1988

On the Sorting of Points along an Algebraic Curve

John K. Johnstone

Chanderjit Bajaj

Report Number:
88-824

ON THE SORTING OF POINTS ALONG AN ALGEBRAIC CURVE

John K. Johnstone*
and
Chanderjit Bajaj†

Computer Sciences Department
Purdue University
Technical Report CSD-TR-824
CAPO Report CER-88-37
November, 1988

* Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218. The work of this author was supported in part by a Natural Sciences and Engineering Research Council of Canada 1967 Graduate Fellowship and an Imperial Esso Graduate Fellowship while the author was a graduate student in the Department of Computer Science, Cornell University, Ithaca, NY, 14853.

† Department of Computer Science, Purdue University, West Lafayette, IN 47907. The work of this author was supported in part by National Science Foundation grant 85-21356 and ARO contract DAAG 29-85-C0018 under Cornell MSI.

ON THE SORTING OF POINTS ALONG AN ALGEBRAIC CURVE

John K. Johnstone*
and
Chanderjit Bajaj†

Computer Sciences Department
Purdue University
Technical Report CSD-TR-824
CAPO Report CER-87-37
November, 1988

Abstract

An operation that is frequently needed during the creation and manipulation of geometric models is the sorting of points along an algebraic curve. Given a segment $\overline{AB}$ of an algebraic curve, a set of points on the curve is sorted from A to B along $\overline{AB}$ by putting them into the order that they would be encountered in traveling continuously from A to B along $\overline{AB}$. A new method for sorting points along an algebraic curve is presented. Key steps in this method are the decomposition of a plane algebraic curve into convex segments and point location in this decomposition. This new method can sort an arbitrary algebraic curve and it is particularly efficient because of its preprocessing, both of which make it superior to conventional methods. The complexity of the new method is analyzed, and execution times of various sorting methods on a number of algebraic curves are presented. The theory developed for sorting can also be used to locate points on an arbitrary segment of an algebraic curve and to decide whether two points lie on the same connected component.

---

* Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218. The work of this author was supported in part by a Natural Sciences and Engineering Research Council of Canada 1967 Graduate Fellowship and an Imperial Esso Graduate Fellowship while the author was a graduate student in the Department of Computer Science, Cornell University, Ithaca, NY, 14853.

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

ON THE SORTING OF POINTS ALONG AN ALGEBRAIC CURVE

JOHN K. JOHNSTONE* and CHANDERJIT L. BAJAJ†

Abstract. An operation that is frequently needed during the creation and manipulation of geometric models is the sorting of points along an algebraic curve. Given a segment $\widehat{AB}$ of an algebraic curve, a set of points on the curve is sorted from A to B along $\widehat{AB}$ by putting them into the order that they would be encountered in travelling continuously from A to B along $\widehat{AB}$. A new method for sorting points along an algebraic curve is presented. Key steps in this method are the decomposition of a plane algebraic curve into convex segments and point location in this decomposition. This new method can sort an arbitrary algebraic curve and it is particularly efficient because of its preprocessing, both of which make it superior to conventional methods. The complexity of the new method is analyzed, and execution times of various sorting methods on a number of algebraic curves are presented. The theory developed for sorting can also be used to locate points on an arbitrary segment of an algebraic curve and to decide whether two points lie on the same connected component.

Key words. Sorting, decomposition, point location, convexity, algebraic curves, geometric modeling, solid modeling.

AMS(MOS) subject classifications. 68U05, 68Q25, 68P10, 14H99.

1 Introduction

The sorting of numbers into increasing order or words into alphabetical order is one of the basic problems of computer science. The purpose of this paper is to establish that the sorting of points along a curve is a basic problem in geometric modeling and computational geometry, and to present

*Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218. The work of this author was supported in part by a Natural Sciences and Engineering Research Council of Canada 1967 Graduate Fellowship and an Imperial Esso Graduate Fellowship while the author was a graduate student in the Department of Computer Science, Cornell University, Ithaca, NY, 14853.

†Department of Computer Science, Purdue University, W. Lafayette, IN 47907. The work of this author was supported in part by National Science Foundation grant 85-21356 and ARO contract DAAG 29-85-C0018 under Cornell MSI.

a universal and efficient method for this sorting. This method relies upon the solution of two problems that are very useful in their own right: convex decomposition of a curve and point location on a segment.

To sort a set of points from A to B along the curve segment $\widehat{AB}$ means to put the points into the order that they would be encountered in travelling continuously from A to B along $\widehat{AB}$ (Figure 1). Points that do not lie on $\widehat{AB}$ are never encountered and are thus ignored. A vector at A is provided to indicate the direction in which the sort is to proceed from A. This vector is especially important when the curve is closed, since there are then two segments between A and B to choose from. All of the points, including A and B, are assumed to be nonsingular, since otherwise their order might be ambiguous.

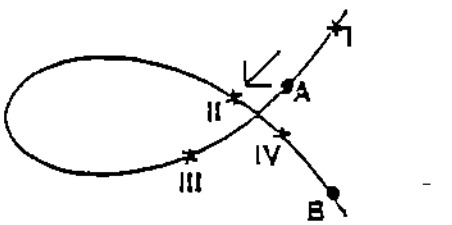

Figure 1: The sorted order from A to B is III, II, IV

Our treatment shall be of irreducible algebraic plane curves (a curve that lies in a plane and is described by an irreducible polynomial[1] $f(x, y) = 0$); in the rest of this paper, all curves are assumed to be of this type and nonlinear. An extension of the methods to algebraic space curves is possible using a suitable projection of the space curve to a plane curve [16].

The next section establishes that sorting is a fundamental operation of geometric modeling. After discussing previous sorting methods in Section 3, we introduce our new sorting method in Section 4. Convex decomposition of a curve and point location on a convex segment are discussed in Sections 5 and 6. Complexity issues and execution times of the various sorting methods are presented in Sections 7 and 8. The relative advantages of the sorting methods are weighed in Section 9 and Section 10 ends with some conclusions.

[1]The coefficient domain of the polynomial can be the integers, rationals, algebraic real numbers, or any other set of numbers that has a finite representation.

2 The importance of sorting

The sorting of points along a curve has many applications in geometric modeling. The following problem is the most natural application.

Restriction
INSTANCE: A set S of points on a curve C and a segment $\widehat{EF}$ of C.
QUESTION: Which points of S lie on $\widehat{EF}$?
SOLUTION: Sort S along $\widehat{EF}$.

Since an edge of a solid model is often defined by a curve and a pair of endpoints, restriction is a very basic problem in geometric modeling. For example, the following edge intersection and bounding box problems are two important problems that can be solved with restriction.

Edge intersection
INSTANCE: Edges E and F on curves C and D, respectively.
QUESTION: What is $E \cap F$?
SOLUTION: Compute $C \cap D$ by well-known methods and restrict to the edges.

Bounding box
INSTANCE: Edge E on curve C with endpoints E_1 and E_2.
QUESTION: Find the smallest rectangle with sides parallel to the coordinate axes that contains E.
SOLUTION: Compute the local extrema of the curve and restrict to the edge, yielding S. Find the minimum x-value ($x_{\min}$) in $S \cup \{E_1, E_2\}$, and so on. The desired box is defined by the lines $x = x_{min}$, $x = x_{max}$, $y = y_{min}$, and $y = y_{max}$.

The bounding box (see [20, p. 372]) is useful for interference detection: the expensive intersection of edges can be reserved for those situations when the edges are close enough that their bounding boxes interfere. Bounding regions are also useful for problems such as the restriction problem, because they allow points that clearly do not satisfy a condition to be discarded quickly.

