ALGORITHMS FOR PLANAR GEOMETRIC MODELS*

Chanderjit Bajaj and Myung-Soo Kim

Computer Sciences Department
Purdue University
CSD-TR-755
March 1988

Abstract

We consider planar geometric models given by an explicit boundary of $O(n)$ algebraic curve segments of maximum degree d. We present an $O(n \cdot d^{O(1)}$ time algorithm to compute its convex hull and an $O((n \log \log n + K) \cdot d^{O(1)}$ time algorithms to compute various decompositions of an object, where K is the characteristic number of this object. Both operations, besides being solutions to interesting computational geometry problems, prove useful in motion planning with planar geometric models.

* Research supported in part by NSF grant MIP-85-21356, ARO contract DAAG29-85-C0018 under Cornell MSI and a David Ross Fellowship.

Proc. of 15th ICALP, Tampere, Finland, LICS Springer Verlag, to appear 1988

Algorithms for Planar Geometric Models*

Chanderjit Bajaj and Myung-Soo Kim

Department of Computer Science
Purdue University
West Lafayette, IN 47907.

Abstract

We consider planar geometric models given by an explicit boundary of $O(n)$ algebraic curve segments of maximum degree d. We present an $O(n \cdot d^{O(1)})$ time algorithm to compute its convex hull and an $O((n \log\log n + K) \cdot d^{O(1)})$ time algorithms to compute various decompositions of an object, where K is the characteristic number of this object. Both operations, besides being solutions to interesting computational geometry problems, prove useful in motion planning with planar geometric models.

1 Introduction

Geometric modeling is the use of a collection of techniques to describe the shape and structure of physical objects. Designed with sufficient generality it can also be used in motion planning, [6]. Such a system finds immediate application in engineering design and verification, automatic control machining and prototyping, [15]. Here we consider applications of planar geometric models given by an explicit boundary of $O(n)$ algebraic curve segments of maximum degree d. Our goal is to build a modeling system for these curved planar objects, which allows the testing of various motion planning strategies amongst fixed geometric obstacles. A powerful technique therein is the use of configuration space (*C-Space*) mappings which reduces the planning of collision free paths for complex objects, to path planning for a point amongst grown *C-Space* obstacles. To achieve this goal, we earlier presented algebraic methods for generating the *C-Space* obstacles, via the convolution computation of two arbitrary geometric models with algebraic curve boundaries, see [7]. Now, we consider a few practical and more robust alternatives of computing these *C-Space* obstacles. For gross motion planning, one may compute the *C-Space* obstacles for the convex hull of the original planar models. Alternatively, one may first decompose the given models into certain primitive pieces and then apply *C-space* obstacle generation on decomposed pieces of object and obstacle pairs. The convolution of primitive models is free from the complicated singularities that beset the general non-convex case.

The algorithms presented in this paper are for (1) computing the convex hull of a general (non-convex) planar geometric model and (2) computing the decomposition of a similar (non-convex) geometric model in terms of the unions and differences of convex geometric models and alternatively as a union of certain primitive geometric models. Further, we also show how this decomposition can

*Research supported in part by NSF grant MIP-85-21356, ARO contract DAAG29-85-C0018 under Cornell MSI and a David Ross Fellowship.

be refined to provide an *inner* polygon (resp. an *outer* polygon) which is a simple polygon totally contained in (resp. totally containing) the geometric model. Both these algorithms, coupled with the convolution techniques of [6,7] and the path planning strategies of [24] then offer a mechanism of planning collision free paths for planar geometric models, see [8]. Additionally, convex hull and decomposition computation are fundamental geometric operations and the algorithms we present here for planar geometric models extend previously known computational geometry methods for planar polygons, [20].

1.1 Convex Hull

Several linear–time algorithms for computing the convex hull of simple planar polygons are known, [11,14,17,19]. These algorithms achieve the more efficient $O(n)$ bound whereas the $\Omega(n \log n)$ lower bound applies to general problem of computing the convex hull of n points in the plane, see [20]. Schäffer and Van Wyk [22] extended these results to a linear–time algorithm for curved objects bounded by piecewise–smooth Jordan curves. Even though this algorithm gives an $O(n \cdot d^{O(1)})$ bound for high degree curves with maximal degree d, the practical efficiency is limited to lower degree algebraic curves because of the ubiquitous computation of common tangents of two curved edges. To improve the practical efficiency for higher degree algebraic boundary curves, it is necessary to reduce the time consuming curved edge operations. By generalizing the method of D.T.Lee [17] to a coordinate–based algorithm we suggest an efficient $O(n \cdot d^{O(1)})$ time algorithm to compute the convex hull of planar geometric object models bounded by algebraic curve segments (possibly non-smooth), with maximum degree d. Our algorithm computes the convex hull in a single pass using a single stack and subdivides the original problem into four subproblems. The convex hull computation of [22] requires two passes and subdivides the original problem into two subproblems. Souvaine [25] also suggests a convex hull computation algorithm based on a polygonal approximation and restricted to a class of curved objects (termed splinegons). Figure 1 shows a difference between simple polygons and curved objects, where a single edge may intersect two different pockets. It also shows a difference between our algorithm and other algorithms for the curved case. Both Schäffer and Van Wyk [22] and Souvaine [25] consider the edge C_j as an *event edge* and apply the time–consuming common tangent operation to such edges. Since edges with such orientation cannot belong to the convex hull boundary, these edges are ignored and not considered as event edges in our algorithm.

1.2 Decompositions

A *splinegon* is a polygon whose edges have been replaced by "well-behaved" curves and the *carrier* polygon of a splinegon is the polygon connecting adjacent vertices of a splinegon. Dobkin, Souvaine, and Van Wyk [13] show that the $O(n \log\log n)$ time algorithm for the horizontal-vertex-visibility partition of a simple polygon [26] can easily be generalized to a simple splinegon, and using this partition they present an algorithm to decompose a simple splinegon into a union of monotone pieces and further into a union of differences of unions of possibly overlapping convex pieces. They also show that simplifying the carrier polygon can be quite expensive by constructing an n-sided splinegon whose smallest simple carrier polygon has $\Omega(n^2)$ edges.

In this paper, we present an $O((n \log\log n + k) \cdot d^{O(1)})$ algorithm to compute a simple *carrier* polygon of planar geometric object model, where n is the number of monotone boundary curve segments and k is the number of edges in the resulting simple carrier polygon. Further, we show that the worst case upper bound of k is the optimal $O(n^2)$. We also present an $O((n \log\log n + K) \cdot d^{O(1)})$

algorithm to construct a simple *characteristic* carrier polygon of planar curved object, where K is the minimum number of edges for (possibly non-simple) characteristic carrier polygons of the object. A carrier polygon is *characteristic* if it differs from the original object by convex regions each of which is either totally contained in the interior of the object or in its exterior.

By refining this decomposition further and using *chords* and *wedges* of decomposed object edges, we can construct within a similar time bound $O((n \log\log n + K) \cdot d^{O(1)})$, an *inner* polygon (resp. an *outer* polygon) which is a simple polygon totally contained in (resp. totally containing) the geometric model. In contrast to the simple carrier polygon construction, the worst-case upper bound for K can be arbitrarily large as the inner angle between two adjacent edges approaches 0 or 2π, however, it is small in practice. K (henceforth called the *characteristic* number) in some sense represents the shape degeneracy of the geometric model. Using the simple inner, outer and characteristic polygons, we can compute (1) a convex decomposition of the geometric model as a difference of unions of disjoint convex models, (2) a decomposition of the geometric model as a union of disjoint certain primitive models.

1.3 Organization of This Paper

The rest of this paper is organized as follows. §2 describes certain preliminary informations of use in later sections. In §2.1 we describe the boundary representation for a planar geometric model with algebraic boundary curves. In §2.2 we present a monotone curve segmentation of boundary curve segments (a pre-processing step of our algorithm) and basic operations on these monotone curve segments. Algebraic curves are treated in each of two internal representations; namely, the implicit and the parametric forms, [2,28]. In §3 we present an $O(n \cdot d^6 \log d)$ (resp. $O(n \cdot (d^{12} \log d + T(d)))$) time algorithm to compute the convex hull of geometric models with parametric (resp. implicit) boundary curves. Here $T(d)$ is the worst case time taken to trace an algebraic curve segment of degree d, [5]. In §4 we present an $O((n \log\log n + K) \cdot d^3 \log d)$ (resp. $O((n \log\log n + K) \cdot (d^6 \log d + T(d)))$ time algorithm to compute various decompositions of geometric models, where K is the *characteristic* number of the object.

