1987

Decompositions of Objects Bounded by Algebraic Curves

Chanderjit Bajaj

Myung-Soo Kim

Report Number:
87-677

DECOMPOSITION OF OBJECTS BOUNDED BY ALGEBRAIC CURVES

Chanderjit Bajaj
Myung-Soo Kim

CSD-TR-677
April 1987
Revised June 1988

Decompositions of Objects Bounded by Algebraic Curves*

Chanderjit Bajaj and Myung-Soo Kim

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

Abstract

We present an algorithm to decompose the edges of planar curved object so that the *carrier* polygon of decomposed boundary is a *simple* polygon. We also present an algorithm to compute a simple *characteristic* carrier polygon. By refining this decomposition further and using the chords and wedges of decomposed edges, we obtain an *inner* polygon (resp. an *outer* polygon) which is a simple polygon totally contained in (resp. totally containing) the object. We also consider various applications of these polygons to object decompositions and collision-avoidance planar robot motion planning problems.

1 Introduction

Most algorithms in computational geometry have been designed for discrete objects like points, lines, polygons, and polyhedra, see [14,16]. Using the *splinegon* as a generalization of the polygon, Souvaine [20] presents methods for extending polygonal algorithms to algorithms for splinegons. A *splinegon* is a polygon whose edges have been replaced by convex, concave or linear curve segments, and the *carrier* polygon of a splinegon is the polygon connecting adjacent vertices of a splinegon. Dobkin, Souvaine, and Van Wyk [11] show that the $O(n \log\log n)$ time algorithm for the horizontal-vertex-visibility partition of a simple polygon [21] 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 curved object, 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 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 the same 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 object.

*Research supported in part by NSF grant DCI-85 21356 and a David Ross Fellowship.

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 to 0 or 2π, however, it is small in practice. K (henceforth called the *characteristic* number) in some sense represents the shape degeneracy of the object. In the construction of the characteristic, inner and outer polygons, we assume the object has no vertex with its inner angle being 0 or 2π. Using these polygons, we can compute (1) a convex decomposition of the object as a difference of unions of disjoint convex objects, (2) a decomposition of the object as a union of disjoint certain primitive objects, and (3) various collision-free paths of a point moving among planar curved obstacles.

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 object 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, [2,23]. In §3 we present an algorithm to compute the simple *carrier*, *characteristic*, *inner* and *outer* polygons. In §4 and §5 we consider various applications of these polygons to object decompositions and collision-free planar robot motion planning problems respectively. Finally, in §6 we conclude this paper.

2 Preliminaries

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

2.1 Algebraic Boundary Representation

A planar object 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, [1,17]. 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, 7,9,10,13]. 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 whether a given point lies within a general algebraic curve segment. For this last problem the method of sorting along the curve [13], also provides an efficient solution for low degree algebraic curves.

We consider the monotone segmentation of a planar curved object and other geometric operations on monotone curve segments. Our model of computation is the arithmetic model where arithmetic operations have unit time cost, see [3].

2.2.1 Monotone Segmentation

We first define monotone 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 does not include any extreme point along the x or y directions.*

The monotone segmentation requires adding singular points, inflection points and extreme points on the curve as extra vertices. First we take care of singular points on curved edges. Singularities are determined for each curved edge and are computed by using Lemma 3.1.1: I (i) and II (i). The boundary of the object is next modified such that nonsingular edges are either *convex*, *concave* or *linear* segments. Such conditions are easily met by adding extra vertices to inflection points of curved edges. Inflection points of curves can be obtained and the edges are marked *convex*, *concave* or *linear* respectively by using Lemma 3.1.1: I (iv), (v), (vi) and II (iv), (v), (vi). We may also assume edges are further segmented so that extreme points along x or y directions added as vertices. These extreme points are computed by using Lemma 3.1.1: I (ii), (iii) and II (ii), (iii).

Lemma 2.1 **(I)** *Let $C : (a,b) \rightarrow R^2$ be a curve parametrized by $t \in (a,b)$ and $p = C(t) = (c_1(t),\ c_2(t))$ be a point on this curve. Then*
(i) p is a (non-self-intersecting) singular point $\Longleftrightarrow c_1'(t) = c_2'(t) = 0$,
(ii) p is a non-singular x-extreme point $\Longleftrightarrow c_2'(t) = 0$ *and* $c_1'(t) \neq 0$,
(iii) p is a non-singular y-extreme point $\Longleftrightarrow c_1'(t) = 0$ *and* $c_2'(t) \neq 0$, *and*
(iv) p is an inflection point of the curve C $\Longleftrightarrow c_1'(t) \cdot c_2''(t) - c_2'(t) \cdot c_1''(t) = 0$.
If C has no inflection point, then
(v) C is convex $\Longleftrightarrow c_1'(t) \cdot c_2''(t) - c_2'(t) \cdot c_1''(t) > 0$, *and*
(vi) C is concave $\Longleftrightarrow c_1'(t) \cdot c_2''(t) - c_2'(t) \cdot c_1''(t) < 0$.

(II) *Let C be a curve implicitly defined by $f(x,y) = 0$ and $p = (x,y)$ be a point on the curve C. Then*
(i) p is a singular point $\Longleftrightarrow f = f_x = f_y = 0$,
(ii) p is a non-singular x-extreme point $\Longleftrightarrow f = f_y = 0$ *and* $f_x \neq 0$,
(iii) p is a y-extreme point $\Longleftrightarrow f = f_x = 0$ *and* $f_y \neq 0$, *and*
(iv) p is an inflection point $\Longleftrightarrow f = f_{xx} \cdot f_y^2 - 2f_{xy} \cdot f_x f_y + f_{yy} \cdot f_x^2 = 0$.
If C has no inflection point, then
(v) C is convex $\Longleftrightarrow f_{xx} \cdot f_y^2 - 2f_{xy} \cdot f_x f_y + f_{yy} \cdot f_x^2 > 0$, *and*
(vi) C is concave $\Longleftrightarrow f_{xx} \cdot f_y^2 - 2f_{xy} \cdot f_x f_y + f_{yy} \cdot f_x^2 < 0$.

Proof: Most of these results are classical, see [23]. □

We analyze the time complexity of the monotone segmentation.