Another fundamental use of sorting[2] is to introduce an even-odd parity to a set of points, as

[2]In this paper, 'sorting' will always refer to the sorting of points along a curve, not the conventional sorting of numbers or words.

illustrated by the following problem.

Solid model intersection

INSTANCE: Two solid models M and N.

QUESTION: What is the intersection of M and N?

SOLUTION: An important step of this computation is to find the segments of an edge of one model that lie in the intersection. This is done by finding and sorting the points of intersection of this edge with a face of the other model. The segments of the edge between the i^{th} and $i+1^{st}$ intersections, for i odd, are contained in the intersection of the models.

Another application of even-odd parity is to decide whether a point lies within a piecewise-algebraic plane patch (or a piecewise-algebraic convex surface patch). This problem, which is fundamental to the display of a geometric model, is fully discussed in [16]. Having established the importance of sorting, in the next section we proceed to a discussion of methods for sorting.

3 Previous work on sorting

There is no serious study of sorting in the literature. This can be explained by the fact that nontrivial sorting problems arise only with curves of degree three or more, and until recently, almost all of the curves in solid models were linear or quadratic. However, as the science of geometric modeling matures and grows more ambitious, curves of degree three and higher are becoming common. For example, the introduction of blending surfaces [15] into a model creates curves and surfaces of high degree.

The lack of a study of sorting can also be explained by the presence of an obvious method for sorting points, which tends to obviate a search for any other method. This obvious method uses a rational parameterization of the curve (i.e., a parameterization $(x(t), y(t))$ such that both $x(t)$ and $y(t)$ can be expressed as the quotient of two polynomials in t), sorting a set of points S along $\widehat{AB}$ as follows.

The parameterization method of sorting

[Preprocessing]

1. Parameterize the curve.

[Solve]

2. Find the parameter values of A and B, say t_1 and t_2.

3. Find the parameter value of each point in S.

[Sort numbers]

4. Sort the parameter values of S from t_1 to t_2, discarding those outside this interval.

We insist upon a rational parameterization because a nonrational parameterization is difficult to represent and difficult to solve. With a nonrational parameterization (such as $x(t) = \sqrt{t}$ or $x(t) = \sin(t)$), two different points may have the same parameter value, which complicates sorting. Finally, there is no algorithm for the automatic parameterization of a curve that does not have a rational parameterization, whereas there is such an algorithm for rational curves [1].

There are many reasons to be dissatisfied with the parameterization method. It is not a universal method, since not all algebraic curves have a rational parameterization. Indeed, a plane algebraic curve has a rational parameterization if and only if its genus is zero, if and only if it has the maximum number of singularities allowable for a curve of its degree [26]. Secondly, even for those curves that do have rational parameterizations, the parameterization method will be slow if the degree of the parameterization is high, since the computation of the parameter values of the points will be expensive. Other weaknesses of the parameterization method will become clear as we compare it with the new method.

There is also a brute-force sorting method, which uses techniques for tracing along a curve [7]. The order of the points is the order in which they are encountered during a trace of the segment. This method is not satisfactory, because its implementation, although robust, is inherently very slow. Moreover, its complexity depends upon the length of the segment that is being sorted rather than upon the number of points in the sort, which is undesirable.

The weaknesses of the parameterization and tracing methods of sorting suggest that another method is necessary: one that will perform more efficiently on a wider selection of algebraic curves. The next section presents such a method. This method works with the implicit representation $f(x, y) = 0$ of a curve (as opposed to the parametric representation), thus allowing the use of tools from algebraic geometry.

4 The convex segment method of sorting

The observation that motivates the new method is that a convex segment can be sorted easily. Since every curve is a collection of convex segments, this suggests a divide and conquer strategy. A segment of a plane algebraic curve is *convex* if no line has more than two distinct intersections with it. (Alternatively, a planar segment is convex if it lies entirely on one side of the closed halfplane determined by the tangent line at any point of the segment [12].) The following theorem shows that sorting a convex segment is simple.

Theorem 1 *Let $p_1, \ldots, p_n$ be points on a convex segment $\widehat{AB}$, and let H be the convex hull of A, B, $p_1, \ldots, p_n$ (Figure 2). The order (from A to B) of $p_1, \ldots, p_n$ is simply the order (from A to B) of the vertices on the boundary of H.*

Proof: [16, p. 20]. ∎

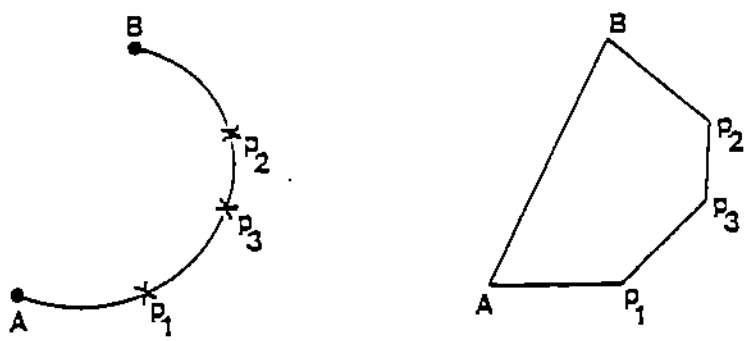

Figure 2: The sorting of a convex segment

Suppose that a curve can be decomposed into convex segments. Also suppose that we can identify the convex segment in this decomposition that contains a query point (point location in convex decomposition). These key problems will be discussed in Sections 5 and 6. The following algorithm shows how to sort a set of points S along the segment $\widehat{AB}$.

The convex segment method of sorting

[Preprocessing]

1. Decompose the curve into convex segments (say $\widehat{W_1W_2}$, $\widehat{W_2W_3}$, ..., $\widehat{W_{\beta-1}W_\beta}$).

[Locate first convex segment]

2. Find the convex segment that contains A (say $W_{i-1}\widehat{W}_i$).
3. Decide whether $\widehat{AB}$ leaves A along $A\widehat{W}_{i-1}$ or $A\widehat{W}_i$ (say $A\widehat{W}_i$).[3]
4. PresentConvexSegment := $A\widehat{W}_i$; j := i ; SortedSet := $\emptyset$; FoundB := false

[Sort one convex segment at a time]

5. Repeat until FoundB
   (a) Find the points of S that lie on PresentConvexSegment.
   If B is one of these points, then FoundB := true.
   (b) Sort these points along PresentConvexSegment, using Theorem 1.
   (c) If not FoundB,
   then SortedSet := Append(SortedSet,{sorted points on PresentConvexSegment})
   else SortedSet := Append(SortedSet,{sorted points on PresentConvexSegment before B})
   (d) PresentConvexSegment := $W_j\widehat{W}_{j+1}$; j := $j+1$

[Output]

6. Return SortedSet.

The expense of this method is concentrated in the preprocessing phase, which is done once off-line. The run-time operations (convex-segment sorting and locating a point on a convex segment) are usually very simple. Therefore, the efficiency of this method is very competitive. The coverage of the convex segment method is the entire set of algebraic curves, since it works directly from the implicit representation of the curve.

Example 4.1 *Consider the sorting of points $P_1, \ldots, P_6$ along the segment $\widehat{AB}$ of Figure 3. The curve is decomposed into convex segments by the dotted lines (Section 5). A lies on $W_1\widehat{W}_8$ and the vector at A identifies that $A\widehat{W}_1$ is the first convex segment. There are no points on $A\widehat{W}_1$, so we move on. The next convex segment is $W_1\widehat{W}_2$. Only P_1 lies on $W_1\widehat{W}_2$ and it becomes the first element of the sorted list. We jump to the next convex segment $W_2\widehat{W}_3$ and sort the two points P_2 and P_3 by creating the convex hull of W_2, W_3, P_2, and P_3. P_2 and P_3 are added to the global sort. We move on to the next convex segment $W_3\widehat{W}_4$, and then $W_4\widehat{W}_5$. The presence of B indicates that*

[3] If V is the vector at A that is given as part of the input, then $\widehat{AB}$ leaves A along $A\widehat{W}_i$ if and only if V points to the halfplane defined by $A\widehat{W}_i$ that contains $A\widehat{W}_i$.

this is the last convex segment. Upon sorting B and P_4, P_4 is discarded because it comes after B. The final sorted list is P_1, P_2, P_3.