2 Preliminaries

In this section, we describe the algebraic boundary model of the planar curved object, and consider monotone curve segmentations and other related geometric operations on monotone curve segments.

2.1 Planar Geometric Model

A planar geometric model with algebraic boundary curves has the following boundary representation.

A single simple oriented cycle of algebraic curve edges, where each edge is directed and incident to two vertices. Each edge also has curve equations, which are implicit and/or rational parametric equations of algebraic curves. An algebraic curve is implicitly defined by a single polynomial equation $f(x, y) = 0$ and parametrically defined by the pair $(x = \frac{f_1(t)}{f_3(t)}, y = \frac{f_2(t)}{f_3(t)})$, where f_1, f_2 and f_3 are polynomials. Further an interior point is also provided on each implicitly defined edge which helps remove any geometric ambiguity in the case of vertices which are singularities of the algebraic curve, [1,21]. Finally, each vertex is exactly specified by Cartesian coordinates.

The curve equations for each edge are chosen such that the direction of the normal at each point of the edge is towards the exterior of the object. For a simple point on the curve the normal

is defined as the vector of partials to the curve evaluated at that point. For a singular point on the curve we associate a range of normal directions determined by normals to the tangents at the singular point. Further the orientation of the cycle of edges is such that the interior of the object is to the left when the edges are traversed.

2.2 Computations with Algebraic Curves

We assume some primitive geometric algorithms to manipulate algebraic curve segments, [2,4,5,7, 10,12,16]. Prior work has considered the generation of rational parametric equations for certain implicitly defined algebraic plane curves, [2], the generation of implicit equations for parametrically defined algebraic curves, [4], as well as the robust tracing of algebraic curve segments with correct connectivity, especially when tracing through complicated singularities, [5]. Tracing for instance is very useful in determining when a given point lies within a general algebraic curve segment. For this last problem the method of sorting along the curve [16], also provides an efficient solution for low degree algebraic curves.

2.2.1 Monotone Segmentation

In [8,9] we consider in some detail the monotone segmentation of a geometric model boundary and other geometric operations on monotone curve segments. We summarize below the time complexities of these operations which are of relevance to timing analysis of the algorithms described in later sections. Our model of computation is the arithmetic model where arithmetic operations have unit time cost, see [3]. We first define monotone edges. See [8,9] for more details on proofs of Lemmas below and criteria for detecting singular, extreme, inflection points, and convex, concave, linear edges.

Definition 2.1 *Let C be a directed boundary edge without any inflection or singular point. Then*
(1) *C is* convex $\Longleftrightarrow$ *the gradient of C turns counter-clockwise along C*
(2) *C is* concave $\Longleftrightarrow$ *the gradient of C turns clockwise along C*
(3) *C is* monotone $\Longleftrightarrow$ *C is either* convex, concave *or* linear, *and the interior of C doesn't include any extreme point along the x or y directions.*

Lemma 2.1 (1) *All the roots of a univariate polynomial equation of degree $O(d)$ can be computed in $O(d^3 \log d)$ time.*
(2) *The common solutions of two polynomial equations of degree $O(d)$ in two variables can be computed in $O(d^6 \log d)$ time.*
(3) *The common solutions of three (resp. four) polynomial equations of degree $O(d)$ in three (resp. four) variables can be computed in $O(d^{12})$ (resp. $O(d^{16})$) time.*

Proof (1) The squarefree part of a univariate polynomial can be calculated in $O(d \log^2 d)$ time using fast techniques for the required GCD computation and division steps, [3], and further all roots can be computed using root isolations in $O(d^3 \log d)$ time, see [23].
(2) We can eliminate one variable from two polynomial equations using the Sylvester resultant in $O(d^4 \log^3 d)$ time, see [10], and then compute the roots of the resulting univariate polynomial equation of degree $O(d^2)$ in $O(d^6 \log d)$ time. Doing this twice for each variable in turn together with the pairwise substitutions then allows computing the common solutions in overall $O(d^6 \log d)$ time.
(3) We can eliminate three (resp. four) variables from three (resp. four) polynomial equations

using the *u–resultant* in $O(d^9)$ (resp. $O(d^{12})$) time, see [12,27], and solve resulting equations of degree $O(d^3)$ (resp. $O(d^4)$) in one variable in $O(d^9 \log d)$ (resp. $O(d^{12} \log d)$) time. Though the resultant computation takes naively $O(d^9)$ (resp. $O(d^{12})$) time and solving real root solutions takes $O(d^9 \log d)$ (resp. $O(d^{12} \log d)$) time, the overall time is bounded by the pairwise substitution which takes $O(d^{12})$ (resp. $O(d^{16})$) time. □

Lemma 2.2 *For a parametric (resp. implicit) algebraic curve segment C of degree d, a monotone segmentation can be obtained in $O(d^3 \log d)$ (resp. $O(d^6 \log d + T(d))$) time, where $T(d)$ is the time required for the curve segment tracing.*

2.2.2 Basic Operations on Monotone Curve Segments

We consider primitive operations on monotone curve segments C and D, and a line segment L.

1. The intersection of C and L,
2. The containment of C in the upper–left halfplane $H^{UL}(L)$ of L,
3. The point p at which C has a tangent line L_p parallel to L,
4. The tangent line L_p of C from a point q,
5. The common tangent line $L_{p,q}$ of C and D.

Lemma 2.3 *For a parametric (resp. implicit) algebraic curve segment C,*

1. *the intersection $C \cap L$ can be computed in $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time,*
2. *the containment $C \subset H^{UL}(L)$ can be tested in $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time,*
3. *the tangent point p at C can be computed in $O(d^3 \log d)$ (resp. $O(d^6 \log d + T(d))$) time, and*
4. *the tangent line L_p of C from q can be computed in $O(d^3 \log d)$ (resp. $O(d^6 \log d + T(d))$) time.*

Lemma 2.4 *The common tangent line $L_{p,q}$ of C and D can be computed*

- *in $O(d^6 \log d)$ time if both C and D are parametric,*
- *in $O(d^9 \log d + T(d))$ time if only one of C and D is parametric,*
- *in $O(d^{12} \log d + T(d))$ time if both C and D are implicit.*

Proof : Though the common solutions of three (resp. four) polynomial equations as stated in Lemma 2.1 take $O(d^{12})$ (resp. $O(d^{16})$) time, for monotone curve segments and the application of common tangents, this can be reduced to $O(d^9 \log d + T(d))$ (resp. $O(d^{12} \log d + T(d))$) time. □

Now, initially assume there are $O(m)$ algebraic curve segments of maximum degree d on the model boundary. Then the monotone segmentation preprocessing can be done in $O(m \cdot d^3 \log d)$ (resp. $O(m \cdot (d^6 \log d + T(d)))$) time if all the boundary curve segments are parametric (resp. implicit). Each parametric (resp. implicit) algebraic curve segment of degree d can be segmented into $O(d)$ (resp. $O(d^2)$) monotone curve segments by adding extra vertices into singular points, inflection points and extreme points. After this preprocessing step of monotone segmentation, we let the total number of boundary edges be n, which is $O(m \cdot d)$ (resp. $O(m \cdot d^2)$) for parametric (resp. implicit) curves. In the following, we assume the object boundary is already segmented into $O(n)$ monotone curve segments and the timing analysis is given in terms of n.

3 Convex Hull of Geometric Model

In this section we present an algorithm to compute the convex hull of a planar geometric model bounded by $O(n)$ monotone curve segments. The algorithm runs in $O(n)$ steps, where each step takes polynomial time in the degree d. In the following we consider the construction of the lower–right subpart of the convex hull boundary which lies between the bottommost vertex p_0 and the rightmost vertex p_M of the original object. The entire convex hull is obtained by applying the same algorithm to the remaining three subparts. W.l.o.g. we may assume there are unique bottommost and rightmost vertices.

In the following, let $C_1, C_2, \ldots, C_M$ be a connected sequence of edges from p_0 to p_M, where each C_i has p_{i-1} and p_i as its starting and ending vertices resectively. For a point $p_\#$ (resp. $p^\#$) we denote its x and y-coordinates by $x_\#$ and $y_\#$ (resp. $x^\#$ and $y^\#$). We also denote the line segment connecting two points p and q by $L(p,q)$ and the path from p to q along the boundary curve by $\gamma(p,q)$. Also let $A \sim B$ denote everything in A which is not in B.