Lemma 2.2 (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.*

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, [18].
(2) We can eliminate one variable from two polynomial equations using the Sylvester resultant in $O(d^4 \log^3 d)$ time, [9], 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. □

Lemma 2.3 *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.*

Proof: The extra points added to C can be computed by solving polynomial equations in a single variable t (resp. systems of two simultaneous polynomial equations in two variables x and y). Finally, the solution points on the implicit curve segment C can be found by tracing the curve segment in $T(d)$ time. □

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 point p at which C has a tangent line L_p parallel to L, and
3. The tangent line L_p of C from a point q.

Line–Curve Segments Intersection

The intersection points $p \in C \cap L$ can be computed as follows.

Lemma 2.4 **(I)** *If C is a parametric curve segment given by $C(s) = (x(s), y(s))$ with $a \le s \le b$ and L is a line segment connecting two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, then C intersects with L at a point $p = C(s) = t \cdot p_1 + (1-t) \cdot p_2$* if and only if *$s$ and t satisfy*

$$\begin{cases} a \le s \le b, \text{ and } 0 \le t \le 1 & (1) \\ x(s) = t \cdot x_1 + (1-t) \cdot x_2 & (2) \\ y(s) = t \cdot y_1 + (1-t) \cdot y_2 & (3) \end{cases}$$

(II) *If C is an implicit algebraic curve segment given by $f(x, y) = 0$, then C intersects with L at a point $p = t \cdot p_1 + (1-t) \cdot p_2$* if and only if *$t$ satisfies*

$$\begin{cases} p \in C, \text{ and } 0 \le t \le 1 & (1) \\ f(t \cdot x_1 + (1-t) \cdot x_2, t \cdot y_1 + (1-t) \cdot y_2) = 0 & (2) \end{cases}$$

Proof: Straightforward. □

Lemma 2.5 *For a parametric (resp. implicit) algebraic curve segment C of degree d, the intersection $C \cap L$ can be computed in $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time, where $T(d)$ is the time required for a curve tracing along C.*

Proof: (I) The elimination of t can be done in $O(1)$ time resulting in a single polynomial of degree d in a single variable s. This polynomial can be solved in $O(d^3 \log d)$ time using root isolation, [18]. There are at most d solutions for s with $a \leq s \leq b$ and the corresponding t to each s can be solved in $O(1)$ time.
(II) When we expand the equation (2) in an increasing order of t, it gives a polynomial of degree d in a single variable t. The expansion can be done in $O(d^2)$ time and the polynomial can be solved in $O(d^3 \log d)$ time using root isolation. Finally, we need to trace along the curve segment C to check whether these solutions are on the curve segment C in $T(d)$ time. □

Tangent Line of a Curve Segment Parallel to a Line

The point p at which C has a tangent line L_p parallel to L can be computed as follows.

Lemma 2.6 **(I)** *If C is given by a parametric curve $C(t) = (x(t), y(t))$ with $a \leq t \leq b$ and (α, β) is a normal direction of L, then $p = (x(t), y(t))$ is given by*

$$\begin{cases} a \leq t \leq b & (1) \\ \alpha \cdot x'(t) + \beta \cdot y'(t) = 0 & (2) \end{cases}$$

(II) *If C is given by an implicit curve $f(x, y) = 0$, then the point $p = (x, y)$ is given by*

$$\begin{cases} f(x, y) = 0, \text{ and } p = (x, y) \in C & (1) \\ \beta \cdot f_x - \alpha \cdot f_y = 0 & (2) \end{cases}$$

Proof: Straightforward. □

Lemma 2.7 *For a parametric (resp. implicit) curve segment C, the point p at which C has a tangent line L_p parallel to L can be computed in $O(d^3 \log d)$ (resp. $O(d^6 \log d + T(d))$) time, where $T(d)$ is the time required for a curve tracing along C.*

Proof: Similar to Lemma 2.3. □

Tangent Line of a Curve Segment from a Point

The tangent point p of L at C from a point q can be computed as follows.

Lemma 2.8 **(I)** *If C is given by a parametric curve $C(t) = (x(t), y(t))$ with $a \leq t \leq b$, then $p = (x(t), y(t))$ is given by*

$$\begin{cases} a \leq t \leq b & (1) \\ (x(t) - \alpha) \cdot y'(t) - (y(t) - \beta) \cdot x'(t) = 0 & (2) \end{cases}$$

(II) *If C is given by an implicit curve $f(x, y) = 0$, then the point $p = (x, y)$ is given by*

$$\begin{cases} f(x, y) = 0, \text{ and } p = (x, y) \in C & (1) \\ (x - \alpha) \cdot f_x + (y - \beta) \cdot f_y = 0 & (2) \end{cases}$$

Proof: Straightforward. □

Lemma 2.9 *For a parametric (resp. implicit) curve segment C, the tangent point p of L at C from a point q can be computed in $O(d^3 \log d)$ (resp. $O(d^6 \log d + T(d))$) time, where $T(d)$ is the time required for a curve tracing along C.*

Proof: Similar to Lemma 2.3. □

Now, initially assume there are $O(m)$ algebraic curve segments of maximum degree d on the object 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 Decomposition of Monotone Boundary Edges

In this section, we consider how to construct a simple carrier polygon of a planar curved object with at most $O(n^2)$ edges, which is optimal since this number is shown to be $\Omega(n^2)$ in [11]. Further, we consider how to construct a simple *characteristic* carrier polygon, an *inner* polygon and an *outer* polygon of the planar curved object. We assume the object boundary has been decomposed into $O(n)$ *monotone* edges in a preprocessing step as discussed in §2. In the following, we denote the x and y-coordinates of a point $p_\#$ by $x_\#$ and $y_\#$ respectively.

