Pattern Recognition, Vol. 24, No. 11, pp. 1019–1035, 1991
Printed in Great Britain

0031-3203/91 $3.00 + .00
Pergamon Press plc

FINDING THE KERNEL OF PLANAR SHAPES

R. BORNSTEIN and A. M. BRUCKSTEIN*
Department of Computer Science, Technion—Israel Institute of Technology, 32000 Haifa, Israel

(*Received* 13 *March* 1990; *in revised form* 5 *March* 1991; *received for publication* 14 *March* 1991)

Abstract—The kernel of a planar shape is the locus of interior points from which all boundary points can be seen. This paper discusses an algorithm for determining the kernel of a planar shape. To find the kernel we could first ask what is the interior region seen from each boundary point. The intersection of these regions corresponding to all boundary points is, by definition, the kernel. Since it would be quite impractical to implement the kernel-finding algorithm just described, we should first determine interior regions that are jointly seen from boundary points that belong to boundary fragments. Based on this idea, a practical algorithm can be designed. It is an efficient way to intersect regions in the plane induced by suitably defined boundary fragments and determined via visibility constraints. It is shown that for a wide class of planar shapes, the resulting procedure is computationally efficient. In particular, for the case of a polygon with N edges, the algorithm has a time complexity of $O(N)$ and hence is optimal. It is usually more efficient and in the worst case at least as good as a previously proposed $O(N)$ algorithm.

Computational geometry Star-shapedness Kernel Planar shape analysis

1. INTRODUCTION

Many industrial vision applications require description and analysis of planar shapes. In this context it is sometimes important to determine, for a given shape, the locus of interior points from which all boundary points can be seen. If this locus is not empty the planar shape is called star-shaped and the locus of points is called its kernel. The boundary of star-shaped figures may then be described by an $r(\phi)$ function, a description used by certain pattern recognition and classification algorithms.

Since polygonal approximations to planar shapes are quite common, it is natural to consider the problem of star-shapedness and the determination of the kernel of polygonal shapes. An algorithm for the determination of the kernel of a polygon was proposed by Lee and Preparata.[1] For a given polygon with N edges their algorithm runs in $O(N)$ time and since a scan of all the polygon edges is necessary, it is optimal. In some cases, it is desirable to describe a planar shape boundary by parametric curves other than straight line segments. As an example, the boundary of a planar shape could be described as a spline curve determined via given control points. Given such a non-polygonal shape, the question of how to determine its kernel arises and here we consider this general problem. We first turn to the formal definition of the problem using basic concepts of topology and differential geometry. Theoretical considerations impose several restrictions on the class of shapes, restrictions defined in terms of natural constraints on the shapes boundary. These constraints are mild and met by most shapes we would encounter in practice. The analysis suggests that partitioning of the shape boundary into concave and convex fragments is useful for kernel determination. The contribution of these fragments to the kernel boundary is then investigated, yielding straightforward rules which are exploited by the kernel-finding algorithm. Sequential application of these rules is used as a preprocessing stage to a kernel extraction algorithm which resembles the one devised by Lee and Preparata for polygons. When the input shape is a polygon with N edges and M concave fragments, our fast $O(N)$ preprocessing stage reduces the number of input elements to the Lee and Preparata type algorithm to J where $2M \leq J \leq N$. Therefore, while Lee and Preparata's original algorithm considers all N input elements and thus runs in $O(N)$ time, a slightly modified version of their algorithm, used as the kernel extraction stage, will run in $O(J)$ time. Since this stage is the most time consuming part, for some polygons, the improvement in the overall runtime of the new algorithm over the original one is significant. Also, since the preprocessing algorithm is very efficient, its overhead time consumption is practically negligible and is certainly worth the potential of savings.

In the body of our paper only main theorems and result are presented. Full details are presented in reference (6).

2. BACKGROUND, MOTIVATION AND MATHEMATICAL PRELIMINARIES

In reference (2) it was shown that the kernel of a polygon is the intersection of all half planes lying to

* Author to whom correspondence should be addressed.

the left of all polygon edges when the boundary is traversed counterclockwise. Based on this result, Shamos and Hoey[3] proposed an algorithm which intersects N half planes having a time complexity of $O(N \log N)$. Lee and Preparata[1] improved this algorithm by taking advantage of the natural order in which the half planes are arranged. Their algorithm runs in $O(N)$ time, and since scanning of all polygon edges is necessary, it is optimal. However, every edge of the polygon receives, on average, the same treatment and since this treatment involves line intersection and point location operations, the constant in the complexity expression is relatively high. Additionally, an extra test during the algorithm is necessary in order to ensure that for some cases the algorithm will terminate the FAIL, i.e. will conclude that the polygon is not star-shaped, before doing an $O(N^2)$ work. This test also involves the same type of time-consuming operations. As a by-product of analysing the more general problem of determining the kernel of non-polygonal shapes, we will show that in some cases the heavy treatment of some of the polygon edges may be avoided, yielding a shorter execution time. Furthermore, the test mentioned above may be replaced by a stand-alone procedure which, for certain shapes, rapidly finds that they are not star-shaped at all, making unnecessary the further application of the kernel extraction algorithm. In this work we first deal with the problem of determining the kernel of an almost arbitrary planar shape. We cannot deal with arbitrary shapes because there exist shapes whose boundary is arbitrarily complex, and in such cases there is little hope of finding a finite procedure for kernel determination. In order to simplify the problem, we introduce some rather natural restrictions on the complexity of the shape boundary in terms of regularity, total arc length, number of convex and concave fragments and the value of a fragment turn angle.

In the case of planar shapes, the left half planes that were considered for polygons are replaced by left regions of each boundary point, having a somewhat similar meaning. As for polygons, the intersection of all these half regions forms the kernel of the shape. However, while for polygons the number of different left half planes was finite, in a shape of the class we are dealing with there may be an infinite number of different left regions. Hence, we are led to consider left regions of entire boundary fragments. To have an algorithm that terminates in a finite number of steps, we need the number of these fragments to be finite. In order to follow the arguments in this work we need some simple background in both topology and differential geometry. A brief overview of these topics is thus first presented.

2.1. *Some topological definitions*

The n-dimensional Euclidian space will be denoted E^n. As special cases we have E^1 as the real number axis and E^2 as the x–y plane.

With respect to a subset S of the plane, each point p has one of the following three properties:

(a) p is interior to S if $p \in S$ and there is a neighborhood of p that is contained in S. The set of all the points interior to S is the interior of S and will be denoted $S/\partial S$;

(b) p is exterior to S if $p \notin S$ and there is a neighborhood of p that is disjoint from S. The set of all the points exterior to S is the exterior of S and will be denoted $\bar{S}$;

(c) p is a boundary point of S if p is neither interior nor exterior to S. The set of all boundary points of S is the boundary of S and will be denoted ∂S.

2.2. *Some basic differential geometry*

Let $I \in E^1$. A planar curve is a mapping $c: I \to E^2$. For each $t \in I$ we have

$$c(t) = (x(t), y(t)). \tag{2.1}$$

Taking the first derivative with respect to t we obtain the tangent vector

$$\mathbf{c}'(t) = (x'(t), y'(t)). \tag{2.2}$$

Clearly $\mathbf{c}'(t)$ exists if $x'(t)$ and $y'(t)$ are defined. If $\mathbf{c}'(t)$ exists and is not zero, we say that $c(t)$ is a regular point. A curve $c: I \to E^2$ is regular if $\forall t, t \in I$, $c(t)$ is regular.[4]

A piecewise regular curve is a continuous function $c: [a, b] \to E^2$ together with a partition

$$a = t_0 < t_1 < \cdots < t_{L-1} < t_L = b$$

of $[a, b]$ such that $c[t_j, t_{j+1}]$ is regular, $0 \le j \le L - 1$. The points $c(t_j)$ are called the corners of c. A piecewise regular curve $c[a, b]$ is closed if $c(a) = c(b)$, and is simple if $c[a, b]$ is one-to-one.

The arc length s measured from a fixed point t_0 is given by

$$s(t) = \int_{t_0}^{t} \|\mathbf{c}'(\tau)\| \, \mathrm{d}\tau. \tag{2.3}$$

The definition of s also applies to piecewise regular curves.

We may reparametrize the curve using its arc length. Henceforth we will assume that this is the case. In this case, the total arc length of a curve $c[a, b]$, $a < b$, is simply $L = b - a$.

The direction of motion when moving from $c(a)$ to $c(b)$, $a < b$, will be denoted as the positive direction. Traversing $c(s)$ in this direction, we move from $c(s^-)$ to $c(s^+)$. Analogously, scanning c in the opposite direction will be referred to as the negative direction.

For regular curves, the tangent vector is a unit length vector

$$\mathbf{c}'(s) = (x'(s), y'(s)) \tag{2.4}$$

or in other words, $\mathbf{c}'(s)$ is a point on the unit circle. This fact leads to another representation of $\mathbf{c}'(s)$.

For a regular curve $c:[a, b] \rightarrow E^2$ there exists a continuous, piecewise differentiable function $\theta:[a, b] \rightarrow E^1$ such that

$$\mathbf{c}'(s) = (\cos(\theta(s)), \sin(\theta(s))). \tag{2.5}$$

Since θ is continuous, we can define the total angular change along the curve as the turn angle of c:

$$\angle(c[a, b]) = \theta(b) - \theta(a). \tag{2.6}$$

This value is independent of the reference value for θ.