3.1 Sequences of Event Edge and Current Hull

We give a constructive definition of a sequence of *event edges* $\{C_{i_k}\}_{k=1}^N$ with $C_{i_N} = C_M$ and a sequence of *current hulls* $\{CH_k\}_{k=1}^N$. Further, we show that the N-th *current hull* CH_N is the lower–right subpart of the convex hull boundary between p_0 and p_M.

3.1.1 Definition of C_{i_k} and CH_k

Let $i_0 = 0$ and $CH_0 = \{p_0\}$. Assume that the index i_k and the k-th *current hull* CH_k $(0 \le k < N)$ have been defined. We define the (k+1)-th *event edge* $C_{i_{k+1}}$ and the (k+1)-th *event component* $EC_{k+1} \subset C_{i_{k+1}}$ in terms of i_k and CH_k as follows, see also Figures 2–3.

1. If $x_{i_k+1} \le x_{i_k}$ and the inner angle of $p_{i_k} < \pi$, then $i_{k+1} = \min\{j \mid j > i_k \text{ and } x_j > x_k\}$.
   (a) $EC_{k+1} = C_{i_{k+1}}$ if $y_{i_{k+1}-1} < y_{i_{k+1}}$ and $C_{i_{k+1}}$ is *convex*,
   (b) $EC_{k+1} = p_{i_{k+1}}$ otherwise.
2. If $x_{i_k} < x_{i_k+1}$ and $y_{i_k} < y_{i_k+1}$, then $i_{k+1} = i_k + 1$.
   (a) $EC_{k+1} = C_{i_{k+1}}$ if $C_{i_{k+1}}$ is *convex*,
   (b) $EC_{k+1} = p_{i_{k+1}}$ otherwise.
3. Otherwise, let j_0 be the smallest j such that $j > i_k + 1$, and either (p_{j-1} is not inside of any pocket of CH_k with $y_{j-1} < y_j$) or (C_j intersects with a lid $L(p', p'')$ at a point p_{**}, but it is not totally inside of the pocket implied by $L(p', p'')$, and further $x_{j-1} < x_j$, $y_{j-1} < y_j$, and $y_{**} < \min\{y_* \mid p_* \in \gamma(p_{i_k}, p_{j-1}) \cap L(p', p'')\}$.
   (a) If p_{j_0-1} is not inside of any pocket of CH_k and the inner angle of $p_{j_0-1} > \pi$, then $i_{k+1} = j_0 - 1$ and $EC_{k+1} = p_{i_{k+1}}$.
   (b) Otherwise, $i_{k+1} = j_0$ and
      i. $EC_{k+1} = C_{i_{k+1}}$ if $C_{i_{k+1}}$ is *convex*, and
      ii. $EC_{k+1} = L(p_{i_{k+1}-1}, p_{i_{k+1}})$ otherwise.

Next we inductively define the (k+1)-th *current hull* CH_{k+1}. It is easy to show there is a unique common tangent line $L_{p',p''}$ of CH_k and EC_{k+1} (with $x' < x''$ and $y' < y''$) such that $CH_k \cup EC_{k+1} \subset H^{UL}(L)$. If there is more than one choice of p' (resp. p''), we choose p' (resp. p'') so that the distance between p' and p'' is minimal. Further, let $FRONT\_CH_{k+1}$ denote the front subarc of CH_k between the points p_0 and p', and $REAR\_CH_{k+1}$ denote the rear subarc of EC_{k+1} between the points p'' and $p_{i_{k+1}}$. The (k+1)-th *current hull* CH_{k+1} is defined as the connected union $FRONT\_CH_{k+1} \cup L(p',p'') \cup REAR\_CH_{k+1}$, see Figure 4. CH_{k+1} is a convex arc along which both x and y-coordinates are strictly increasing. $L(p',p'')$ is called the *lid* determined by p' and p''. Let $\bar{\gamma}$ be the closed path given as $\gamma(p',p'')$ followed by a path from p'' to p' along $L(p',p'')$. If $\bar{\gamma}$ has no self-intersection, the region bounded by $\bar{\gamma}$ is called the *pocket* determined by the lid $L(p',p'')$. Otherwise, $\gamma(p',p'')$ has an even number of intersections with the lid $L(p',p'')$ counting intersections with multiplicities and $\bar{\gamma}$ divides the plane into a finite number of connected regions. The union of all the regions which are to the right of $\bar{\gamma}$ is the *pocket* implied by $L(p',p'')$, see Figures 5–6.

3.1.2 Properties of CH_k

We prove two important properties of CH_k in the following Lemmas 3.1–3.2.

Lemma 3.1 *If a point $p \in C_i$ $(1 \le i \le i_k)$ is on the convex hull boundary, then $p \in CH_k$.*

Proof : Using induction we can easily show that the interior of the path $\gamma(p_{i_k}, p_{i_{k+1}-1})$, the arc $C_{i_{k+1}} \sim EC_{k+1}$, and $(CH_k \cup C_{i_{k+1}}) \sim CH_{k+1}$ are in the convex hull interior. □

Lemma 3.2 *If a point $p \in CH_k$ is on the convex hull boundary, then the front subarc of CH_k between p_0 and p is contained entirely in the convex hull boundary.*

Proof: The case $k = 1$ is easy to show. By induction, we assume for k $(1 \le k < N)$ and consider $k+1$. Suppose a point $p \in CH_{k+1}$ is on the convex hull boundary. (a) If $p \in FRONT\_CH_{k+1} \subset CH_k$, then the statement follows by induction. (b) If $p \in L(p',p'')$, then $L(p',p'')$ is also on the convex hull boundary. Further, $FRONT\_CH_{k+1}$ is on the convex hull boundary by induction. (c) If $p \in REAR\_CH_{k+1}$, then there is a supporting line L_p at p. We now prove that the lid $L(p',p'')$ is on the convex hull boundary. Suppose there is a boundary point q in the region R_1, see Figure region. We may assume q is extreme to the outward normal direction of the lid and thus on the convex hull boundary. (i) If $q \in C_j$ $(1 \le j \le i_{k+1})$, then Lemma 4.1.2 implies $q \in CH_{k+1}$, however, this is impossible since CH_{k+1} is convex. (ii) Otherwise, there is a continuous path from $p_{i_{k+1}}$ to q. This path should pass through either the region R_2 or R_3, however, both are impossible. Hence the lid $L(p',p'')$ is on the convex hull boundary, and by induction $FRONT\_CH_{k+1}$ is also on the convex hull boundary. Similarly one can show that the subarc of $REAR\_CH_{k+1}$ between p'' and p is on the convex hull boundary. □

Since p_M is on the convex hull boundary and it is the end point of CH_N, Theorem 3.1 below follows easily from Lemmas 3.1–3.2.

Theorem 3.1 *CH_N is the lower-right part of the convex hull boundary between p_0 and p_M.*

3.2 Description of Algorithm

We describe an algorithm to compute the sequences of *event edges* $\{C_{i_k}\}_{k=1}^{N}$ and *current hulls* $\{CH_k\}_{k=1}^{N}$ by using a single stack CH. CH contains segments of the k-th *current hull* CH_k which are subarcs of some *convex* edges, some *linear* edges and the lids of pockets. Adjacent elements on the stack share a common end point and the connected sequence of elements on the stack CH generate the *current hull* CH_k (we say then that "the stack CH *implies* the *current hull* CH_k").

3.2.1 Computing Event Edges

We start with pushing a single point interval $[p_0, p_0]$ into an empty stack CH. The stack CH implies the *current hull* $CH_0 = \{p_0\}$. Assume i_k is detected and the stack CH implies the k-th *current hull* CH_k $(0 \leq k < N)$. We consider how to detect i_{k+1} using the stack CH. Since the cases 1 and 2 of §3.1.1 are easy, we consider only case 3. We detect $j_0 = \min \Omega_k$ by using a loop variable j initialized to i_k and maintaining the following invariant for j at the beginning of each loop :

"p_j is outside (including the boundary) of any pocket of CH_k and TOP(CH) is not strictly below the horizontal line $y = y_j$."

