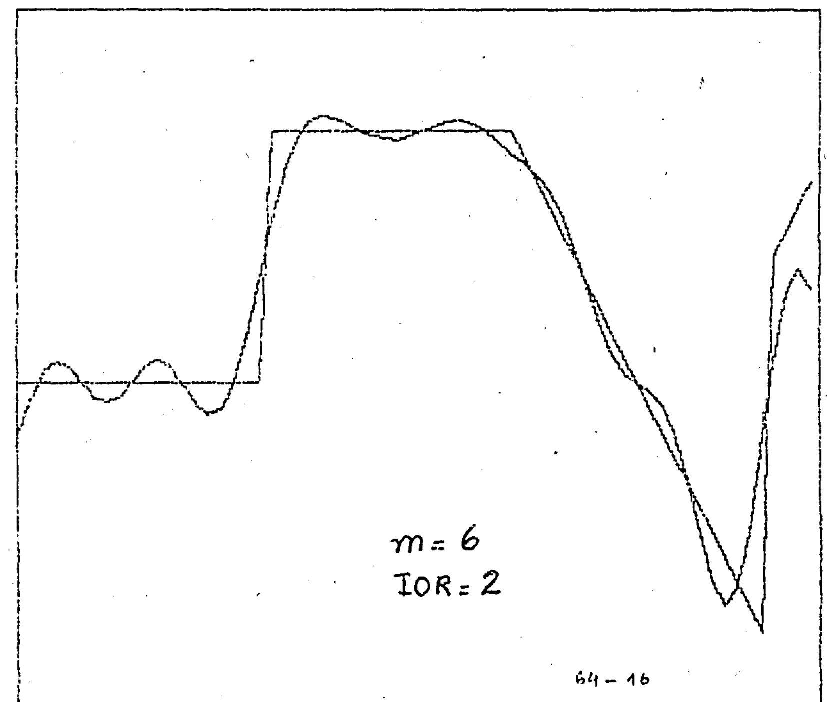
In general even very few coefficients can yield enough information for discrimination of shapes with same NOS and n_i (i = 1,...,NOS) values.

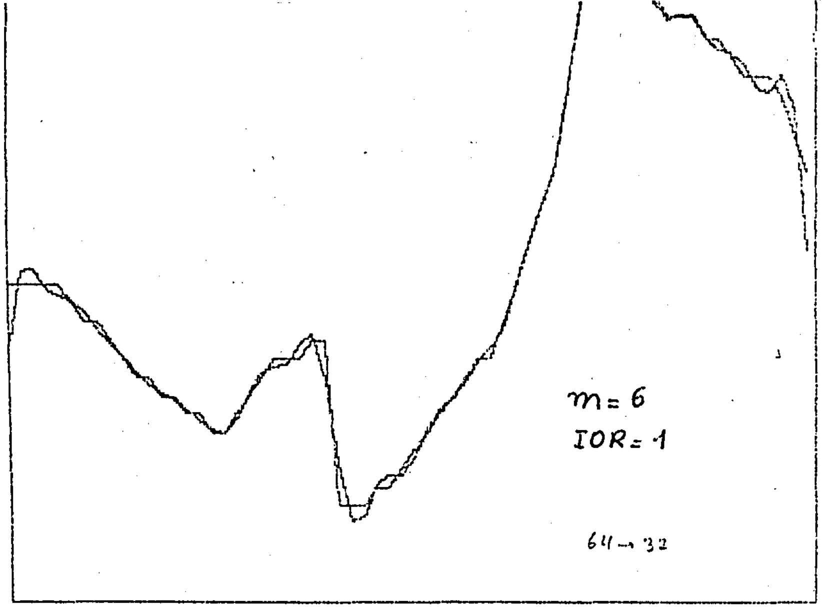
Since the final length of the representation vector depends only on the difference (m - IOR), it is advisable to increase both m and IOR for better result without changing the rep. vector length.

2. The presented method for shape recognition has the following handicaps:

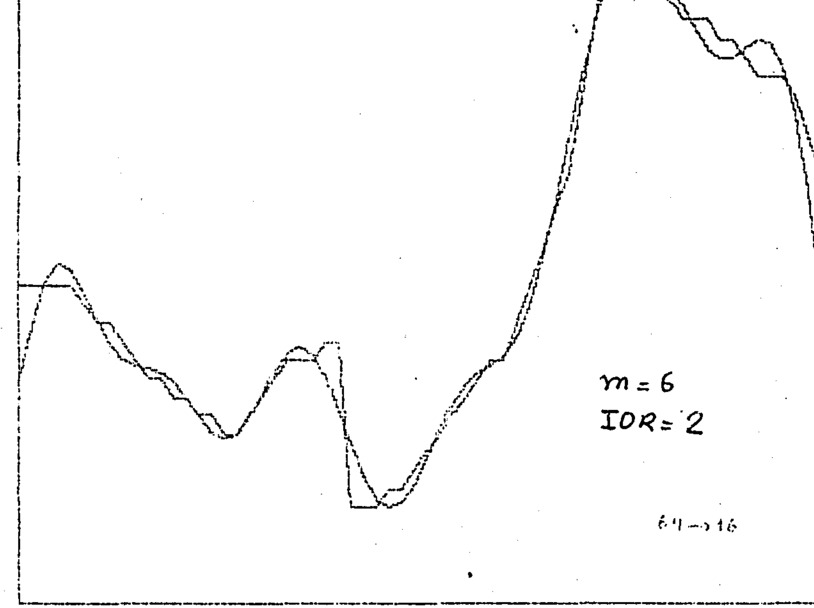
a) As can be uderstood from the algorithm II.2 only the outmost boundary is extracted, in other words the "holes" in a shape - if there are some - are ignored.


m = 6
IOR = 1

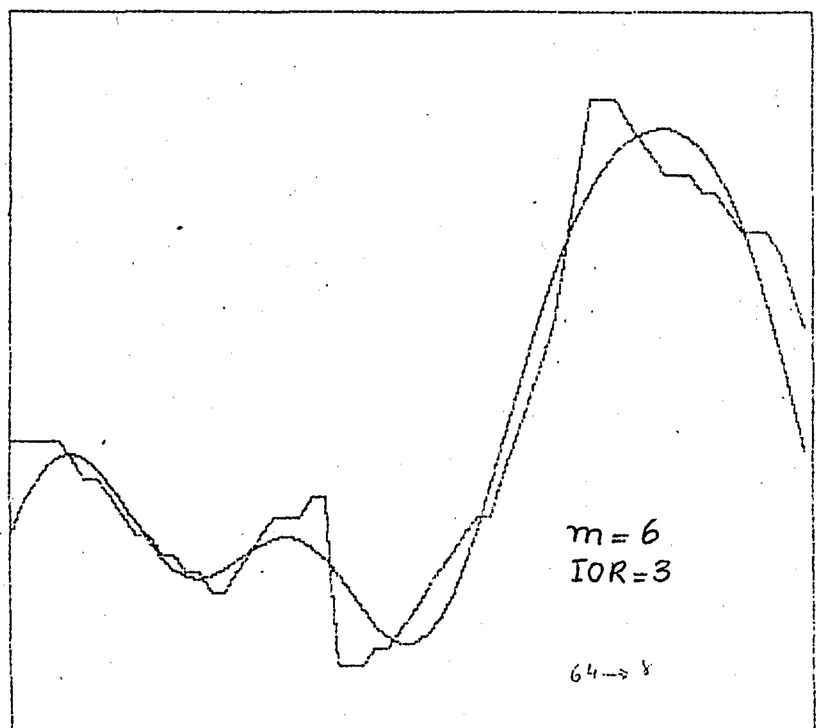

m = 6
IOR = 2
64 – 16



m = 6
IOR = 1
64 → 32



m = 6
IOR = 2
64 → 16



m = 6
IOR = 3
64 → 8

b) As stated in I.3, a "distance" is defined only between shapes with same NOS and n_i , i = 1,...,NOS. Whereas shapes with different NOS or n_i, i = 1,...,NOS are automatically considered as belonging to different classes without allowing any tolerance. But this property can also be considered as an advantage as will be explained in the next chapter.

CHAPTER IV

DISCUSSION AND CONCLUSION

1. Discussion:

As summarized by the simple illusration in I.2, there are very different methods of shape recognition each suitable for another application area.

Classifications of different methods can be made according to various criteria:

a) External and internal methods, depending on whether they consider the whole (object) region or only the boundary, respectively (the method presented in this thesis is an internal one).

b) Scalar transform and space domain techniques, depending on, whether they transform the picture into an array of scalar or into another picture, respectively (our method is a scalar transform technique).

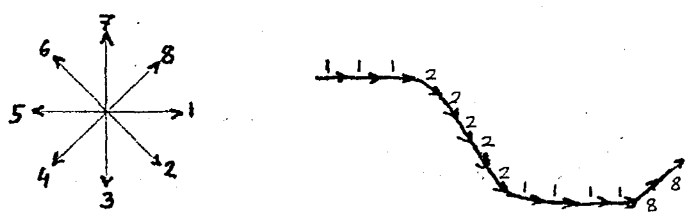
c) Information preserving and information nonpreserving techniques, depending on whether a reasonable approximation of the picture can be reconstructed from the shape descriptors or not. (the classification of our method according to this criteria depends to a great extend on the value of IOR).

Let us go over the most important techniques:

A very common class of techniques is the contour processing in space domain, where the shape can be approximated by polygons or higher order curves or a syntactic analysis can be made.

Polygonal approximations are presented by Pavlidis |2|, Davis |3|, Freeman |4|. In synactic analysis the contour is encoded into a string of the form $V = V_i$, $i = 1, \ldots, n$ where V_i can be an element of a chain code, side of a polygonal approximation, a quadratic arc etc.

As a matter of fact the Freeman chain code is a simple but efficient description of the boundary, where the 8 principal directions are given numbers and the boundary is described in term of these numbers.