At the corners of a piecewise regular curve θ may be undefined. Nevertheless, the exterior angle α_j at a corner point $c(s_j)$ is the oriented angle from $\mathbf{c}'(s_j^-)$ to $\mathbf{c}'(s_j^+)$. Thus we define

$$\theta(s_j^+) = \theta(s_j^-) + \alpha_j. \tag{2.7}$$

We will assume that for each corner point $c(s_j)$, $-\pi < \alpha_j < +\pi$.

$c(s_j)$ is a convex corner point, if $0 < \alpha_j < +\pi$.

$c(s_j)$ is a concave corner point, if $0 > \alpha_j > -\pi$.

Using the tangent vector we can define the tangent line $g(s)$ as the curve

$$c(s) + t\mathbf{c}'(s), t \in E^1. \tag{2.8}$$

Consider also the two rays composing $g(s)$ named the backward ray and the forward ray, denoted $r_b(s)$ and $r_f(s)$, which are the curves

$$c(s) - t\mathbf{c}'(s)$$

and

$$c(s) + t\mathbf{c}'(s), t \geq 0, \tag{2.9}$$

respectively.

Note that at a corner point $c(s_j)$, the tangents $\mathbf{c}'(s_j^-)$ and $\mathbf{c}'(s_j^+)$ may be different and in this case, two different tangent lines exist.

In case of a simple curve $c[a, b]$, given a point q on the curve, $q = c(s)$, s is uniquely determined and we shall refer to it as s_q.

2.3. *Additional definitions*

The direction of $g(s)$, $r_b(s)$ and $r_f(s)$ is defined as the direction of $\mathbf{c}'(s)$. The left and right sides of $g(s)$ are thus determined.

It is agreed that while scanning a shape boundary in the positive direction, the interior near a boundary point c is to the left of the tangent line at that point.

Let $LINE\ (a, b)$ be the straight line passing through a and b.

Let $SEG[a, b]$, $SEG(a, b)$, $SEG[a, b)$, $SEG(a, b]$ be the close, open and half open straight line segments between a and b, respectively.

Let $RAY[a, b)$, $RAY(a, b)$ be the close and open rays, respectively, which emanate from a, pass through b and go to infinity.

Let us define now two predicates which play an important role in further definitions and theorems.

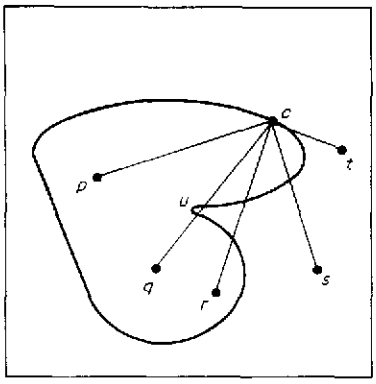


Fig. 1. An illustration of *INT* and *LEFT* predicates.

Both predicates refer to a planar shape S and a point c on its boundary.

2.3.1. *Definition.* $INT(p, c)$ is true if and only if either $p \equiv c$ or $SEG(p, c) \subset S/\partial S$.

2.3.2. *Definition.* $LEFT(p, c)$ is true if and only if either $p \equiv c$ or $\exists m$, $m \in SEG[p, c)$, such that $INT(m, c)$.

These definitions are illustrated in Fig. 1. For instance, $INT(p, c)$, $INT(u, c)$, $-INT(q, c)$, $-INT(t, c)$. $LEFT(p, c)$, $LEFT(q, c)$, $LEFT(s, c)$, $-LEFT(t, c)$.

In addition, let us define convex and concave curves.

2.3.3. *Definition.* A regular curve $c[a, b]$ is

concave, if $\theta(s)$, $a \leq s \leq b$, is a monotone non-decreasing function and $\theta(b) - \theta(a) > 0$.

concave, if $\theta(s)$, $a \leq s \leq b$, is a monotone non-increasing function and $\theta(b) - \theta(a) < 0$.

The above definition also applies to piecewise regular curves if we set the value of θ at each corner point $c(s_j)$ to any value between $\theta(s_j^-)$ and $\theta(s_j^+)$. For instance,

$$\theta(s_j) = \tfrac{1}{2}(\theta(s_j^-) + \theta(s_j^+)). \tag{2.10}$$

3. THE KERNEL OF A PLANAR SHAPE

A simple shape is a shape whose boundary is a simple closed curve which partitions the plane into two disjoint regions, the interior and the exterior of the shape (Jordan curve theorem). In this work we shall deal with simple shapes only.

3.0.1. *Definition.* The kernel of a planar shape S, $K(S)$, is the locus of all points p that satisfy:

$$\forall c, c \in \partial S, INT(p, c). \tag{3.1}$$

A planar shape that has a nonempty kernel is called star-shaped (Fig. 2).

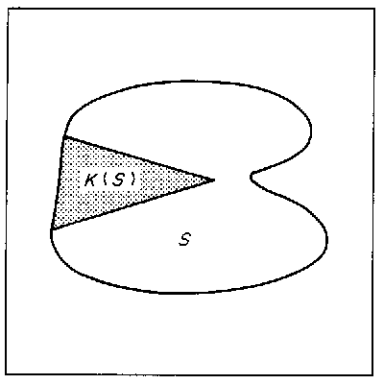


Fig. 2. A planar shape and its kernel.

Note that according to the definition, there are shapes for which points on the kernel boundary belong to the kernel (as with a circle, for instance) while in others, kernel boundary points are excluded from the kernel (as with polygons). In order to simplify matters, when dealing with theoretical aspects of the problem we shall treat all kernel boundary points as part of the kernel. In practice, however, we shall exclude these points from the kernel with the possibility of thereby incurring a small inaccuracy.

Define a convex region as a region T which for every two points $a, b \in T$, $SEG[a, b] \subset T$. Define a convex shape to be a bounded convex region. A convex shape S is star-shaped and $K(S) = S$.

Since finding the kernel in a straightforward way using its definition may be a hard task, we shall use the following theorem. This theorem and its successor certainly apply for a class of planar shapes named X-type shapes which will be defined later.

3.0.2. *Theorem.* $p \in K(S)$ if and only if $\forall c, c \in \partial S$, $LEFT(p, c)$. However, the proof for the latter theorem assumes the validity of the next theorem.

3.0.3. *Theorem.* Let S be a planar shape. Let c and d be any two points in E^2 such that $SEG[d, c] \not\subset \partial S$. Then either

$\exists m, m \in SEG[d, c)$, such that $SEG(m, c) \subset \partial S$ or

$\exists m, m \in SEG[d, c)$, such that $SEG(m, c) \cap \partial S = \emptyset$.

Consider the two pathological shapes illustrated in Fig. 3. They certainly do not satisfy Theorem 3.0.3. Therefore, we must restrict the class of planar shapes we are dealing with to X-type shapes.

3.0.4. *Definition.* An X-type curve is a piecewise regular, simple, closed curve having a finite total arc length and a finite number of convex and concave fragments, each with a finite value turn angle. An X-type shape is a planar shape whose boundary is an X-type curve.

For a given boundary point c, let us define the

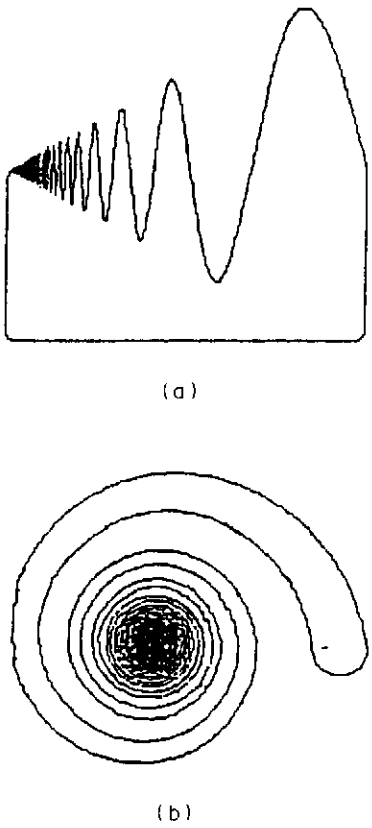


Fig. 3. Two pathological shape boundaries: (a) a boundary with an infinite number of convex and concave fragments; (b) an endless curling spiral exhibits a fragment with an infinite value turn angle.

locus of points p satisfying $LEFT(p, c)$ as the left region of c:

$$LR(c) = \{p | LEFT(p, c)\}. \tag{3.2}$$

It can be shown that $LR(c(s))$ is the region whose boundary is formed by $r_b(s^-)$ and $r_f(s^+)$ (Fig. 4).

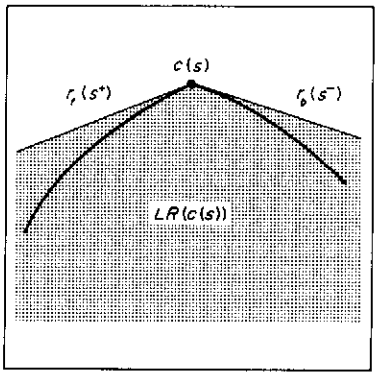


Fig. 4. A demonstration of a left region of a point, in this case, a convex boundary point. For a given boundary point c, the left region of c is the locus of points p satisfying $LEFT(p, c)$.

Then, from Theorem 3.0.2, it follows that

$$K(S) = \bigcap_{c \in \partial S} LR(c). \tag{3.3}$$

However, this might require the intersection of an infinite number of different left regions. Hence, we should try to cluster several left regions to form the left region of a boundary fragment:

let $\{R_i\}$ be a partition of ∂S into non-overlapping boundary fragments such that $\bigcup_i R_i = \partial S$.