3.1 Simple Carrier Polygon

Consider the horizontal vertex visibility partition of both the interior and exterior of a planar curved object, [11,21], where the exterior is partitioned within a box enclosing the object, see Figure 1. 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 2. 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 object 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 further the carrier polygon determined by these extra vertices 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 q_i is strictly to the left of p_i $(1 \leq i \leq k_{\bar{C}} = k_{\bar{D}})$ and the corresponding carrier polygonal arcs $P_{\bar{C}}$ and $P_{\bar{D}}$ do not intersect except at the end points. Further it follows that the carrier polygon $\cup P_{\bar{C}}$ is simple. Since there are

$O(n)$ such horizontal lines and boundary edges, $k_{\bar{C}} = O(n)$ and the carrier polygon has at most $O(n^2)$ 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 have to construct $P_{\bar{D}}$ recursively. Assume we have constructed $P_{\bar{D}}$ by adding a minimal 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 $P_{\bar{C}}$ not intersecting with $P_{\bar{D}}$ except at the end points, see Figure 3. 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 3.1 *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 $O(n \log\log n + n \cdot d^3 \log d)$ (resp. $O(n \log\log n + n \cdot (d^3 \log d + T(d)))$) time, [20]. 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. □

3.2 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 object or in the exterior of the object, where $S(E)$ is the convex region bounded by the edge E and its chord, see Figure 4. If E is a convex (resp. a concave) edge, then $S(E)$ is totally contained in the interior (resp. in the exterior) of the object 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, see Figure 5. Otherwise, $S(\bar{C})$ is an additive convex region. Now, we prove the decomposition of C terminates within a finite number of steps.

Theorem 3.2 *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 C is convex (resp. concave) and 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 by x_E, $x_i \to x$ for some $x \leq x_E$. Let $p_L \in C$ be such that $x_L = x$, then $p_i \to p_L$. 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 $i \geq N$ and $q_i \to p_L$. We can easily show that D is a concave (resp. convex) edge and p_L is a common vertex of C and D, and further the inner angle at $p_L < \pi/2$ (resp. $> 3\pi/2$). Since $L(p_{i-1}, p_i)$ is tangent to D at q_i with $p_i \to p_L$ and $q_i \to p_L$, C and D have the same tangent line at p_L. Hence, the inner angle of the object at p_L is 0 (resp. 2π). It is impossible since we assume no such vertex on the object 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 object, 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, see Figure 6. Thus, using at most $2K$ edges we can construct a simple characteristic carrier polygon.

Theorem 3.3 *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 to curve segments. □

3.3 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 7.

If we further decompose the object boundary with the first characteristic polygon 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 object, 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^X be the first characteristic polygonal arc of C, then C is to the right of P_C^X and D is to the left of P_C^X. We may assume $p_B \neq q_B$ or $p_T \neq q_T$. If D is a convex edge, then P_C^X 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 the line segments $L(p_B, p_1), L(p_1, p_2), \ldots, L(p_{k_C}, p_T)$ on P_C^X, where $p_B, p_1, p_2, \ldots, p_{k_C}, p_T$ are the vertices of P_C^X in the order they appear along C. Note that $q_1 = q_B$ if $p_B = q_B$, and $k_C = 0$ if $D \cap P_C^X = \emptyset$. We add an extra vertex p'_i to each subedge of C between p_{i-1} and p_i ($i = 1, \ldots, k_C + 1$), where $p_0 = p_B$ and $p_{k_C+1} = p_T$, see Figure 8.

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^X except at 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 $L(p_{k_C}, p_T)$. 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. P_D^{inner} is to the left of P_C^X and strictly to the left of P_C^X at the horizontal lines $y = y_1, y = y_2, \ldots, y = y_{k_C}$. Thus, P_C^{inner} and P_D^{inner} do not intersect except at the end points.

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. Let L_1 (resp. L_2) be the tangent line from p_B (resp. q_B) to D (resp. C), and p^* be the intersection point of L_1 and L_2, see Figure 9. Since p^* lies on L_1 and L_2, p^* is to the right of D and to the left of C. Thus, p^* is in the horizontal visibility cell H. Let L'_1 (resp. L'_2) be the tangent line from p^* to D (resp. C), and L be a line containing the point p^* and having its slope strictly less (resp. greater) than the slopes of L_1 and L'_1 (resp. L_2 and L'_2).

Further, 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), see Figure 10. Since P_C^{inner} (resp. P_D^{inner}) is strictly to the right (resp. to the left) of L, $P_C^{inner} \cap P_D^{inner} = \emptyset$.

Since P_C^{inner} and P_D^{inner} do not intersect except at the end points, the *inner* polygon $P^{inner} = \cup P_C^{inner}$ is a simple polygon.

Theorem 3.4 *Assume all the monotone edges are parametric (resp. implicit). Both simple 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. □

3.4 The Case of Multiple Disjoint Objects

So far we have considered only a single object in the plane. However, the above method can also be applied to multiple disjoint objects in the plane once the horizontal vertex visibility partition is given for the interiors and the exterior of the objects.

When there is more than one object in the plane, we cannot directly use the algorithm of [21] to compute the horizontal vertex visibility partition of the exterior of objects in $O(n \log\log n)$ time, where n is the total number of edges of the objects. We can show this by reducing the problem of sorting $O(n)$ real numbers to the problem of computing the horizontal vertex visibility partition of the exterior of $O(n)$ small triangles within $O(n)$ time, see Figure 11. We assume all the numbers are

distinct and $\epsilon \geq 0$ is smaller than the minimum distance between any two distinct numbers. Then for each real number a_i we construct a triangle with vertices (i, a_i), $(i-\epsilon, a_i-\epsilon)$ and $(i+\epsilon, a_i-\epsilon)$. Assume that we can construct the horizontal vertex visibility partition of the exterior of these $O(n)$ triangles inside an enclosing box in $O(n \log\log n)$ time. Then there is exactly one vertex of each height a_i and each vertex (i, a_i) is visible from the right side of the enclosing box. The sequence of a_i's in the order the vertices (i, a_i)'s are horizontally visible while tracing along the right side of the enclosing box is the same as the sorted sequence of the given $O(n)$ real numbers. Our above assumption then would allow us to sort $O(n)$ real numbers within $O(n \log\log n)$ time, which is impossible. Thus $\Omega(n \log n)$ is the lower bound for this problem in general.

