To the Graduate Council:

I am submitting herewith a thesis written by Juan Antonio Herrera-Ball entitled "Finding Convex Deficiencies in Linear Time." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

R. C. Gonzalez

R. C. Gonzalez, Major Professor

We have read this thesis
and recommend its acceptance:

Michael G Thomason

Robert E Barlow

Accepted for the Council:

Carl Minkel

The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature 

Date 2/20/85

FINDING CONVEX DEFICIENCIES IN LINEAR TIME

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Juan Antonio Herrera-Ball

March 1985

A mis padres.

ACKNOWLEDGEMENTS

I wish to thank Dr. R. C. Gonzalez for his invaluable guidance and assistance in the research, writing, and proofreading of this thesis. I also thank Dr. R. E. Bodenheimer and Dr. M. G. Thomason for serving in my Master's committee. To Connie Gonzalez and Linda Murdock I express my most sincere gratitude for their assistance in typing this manuscript. Finally, I must thank my wife, Carmen, for her patience, understanding and encouragement throughout the course of this investigation.

This work was supported by Perceptics Co. through a contract with the University of Tennessee.

ABSTRACT

A new algorithm for obtaining the convex deficiencies of an arbitrary 8-connected digital figure in linear time and space is presented. It is shown that the boundary of a digital figure is a closed polygonal curve that belongs to a family of ordered crossing polygons. This family of polygons is best described as polygons such that no interior points lie to the left as the boundary is traversed in the clockwise direction, with no restriction on the possible occurrence of overlapping sides and collinear vertices. All simple polygons are shown to belong to this family. A formal proof of correctness is also included and a convex hull invariant condition is introduced. It is also shown that the convex hull invariant condition presented by S. K. Ghosh and R. K. Shyamasundar is a special case of the one presented here, and that theirs is necessary but not sufficient to guarantee the correctness of their algorithm for obtaining the convex hull of a simple polygon. The latter is proved by means of a counter-example. A comparative evaluation of the convex hull algorithm presented here with reference to other algorithms that have appeared in the recent literature is included. Pattern Recognition applications, in which both the convex hull and the convex deficiency of digital figures are used as shape descriptors, are briefly discussed.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The convex hull of a finite planar set of points is defined as the minimum area convex set containing the original set. The set difference CH(S) - S, where S is a finite set of points and CH(S) is its convex hull, is known as the convex deficiency CD(S). There are many problems in pattern recognition, digital image processing, and computer graphics where the solution involves determining the convex hull of a set of points.

The importance of this subject is demonstrated by the numerous papers that have been devoted to studying the problem in recent years. Some of the applications in which the use of the convex hull has been reported include normalization of patterns in image processing, shape description and polygonal decomposition in syntactic pattern recognition, topological feature extraction, pattern separability, clustering, and triangulation of sets of points [Toussaint, 1980].

In this paper a linear time algorithm for finding the convex deficiency of a digital region is proposed. Along with this algorithm, a new linear time convex hull algorithm is introduced. The latter algorithm is based on a new convex hull invariant condition that guarantees the correctness of the algorithm for arbitrary 8-connected digital regions.

The algorithms presented in this paper were programmed in the FORTRAN 77 language and implemented using a DEC PDP-11/73 minicomputer

with 1.5 megabytes of core memory. A standard closed-circuit television camera was used to input the images used during the course of this investigation to the computer via a Perceptics Corporation IP9200 Image Processor. The IP9200 was equipped with a digitizer or sampling module, a display module and one memory module capable of storing two 512 x 512 8-bit images. Once digitized, the images were stored on a DSD 880 Winchester disk. For the purpose of generating the sample experimental results that accompany this paper, 64 x 64 images (stored on a Winchester disk) were used, and the output, rather than being displayed on a color television monitor by means of the IP9200 display module, were printed on paper using a DEC LA12 line printer.

In section II the notation and the terminology used in this paper are discussed along with some of the background work already reported in this field. The convex deficiency algorithm is developed in section III. Section IV consists of an analysis of the algorithm, which includes a proof of correctness, a discussion of its computational complexity, a comparative evaluation with respect to other linear time convex hull algorithms and a discussion of some experimental results. Finally, some pattern recognition applications for the convex deficiency of digital figures are treated in section V.

II. FOUNDATION

In this section, the notation and the terminology used throughout this paper are given along with a brief overview of the background in this area of computational geometry.

A. Background

Two types of convex hull problems can be identified [Toussaint, 1980]. The first one deals with determining the convex hull of a set of n points in two dimensions represented just by their cartesian coordinates and listed in arbitrary order. The second class of problems consists of computing the convex hull of an n-sided polygon given in standard form, i.e., as a list of vertices along with their cartesian coordinates with the restriction that all its vertices are distinct and that no three consecutive vertices are collinear. Obviously, the second class of problems is just a special case of the first. The first class of convex hull problems will be referred to as the general CH problem while the second class will be identified as the simple CH problem.

Several algorithms have been proposed for solving the general CH problem. Graham [Graham, 1972] proposed a very efficient algorithm that computes the convex hull of a set of n points in order $O(n\log_2 n + c.n)$ where "c" is a small positive constant which depends on what it is meant by an operation. Graham's algorithm is based on a reordering of every point according to its polar coordinates. Koplowitz and Jouppi [Koplowitz, 1978] proposed several ways in which Graham's algorithm could be optimized but the running time of this algorithm is dominated

by the sorting step resulting in an $O(n\log_2 n)$ algorithm. Jarvis [Jarvis, 1973] presented an algorithm of order $O(hn)$ where h is the number of vertices belonging to the convex hull. This turns out to be a very efficient algorithm for some cases but has a worst case running time of order $O(n^2)$. Bykat [Bykat, 1978] proposed an algorithm that he claimed was $O(n\log_2 n)$ but it was later proved by Fournier [Fournier, 1979] and Dévai and Szedrényi [Dévai, 1979] that it was actually an $O(n^2)$ algorithm. All the algorithms that have been mentioned are "off-line", that is, the complete set of points need be known before the algorithm is applied. Preparata [Preparata, 1979] described an algorithm that runs in real time, i.e., updates the convex hull of a set of points everytime a new point is received. The algorithm has an update time of $O(\log_2 n)$, and a total running time of order $O(n\log_2 n)$. Furthermore, it has been shown that the lower bound for solving the general CH problem is $O(n\log_2 n)$ [Toussaint, 1980].

Several O(n) algorithms have been proposed for solving the simple CH problem. Sklansky [Sklansky, 1972] proposed a very simple algorithm for obtaining the convex hull of simple polygons. However, Bykat [Bykat, 1978] showed that Sklansky's algorithm fails for some types of simple polygons. Toussaint and Avis [Toussaint, 1982] proved that Sklansky's algorithm works for a subclass of simple polygons known as weakly externally visible polygons. Another linear time algorithm was proposed by Shamos [Shamos, 1977] but it has been discovered that the algorithm fails for some types of simple polygons [Toussaint, 1980]. McCallum and Avis [McCallum, 1979] proposed an O(n) algorithm that uses two stacks but which is very complicated. Lee [Lee, 1980] and Ghosh and

Shyamasundar [Ghosh, 1983] proposed linear time algorithms that require only one stack. However, a counter example to Ghosh and Shyamasundar's algorithm has been found and is presented in this paper along with a brief description of why their algorithm fails for some simple polygons. Kim and Rosenfeld [Kim, December, 1980] proposed an algorithm for finding the convex hull of a digital region that works only on regions that are not trivially concave (cf. [Kim, December, 1980]).

Although the above mentioned algorithm uses part of Graham's algorithm, it results in a linear time algorithm since the ordering (the step that takes $O(n\log_2 n)$ in Graham's) is not necessary due to the fact that the boundary points of any digital region that is not trivially concave meet Graham's ordering condition. Kim [Kim, October, 1980] proposed an $O(n)$ algorithm for finding the convex hull of an ordered crossing polygon but some counter-examples to this algorithm have been reported (cf. [Toussaint, 1982]). Ghosh and Shyamasundar [Ghosh, 1984] have proposed a linear time algorithm for finding the convex hull of an ordered crossing polygon that uses three stacks but which turns out to be computationally expensive due to its complexity.

It will be shown that the convex hull algorithm presented in this paper finds the convex hull of a subset of ordered crossing polygons that includes simple polygons in linear time and space. Before formally stating the algorithm, some basic definitions and the notation used in this paper will be introduced.

B. Notation

A point will be represented by its x- and y- coordinates, where the x- and y-axes form a right hand coordinate system. A digital image will

be represented by a set of unit square cells or pixels. Each pixel will be associated with a lattice point or pair of integers (h,k) that denote its position in the image.

If S is a set of lattice points then $\overline{S}$ is its set complement, i.e., the set of all lattice points not in S. Let S′ the set of cells associated with the points in S. The set of points in S′ will be denoted by s(S) and the boundary of s(S) by $\partial s(S)$. A digital region R is defined as a finite connected set of lattice points and will be represented by either R or R′. Unless otherwise stated, digital regions in this paper are assumed to be simply 8-connected [Rosenfeld, 1970] and [Gonzalez, 1977]. Also, the terms digital region, digital figure and digitized silhouette are used interchangeably.

Figure 2.1 illustrates a digital region R and shows the sets R′, $\overline{R}$ and $\partial s(R)$. A point of R whose four 4-neighbors [Rosenfeld, 1970] are also in R is called an interior point of R. A boundary point of R is a point which has at least one 4-neighbor in $\overline{R}$. The digital boundary of a region R will be defined as the set of all the boundary points of R. The contour of the region R, denoted by B(R), is defined as the closed polygonal chain whose vertices are all the lattice points of the digital boundary of R traversed in a clockwise direction. Since the boundary points of R are 8-connected [Rosenfeld, 1970] the polygonal chain B(R) consists of straight line segments whose length is either 1 or $\sqrt{2}$. In this paper, the contour of a digital region and its digital boundary will not always be differentiated. Figure 2.2 shows the contour of a digital region R and how its boundary points are traversed

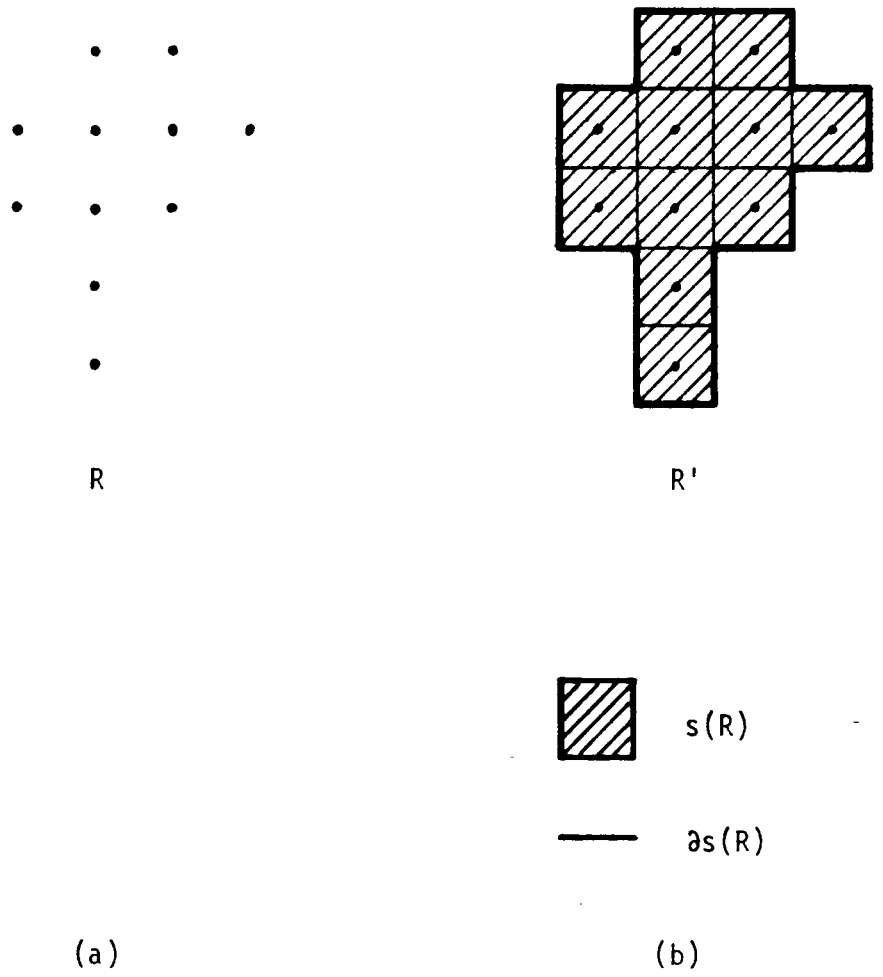
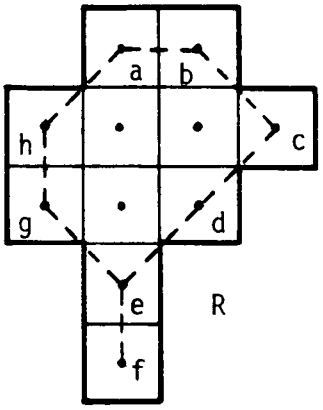


Figure 2.1. A digital region represented by
(a) its lattice points, (b) its unit square cells.



——— ∂s(R)

– – – B(R)

Figure 2.2. A digital region R showing the cell boundary ∂s(R) and the digital boundary B(R). Note that B(R) = abcdefegh.

by B(R). A polygon is represented by a list of points, called vertices, in the order that are encountered as the boundary is traversed in a clockwise direction.