Define the left region of a fragment:

$$LR(R_i) = \bigcap_{c \in R_i} LR(c). \tag{3.4}$$

Then

$$K(S) = \bigcap_i LR(R_i). \tag{3.5}$$

A finite partition is necessary in order to make the solution computationally feasible. So the current question is, how to partition ∂S?

4. PARTITIONING THE SHAPE BOUNDARY INTO FRAGMENTS

Theoretically, any finite partition of ∂S may be appropriate, so why should we choose one over the other? One criterion of goodness could be the number of fragments. A second motivation could be the complexity of the left regions formed due to the partition and the effort needed to intersect them. A natural partition is into concave and convex boundary fragments. Their number is finite due to the definition of an X-type curve and their left regions are relatively simple and have a fixed structure as we shall see.

Let us denote the transition points from concavity to convexity and vice versa in the positive direction as e-type and d-type points, respectively. Thus, a convex fragment R_i is delimited by e_i and d_i while a concave fragment R_i is delimited by d_i and e_{i+1}. Two problems arise:

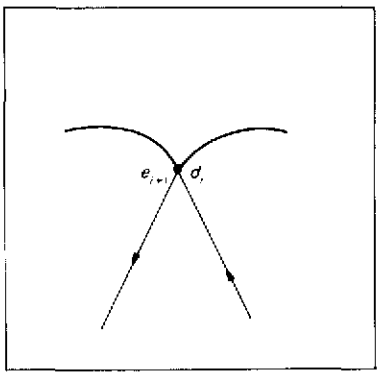
(1) What should be done with a straight line segment common to both a concave fragment and a convex fragment?

(2) In case an e-type point or a d-type point are corner points, is it important how θ is defined there?

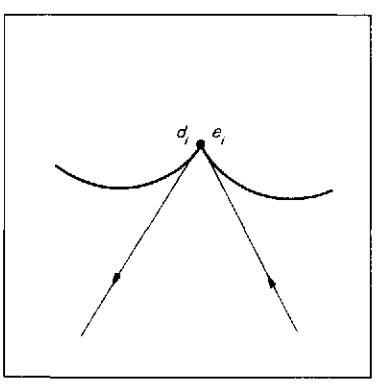
As for the first issue, any point along the straight line segment may serve as a transition point. It will be evident later that shorter convex fragments are preferable. Thus, the straight line segment will be defined to be part of the concave fragment. The answer to the second question is positive in order to describe efficiently the various cases that may occur.

Thus, if $c(s)$ is an e-type point,

$$\theta(s) = \begin{cases} \theta(s^+), & \text{if } \theta(s^+) < \theta(s^-) \\ \theta(s^-), & \text{otherwise} \end{cases}$$



(a)



(b)

Fig. 5. The case when a point may be both e-type and d-type: (a) a concave fragment consists of a single point; $d_i = e_{i+1}$; (b) a convex fragment consists of a single point; $e_i = d_i$. The value of θ at such a point will change depending on the current meaning of the point (either e-type or d-type).

and if $c(s)$ is a d-type point,

$$\theta(s) = \begin{cases} \theta(s^-), & \text{if } \theta(s^+) < \theta(s^-) \\ \theta(s^+), & \text{otherwise.} \end{cases} \tag{4.1}$$

However, a point may be both an e-type and a d-type point, introducing ambiguity in the value of θ (Fig. 5). Depending on the context, such a point will be regarded as either an e-type or a d-type point. Thus, the ambiguity is resolved.

Let us turn now to the analysis of concave and convex fragments. The curves that we refer to in all further theorems are boundary fragments of an X-type shape S.

4.1. *The left region of a concave fragment*

Let R_i be a concave fragment. We distinguish between two complementary cases:

(1) $\angle(R_i) > -\pi$.
(2) $\angle(R_i) \leq -\pi$.

The following theorem characterizes the structure of the left region in the first case.

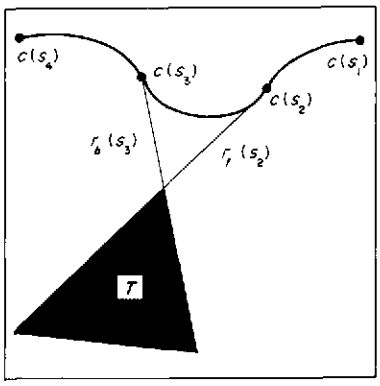


Fig. 6. The left region of a concave region.

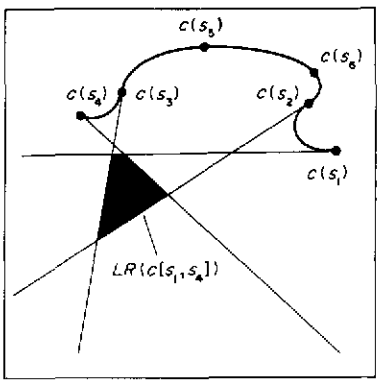


Fig. 7. The left region of a convex fragment is not involved in determining $K(S)$.

4.1.1. *Theorem.* Let $c[s_1, s_4]$ be a curve for which there exist s_2, s_3 such that:

$s_1 < s_2 \leq s_3 < s_4$;
$c[s_1, s_2]$ is a convex fragment;
$c[s_2, s_3]$ is a concave fragment, $\theta(s_3) - \theta(s_2) > -\pi$;
$c[s_3, s_4]$ is a convex fragment;
$\theta(s_1) \geq \theta(s_3)$;
$\theta(s_4) \leq \theta(s_2)$.

Then $LR(c[s_1, s_4])$ is the region T formed by the intersection of $r_f(s_2)$ and $r_b(s_3)$ (Fig. 6). The second case is now considered.

4.1.2. *Theorem.* Let $c[s_1, s_4]$ be the curve described in Theorem 4.1.1 except that $\theta(s_3) - \theta(s_2) \leq -\pi$. Then $LR(c[s_1, s_4]) = \varnothing$.

The last theorem leads to the following conclusion: if the turn angle of a concave fragment R_i satisfies $\angle(R_i) \leq -\pi$, then $K(S) = \varnothing$. For this reason we call a concave fragment R_i a forbidden concave fragment if $\angle(R_i) \leq -\pi$. Let us now turn to the investigation of convex fragments.

4.2. *The left region of a convex fragment*

As with the theorems concerning concave fragments, we need to regard the two neighbors of a convex fragment. However, the two complementary cases that appear here are not concerned with the convex fragment directly but, instead, with its two neighbors. The following theorems illustrate these cases.

4.2.1. *Theorem.* Let $c[s_1, s_4]$ be a curve for which exist s_2, s_3 such that:

$s_1 < s_2 \leq s_3 < s_4$;
$c[s_1, s_2]$ is a concave fragment, $\theta(s_2) - \theta(s_1) > -\pi$;
$c[s_2, s_3]$ is a convex fragment;
$c[s_3, s_4]$ is a concave fragment, $\theta(s_4) - \theta(s_3) > -\pi$;
$\theta(s_4) - \theta(s_1) \leq 0$.

Then $LR(c[s_1, s_4]) = LR(c[s_1, s_2]) \cap LT(c[s_3, s_4])$ (Fig. 7).

4.2.2. *Theorem.* Let $c[s_1, s_4]$ be the curve as was described in Theorem 4.2.1 except that $\theta(s_4) - \theta(s_1) > 0$. Then there exists two points, $c(s_5)$ and $c(s_6)$, satisfying

(1) $\theta(s_5^-) \leq \theta(s_1) \leq \theta(s_5^+), s_2 < s_5 \leq s_3$;
(2) $\theta(s_6^-) \leq \theta(s_4) \leq \theta(s_6^+), s_2 \leq s_6 < s_3$;
(3) $s_5 < s_6$;

and $LR(c[s_1, s_4]) = LR(c[s_1, s_2]) \cap LR(c[s_5, s_6]) \cap LR(c[s_3, s_4])$.

The first case states that the left region of such a convex fragment is not involved in determining $K(S)$. As for the second, here the situation is more complicated and several possibilities may happen. They are summarized in the next theorem.

4.2.3. *Theorem.* Let $c[s_1, s_4]$ be a curve as was defined in Theorem 4.2.2. Define

$$Q = LR(c[s_1, s_2] \cap LR(c[s_3, s_4].$$

Then one of following possibilities may occur:

(1) $Q = \varnothing$.
(2) $Q \cap LR(c[s_5, s_6]) = \varnothing$.
(3) $Q \cap LR(c[s_5, s_6]) = Q$.
(4) $Q \cap LR(c[s_5, s_6]) = T$, where there exist s_7 and s_8 for which:

$s_5 < s_7 < s_8 < s_6$,
$T = Q \cap LR(c[s_7, s_8])$,
$\theta(s_8) - \theta(s_7) < 2\pi$,
$c(s_7)$ is to the left of $g(s_8)$ and
$c(s_8)$ is to the left of $g(s_7)$.

The interesting case is certainly (4), as illustrated in Fig. 8. It will be useful to introduce an explicit definition of such a left region.