We can easily construct the horizontal vertex visibility partition of the exterior of multiple disjoint objects with totally $O(n)$ edges in $O(n \log n + n \cdot d^{O(1)})$ time using a plane sweep algorithm. Though this is optimal in the worst case as in the above example, it may be possible to improve the time complexity when there are only $O(k)$ disjoint objects with $O(n)$ total number of edges. If it is possible to connect these $O(k)$ disjoint objects into a single connected object using $O(n)$ extra edges within $O((n + k \log k) \cdot d^{O(1)})$ time, we can use the $O(n \log\log n)$ time algorithm to compute the horizontal vertex visibility partition of the exterior of this single object. The visibility partition of the exterior of the objects could be constructed by deleting the extra edges added and merging some adjacent visibility cells together. The total time complexity could then be $O(n \log\log n + (n + k \log k) \cdot d^{O(1)})$. We thus raise the following problem.

Problem 3.1 *Can $O(k)$ disjoint objects with $O(n)$ total number of edges be connected into a single connected object in $O((n + k \log k) \cdot d^{O(1)})$ time using at most $O(n)$ extra edges.*

4 Object Decompositions

We consider various applications of the polygons $P^{\times}$ and P^{inner} constructed in §3 to the object decomposition problems. Dobkin, Souvaine, and Van Wyk [11] show 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 decompositions, described below, involve $O(K)$ regions, where K is often linear in practice. Further, the decomposition structures we compute in terms of unions and differences are somewhat simpler than that of [11]. Also when the simple characteristic polygon has a small number of edges, for example K is almost linear, our decomposition method proves to be more useful.

4.1 Convex Decomposition

We can decompose the simple characteristic polygon $P^{\times}$ into unions of disjoint convex polygons $\cup_i P_i$, (see [14] 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 the P_i's and U_j's are disjoint, and the interiors of the U_j's and V_k's are disjoint, however, the interiors of the 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 $P^{\times}$ is highly parallel and would be useful in a parallel implementation of the object decomposition algorithm.

4.2 Disjoint Decomposition

The main purpose of object decomposition is to simplify a problem for complex object 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 a planar curved object. 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, see [7]) which commutes with set union, but not with set difference, we may consider decomposing an object into a disjoint union of certain primitive objects. 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 $A \ominus B = \{a - b \mid a \in A \text{ and } b \in B\}$. When $A = \cup_i S_i$ and $B = \cup_{i'} S'_{i'}$, we have $A \oplus B = \cup(S_i \oplus S'_{i'})$ and $A \ominus B = \cup(S_i \ominus S'_{i'})$.

One way to decompose a planar curved object into a disjoint union of certain primitive objects is as follows. First, we decompose an inner polygon P^{inner} into a union of disjoint convex polygons $\cup_i P_i$. The difference between the object and the inner polygon P^{inner} is the union $(\cup_j U_j) \cup (\cup_k V_k)$, where each U_j is an additive convex region bounded by a convex edge and its chord and each V_k is a region bounded by a concave edge and its wedge, see Figure 12. We can thus represent the original object as a disjoint union $(\cup_i P_i) \cup (\cup_j U_j) \cup (\cup_k V_k)$.

The horizontal vertex visibility partition of the interior of an object also represents the original object as a disjoint union $\cup_i H_i$, where H_i is a horizontal visibility cell with linear top and bottom edges and monotone right and left side edges.

5 Planar Gross Motion Planning

In this section, we consider the applications of various decomposition methods to collision-free planar robot motion planning problems for curved objects with fixed orientation. Since this problem can be transformed into an equivalent point motion planning problem among algebraic curved *C-space obstacles* [7], we consider only point motions in the following.

5.1 Voronoi Diagram for the Outer Polygons

As discussed in §3.4, we can construct disjoint outer polygons of multiple disjoint planar obstacles within $O(n \log n + K \cdot d^{O(1)})$ time, where K is the characteristic number of the obstacles. The free space of these outer polygons is a topological retract of the free space of original obstacles. Since each outer polygonal edge and its corresponding obstacle boundary edge are horizontally visible from each other, a point p which is inside an outer polygon, but not in any obstacle, can move along a horizontal line segment into an outer polygonal edge without colliding with any obstacle. Thus, we may assume the start point p_S and the goal point p_G are in the free space of outer polygons. Now, we can consider the topological retraction of the free space of outer polygons onto the Voronoi diagram of outer polygons using the method of [15] and construct a collision-free path γ from the starting point to the goal point. Since the outer polygons of obstacles have total $O(K)$ number of edges, this method is attractive when K is almost linear or reasonably small.

5.2 Voronoi Diagram for the Simple Carrier Polygons

We can construct disjoint simple carrier polygons of disjoint obstacles within $O(n \log n + k \cdot d^{O(1)})$ time, where k is the total number of edges in the carrier polygons, which is at most $O(n^2)$. We first consider how to move a point p which is inside a carrier polygon, but not in any obstacle,

without colliding with obstacles. Let H be the horizontal visibility cell containing the point p and in the exterior of obstacles. Assume C and D are the right and left sides of H. Since p is inside a carrier polygon, either C or D is concave. We may assume C is concave and p is contained in the convex region $S(C)$. We move p until it hits a carrier polygonal edge or the left side D. If it hits D first, then we can move p along D until it hits a carrier polygonal edge, see Figure 13. Thus, we may assume the start point p_S and the goal point p_G are in the free space of the carrier polygons. Then a Voronoi diagram of these disjoint simple carrier polygons and a path avoiding collisions with these carrier polygons can be constructed within $O(k \log k)$ time, [15]. Though this path may intersect with some obstacles, this collision could occur only with convex edges of the obstacle, see Figure 14. Since we assume each convex edge is monotone, it is obvious in which direction we have to move along the colliding convex edge. Further the construction of a simple carrier polygon is more efficient than the construction of an outer polygon in the worst case. As long as we are allowed to follow the boundary of obstacle in motion planning, this method would be more efficient than the Voronoi diagram based method of §5.1 when the characteristic number K is large.

5.3 Horizontal Visibility Cells Graph

Using the horizontal vertex visibility partition of the outside of obstacles we can decompose the free space into $O(n)$ simple cells. Then we can represent the connectivity of these cells in terms of a graph and check the connectivity of this graph between the two cells containing the starting point p_S and the goal point p_G by doing a search on this connectivity graph, see Figure 15. There is a collision-free path between the starting and goal points if and only if there is a connected edge path between the starting cell and goal cell in this graph. Once we have a sequence of connected cells on this graph, we can construct a collision-free motion path in various ways, see Figure 16.