A simple polygon is a polygon whose boundary is a simple closed curve. An arbitrary crossing polygon is a finite set of points connected such that P_i is connected to P_{i+1} for i = 1, 2, ..., n-1 and P_n is connected to P_1.

The convex hull of a finite set of points S, denoted by CH(S), is defined by the minimum area convex set containing the set S. A digital region R is convex if and only if its convex hull CH(R) does not contain a point of $\overline{R}$ (Lemma 18 of [Kim, December, 1980]). The convex deficiency of the digital region R, denoted by CD(R), is defined by the set difference CH(R) - R. Obviously, the convex deficiency of a convex digital figure results in the empty set.

The area of the convex deficiency of R will be defined as the total area of the set of polygons whose boundaries consist of parts of CH(R) and B(R) and whose interiors contain at least one point in $\overline{R}$. The shaded areas in Figure 2.3 illustrate the convex deficiency of the digital figure R.

An extreme point of a polygon P is a point of P that lies on the boundary of CH(P). Similarly, an extreme point of a digital region R will be defined as a point of R that lies on the digital boundary of CH(R). Let H = CH(R), then the set of extreme points of R, E(R), is defined as $E(R) = B(H) \cap B(R)$.

An ordered crossing polygon P is a crossing polygon whose convex hull vertices occur in the same order in both P and its convex hull,

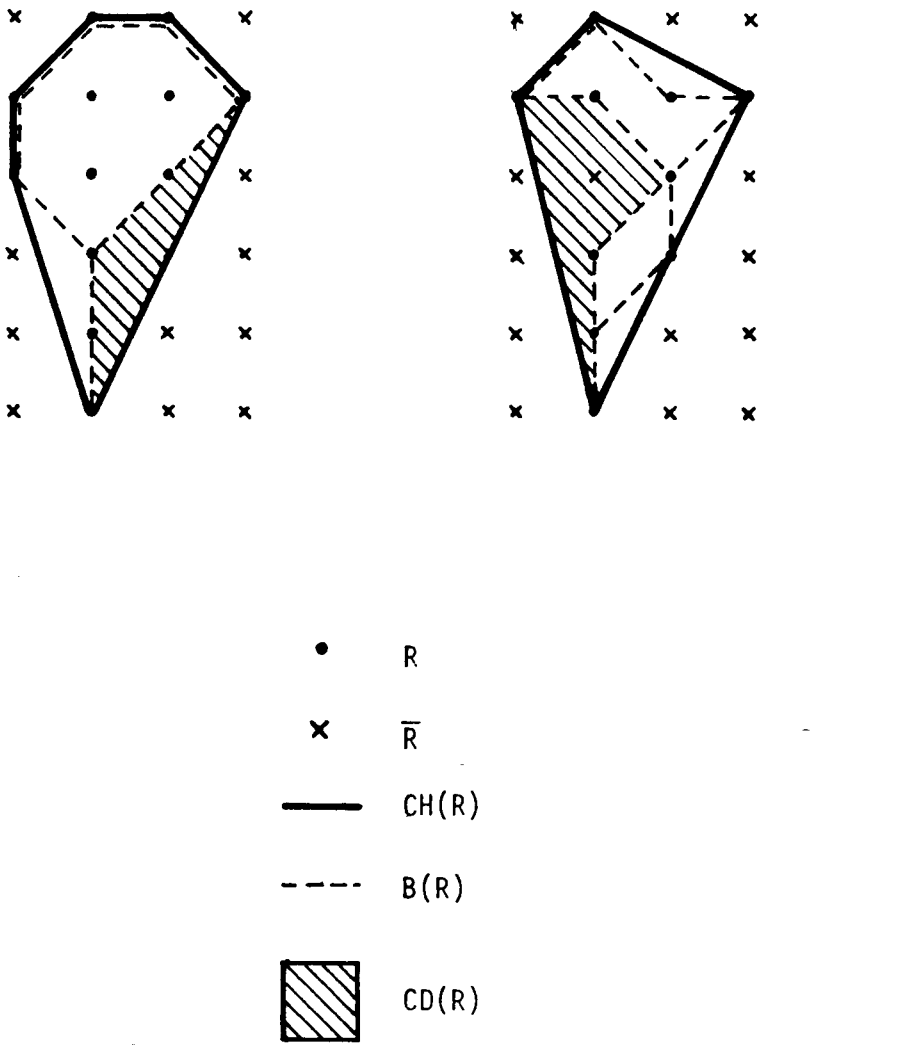


Figure 2.3. Digital figures and their convex deficiency. Note that $R \subseteq CH(R)$ and $CD(R) = CH(R) - R$.

CH(P). The contour of a digital image S consists of a closed 8-path that traverses all the boundary points of S in a clockwise direction. Moreover, since the boundary points of S each have at least one 4-neighbor in $\overline{S}$, it is easily shown that when the boundary of S is traversed in a clockwise direction, the lattice points that lie to the left belong to $\overline{S}$. Therefore, the contour of the digital region results in an ordered crossing polygon with some special properties, i.e., no interior points lie to the left as the boundary is traversed clockwise. By definition, a simple polygon is an ordered crossing polygon. Unfortunately, the contour of a digital region is not necessarily a simply polygon as shown in Figure 2.2.

For any vertex P_i, x_i and y_i will denote its x- and y- coordinates respectively. P_{xmin} and P_{xmax} will denote the vertices with minimum and maximum x-coordinates respectively. Similarly, P_{ymin} and P_{ymax} will denote the vertices with minimum and maximum y-coordinates respectively.

Given three points $P_i = (x_i,y_i)$, $P_j = (x_j,y_j)$ and $P_k = (x_k,y_k)$, then the angle subtended by P_i,P_j,P_k is defined as the clockwise angle that the vector $\overrightarrow{P_iP_j}$ needs to be rotated through about P_j to coincide with the vector $\overrightarrow{P_jP_k}$.

Let $S(P_i,P_j,P_k) = (x_k - x_j)\ (y_j - y_i) - (y_k - y_j)\ (x_j - x_i)$. Depending on the value of $S(P_i,P_j,P_k)$, the following can be asserted about the points P_i,P_j,P_k:

a) if $S(P_i,P_j,P_k) < 0$, then the angle subtended by P_i,P_j,P_k is concave.

b) if $S(P_i,P_j,P_k) = 0$, then P_i,P_j and P_k are collinear.

c) if $S(P_i,P_j,P_k) > 0$, then the angle subtended by P_i,P_j,P_k is convex.

The proof to these assertions can be found in [Sklansky, 1972].

A convex polygon is a polygon that completely encloses the line segment joining any two interior points. A convex polygonal chain that does not intersect itself is referred to as a simple convex chain. A polygon chain P_1, P_2, ..., P_n, denoted by $\langle P_1,P_n\rangle$, is monotonic if the sequences x_1, x_2, ..., x_n and y_1, y_2, ..., y_n are both monotonic.

A maximal polygon is defined as a simple polygon in which there exists four extreme points in the two orthogonal directions (for instance, P_{xmax}, P_{xmin}, P_{ymax}, P_{ymin}, since the x- and y-axes are chosen as the preferred directions) such that between every pair of consecutive extrema, such as P_{xmin} and P_{ymax}, there exists a polygonal chain, $\langle P_{xmin}, P_{ymax}\rangle$, monotonic in both preferred directions. This definition will be adjusted to include the case where more than one extreme point in any preferred direction exists. For example, if $P_i \neq P_j$ and $x_i = x_j > x_k$ for $k = 1, 2, \ldots, n$ and $k \neq i$, $k \neq j$, then $P_{xmax} = P_i$ if and only if $y_i > y_j$. P_{xmin}, P_{xmax}, P_{ymin}, and P_{ymax} will be referred to as the principal extreme points. It can be easily shown that a plane figure can have either two, three, or four principal extreme points.

III. DEVELOPMENT OF THE ALGORITHM

In this section, a linear time algorithm for obtaining the convex deficiency of digital regions is described. The performance and correctness of the algorithm are discussed in the next section, where a set of theorems are introduced to prove that the algorithm does in fact work for digital regions and that it is linear in both time and space.

The convex deficiency algorithm is now introduced. The input to the algorithm consists of a simply 8-connected digital region R, where the pixels belonging to the background ($\overline{R}$) are labeled 1 (white) while the pixels belonging to the object or figure (R) are labeled 0 (black). The output of the algorithm is a list of vertices representing the contour B(R) as the digital boundary of R is traversed clockwise.

Upon output, those points in B(R) that are vertices of CH(R) are labeled with a positive value; those points in B(R) that lie on the boundary of CH(R) but are not vertices of CH(R) are labeled with a zero; and those points in B(R) that are interior points to the convex hull of R, CH(R), are labeled with a negative value. It will be demonstrated in the next section that the existence of at least one point in B(R) that is also an interior point of CH(R) is a necessary and sufficient condition for the convex deficiency of R to be non-empty.

The convex deficiency algorithm can be stated as follows:

Algorithm: CONVEXDEFICIENCY (R)

Step 1: Get the contour of R,

$B(R) = P = P_1, P_2, \ldots, P_n$.

Step 2: Get the convex hull of R, CH(R), from B(R).

Step 3: Given CH(R), obtain the convex deficiency CD(R).

The boundary of the digital figure can be obtained by using a contour following algorithm like the ones reported in [Duda, 1967], [Symons, 1968], [Pavlidis, 1982]. This paper will be concerned primarily with the algorithms that perform Steps 2 and 3 in algorithm CONVEXDEFICIENCY.

Once the boundary of R has been obtained, then algorithm CONVEXHULL can be used to get CH(R). The input to the convex hull finder algorithm consists of an array containing the x- and y-coordinates of the n points of the ordered crossing polygon P that represents the digital boundary of R. The output of this algorithm consists of a circular linked list containing the indices of the vertices of CH(R), listed in counter-clockwise order, starting with P_{xmin}. The linked list is stored in an array called Hull that has the following structure:

$$P_{xmin} \rightarrow \text{Hull}\ (P_{xmin}) \rightarrow \text{Hull}\ (\text{Hull}(P_{xmin})) \rightarrow \ldots \rightarrow P_{xmin}$$

The algorithm uses no stacks at all. Instead, it starts out by making the initial assumption that CH(R) = P, i.e., the array Hull initially contains the indices of all the points in P listed in counter-clockwise order. Then, the algorithm scans the points of P in clockwise order, starting with P_{xmin}. The Hull array is updated only when a vertex is rejected as an extreme point. The algorithm backtracks only when a concave exterior angle is encountered. Furthermore, at each iteration, the vertex being considered is either accepted as a potential extreme point or rejected and never considered again, resulting in a

linear time algorithm. The proof to this claim, as well as a detailed explanation of how and why the algorithm works is presented in the next section.

The convex hull finding algorithm can be described as follows:

Algorithm: CONVEXHULL (P, HULL, n, P_{start})

Step 1: Find the vertices with minimum and maximum x-coordinates and call them P_{xmin} and P_{xmax} respectively. Similarly, find the vertices with minimum and maximum y-coordinates and call them P_{ymin} and P_{ymax} respectively.

Let $P_{start} = P_{xmin}$

Step 2: Initialize the Hull array:

Hull(n) = n-1, Hull(n-1) = n-2, ..., Hull(2) = 1,

Hull(1) = n.

Step 3: Call P_{start} and its next two consecutive clockwise points P_i, P_j and P_k respectively. Let $\Delta x = 1$. If $P_{start} \neq P_{ymax}$ then let $\Delta y = 1$, otherwise, let $\Delta y = -1$. If $n \leq 2$, RETURN.

Step 4: (a) IF $P_j = P_{ymax}$ THEN $\Delta y = -1$

IF $P_j = P_{xmax}$ THEN $\Delta x = -1$

IF $P_j = P_{ymin}$ THEN $\Delta y = 1$

IF $P_j = P_{xmin}$ THEN RETURN

IF $P_j = P_{ymax}$ OR $P_j = P_{xmax}$ OR $P_j = P_{ymin}$ THEN

Advance one point

GOTO Step 4(a)

ENDIF

(b) LET $m_x = x_k - x_j$

LET $m_y = y_k - y_j$

LET $S = m_x(y_j - y_i) - m_y(x_j - x_i)$

IF $\Delta x < 0$ THEN $m_x = -m_x$

IF $\Delta y < 0$ THEN $m_y = -m_y$

(c) IF $S < 0$ THEN

Delete P_j

IF $P_i = P_{xmin}$ OR $P_i = P_{xmax}$ OR

$P_i = P_{ymin}$ OR $P_i = P_{ymax}$ THEN

Advance one point

ELSE

Move one point backward

ENDIF

GOTO Step 4(a)

ENDIF

(d) IF $m_x \geq 0$ AND $m_y \geq 0$ THEN

IF $S = 0$ THEN Delete P_j

Advance one point

ELSE

Delete P_k

ENDIF

GOTO Step 4(a)

Once the convex hull is obtained by means of algorithm CONVEXHULL, the following convex deficiency finding algorithm can be applied to determine the existence of a convex deficiency. The same algorithm can

be used to determine the convexity of the region R, since a digital region S is convex if and only if its convex deficiency CD(S) is the empty set. The input to algorithm DEFICIENCY consists of the convex hull CH(R) (in the form of a linked list), the contour of R, B(R) = P, the number of points in P and the index P_{start} to the start of the linked list. The output of the algorithm consists of a list of all the points in the contour of R in which each point P_i is labeled with a positive value if it is a convex hull vertex, with a zero if it is also an extreme point, i.e., if it lies on the convex hull boundary; or with a negative value if it is also an interior point of the convex hull CH(R). In order to conserve memory space the array Hull can be used to store the label of every corresponding boundary point so that in effect, Hull (P_i) contains the label corresponding to the point P_i. The relationship between vertices labeled with negative values and the convex deficiency of a digital region will be discussed in the next section.