4.2.4. *Theorem.* Let $c[s_1, s_2]$ be a convex fragment such that $\angle(c[s_1, s_2]) < 2\pi$, $c(s_1)$ is to the left of $g(s_2)$ and $c(s_2)$ is to the left of $g(s_1)$. Then $LR(c[s_1, s_2])$ is the region T whose boundary is formed by $c[s_1, s_2]$

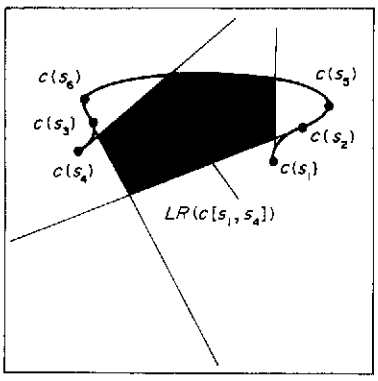
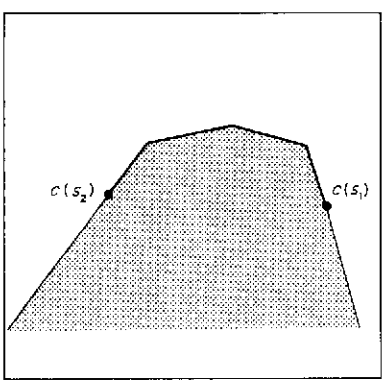
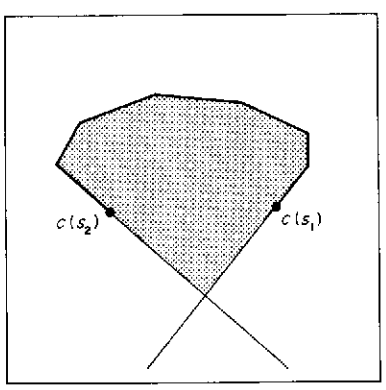


Fig. 8. The left region under the terms of Theorem 4.2.3, case (4). The left region of the compound fragment is given by $LR(c[s_1, s_2]) \cap LR(c[s_5, s_6]) \cap LR(c[s_3, s_4])$.



(a)



(b)

Fig. 9. The left region of a stand alone convex fragment $c[s_1, s_2]$ under the terms of Theorem 4.2.4: (a) $\angle(c[s_1, s_2]) \leq \pi$; (b) $\pi < \angle(c[s_1, s_2]) < 2\pi$.

and parts of $r_b(s_1)$ and $r_f(s_2)$ (Fig. 9). The above results lead to the following definition.

Let $c[s_1, s_4]$ be a curve such as was described in Theorem 4.2.2. Refer to points $c(s_1)$, $c(s_2)$, $c(s_3)$ and $c(s_4)$ as d_{i-1}, e_i, d_i and e_{i+1}, respectively. Then the points $c(s_5)$ and $c(s_6)$ are of E-type and D-type, and will be referred to as E_i and D_i, respectively. Under the terms of Theorem 4.2.1, s_5 and s_6 either do not exist or $s_5 \geq s_6$. In both cases, we will define $E_i = D_i$.

Let us now summarize the last two sections.

4.3. *The structure of* K(S) *boundary*

It is evident that dealing with either concave fragments or convex fragments, the left region of both types is a convex region. This, together with the fact that $K(S) \subset S$, leads to the immediate consequence that $K(S)$ is a convex shape. Hence, the boundary of $K(S)$, K, is a convex curve. The contributions of concave fragments and convex fragments to this curve are as follows:

Concave fragments: let $c[s_{d_i}, s_{e_{i+1}}]$ be a concave fragment. If this fragment is not forbidden, then K may include parts of $r_f(s_{d_i})$ and of $r_b(s_{e_{i+1}})$ only, i.e. $c[s_{d_i}, s_{e_{i+1}}]$ itself does not take part in K.

Convex fragments: let $c[s_{e_i}, s_{d_i}]$ be a concave fragment. If $\theta(s_{d_{i+1}}) - \theta(s_{e_{i-1}}) \leq 0$, then $c[s_{e_i}, s_{d_i}]$ certainly does not take part in K. We shall refer to such a fragment as a non-potential convex fragment. If, on the other hand, $\theta(s_{d_{i+1}}) - \theta(s_{e_{i-1}}) > 0$, then $c[s_{e_i}, s_{d_i}]$ is potential. E_i and D_i exist and part of $c[s_{E_i}, s_{D_i}]$ may determine K.

Due to these facts, the first assignment of a future kernel algorithm is clear: locate e-type and d-type points along ∂S.

5. FORBIDDEN FRAGMENT EXISTENCE TEST

We have seen that the existence of a forbidden concave fragment implies that $K(S) = \varnothing$. It is a simple matter then to check the list of concave fragments turn angle to see if such a fragment exists. Nevertheless, consider Fig. 10, where both concave fragments are allowed but the left region of the whole curve is empty. This leads to a natural extension of the notion of a concave fragment.

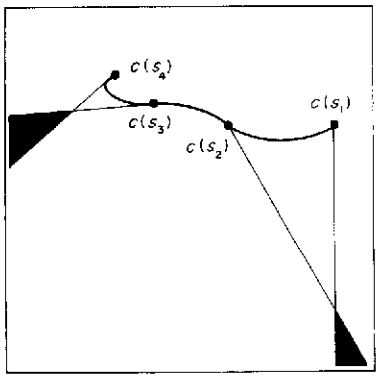


Fig. 10. The two concave regions are not forbidden but the left region of the entire curve is empty.

5.0.1. *Definition.* An alternating sequence of convex and concave consecutive fragments forms a generalized concave fragment if it starts and ends with concave fragments. Obviously, a single concave fragment is also a generalized one.

As in the case of concave fragments we can show that:

5.0.2. *Theorem.* If R is a generalized concave fragment and $\angle(R) \leq -\pi$, then $K(S) = \varnothing$ and hence it deserves to be called a forbidden generalized concave fragment.

The problem now is to devise a fast algorithm that answers the question: is there along ∂S a forbidden generalized concave fragment? It is claimed that the following algorithm does the job.

Suppose that there are M convex and M concave fragments along ∂S. Let us denote the concatenation of the mth to the nth consecutive fragments, $m \leq n$, as R_m^n. The input to the algorithm is the turn angles of $2M$ consecutive alternating concave and convex fragments forming ∂S:

$$\angle(R_0^0), \angle(R_1^1), \cdots \angle(R_{2M-1}^{2M-1}).$$

The output of this procedure is FAIL if a forbidden fragment exists and SUCCESS otherwise. The algorithm scans the list of turn angles twice in case we started in the middle of the only forbidden generalized concave fragment.*

Algorithm A1

Initial step:
  $b_{-1} = 0$.
General step:
  For $n = 0$ to $4M - 1$
  begin
    $a_n = b_{n-1} + \angle(R_n^n)$.
    if $a_n \geq 0$ then $b_n = 0$.
    else $b_n = a_n$.
    if $b_n \leq -\pi$, return FAIL.
  end
  return SUCCESS.

5.0.3. *Theorem.* Algorithm A1 ends with FAIL if and only if along ∂S there exists a forbidden generalized concave fragment.

Since the turn angle of ∂S is 2π, the existence of a forbidden generalized concave curve R_1 implies that the turn angle of the complementary curve $R_2 = \partial S / R_1$, satisfies: $\angle(R_2) \geq 3\pi$. This is why R_2 is called a forbidden generalized convex fragment and the existence of two such complementary forbidden fragments is mutually dependent.

The latter fact yields a similar algorithm to A1 which scans the list of turn angles only once:

Algorithm A1a

Initial step:
  $b_{-1} = 0$.
  $B_{-1} = 0$.
General step:
  For $n = 0$ to $2M - 1$
  begin
    $a_n = b_{n-1} + \angle(R_n^n)$.
    if $a_n \geq 0$ then $b_n = 0$.
    else $b_n = a_n$.
    if $b_n \leq -\pi$, return FAIL.
    $A_n = B_{n-1} + \angle(R_n^n)$.
    if $A_n \leq 0$ then $B_n = 0$.
    else $B_n = A_n$.
    if $B_n \geq 3\pi$, return FAIL.
  end
  return SUCCESS.

Assuming that this stage has terminated successfully, we are ready to extract K. First, for each non-potential convex fragment, set $E_i = D_i$. For each potential convex fragment, locate E_i and D_i if this is convenient or set $E_i = e_i$ and/or $D_i = d_i$ otherwise. Then start the kernel extraction algorithm.

6. THE KERNEL ALGORITHM

The main kernel algorithm includes all steps in order to determine the kernel of an X-type shape. It will be introduced in the last section of this chapter. Firstly, let us formulate the kernel extraction algorithm which is the most complicated step in the process.

6.1. *The kernel extraction algorithm*

Those familiar with Lee and Preparata's algorithm,[1] which will be referred to as the original algorithm, will recognize that the proposed algorithm follows its main lines. However, three major differences deserve consideration:

(1) The original algorithm deals only with straight lines, rays and segments while in the new algorithm, X-type boundary fragments may appear, a fact which demands a special treatment.

(2) The input to the original algorithm consists of all edges while in the new one only the necessary parts are introduced. Thus, while the original algorithm maintains the subkernel boundary as a continuous curve, the new procedure may introduce, temporarily, subkernel boundary gaps. These gaps are bridged up in subsequent steps so their existence is eventually unnoticeable.

(3) In the original algorithm there is an efficiency test that ensures that in some cases, the time complexity of the algorithm will not reach $O(N^2)$. This situation may happen when the boundary wraps around the current subkernel in an angle greater than 3π. Here this check was already done by algorithm A1 and we will assume that A1 was executed prior to the application of the kernel extraction algorithm.

The input to the algorithm is M triplets, each triplet consisting of two rays and one boundary fragment:

* All indices are modulo $2M$.

Consider the convex fragment $R_i = c[s_{e_i}, s_{d_i}]$. The two rays are $r_b(s_{e_i})$ and $r_f(s_{d_i})$. The former ray will be referred to as $\Delta\theta_{e_i}e_i$ meaning a ray which comes from infinity in the direction of $\theta(s_{e_i})$ and that ends at e_i while the latter ray will be $d_i\theta_{d_i}\Lambda$, meaning a ray which emanates from d_i in the direction of $\theta(s_{d_i})$ and goes to infinity. The fragment is $c[s_{E_i}, s_{D_i}]$ which will be referred to as $R_i[E_i, D_i]$.