In each loop, j is first incremented by 1. Then, (1) If $y_{j-1} \leq y_j$, then $j_0 = j$. (2) If $y_{j-1} > y_j$, then pop all the stack elements until (a) TOP(CH) which does not intersect with C_j and is not strictly above the line $y = y_j$, or (b) TOP(CH) which is a lid intersecting with C_j. In the case (a), repeat the j–loop. In the case (b), initially let p_* be the first intersection of C_j with $L(p', p'')$. Next, walk along the path $\gamma(p_{j-1}, p_M)$ and count the number of left–to–right cuts and right–to–left cuts this path makes with the lid $L(p', p'')$. At each intersection point, say p_{**}, make $p_* = p_{**}$ if $y_{**} \leq y_*$. While the number of left–to–right cuts is larger than right–to-left cuts, we are inside of the pocket implied by $L(p', p'')$. In Figure ??, a path $\gamma(p_*, p_j)$ is totally contained in a self–intersecting pocket and has its first interior intersection with a lid $L(p', p'')$ in a right–to–left direction. When $\gamma(p_*, p_j)$ comes out of a pocket through a point p_{**}, the total cuts in both directions are equal. If these two numbers become equal at a point $p_{**} \in C_{j'}$, compare y_{**} with y_*. If $y_{**} > y_*$, then continue the walking along the path $\gamma(p_{**}, p_M)$. If $y_{**} \leq y_*$, then (i) let $j_0 = j'$ if $y_{j'-1} \leq y_{j'}$, and (ii) otherwise, let $j = j'$ and further treat p_{**} as p_j and repeat the j-loop. It is easy to show that the j–loop invariant holds everytime the loop is repeated and j_0 is detected correctly.

3.2.2 Computing Current Hulls

We consider next how to construct CH_{k+1} from CH_k and EC_{k+1} by using the stack CH. Though we have popped some stack elements from CH, the elements on the stack CH imply a convex arc Γ which contains $FRONT\_CH_{k+1}$ as its front subarc. To remove redundant elements $\Gamma \sim FRONT\_CH_{k+1}$ from CH, we consider the top stack element TOP(CH). We check whether TOP(CH) contains the common tangent point p' for the new lid $L(p', p'')$ of CH_{k+1}. Since EC_{k+1} is not strictly below $y = y_S$, we have (1) $p' = p_E$ if and only if $EC_{k+1} \subset H^{UL}(L_{p_E})$ and EC_{k+1} is not strictly below the horizontal line $y = y_E$, (2) $p' \in$ TOP(CH) if and only if $EC_{k+1} \subset H^{UL}(L_{p_S})$, and (3) $p' \notin$ TOP(CH) otherwise. Here p_S and p_E are the starting and ending points of TOP(CH), and L_{p_S} and L_{p_E} are the tangent lines of TOP(CH) at p_S and p_E respectively. In the cases (a) and (b), p' and p'' can be computed by using Lemmas 2.3–2.4. In the case (c), we pop TOP(CH) and repeat the same procedure. Once we have computed p' and p'', we can adjust the stack appropriately to imply CH_{k+1}.

3.3 Timing Analysis

There are basically two types of operations in this algorithm. Operations to manipulate the edge sequence itself or operations to manipulate the stack. Each edge as input is processed within a constant number of steps before it is determined whether it is an event edge or not. If it is an event edge, a segment of this edge is pushed on the stack. While an edge is on the stack, it may be used to process other edges and/or be changed into a shorter subsegment. Finally, it is popped if it is not on the convex hull boundary or it appears in the final output as an edge of the convex hull boundary.

Before an edge is determined to be an event edge or not, we apply to this edge such computations like coordinate comparisons, an inner angle computation, and intersections with stack elements. If a stack element is popped after certain computations, we charge the cost of these computations to this popped stack element. Since the popping occurs at most once to an edge, the cost at the popping time will be charged at most once to each edge. Further, there is at most one stack element which is involved in the operation with the input edge and still remains on the stack. We can charge this cost and other trivial computation costs to the input edge. The most expensive computation cost here is line–curve segment intersection which is $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time for a parametric (resp. implicit) curve segment. Since there are totally $O(n)$ input edges and popped edges, the total cost for event edge detections is $O(n \cdot d^3 \log d)$ (resp. $O(n \cdot (d^3 \log d + T(d)))$) time.

After an event edge is detected, the stack is modified to imply the new current hull. The stack elements are popped after two halfplane containment testings which cost $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time. The new stack is constructed by computing the lid $L(p', p'')$ and modifying itself appropriately, which costs at most $O(d^6 \log d)$ (resp. $O(d^{12} \log d + T(d))$) time for common tangent computations. Note that this last operation takes $O(1)$ time if both p' and p'' are known, and $O(d^3 \log d)$ ($O(d^6 \log d + T(d))$) time if one of p' and p'' is known. We charge this cost to the event edge. Thus, the total cost for the current hull computation is $O(n \cdot d^6 \log d)$ (resp. $O(n \cdot (d^{12} \log d + T(d)))$) time, which is also the overall time complexity of the convex hull algorithm.

4 Decomposition of Planar Geometric Models

4.1 Decomposition of Monotone Boundary Edges

In this section, we consider how to construct a simple carrier polygon of a geometric model object with at most $O(n^2)$ edges, which is optimal since this number is shown to be $\Omega(n^2)$ in [13]. Further, we consider how to construct a simple *characteristic* carrier polygon, an *inner* polygon and an *outer* polygon of the geometric model. We assume the model boundary has been decomposed into $O(n)$ *monotone* edges in a preprocessing step as discussed in §2.

4.1.1 Simple Carrier Polygon

Consider the horizontal vertex visibility partition of both the interior and exterior of a geometric model, see [13,26], where the exterior is partitioned within a box enclosing the object, see Figure 10. Let H be a visibility cell of the partition, and the right and left sides of H be bounded by the edges C and D respectively, see Figure 9. Note that each side of H may be a proper subsegment of C or D. Let us assume H is in the interior of the geometric model and C is a convex edge. The cases of H being in the exterior and/or C being a concave edge can be handled in similar ways. Let p_B and p_T be the bottom and top points of the right side of H, and $\bar{C}$ be the subsegment of C between p_B

and p_T. Further, let q_B and q_T be the bottom and top points of the left side of H, and $\bar{D}$ be the subsegment of D between q_B and q_T. To make the construction easier, we add p_B and p_T (resp. q_B and q_T) as extra vertices to C (resp. D). Since there are only $O(n)$ such extra vertices, the total number of edges after this decomposition is still $O(n)$.

We can add extra vertices $p_1, p_2, \ldots, p_{k_{\bar{C}}}$ to the edge $\bar{C}$ so that the *carrier polygonal arc* $P_{\bar{C}}$ connecting the vertices $p_B, p_1, \ldots, p_{k_{\bar{C}}}, p_T$ does not intersect with any other part of the carrier polygon except at p_B and p_T and also with the extra vertices the carrier polygon has at most $O(n^2)$ edges. This is achieved by the following construction. At each vertex p, construct a horizontal line L containing p and parallel to the x-axis. Let $p_1, p_2, \ldots, p_{k_{\bar{C}}}$ (resp. $q_1, q_2, \ldots, q_{k_{\bar{D}}}$) be the intersection points of $\bar{C}$ (resp. $\bar{D}$) with these horizontal lines. Then the corresponding carrier polygonal arcs $P_{\bar{C}}$ and $P_{\bar{D}}$ do not intersect except at the end points and further it follows that the carrier polygon $\cup P_{\bar{C}}$ is simple. Since there are $O(n)$ such horizontal lines and boundary edges, and $k_{\bar{C}} = O(n)$, the carrier polygon has $O(n^2)$ edges. Though $O(n^2)$ is optimal asymptotically, the above construction does not generate the minimal number of extra vertices. Steps can be taken to reduce the number of extra vertices added. For example, when the chord $L(p_B, p_T)$ of $\bar{C}$ does not intersect with $\bar{D}$ except at p_B and p_T, we do not need to add any extra vertices. Thus $P_{\bar{C}}$ is $L(p_B, p_T)$ with $k_{\bar{C}} = 0$. Further, when $L(p_B, p_T)$ intersects with $\bar{D}$ at a point other than p_B and p_T, we construct $P_{\bar{C}}$ so that $P_{\bar{C}}$ does not intersect with $P_{\bar{D}}$ except at p_B and p_T though $P_{\bar{C}}$ may intersect with the edge $\bar{D}$. Thus, we construct $P_{\bar{D}}$ recursively. Assume we have constructed $P_{\bar{D}}$ by adding a small number of extra vertices to $\bar{D}$. Then by scanning H from the top to the bottom, we can add at most $k_{\bar{D}}$ extra vertices to the edge $\bar{C}$ to make the corresponding carrier polygonal arc $P_{\bar{C}}$ simple. Thus we have the relation $k_{\bar{C}} \leq k_{\bar{D}}$. Though for simplicity we assume each boundary edge is segmented into monotone edges and each monotone edge is further decomposed so that each side of H is an edge, we can easily modify the above construction so that we may need to add extra vertices only to y-extreme points and apply the same recursive construction to add a minimal number of extra vertices to each y-monotone edge.