6 Conclusion

We presented an $O(n \log\log n + K \cdot d^{O(1)})$ time algorithm to compute various decompositions of a planar algebraic curved object boundary, where K is a number which captures the shape degeneracy of the object. The exponent of the $O(d^{O(1)})$ is $O(d^3 \log d)$ (resp. $O(d^6 \log d + T(d))$) for parametrically (resp. implicitly) defined algebraic curves. Also in practice essentially all planar curved objects considered are those bounded by curves of maximal degree $d \leq 4$ of which all $d = 2$ and most $d = 3$ curves are parametrizable. Our algorithms rely on simple data structures and are implementable. Using these decompositions, we presented algorithms for object decompositions and collision-free planar robot motion planning problems in an algebraic curved environment.

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., (1987a), "Compliant Motion Planning with Geometric Models," *Proc. of the 3rd ACM Symposium on Computational Geometry*, Waterloo, Canada, pp. 171–180, Modified version with title "Generation of Configuration Space Obstacles: The Case of Moving Algebraic Surfaces," to appear in *The International Journal of Robotics Research.*

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

[9] 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.

[10] Canny, J., (1988), "The Complexity of Robot Motion Planning," *ACM Doctoral Dissertation Series,* MIT Press, Cambridge, Mass.

[11] 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.

[12] Hopcroft., J., (1986), "The Impact of Robotics on Computer Science," *Communications of the ACM*, Vol. 29, No. 6, pp. 486–498.

[13] Johnstone, J., (1987), "The Sorting of Points along an Algebraic Curve," Ph.D Thesis, Dept. of CS, Cornell University.

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

[15] Ó'Dúnlaing, C., Sharir, M., and Yap, C.K., (1983), "Retraction: A New Approach to Motion-Planning," *Proc. of the 15th Symp. on the Theory of Computing*, Boston, pp. 207–220.

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

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

[18] Schwartz, J.T. and Sharir, M., (1983), "On the Piano Movers Problem: II, General Techniques for Computing Topological Properties of Real Algebraic Manifolds," *Advances in Applied Mathematics* Vol. 4, pp. 298–351.

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

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

[21] 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.

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

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

[24] Yap, C.K., (1987), "An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments," *Discrete and Computational Geometry*, Vol. 2, No. 4, 1987, pp. 365–393.