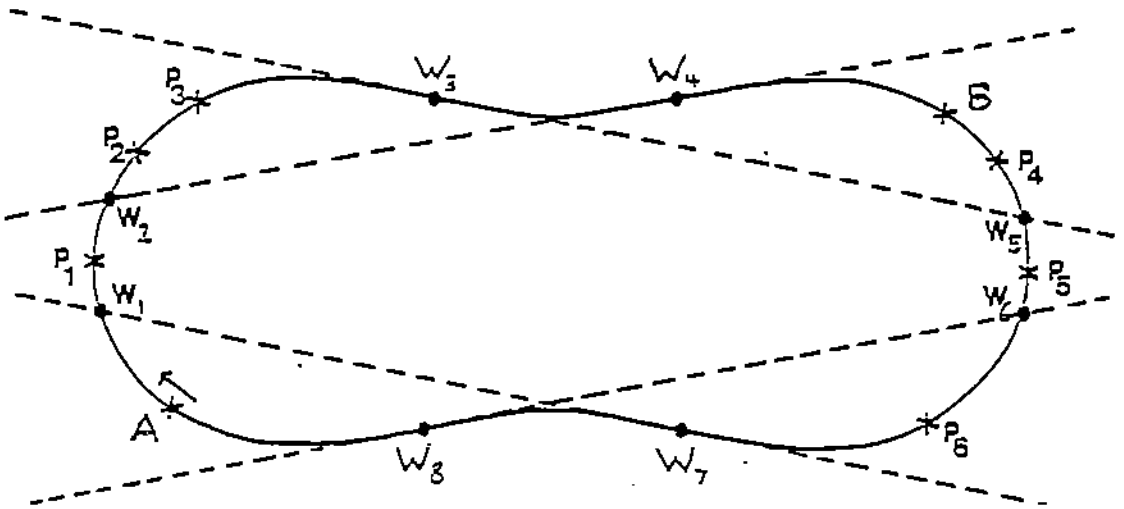

Figure 3: Sorting a curve by convex segments

It remains to discuss how a curve can be decomposed into convex segments and how a point can be located in this convex decomposition. These two problems, which are at the heart of the convex segment method of sorting, are solved in the following two sections.

5 Convex decomposition of a curve

The decomposition of an object into simple objects is an important theme in computational geometry. Decomposition proves to be particularly useful in divide-and-conquer algorithms, since simple objects are easily conquered. There has been a good deal of work on the decomposition of (simple, multiply connected, or rectilinear) polygons into simple components (e.g., triangles [10,13,14,24], quadrilaterals [23], trapezoids [5], convex polygons [9,25], and star-shaped polygons [6]), sometimes with added criteria (e.g., minimum decomposition [9,17], minimum covering [21], no Steiner points [17]). However, all of this work has been in the polygonal (or at best polyhedral) domain. The decomposition of a plane algebraic curve of arbitrary degree into convex segments is an extension of decomposition to the curved world.

A version of Bezout's Theorem states that two irreducible plane algebraic curves of degree m and n have exactly mn intersections (properly counted), unless the curves are identical [26]. Therefore, all plane algebraic curves of degree one (lines) and two (conics) are already convex.

For the convex decomposition of curves of degree three and higher, the singularities and points of inflection are instrumental. A *singularity* of the curve $f(x, y) = 0$ is a point P of the curve such that $f_x(P) = f_y(P) = 0$ (where f_x is the derivative of f with respect to x). It is a point where the curve crosses itself or changes direction sharply. A nonsingular point is also called a *simple* point. A *point of inflection* is a simple point P of the curve whose tangent has three or more intersections with the curve at P. (It is also a point of zero curvature.) We restrict our attention to points of inflection P such that P's tangent has an odd number of intersections with the curve at P, which we call *flexes* for short. Fundamental in algebraic and differential geometry, singularities and flexes form a skeleton of the curve and can be used in many useful ways. (For example, singularities can be used to parameterize a plane algebraic curve [1].) Their use in convex decomposition underlines their importance to computational geometry of higher degrees.

The tangents at the singularities and flexes of a curve form an arrangement of lines that subdivide the plane of the curve into several cells, called a *cell partition* (Figures 3-4). The tangents also split the curve into several segments. The following theorem establishes that each of these segments is convex.

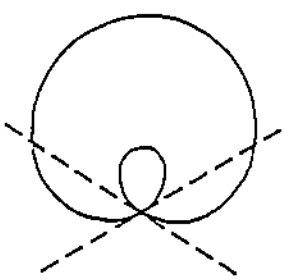

Figure 4: Convex segmentation of limacon of Pascal

Theorem 2 *The tangents of the singularities and flexes of a plane algebraic curve slice the curve into convex segments. That is, if $\widehat{PQ}$ is a nonconvex segment, then some tangent of a singularity or flex will intersect $\widehat{PQ}$.*[4]

Proof:

Let $\widehat{PQ}$ be a nonconvex segment of an algebraic curve. Assume without loss of generality that $\widehat{PQ}$ does not contain a singularity or a flex. It can be shown that there exists a line L that crosses

[4]The simple points at which a singularity/flex tangent touches, but does not cross, the curve are redundant and should not be treated as convex segment endpoints in the decomposition.

$\widehat{PQ}$ at three (or more) distinct points [16, p. 117].[5] Let x_1, x_2, and x_3 be three of these points, such that $x_2 \in \widehat{x_1x_3}$ and $\widehat{x_1x_3} \cap L = \{x_1, x_2, x_3\}$. $\widehat{x_1x_3}$ does not change its direction of curvature, since there is no singularity or flex on $\widehat{PQ}$. $\widehat{x_1x_3}$ is not a line segment, otherwise Bezout's Theorem would imply that the algebraic curve that contains $\widehat{x_1x_3}$ is a line, which it cannot be since it contains a nonconvex segment. Therefore, it can be assumed without loss of generality that $\widehat{x_1x_3}$ looks like Figure 5(a). Let R be the closed region bounded by $\widehat{x_1x_3}$ and $\overline{x_1x_3}$. We will show that R contains a singularity or a flex. This will complete the proof, since the tangent of a point inside R must intersect $\widehat{x_1x_3} \subset \widehat{PQ}$ at least once. (The tangent must cross the boundary of R twice, and at most one of these intersections can be with $\overline{x_1x_3}$.) The curve lies inside of R as it leaves $\widehat{x_1x_3}$ from x_1 and outside of R as it leaves $\widehat{x_1x_3}$ from x_3. Therefore, the curve must cross the boundary of R after it leaves $\widehat{x_1x_3}$ from x_1, either because it must join with x_3 (if the curve is closed) or because an infinite segment of an algebraic curve cannot remain within a closed region (if the curve is open) [16]. The curve cannot intersect the $\widehat{x_1x_3}$ boundary of R, since $\widehat{x_1x_3} \subset \widehat{PQ}$ is nonsingular by assumption. Therefore, the curve must cross $\overline{x_1x_3}$ after it leaves $\widehat{x_1x_3}$ from x_1.