Theorem 4.1 *Assume all the monotone edges are parametric (resp. implicit) algebraic curve segments. A simple carrier polygon of an object with at most $O(n^2)$ edges can be constructed in $O((n \log\log n + k) \cdot d^3 \log d)$ (resp. $O((n \log\log n + k) \cdot (d^3 \log d + T(d)))$) time, where k is the number of edges in the simple carrier polygon.*

Proof: The horizontal vertex visibility partition of both the interior and exterior of an object can be constructed in $O(n \log\log n \cdot d^3 \log d)$ (resp. $O(n \log\log n \cdot (d^3 \log d + T(d)))$) time. A simple carrier polygon can be constructed in $O(k \cdot d^3 \log d)$ (resp. $O(k \cdot (d^3 \log d + T(d)))$) time by computing $O(k)$ line–curve segment intersections. □

4.1.2 Simple Characteristic Carrier Polygon

A carrier polygon is *characteristic* if, for each edge E, $S(E)$ is totally contained either in the interior of the geometric model or in the exterior of the model, where $S(E)$ is the convex region bounded by the edge E and its chord. If E is a convex (resp. a concave) edge, then $S(E)$ is totally contained in the interior (resp. in the exterior) of the model and is called an *additive* (resp. a *subtractive*) convex region. Assume C and D are the same as $\bar{C}$ and $\bar{D}$ of §3.1 respectively. We can add extra vertices $p_1, p_2, \ldots, p_{k_C}$ to the edge C so that the convex regions of the decomposed subsegments of C are *additive* convex regions. The case of C being a concave edge can be handled in a similar way and the convex regions of the decomposed subsegments of C become *subtractive* convex regions in

this case. This decomposition is achieved as follows. If $S(C)$ and $S(D)$ do not intersect, we do not add any extra vertex to C and $S(C)$ is an additive convex region, thus $k_C = 0$. Otherwise, let L_1 be the tangent line from p_B to D, p_1 be the intersection point of L_1 with C, and C_1 be the subsegment of C between p_B and p_1. Then, $S(C_1)$ is an additive convex region. Let $\bar{C}$ be the subsegment of C between p_1 and p_T. If $S(\bar{C})$ intersects with $S(D)$, then we compute a tangent line L_2 from p_1 to D and the intersection point p_2 of L_2 and C, and repeat the same procedure. Otherwise, $S(\bar{C})$ is an additive convex region. Now, we prove the decomposition of C terminates within a finite number of steps.

Theorem 4.2 *Assume no vertex has its inner angle as 0 or 2π. Each edge C can be decomposed into a finite number of subedges $C_1, C_2, \ldots, C_{k_C}$ so that the convex regions are additive.*

Proof Suppose there is an infinite sequence of C_i's ($i = 1, 2, \ldots$) constructed as above. Let p_i be the end point of C_i, then $x_S < x_i < x_{i+1} < x_E$ for all i. Since x_i is a strictly increasing sequence bounded above, $x_i \rightarrow x$ for some $x \leq x_E$. Let $p_L \in C$ be such that $x_L = x$, then $p_i \rightarrow p_L$. In an arbitrary small neighborhood U of p_L, there is an integer N such that $p_i \in U$ and $C_i \subset U$ for all $i \geq N$. Let q_i be the tangent point of $L(p_{i-1}, p_i)$ with D, then $q_i \in U$ for all $i \geq N$. Since q_i's are on D and in the interior of $S(C)$, this means that p_L is a limit point of D which are in the interior of $S(C)$. We can easily show that D is a concave edge and p_L is a common vertex of C and D, and further the inner angle at $p_L > 3\pi/2$ (resp. $< \pi/2$). Since $L(p_{i-1}, p_i)$ is tangent to D at q_i with $p_i \rightarrow p_L$ and $q_i \rightarrow p_L$, C and D have the same tangent line at p_L. Hence, the inner angle of the model at p_L is 2π (resp. 0). It is impossible since we assume no such vertex on the geometric model boundary. □

We call the polygonal arc P_C^X connecting the vertices $p_S, p_1, \ldots, p_{k_C}, p_E$ as the *first characteristic polygonal arc* of C, the union $\cup P_C^X$ as the *first characteristic polygon* of the geometric model, and $K = \sum(k_C + 1)$ as the *characteristic number* of the object. It is easy to show that the edges of characteristic polygon do not intersect each other transversally and two different vertices do not overlap. However, a vertex may lie on the interior of some characteristic polygon edge. By decomposing each edge with a vertex on its chord interior into two subedges, we can make the carrier polygon simple. Thus, using at most $2K$ edges we can construct a simple characteristic carrier polygon.

Theorem 4.3 *Assume all the monotone edges are parametric (resp. implicit). A simple characteristic carrier polygon of an object with at most $2K$ edges can be computed in $O((n \log\log n + K) \cdot d^3 \log d)$ (resp. $O((n \log\log n + K) \cdot (d^6 \cdot \log d + T(d)))$) time.*

Proof: After the horizontal vertex visibility partition is computed, a simple characteristic carrier polygon can be constructed in $O(K \cdot d^3 \log d)$ (resp. $O(K \cdot (d^6 \log d + T(d))))$ time by computing $O(K)$ tangent lines from given points. □

4.1.3 Inner and Outer Polygons

Let C be a monotone edge with p_S as its starting point and p_E as its ending point. For any two different points p and q on C, $L(p, q)$ denotes the line segment connecting p and q, and L_p denotes the tangent line of C at p. Let p^* be the intersection point of two tangent lines L_{p_S} and L_{p_E}. We call the line segment $L(p_S, p_E)$ as the *chord* of C and the polygonal arc $\wedge(C) = L(p_S, p^*) \cup L(p^*, p_E)$ as the *wedge* of C. Let $p_S, p_1, p_2, \ldots, p_k, p_E$ be a sequence of points on C in the order they appear

along C, then the polygonal arc $P_C^{chord}(p_S, p_1, p_2, \ldots, p_k, p_E) = L(p_S, p_1) \cup L(p_1, p_2) \cup \ldots \cup L(p_k, p_E)$ is called as the *chordal* polygonal arc of C determined by the sequence p_S, p_1, p_2, $\ldots$, p_k, p_E. Let p_i^* be the intersection point of $L_{p_{i-1}}$ and L_{p_i} $(i = 1, \ldots, k+1)$, where $p_0 = p_S$ and $p_{k+1} = p_E$. Then the polygonal arc $P_C^{tangent}(p_S, p_1, p_2, \ldots, p_k, p_E) = L(p_S, p_1^*) \cup L(p_1^*, p_2^*) \cup \ldots \cup L(p_k^*, p_{k+1}^*) \cup L(p_{k+1}^*, p_E)$ is called as the *tangential* polygonal arc of C determined by the sequence p_S, p_1, p_2, $\ldots$, p_k, p_E, see Figure 11.

If we decompose the geometric model boundary so that the chords of convex (resp. concave) decomposed edges and the wedges of concave (resp. convex) decomposed edges are totally contained in the interior (resp. the exterior) of the model, then the union of these chords and wedges defines an *inner* (resp. *outer*) polygon which is totally contained in the interior (resp. the exterior) of the model. In the following, we will consider the construction of *inner* polygonal arcs P_C^{chord} of edges C. The *outer* polygonal arcs $P_C^{tangent}$ of edges C can be constructed in a similar way.

We first consider the case of C being a convex edge. Let $P_C^{\times}$ be the first characteristic polygonal arc of C, then C is to the right of $P_C^{\times}$ and D is to the left of $P_C^{\times}$. We may assume $p_B \neq q_B$ or $p_T \neq q_T$. If D is a convex edge, then $P_C^{\times}$ is the chord $L(p_B, p_T)$ of C and the chords of C and D do not intersect except at an end point. Further, the chords of C and D are contained in the interior of the object. We call $L(p_B, p_T)$ and $L(q_B, q_T)$ as the *inner* polygonal arcs of C and D respectively. Now, if D is a concave edge, there are points q_1, q_2, $\ldots$, q_{k_C} on D which are tangent to some lines segments on $P_C^{\times}$. Let p_B, p_1, p_2, $\ldots$, p_{k_C}, p_T be the vertices of $P_C^{\times}$ in the order they appear along C. Note that $q_1 = q_B$ if $p_B = q_B$, and $k_C = 0$ if $D \cap P_C^{\times} = \emptyset$. Further, q_i is the tangent point of $L(p_{i-1}, p_i)$ on D $(i = 1, \ldots, k_C)$, where $p_0 = p_B$ and $p_{k_C+1} = p_T$. We add an extra vertex p'_i to each subedge of C between p_{i-1} and p_i $(i = 1, \ldots, k_C + 1)$. We call the polygon P_C^{inner} connecting $p_B, p'_1, p_1, \ldots, p_{k_C}, p'_{k_C+1}, p_T$ as the *inner* polygonal arc of C. P_C^{inner} is strictly to the right of $P_C^{\times}$ except the points $p_B, p_1, p_2, \ldots, p_{k_C}, p_T$. Further, we add an extra vertex q'_i to each subedge of D between q_{i-1} and q_i $(i = 1, \ldots, k_C + 1)$, where $q_0 = q_B$ and $q_{k_C+1} = q_T$. We choose q'_{k_C+1} so that the tangent line of D at this point is parallel to the line segment $L(p_{k_C-1}, p_{k_C})$. Note that $q_B = q'_1 = q_1$ if $p_B = q_B$, and $q_T = q'_{k_C+1}$ if $p_T = q_T$. We call the tangential polygonal arc $P_D^{tangent}(q_B, q'_1, q_1, \ldots, q_{k_C}, q'_{k_C+1}, q_T)$ as the inner polygonal arc P_D^{inner} of D.