The convex deficiency finding algorithm can be stated as follows.

Algorithm: DEFICIENCY (P, Hull, n, P_{start})

Step 1: LET $P_k = P_{start}$

Step 2: LET P_i = Hull (P_k)

Step 3: IF $P_{i+1} \neq P_k$ THEN

IF $|x_k - x_i| > |y_k - y_i|$ THEN

LET $x_{inc} = 0$, $y_{inc} = +1$

IF $x_k > x_i$ THEN $y_{inc} = -1$

ELSE

LET $x_{inc} = -1$, $y_{inc} = 0$

IF $y_k > y_i$ THEN $x_{inc} = +1$

ENDIF

LET $P_{str} = (x_i + x_{inc}, y_i + y_{inc})$

LET $P_{end} = (x_k + x_{inc}, y_k + y_{inc})$

DO FOR j = i+1 TO k-1

Hull $(P_j) = 0$

IF $S(P_{str}, P_j, P_{end}) \leq 0$ THEN Hull $(P_j) = -1$

END

ENDIF

Step 4: LET $P_k = P_i$

Step 5: IF $P_k \neq P_{start}$ GOTO Step 2

Step 6: RETURN

Finally, algorithm CONVEXDEFICIENCY can be implemented as a sequence of routine calls in which algorithms CONVEXHULL and DEFICIENCY are called upon, resulting in the following procedure:

Algorithm: CONVEXDEFICIENCY (R)

Step 1: CALL CONTOUR (R, P, n)

Step 2: CALL CONVEXHULL (P, Hull, n, P_{start})

Step 3: CALL DEFICIENCY (P, Hull, n, P_{start})

Step 4: STOP

IV. ANALYSIS OF THE ALGORITHM

In this section, the performance on the convex deficiency algorithm is studied and a proof of correctness is introduced. First, the correctness of the algorithm CONVEXHULL is proved by means of a theorem. Next, the concepts of digital convexity introduced in [Kim, December, 1980] are used to prove the correctness of the algorithm DEFICIENCY. The complexity of both algorithms in terms of space and execution time is then discussed. Finally, a comparative evaluation of the algorithm CONVEXHULL with respect to other convex hull algorithms that have appeared in the recent literature is also included.

A. Proof of Correctness

To prove the correctness of the algorithm CONVEXHULL, some basic properties of convex polygons and a new convex hull invariant condition need be established.

One basic property of convex polygons is described by the following lemma:

Lemma 1: Every convex polygon is maximal.

Proof: Let P be a convex polygon with clockwise vertices $P_1, P_2, \ldots, P_i, \ldots, P_j, \ldots, P_n$ such that $P_i = P_{xmin}$ and $P_j = P_{ymax}$. Consider the polygonal chain $\langle P_{xmin}, P_{ymax} \rangle = P_i, P_{i+1}, \ldots, P_{j-1}, P_j$, as shown in Figure 4.1. Consider now the relative position of P_{i+1} with respect to P_i and P_j. Since $P_i = P_{xmin}$ and $P_j = P_{ymax}$, then it follows that $x_i \leq x_{i+1}$ and $y_{i+1} \leq y_j$. Also, since a convex polygon must contain the line

segment that joins every two of its vertices, the vertex P_{i+1} must lie to the left of the line segment $\overline{P_iP_j}$. Therefore, $y_i \leq y_{i+1}$ as shown in Figure 4.1. Using the same argument for all the vertices in $\langle P_i, P_j \rangle$ it can be shown by induction that $x_i \leq x_{i+1} \leq \ldots \leq x_{j-1} \leq x_j$ and $y_i \leq y_{i+1} \leq \ldots \leq y_{j-1} \leq y_j$. Therefore, $\langle P_i, P_j \rangle$ is a monotonic polygonal chain. Similar arguments can be used to prove that the other three chains forming P, $\langle P_{ymax}, P_{xmax} \rangle$, $\langle P_{xmax}, P_{ymin} \rangle$, $\langle P_{ymin}, P_{xmin} \rangle$ are all also monotonic. Hence P is also a maximal polygon.

Q.E.D.

The following lemma is a direct consequence of lemma 1.

Lemma 2: (Convex hull invariant condition)

Let $P_{i1}, P_{i2}, \ldots, P_{in}$ be the vertices of the convex hull of an ordered crossing polygon P with principal extreme points $P_{xmin}, P_{ymax}, P_{xmax}, P_{ymin}$. If P_i and P_j denote two consecutive principal extreme points in P, then the chain of vertices $\langle P_i, P_j \rangle = P_i, P_{i+1}, \ldots, P_{j-1}, P_j$ between P_i and P_j forms a monotonic convex chain.

Proof: The set of vertices $P_{i1}, P_{i2}, \ldots, P_{in}$ form, by definition, a closed convex polygonal chain. Since $CH(P) = P_{i1}, P_{i2}, \ldots P_{in}$ is a convex polygon, by Lemma 1, it is also maximal. Therefore, between any

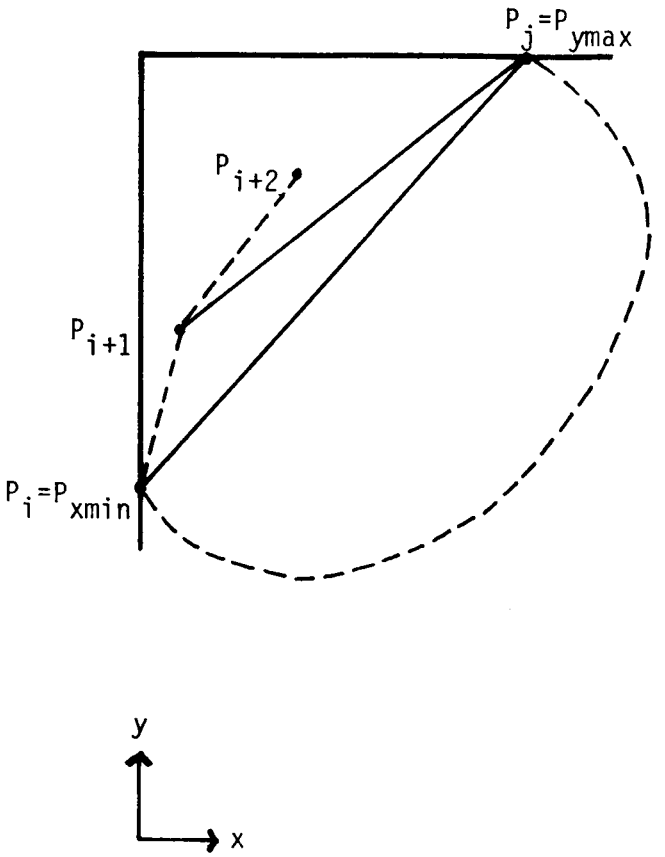


Figure. 4.1. Proof to Lemma 1.

two consecutive principal extreme points P_i and P_j, there exists a convex polygonal chain monotonic in both the x and y directions. Also, a monotonic polygonal chain cannot intersect itself, i.e., a monotonic chain is also simple.

Q.E.D.

Lemma 2 will be referred to as the convex hull invariant condition. This condition can be expressed in the following way:

(a) For any three consecutive points P_k, P_{k+1}, and P_{k+2} between P_{xmin} and P_{ymax}, the following holds true:

$S(P_k, P_{k+1}, P_{k+2}) > 0$;

$x_k \leq x_{k+1} \leq x_{k+2}$; and $y_k \leq y_{k+1} \leq y_{k+2}$

(b) For any three consecutive points P_k, P_{k+1}, and P_{k+2} between P_{ymax} and P_{xmax}, the following holds true:

$S(P_k, P_{k+1}, P_{k+2}) > 0$;

$x_k \leq x_{k+1} \leq x_{k+2}$; and $y_k \geq y_{k+1} \geq y_{k+2}$

(c) For any three consecutive points P_k, P_{k+1}, and P_{k+2} between P_{xmax} and P_{ymin}, the following holds true:

$S(P_k, P_{k+1}, P_{k+2}) > 0$;

$x_k \geq x_{k+1} \geq x_{k+2}$; and $y_k \geq y_{k+1} \geq y_{k+2}$

(d) For any three consecutive points P_k, P_{k+1}, and P_{k+2} between P_{ymin} and P_{xmin}, the following holds true

$S(P_k, P_{k+1}, P_{k+2}) > 0$;

$x_k \geq x_{k+1} \geq x_{k+2}$; and $y_k \leq y_{k+1} \leq y_{k+2}$

Note that the convex hull invariant condition presented here is different from the one introduced in [Ghosh, 1983]. Explicitly, the one presented in [Ghosh, 1983], hereafter referred to as invariant condition GS, is stated in the following manner:

(a) For any three consecutive points P_k, P_{k+1}, and P_{k+2} between P_{xmin} and P_{xmax} (in a clockwise direction) the following holds true:

$$S(P_k, P_{k+1}, P_{k+2}) > 0 \text{ and } x_k \leq x_{k+1} \leq x_{k+2}$$

Note that the condition $[(x_k < x_{k+1})$ or $(x_{k+1} < x_{k+2})]$ follows from $S(P_k, P_{k+1}, P_{k+2}) > 0$

(b) For any three consecutive points P_k, P_{k+1} and P_{k+2} between P_{xmax} and P_{xmin} (in a clockwise direction) the following holds true:

$$S(P_k, P_{k+1}, P_{k+2}) > 0 \text{ and } x_k \geq x_{k+1} \geq x_{k+2}$$

If only the x-direction is considered, both the convex hull invariant condition presented here and the invariant condition GS are equivalent. However, the invariant condition GS, when used as the only criteria to delete the points of a polygon P that cannot belong to the convex hull CH(P), proves to be necessary but not sufficient. The convex hull algorithm presented by Ghosh and Shyamasundar (Algorithm A1 in [Ghosh, 1983]) deletes a point only if it does not satisfy the invariant condition GS. Figure 4.2, shows a simple polygon for which that algorithm fails to determine its convex hull.

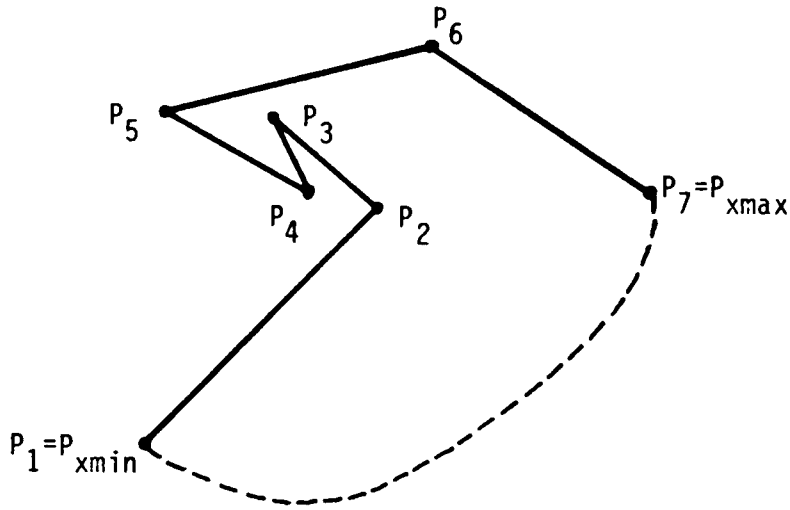


Figure 4.2. Example of a simple polygon for which algorithm A1 in [Ghosh, 1983] fails.

Consider the polygonal chain $\langle P_{xmin}, P_{xmax} \rangle$ in Figure 4.2. Algorithm A1 considers the triplet P_1,P_2,P_3 and deletes P_2 since $S(P_1,P_2,P_3) < 0$. It then considers the triplet P_1,P_3,P_4, which satisfies the invariant condition GS, i.e., vertex P_3 is accepted as a potential convex hull vertex. The triplet P_3,P_4,P_5 is then considered resulting in the deletion of P_5 since $S(P_3,P_4,P_5) > 0$ and $x_4 > x_5$ (condition GS is not met). Now, P_5 is obviously in the convex hull of Figure 4.2, so algorithm A1 fails to obtain its convex hull correctly. The reason for this failure is that algorithm A1 does not guarantee that a self-intersecting polygonal chain is not created whenever a vertex is deleted.

It will be shown now that the algorithm CONVEXHULL uses the convex hull invariant condition presented here to guarantee that the polygon resulting from the deletion of any point is not self-intersecting.

Algorithm CONVEXHULL operates in the following manner: it starts by assuming that all the points of the input polygon P are vertices of its convex hull CH(P). The algorithm then scans all the points in P in a clockwise direction starting with P_{xmin}. Everytime a point that is not a vertex of CH(P) is encountered, it is deleted. At the end of the algorithm only the convex hull vertices are left in a linked list that stores them in counter-clockwise order. The algorithm never deletes a convex hull vertex since those vertices satisfy the convex hull invariant condition.