The output from the algorithm is either a FAIL answer in case $K(S) = \varnothing$ or a circular list of vertices, θ values and boundary fragments which represents K.

During the activation of the algorithm, the list represents the boundary of the current subkernel $K_n(S)$. The vertices are denoted $w_0, w_1, \ldots$, where w_j is the successive vertex of w_{j-1} in the positive direction scan of K_n, and the entity between each $w_{j-1} - w_j$ pair, (w_{j-1}, w_j), may be either a θ value or a boundary fragment representation. In case (w_{j-1}, w_j) is a θ value, it will be referred to as θ_j.

The algorithm deals separately with the intersections of the current subkernel boundary, K_n, with $d_i\theta_{d_i}\Lambda$, $R_i[E_i, D_i]$ and $\Lambda\theta_{e_i}e_i$, $0 \le i \le M-1$. If an intersection occurs, we update K_n to yield K_{n+1}. As we find the entrance intersection point with the convex shape $K_n(S)$, we immediately search for the exit point. Both points together with the curve between them determine K_{n+1}. In cases where no entrance point was found, either $K(S) = \varnothing$ or $K_{n+1} = K_n$.

K_n is maintained as a doubly linked list. In the initial step, $K_0(S)$ is an unbounded convex region and the list has a list head and list tail. After several steps $K_n(S)$ turns to be bounded and the list becomes circular. w_{head} and w_{tail} are points at infinity of the unclosed K_n. Correspondingly, θ_{head} and θ_{tail} are the directions of the first and last rays. As a guiding tool, we use $LINE(w_{j-1}, w_j)$. The intersection of the current item (either a ray or a fragment) with this line navigates the next step of the algorithm. Two vertices, w_F and w_L, have a special meaning. Briefly we can say that they hold a similar meaning to F and L in the original algorithm. Moving in the positive direction means also that along a convex boundary, θ is a nondecreasing function. Since $K_n(S)$ is convex by the intersections of convex regions, in case $\theta_j < \theta_{j-1}$ we must make a correction by adding 2π to θ_j. This will ensure that θ is consistent along the boundary of K_n. The procedure forward(j) handles this situation, i.e. increments j and adjusts θ_j if necessary. Backward(j) does the opposite by decrementing j and subtracting 2π from θ_j when necessary.

A pseudo-code listing of the complete algorithm, which will be called Algorithm A2, is given in the appendix. Algorithm A2 constructively intersects the necessary rays and fragments as were defined in former sections. Therefore, if not terminated with a FAIL answer, its output will be the boundary of the kernel. The properties of Algorithm A2 are summarized in the following theorems.

6.1.1. *Theorem.* In stage (1.1) of the algorithm, if $(w_{j-1}, w_j) \subset \partial S$, then $K(S) = \varnothing$.

6.1.2. *Theorem.* In stage (2.1) of the algorithm, (w_{j-1}, w_j) cannot be part of ∂S.

6.1.3. *Theorem.* In stage (3.1) of the algorithm while scanning in the negative direction, if $(w_{j-1}, w_j) \subset \partial S$, then $K(S) = \varnothing$.

6.1.4. *Theorem.* In stage (3.1) of the algorithm, while scanning in the positive direction, if $(w_{j-1}, w_j) \subset \partial S$, then no further consideration of $d_i\theta_{d_i}\Lambda$ is necessary.

Algorithm A2 describes the process schematically. There are cases where additional information concerning the shape boundary may be exploited. This may yield a more efficient implementation. For instance, if R_i is a piecewise regular curve:

$$R_i[E_i, D_i] = \bigcup_{j=0}^{L_i-1} c[s_j, s_{j+1}], \quad s_0 = E_i, s_{L_i} = D_i$$

and the rays $r_f(s_j)$ at the corner points are known, we may regard each boundary point as a d-type point and execute stages (2) and (3) for each $c[s_j, s_{j+1}]$ fragment repeatedly. In particular, for polygons, exploiting this information leads, in fact, to Lee and Preparata's algorithm. We shall assume that this is the case when we deal with polygons.

We are now ready to perform the sequence of steps in order to find the kernel of an X-type shape.

6.2. *The main kernel algorithm*

Algorithm A3

(1) Locate all e-type and d-type points along ∂S. If there are no such points, stop: $K(S) = S$.

(2) Activate algorithm A1. If there is any forbidden generalized concave fragment along ∂S, stop: $K(S) = \varnothing$.

(3) For all convex fragments R_i: if R_i is non-potential, set $E_i = D_i$; if R_i is potential, locate E_i and D_i if this is convenient. Otherwise set $E_i = e_i$ and/or $D_i = d_i$.

(4) Activate algorithm A2 in order to extract K.

Since polygons are a subclass of the X-type shapes, it is not unreasonable to expect that if the kernel of a polygon is to be determined, the performance of the proposed kernel algorithm will be similar if not better than the performance of the original one.

A comparison, from both theoretical and practical points of view, is made in the following two chapters.

7. PERFORMANCE ANALYSIS

Since in polygons each edge may intersect K_n only once, contrary to an arbitrary X-type shape boundary fragment, we shall make the analysis on polygons.

Let us consider a polygon with N boundary edges and M concave boundary fragments, $M > 0$. The polygon is represented by its vertical and, in addition, the exterior angle at each vertex is known.

Points of e- and d-type are always convex corner points. Scanning the boundary in the positive direction, an e-type point is the first convex corner point after a concave corner point while a d-type point is the last convex corner point prior to a concave corner point. Hence, determination of the $2M$ e-type and d-type points along the boundary is carried out in a fast $O(N)$ sequence. The turn angle of a concave fragment is the sum of all its (concave) exterior angles. Similarly, the turn angle of a convex fragment is the sum of all its (convex) exterior angles. A1 deals with $4M$ angle values sequentially and hence it is of $O(M)$ time complexity. Again, this part is fast since it involves only a few elementary operations. Setting the values of E_i and D_i, $0 \le i \le M - 1$, is again done in a fast $O(N)$ procedure.

The set of input elements to algorithm A2 consists of rays and polygon edges that belong to R_i fragments. Let this set contain J elements. As was stated before, we exploit the forward rays at each corner point along an R_i fragment and thus we actually get the original algorithm with minor modifications. We will show that algorithm A2 runs in $O(J)$ time. Since in polygons $2M \le J \le N$, A2 may run on $O(N)$ in the worst case, while in other cases the order is lower.

7.1. *A2 Performance analysis*

The analysis of A2 is similar in spirit to the original algorithm method of analysis. The inspection of various A2 stages will be divided into tasks concerning K_n updating and w_F and w_L updating.

In stage (1), w' is searched for by scanning from w_F in the positive direction while w'' is searched for by scanning from w_F in the negative direction. Let the total number of K_n edges scanned be v_n. In stages (2) and (3) we may reach w' either by a positive direction scan or by a negative direction scan. In the first case, the scan process until we reach w' will be considered as w_L updating. Thereafter the scan resumes in the same direction. In the latter case, no update of w_L takes place. Finding w' is by a negative direction scan and finding w'' is by a positive direction scan, both starting from w_L.

As in stage (1), denote the total number of K_n edges scanned between w' and w'' by v_n. The number of edges of K_n that were removed in each of the three stages is $v_n - 2$. Since each edge is a part of either a ray or boundary edge, then $\sum_n (v_n - 2) \le J$. Thus, the total number of updating of K_n is carried out in $O(J)$ time.

Consider now updating w_F and w_L. Note that these vertices advance always in the positive direction. We will show that the number of times each vertex is visited is at most two. Suppose conversely that a vertex is visited for the third time. In this case ∂S wraps around K_n at least twice, i.e. along ∂S there is a fragment R_1 with a turn angle $\angle(R_1) \ge 3\pi$, in contradiction to the fact that A1 was ended successfully. Thus, updating w_F and w_L is done in $O(J)$ time.

An overall calculation thus shows that A2 runs in $O(J)$ time.