We consider the case of C being a concave edge. Since the case of D being convex can be handled in a similar way as above, we assume D is concave. There are two common tangent lines L_1 and L_2 of C and D. Let p^* be the intersection point of L_1 and L_2, and L be a line containing the point p^* and having its slope strictly between the slopes of L_1 and L_2. Let p' (resp. q') be a point on C (resp. D) at which C (resp. D) has a tangent line parallel to the line L. We call the tangential polygonal arc $P_C^{tangent}(p_B, p', p_T)$ (resp. $P_D^{tangent}(q_B, q', q_T)$) as the *inner* polygonal arc P_C^{inner} of C (resp. P_D^{inner} of D). Since P_C^{inner} (resp. P_D^{inner}) is strictly to the right (resp. to the left) of L, $P_C^{inner} \cap P_D^{inner} = \emptyset$.

Theorem 4.4 *Assume all the monotone edges are parametric (resp. implicit). Both inner and outer carrier polygons of an object with at most $2K$ edges can be computed in $O((n \log\log n + K) \cdot d^3 \log d)$ (resp. $O((n \log\log n + K) \cdot (d^6 \cdot \log d + T(d)))$) time.*

Proof: Similar to Theorem 4.3. □

4.2 Object Decompositions

We consider various applications of the polygons $P^{\times}$ and P^{inner} constructed in §3 to the object decompositions. Dobkin, Souvaine, and Van Wyk [13] shows an $O(n \log\log n)$ algorithm to decompose a simple splinegon into a union of differences of unions of possibly overlapping convex

pieces. Our decomposition may involve $O(K)$ where K is often linear in practice. Further, the decomposition structure in terms of unions and differences is simpler than that of [13]. However, when the simple characteristic polygon has small number of edges like K is almost linear, then our decomposition method may be more useful since this decomposition has nicer structure.

4.2.1 Convex Decomposition

We can decompose the simple characteristic polygon P^X into unions of disjoint convex polygons $\cup_i P_i$, (see [18] for a survey on convex decomposition algorithms for simple polygons). Let $\cup_j U_j$ (resp. $\cup_k V_k$) be the union of all the *additive* (resp. *subtractive*) convex regions. Then, the original object can be represented as $(\cup_i P_i) \cup (\cup_j U_j) \sim (\cup_k V_k)$. Further, the interiors of P_i's and U_j's are disjoint, and the interiors of U_j's and V_k's are disjoint, however, the interiors of P_i's and V_k's may have non-empty intersections. Thus the orders of union operations in the unions $(\cup_i P_i) \cup (\cup_j U_j)$ and $\cup_k V_k$ are interchangeable. Further, as long as $\cup_i P_i$ has been computed first, the order of adding each U_j and subtracting each V_k is interchangeable. The construction of P^X is highly parallel and would be useful in a parallel implementation of the model decomposition algorithm.

4.2.2 Primitive Decomposition

The main purpose of geometric model decomposition is to simplify a problem for complex geometric model into a number of simpler subproblems dealing with "nice" boundary. In most of the cases a decomposition in terms of a union of disjoint convex pieces is useful and this is always possible for simple polygons. However, this fact is certainly not true for planar geometric models. In §4.1 we thus considered an alternative way, namely in decomposing an object into unions and differences of convex objects. However, in some applications involving a Minkowski operation (i.e. convolution) which commutes with union we may consider decomposing a geometric model into a disjoint union of certain primitive objects.

We can decompose an inner polygon P^{inner} into a union of disjoint convex polygons $\cup_i P_i^{inner}$. The difference between the model and the inner polygon P^{inner} is the union $(\cup_j U_j^{inner}) \cup (\cup_k V_k^{inner})$, where each U_j^{inner} is an additive convex region bounded by a convex edge and its chord and each V_k^{inner} is a region bounded by a concave edge and its wedge. We can thus represent the original model as a disjoint union $(\cup_i P_i^{inner}) \cup (\cup_j U_j^{inner}) \cup (\cup_k V_k^{inner})$.

Now, consider the application of this simple decomposition to Minkowski operation. For objects A and B, the Minkowski addition $A \oplus B = \{a+b \mid a \in A \text{ and } b \in B\}$ and the Minkowski subtraction $A \ominus B = \{a - b \mid a \in A \text{ and } b \in B\}$. Let $A = \cup_i S_i$ and $B = \cup_{i'} S'_{i'}$, where S_i and $S'_{i'}$ are simple objects. Then $A \oplus B = \cup(S_i \oplus S'_{i'})$ and $A \ominus B = \cup(S_i \ominus S'_{i'})$.

5 Conclusion

In this paper, we presented an $O(n \cdot (d^{12} \log d + T(d)))$ algorithm to compute the convex hull of planar geometric model, bounded by $O(n)$ algebraic monotone curve segments of maximal degree d. We also presented an $O((n \log\log n + K) \cdot (d^6 \log d + T(d)))$ time algorithm to compute various decomposition of geometric models, where K is a number which captures the shape degeneracy of the geometric model. It should be noted that this worst case time bound in computing the exponent of the $d^{O(1)}$ is for implicitly defined algebraic curves. This exponent is considerably less for curves in their parametric representation. Also in practice essentially all planar geometric models considered are those bounded by curves of maximal degree $d \leq 4$ of which all $d = 2$ and

most $d = 3$ curves are parametrizable. Furthermore, our algorithms rely on simple data structures and are implementable.

References

[1] Abhyankar, S., (1983), "Desingularization of Plane Curves," *Proc. of the Symp. in Pure Mathematics*, Vol. 19, No. 1, pp. 11–14.

[2] Abhyankar, S., and Bajaj, C., (1988), "Automatic Rational Parameterization of Curves and Surfaces III: Algebraic Plane Curves," *Computer Aided Geometric Design*, to appear.

[3] Aho, A., Hopcroft, J., and Ullman, J., (1974), *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass.

[4] Bajaj, C., (1988), "Algorithmic Implicitization of Algebraic Curves and Surfaces," CAPO Research Report CER-88-11, Department of Computer Science, Purdue University.

[5] Bajaj, C., Hoffmann, C.M., Hopcroft, J.E., and Lynch, R.E., (1988), "Tracing Surface Intersections," *Computer Aided Geometric Design*, to appear.

[6] Bajaj, C., and Kim, M.-S., (1987), "Compliant Motion Planning with Geometric Models," *Proc. of the 3rd ACM Symposium on Computational Geometry*, pp. 171–180, Modified version with title "Generation of Configuration Space Obstacles : The Case of Moving Algebraic Surfaces," to appear in *International Journal of Robotics Research.*

[7] Bajaj, C., and Kim, M.-S., (1987), "Generation of Configuration Space Obstacles : The Case of Moving Algebraic Curves," *Proc. 1987 IEEE International Conference on Robotics and Automation*, North Carolina, pp. 979–984, Modified version to appear in *Algorithmica.*

[8] Bajaj, C., and Kim, M.-S., (1987), "Decompositions of Objects Bounded by Algebraic Curves," Computer Science Technical Report CSD-TR-677, Purdue University.

[9] Bajaj, C., and Kim, M.-S., (1987), "Convex Hull of Objects Bounded by Algebraic Curves," Computer Science Technical Report CSD-TR-697, Purdue University.

[10] Bajaj, C., and Royappa, A., (1988), "A Note on an Efficient Implementation of the Sylvester Resultant for Multivariate Polynomials," Computer Science Technical Report CSD-TR-718, Purdue University.