At every iteration, the sequence P_i,P_j,P_k, where P_i, P_j, and P_k are in clockwise order, is studied. If P_j is a concave (or reflex) vertex, as shown in Figure 4.3(a), it is deleted since the edge (P_iP_k) guards

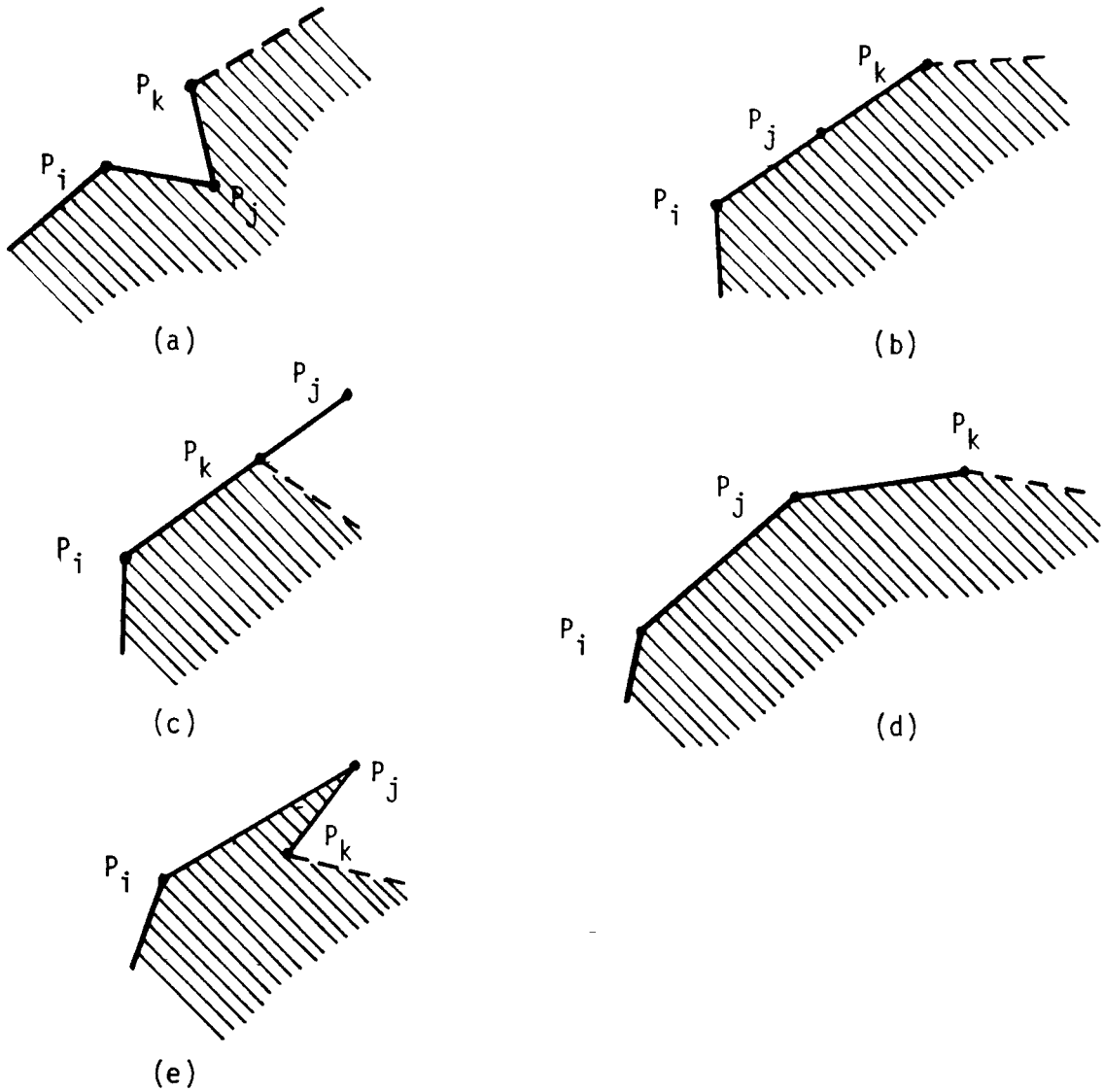


Figure 4.3. Example of a sequence P_i, P_j, P_k, that is (a) reflex. (b) collinear and monotonic. (c) collinear but not monotonic. (d) convex and monotonic. (e) convex but not monotonic

the area guarded by the edges (P_iP_j) and (P_jP_k). In order to assure that the chain P_{xmin}, ..., P_i,P_k forms a piecewise monotonic convex chain a backtracking step is taken to check that the triplet P_{i-1},P_i,P_k also satisfies the convex hull invariant condition. This is not necessary if P_i is a principal extreme point. Whenever the sequence P_i,P_j,P_k is both collinear and monotonic (Figure 4.3(b)), then P_j is deleted since the edge (P_iP_k) can replace the edges (P_iP_j) and (P_jP_k) without changing the area guarded by the chain P_{xmin}, ..., P_i,P_j,P_k. If the sequence is collinear but not monotonic (Figure 4.3(c)), then P_k is deleted since the edge (P_iP_j) covers the edge (P_jP_k). If the sequence of vertices P_i,P_j,P_k is not monotonic (Figure 4.3(e)), then P_k is deleted since it cannot be a convex hull vertex. Whenever the sequence P_i,P_j,P_k is both convex and monotonic, as in Figure 4.3(d), the vertex P_j is accepted as a potential convex hull vertex and the algorithm advances to the next point until the vertex P_{xmin} is reached again, in which case, the algorithm terminates.

The following theorem is now presented.

Theorem 1: At the end of every forward step taken by algorithm CONVEXHULL, the sequence of vertices P_{xmin}, ..., P_i,P_j forms a simple and piecewise monotonic convex chain.

Proof: The algorithm starts with $P_i = P_{xmin}$ and advances one point everytime the sequence P_i,P_j,P_k satisfies the convex hull invariant condition, which tests for convexity and piecewise monotonicity. Whenever P_j is deleted, a backtracking step is taken to check if P_{i-1},P_i,P_k satisfies the invariant condition. Since

the algorithm starts with a known convex hull vertex (P_{xmin}) and advances the P_j pointer only when the convex hull invariant condition is satisfied, it means that after a forward step is taken the chain P_{xmin}, ..., P_i,P_j is a simple and piecewise monotonic convex chain.

Q.E.D

The correctness of algorithm CONVEXHULL can be proven by means of the following theorem:

Theorem 2: The algorithm CONVEXHULL obtains the convex hull of an arbitrary digital figure.

Proof: The boundary of an arbitrary digital region is a non self-intersecting ordered crossing polygon. Call it polygon P. According to theorem 1, after every forward step, the chain of vertices P_{xmin}, ..., P_i,P_j forms a simple (piecewise) monotonic convex chain. Furthermore the algorithm terminates whenever $P_j = P_{xmin}$. Since the algorithm starts with $P_i = P_{xmin}$ and it never backtracks when P_i points to a principal extreme point, P_{xmin} can only be reached with a forward step, after which the sequence of vertices P_{xmin}, ..., P_i,$P_j = P_{xmin}$ forms a closed (piecewise) monotonic convex chain. The fact that the condition $P_j = P_{xmin}$ is ever reached and the algorithm terminates will be proven shortly. Since only the vertices that do not meet the convex hull invariant condition are deleted, the result

of the algorithm is a list with all the vertices of the convex hull of the polygon P. The test or piecewise monotonicity built into the convex hull condition guarantees that at every iteration the chain being considered is simple, but this holds only if the polygon P itself is not self-intersecting. Since the boundary of an arbitrary digital figure is not self-intersecting, algorithm CONVEXHULL obtains its convex hull.

Q.E.D.

The correctness of algorithm DEFICIENCY can be proved by using some concepts of digital convexity already presented in [Kim, December,1980].

The convex deficiency of a digital region R, denoted by CD(R), has already been defined as the set difference CH(R) - R. Thus, searching for the convex deficiency of a digital region R reduces to finding the lattice points in $\overline{R}$ that are contained in CH(R). This can be a very exhaustive method, so in turn the contour of the digital region is studied to determine whether or not a convex deficiency exists.

Let m and m′ be two parallel lines such that the smaller of the vertical and horizonal distances between them is equal to 1. Let u and v be two lattice points between m and m′ and possibly on m but not on m′. If L is the set that, besides u and v, contains all the lattice points between u and v such that they lie between m and m′ and on m but not on m′ , then L is a digital straight line (lemma 13 in [Kim, December, 1980]). The set L is also unique. Figure 4.4 shows an example of a digital straight line.

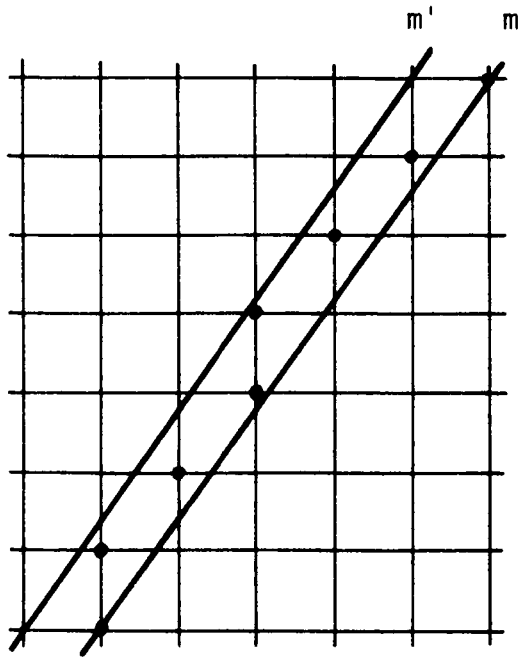


Figure 4.4. Example of a digital straight line.

Theorem 14 of [Kim, December, 1980] states that a digital region R is convex if and only if any two points in R are connected by a digital straight line in R, in other words, if there exists at least a pair of points in R such that they cannot be connected by a digital straight line entirely in R, then the region R is not convex.

For simply 8-connected digital regions, like the ones considered in this paper, the above theorem reduces to the following lemma:

Lemma 3: Let R be a simply 8-connected digital region and B(R) and CH(R) denote its digital boundary and convex hull respectively. Then R is convex if and only if every pair of consecutive vertices in CH(R) are connected by a digital straight line in B(R).

Proof: Let H = CH(R). R is convex if and only if H = R, i.e., if and only if B(H) = B(R). Now, if every pair of consecutive vertices in H are connected by a digital straight line in B(R), then B(H) = B(R), which means that R is convex.

Q.E.D.

Let v_i and v_j be two vertices on the digital boundary of a digital region S. Hence v_i, $v_j \in B(S)$. Now, let $B(v_i, v_j ; S)$ be the shortest section of B(S) that connects v_i with v_j. According to Theorem 14 of [Kim, December, 1980], if v_1, v_2, ..., v_m are the vertices of the convex hull H = CH(R), then $B(v_i, v_{i+1} ; H)$ is a digital straight line for i = 1, 2, ..., m. Lemma 3 shows that the region R is convex if and only if $B(v_i, v_{i+1}; R)$ is a unique digital straight line for v_1, v_2, ..., $v_n \in CH(R)$.

The following lemma can now be introduced:

Lemma 4: Let CH(R) be the convex hull of R with vertices v_1, v_2, ..., v_m. The chain $B(v_i, v_{i+1} ; R)$ is not a digital straight line if and only if there exists at least one lattice point of $\overline{R}$ in $B(v_i, v_{i+1} ; CH(R))$.

Proof: Let v_i and v_{i+1} be two consecutive vertices of CH(R). By definition, $B(v_i, v_{i+1} ; CH(R))$ is a unique digital straight line. If $B(v_i, v_{i+1} ; R)$ is not a digital straight line, then $B(v_i, v_{i+1} ; CH(R)) \neq B(v_i, v_{i+1} ; R)$ which means that there exists at least one point in $B(v_i, v_{i+1} ; CH(R))$ and therefore in CH(R) that does not belong to R.

Q.E.D.

Lemma 4 establishes the basis for the correctness of algorithm DEFICIENCY since it provides a way of finding those lattice points in $\overline{R}$ that belong to CH(R) by considering only the contour of R. Thus, the correctness of algorithm DEFICIENCY can now be established.

Theorem 3: Algorithm DEFICIENCY finds the convex deficiency of a digital region R by determining all the chains $B(v_i, v_{i+1} ; CH(R))$, $i = 1, 2, \ldots, m$ that contain at least one lattice point from $\overline{R}$.

Proof: Algorithm DEFICIENCY examines every point P_j on the boundary of R, B(R), and determines whether or not P_j is part of the set of extreme points of R, $E(R) = B(R) \cap B(CH(R))$. If $P_j \in B(v_i, v_{i+1} ; R)$ for some $i = 1, 2, \ldots, m$ but $P_j \notin B(v_i, v_{i+1} ; CH(R))$, then by

lemma 4, there exists a concavity in the chain $B(v_i, v_{i+1}$; R). For every pair of vertices v_i, v_{i+1}, in CH(R) the algorithm checks if every lattice point P_j in the chain $B(v_i, v_{i+1}$; R) lies on the digital straight line $B(v_i, v_{i+1}$; CH(R)). All those points that lie on $B(v_i, v_{i+1}$; CH(R)) are labeled with the value zero while those points that lie outside of $B(v_i, v_{i+1}$; CH(R)) are labeled with a negative value to signal that there exists a concavity between the vertices v_i and v_{i+1} .

Q.E.D.