As the curve leaves $\widehat{x_1x_3}$ from x_1, it lies on the opposite side of x_1's tangent from $\overline{x_1x_3}$. Therefore, after the curve leaves $\widehat{x_1x_3}$ from x_1 and before it leaves R, the curve must cross x_1's tangent inside of R, in order to reach $\overline{x_1x_3}$. In order to cross over x_1's tangent, the curve must cross itself or change its curvature inside of R (Figure 5(b)), otherwise it will spiral around inside R forever. Therefore, R contains a singularity or a flex. ■

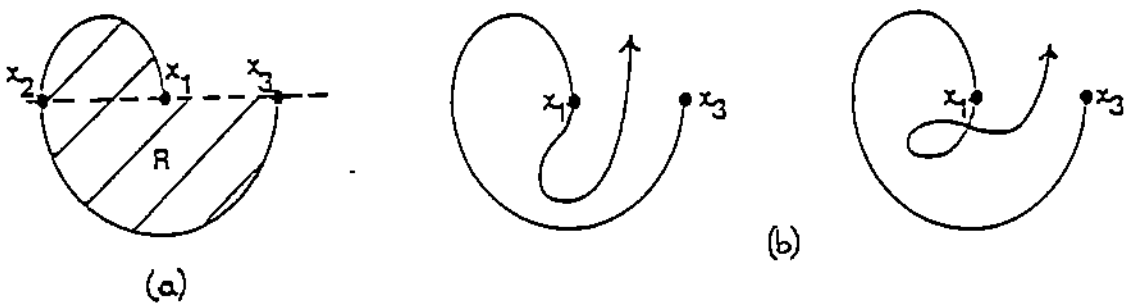

Figure 5: (a) $\widehat{x_1x_3}$ and R (b) travelling from x_1 to $\overline{x_1x_3}$

We include here a word about robustness. Consider the accuracy required in the computation of the singularities, flexes, and their tangents in order to guarantee a true division into convex

[5] Already, by the definition of convexity, there must exist a line that intersects $\widehat{PQ}$ three (or more) times.

segments. Suppose that, in the proof of Theorem 2, the tangent of a singularity/flex inside the region R is used to split a nonconvex segment. Any line through a point in the interior of R would work equally well in splitting the nonconvex segment. Thus, in this case the method is robust under slight errors in tangents, singularities, and flexes. The other case is if a nonconvex segment S is split into convex segments by a singularity or flex lying on S. The computed convex segment will differ from the actual convex segment by the same amount as the computed flex (say) differs from the actual flex. The only points that might be treated improperly are those that lie on the segment between the computed and actual flex. In other words, points that are within (some function of) machine precision of each other cannot be distinguished by the method and must be considered equivalent. This equivalence of points within machine precision is inherent to any sorting algorithm.

Theorem 2 does not solve the convex decomposition problem, because it yields a confused collection of endpoints of convex segments, not a collection of convex segments. The more challenging step of pairing up the endpoints remains, where two endpoints are *partners* if they define a convex segment of the decomposition. This pairing problem will be attacked in Sections 5.3 and 5.4, but first the collection of convex segments must be refined.

5.1 Refinement of convex segments I: Singularities

Many of the endpoints of the convex segments created by Theorem 2 are singularities. However, singular endpoints cause problems in pairing. Consider a convex segment whose two endpoints are the same point, which might occur around a singularity (Figure 4). This situation is to be avoided, since pairing will turn out to be easier if the two endpoints of a convex segment are different. It is also possible for a singularity to have more than two partners and, in particular, two partners in the same cell. This situation is also to be avoided, since it is easier to find the partner of an endpoint in a cell if this partner is unique.

Another problem with singular endpoints is that the ordering of points about a singularity can be ambiguous. Does P_2 or P_3 follow A in Figure 6(a)? What is the order of the points in Figure 6(b): S, P_1, P_2, P_3, S or S, P_3, P_2, P_1, S? As a result of these problems, all convex segments with singular endpoints will be replaced by convex segments with nonsingular endpoints.

A pair of points will be found on each branch of the curve that passes through a singularity, one on either side of (and very close to) the singularity. The added points will receive the convex segments that enter the singularity. After each singularity of the curve has been decomposed in this manner, every convex segment of the curve will be bounded by simple points, as desired.

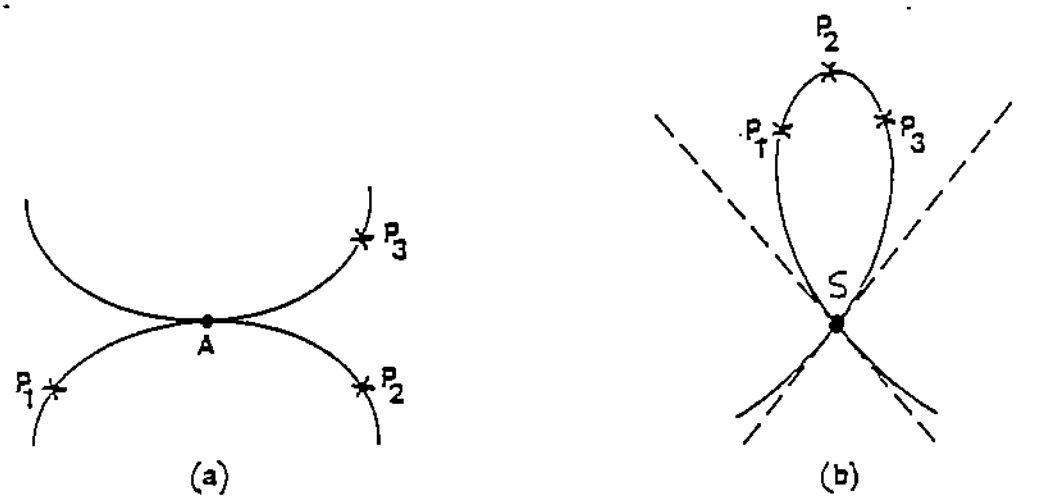

Figure 6: Ambiguity about a singularity

Example 5.1 *Four points are associated with the singularity A of Figure 7: V_1 and V_2 from one branch, W_1 and W_2 from the other. The convex segments of the two cells are now $\widehat{PV_1}$, $\widehat{V_1V_2}$, $\widehat{V_2Q}$, $\widehat{RW_1}$, $\widehat{W_1W_2}$, and $\widehat{W_2S}$. Notice that this refinement makes it clear that Q (not S) must follow P.*

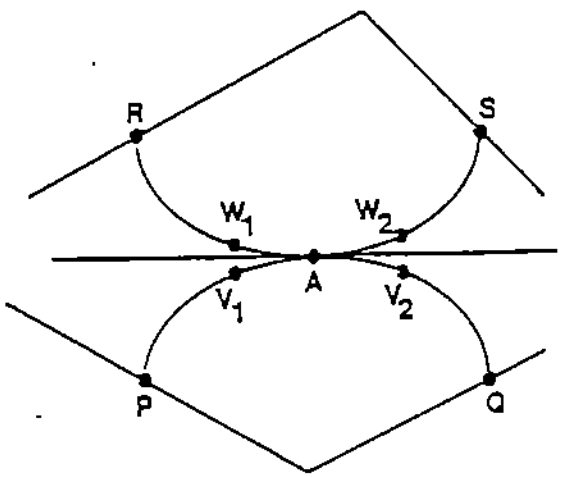

Figure 7: The refinement of a singularity

Consider the problem of finding two points on each branch, one on either side of the singularity. We would like to do this by tracing a small distance along the branch in both directions from the singularity. However, there is no reliable way of tracing along a branch as it passes through a singularity, because the other branches create too much confusion. Therefore, each branch of the singularity must be isolated so that it can be traced robustly. This isolation is accomplished by

blowing up the curve at the singularity by a series of quadratic transformations [7,26], as follows.