[11] Bhattacharya, B., and El Gindy, H., (1984), "A New Linear Convex Hull Algorithm for Simple Polygons," *IEEE Trans. Inform. Theory*, Vol. 30, No. 1, pp. 85–88.

[12] Canny, J., (1987), "The Complexity of Robot Motion Planning," Ph.D Thesis, Dept. of EE and CS, MIT.

[13] Dobkin, D.P., Souvaine, D.L., and Van Wyk, C.J., (1986), "Decomposition and Intersection of Simple Splinegons," Computer Science Technical Report CS-TR-051-86, Princeton University.

[14] Graham, R., and Yao, F., (1983), "Finding the Convex Hull of a Simple Polygon," *Journal of Algorithms*, Vol. 4, pp. 324–331.

[15] Hopcroft., (1986), "The Impact of Robotics on Computer Science," *CACM*, Vol. 29, No. 6, pp. 486–498.

[16] Johnston, J., (1987), "The Sorting of Points along an Algebraic Curve," Ph.D Thesis, Cornell Univeristy.

[17] Lee, D.T., (1983), "On Finding the Convex Hull of a Simple Polygon," *International Journal of Computer and Information Sciences*, Vol. 12, No. 2, pp. 87–98.

[18] Lee, D.T., and Preparata, F.P., (1984), "Computational Geometry – A Survey," *IEEE Transactions on Computers*, Vol. C-33, No. 12, pp. 872–1101.

[19] McCallum, D., and Avis, D., (1979), "A Linear Algorithm for Finding the Convex Hull of a Simple Polygon," *Information Processing Letters*, Vol. 9, No. 5, pp. 201-206.

[20] Preparata, F.P., and Shamos, M.I., (1985), *Computational Geometry: An Introduction*, Springer-Verlag, New York.

[21] Requicha, A., (1980), "Representations of Rigid Solid Objects," *Computer Aided Design*, Springer Lecture Notes in Computer Science 89, pp. 2–78.

[22] Schäffer, A., and Van Wyk C., (1987), "Convex Hulls of Piecewise-Smooth Jordan Curves," *Journal of Algorithms*, Vol. 8, No. 1, pp. 66–94.

[23] Schwartz, J. and Sharir, M., (1983), "On the Piano Movers Problem: II, General Techniques for Computing Topological Properties of Real Algebraic Manifolds," *Adv. Appl. Math.* Vol. 4, pp. 298–351.

[24] Schwartz, J. and Sharir, M., (1986), "Motion Planning and Related Geometric Algorithms in Robotics," Robotics Research Technical Report, No. 79, New York University.

[25] Souvaine, D.L., (1986), *Computational Geometry in a Curved World*, Ph.D Thesis, Computer Science Technical Report CS-TR-094-87, Princeton University.

[26] Tarjan, R.E., and Van Wyk, C.J., (1988), "An $O(n \log\log n)$-Time Algorithm for Triangulating Simple Polygons", *SIAM Journal on Computing*, Vol. 17, No. 1, pp. 143–178.

[27] van der Waerden, B., (1950), *Modern Algebra Vol. II*, New York, Ungar.

[28] Walker, R., (1978), *Algebraic Curves*, Springer Verlag, New York.