Another class of techniques is the transformation into graphs and thinning algorithms. The well-known method of medial axis transformation (Skeleton) is presented by Blum and Nagel |5| and used for a linear approximation of the skeleton by Montanani |6|.

There are also projection methods - projections of the shape taken in one or several directions - and an interesting technique is the one that makes use of the probabilistic distribution of all possible cord lengths of the boundary curve for shape matching by S.P. Smith and A.K. Jain |7|.

A set of more promising techniques involve the Fourier transform of the boundary. This is expressed in terms of tangent angle versus arc length as presented by Zahn and Roskies |8| and some others or as the complex function $b_x(*) - jb_y(*)$ as used by Persoon and Fu |9|.

Among these methods thinning algorithms are those that give emphasis to the skeletal structure rather than the boundary curve. Because of this property they are more suitable for application areas like biological recognition or hand printed letter recognition, where the skeleton of the shape supplies more information necessary for the discrimination of different classes than the boundary does.

The projection techniques - although they require less computation result in considerable loss of information.

Almost all other methods deal with the boundary.

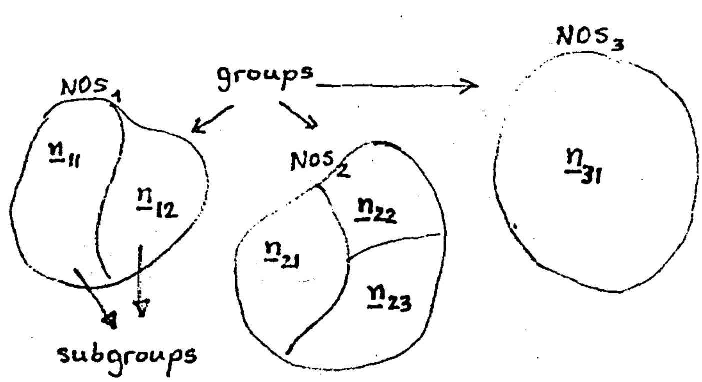
The method presented in this thesis can be considered as a combination of syntactic analysis of the boundary and Fourier transform methods.

In the first part we try to achieve the recognition on the basis of the string $V = V_i$, $i = 1, \ldots, n$ where V_i 's consist of the number of boundary segments (NOS) and n_i 's (defined in Chapter I). If no unique result can be obtained at that stage, the Fourier coefficients are introduced for final discrimination.

The basic advantage of our method is that the computational complexity is not inserted until necessary.

2. Conclusion:

Although clustering methods can be applied in cases where no knowledge about the classes is available (below figure shows to n_i's; clustering methods can be applied within the subgroups), applications where a certain knowledge abot the classes is available can make use of the above mentioned advantage of the presented method.



b) Another property of the method is its sensitivity to the proportions of the shapes. Variations of the proportions that could be tolerated in other techniques can cause a shift in the location of the center of the smallest circle, which again can cause a change in the number of segments - the most important criterion used for the recognition.

So it is preferable to use this method in cases where the samples to be assigned to the same class are expected to have completely identical shape and proportions - for example this is not the case in hand-printed letter recognition or biological applications -.

An area that satisfies both conditions a) and b) is certainly the machine part recognition. In a factory the number of different machine-part-types to be produced are usually known (these known classes can even be discriminated using only NOS values = a great simplification of the recognition problem), on the other hand in series production different samples of the same machine part must have exactly the same shape.

So,a system where this methos is expected to yield successful application can be described as follows:

. An assembly line carrying different types of machine parts waiting to be sorted.

. A digital camera taking snapshots synchronized with the line motion, such that it supplies single-object.pictures of each part that constitute the input of the algorithm.

. A robot arm that sorts the machine parts according to the decisions of the algorithm.

BIBLIOGRAPHY

|1|. T. Pavlidis, Structural Pattern Recognition, Springer - Verlag, 2nd Edition 1980.

|2|. T. Pavlidis, "Polygonal Approximations by Newton's Method" IEEE Trans. Computers vol.C-26 (1977), pp.800-807.

|3|. Davis, L.S. "Understanding Shape: Angles and Sides" IEEE Trans. Computers vol.C-2 (1977), pp.236-242.

|4|. Freeman, H. "Boundary Encoding and Processing" Picture Processing and Psychopictorics (B.S. Lipkin and A. Rosenfeld, editors), Acad. Press, 1970 pp.241-266.

|5|. Blum, H. and Nagel, R. "Shape Description using Weighted Symmetric Axis Features "Proc. IEEE Pattern Recognition and Image Processing Conference", Troy, N.Y., June 6-8, 1977, pp.203-215.

|6|. Ugo Montanori, "Continuos Skeletons from Digitized Images", Journal of the Association for Computing Machinary, Vol.16, No.4, October 1969, pp.534-549.

|7|. S.P. Smith and A.K. Jain, "Chord Distributions for Shape Matching", Computer Graphics and Image Processing 20, pp.259-271, (1982).

|8|. Zahn and Roskies, "Fourier Descriptions for Plane Closed Curves", IEEE Trans. on Computers, vol. C-21 (1972), pp.269-281.

|9|. Persoon and Fu, "Shape Discrimination Using Fourier Descriptors", <u>IEEE Trans. on Systems, Man and Cybernetics</u>, vol. SMC-7 (1977), pp.170-179.