The first step in blowing up a singularity is to translate it to the origin.[6] Let the new equation of the curve be $f(x,y)=0$. A quadratic transformation is applied to the curve. The *affine quadratic transformation* $x = x_1,\ y = x_1 y_1$ [26] has three important properties:

- It maps the origin to the entire y_1-axis and the rest of the y-axis to infinity: $y_1 = \frac{y}{x}$ so $(0,b)$ maps to $(0, \frac{b}{0})$, which is a point at infinity unless $b = 0$.
- It is one-to-one for all points (x,y) with $x \neq 0$.
- $y = mx$, a line through the origin, is mapped to the horizontal line $y_1 = m$: $y = mx \rightarrow x_1 y_1 = m x_1 \rightarrow y_1 = m$.

Thus, a quadratic transformation maps distinct tangent directions of the various branches of f at the origin to different points on the *exceptional line* $x_1 = 0$. The intersections of the transformed branches with the exceptional line correspond to the transformed points of the origin (Figure 8). If a point of $f(x_1, x_1 y_1)$ on the exceptional line is singular, then the procedure is applied recursively (Figure 9). The following lemma establishes that the various branches of the curve in the neighbourhood of the singularity eventually get transformed to separate branches.

Figure 8: (a) node and (b) its quadratic transformation

Lemma 1 ([1,26]) *A singularity can be reduced to a number of simple points by a finite number of applications of the quadratic transformation. An ordinary singularity can be reduced to simple points by a single quadratic transformation, where a singularity of multiplicity r is ordinary if its r tangents are all distinct.*

[6] Since the quadratic transformation does not map the line $x = 0$ properly, the curve should also be rotated (if necessary) so that it is not tangent to $x = 0$ at the origin (see [16]).

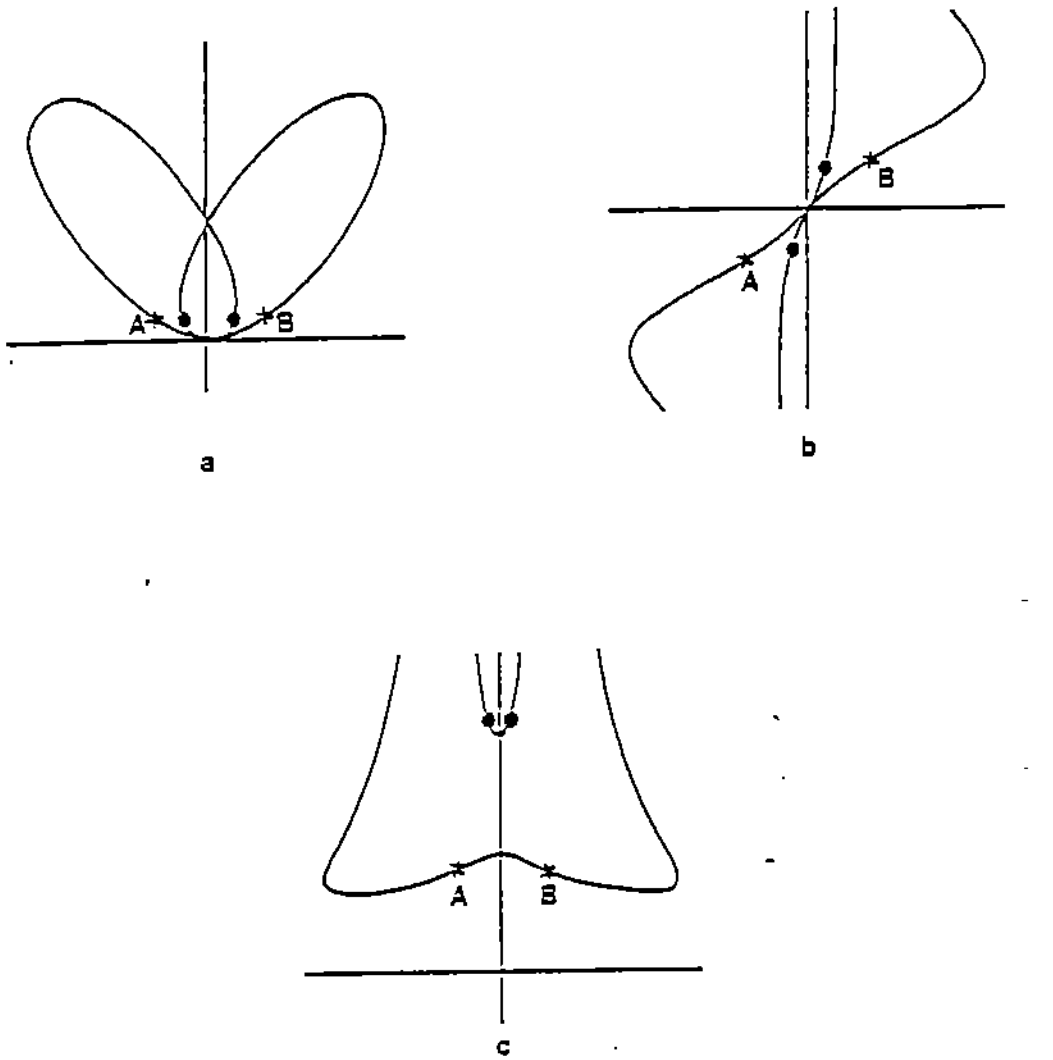

Figure 9: (a) the original singularity (b) after one quadratic transformation (c) after a second transformation: the original singularity successfully transformed into two simple points

To summarize, each singularity is translated to the origin and transformed into a set of non-singular points through the application of a series of quadratic transformations. Each branch of the transformed curve intersects the exceptional line in a simple point, so this image branch can be traced from the image singularity without confusion. Therefore, upon each image branch, two points are found by tracing a very short distance in either direction from the image singularity. Finally, these points are mapped back to the original curve to become new endpoints, replacing the singularity. These new endpoints clarify the branch connectivity at the singularity and simplify the job of pairing.

Care must be taken with the short segment that is essentially sliced out of the curve during the refinement of the singularity, such as $\widehat{V_1V_2}$ in Figure 7. It is a special convex segment and points that lie on it are sorted in a special way, by mapping them to the blown-up, desingularized, image curve and using the tracing method. This is not expensive because the sliced-out segment is very short and very few steps are needed to trace over it.

5.2 Refinement of convex segments II: Infinite segments

Convex segments with singular endpoints are not the only ones that must be refined: infinite convex segments are also problematic. The pairing process is simplified if each convex segment has two endpoints, but an infinite convex segment has only one endpoint. Therefore, an artificial endpoint is added to each infinite segment, as follows.

Every open cell is artificially closed by a collection of line segments (Figure 10). These line segments are chosen carefully so that they only intersect infinite convex segments (if any) in the cell, and each of these exactly once (unless the infinite segment is entirely contained in the cell and thus proceeds to infinity at both ends, in which case two intersections are allowed). The resulting artificially-closed cell should also be a convex polygon. A point of intersection of an infinite convex segment with the new boundary of its cell becomes an *(artificial) endpoint* (Figure 18). Thus, infinite convex segments are transformed into finite convex segments with two endpoints. After every endpoint has been assigned a partner, pairs that contain an artificial endpoint are recognized as infinite convex segments. A pair of artificial endpoints represents an entire connected component that does not cross any of the singularity/flex tangents.

After the above two refinements, the set of endpoints of convex segments assumes the following normal form:

- every endpoint has exactly two partners
- every cell is a closed polygon

The normalization stage not only makes pairing easier: it also creates a cleaner set of convex segments that better reflects the curve. For example, due to the first normal condition, pairing will create a collection of convex segments with an implicit order.