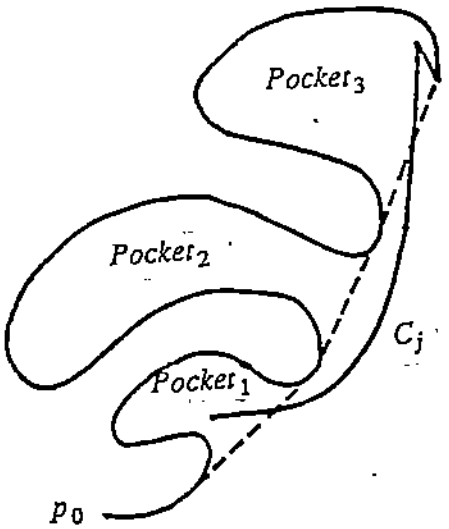

Figure 1: Difference between Polygonal and Curved Cases

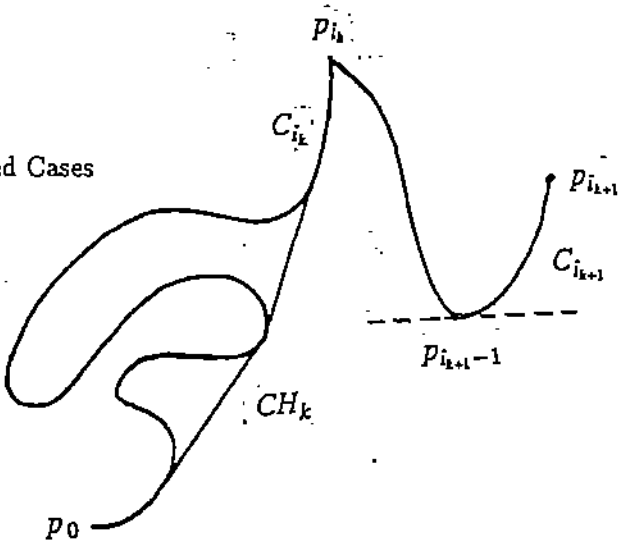

Figure 2: Event Edge of Case 3 with $EC_{k+1} = C_{i_{k+1}}$

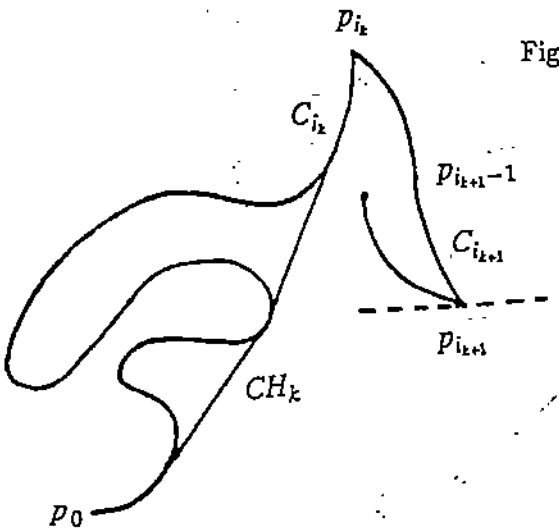

Figure 3: Event Edge of Case 3 with $EC_{k+1} = p_{i_{k+1}}$

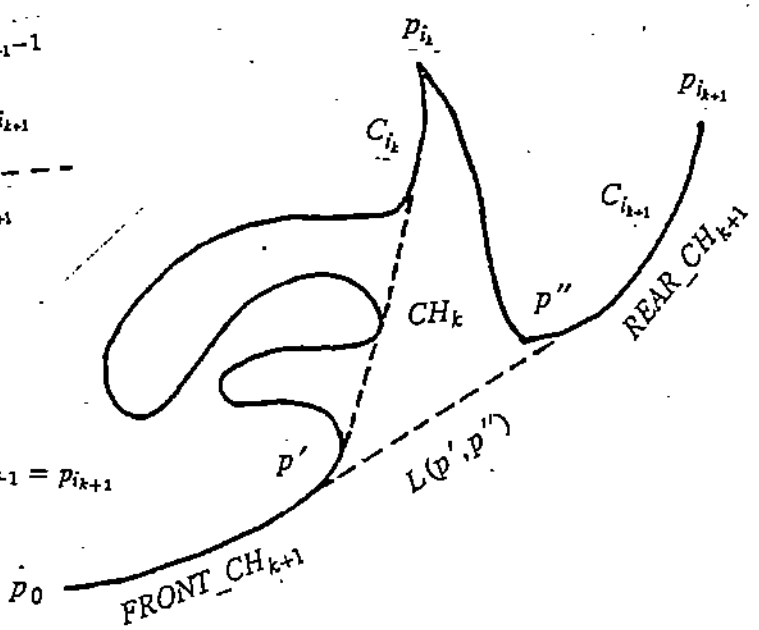

Figure 4: The (k+1)-th *current hull* CH_{k+1}

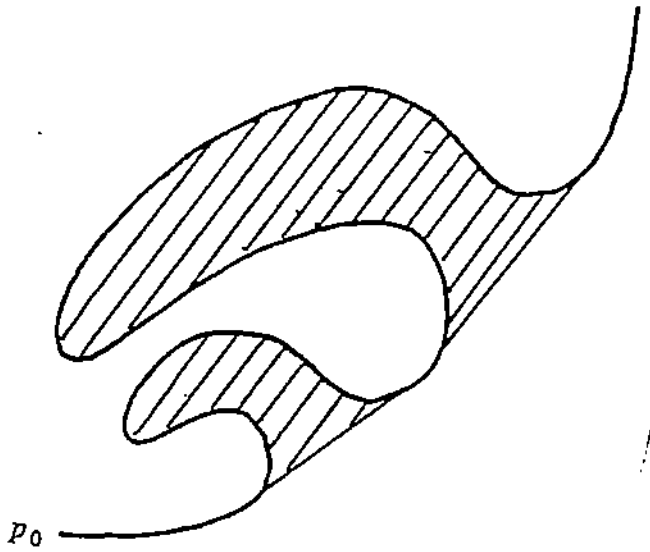

Figure 5: Simple Pockets

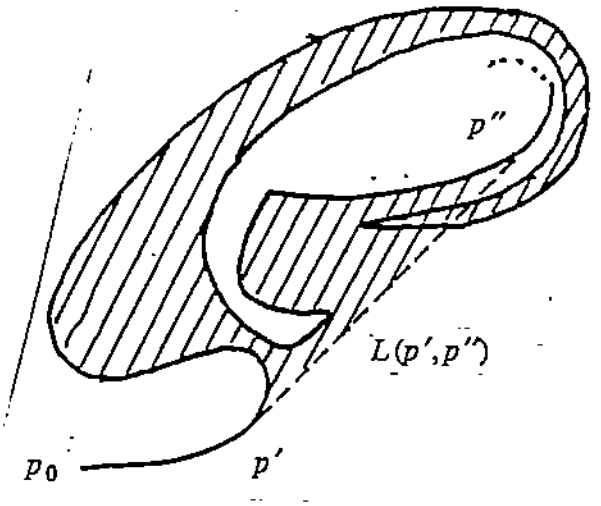

Figure 6: Self–intersecting Pocket

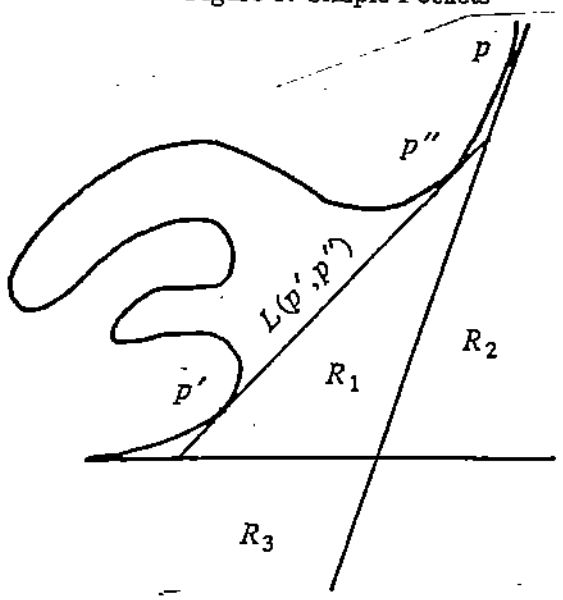

Figure 7: The Regions R_1, R_2 and R_3

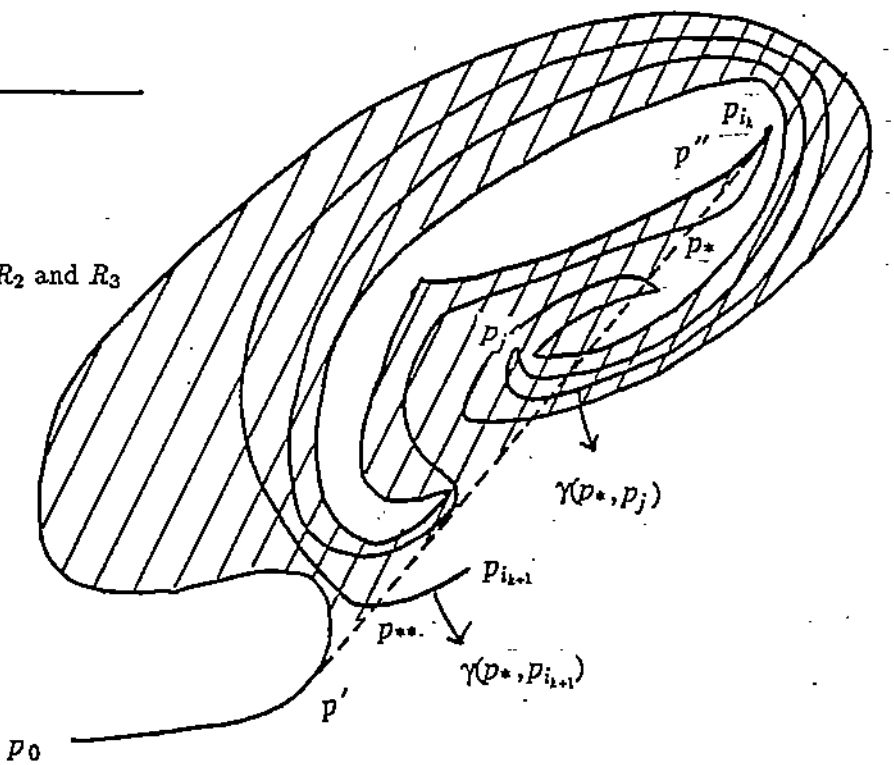

Figure 8: $\gamma(p_*, p_j)$ in the interior of a pocket

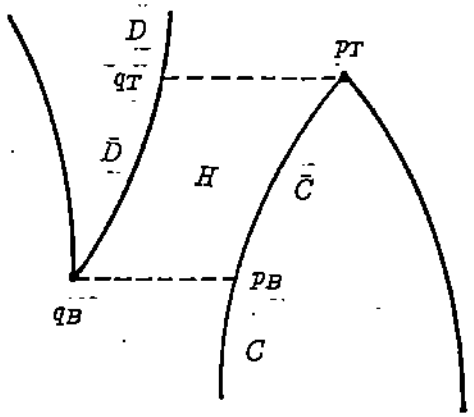

Figure 9: Horizontal Vertex Visibility Cell H

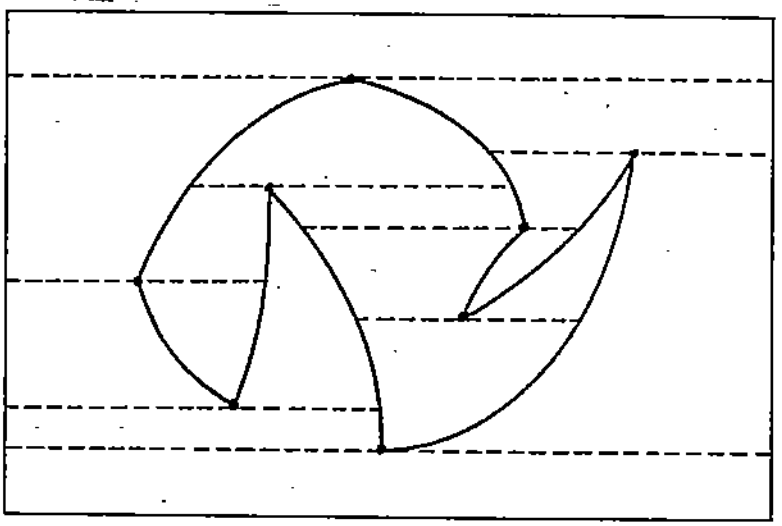

Figure 10: Horizontal Vertex Visibility Partition

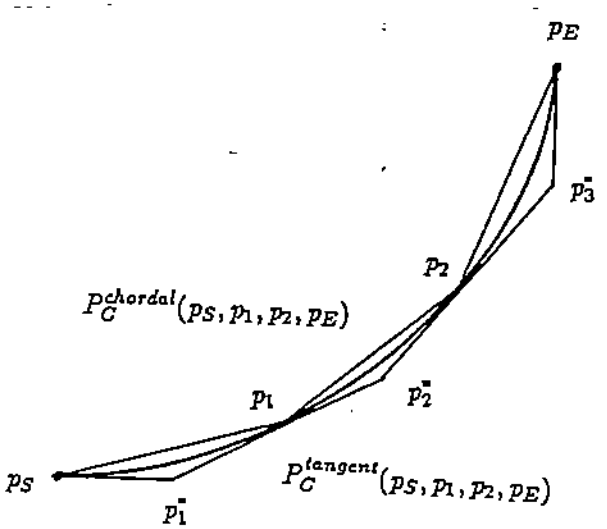

Figure 11: Chordal and Tangential Polygonal Arcs