To check whether a point P_j lies on the digital straight line between v_i and v_{i+1}, algorithm DEFICIENCY draws an imaginary line m′ such that the smaller of the two distances, the horizonal and the vertical, between m′ and $\overline{v_i v_{i+1}}$ is equal to 1 and m′ goes through the interior of the region R. If P_j lies between m′ and $\overline{v_i v_{i+1}}$ and on $\overline{v_i v_{i+1}}$ but not on m′ ,then P_j lies on the digital straight line $B(v_i, v_{i+1}$; CH(R)). Otherwise, P_j is not on $B(v_i, v_{i+1}$; CH(R)) and hence it points to a concavity.

Based on theorems 2 and 3, the correctness of the algorithm CONVEXDEFICIENCY can be established.

Theorem 4: Algorithm CONVEXDEFICIENCY obtains the convex deficiency of a simply 8-connected digital region.

Proof: It has been shown that there exists algorithms for obtaining the contour of an 8-connected digital region in linear time [Rosenfeld, 1970]. One such algorithm

is presented in Appendix A. Theorem 2 proves that the algorithm CONVEXHULL obtains the convex hull of an arbitrary digital figure. Theorem 3 proves that the algorithm DEFICIENCY obtains the concavities and hence, the convex deficiency of a simply 8-connected digital region.

Q.E.D.

If a contour tracking algorithm such as the one on page 146 of [Pavlidis, 1982] is used to obtain both the outside contour and the contour of any interior holes, then algorithm CONVEXDEFICIENCY would be able to obtain the convex deficiency of an arbitrary 8-connected digital region. Algorithm DEFICIENCY would take care of obtaining the exterior concavities or bays while Pavlidis' algorithm would obtain the holes. These two types of features are of great value when trying to describe a digital region, especially in pattern recognition.

B. Computational Properties

To prove that algorithm CONVEXDEFICIENCY is linear two more theorems need be introduced.

Theorem 5: Algorithm CONVEXHULL obtains the convex hull of a digital region in linear time.

Proof: Let n be the number of points on the boundary B(R) of a digital region R. Obviously, steps 1 and 2 of the algorithm can be performed in n iterations involving only assignment statements and comparisons. Step 3 is performed only once. The fourth step is the core of

the algorithm and proving that the number of iterations involved in the worst case condition is linear will suffice to prove that the algorithm is in fact linear. Step 4(a) is performed to guarantee the proper monotonicity test and to terminate the algorithm. It is also responsible for skipping the four principal extreme points. Step 4(b) is the most expensive step, involving at most five additions, two multiplications, and two negations. Steps 4(c) and 4(d) are responsible for the tests that cause the algorithm to either advance or backtract one point at a time. Note that every point is considered at least once, with the exception of P_{xmin}, P_{xmax}, P_{ymin}, and P_{ymax} which are never considered. Every point considered is either accepted as a possible convex hull vertex or deleted and never considered again. One point has to be deleted before a backtracking can occur. Since the algorithm never backtracks into a principal extreme point, the worst possible case is when it has to backtrack every point considered. Thus, the worst case number of backtracks is $(n - m) - m$ where m is the number of distinct principal extreme points. Since it was already established that at most $n - m$ forward iterations are performed, then the absolute worse case total number of iterations is $(n - m) + ((n - m) - m) = 2n - 3m$, which is possible only for $n \geq 3m$ where

$2 \leq m \leq 4$. Therefore, the algorithm is linear.

Q.E.D.

Theorem 6: Algorithm DEFICIENCY obtains the convex deficiency of a simply 8-connected digital figure in linear time.

Proof: Let n be the number of points on the boundary B(R) of a digital region R. Let m be the number of convex hull vertices. Step 1 of the algorithm is executed once. Steps 2, 4, and 5 are executed m times but no calculations are involved. Step 3 is also executed m times, requiring six additions per iteration to compute the location of the imaginary line m′. It also contains an internal loop that considers only those points in B(R) that are not vertices of CH(R), that is, it only considers (n - m) points. For each point, five additions and two multiplications are computed. If addition and multiplication are considered as the same "operation," the total cost of algorithm DEFICIENCY is $6m + 7(n - m) = 7n - m$ "operations." Therefore the algorithm is linear.

Q.E.D.

A theorem establishing the linearity of algorithm CONVEXDEFICIENCY is now introduced.

Theorem 7: Algorithmm CONVEXDEFICIENCY obtains the convex deficiency of a simply 8-connected digital region in linear time and space.

Proof: Theorems 5 and 6 show that the algorithms CONVEXHULL, and DEFICIENCY, respectively, are all linear in time with respect to n (the number of points in the boundary of the digital region). Rosenfeld, in [Rosenfeld, 1970], shows that the contour of a digital image can be obtained in linear time. Therefore, the algorithm CONVEXDEFICIENCY is linear in time. Algorithm CONVEXDEFICIENCY runs in linear space since the arrays P and Hull are of size n, since only n points are considered.

Q.E.D.

C. Comparative Evaluation

Several linear time convex hull algorithms have been reported in the recent literature. McCallum and Avis reported a linear time algorithm for finding the convex hull of a simple polygon [McCallum, 1979]. That algorithm uses two stacks: stack A is used to contain the points of the polygon that have been scanned so far (starting from P_{xmax}) and have been accepted as possible convex hull extreme points; stack T contains some of the rejected points that form a convex chain starting from P_{xmin}. The output is reported in a linked list. The major drawback of this algorithm is that even though it is linear, the use of two stacks renders it very inefficient and complicated since it needs to backtrack both stacks everytime a new point is considered to check that the vertices in both stacks still form two convex chains. Since each test requires at least five additions and two

multiplications, the cost of the algorithm in [McCallum, 1979] is much higher in both execution time and memory space than that of algorithm CONVEXHULL. Also its complexity requires a very cumbersome proof of correctness. By using the convex hull invariant condition, algorithm CONVEXHULL avoids the use of stacks.

C. E. Kim proposed a linear time algorithm as a special case of an algorithm for obtaining the convex hull of an ordered crossing polygon. The problem with that algorithm is that some counter examples have been reported (cf. [Toussaint, 1982]).

S. K. Ghosh and R. K. Shyamasundar have proposed two convex hull algorithms: One works for simple polygons [Ghosh, 1983] and the other works for ordered crossing polygons [Ghosh, 1984]. A counter example for the former has already been presented in this paper along with an explanation of why their algorithm does not always work. Their convex hull algorithm for ordered crossing polygons (algorithm CROSS-POL) is linear and could be used instead of algorithm CONVEXHULL. However, that algorithm has several drawbacks that render it less efficient than algorithm CONVEXHULL when dealing with digital figures. Algorithm CROSS-POL uses three stacks: stack C contains the vertices so far scanned that have been accepted as possible convex hull vertices; stack P contains the vertices so far scanned that have been neither rejected nor accepted as convex hull vertices; and stack T is used for temporary storage. Obviously, in terms of memory space, algorithm CROSS-POL is more expensive than algorithm CONVEXHULL. In terms of computation cost algorithm CROSS-POL is much more expensive than algorithm CONVEXHULL since it has to handle three stacks when it backtracks to check whether

accepting or rejecting some vertex affects the status of the vertices already pushed onto the stacks. Also this algorithm backtracks more frequently since it has to check for self-intersecting sides due to two reasons. One reason is that it uses the convex hull invariant condition of [Ghosh, 1983], which as shown in this paper, does not guarantee that a self-intersecting polygon is not generated when a vertex is deleted. The other reason is that it was originally designed to work with arbitrary ordered polygons, and as it has been shown, the boundary of a digital region has more structure. Also, the worst case iteration cost is very high and very hard to calculate since some iterations can be quite complicated (cf. [Ghosh, 1984]). Therefore, the advantage of using algorithm CONVEXHULL over algorithm CROSS-POL to find the convex deficiency of a digital region should be apparent.

D. Experimental Results

Some experimental results are now presented to demonstrate how the algorithm CONVEXDEFICIENCY works. The algorithm was implemented in FORTRAN-77 on a DEC PDP 11/73 computer. The input consisted of 64 x 64 images containing one digital figure each. Figures 4.5, 4.6, and 4.7 show the results of applying algorithm CONVEXDEFICIENCY to three different images. In all three figures, each "*" represents an interior point, each "V" represents a convex hull vertex, each "E" represents an extreme point, and each "C" represents a convex deficiency boundary point. Figure 4.5 depicts a digital region for which Sklansky's algorithm [Sklansky, 1972] would fail since its contour is not a weakly externally visible polygon. Nevertheless, algorithm CONVEXHULL obtains

Figure 4.5. Result No. 1.

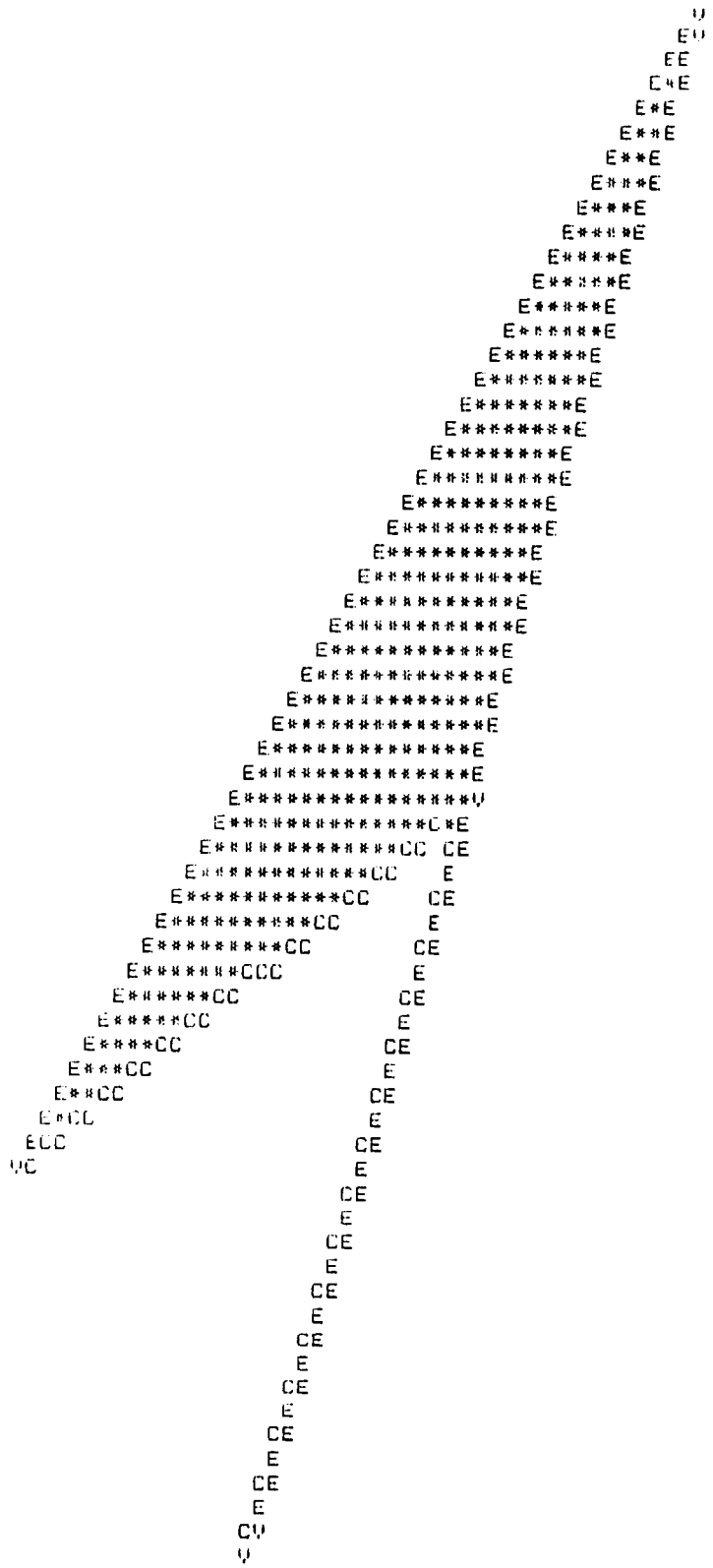


Figure 4.6. Result No 2.