8. COMPARING THE ORIGINAL AND THE NEW ALGORITHMS

In order to demonstrate the advantages arising using the results of this work, a comparison between Lee and Preparata's algorithm and the proposed implementation was done. For this purpose, a family of polygons was created. The main feature of this family is that the convex fragments along each polygon boundary deteriorate to single points. Under such a term, the original algorithm will deal with all of the polygon edges while the new version will scan only the necessary rays. M, the number of concave fragments, and L, the number of polygon edges along each concave fragment, are two independent

Table 1. Runtime of several stages as a function of M and L. The overall runtime of both algorithms appears in the two right columns

M	L	e-type and d-type point location	Forbidden fragment test	Lee and Preparata's algorithm	Kernel extraction algorithm	Original algorithm runtime	New algorithm runtime
200	2	0.33	1.04	34.56	34.56	35.60	35.93
100	4	0.33	1.04	25.93	17.75	26.97	19.12
50	8	0.33	1.04	20.05	6.98	21.09	8.35
25	16	0.33	1.04	20.27	4.45	21.31	5.82
10	40	0.33	1.04	19.84	1.65	20.88	3.02
8	50	0.33	1.04	20.00	1.32	21.04	2.69
5	80	0.33	1.04	20.88	0.82	21.92	2.19
4	100	0.33	1.04	21.37	0.49	22.41	1.86
3	133	0.33	1.04	21.59	0.38	22.63	1.75

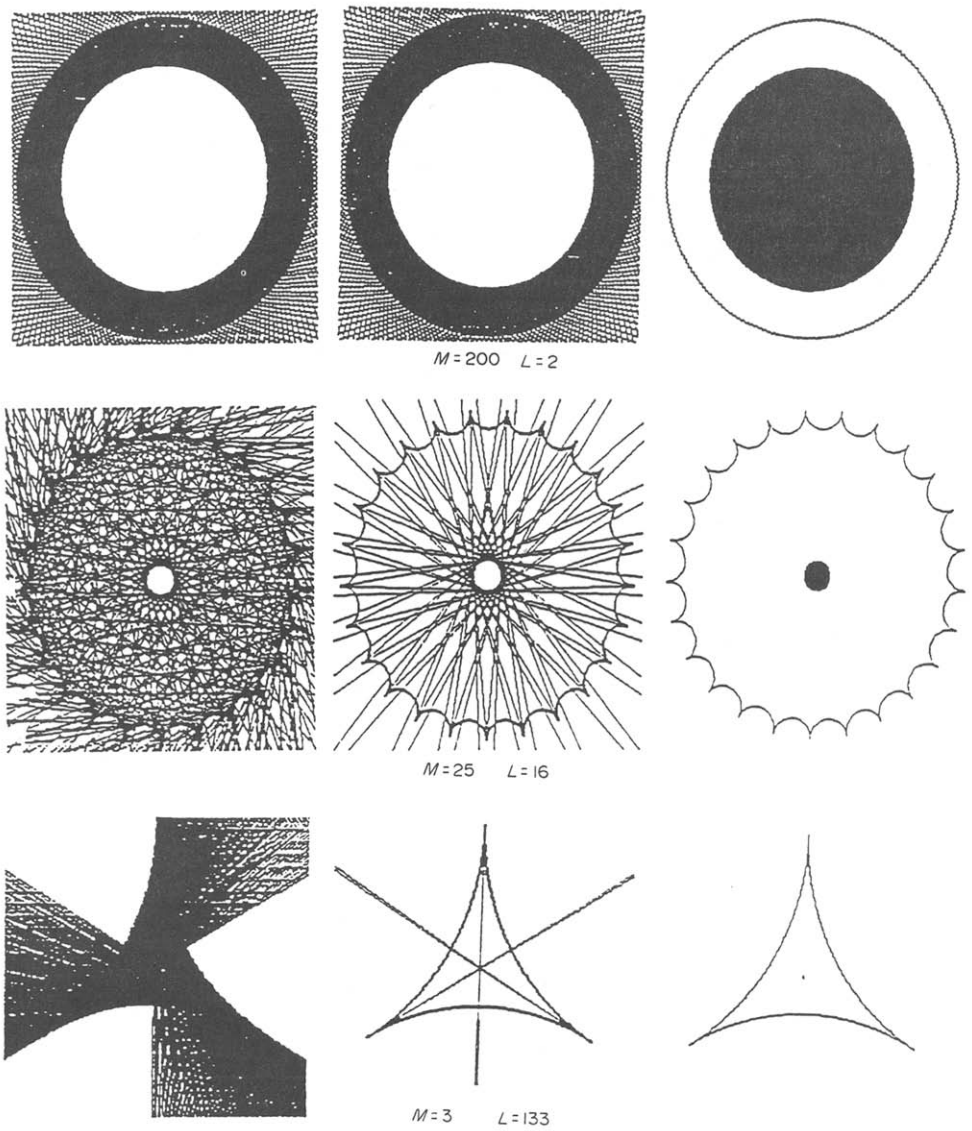


Fig. 11. The shapes, rays concerned and kernel for various M and L values. The left column of the figure shows the rays considered by the original algorithm, the middle column shows the rays considered by the kernel extraction algorithm and the right column illustrates the kernel of the shapes.

parameters. We have set

$$M \cdot L = N = 400,$$

in order to work on a fixed number of polygon edges.

The simulation was done on an IBM-AT computer and the algorithms were written in *C*. The efficiency test done by the original algorithm was replaced by a prior application of A1. The kernel extraction algorithm in this case was, in fact, the original algorithm, with minor modifications. Only *e*-type and *d*-type points along the boundary were located. The runtime of each stage was recorded automatically. These values for several M and L values are presented in Table 1.

The rays each algorithm has scanned and the final kernel that was found are presented in Fig. 11. The overall runtime of each kernel algorithm is graphically displayed in Fig. 12. Obviously, as M decreases, the new algorithm runs much faster.

9. CONCLUDING REMARKS

Star-shapedness is an important feature of planar objects. Determining whether a planar object is star-shaped or not, and if star-shaped determining the kernel, are important tasks in shape analysis and pattern recognition. The complexity of determining shape related features becomes an important factor in the design of pattern recognition and analysis tasks.[5] Therefore much attention was paid to the complexity of finding the kernel of planar polygonal

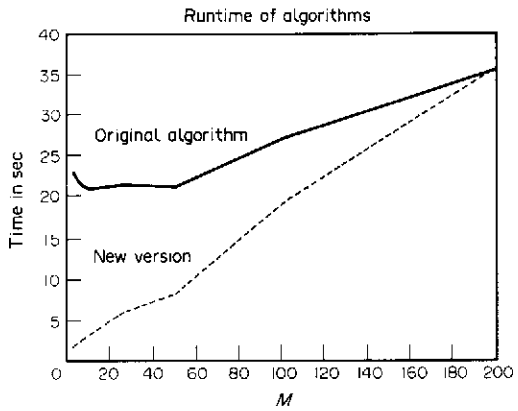


Fig. 12. Runtime of the two algorithms. Starting from a similar point for the worst case ($2M = N$), the performance of the new version improves as M decreases.

objects. The algorithm of Lee and Preparata,[1] having a complexity of $O(N)$, where N is the number of edges of the polygonal shape, is worst case optimal, since clearly we need to input the information to the algorithm, and this already takes N steps. The computational geometry community thus regarded this problem as completely settled and did not pay any further attention to it. However, the algorithm of Lee and Preparata involves performing quite a complicated procedure, for each side of the polygon under consideration, and their algorithm may become impractical when N is large, as happens when we do polygonal approximation of shapes with smooth boundaries.

We have presented an algorithm for determining the kernel of planar shapes with smooth, non-degenerate boundaries. The algorithm is based on the observation that the inflection points provide a natural segmentation of the object boundary that helps to determine the kernel as the intersection of the regions within the shape from which each concave portion of the boundary is seen. From the development of our algorithm it is clear that after the boundary information is read, the complexity of finding the kernel is proportional to the number of inflection points of the boundary, the constant of proportionality being similar to the one involved in the algorithm of Lee and Preparata. Reading the boundary information for a polygonal object is proportional to the number of polygon edges; however, the constant of proportionality is very small. When the boundary is described by a spline function or any other method for describing or approximating curved boundaries, the complexity of reading the boundary data is given by the number of control points, or knots. This number will be considerably smaller than the number of edges of a reasonable polygonal approximation of the given object. Hence for objects with smooth boundaries described by methods that are frequently used in graphics and CAD tools, our kernel finding algorithm leads to significant savings in execution time. Furthermore, even in the case of polygonal shapes, our algorithm yields savings in execution time, that result from its dependence on an inner complexity measure of the shape under consideration, namely, the number of inflection edges of its boundary. Only in some worst case examples will our algorithm run in the same time as the Lee and Preparata method. It is somewhat surprising that in the long time that has elapsed since the Lee and Preparata algorithm was put forward, no one has made the observations that led to the algorithm described above. Our motivation for analysing this issue in fact came from a failure of the classical algorithm to perform efficiently for polygonal approximations of planar shapes with smooth boundaries, described via splines.

We did not provide proofs for the theorems stated in this paper, due to lack of space. These are found in the Technical Report,[6] available from the authors.

REFERENCES

1. D. T. Lee and F. P. Preparata, An optimal algorithm for finding the kernel of a polygon, *J. ACM* **26**, 415–421 (1979).
2. I. M. Yaglom and V. G. Boltianskii, *Convex Figures*. Holt, Rinehart and Winston, New York (1961).
3. M. I. Shamos and D. Hoey, Geometric intersection problems, *17th Annual IEEE Symp. Foundations Comput. Sci.*, pp. 208–215 (October 1976).
4. C. C. Hsiung, *A First Course in Differential Geometry*. Wiley-Interscience, New York (1981).
5. G. T. Toussaint, Pattern recognition and geometrical complexity, *Proc. Int. jt Conf. Pattern Recognition* (December 1980).
6. R. Bornstein and A. M. Bruckstein, *Finding the Kernel of Planar Shapes*, EE Publication No. 694, Technion, Israel Institute of Technology, Haifa (December 1988).

APPENDIX

Algorithm A2

Initial step:

$n = 0$.

K_0 is taken to be $\Lambda\theta_{e_0} w_0 \theta_{d_0} \Lambda$, where w_0 is the intersection point between $\Lambda\theta_{e_0} e_0$ and $d_{M-1}\theta_{d_{M-1}}\Lambda$ (Fig. A1).

$w_{F-1} = w_0$

$w_{L-1} = w_0$

General step:

For $i = 0$ to $M - 1$ do

begin

(1) /*$\Lambda\theta_{e_i} e_i$ is considered */

If $i = 0$ goto (2).

(1.1) /* looking for w' */

$a = e_i$

if $\theta_{e_i} \geq \theta_F$ or $\Lambda\theta_{e_i} a$ intersects $RAY[w_F, w_{F-1})$/$SEG[w_F, w_{F-1})$

goto (2) /*$K_{n+1} = K_n$ */.