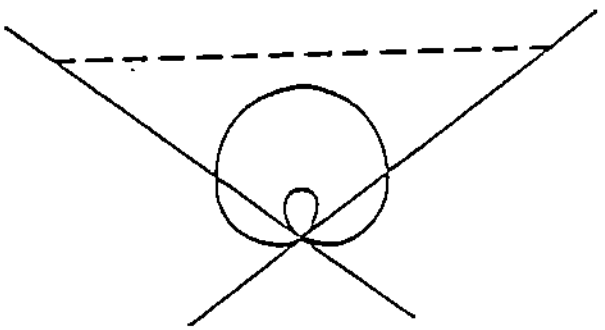

Figure 10: The artificial closure of an open cell

5.3 Pairing of endpoints I: Properties of the partner

We are now ready to show how to pair the endpoints of convex segments. Consider a convex segment in cell C and an endpoint E of this segment. E's partner in C must obviously be another endpoint in C. Therefore, the determination of partners in all single-segment cells is trivial. Corollary 1 will present other conditions that E's partner must satisfy and Theorem 3 will show how to isolate the partner if several endpoints satisfy all of these conditions. In preparation, some terminology must be introduced and a crucial lemma proved.

Definition

If P is a singularity or flex, then P's tangent is a cell wall and the *inside of P's tangent* w.r.t. (with respect to) a cell C is the halfplane that contains C. Otherwise, the inside is the halfplane that contains all of the curve in the neighbourhood of P (Figure 11). The inside includes the tangent, while the strict inside does not.

Let P be a flex that lies on the wall W of cell C, and let P_ϵ be a point of the curve inside cell C at distance $\epsilon > 0$ from P. (P_ϵ may be found by tracing the curve into C from P.) The *outside wallpoint* of W w.r.t. C is the endpoint of W that lies outside of P_ϵ's tangent, for ϵ small (E in Figure 12).

If P is not a flex, then P *faces* Q if Q lies on the inside of P's tangent (Figure 11(a)). Otherwise, P faces Q w.r.t. cell C if (1) Q lies strictly inside P's tangent w.r.t. C or (2) Q lies on P's tangent and on the opposite side of P from the outside wallpoint of P's wall w.r.t. C (Figure 12).

Notation 1 *$\#\{S\}$ is the number of elements in the set S and $\overline{xy}$ is the line segment between x and y. $\overline{xy}$ does not include its endpoints x and y.*

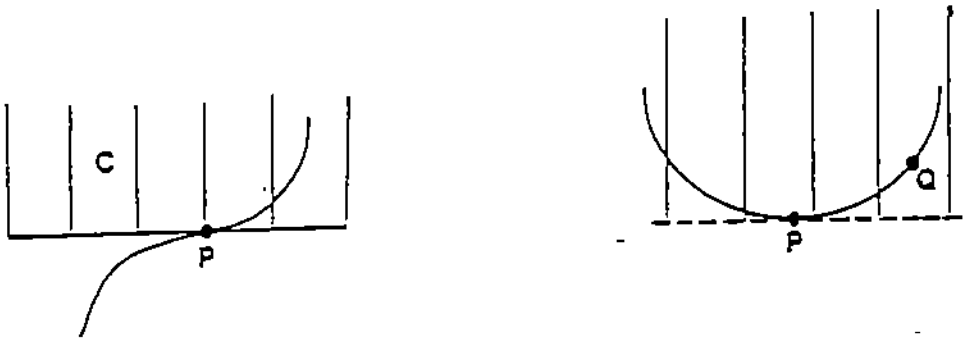

Figure 11: The inside of P's tangent

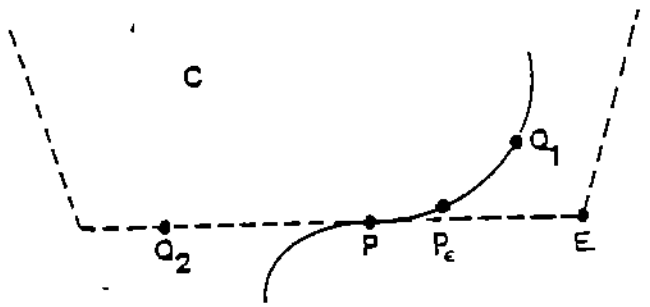

Figure 12: P faces both Q_1 and Q_2 with respect to C

Lemma 2 *Consider the cell partition of a curve F. Let X and Y be two nonsingular points of a convex segment in the cell C. Then*

1. The curve crosses[7] *$\overline{XY}$ at an even number of points, ignoring singularities.*

[7] If P is a point of intersection of the curve with $\overline{XY}$, then the curve *crosses* $\overline{XY}$ at P if it lies on both sides of $\overline{XY}$ in any neighbourhood of P; otherwise it only touches $\overline{XY}$ at P.

2. $\#\{P \in \overline{XY} \cap F : P \text{ faces } X \text{ w.r.t. } C\} = \#\{P \in \overline{XY} \cap F : P \text{ faces } Y \text{ w.r.t. } C\}$

3. $\forall \alpha \in \overline{XY},\ \#\{P \in \overline{X\alpha} \cap F : P \text{ faces } X \text{ w.r.t. } C\} \leq \#\{P \in \overline{X\alpha} \cap F : P \text{ faces } Y \text{ w.r.t. } C\}$

Example 5.2 *Figure 13 is a hypothetical example for Lemma 2. The curve F crosses $\overline{XY}$ an even number of times. $\{P \in \overline{XY} \cap F :\ P \text{ faces } X\} = \{P_2, P_3, P_6\}$ is of the same size as $\{P \in \overline{XY} \cap F :\ P \text{ faces } Y\} = \{P_1, P_3, P_4\}$. Moreover, $\{P \in \overline{X\alpha} \cap F :\ P \text{ faces } X\} = \{P_2\}$ is smaller than $\{P \in \overline{X\alpha} \cap F :\ P \text{ faces } Y\} = \{P_1, P_3, P_4\}$.*

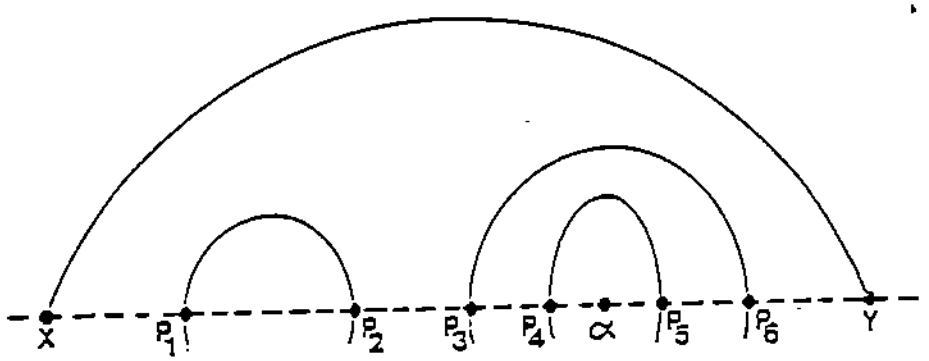

Figure 13

Proof of Lemma 2:

Consider the closed region R_{XY} bounded by $\overline{XY}$ and $\widehat{XY}$. Since $\widehat{XY}$ lies in the cell C and C is a convex polygon, $\overline{XY}$ must also lie in C. Therefore, again by convexity, R_{XY} must lie in C. Since X and Y are nonsingular and the rest of $\widehat{XY}$ lies in the interior of the cell, $\widehat{XY}$ does not contain a singularity. Therefore, the curve can only cross into R_{XY} through $\overline{XY}$. If the curve enters R_{XY}, then it must also leave, since an infinite segment cannot remain within a closed region and an algebraic curve of finite length is closed (viz., the curve cannot stop short in the middle of R_{XY}). We claim that the point of departure D must be distinct from the point of entry E, unless all of the tangents at $D = E$ are $\overleftarrow{XY}$, as in Figure 14. Otherwise, if $D = E$, then at least one of the tangents of the singularity D will cross into R_{XY} and form a wall of the cell partition which will split R_{XY} in two, contradicting the fact that all of R_{XY} lies in the same cell. Therefore, with the exception of the special singularities of Figure 14, the crossings of $\overline{XY}$ by the curve occur in pairs, called *couples*. This establishes condition (1) of the lemma.