APPENDIX

```
M MAIN          74/176  OPT=0,ROUND= A/ S/ M/-D,-DS       FTN 5.1+577
-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST
TREC3,B=BIN.


      PROGRAM MAIN (INPUT,OUTPUT,TRANS,TAPE6=OUTPUT,TAPE8=TRANS)
C
      DIMENSION IX(900),IY(900)
      DIMENSION X(900),Y(900),ISXA(2),ISXI(2),ISYA(2),ISYI(2)
      DIMENSION IPOLY(8),C(8),SIDE(8),ITU(800)
      REAL NX(8),NY(8)
      DIMENSION LIR(90),A(900),K(90),R(900),TH(900)
      DIMENSION INEAR(80),B(80),ICHECK(900)
      DIMENSION INUM(100),ISEG(100,500),ITURN(100)
C     DATA (IX(I),I=1,291)/20,21,22,23,24,25,26,27,28,29,30,31,
C    &32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,
C    &51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,
C    &70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,
C    &89,10*90,91,92,93,94,95,96,97,98,10*98,98,97,96,95,94,93,
C    &92,91,10*91,90,89,88,87,86,85,84,83,82,81,80,79,78,77,76,
C    &75,74,73,72,71,9*71,70,69,68,67,66,65,64,63,62,61,60,59,58,
C    &57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,
C    &37,36,35,34,33,32,31,30,20*30,31,32,33,34,35,36,37,38,39,40,
C    &40,8*41,3*40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,
C    &23,22,22,2*21,5*20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,
C    &35,36,37,38,39,40,41,42,43,3*44,3*45,6*46,45,44/
C     DATA (IX(I),I=292,516)/43,42,41,40,39,38,37,36,35,34,33,32,31,
C    &30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,
C    &9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4,-5,-6,-7,-8,-9,-10,-11,-12,-13,
C    &-14,-15,-16,-17,-18,-19,-20,-21,-22,-23,-24,-25,-26,-27,-28,-29,
C    &-30,-31,-32,-33,-34,-35,-36,-37,-38,-39,-40,-41,-42,-43,-44,
C    &10*-44,-43,-42,-41,-40,-39,-38,-37,-36,-35,-34,-33,-32,-31,-30,
C    &-29,-28,-27,-26,-25,-24,-23,-22,-21,2*-20,5*-19,2*-20,-21,-22,
C    &-23,-24,-25,-26,-27,-28,-29,-30,-31,-32,-33,-34,-35,-36,-37,-38,
C    &-39,8*-40,-39,-38,-37,-36,-35,-34,-33,-32,-31,-30,19*-30,-31,
C    &-32,-33,-34,-35,-36,-37,-38,-39,-40,-41,-42,-43,-44,-45,-46,
C    &-47,-48,-49,-50,-51,-52,-53,-54,-55,-56,-57,-58,-59,-60,-61,-62,
C    &-63,-64,-65,-66,-67,-68,-68/
C     DATA (IX(I),I=517,815)/-69,9*-70,-71,-72,-73,-74,-75,-76,-77,
C    &-78,-79,-80,-81,-82,-83,-84,-85,-86,-87,-88,-89,9*-89,-90,-91,
C    &-92,-93,-94,-95,-96,-97,11*-97,-96,-95,-94,-93,-92,-91,-90,
C    &-89,9*-89,-88,-87,-86,-85,-84,-83,-82,-81,-80,-79,-78,-77,-76,
C    &-75,-74,-73,-72,-71,-70,-69,-68,-67,-66,-65,-64,-63,-62,-61,
C    &-60,-59,-58,-57,-56,-55,-54,-53,-52,-51,-50,-49,-48,-47,-46,-45,
C    &-44,-43,-42,-41,-40,-39,-38,-37,-36,-35,-34,-33,-32,-31,-30,-29,
C    &-28,-27,-26,-25,-24,-23,-22,-21,-20,-20,9*-19,2*-20,-21,-22,-23,
C    &-24,-25,-26,-27,-28,-29,-29,6*-30,2*-29,-28,-27,-26,-25,-24,
C    &-23,-22,-21,-20,9*-19,-18,-17,-16,-15,-14,-13,-12,-11,-10,9*-10
C    &-9,-8,-7,-6,-5,-4,-3,-2,-2,-1,0,10*1,2,3,4,5,6,7,8,9,10,10,10*1
C    &12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
C    &9*31,2*30,29,28,27,26,25,24,23,22,21,20,8*20/
C     DATA (IY(I),I=1,216)/71*-9,-8,-7,-6,-5,-4,-3,-2,-1,0,8*0,1,2,3,
C    &4,5,6,7,8,9,10,8*11,12,13,14,15,16,17,18,19,20,21,20*21,22,23,
C    &24,25,26,27,28,29,30,41*30,31,32,33,34,35,36,37,38,39,40,41,42,
C    &43,44,45,46,47,48,49,50,10*50/
C     DATA (IY(I),I=217,421)/51,52,53,54,55,56,57,58,59,59,60,61,
C    &18*61,62,63,64,65,66,67,68,69,70,71,3*72,7*73,4*74,3*75,76,2*77
C    &78,79,80,81,82,83,84,85,86,87,88,89,90,90*90,89,88,87,86,85,84,
C    &83,82,81,80,79,78,77,3*76,6*75,4*74,4*73,2*72,71,70,69,68,67,66
C    &65,64,63,62/
```

```
C        DATA (IY(I),I=422,660)/3*61,15*60,59,58,57,56,55,54,53,52,51,51,
C       &9*50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,34,33,32,31,
C       &38*31,2*30,29,28,27,26,25,24,23,22,21,15*21,4*20,19,18,17,16,15,
C       &14,13,12,11,8*11,10,9,8,7,6,5,4,3,2,1,0,8*0,-1,-2,-3,-4,-5,-6,-7,
C       &-8,-9,69*-10,-9/
C        DATA (IY(I),I=661,815)/-8,-7,-6,-5,-4,-3,-2,-1,0,0,1,9*1,2,3,4,5,
C       &6,7,8,9,10,9*11,12,13,14,15,16,17,18,19,20,20,6*21,2*20,19,18,
C       &17,16,15,14,13,12,11,8*10,3*11,12,13,14,15,16,17,18,19,20,21,9*21,
C       &20,19,18,17,16,15,14,13,12,11,10,20*10,9,8,7,6,5,4,3,2,1,1,0,
C       &10*0,-1,-2,-3,-4,-5,-6,-7,-8/
CC       DATA(IX(I),I=1,210)/71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,
CC      &86,87,88,89,90,91,92,2*93,94,3*95,2*96,2*97,2*98,2*99,2*100,2*101
CC      &,2*102
CC      &,103,2*104,2*105,106,2*107,3*108,109,107,106,2*105,104,103,102,
CC      &101,100,99,2*98,97,2*96,95,94,93,92,91,90,89,88,87,86,85,84,83,82
CC      &,81,80,79,78,77,76,75,2*74,2*75,2*76,2*77,2*78,2*79,80,2*81,2*82,
CC      &2*83,2*84,2*85,2*86,2*87,2*88,2*89,2*90,2*91,2*92,2*93,2*94,2*95,
CC      &2*96,3*97,2*98,2*99,2*100,3*101,2*102,2*103,2*104,2*105,2*106,2*
CC      &107,2*108,2*109,2*110,2*111,2*112,2*113,114,115,116,117,118,119,
CC      &120,121,122,123,124,125,126,127,128,129,2*130,131,132,133,134,135
CC      &,136,137,138,139,140,141,142,143,144,145,146,147,148,149,2*150/
CC       DATA(IX(I),I=211,420)/2*151,2*152,2*153,2*154,2*155,2*156,2*157,2
CC      &*158,157,156,155,154,153,152,151,150,149,148,147,146,145,144,143,
CC      &142,141,140,139,138,137,136,135,134,133,2*132,131,130,129,128,127
CC      &,126,125,124,2*123,122,121,120,119,118,117,2*116,115,114,113,112,
CC      &2*111,110,109,108,107,106,105,104,103,2*102,101,100,99,98,97,96,
CC      &2*95,94,93,92,91,2*90,89,88,87,86,85,84,83,82,81,80,79,78,77,76,
CC      &75,74,73,2*72,71,70,3*69,2*68,2*67,2*66,2*65,2*64,2*63,2*62,2*61,
CC      &2*60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,
CC      &81,82,83,84,85,86,87,88,89,90,91,92,93,94,3*95,94,2*93,2*92,2*91,
CC      &90,2*89,2*88,2*87,2*86,85,2*84,2*83,82,2*81,2*80,2*79,78,2*77,2*
CC      &76,2*75,74,2*73,2*72/
CC       DATA(IX(I),I=421,628)/2*71,2*70,2*69,2*68,2*67,2*66,2*65,2*64,2*
CC      &63,2*62,2*61,60,2*59,2*58,2*57,56,55,54,53,52,51,50,49,48,47,46,
CC      &45,44,43,42,41,40,39,38,37,36,35,34,2*33,32,31,30,29,28,27,26,25,
CC      &24,23,22,21,20,2*19,2*18,2*17,16,2*15,2*14,2*13,12,2*11,2*10,2*9,
CC      &2*8,2*7,2*6,2*5,2*4,3,2,3*1,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
CC      &,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,2*32,33,34,35,36
CC      &,2*37,2*38,39,2*40,41,2*42,2*43,44,2*45,2*46,47,48,49,50,51,52,53
CC      &,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,
CC      &75,76,77,78,79,3*80,2*79,2*78,2*77,76,2*75,2*74,2*73,72,71/
CC       DATA(IY(I),I=1,210)/11,10,2*9,2*8,2*7,2*6,5,2*4,2*3,2,2*1,2*2,5,
CC      &6,8,2*9,10,11,12,13,14,15,16,17,18,19,2*20,21,22,23,24,25,26,27,
CC      &28,29,30,31,32,33,34,35,36,35,36,2*37,2*38,2*39,2*40,2*41,3*42,2*
CC      &43,2*44,2*45,46,2*47,2*48,2*49,50,2*51,2*52,2*53,2*54,55,56,57,58
CC      &,59,2*60,61,62,63,64,65,66,67,68,69,70,71,2*72,73,74,75,76,77,78,
CC      &79,80,2*81,82,83,84,85,86,87,2*88,89,90,91,92,93,94,2*95,96,2*97,
CC      &98,99,100,101,2*102,103,2*104,105,106,107,108,109,110,111,112,2*
CC      &113,114,115,2*116,117,118,119,120,121,122,123,124,125,2*124,2*123
CC      &,2*122,2*121,120,2*119,2*118,2*117,2*116,2*115,2*114,2*113,112,2*
CC      &111,2*110,2*109,2*108,107,3*106,107,108,109/
CC       DATA(IY(I),I=211,420)/110,111,112,113,114,2*115,116,117,2*118,119
CC      &,120,121,122,123,2*124,2*125,2*126,127,2*128,2*129,2*130,2*131,
CC      &132,2*133,2*134,2*135,2*136,2*137,2*138,2*139,2*140,2*141,2*142,2
CC      &*143,2*144,2*145,2*146,2*147,2*148,2*149,2*150,2*151,2*152,2*153,
CC      &2*154,2*155,2*156,2*157,2*158,2*159,2*160,2*161,2*162,2*163,2*164
CC      &,2*165,2*166,167,2*168,2*169,2*170,2*171,2*172,173,172,171,170,2*
```

```