$w_j = w_F$

while $(w_{j-1}, w_j) \not\subset \partial S$ and $\Lambda\theta_{e_i} a$ intersects $LINE(w_{j-1}, w_j)$ at p do begin

```
      if p ∈ RAY[w_{j-1}, w_j)/SEG[w_{j-1}, w_j) (Fig. A2(a))
      begin
        a = p.
        forward(j).
      end.
      else /*p ∈ SEG(w_{j-1}, w_j)*/
begin
        w' = p.
        a = p.
goto (1.2).
      end
    end
    return FAIL /* either the condition in Theorem 6.1.1
    is met or no intersection with K_n was found */.
(1.2)
    /* looking for w" */
    If K_n is unbounded
    begin
      If θ_{e_i} > θ_head (Fig. A2(b)).
      begin
        w_r = w_{F-1}.
        while Λθ_{e_i}a does not intersect (w_{r-1}, w_r) do
          backward(r).
        Let the intersection point be w".
        Letting K_n = α(w_{j-1}, w_j)β(w_{j-1}, w_j)γ,
        we set K_{n+1} = α(w_{r-1}, w")w"θ_{e_i}w'(w', w_j)γ/* is
        also unbounded */.
      end
      else
      If θ_{e_i} < θ_tail − π (Fig. A2(c)).
      begin
        w_r = w_tail.
        while Λθ_{e_i}a does not intersect (w_{r-1}, w_r) do
          backward(r).
        Let the intersection point be w".
        Letting K_n = α(w_{j-1}, w_j)β(w_{r-1}, w_r)γ,
        we set K_{n+1} = (w', w_j)β(w_{r-1}, w")w"θ_{e_i}w' /* and
        it becomes bounded */.
      end
      else /* θ_head ≥ θ_{e_i} ≥ θ_tail − π*/ (Fig. A2(d)).
      begin
        Letting K_n = α(w_{j-1}, w_j)β,
        we set K_{n+1} = Λθ_{e_i}w' (w', w_j)β/* is also
        unbounded */.
      end
    end
    else /*K_n is bounded */ (Fig. A2(e)).
    begin
      w_r = w_{F-1}.
      while Λθ_{e_i}a does not intersect (w_{r-1}, w_r) do
        backward(r).
```

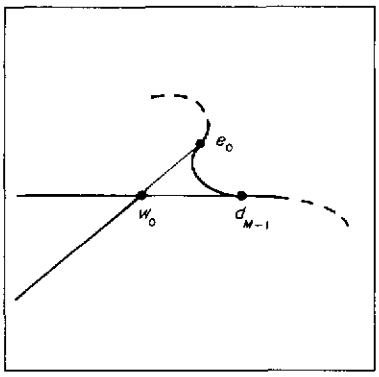


Fig. A1. The initial subkernel boundary, K_0, is formed by the intersection of $\Lambda\theta_{e_0}e_0$ with $d_{M-1}\theta_{d_{M-1}}\Delta$.

```
      Let the intersection point be w".
      Letting K_n = α(w_{j-1}, w_j)β(w_{r-1}, w_r),
      we set K_{n+1} = (w', w_j)β(w_{r-1}, w")w"θ_{e_i}w'.
    end
    w_{F-1} = w'
(2)
    /*R_i is considered */
    w_j = w_L
    if E_i = D_i goto (3). /*R_i does not participate in the
    boundary of K(S) */
    a = E_i.
(2.1)
    /* looking for w' */
    while R_i[a, D_i] intersects LINE(w_{j-1}, w_j) at p do
    begin
      if p ∈ RAY[w_{j-1}, w_j)/SEG[w_{j-1}, w_j) (Fig. A3(a))
      begin
        a = p.
        forward(j).
        w_L = w_j
        w_F = w_j
      end.
      else
      if p ∈ RAY[w_j, w_{j-1})/SEG[w_j, w_{j-1}) (Fig. A3(b)).
      begin
        a = p.
        backward(j)
      end.
      else /*p ∈ SEG(w_{j-1}, w_j)*/
      begin
        w_r = w_L
        w' = p.
        a = p.
        goto (2.2).
      end
    end
    If θ_{d_i} ≥ θ_i + π return FAIL. /*K(S) = ∅*/
    else goto (3) /*K_{n+1} = K_n */
(2.2)
    /* looking for w" */
    If w_r is a point at infinity /* only when K_n is unbounded
    */
    begin
      if θ_{d_i} ≤ θ_head + π (Fig. A3(c)).
      begin
        Letting K_n = α(w_{j-1}, w_j)β
        we set K_{n+1} = α(w_{j-1}, w')w'R_i[w', D_i]d_iθ_{d_i}Λ/* is
        also unbounded */.
        w_{F-1} = d_i.
        w_{L-1} = d_i.
        goto general step.
      end
      else /* θ_{d_i} > θ_head + π*/
      begin
        Let w_{r-1} = w_head.
        goto (2.3).
      end
    end
    else /* w_r is not a point at infinity */
    begin
      forward(r).
      goto (2.3).
    end
(2.3)
    /* Looking for a second intersection of R_i with K_n */
    If (w_{r-1}, w_r) ⊂ ∂S, goto (2.4).
    If R_i[a, D_i] does not intersect (w_{r-1}, w_r) goto (2.4).
    If r = j, return FAIL. /* K(S) = ∅*/
    else (Fig. A3(d)).
    begin
      let w" be the first intersection point.
      Letting K_n = α(w_{j-1}, w_j)β(w_{r-1}, w_r)γ,
```

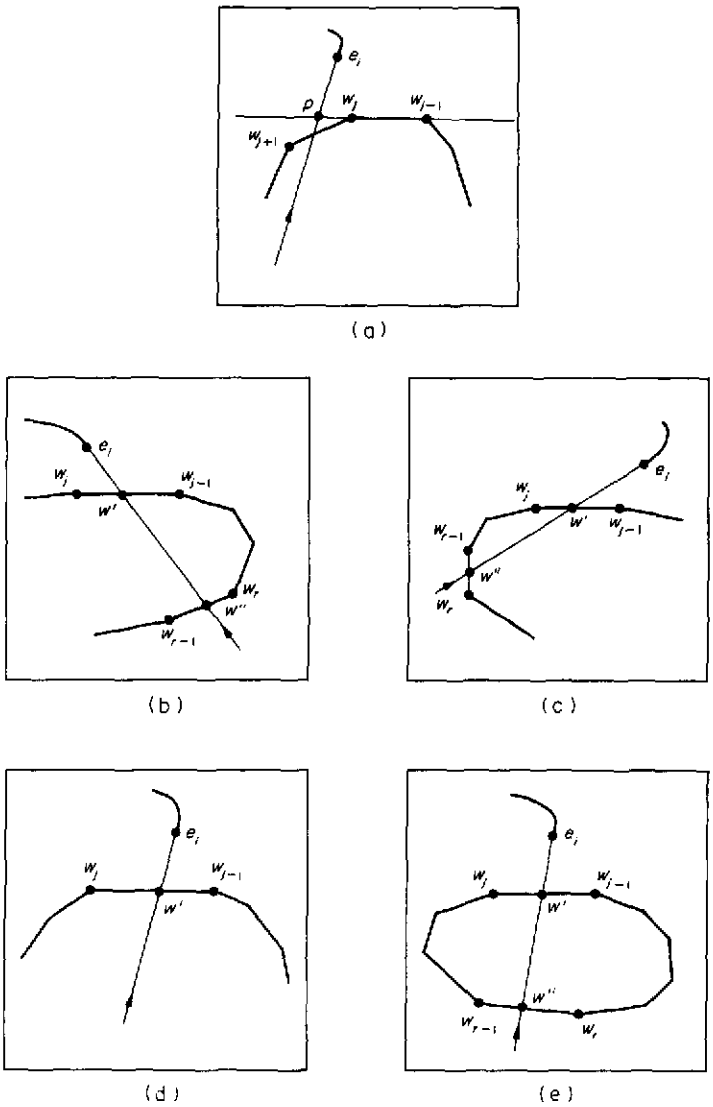


Fig. A2. Several possible cases in the evaluation of $\Lambda\theta_{e_i}e_i$. Refer to stage (1) of algorithm A2 for explicit definitions.

we set $K_{n+1} = \alpha(w_{j-1}, w')w'R_i[w', w'']w''(w'', w_r)\gamma$.
$w_{F-1} = w''$.
$w_{L-1} = w''$.
goto (2.1).
end

(2.4)
/* Looking for an intersection of $d_i\theta_{d_i}\Lambda$ with K_n */
If $d_i\theta_{d_i}\Lambda$ does not intersect (w_{r-1}, w_r)
begin
forward(r).
goto (2.2).
end
else (Fig. A3(e)).
begin
Let w'' be the intersection point.
Letting $K_n = \alpha(w_{j-1}, w_j)\beta(w_{r-1}, w_r)\gamma$,
we set $K_{n+1} = \alpha(w_{j-1}, w')w'R_i[w', D_i]$
$d_i\theta_{d_i}w''(w'', w_r)\gamma$.
$w_{F-1} = d_i$.
$w_{L-1} = d_i$.
goto general step.
end

(3)
/* $d_i\theta_{d_i}\Lambda$ is considered */
If $i = M - 1$ return K_n/* The algorithm is finished. K_n is the boundary of $K(S)$ */

(3.1)
/* looking for w' */
$a = d_i$.
while $(w_{j-1}, w_j) \subset \partial S$ and $a\theta_{d_i}\Lambda$ intersects $LINE(w_{j-1}, w_j)$ at p do
begin
if $p \in RAY[w_{j-1}, w_j)/SEG[w_{j-1}, w_j)$ (Fig. A4(a)).
begin
$a = p$.
dir = positive.
$w_L = w_j$.
forward(j).
end.