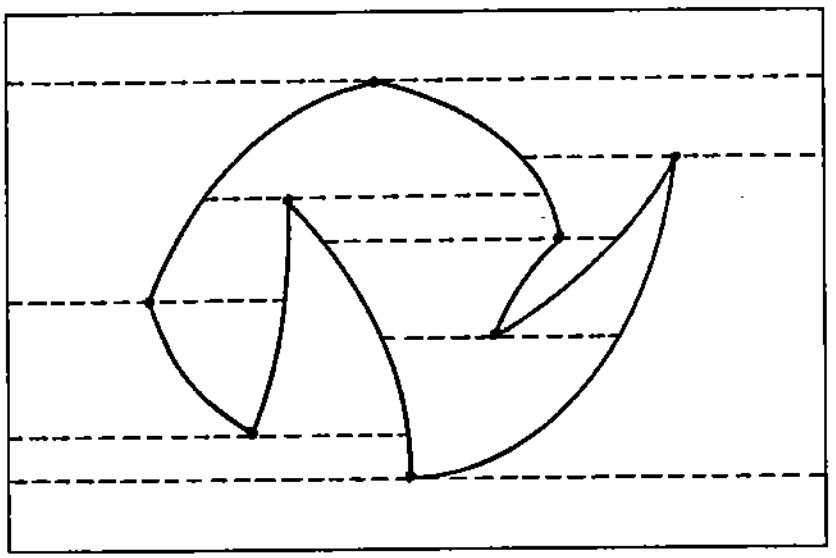

Figure 1: Horizontal Vertex Visibility Partition

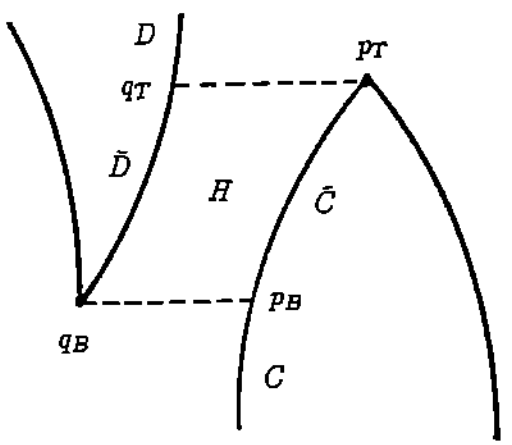

Figure 2: Horizontal Vertex Visibility Cell H

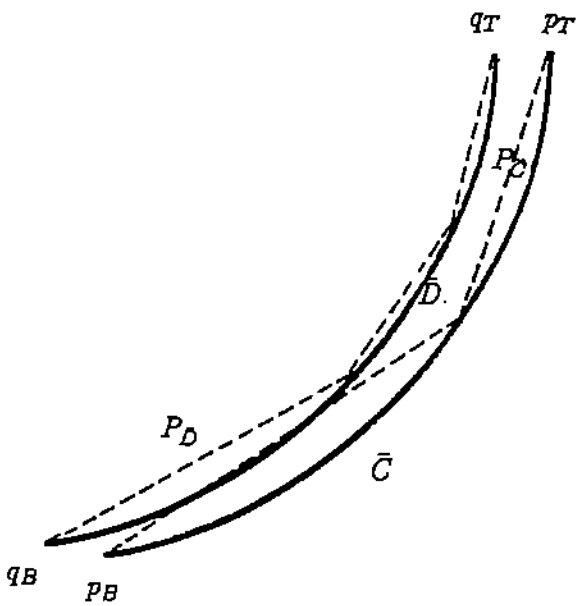

Figure 3: Construction of $P_{\bar{C}}$ with $k_{\bar{C}} \leq k_D$

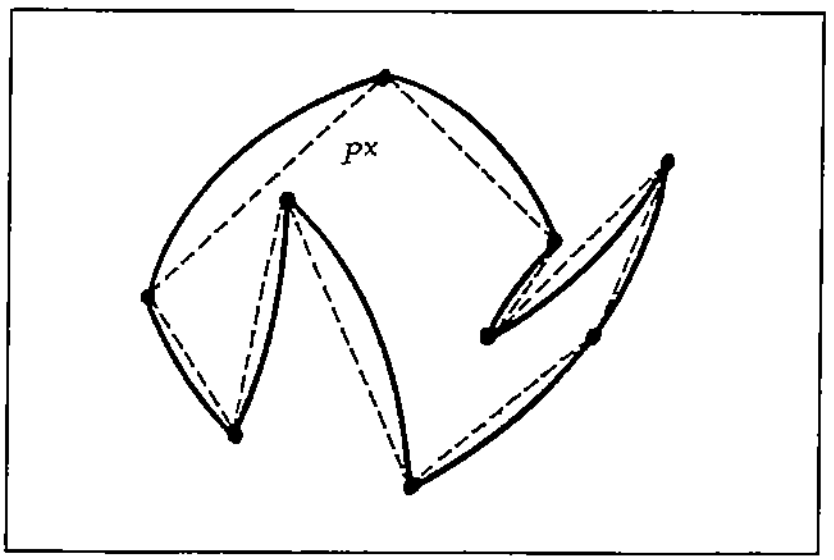

Figure 4: Simple Characteristic Carrier Polygon

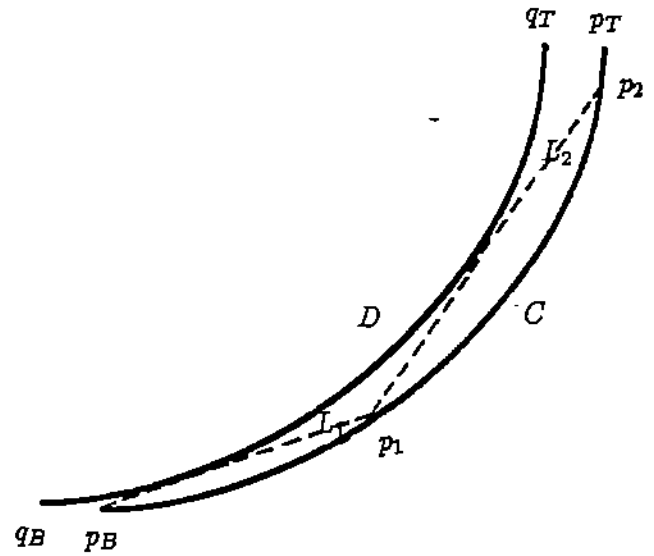

Figure 5: Construction of P_C^X