Consider condition (2). The special singularities of Figure 14 (as well as the points where the curve only touches $\overline{XY}$) can be ignored during the consideration of conditions (2) and (3), since

Figure 14: The only type of singularity that can lie on $\overline{XY}$

they face both X and Y and contribute the same amount to the left-hand side and right-hand side of the expressions of conditions (2) and (3). Therefore, we can concentrate on the remaining crossings of $\overline{XY}$: the distinct couples. Let $A, B \in \overline{XY}$ be a couple and assume, without loss of generality, that A lies closer to X than B does. $\widehat{AB}$ is a convex segment since it lies within a cell of the cell partition. Therefore, A and B face each other (w.r.t. cell C). Since A faces B, A faces Y. Similarly, since B faces A, B faces X. Therefore, one member of each couple faces X and the other faces Y, yielding condition (2). Moreover, the point of a couple that faces Y (A) is closer to X than the point that faces X (B), yielding condition (3). ∎

Corollary 1 *Let W_1 be an endpoint in the cell C. W_1's partner W_2 in C must satisfy the following properties:*

1. *W_1 and W_2 must face each other (w.r.t. C)*
2. *the curve must cross $\overline{W_1W_2}$ at an even number of points, ignoring singularities*
3. *the number of these crossings that face W_1 (w.r.t. C) is equal to the number that face W_2 (w.r.t. C)*
4. *for any $\alpha \in \overline{W_1W_2}$, the number of crossings in the interval $\overline{W_1\alpha}$ that face W_1 is bounded by the number of crossings in this interval that face W_2*

These conditions, which capture the fact that the intersections of the curve with $\overline{W_1W_2}$ pair up into couples that face each other, will often isolate the partner.

Example 5.3 *Consider the cell partition of Figure 15 and the cell containing the convex segments $\widehat{W_1W_2}$ and $\widehat{W_3W_4}$. Suppose that we wish to find the partner of W_1. W_3 violates condition (1) and W_4 violates condition (2), so W_2 must be W_1's partner.*

The following technical lemma is necessary for later proofs.

Lemma 3 ([16]) *Let W_1 and W_2 be partners. If W_2 lies on W_1's tangent, then W_1 must be a flex.*

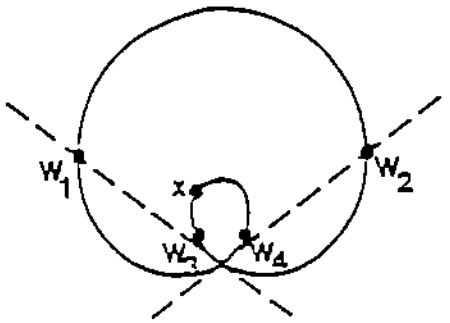

Figure 15

5.4 Pairing of endpoints II: Distinguishing between candidates

The remaining question in endpoint-pairing is how to find the partner of an endpoint W_1 in C if several endpoints in C satisfy all of the conditions of Corollary 1. This will be done by sorting the candidates about the cell boundary (Theorem 3). Unfortunately, the refinement of singularities moved some of the endpoints of convex segments into the interior of cells. Therefore, in order to allow sorting about the boundary, we must associate a point W' on the cell boundary with each endpoint W that was created in the singularity refinement stage, as follows. If $W \neq W_1$, then W' is the intersection of the ray $W_1\bar{W}$ with the cell boundary (Figure 16(a)). If $W = W_1$, then W' is one of the (two) intersections of W_1's tangent with the cell boundary: the one that lies on a tangent of the singularity from which W_1 was derived (Figure 16(b)). For notational consistency, $W' = W$ if W is an endpoint that already lies on the cell boundary.

Theorem 3 *Let W_1 be an endpoint in cell C of the cell partition of a curve F, $R(W_1)$ the set of endpoints in C that satisfy the conditions of Corollary 1 (w.r.t. W_1), and $S(W_1)$ the set of endpoints in $R(W_1)$ that lie* strictly *inside of W_1's tangent (w.r.t. C).*

If $S(W_1) \neq \emptyset$, let $S'(W_1) := \{ W' : W \in S(W_1) \}$. If W_1 is not a flex, let $X \neq W_1'$ be the other intersection of W_1's tangent with the cell boundary, otherwise let X be the outside wallpoint of W_1's wall w.r.t. C (Figure 17). W_1' and X split the cell boundary into two halves. Since every endpoint in $S'(W_1)$ will lie on the same half, a sort of $S'(W_1)$ from W_1' to X is well-defined. Let $S_1', S_2', \ldots, S_p'$ be the result of this sort (i.e., S_i' is encountered before S_{i+1}' in a traversal of the cell boundary from W_1' to X). The partner of W_1 in C is S_p (the endpoint associated with S_p').

If $S(W_1) = \emptyset$, let $T(W_1)$ be the set of endpoints in $R(W_1)$ that lie on the same wall as W_1. The partner of W_1 in C is the element of $T(W_1)$ that is closest to W_1.

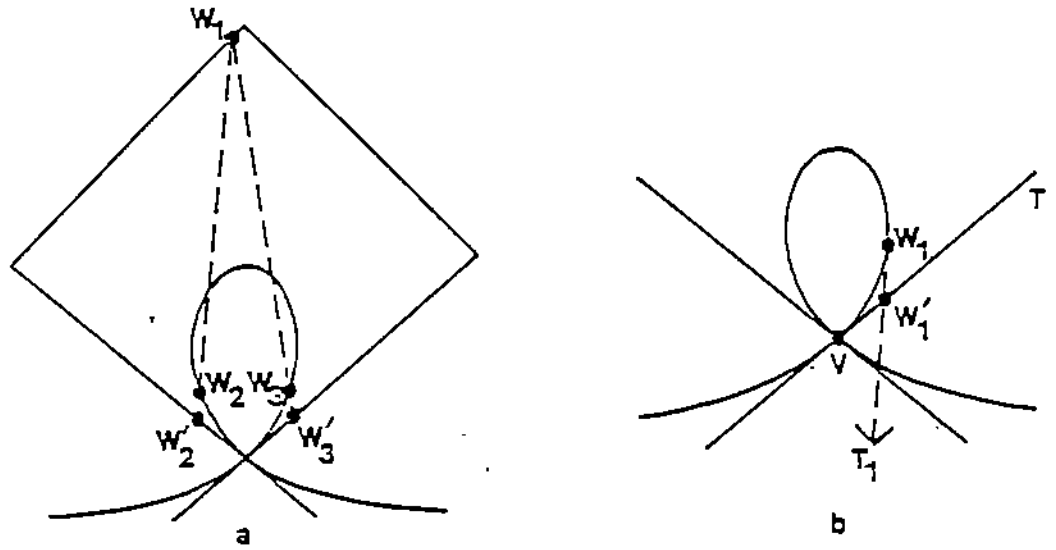

Figure 16: The boundary points W_i'

Example 5.4 *Consider the computation of W_1's partner in Figure 18, where W_1 is the endpoint of an infinite convex segment. $R(W_1) = S(W_1) = \{W_2, W_3, W_4\}$ and $S'(W_1) = \{W_2, W_3, W_4'\}$. The sorted order of $S'(W_1)$ along the boundary from $W_1' = W_1$ to X is W_3, W_4', W_2, so W_2 is the partner of W_1. Since W_2 is an artificial endpoint, W_1 must be the endpoint of an infinite convex segment.*

Consider the computation of the partner of W_1 in Figure 19, where $S(W_1) = \emptyset$. V_1, V_2 and V_4 are ruled out by condition (1) of $R(W_1)$, while V_3 and V_6 are ruled out by condition (2). Therefore, $T(W_1) = \{V_5, W_2\}$. W_2 is the closest element of $T(W_1)$ to W_1, so it is W_1's partner.