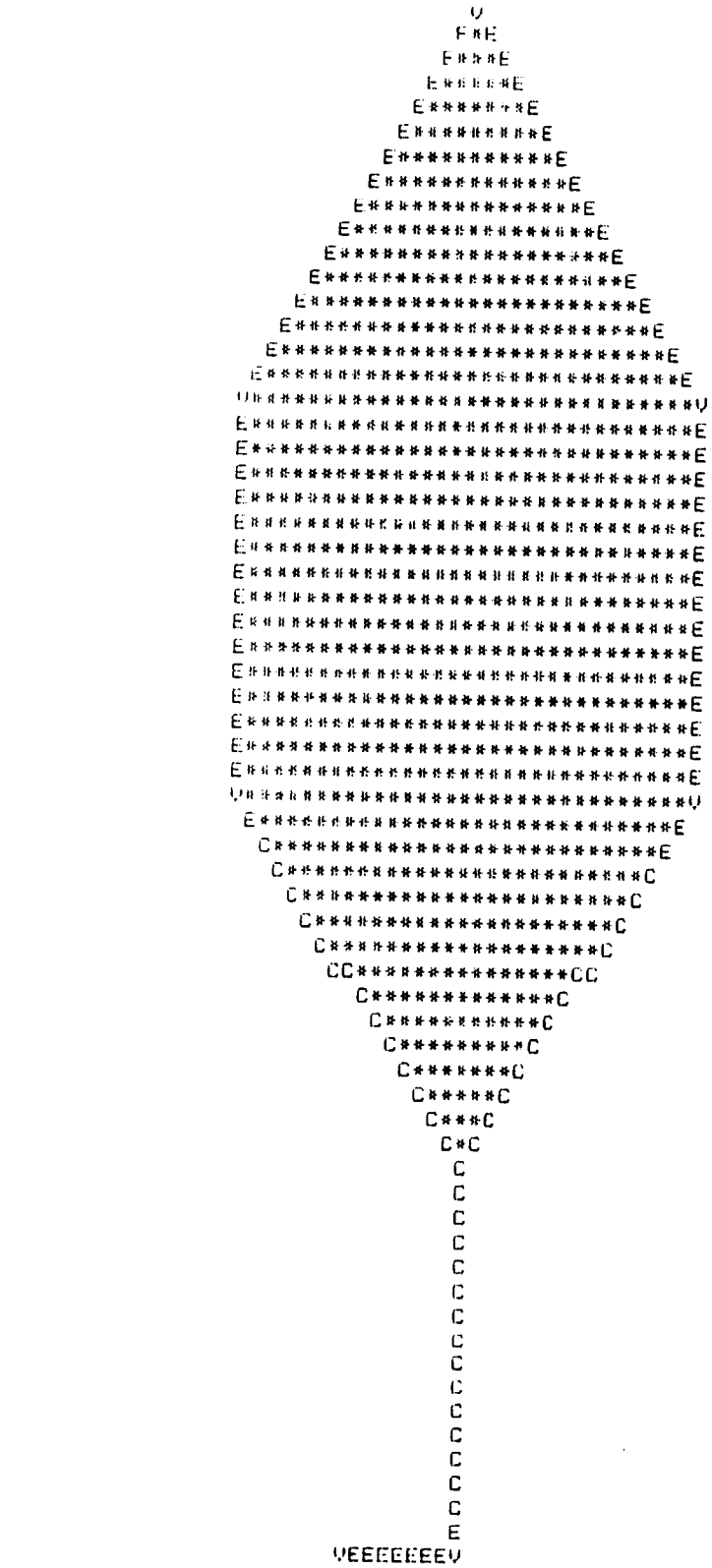


Figure 4.7. Result No. 3.

the correct convex hull and algorithm DEFICICIENCY singles out its three bays. Figure 4.6 shows a region in which the convex deficiency consists of only one bay. Note that on that figure the boundary at the lower leg retraces itself at every other point. Those points belong both to the set E(R) of extreme points and to the boundary of the convex deficiency. For that reason they are often called multiple boundary points as opposed to simple boundary points. Figure 4.7 shows a region in which the contour retraces itself, i.e., it is not a simple polygon. Nevertheless, the correct result was obtained by algorithm CONVEXDEFICIENCY. Note that all the points between the bottom two convex hull vertices are all labeled "E" because, when a pixel appears twice in the contour sequence, the label "V" takes precedence over "E" and "E" over "C." It should be added that the use of most other linear time convex hull algorithms on a region like the one depicted in Figure 4.7. is doomed to failure due to the fact that its contour does not constitute a simple curve.

V. APPLICATIONS TO PATTERN RECOGNITION

This section describes how the algorithms developed in this paper can be used in some pattern recognition applications. Several uses for the convex hull in pattern recognition have already been reported in [Toussaint, 1978].

Many recognition systems, i.e. character readers, parts classifiers, etc., require that the objects to be recognized be normalized to a certain size before they are processed. The use of the height and width of the object's silhouette for normalization purposes does not yield good results since they both vary according to the degree of rotation of the object. Figure 5.1. shows an example of how the rotation of an object affects the height and width. The convex hull, on the other hand, is a feature that is rotation independent, therefore, its area constitute an excellent normalization criteria. Algorithm CONVEXHULL can be used to obtain the convex hull of the figure, then the area of the convex hull can be computed by the following formula:

$$\text{Area} = \frac{1}{2} \sum_{P_i \in CH(R)} (x_i y_{i+1} - x_{i+1} y_i)$$

where the index additions are modulo-n, $P_i = (x_i, y_i)$, and $P_1, P_2, \ldots, P_n \in CH(R)$ and are listed in counter-clockwise order. This can be achieved easily since algorithm CONVEXHULL generates the linked list that contains the convex hull vertices in counter-clockwise order.

A more important application for the algorithms developed in section III consists of using them in a shape analysis scheme that can

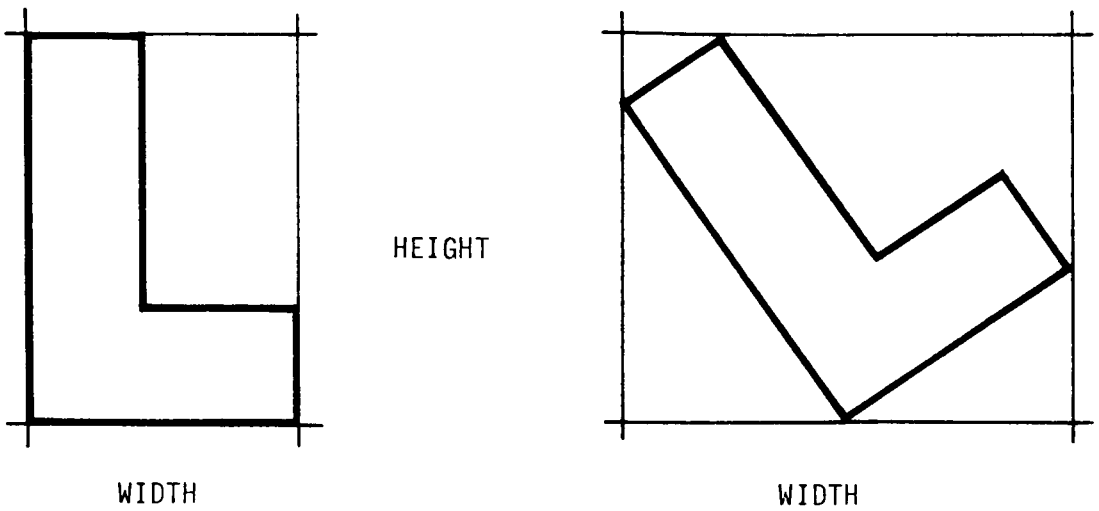


Figure 5.1. Effects of rotation on the height and width of digital regions.

be part of a complete pattern recognition system. The shape analysis strategy suggested is one that uses both the convex hull and convex deficiency as shape descriptors. Algorithm CONVEXDEFICIENCY can be modified to comply with such a scheme and become a feature extraction and description module. A shape recognizer module that studies the description of the shape of an object and matches it against that of a "known" shape can then be added.

A simplified shape analysis scheme is now presented. In this scheme, the shape of a connected figure is characterized by means of its convex hull and convex deficiency. The accuracy of the description depends on the particular application in which it is used. In most situations, the convex hull can be used to determine the orientation of the object. This is done by using the convex hull to obtain the hull diameter in linear time [Toussaint, 1980]. The counter-clockwise angle between the the hull diameter and the horizontal represents a rough estimate of the orientation of the object in the plane. Other possible features include (a) the hull area, A; (b) the hull perimeter, P; (c) the hull thinness ratio, defined by the formula:

$$\text{Thinness} = T = 4\pi \frac{A}{P^2} .$$

The convex deficiency can be characterized by a description of all its simply-connected components. Two types of convex deficiency components are identified:

1. holes, which are those components completely surrounded by object points;

2. bays, which are those components where at least one pixel has a 4-neighbor that lies outside the convex hull.

Figure 5.2. shows a convex deficiency with both types of components.

As mentioned in section IV, the holes can be extracted by a contour following algorithm like the one in [Pavlidis, 1982]. A simple description of the holes includes features like: (a) the centroid; (b) the area; (c) the perimeter; (d) the thinness ratio. Since all these features are readily available once the contour of the hole is determined, they can be computed at run time by a simple modification of Pavlidis' algorithm. In the modified algorithm, the calculation of the hole features is updated everytime a contour point is obtained. The type of modification suggested here does not affect the linear time performance of the algorithm.

Likewise, algorithm DEFICIENCY can be modified to compute the features of the bays as soon as they are found. Some of the features that can be used to described the bays are: (a) the area; (b) the perimeter; (c) the centroid; (d) the size of the opening; (e) the relative position on the boundary with respect to a reference point; (f) the orientation wih respect to the horizontal. To simplify the computation of these features, it is best to describe them in terms of the pixels that lie on the boundary of the bay since they are readily available. Some of these features are described in the example of Figure 5.3. Algorithm DEFICIENCY can be modified by adding one more step after step 3. The new step can compute the bay features by retracing the boundary of the bay. Theorem 6 of section IV, after a few

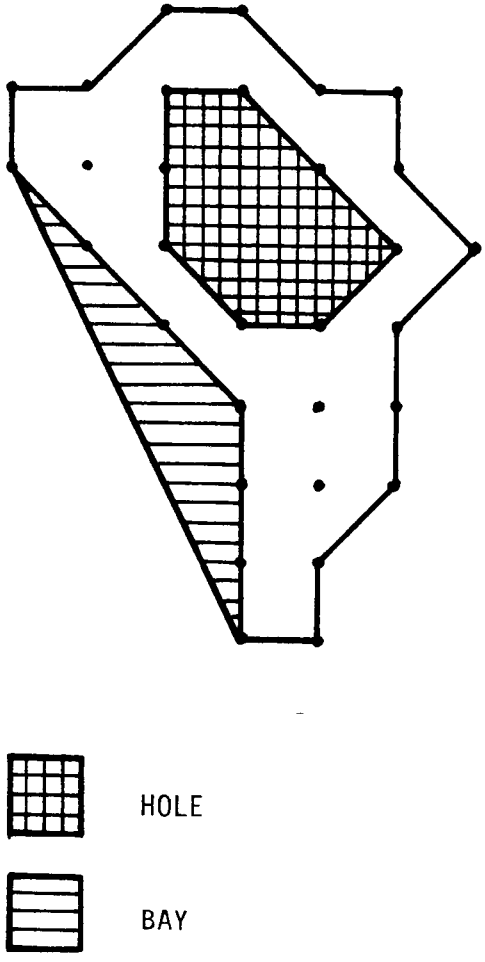


Figure 5.2. Types of connected components in a convex deficiency.

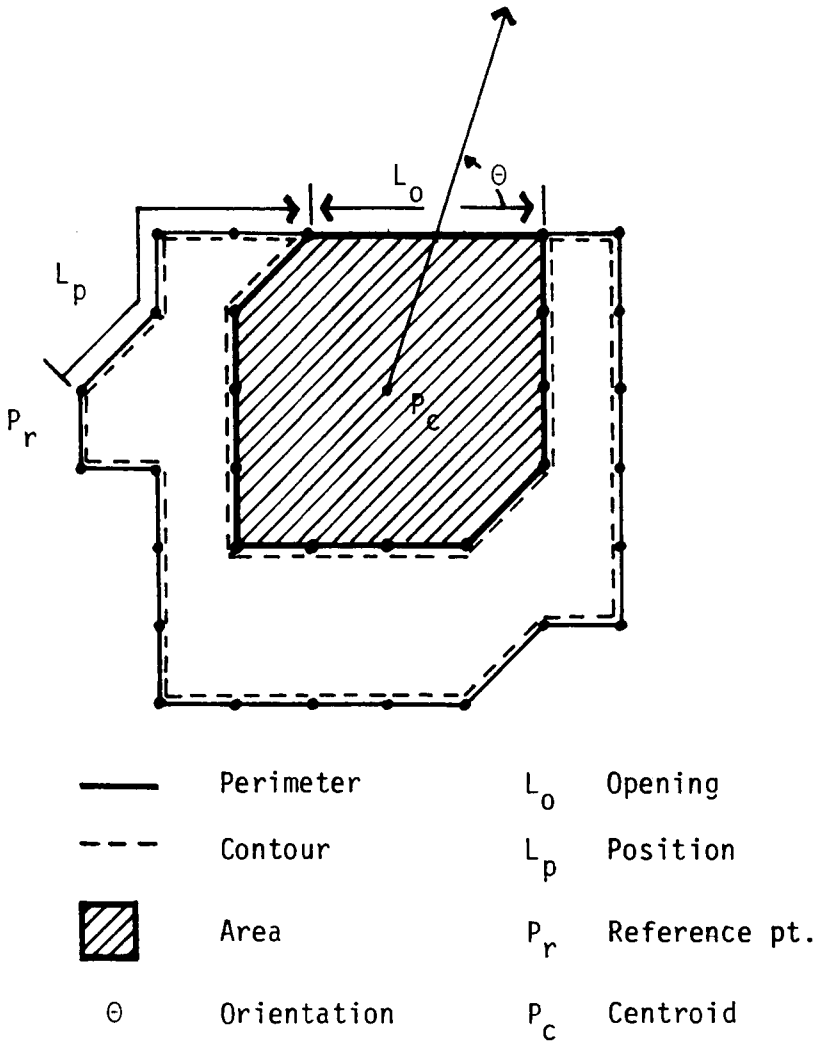


Figure 5.3. Examples of feature descriptors for bays.

changes, can be used to prove that the modified DEFICIENCY algorithm yields a bay-finder-descriptor algorithm that also runs in linear time.

Finally, a simple shape descriptor algorithm is proposed by assembling the algorithms presented in this section into the one that follows.

Algorithm: SHAPE_DESCRIPTOR (Modified CONVEXDEFICIENCY)

Step 1: Obtain the exterior contour along with the contour of the holes, i.e. use algorithm of page 146 in [Pavlidis, 1982]. For each hole in the figure compute the set of features: area, perimeter, centroid, thinness ratio, etc.