CC    &169,168,2*167,166,165,164,163,2*162,161,160,159,158,157,156,2*155
CC    &,2*154,2*153,152,2*151,2*150,149,2*148,2*147,2*146,145,2*144,2*
CC    &143,2*142,141,2*140,2*139,2*138,2*137,2*136,135,134,133,132,131,
CC    &130,129,128,127,126,125,124,123,122,121,120,119,118,117,116,115,
CC    &114,113,112,111,110,109,108,107,106,105,104,103,102,101,100,99,98
CC    &,97,96,95,94/
CC     DATA(IY(I),I=421,628)/93,92,91,90,89,2*88,87,86,85,84,83,82,2*81,
CC    &80,79,78,77,76,75,74,73,72,71,70,69,68,67,2*66,2*67,2*68,2*69,70,
CC    &2*71,2*72,2*73,74,2*75,2*76,2*77,2*78,2*79,2*80,2*81,2*82,83,2*84
CC    &,3*85,84,83,82,81,80,79,78,77,76,75,74,73,72,71,70,69,68,67,66,65
CC    &,64,2*63,62,61,60,59,58,57,56,55,54,53,52,51,2*50,2*49,2*48,2*47,
CC    &46,2*45,44,2*43,2*42,2*41,2*40,39,2*38,2*37,2*36,2*35,2*34,2*33,2
CC    &*32,2*31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,2*47,48,2*
CC    &47,46,2*45,2*44,2*43,42,2*41,2*40,2*39,2*38,37,2*36,2*35,2*34,2*
CC    &33,2*32,31,2*30,2*29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14
CC    &,13,12/
       DATA(IX(I),I=1,213)/0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
      &18,19,11*20,21,22,23,24,25,26,
      &27,28,29,30,31,32,33,34,35,36,37,38,39,11*40,41,42,43,44,45,46,47
      &,48,49,21*50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,34,33,
      &32,31,41*30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48
      &,49,11*50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,34,33,32,
      &31,30,29,28,27,26,25,24,23,22,21,20,19,18/
       DATA(IX(I),I=214,340)/17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,
      &11*0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,41*20,19,18,
      &17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,20*0/
       DATA(IY(I),I=1,213)/21*0,1,2,3,4,5,6,7,8,9,21*10,9,8,7,6,5,4,3,2,
      &1,11*0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,21*20,21,
      &22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,
      &43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,21*60,61,62,
      &63,64,65,66,67,68,69,33*70/
       DATA(IY(I),I=214,340)/18*70,69,68,67,66,65,64,63,62,61,21*60,59,
      &58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,
      &37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,22,21,21*20,19,18,
      &17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1/
       NEN=340
       DO 1 I=1,NEN
       WRITE(6,*)'ENTRY: ',I,'  X: ',IX(I),'  Y: ',IY(I)
       X(I)=FLOAT(IX(I))
1      Y(I)=FLOAT(IY(I))
       DATA XMAX,YMAX,XMIN,YMIN/2*-1000000.,2*1000000./
       DATA NXA,NXI,NYA,NYI/4*0/
       DO 11 I=1,2
       ISXA(I)=1
       ISXI(I)=1
       ISYA(I)=1
       ISYI(I)=1
11     CONTINUE
       DO 100 I=2,NEN
       IF(X(I)-XMAX)19,10,20
10     IF(NXA-1)2,3,4
4      IF(Y(I).LT.Y(ISXA(1)))GO TO 5
       IF(Y(I).GT.Y(ISXA(2)))GO TO 6
       GO TO 19
5      ISXA(1)=I
       GO TO 19
6      ISXA(2)=I
```

```
      GO TO 19
3     IF(Y(I)-Y(ISXA(1)))7,7,8
7     ISXA(2)=ISXA(1)
      ISXA(1)=I
      GO TO 9
8     ISXA(2)=I
9     NXA=2
      GO TO 19
2     ISXA(1)=I
      NXA=1
      GO TO 19
20    XMAX=X(I)
      ISXA(1)=I
      NXA=1
19    IF(XMIN-X(I))39,30,40
30    IF(NXI-1)22,23,24
24    IF(Y(I).LT.Y(ISXI(1)))GO TO 25
      IF(Y(I).GT.Y(ISXI(2)))GO TO 26
      GO TO 39
25    ISXI(1)=I
      GO TO 39
26    ISXI(2)=I
      GO TO 39
23    IF(Y(I)-Y(ISXI(1)))27,27,28
27    ISXI(2)=ISXI(1)
      ISXI(1)=I
      GO TO 29
28    ISXI(2)=I
29    NXI=2
      GO TO 39
22    ISXI(1)=I
      NXI=2
      GO TO 39
40    XMIN=X(I)
      ISXI(1)=I
      NXI=1
39    IF(Y(I)-YMAX)59,50,60
50    IF(NYA-1)42,43,44
44    IF(X(I).LT.X(ISYA(1)))GO TO 45
      IF(X(I).GT.X(ISYA(2)))GO TO 46
      GO TO 59
45    ISYA(1)=I
      GO TO 59
46    ISYA(2)=I
      GO TO 59
43    IF(X(I)-X(ISYA(1)))47,47,48
47    ISYA(2)=ISYA(1)
      ISYA(1)=I
      GO TO 49
48    ISYA(2)=I
49    NYA=2
      GO TO 59
42    ISYA(1)=I
      NYA=1
      GO TO 59
60    YMAX=Y(I)
      ISYA(1)=I
```

```
      NYA=1
59    IF(YMIN-Y(I))79,70,80
70    IF(NYI-1)62,63,64
64    IF(X(I).LT.X(ISYI(1)))GO TO 65
      IF(X(I).GT.X(ISYI(2)))GO TO 66
      GO TO 79
65    ISYI(1)=I
      GO TO 79
66    ISYI(2)=I
      GO TO 79
63    IF(X(1)-X(ISYI(1)))67,67,68
67    ISYI(2)=ISYI(1)
      ISYI(1)=I
      GO TO 69
68    ISYI(2)=I
69    NYI=2
      GO TO 79
62    ISYI(1)=I
      NYI=2
      GO TO 79
80    YMIN=Y(I)
      ISYI(1)=I
      NYI=1
79    CONTINUE
100   CONTINUE
C     WRITE(6,*)'XMAX,XMIN,YMAX,YMIN: ',XMAX,XMIN,YMAX,YMIN
      NOPO=0
      DO 90 I=1,NXA
      NOPO=NOPO+1
90    IPOLY(NOPO)=ISXA(I)
      IF(ISXA(NXA).EQ.ISYA(NYA))NOPO=NOPO-1
      DO 95 I=1,NYA
      NOPO=NOPO+1
95    IPOLY(NOPO)=ISYA(NYA+1-I)
      IF(ISYA(1).EQ.ISXI(NXI))NOPO=NOPO-1
      DO 102 I=1,NXI
      NOPO=NOPO+1
102   IPOLY(NOPO)=ISXI(NXI+1-I)
      IF(ISXI(1).EQ.ISYI(1))NOPO=NOPO-1
      DO 105 I=1,NYI
      NOPO=NOPO+1
105   IPOLY(NOPO)=ISYI(I)
      IF(ISYI(NYI).EQ.ISXA(1))NOPO=NOPO-1
      DO 110 I=1,NOPO-1
      NX(I)=Y(IPOLY(I))-Y(IPOLY(I+1))
      NY(I)=X(IPOLY(I+1))-X(IPOLY(I))
      C(I)=X(IPOLY(I+1))*Y(IPOLY(I))-X(IPOLY(I))*Y(IPOLY(I+1))
      IF(I.EQ.NOPO-1)GO TO 111
      CALC=NX(I)*X(IPOLY(I+2))+NY(I)*Y(IPOLY(I+2))-C(I)
      GO TO 112
111   CALC=NX(I)*X(IPOLY(1))+NY(I)*Y(IPOLY(1))-C(I)
112   SIDE(I)=SIGN(1.,CALC)
110   CONTINUE
C     WRITE(6,*)'NO.S OF THE CORNERS',(IPOLY(I),I=1,NOPO)
C     DO 115 LL=1,NOPO
C115   WRITE(6,*)X(IPOLY(LL)),'   ',Y(IPOLY(LL))
      NX(NOPO)=Y(IPOLY(NOPO))-Y(IPOLY(1))
```

```
      NY(NOPO)=X(IPOLY(1))-X(IPOLY(NOPO))
      C(NOPO)=X(IPOLY(1))*Y(IPOLY(NOPO))-X(IPOLY(NOPO))*Y(IPOLY(1))
      CALC=NX(NOPO)*X(IPOLY(2))+NY(NOPO)*Y(IPOLY(2))-C(NOPO)
      SIDE(NOPO)=SIGN(1.,CALC)
 ********************************************************************
      NU=0
      DO 120 I=1,NEN
      DO 130 KK=1,NOPO
      IF(I.NE.IPOLY(KK))GO TO 130
      NU=NU+1
      ITU(NU)=I
      GO TO 120
30    CONTINUE
      DO 140 KK=1,NOPO
      CALC=NX(KK)*X(I)+NY(KK)*Y(I)-C(KK)
      IF(SIGN(1.,CALC).EQ.SIDE(KK))GO TO 140
      NU=NU+1
21    ITU(NU)=I
      GO TO 120
40    CONTINUE
20    CONTINUE
      WRITE(6,*)'NO OF PNT.S OUTSIDE:    ',NU
      DO 125 JJ=1,NU
125    WRITE(6,*)ITU(JJ)
 ********************************************************************
                  FIND THE CENTER OF THE SMALLEST CIRCLE
 ********************************************************************
      STEP=5.
      XC=(XMAX+XMIN)/2.
      YC=(YMAX+YMIN)/2.
      RMAX=(XMAX-XMIN+YMAX-YMIN)
      EPS=1.
45    LA=0
      RO=0.
      DO 150 I=1,NU
      RS=SQRT((X(ITU(I))-XC)**2.+(Y(ITU(I))-YC)**2.)
      IF(RO-RS)151,152,153
51    RO=RS
      LA=1
      LIR(LA)=ITU(I)
      IF(I.EQ.1)GO TO 150
      DO 154 J=1,I-1
      ZR=SQRT((X(ITU(J))-XC)**2.+(Y(ITU(J))-YC)**2.)
      IF((RO-ZR).GT.EPS)GO TO 154
      LA=LA+1
      LIR(LA)=ITU(J)
54    CONTINUE
      GO TO 150
52    LA=LA+1
      LIR(LA)=ITU(I)
      GO TO 150
53    IF((RO-RS).GT.EPS)GO TO 150
      LA=LA+1
      LIR(LA)=ITU(I)
50    CONTINUE
      WRITE(6,*)'XC: ',XC,'   ','YC: ',YC
      WRITE(6,*)'R FOR THIS CENTER:    ',RO,'    ','MAX R:  ',RMAX
```

```
C       WRITE(6,*)'NO OF TOUCHING PNT.S:    ',LA
        IF(RO.LE.RMAX)GO TO 159
        GO TO 156
156     XC=(XC+XC1)/2.
        YC=(YC+YC1)/2.
C       WRITE(6,*)'STEP TOO LARGE-NEW CENTER','XC:',XC,'YC:',YC
        GO TO 145
C ************************************************************************
159     RMAX=RO
158     DO 160 I=1,LA
        A(LIR(I))=ATAN3((X(LIR(I))-XC),(Y(LIR(I))-YC))
        IF(I.NE.1)GO TO 161
        NUM=1
        K(NUM)=LIR(I)
        GO TO 160
161     DO 170 J=1,NUM