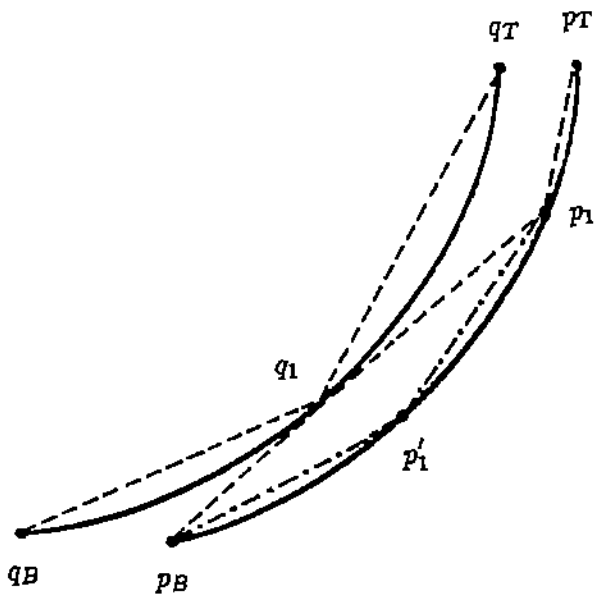

Figure 6: Degenerate First Characteristic Carrier Polygon

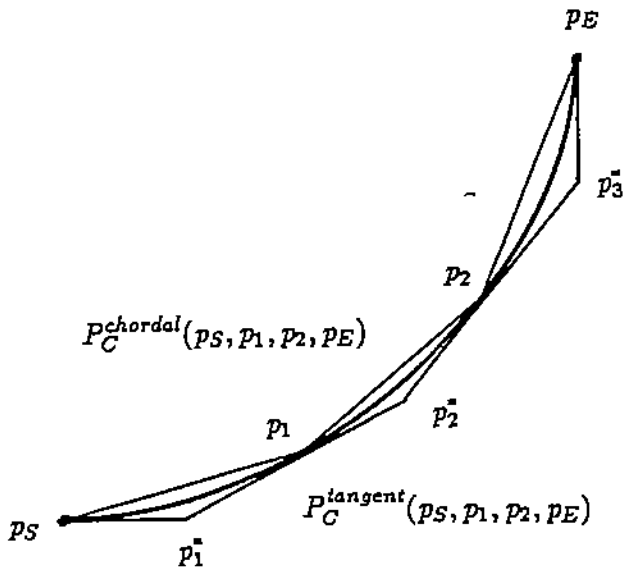

Figure 7: Chordal and Tangential Polygonal Arcs

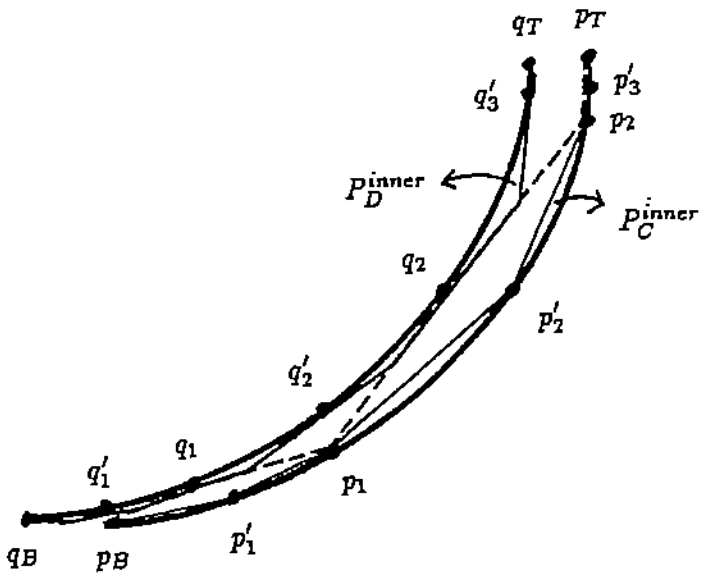

Figure 8: Construction of P_C^{inner} and P_D^{inner}

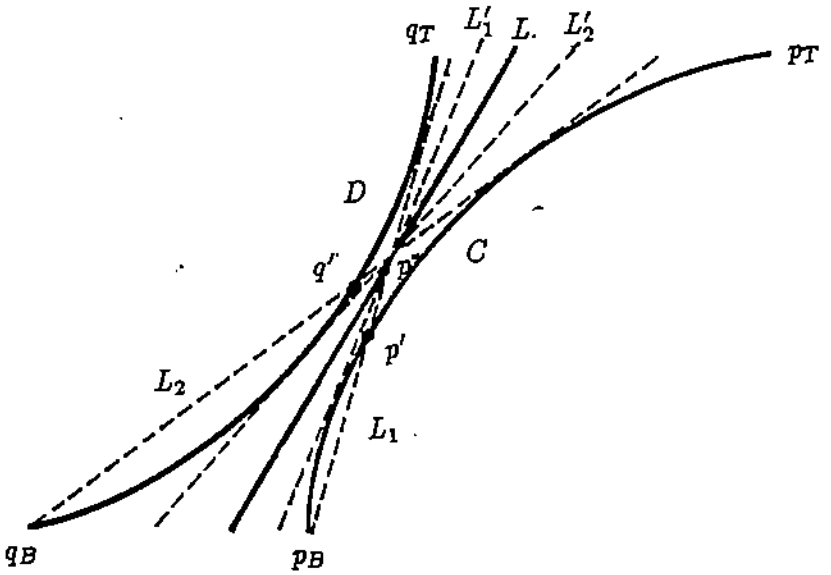

Figure 9: Line L Seperating C and D

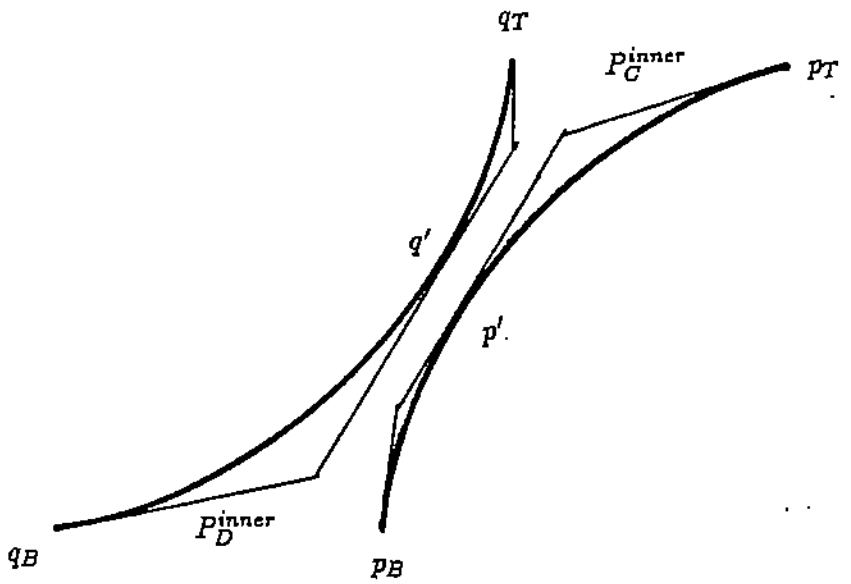

Figure 10: P_C^{inner} and P_D^{inner} for Concave C and D

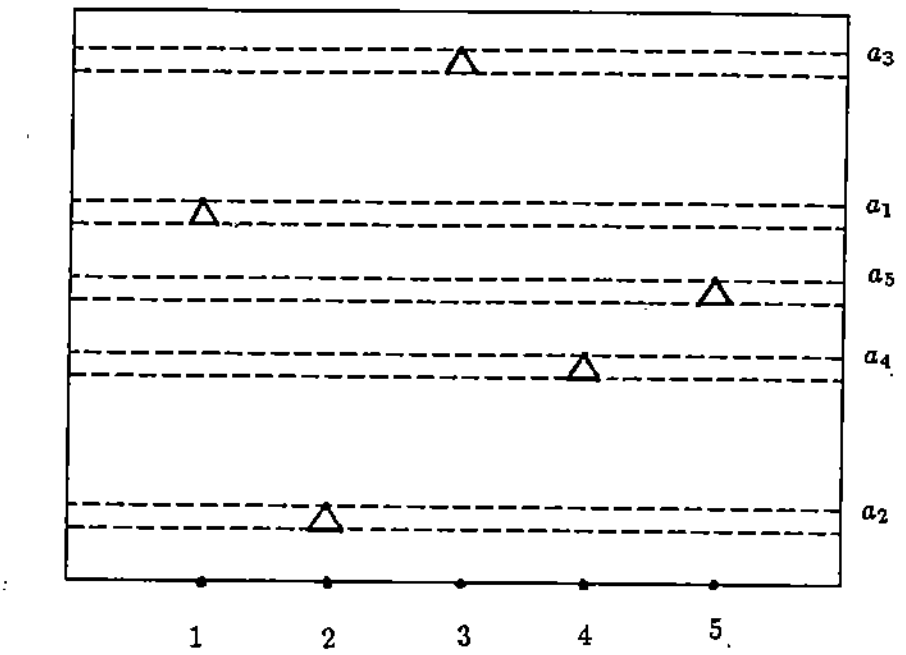