Step 2: Obtain the convex hull, i.e. use algorithm CONVEXHULL. Compute the convex hull features: area, perimeter, thinness ratio, etc.

Step 3: Use the modified DEFICIENCY algorithm to obtain all the bays of the figure and describe them with a set of features: area, perimeter, opening, orientation, centroid, relative position, etc.

No solution to the general pattern description problem has been reported yet. Several methods have been proposed in the past with mixed results. So far, the field of feature extraction and description have been an application dependent one. The shape analysis strategy suggested here is intended for those applications where the object to be recognized can be segmented into a single connected component and a simple description of the convex hull and the convex deficiency are sufficient to classify the objects correctly. For a more sofisticated

application, new hole and bay feature descriptors may need to be developed to characterize the more complicated shapes. Nevertheless, the structure of algorithm SHAPE_DESCRIPTOR could probably remain the same.

VI. CONCLUSIONS AND RECOMMENDATIONS

A linear time algorithm for obtaining the convex deficiency of simply 8-connected digital figure has been developed and implemented. This algorithm used no stacks and, when combined with a hole tracking algorithm, it can obtain the convex deficiency of an arbitrary digital figure. It was shown that the contour or boundary of a digital figure is an ordered crossing polygon such that, as the boundary is traversed in a clockwise direction, no interior points lie to the left. A new convex hull invariant condition was also established based on some properties of convex polygons. Also, it was shown that both the holes and the bays of a digital figure can be of great value as shape descriptors in pattern recognition problems involving digital regions. More research is necessary concerning the description of the convex arcs, i.e. connected sets of extreme points, in digital images. Such descriptors, along with a more sofisticated set of features, could be used to complement the approach to shape description that has been suggested in this paper.

LIST OF REFERENCES

LIST OF REFERENCES

Bykat, A., "Convex Hull of a Finite Set of Points in Two Dimensions," Information Processing Letters, vol. 7, 1978, pp. 296-298.

Dévai, F. and Szendrényi, T., "Comments on Convex Hull of a Finite Set of Points in Two Dimensions," Information Processing Letters, vol. 9, 1979, pp. 141-142.

Duda, R. O. and Munson, J. H., "Graphical-Data-Processing Research Study and Experimental Investigation," AD657670, 1967, pp. 4-8.

Graham, R., "An Efficient Algorithm for Determining the Convex Hull of a Planar Set," Information Processing Letters, vol. 1, 1972, pp. 132-133.

Fournier, A. "Comments on Convex Hull of a Finite Set of Points in Two Dimensions," Information Processing Letters, vol. 8, 1979, p. 173.

Ghosh, S. K. and Shyamasundar, R. K., "A Linear Time Algorithm for Obtaining the Convex Hull of a Simple Polygon," Pattern Recognition, vol. 16, 1983, pp. 587-592.

Ghosh, S. K. and Shyamasundar, R. K., "A Linear Time Algorithm for Computing the Convex Hull of an Ordered Crossing Polygon," Pattern Recognition, vol. 17, 1984, pp. 351-358.

Gonzalez, R. C. and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Co., Reading, Massachusetts, 1977, p. 347.

Jarvis, R. A., "On the Identification of the Convex Hull of a Finite Set of Points in the Plane," Information Processing Letters,, vol. 2, 1973, pp. 18-21.

Kim, C. E., "A Linear Time Convex Hull Algorithm for Simple Polygons," Technical Report TR-956, Department of Computer Science, University of Maryland, October, 1980.

Kim, C. E. and Rosenfeld, A., "On the Convexity of Digital Regions," Proc. 5th International Conference on Pattern Recognition, Miami Beach, December, 1980, pp. 1010-1015.

Koplowitz, J. and Jouppi, D., "A More Efficient Convex Hull Algorithm," Information Processing Letters, vol. 7, 1978, pp. 56-57.

Lee, D. T., "On Finding the Convex Hull of a Simple Polygon," Technical Report #80-03-FC-01, Department of Electrical Engineering and Computer Science, Northwestern University, 1980.

McCallum, D. and Avis, D., "A Linear Algorithm for Finding the Convex Hull of a Simple Polygon," Information Processing Letters, vol. 9, 1979, pp. 201-206.

Pavlidis, T. Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, Maryland, 1982, pp. 142-159.

Preparata, F. P., "An Optimal Real-Time Algorithm for Planar Convex Hulls," Communications of the ACM, vol. 22, July, 1979, pp. 402-405.

Rosenfeld, A., "Connectivity in Digital Pictures," Journal of the ACM, vol. 17, 1970, pp. 146-160.

Shamos, M., Problems in Computational Geometry, Carnegie-Mellon University Press, Pittsburgh, 1977.

Sklansky, J., "Measuring Concavity on a Rectangualar Mosaic," I.E.E.E. Trans. on Computers, vol. C-21, 1972, pp. 1355-1364.

Symons, M., "A New Self-Organizing Pattern Recognition System," I.E.E. Conference on Pattern Recognition, Conference Publication 42, I.E.E., London, England, 1968, pp. 11-20.

Toussaint, G. T., "The Convex Hull as a Tool in Pattern Recognition," Proc. AFOSR Workshop in Communication Theory and Applications, Princeton, Massachusetts, September, 1978, pp. 43-46.

Toussaint, G. T., "Pattern Recognition and Geometrical Complexity," Proc. 5th International Conference on Pattern Recognition, Miami Beach, December, 1980, pp. 1324-1347.

Toussaint, G. T and Avis, D., "On a Convex Hull Algorithm for Polygons and Its Application to Triangulation Problems," Pattern Recognition, vol. 15, 1982, pp. 23-29.

APPENDIXES

APPENDIX A

CONTOUR TRACKING ALGORITHMS

Several contour following algorithms have been reported (e.g. [Duda, 1967], [Symons, 1968], [Pavlidis, 1982]). All of them are based on the same basic principles:

1. Obtain an initial border element and call it x_0.
2. Scan the 8-neighbors of x_0 in any direction, say clockwise, until the next border element is found. Call it x_1.
3. Obtain the rest of the elements of the contour of the digital figure in the same manner in which x_1 was obtained.
4. Stop when the sequence x_0x_1 appears again.

Rosenfeld [Rosenfeld, 1970] proved that such algorithms work in finding the contour of a digital image in linear time. Furthermore, he showed that there exists a smallest positive integer m such that the sequence $\{x_0, x_1, \ldots, x_{m-1}\}$ contains every border element of the digital figure S at least once and at most twice, with the latter holding if and only if the element has just two nonconsecutive 4-neighbors in $\bar{S}$; and $x_m = x_0$, and $x_{m+1} = x_1$. Since the scanning of the 8-neighbors of a point involves at most 8 points the complexity of such contour following algorithms is linear in n, where n is the number of boundary points in S.

Let the neighbors of x_i be defined as follows:

$$\begin{matrix} y_7 & y_0 & y_1 \\ y_6 & x_i & y_2 \\ y_5 & y_4 & y_3 \end{matrix}$$

where y_k is the kth neighbor of x_i. Then, a basic contour following algorithm can be implemented as follows.

Algorithm CONTOUR

Step 1: Find a boundary point and call it x_o. Let i = 0 and set the value of j (the initial search direction) accordingly.

Step 2: Scan the 8-neighbors of the ith boundary point, x_i, clockwise and starting with y_j until the next border point, x_{i+1} is found.

Step 3: If $x_i = x_o$ and point x_{i+1} has been flagged, stop. Otherwise flag point x_{i+1} and add it to the contour of R, B(R).

Step 4: If $x_{i+1} = y_{2k}$ then let j = 2k-1 (modulo-8)
If $x_{i+1} = y_{2k+1}$ then let j = 2k-3 (modulo-8)

Step 5: Let i = i + 1 and go to Step 2.

APPENDIX B

PROGRAM LISTINGS

This appendix describes the program that implements the algorithms presented in this paper.

This program was written to run under the RSX-11 operating system on the DEC PDP-11/73. The main program and all the subroutines were written in RSX-11 FORTRAN 77.

The input images are read from disk. The program asks the operator to enter the file name of the input file. Then the file is read into a 64 x 64 byte array called IMAGE where the object points are labeled with a "*" while the background points are labeled with a blank. The main program then calls the subroutine CONTUR which finds the contour of the image and returns the coordinates of the boundary points in a pair of integer arrays X and Y, along with the number of points in the boundary, N. Next, the subroutine CVXHUL is called from the main program to determine the convex hull of the region. CVXHUL returns an ordered list of the convex hull vertices in an integer array called HULL. Subroutine HULDEF is then called to determine the location of the bays. Finally, the results are stored on the byte array IMAGE by routine WRTIMG, which labels the convex hull vertices with a "V", the extreme points with an "E" and the convex deficiency points with a "C". The resulting 64 x 64 image is stored back on disk on a file with the same filename as the input but with the file extension ".HUL".