        IF(A(LIR(I)).GE.A(K(J)))GO TO 170
        DO 180 L=0,NUM-J
180     K(NUM+1-L)=K(NUM-L)
        K(J)=LIR(I)
        NUM=NUM+1
        GO TO 160
170     CONTINUE
        NUM=NUM+1
        K(NUM)=LIR(I)
160     CONTINUE
C       WRITE(6,*)'ANGLES WITH INCREASING ORDER'
C       WRITE(6,*)'NO:  ','I:   ','ANGL:'
C       DO 165 I=1,LA
C165     WRITE(6,*)I,'   ',K(I),'   ',A(K(I))
C ************************************************************************
        IF(LA.NE.1)GO TO 189
        XC1=XC
        YC1=YC
        VSIZE=SQRT((X(K(1))-XC)**2.+(Y(K(1))-YC)**2.)
        XC=XC+STEP*(X(K(1))-XC)/VSIZE
        YC=YC+STEP*(Y(K(1))-YC)/VSIZE
        GO TO 145
189     DO 190 I=1,LA-1
        IF(A(K(I+1))-A(K(I)).LE.3.14159)GO TO 190
        XC1=XC
        YC1=YC
        XX=X(K(I+1))+X(K(I))-2.*XC
        YY=Y(K(I+1))+Y(K(I))-2.*YC
        XC=XC+STEP*XX/SQRT(XX**2.+YY**2.)
        YC=YC+STEP*YY/SQRT(XX**2.+YY**2.)
        GO TO 200
190     CONTINUE
        IF((A(K(1))+2.*3.14159-A(K(LA))).LE.3.14159)GO TO 210
        XC1=XC
        YC1=YC
        XX=X(K(1))+X(K(LA))-2.*XC
        YY=Y(K(1))+Y(K(LA))-2.*YC
        XC=XC+STEP*XX/SQRT(XX**2.+YY**2.)
        YC=YC+STEP*YY/SQRT(XX**2.+YY**2.)
C       WRITE(6,*)'ANGL GT PI-SHIFT'
200     GO TO 145
```

```
C ******************************************************************
210    WRITE(6,*)'CENTER COORDINATES:  ',XC,'   ',YC
       DO 230 I=1,NEN
       X(I)=X(I)-XC
       Y(I)=Y(I)-YC
       R(I)=SQRT(X(I)**2.+Y(I)**2.)
       TH(I)=ATAN3(X(I),Y(I))
230    CONTINUE
C ******************************************************************
C                          SEGMENTATION ALGORITHM
C ******************************************************************
       DATA(ICHECK(I),I=1,340)/340*0/,(INUM(I),I=1,100)/100*0/
       CONTOL=4.*1.01*2.**0.5
       NOS=1
       NOSS=LIR(1)
248    INOW=NOSS
       INUM(NOS)=INUM(NOS)+1
       ISEG(NOS,INUM(NOS))=INOW
C
249    E=1.01*2.**0.5
247    NNE=0
C      WRITE(6,*)'INOW:  ',INOW,' SEG: ',NOS,' ENTRY: ',INUM(NOS),
C     &'  X: ',IX(INOW),'  Y: ',IY(INOW)
       DO 250 I=1,NEN
       IF(ICHECK(I).EQ.1)GO TO 250
       DIS=((X(I)-X(INOW))**2.+(Y(I)-Y(INOW))**2.)**0.5
       IF(DIS.GT.E)GO TO 250
       NNE=NNE+1
       INEAR(NNE)=I
250    CONTINUE
C      WRITE(6,*)NNE,' NEAREST PNT.S'
C
       IF(NNE.NE.0)GO TO 251
C      WRITE(6,*)'NO NEAREST PNT. WITH E: ',E,' INCREASE E'
       E=E+1.01*2.**0.5
       IF(E.GT.RMAX)GO TO 300
       GO TO 247
C
251    IF(INOW.NE.LIR(1))GO TO 280
       NOP=0
       RM=-1.
C      WRITE(6,*)'NEAREST PNT.S:'
       DO 255 J=1,NNE
       CALL FNDDIR(TH(INOW),TH(INEAR(J)),IR)
C      WRITE(6,*)'                  INEAR: ',INEAR(J),' IR: ',IR,' X: '
C     &,IX(INEAR(J)),' Y: ',IY(INEAR(J))
       IF(IR.LE.0)GO TO 255
       NOP=NOP+1
       IF(R(INEAR(J)).LE.RM)GO TO 255
       RM=R(INEAR(J))
       INEXT=INEAR(J)
255    CONTINUE
       ILOCP=1
C
       IF(NOP.NE.0)GO TO 260
C
       RMI=1000.
```

```
      DO 256 J=1,NNE
      CALL FNDDIR(TH(INOW),TH(INEAR(J)),IR)
      IF(IR.EQ.0)GO TO 256
      NOP=NOP+1
      IF(R(INEAR(J)).GE.RMI)GO TO 256
      RMI=R(INEAR(J))
      INEXT=INEAR(J)
256   CONTINUE
C
      IF(NOP.NE.0)GO TO 260
C
      DO 257 J=1,NNE
      IF(R(INEAR(J)).LE.RM)GO TO 257
      RM=R(INEAR(J))
      INEXT=INEAR(J)
257   CONTINUE
      NOSS=INEXT
      ICHECK(INEXT)=1
      GO TO 248
C
260   ITURN(NOS)=ILOOP
      GO TO 270
C
280   NOP=0
      RN=100000000.
      RX=-1.
      DO 258 J=1,NNE
      CALL FNDDIR(TH(INOW),TH(INEAR(J)),IR)
C     WRITE(6,*)'                 INEAR: ',INEAR(J),' IR: ',IR,' X: '
C    &,IX(INEAR(J)),' Y: ',IY(INEAR(J))
      IF(IR.EQ.(-1*ITURN(NOS)))GO TO 258
      IF(ITURN(NOS).NE.(-1*ILOOP))GO TO 259
C
      IF(R(INEAR(J)).GE.RN)GO TO 258
      RN=R(INEAR(J))
      INEXT=INEAR(J)
      NOP=NOP+1
      GO TO 258
C
259   IF(R(INEAR(J)).LE.RX)GO TO 258
      RX=R(INEAR(J))
      INEXT=INEAR(J)
      NOP=NOP+1
258   CONTINUE
C
      IF(NOP.NE.0)GO TO 270
C
261   E=E+1.01*2.**0.5
      IF(E.LE.3.*1.01*2.**0.5)GO TO 247
      NOS=NOS+1
      ITURN(NOS)=ITURN(NOS-1)*(-1)
      INUM(NOS)=0
C     WRITE(6,*)'**********************************'
      GO TO 249
C
270   IF(NOS.EQ.1)GO TO 271
      IF(INUM(NOS).NE.0)GO TO 271
```

```
      IF(ITURN(NOS).NE.ITURN(NOS-1))GO TO 271
      CALL FNDDIR(TH(ISEG(NOS-1,INUM(NOS-1))),TH(INEXT),IR)
      IF(IR.NE.ITURN(NOS))GO TO 272
      NOS=NOS-1
C
271   INUM(NOS)=INUM(NOS)+1
      ISEG(NOS,INUM(NOS))=INEXT
      IF(INEXT.EQ.LIR(1))GO TO 300
      ICHECK(INEXT)=1
      INOW=INEXT
      GO TO 249
C
272   IF(INEXT.EQ.LIR(1))GO TO 300
      ICHECK(INEXT)=1
      INOW=INEXT
      GO TO 249
C
300   IF(ISEG(1,1).NE.ISEG(NOS,INUM(NOS)))GO TO 304
      IF(ITURN(1).NE.ITURN(NOS))GO TO 304
      DO 301 I=1,INUM(1)
      B(I)=ISEG(1,I)
301   CONTINUE
      DO 302 I=1,INUM(NOS)
      ISEG(1,I)=ISEG(NOS,I)
302   CONTINUE
      DO 303 I=1,INUM(1)
      ISEG(1,INUM(NOS)-1+I)=B(I)
303   CONTINUE
      NOS=NOS-1
304   CONTINUE
C
      DO 350 I=1,NOS
      DO 351 J=1,INUM(I)
      WRITE(6,*)'PNT. NO: ',ISEG(I,J),' SEG: ',I,' ENTRY: ',J
351   CONTINUE
      WRITE(6,*)'*********************************************'
350   CONTINUE
355   CONTINUE
C
      WRITE(8,305)(R(I),I=1,NEN),(TH(I),I=1,NEN),RMAX
      WRITE(8,310) NOS,ILOOP
      WRITE(8,306)(INUM(I),I=1,NOS)
      WRITE(8,307)(ITURN(I),I=1,NOS)
      WRITE(8,309)((ISEG(I,J),J=1,INUM(I)),I=1,NOS)
305   FORMAT(8F11.6)
306   FORMAT(8I10)
307   FORMAT(8I10)
309   FORMAT(8I10)
310   FORMAT(I5)
C*****************************************************************
C304   AMAX=0.
C     DO 320 I=1,NOS
C     IF(ITURN(I).EQ.1)GO TO 325
C     DO 330 J=1,INT(INUM(I)/2)
C     MIS=ISEG(I,J)
C     ISEG(I,J)=ISEG(I,INUM(I)+1-J)
C     ISEG(I,INUM(I)+1-J)=MIS
```

```
  FUNCTION ATAN3        74/176  OPT=0,ROUND= A/ S/ M/-D,-DS        FTN 5.1+577
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-F
 FTN5,I=PATREC3,B=BIN.


         C      .....................................))
                REAL FUNCTION ATAN3(XXX,YYY)
                IF(XXX.NE.0.)GO TO 1
                IF(YYY)2,3,4
         2      ATAN3=1.5*3.14159
                GO TO 5
         3      ATAN3=0.
                GO TO 5
         4      ATAN3=0.5*3.14159
                GO TO 5
         1      Z=ATAN2(YYY,XXX)
                Z=((SIGN(1.,Z)+1.)/2.)*Z+((1.-SIGN(1.,Z))/2.)*(Z+2.*3.141
                ATAN3=Z
         5      CONTINUE
                RETURN
                END

LE MAP--(LO=A)
--ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

      62B                               REAL
       1   DUMMY-ARG                    REAL
       2   DUMMY-ARG                    REAL
      63B                               REAL

URES--(LO=A)
-----TYPE--------ARGS------CLASS-----

     GENERIC          2       INTRINSIC
     GENERIC          2       INTRINSIC

ENT LABELS--(LO=A)
-ADDRESS-----PROPERTIES----DEF

     27B                     11
  INACTIVE                    5
     21B                      7
     24B                      9
     46B                     14

POINTS--(LO=A)
--ADDRESS--ARGS---

       6B      2
```

```
SUBROUTINE FNDDIR      74/176  OPT=0,ROUND= A/ S/ M/-C,-DS      FTN 5.1+57
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-
FTN5,I=PATREC3,B=BIN.

                        SUBROUTINE FNDDIR(THNOW,THNEXT,IYON)
                        IF(THNEXT-THNOW)311,312,313
              311       IF((THNOW-THNEXT).GT.3.14159)GO TO 314
                        IYON=-1
                        GO TO 360
              314       IYON=1
                        GO TO 360
              312       IYON=0
                        GO TO 360