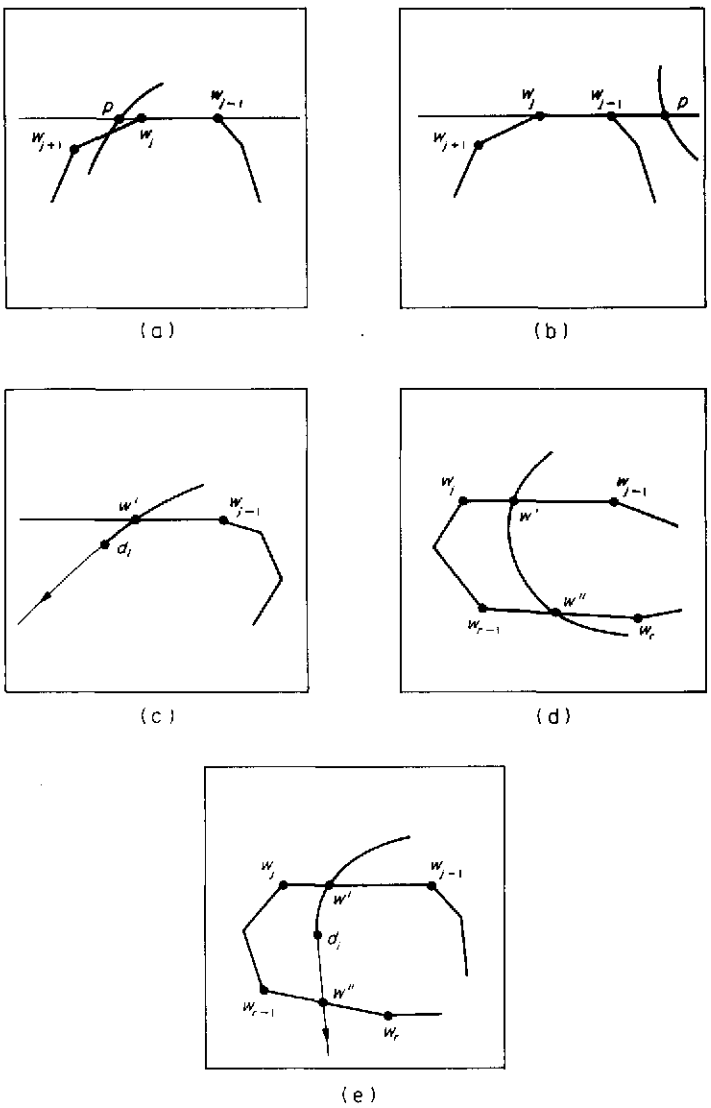


Fig. A3. Several possible cases in the evaluation of R_i. Refer to stage (2) of algorithm A2 for explicit definitions.

```
    else
    if p ∈ RAY[w_j, w_{j-1})/SEG(w_{j-1}, w_j] (Fig. A4(b)).
    begin
      a = p.
      dir = negative.
      backward(j).
    end.
    else /* p ∈ SEG(w_{j-1}, w_j) */
    begin
      w' = p.
      a = p.
      goto (3.2).
    end
  end
  if (w_{j-1}, w_j) ⊂ ∂S
  begin
    if dir = negative return FAIL /* refer to Theorem
    6.1.3 */.
    else goto (3.3) /* refer to Theorem 6.1.4 */.
  end
```

```
  if θ_{d_i} > θ_j + π, return FAIL. /* K(S) = ∅*/
  else goto (3.3).

(3.2)
  /* looking for w'' */
  If K_n is unbounded
  begin
    If θ_{d_i} < θ_tail (Fig. A4(c))
    begin
      w_{r-1} = w_L.
      while aθ_{d_i}Λ does not intersect (w_{r-1}, w_r) do
        forward(r).
      Let the intersection point be w''.
      Letting K_n = α(w_{j-1}, w_j)β(w_{r-1}, w_r)γ,
      we set K_{n+1} = α(w_{j-1}, w')w'θ_{d_i}w''(w'', w_r)γ/* is
      also unbounded */
    end
    else
    If θ_{d_i} > θ_head + π (Fig. A4(d))
```

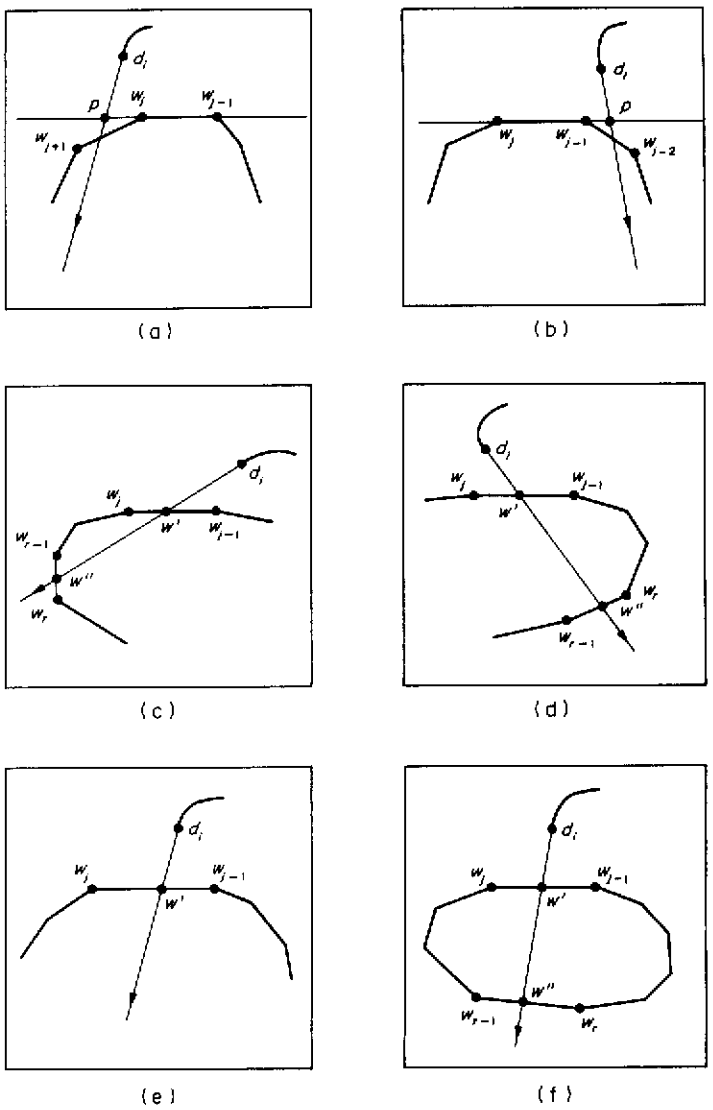


Fig. A4. Several possible cases in the evaluation of $d_i\theta_{d_i}\Lambda$. Refer to stage (3) of algorithm A2 for explicit definitions.

```
    begin
      w_{r-1} = w_head.
      while aθ_{d_i}Λ does not intersect (w_{r-1}, w_r) do
        forward(r).
      Let the intersection point be w''.
      Letting K_n = α(w_{r-1}, w_r)β(w_{j-1}, w_j)γ,
      we set K_{n-1} = (w'', w_r)β(w_{j-1}, w')w'θ_{d_i}w''.  /*
      and it becomes bounded */
    end
    else /* θ_tail ≤ θ_{d_i} ≤ θ_head + π */ (Fig. A4(e))
    begin
      Letting K_n = α(w_{j-1}, w_j)β,
      we set K_{n-1} = α(w_{j-1}, w')w'θ_{d_i}Λ./*  is  also
      unbounded */
    end
  end
  else /* K_n is bounded */ (Fig. A4(f))
  begin
    w_{r-1} = w_j.
    while aθ_{d_i}Λ does not intersect (w_{r-1}, w_r) do
      forward(r).
    Let the intersection point be w''.
    Letting K_n = α(w_{j-1}, w_j)β(w_{r-1}, w_r),
    we set K_{n+1} = α(w_{j-1}, w')w'θ_{d_i}w''(w'', w_r).
  end
  w_{L-1} = w'

(3.3)
  /* updating w_F */
  while e_{i+1} is on or to the left of LINE(w_{F-1}, w_F) do
    forward(F).
end
```

Algorithm A2 constructively intersects the necessary rays and fragments as were defined in former sections. Thus, if not terminated with a FAIL answer, the output is the boundary of the kernel.

About the Author—ALFRED M. BRUCKSTEIN was born in Sighet, Transylvania, Romania, on 24 January 1954. He received the B.Sc. and M.Sc. degrees in Electrical Engineering, from the Technion, Israel Institute of Technology, in 1977 and 1980, respectively, and his Ph.D. degree in Electrical Engineering from Stanford University, Stanford, California, in 1984. From October 1984 until June 1989 he was with the Faculty of Electrical Engineering at the Technion, Haifa. Presently he is with the Faculty of Computer Science there. His research interests are in computer vision, image processing, estimation theory, signal processing, algorithmic aspects of inverse scattering, point processes and mathematical models in neurophysiology. Prof. Bruckstein is a member of SIAM, MAA and AMS.

About the Author—RAANAN BORNSTEIN received his M.Sc. degree from the Electrical Engineering Department of the Technion, in December 1988. Since then he has been working in the Silicon Valley in California.