The main program is listed first followed by the subroutine listings in alphabetical order.

```
      PROGRAM CVXDEF
C
C  THIS PROGRAM FINDS THE CONVEX DEFICIENCY OF A DIGITAL
C  FIGURE BY IMPLEMENTING THE ALGORITHM CONVEXDEFICIENCY.
C  WHICH WAS DESCRIBED IN CHAPTER III.
C
C  INPUT  -- A 64x64 IMAGE READ FROM DISK. WHERE THE OBJECT
C            POINTS ARE LABELED '*'. AND THE BACKGROUND
C            POINTS ARE LABELED ' '.
C  OUTPUT -- A 64x64 IMAGE WHERE THE CONTUR POINTS ARE LABELED
C            AS FOLLOWS:
C            'V'  FOR CONVEX HUL VERTICES
C            'E'  FOR EXTREME POINTS
C            'C'  FOR POINTS ON THE BOUNDARY OF THE
C                 CONVEX DEFICIENCY
C
C  SUBROUTINES CALLED:
C    CONTUR. CVXHUL. HULDEF. WRTIMG.
C
C
      BYTE IMAGE (0:63, 0:63)
      INTEGER X(512), Y(512), HULL(512), PSTART, N
      CHARACTER*20 INFIL, OUTFIL
      PARAMETER DSKLUN=3, LPLUN=5
C
C  INPUT THE NAME OF DISK FILE VIA THE KEYBOARD.
C  OUTPUT FILE HAS THE SAME NAME BUT THE FILE EXTENSION
C  IS ".HUL".
      WRITE (LPLUN, 10)
   10 FORMAT ('$Enter input file: ')
      READ (LPLUN, 20, END=999) INFIL
   20 FORMAT (A)
      OUTFIL = '                    '
      J = INDEX (INFIL, '.')
      OUTFIL = INFIL
      OUTFIL(J:J+3) = '.HUL'
C
C  READ THE IMAGE FROM DISK
      OPEN (UNIT=DSKLUN, FILE=INFIL, STATUS='OLD')
      READ (DSKLUN, 750, END=999, ERR=9999)
     X     ((IMAGE (I, J), J=0,63), I=0,63)
  750 FORMAT (64A1)
      CLOSE (UNIT=DSKLUN)
C
C  GET THE CONTOUR OF THE DIGITAL FIGURE
      CALL CONTUR (IMAGE, X, Y, '*', '$', N)
C
C  GET THE CONVEX HULL OF THE DIGITAL FIGURE
      CALL CVXHUL (X, Y, HULL, N, PSTART)
C
C  GET THE CONVEX DEFICIENCY OF THE DIGITAL FIGURE
      CALL HULDEF (X, Y, HULL, N, PSTART)
```

```
C
C  WRITE THE RESULTS ON THE IMAGE
      CALL WRTIMG (IMAGE, X, Y, HULL, N, 'E', 'V', 'C')
C
C  STORE THE RESULTING IMAGE ON DISK
      OPEN (UNIT=DSKLUN, FILE=OUTFIL, STATUS='NEW')
      WRITE (DSKLUN, 750)
     X       ((IMAGE (I, J), J=0,63), I=0,63)
      CLOSE (UNIT=DSKLUN)
  999 STOP
 9999 STOP 'Error reading input file.'
      END




      SUBROUTINE CONTUR (IMAGE, X, Y, OBJECT, BORDER, N)
C
C  THIS ROUTINE FINDS THE CONTOUR OF A DIGITAL FIGURE BY
C  IMPLEMENTING THE ALGORITHM CONTOUR, WHICH WAS DESCRIBED
C  IN APPENDIX A.
C
C  VARIABLES USED:
C    IMAGE   BYTE ARRAY CONTAINING THE FIGURE
C     X, Y   COORDINATES OF  THE BOUNDARY PIXELS
C   OBJECT   PIXEL VALUE OF THE INTERIOR POINTS
C   BORDER   PIXEL VALUE OF THE BOUNDARY POINTS
C        N   NUMBER OF PIXELS IN THE BOUNDARY
C
C  NXTDIR IS THE SEARCHING DIRECTION:
C           7 0 1       + - y
C           6 * 2       |
C           5 4 3       x
C
      BYTE IMAGE (0:63, 0:63), OBJECT, BORDER
      INTEGER STARTX, STARTY, X(*), Y(*), COUNT,
     X        NEXTX(0:7), NEXTY(0:7), IX, IY, NXTDIR
      LOGICAL DONE
      DATA NEXTX /-1,-1,0,1,1,1,0,-1/,
     X     NEXTY /0,1,1,1,0,-1,-1,-1/
C
C  GET THE FIRST POINT OF THE BOUNDARY
      STARTX = 32
      DO 100 STARTY = 0, 63
          IF (IMAGE (STARTX, STARTY) .EQ. OBJECT) GO TO 200
  100 CONTINUE
      STOP 'Failed to find the object.'
  200 N = 0
      COUNT = 0
      NXTDIR = 7
```

```
      IX = STARTX + NEXTX(NXTDIR)
      IY = STARTY + NEXTY(NXTDIR)
      DONE = .FALSE.
C
C  LOOK FOR A POINT THAT IS LEFT-MOST FROM THE INITIAL
C  DIRECTION. THEN KEEP MOVING CLOCKWISE UNTIL AN
C  OBJECT POINT IS FOUND. STOP WHEN BACK TO STARTING POINT
C  (DONE=.TRUE.) AND NO MORE BORDER POINTS CAN BE FOUND.
  300 IF ((DONE) .AND. (IMAGE(IX,IY) .EQ. BORDER)) RETURN
      IF ((IX .GE. 0) .AND. (IX .LE. 63) .AND.
     X    (IY .GE. 0) .AND. (IY .LE. 63) .AND.
     X    ((IMAGE(IX,IY) .EQ. OBJECT) .OR.
     X    (IMAGE(IX,IY) .EQ. BORDER))) THEN
          IMAGE(IX,IY) = BORDER
          N = N + 1
          IF (N .GT. 512) STOP 'Too many points in boundary'
          X(N) = IX
          Y(N) = IY
          DONE = .FALSE.
          IF ((IX .EQ. STARTX) .AND. (IY .EQ. STARTY))
     X        DONE = .TRUE.
          NXTDIR = ((NXTDIR + 6) .OR. 1) .AND. 7
          COUNT = 0
      ELSE
          NXTDIR = (NXTDIR + 1) .AND. 7
          COUNT = COUNT + 1
          IF (COUNT .GT. 8) STOP 'Isolated pixel.'
      END IF
      IX = X(N) + NEXTX (NXTDIR)
      IY = Y(N) + NEXTY (NXTDIR)
      GO TO 300
      END




      SUBROUTINE CVXHUL (X, Y, HULL, N, PSTART)
C
C  THIS SUBROUTINE DETERMINES THE CONVEX HULL OF A
C  DIGITAL IMAGE BY IMPLEMENTING THE ALGORITHM CONVEXHULL.
C  WHICH WAS DESCRIBED IN CHAPTER III.
C
C  VARIABLES USED:
C        X  ARRAY CONTAINING THE X-COORDINATES OF THE POINTS
C        Y  ARRAY CONTAINING THE Y-COORDINATES OF THE POINTS
C     HULL  LINKED LIST CONTAINING THE CONVEX HULL
C        N  NUMBER OF POINTS
C   PSTART  ON OUTPUT, POINT WITH MINIMUM X-COORDINATE
C
C
```

```
      INTEGER X(*), Y(*), HULL(*), N, PSTART, PXMIN,
     X        PXMAX, PYMIN, PYMAX, PI, PJ, PK, DX, DY
C
C  FIND THE POINTS WITH MINIMUM AND MAXIMUM X-COORDINATES AND
C  CALL THEM PXMIN AND PXMAX RESPECTIVELY.
C  FIND THE POINTS WITH MINIMUM AND MAXIMUM Y-COORDINATES AND
C  CALL THEM PYMIN AND PYMAX RESPECTIVELY.
      PXMIN = 1
      PXMAX = 1
      PYMIN = 1
      PYMAX = 1
      DO 100 PI = 1, N
          IF (((X(PI) .EQ. X(PXMIN)) .AND.
     X         (Y(PI) .LT. Y(PXMIN))) .OR.
     X         (X(PI) .LT. X(PXMIN))) PXMIN = PI
          IF (((X(PI) .EQ. X(PXMAX)) .AND.
     X         (Y(PI) .GT. Y(PXMAX))) .OR.
     X         (X(PI) .GT. X(PXMAX))) PXMAX = PI
          IF (((Y(PI) .EQ. Y(PYMIN)) .AND.
     X         (X(PI) .GT. X(PYMIN))) .OR.
     X         (Y(PI) .LT. Y(PYMIN))) PYMIN = PI
          IF (((Y(PI) .EQ. Y(PYMAX)) .AND.
     X         (X(PI) .LT. X(PYMAX))) .OR.
     X         (Y(PI) .GT. Y(PYMAX))) PYMAX = PI
  100 CONTINUE
C
C  INITIALIZE THE LINK LIST
      HULL(1) = N
      DO 200 I = 1, N - 1
          HULL(I+1) = I
  200 CONTINUE
C
C  ALGORITHM STARTS AT PXMIN AND SCANS THE POINTS IN THE
C  CLOCKWISE DIRECTION.
C       PI POINTS TO THE KTH POINT IN THE CONVEX HULL
C       PJ POINTS TO THE K+1TH POINT IN THE CONVEX HULL
C       PK POINTS TO THE K+2TH POINT IN THE CONVEX HULL
C  IF THERE ARE ONLY TWO POINTS, STOP.
  300 PSTART = PXMIN
      PI = PSTART
      PJ = IMOD (PI, N) + 1
      PK = IMOD (PJ, N) + 1
      DX = +1
      DY = +1
      IF (PSTART .EQ. PYMAX) DY = -1
      IF (N .LE. 2) RETURN
C
C  CHECK FOR MONOTONICITY. STOP IF BACK TO PXMIN
  400 IF (PJ .EQ. PYMAX) DY = -1
      IF (PJ .EQ. PXMAX) DX = -1
      IF (PJ .EQ. PYMIN) DY = +1
      IF (PJ .EQ. PXMIN) RETURN
```

```
C
C   SKIP PRINCIPAL EXTREME POINTS.
          IF ((PJ .EQ. PYMIN) .OR. (PJ .EQ. PXMAX) .OR.
         X     (PJ .EQ. PYMAX)) THEN
                PI = PJ
                PJ = PK
                PK = IMOD (PK, N) + 1
                GO TO 400
          END IF
C
C   COMPUTE THE FUNCTION CONVEX = S(I,J,K) WHERE
C     S (I,J,K) < 0    ==>  SEQUENCE PI,PJ,PK IS CONCAVE
C               = 0    ==>  SEQUENCE PI,PJ,PK IS COLLINEAR
C               > 0    ==>  SEQUENCE PI,PJ,PK IS CONVEX
C   MONOTONICITY TEST:
C     MX  >=  0        ==>  SEQUENCE XI,XJ,XK IS MONOTONIC
C     MY  >=  0        ==>  SEQUENCE YI,YJ,YK IS MONOTONIC
          MX = X(PK) - X(PJ)
          MY = Y(PK) - Y(PJ)
          CONVEX = MX * (Y(PJ) - Y(PI)) - MY * (X(PJ) - X(PI))
          IF (DX .LT. 0) MX = -MX
          IF (DY .LT. 0) MY = -MY
          IF (CONVEX .LT. 0) THEN
C             DELETE PJ
                HULL (PK) = PI
                IF ((PI .EQ. PXMAX) .OR. (PI .EQ. PXMIN) .OR.
         X      (PI .EQ. PYMAX) .OR. (PI .EQ. PYMIN)) THEN
C                  ADVANCE ONE POINT
                      PJ = PK
                      PK = IMOD (PK, N) + 1
                ELSE
C                     BACKWARD ONE POINT
                      PJ = PI
                      PI = HULL (PI)
                END IF
                GO TO 400
          END IF
          IF ((MX .GE. 0) .AND. (MY .GE. 0)) THEN
                IF (CONVEX .EQ. 0) THEN
C                     DELETE PJ.
                      HULL (PK) = PI
                ELSE
C                     ADVANCE ONE POINT.
                      PI = PJ
                END IF
                PJ = PK
                PK = IMOD (PK, N) + 1
                GO TO 400
```

```
        ELSE
C           DELETE PK AND ADVANCE ONE POINT.
            PK = IMOD (PK, N) + 1
            HULL (PK) = PJ
            GO TO 400
        END IF
        END




        SUBROUTINE HULDEF (X, Y, HULL, N, PSTART)
C
C   THIS ROUTINE OBTAINS THE CONVEX DEFICIENCY OF A DIGITAL IMAGE
C   AFTER THE CONVEX HULL HAS BEEN DETERMINED. IT IMPLEMENTS THE
C   ALGORITHM DEFICIENCY, WHICH WAS DESCRIBED IN CHAPTER III OF
C   THIS PAPER.
C
C   VARIABLES USED:
C         X   ARRAY CONTAINING THE X-COORDINATES OF THE POINTS
C         Y   ARRAY CONTAINING THE Y-COORDINATES OF THE POINTS
C      HULL   LINKED-LIST CONTAINING THE CONVEX HULL
C         N   NUMBER OF POINTS
C    PSTART   STARTING POINT OF THE LINKED-LIST HULL
C
C
        INTEGER X(*), Y(*), HULL(*), PSTART, N, PI, PJ, PK,
       X          S, XI, YI, XK, YK, XINC, YINC
C
C   STARTING WITH PSTART, SCAN CONVEX HULL VERTICES IN
C   COUNTER-CLOCKWISE ORDER. IF A PAIR OF VERTICES ARE
C   NOT CONSECUTIVE, TEST FOR A CONVEX DEFICIENCY.
        PK = PSTART
   100  PI = HULL (PK)
        IF (IMOD (PI, N) + 1 .NE. PK) THEN
C           DETERMINE THE COORDINATES OF THE END POINTS
C           OF THE IMAGINARY LINE M'
            IF (IABS (X(PK) - X(PI)) .GT.
       X         IABS (Y(PK) - Y(PI))) THEN
                XINC = 0
                YINC = +1
                IF (X(PK) .GT. X(PI)) YINC = -1
            ELSE
                YINC = 0
                XINC = -1
                IF (Y(PK) .GT. Y(PI)) XINC = +1
            END IF
            XI = X(PI) + XINC
            YI = Y(PI) + YINC
            XK = X(PK) + XINC
            YK = Y(PK) + YINC
```

```
C       SCAN ALL POINTS BETWEEN PI AND PK. LABEL ANY POINT
C       INTERIOR TO LINE M' WITH THE VALUE -1. OTHERS
C       WITH THE VALUE 0.
            PJ = IMOD (PI, N) + 1
  200       HULL (PJ) = 0
            S = (XK - X(PJ)) * (Y(PJ) - YI) -
           X        (YK - Y(PJ)) * (X(PJ) - XI)
            IF (S .LE. 0) HULL (PJ) = -1
            PJ = IMOD (PJ, N) + 1
            IF (PJ .NE. PK) GO TO 200
         END IF
         PK = PI
         IF (PK .NE. PSTART) GO TO 100
         RETURN
         END




         SUBROUTINE WRTIMG (IMAGE, X, Y, HULL, N,
        X                   EXTREM, VERTEX, BAY)
C
C  THIS ROUTINE WRITES THE RESULT OF THE CONVEX DEFICIENCY
C  ALGORITHM TO THE IMAGE.
C
C  VARIABLES USED:
C    IMAGE  BYTE ARRAY CONTAINING THE DIGITAL FIGURE
C     X, Y  COORDINATES OF THE BOUNDARY POINTS
C     HULL  LINKED LIST CONTAINING THE CONVEX HULL AND
C           CONVEX DEFICIENCY
C        N  NUMBER OF POINTS IN BOUNDARY
C   EXTREM  PIXEL VALUE OF EXTREME POINTS
C   VERTEX  PIXEL VALUE OF CONVEX HULL VERTICES
C      BAY  PIXEL VALUE OF CONVEX DEFICIENCY
C
         INTEGER X(*), Y(*), HULL(*), N
         BYTE IMAGE (0:63, 0:63), EXTREM, VERTEX, BAY
C
C  HULL(I) CONTAINS THE LABEL CORRESPONDING TO THE ITH
C  POINT ON THE BOUNDARY. IF HULL(I) > 0 THEN ITH POINT
C  IS A CONVEX HULL VERTEX. IF HULL(I) = 0 THEN ITH POINT
C  IS AN EXTREME POINT. IF HULL(I) < 0 THEN ITH POINT
C  FORMS PART OF THE CONVEX DEFICIENCY.
         DO 100 I = 1, N
            IF (HULL(I) .GT. 0) IMAGE (X(I), Y(I)) = VERTEX
            IF ((HULL(I) .EQ. 0) .AND.
        X       (IMAGE (X(I), Y(I)) .NE. VERTEX))
        X           IMAGE (X(I), Y(I)) = EXTREM
```

```
      IF ((HULL(I) .LT. 0) .AND.
     X         (IMAGE (X(I), Y(I)) .NE. EXTREM) .AND.
     X         (IMAGE (X(I), Y(I)) .NE. VERTEX))
     X             IMAGE (X(I), Y(I)) = BAY
100  CONTINUE
     RETURN
     END
```

VITA

Juan Antonio Herrera Ball was born in Caracas, Venezuela, on October 30, 1960. He attended elementary schools in that city and graduated from Colegio Santiago de león de Caracas in July, 1978.

The following September he entered Universidad Simón Bolívar. He then transferred to the University of Tennessee, Knoxville, in January, 1979. In August, 1982, he received a Bachelor of Science degree in Electrical Engineering and a Bachelor of Arts degree with a major in Mathematics, both with high honors. In September, 1979, he began study toward a Master's degree. He received the Master of Science degree with a major in Computer Science in March, 1985.

The author is a member of the Institute of Electrical and Electronic Engineers, Eta Kappa Nu, and Tau Beta Pi. Mr. Herrera is currently enrolled in the University of Tennessee, Knoxville, working toward a Ph.D. degree.