              313       IF((THNEXT-THNOW).GT.3.14159)GO TO 315
                        IYON=1
                        GO TO 360
              315       IYON=-1
              360       CONTINUE
                        RETURN
                        END

ABLE MAP--(LO=A)
---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

        3   DUMMY-ARG                     INTEGER
XT      2   DUMMY-ARG                     REAL
N       1   DUMMY-ARG                     REAL

EMENT LABELS--(LO=A)
L-ADDRESS-----PROPERTIES----DEF       -LABEL-ADDRESS-----PROPERTIES----DEF

1  INACTIVE                    3          314      22B                    6
2      25B                     8          315      37B                   13
3      30B                    10          360      41B                   14

Y POINTS--(LO=A)
---ADDRESS--ARGS---

IR       5B     3

ESTICS--

RAM-UNIT LENGTH                    52B =   42
STORAGE USED                    61700B =   25536
ILE TIME                        0.127 SECONDS
 1   X: 0   Y: 0
 2   X: 1   Y: 0
 3   X: 2   Y: 0
 4   X: 3   Y: 0
 5   X: 4   Y: 0
 6   X: 5   Y: 0
 7   X: 6   Y: 0
 8   X: 7   Y: 0
 9   X: 8   Y: 0
 10   X: 9   Y: 0
 11   X: 10   Y: 0
 12   X: 11   Y: 0
```

```
PROGRAM MAIN2        74/176  CPT=0,ROUND= A/ S/ M/-D,-DS        FTN 5.1+577
=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-
5,I=PATREC6,B=BIN.

      PROGRAM MAIN2(INPUT,OUTPUT,TRANS,VECTO1,VECTO2,DS1,DS2,
     &TAPE6=OUTPUT,TAPE8=TRANS,TAPE9=VECTO1,TAPE10=VECTO2,TAPE11=DS
     &TAPE12=DS2)
C
      COMPLEX H(65)
      DIMENSION KAT(50),ILONG(10),ILONG1(10),REPVEC(1000)
      DIMENSION R(900),TH(900),INUM(50),ITURN(50),ANDUR(50)
      DIMENSION ISEG(50,500),STANG(50),DS(50,600),DD(128)
      EQUIVALENCE (DD,H)
      READ(8,305)(R(I),I=1,340),(TH(I),I=1,340),RMAX
      READ(8,310) NOS,ILOOP
      READ(8,306)(INUM(I),I=1,NOS)
      READ(8,307)(ITURN(I),I=1,NOS)
      READ(8,309)((ISEG(I,J),J=1,INUM(I)),I=1,NOS)
305   FORMAT(8F11.6)
306   FORMAT(8I10)
307   FORMAT(8I10)
309   FORMAT(8I10)
310   FORMAT(I5)
C*******************************************************************
304   AMAX=0.
      NB2=7
      IOR=3
      DO 320 I=1,NOS
C     DO 510 III=1,INUM(I)
C510   WRITE(6,*)'NO: ',ISEG(I,III),' TH: ',TH(ISEG(I,III))
C
      IF(ITURN(I).EQ.1)GO TO 325
      DO 330 J=1,INT(INUM(I)/2)
      MIS=ISEG(I,J)
      ISEG(I,J)=ISEG(I,INUM(I)+1-J)
      ISEG(I,INUM(I)+1-J)=MIS
330   CONTINUE
C
325   STANG(I)=TH(ISEG(I,1))
      ROT=0.
      DO 336 J=2,INUM(I)
      TH(ISEG(I,J))=TH(ISEG(I,J))+ROT*2.*3.14159
      IF(TH(ISEG(I,J)).GE.TH(ISEG(I,J-1)))GO TO 336
      ROT=ROT+1.
      TH(ISEG(I,J))=TH(ISEG(I,J))+2.*3.14159
336   CONTINUE
C
      ANDUR(I)=TH(ISEG(I,INUM(I)))-STANG(I)
      KAT(I)=INT(ANDUR(I)/(2.*3.14159))+1
      IF(ANDUR(I).GT.AMAX)AMAX=ANDUR(I)
      DELTA=FLOAT(KAT(I))*2.*3.14159/(2.**NB2-1.)
C
      MEX=0
      ILAS=INT(TH(ISEG(I,INUM(I)))/DELTA)+1
      DO 335 J=1,ILAS
      IF((FLOAT(J-1)*DELTA).GE.STANG(I))GO TO 247
      DS(I,J)=0.
      GO TO 335
247   IF(DELTA*FLOAT(J-1)-TH(ISEG(I,MEX+1)))341,342,343
```

```
      341   DS(I,J)=R(ISEG(I,MEX))+(R(ISEG(I,MEX+1))-R(ISEG(I,MEX))
           &(DELTA*FLOAT(J-1)-TH(ISEG(I,MEX)))/(TH(ISEG(I,MEX+1))
           &-TH(ISEG(I,MEX)))
            IF(J.GT.2**NB2)DS(I,J-2**NB2)=DS(I,J)
            GO TO 335
      342   DS(I,J)=R(ISEG(I,MEX+1))
            IF(J.GT.2**NB2)DS(I,J-2**NB2)=DS(I,J)
            MEX=MEX+1
            GO TO 335
      343   MEX=MEX+1
            GO TO 247
      335   CONTINUE
            DO 1000 K=1,INT(2.**NB2)
      1000  WRITE(6,*)'DS(',I,',',K,')= ',DS(I,K)
      C1000  WRITE(11,*)DS(I,K)
      320   CONTINUE
            WRITE(6,*)'MAX ANGULAR SPAN: ',AMAX
      C*****************************************************
            EPR=RMAX/10.
            EPAN=2.*3.14159/2.**NB2
            LONG=0
            DO 370 I=1,NOS
            IF(ANDUR(I).LT.(AMAX-EPAN))GO TO 370
            LONG=LONG+1
            ILONG(LONG)=I
      370   CONTINUE
      C
            ILONG1(1)=ILONG(1)
            IF(LONG.EQ.1)GO TO 400
      C
            I2=0
            I=1
      397   LONG1=0
            RR=0.
            DO 390 J=1,LONG
            DELTA=FLOAT(KAT(ILONG(J)))*2.*3.14159/(2.**NB2-1.)
            JO=INT(STANG(ILONG(J))/DELTA)+1+I
            IF(JO.GT.INT(2.**NB2))JO=JO-INT(2.**NB2)
            D=DS(ILONG(J),JO)
            IF(D.LE.RR)GO TO 390
            RR=D
            KATMAX=KAT(J)
      390   CONTINUE
      394   DO 395 J=1,LONG
            DELTA=FLOAT(KAT(ILONG(J)))*2.*3.14159/(2.**NB2-1.)
            JO=INT(STANG(ILONG(J)+I2)/DELTA)+1+I
            IF(JO.GT.INT(2.**NB2))JO=JO-INT(2.**NB2)
            D=DS(ILONG(J)+I2,JO)
            IF(D.LT.(RR-EPR))GO TO 395
            LONG1=LONG1+1
            ILONG1(LONG1)=ILONG(J)
      395   CONTINUE

            IF(LONG1.EQ.1)GO TO 400

            DO 396 J=1,LONG1
      396   ILONG(J)=ILONG1(J)
```

```
      I=I+1
      DELTA=FLOAT(KAT(ILONG(J)+I2))*2.*3.14159/(2.**NB2-1.)
      IF(I.GT.(INT(ANDUR(ILONG(J)+I2)/DELTA)+1))GO TO 399
      GO TO 397
C
399   I2=I2+1
      IF(I2.GE.NOS)GO TO 400
      I=1
      GO TO 394
C
400   IZ=ILONG1(1)
      ZERO=STANG(IZ)
      WRITE(6,*)'LONGEST SEG: ',IZ,' ANDUR: ',ANDUR(IZ)
C*****************************************************************
420   DO 410 I=1,NOS
      DO 360 III=1,INT(2.**NB2)
360   DD(III)=DS(I,III)
      CALL RFFT(H,64,6,1)
      DO 361 III=1,INT(2.**(NB2-IOR))
361   DS(I,III)=DD(III)
      DO 362 J=1,INT(2.**(NB2-IOR-1))
      STR=SQRT(DS(I,J*2-1)**2.+DS(I,J*2)**2.)
      STH=ATAN3(DS(I,J*2-1),DS(I,J*2))
      STR=STR/(RMAX)
      STH=STH-ZERO/FLOAT(KAT(I))
      DS(I,J*2-1)=STR*COS(STH)
      DS(I,J*2)=STR*SIN(STH)
      WRITE(6,*)'PROCESSED DS: ',DS(I,J)
362   CONTINUE
410   CONTINUE
C*****************************************************************
C            CONSTRUCTION OF THE REPRESENTATION VECTOR
C*****************************************************************
440   REPVEC(1)=NOS
      INDEX=2
      DO 450 I=IZ,NOS
      REPVEC(INDEX)=FLOAT(KAT(I))
450   INDEX=INDEX+1
      IF(IZ.EQ.1)GO TO 452
      DO 451 I=1,IZ-1
      REPVEC(INDEX)=FLOAT(KAT(I))
451   INDEX=INDEX+1

452   DO 460 I=IZ,NOS
      DO 461 J=1,INT(2.**(NB2-IOR))
      REPVEC(INDEX)=DS(I,J)
461   INDEX=INDEX+1
460   CONTINUE
      IF(IZ.EQ.1)GO TO 500
      DO 470 I=1,IZ-1
      DO 471 J=1,INT(2.**(NB2-IOR))
      REPVEC(INDEX)=DS(I,J)
471   INDEX=INDEX+1
470   CONTINUE
500   DO 501 JJ=1,INDEX-1
      WRITE(6,*)' NO: ',JJ,' REPVEC: ',REPVEC(JJ)
      WRITE(10,*)REPVEC(JJ)
```

```
              501   CONTINUE
              502   CONTINUE
                    STOP
                    END
```

```
RIABLE MAP--(LO=A)
IE---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDR

AX     163251B                           REAL                   J       1632
OUR      7344B                           REAL            5C     JJ      1633
       163300B                           REAL                   JO      1632
       163040B        EQV                REAL           128     K       1632
LTA    163261B                           REAL                   KAT       15
        70360B                           REAL         30C0C     KATMAX  1633
AN     163270B                           REAL                   LONG    1632
R      163267B                           REAL                   LONG1   1632
       163040B        EQV                COMPLEX         65     MEX     1632
       163242B                           INTEGER                MIS     1632
I      163307B                           INTEGER                NB2     1632
AS     163263B                           INTEGER                NOS     1632