Figure 11: Reducing Sorting to Horizontal Vertex Visibility Partition

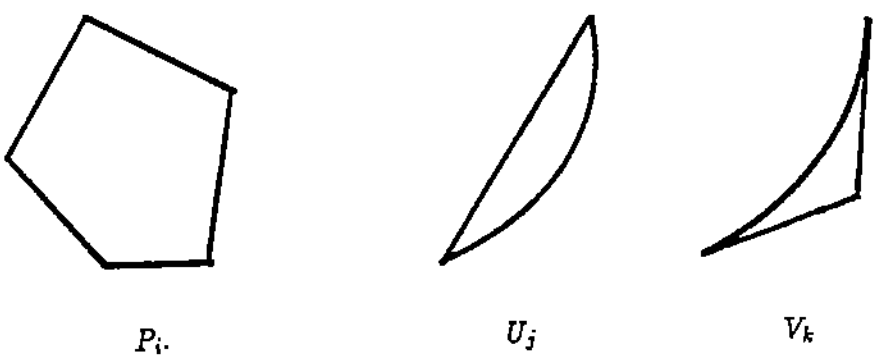

Figure 12: Primitive Objects

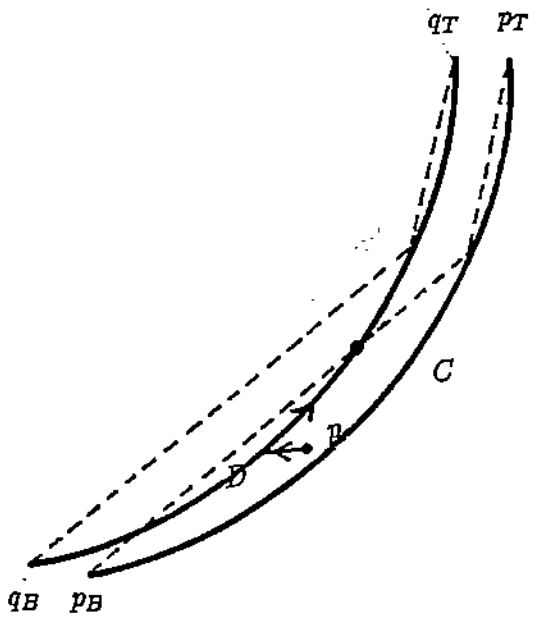

Figure 13: Moving p Along D

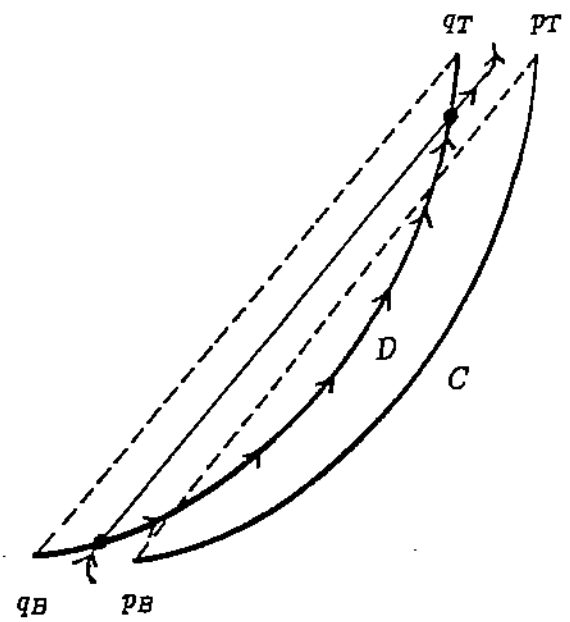

Figure 14: Moving p Along Colliding Convex Edge

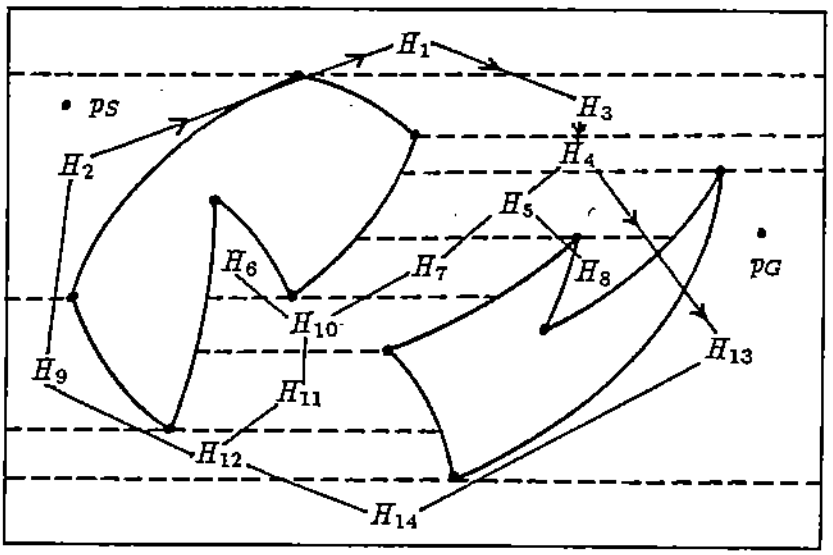

Figure 15: Horizontal Visibility Cells Graph

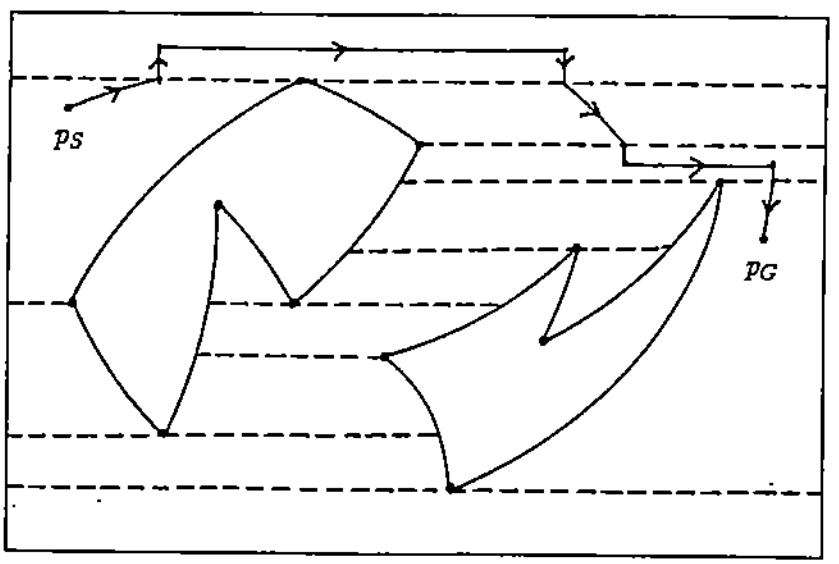

Figure 16: Path Construction