ONG      1574B                           INTEGER         1C     R         35
ONG1     1606B                           INTEGER         1C     REPVEC    16
OOP    163245B        *S*                INTEGER                RMAX    1632
DEX    163315B                           INTEGER                ROT     1632
UM       7200B                           INTEGER         50     RR      1632
R      163253B                           INTEGER                STANG    702
EG       7426B                           INTEGER      2500C     STH     1633
URN      7262B                           INTEGER         5C     STR     1633
       163304B                           INTEGER                TH        53
       163273B                           INTEGER                ZERO    1633
```

```
OCEDURES--(LO=A)
IE------TYPE--------ARGS------CLASS-----       -NAME------TYPE--------ARGS--

AN3      REAL          2      FUNCTION         RFFT                      4
S        GENERIC       1      INTRINSIC        SIN        GENERIC        1
OAT      REAL          1      INTRINSIC        SQRT       GENERIC        1
T        GENERIC       1      INTRINSIC
```

```
ATEMENT LABELS--(LO=A)
BEL-ADDRESS-----PROPERTIES----DEF       -LABEL-ADDRESS-----PROPERTIES----DE

247     323B                    55          341  INACTIVE                   5
304 *NO REFS*                   21          342    401B                     6
305    1367B     FORMAT         15          343    431B                     6
306    1371B     FORMAT         16          360  INACTIVE     DO-TERM      12
307    1373B     FORMAT         17          361  INACTIVE     DO-TERM      13
309    1375B     FORMAT         18          362  INACTIVE     DO-TERM      14
310    1377B     FORMAT         19          370    520B       DO-TERM       8
320  INACTIVE    DO-TERM        71          390    611B       DO-TERM       9
325     175B                    35          394    616B                     9
330  INACTIVE    DO-TERM        33          395    670B       DO-TERM      10
335     434B     DO-TERM        67          396  INACTIVE     DO-TERM      11
336     240B     DO-TERM        42          397    536B                     8
```

```
FTN5,I=PATREC6,B=BIN.
```

```
C.....................................................................
C                            **  FEATURE EXRACTION FINISHED   **
C.....................................................................
      REAL FUNCTION ATAN3(XXX,YYY)
      IF(XXX.NE.0.)GO TO 1
      IF(YYY)2,3,4
2     ATAN3=1.5*3.14159
      GO TO 5
3     ATAN3=0.
      GO TO 5
4     ATAN3=0.5*3.14159
      GO TO 5
1     Z=ATAN2(YYY,XXX)
      Z=((SIGN(1.,Z)+1.)/2.)*Z+((1.-SIGN(1.,Z))/2.)*(Z+2.*3.1415
      ATAN3=Z
5     CONTINUE
      RETURN
      END
```

```
LE MAP--(LO=A)
-ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

     62B                              REAL
      1    DUMMY-ARG                  REAL
      2    DUMMY-ARG                  REAL
     63B                              REAL
```

```
URES--(LO=A)
----TYPE--------ARGS------CLASS-----

    GENERIC        2      INTRINSIC
    GENERIC        2      INTRINSIC
```

```
ENT LABELS--(LO=A)
ADDRESS-----PROPERTIES----DEF

    27B                 13
 INACTIVE                7
    21B                  9
    24B                 11
    46B                 16
```

```
POINTS--(LO=A)
-ADDRESS--ARGS---

      6B     2
```

```
SUBROUTINE FNDDIR        74/176  OPT=0,ROUND= A/ S/ M/-D,-DS       FTN 5.1+5
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/
FTN5,I=PATREC6,B=BIN.

          C.......................................................................
                SUBROUTINE FNDDIR(THNOW,THNEXT,IYON)
                IF(THNEXT-THNOW)311,312,313
          311   IF((THNOW-THNEXT).GT.3.14159)GO TO 314
                IYON=-1
                GO TO 360
          314   IYON=1
                GO TO 360
          312   IYON=0
                GO TO 360
          313   IF((THNEXT-THNOW).GT.3.14159)GO TO 315
                IYON=1
                GO TO 360
          315   IYON=-1
          360   CONTINUE
                RETURN
                END

IABLE MAP--(LO=A)
E---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

N        3   DUMMY-ARG                  INTEGER
EXT      2   DUMMY-ARG                  REAL
OW       1   DUMMY-ARG                  REAL

TEMENT LABELS--(LO=A)
EL-ADDRESS-----PROPERTIES----DEF        -LABEL-ADDRESS-----PROPERTIES----DEF

11  INACTIVE                   4          314     22B                    7
12     25B                     9          315     37B                    14
13     30B                    11          360     41B                    15

RY POINTS--(LO=A)
E---ADDRESS--ARGS---

DIR      5B      3

TISTICS--

GRAM-UNIT LENGTH              52B =   42
 STORAGE USED              61700B =   25536
PILE TIME                  0.130 SECONDS
```

```
SUBROUTINE FFT          74/176  OPT=0,ROUND= A/ S/ M/-C,-DS
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-S
FTN5,I=PATREC6,B=BIN.

      C...............................................
            SUBROUTINE FFT(X,N,L2N)
            COMPLEX X(N),W,B
            DATA PI/3.141592653589793/
IAL  *      CONSTANT TOO LONG , EXCESS DIGITS TRUNCATED
            NV2=N/2
            NM1=N-1
            L=1
            DO 3 K=1,NM1
            IF(K.GE.L) GO TO 1
            B=X(L)
            X(L)=X(K)
            X(K)=B
          1 M=NV2
          2 IF(M.GE.L) GO TO 3
            L=L-M
            M=M/2
            GO TO 2
          3 L=L+M
            K=1
            DO 5 L=1,L2N
            M=K
            BM=M
            K=2*K
            W=(1.,0.)
            DO 5 J=1,M
            BJ=J
            DO 4 I=J,N,K
            I2=I+M
            B=X(I2)*W
            X(I2)=X(I)-B
          4 X(I)=X(I)+B
            ARG=PI*BJ/BM
          5 W=CMPLX(COS(ARG),-SIN(ARG))
            RETURN
            END

ARIABLE MAP--(LO=A)
AME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

RG       247B                              REAL
         225B                              COMPLEX
J        242B                              REAL
1        237B                              REAL
         243B                              INTEGER
2        246B                              INTEGER
         240B                              INTEGER
         233B                              INTEGER
         232B                              INTEGER
```

```
SUBROUTINE RFFT        74/176  OPT=0,ROUND= A/ S/ M/-C,-DS
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB
FTN5,I=PATREC6,B=BIN.
```

```
      C.................................................
            SUBROUTINE RFFT(H,N,L2N,ID)
            COMPLEX H(1),HK,HNK,XK,YK,W
            DATA PI/3.141592653589793/
IVIAL  *   CONSTANT TOO LONG , EXCESS DIGITS TRUNCATED
            NV2=N/2
            IF(ID.GE.0) GO TO 2
            RO=REAL(H(N+1))/2.
            GO=REAL(H(1))/2.
            H(1)=CMPLX(RO+GO,RO-GO)
            GO TO 4
          2 CALL FFT(H,N,L2N)
            RO=REAL(H(1))
            GO=AIMAG(H(1))
            H(1)=CMPLX(RO+GO,0.)
            H(N+1)=CMPLX(RO-GO,0.)
          4 H(NV2+1)=CONJG(H(NV2+1))
            IF(NV2.LE.1) GO TO 3
            PN=PI/N
            KU=NV2-1
            DO 1 K=1,KU
            HK=H(K+1)/2.
            HNK=CONJG(H(N-K+1))/2.
            XK=HNK+HK
            YK=HNK-HK
            ARG=K*PN
            W=CMPLX(SIN(ARG),COS(ARG))
            W=W*YK
            H(K+1)=XK+W
          1 H(N-K+1)=CONJG(XK-W)
          3 IF(ID.GE.0) RETURN
            CALL FFT(H,N,L2N)
            RETURN
            END
```

VARIABLE MAP--(LO=A)

NAME	ADDRESS	BLOCK	PROPERTIES	TYPE	SIZE
ARG	253B			REAL	
GO	246B			REAL	
H	1		DUMMY-ARG	COMPLEX	1
HK	231B			COMPLEX	
HNK	233B			COMPLEX	
ID	4		DUMMY-ARG	INTEGER	
K	251B			INTEGER	
KU	250B			INTEGER	
L2N	3		DUMMY-ARG	INTEGER	

```
NO:   1 REPVEC: 12.
NO:   2 REPVEC: 1.
NO:   3 REPVEC: 1.
NO:   4 REPVEC: 1.
NO:   5 REPVEC: 1.
NO:   6 REPVEC: 1.
NO:   7 REPVEC: 1.
NO:   8 REPVEC: 1.
NO:   9 REPVEC: 1.
NO:   10 REPVEC: 1.
NO:   11 REPVEC: 1.
NO:   12 REPVEC: 1.
NO:   13 REPVEC: 1.
NO:   14 REPVEC: -1.958008370057
NO:   15 REPVEC: -9.006838951722
NO:   16 REPVEC: .3648857381072
NO:   17 REPVEC: 6.145631320304
NO:   18 REPVEC: .4694988483658
NO:   19 REPVEC: .01577791649182
NO:   20 REPVEC: .03593474484914
NO:   21 REPVEC: -3.549275431105
NO:   22 REPVEC: -.5336162175551
NO:   23 REPVEC: 2.033645201222
NO:   24 REPVEC: .03196323984206
NO:   25 REPVEC: 1.333667515186
NO:   26 REPVEC: .6656439204108
NO:   27 REPVEC: -2.218269644992
NO:   28 REPVEC: -.3686879179193
NO:   29 REPVEC: .2104307581831
NO:   30 REPVEC: -1.322623935308
NO:   31 REPVEC: -6.084070406025
NO:   32 REPVEC: 5.208478379746
NO:   33 REPVEC: 3.215121475653
NO:   34 REPVEC: -5.631768014339
NO:   35 REPVEC: 1.434063413196
NO:   36 REPVEC: 2.651344768989
NO:   37 REPVEC: -4.608689624419
NO:   38 REPVEC: 1.25417248433
NO:   39 REPVEC: 4.495751794596
NO:   40 REPVEC: -3.398557869082
NO:   41 REPVEC: -1.918295467634
NO:   42 REPVEC: 2.970712146902
NO:   43 REPVEC: -.7717184314462
NO:   44 REPVEC: -1.222340072939
NO:   45 REPVEC: 1.853910482979
```

```
:    47 REPVEC: -6.568611681412
:    48 REPVEC: 5.57501709998
:    49 REPVEC: 3.679375007089
:    50 REPVEC: -6.435490145029
:    51 REPVEC: 1.239223894881
:    52 REPVEC: 3.618009679867
:    53 REPVEC: -5.21465228209
:    54 REPVEC: 1.014656569456
:    55 REPVEC: 5.978267118979
:    56 REPVEC: -4.62448267108
:    57 REPVEC: -3.350638018 12
:    58 REPVEC: 5.236159603281
]:   59 REPVEC: -.7915823892728
]:   60 REPVEC: -2.889859823472
]:   61 REPVEC: 3.867200924969
]:   62 REPVEC: -.1542688099567
]:   63 REPVEC: -.7096365612133
]:   64 REPVEC: .7260974454854
]:   65 REPVEC: -.01286135278937
]:   66 REPVEC: -.1290365661497
]:   67 REPVEC: .714655496958
]:   68 REPVEC: -.6757498067337
]:   69 REPVEC: -.2659795357455
]:   70 REPVEC: .3927043029658
]:   71 REPVEC: -.6108733462578
]:   72 REPVEC: .5225227772776
]:   73 REPVEC: .5043340777324
]:   74 REPVEC: -.5965803790567
]:   75 REPVEC: .4140951171658
]:   76 REPVEC: -.2897513373444
]:   77 REPVEC: -.6659032041094
]:   78 REPVEC: -1.810346840926
]:   79 REPVEC: -8.327595683824
0:   80 REPVEC: 7.709376979242
0:   81 REPVEC: -3.216007536866
0:   82 REPVEC: 4.285778446954
0:   83 REPVEC: 6.587124651967
0:   84 REPVEC: -5.129090042867
0:   85 REPVEC: 4.86988324216
0:   86 REPVEC: -4.900224775832
0:   87 REPVEC: -3.548198207029
0:   88 REPVEC: 2.063579111381
0:   89 REPVEC: -4.400401976142
0:   90 REPVEC: 3.478167107903
0:   91 REPVEC: .8621150045215
0:   92 REPVEC: -.06747855313102
0:   93 REPVEC: 2.302464209152
0:   94 REPVEC: -.336363452578
0:   95 REPVEC: -1.547271959071
0:   96 REPVEC: 1.278989718691
0:   97 REPVEC: -.9326682078432
0:   98 REPVEC: 1.368741097991
0:   99 REPVEC: .7922862603423
0:   100 REPVEC: -.1713248150029
0:   101 REPVEC: 1.569810385985
0:   102 REPVEC: -1.501988253443
0:   103 REPVEC: .4766502467193
0:   104 REPVEC: -1.040124816554
0:   105 REPVEC: -1.178073575118
0:   106 REPVEC: .6551231899811
0:   107 REPVEC: -1.422730218721
0:   108 REPVEC: 1.559970294432
0:   109 REPVEC: .02425587568999
0:   110 REPVEC: -2.335238580044
     111 REPVEC: -10.74209800426
```

```
116 REPVEC: 5.669660085709
117 REPVEC: 7.58930646951
118 REPVEC: -2.222615347154
119 REPVEC: 8.084505508923
120 REPVEC: -6.574351406933
121 REPVEC: 2.67529586652
122 REPVEC: -4.877680385165
123 REPVEC: -2.926462344427
124 REPVEC: -.3494899473588
125 REPVEC: -4.223446615555
126 REPVEC: -1.367545467587
127 REPVEC: -6.29070946482
128 REPVEC: 4.113469745992
129 REPVEC: -4.787109408285
130 REPVEC: 5.931493407935
131 REPVEC: .3642229808904
132 REPVEC: 2.984414165367
133 REPVEC: 4.448441534931
134 REPVEC: -1.448836158434
135 REPVEC: 4.361095830466
136 REPVEC: -3.48604775798
137 REPVEC: 1.276334139939
138 REPVEC: -2.399877929774
139 REPVEC: -1.384269389916
140 REPVEC: -.422477274541
141 REPVEC: -1.792733807443
142 REPVEC: -1.977615669998
143 REPVEC: -9.097032535951
144 REPVEC: -2.201368047539
145 REPVEC: -5.823249449242
146 REPVEC: -.6631832522232
147 REPVEC: .2772033001932
148 REPVEC: 1.049188927632
149 REPVEC: 3.468267177922
150 REPVEC: 1.178359077541
151 REPVEC: 1.80769199339
152 REPVEC: -.1125645819703
153 REPVEC: -1.410193052008
154 REPVEC: -1.048954913042
155 REPVEC: -2.125506780608
156 REPVEC: -.5224799652771
157 REPVEC: -.1046258443227
158 REPVEC: -1.610017035039
159 REPVEC: -7.406078730759
160 REPVEC: -7.056978637887
161 REPVEC: -2.501979315835
162 REPVEC: -5.634489390668
163 REPVEC: 4.508587619519
164 REPVEC: .8021682874916
165 REPVEC: 6.729818215638
166 REPVEC: 5.602124936952
167 REPVEC: 2.63219559205
168 REPVEC: 4.608948081024
169 REPVEC: -2.960798104723
170 REPVEC: -.02892736471722
171 REPVEC: -4.672447696015
172 REPVEC: -3.213624886289
173 REPVEC: -2.041087288126
174 REPVEC: -5.791131941519
175 REPVEC: -26.63920826034
176 REPVEC: -24.05018584096
177 REPVEC: 5.21392699527
178 REPVEC: 3.712170695526
179 REPVEC: 17.13932351256
```

```
NU:    182 REPVEC: -.02289741559325
NU:    183 REPVEC: .1076998955825
NU:    184 REPVEC: 5.41433493382
NU:    185 REPVEC: -1.09408613338б
NU:    186 REPVEC: -1.379619823968
NU:    187 REPVEC: -6.802836802711
NU:    188 REPVEC: -4.830586083808
NU:    189 REPVEC: .9668315775347
NU:    190 REPVEC: -1.707874218271
NU:    191 REPVEC: -7.856221796087
NU:    192 REPVEC: -6.07650846808
NU:    193 REPVEC: 5.086144520987
NU:    194 REPVEC: 7.012750361836
NU:    195 REPVEC: 2.886428230077
NU:    196 REPVEC: -.6620465124008
NU:    197 REPVEC: -7.003985072188
NU:    198 REPVEC: -5.271549417725
NU:    199 REPVEC: 3.462691716673
NU:    200 REPVEC: 4.773237502471
NU:    201 REPVEC: 2.60124635736
NU:    202 REPVEC: .008866367355318
NU:    203 REPVEC: -4.465588113873
NU:    204 REPVEC: -3.011366639948
NU:    205 REPVEC: 1.671352886565
```

```
UGRAM SGN          74/176  OPT=C,RCLNO= A/ S/ M/-C,-DS      FTN 5.1+577
UNG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,CE=-TB/-SB/-SL/ ER/-ID/-PMD/-ST
I=PATREC7,B=BIN.

         PROGRAM SGN(INPUT,OUTPUT,VECTO1,VECTC2,TAPE6=OUTPUT,TAPE9=VECTO
        &TAPE10=VECTC2)
C
         DIMENSION VEC(2,1500)
         NB2=7
         IOR=3
         THRESH=10.
         READ(9,*)VEC(1,1)
         NOS=INT(VEC(1,1))
         LENGTH=1+NOS+NOS*INT(2.**(NB2-IOR))
         DO 2 J=2,LENGTH
         READ(9,*)VEC(1,J)
2        CONTINUE
         READ(10,*)VEC(2,1)
         NOS=INT(VEC(2,1))
         LENGTH=1+NOS+NOS*INT(2.**(NB2-IOR))
         DO 3 J=2,LENGTH
         READ(10,*)VEC(2,J)
3        CONTINUE
C
         IF(VEC(1,1).NE.VEC(2,1))GO TO 50
         DIFFER=0.
         DO 10 I=1,INT(VEC(1,1))
         WRITE(6,*)'VEC1 ',VEC(1,I+1),' VEC2 ',VEC(2,I+1),' DIFF ',DIFFE
         DIFFER=DIFFER+(VEC(1,I+1)-VEC(2,I+1))**2.
10       CONTINUE
         IF(DIFFER.NE.0.)GO TO 50
         DIFFER=0.
         DO 20 I=2+NOS,LENGTH
         FARK=(VEC(1,I)-VEC(2,I))**2.
         DIFFER=DIFFER+FARK
         WRITE(6,*)'VEC1    ',VEC(1,I),'        VEC2 ',VEC(2,I)
20       CONTINUE
         DIFFER=SQRT(DIFFER)/NOS
         IF(DIFFER.GT.THRESH)GO TO 30
         WRITE(6,*)'SAME CLASS,DISTANCE: ',DIFFER,'  THRESHOLD: ',THRESH
         GO TO 60
50       WRITE(6,*)'SHAPES 1 AND 2 ARE FROM DIFFERENT CLASSES'
         GO TO 60
30       WRITE(6,*)'DFRNT CLAS.,DISTANCE: ',DIFFER,' THRESHOLD: ',THRESH
60       CONTINUE
         STOP
         END

?--(LO=A)
ESS--BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS--BLO

57B                             REAL                  LENGTH    6253B
53B                             REAL                  NB2.      6247B
50B                             INTEGER               NOS       6252B
50B                             INTEGER               THRESH    6251B
54B                             INTEGER               VEC        357B
```