Proof of Theorem 3: Suppose that $S(W_1) \neq \emptyset$. Let W_2 be W_1's partner, and let $\widehat{W_1W_2}$ be the boundary of the cell from W_1' to W_2', such that $X \notin \widehat{W_1W_2}$ (Figure 20(a)). I claim that it is sufficient to show that $W_2' \in S'(W_1) \subset \widehat{W_1W_2}$. Suppose that this is true, and consider a traversal of the cell boundary from W_1' to X. Since W_2' is an endpoint of $\widehat{W_1W_2}$ and $X \notin \widehat{W_1W_2}$ (by definition), W_2' must be the last element of $S'(W_1)$ that is met during this traversal. In other words, $W_2' = S_p'$ ($W_2 = S_p$) as desired. (Since it can be shown that $S_i' \neq S_j'$ whenever $i \neq j$, there is no ambiguity in choosing the last member of $S'(W_1)$ or in associating S_i' with S_i [16].)

We will first show that $S'(W_1) \subset \widehat{W_1W_2}$. Let $s \in S(W_1)$. Suppose, for the sake of contradiction, that $\overline{W_1s}$ crosses $\widehat{W_1W_2}$ at $y \neq W_2$ (Figure 20(b-c)). There are two cases to consider: $y \in \overline{W_1s}$ and

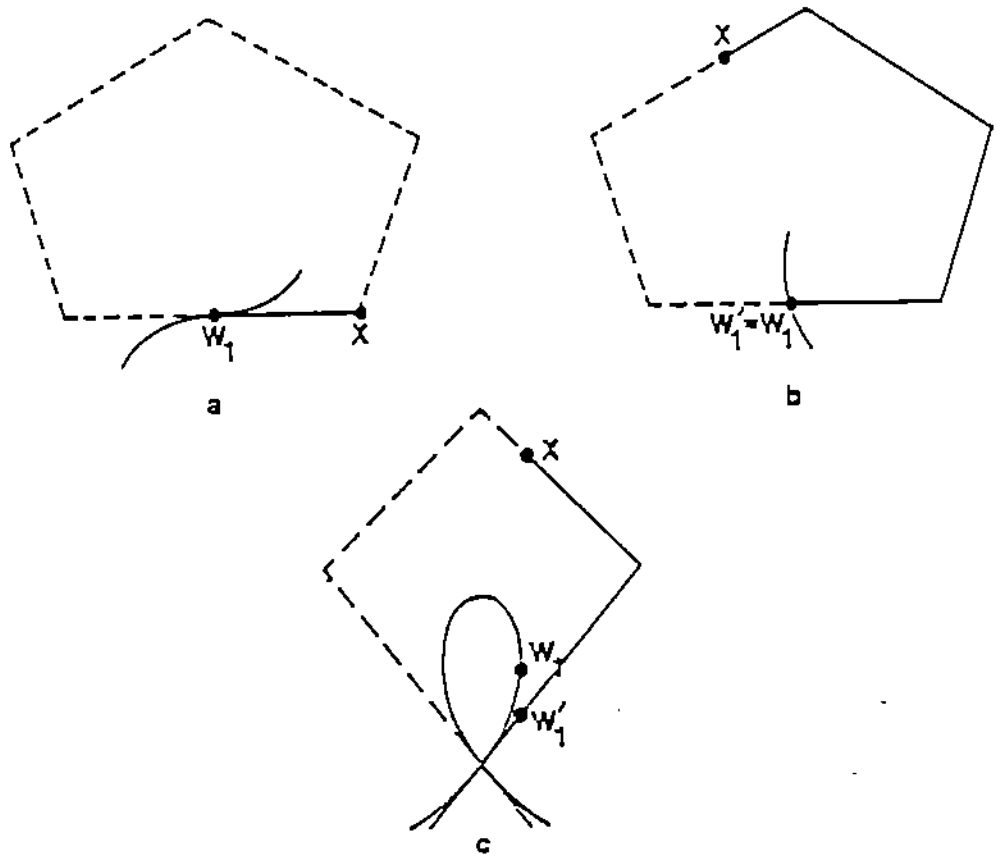

Figure 17: Partitioning the boundary of a cell

$s \in \overline{W_1 y}$. Suppose that $y \in \overline{W_1 s}$ (Figure 20(b)). By Lemma 2,

$$\#\{P \in \overline{W_1 y} \cap F : \text{ P faces } W_1\} = \#\{P \in \overline{W_1 y} \cap F : \text{ P faces y}\}$$

But y faces W_1, since W_1 and y are on the same convex segment. Therefore, there exists $\alpha \in \overline{W_1 s}$ such that

$$\#\{P \in \overline{W_1 \alpha} \cap F : \text{ P faces } W_1\} > \#\{P \in \overline{W_1 \alpha} \cap F : \text{ P faces s}\}$$

in contradiction of $s \in S(W_1)$. Now suppose that $s \in \overline{W_1 y}$ (Figure 20(c)). By the argument of the proof of Lemma 2, the points of intersection of the curve F with $\overline{W_1 y}$ pair up. Let t be the partner of s . Since $\widehat{st}$ is convex, s faces t; since $s \in S(W_1)$, s faces W_1. Therefore, $t \in \overline{W_1 s}$. Since $s \in S(W_1)$,

$$\#\{P \in \overline{W_1 s} \cap F : \text{ P faces } W_1\} = \#\{P \in \overline{W_1 s} \cap F : \text{ P faces s}\}$$

Noting that $\overline{W_1 s} = \overline{W_1 t} \cup \overline{ts} \cup \{t\}$ and t faces s, this becomes

$$\#\{P \in \overline{W_1 t} \cap F : \text{ P faces } W_1\} + \#\{P \in \overline{ts} \cap F : \text{ P faces } W_1\} + 0 =$$

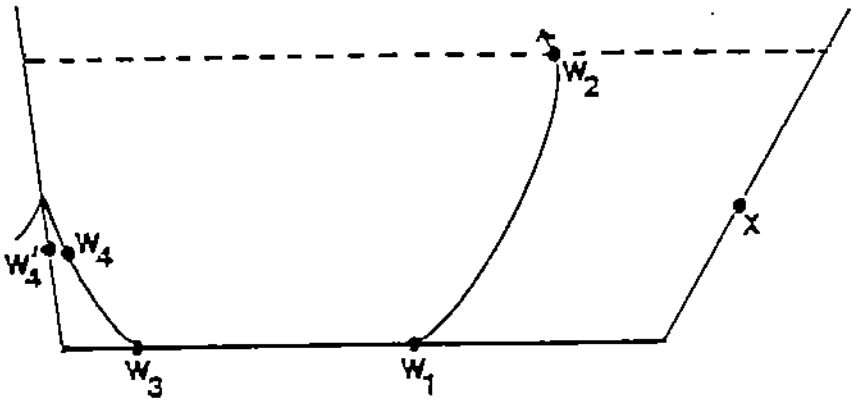

Figure 18: Computing the partner of the endpoint of an open convex segment

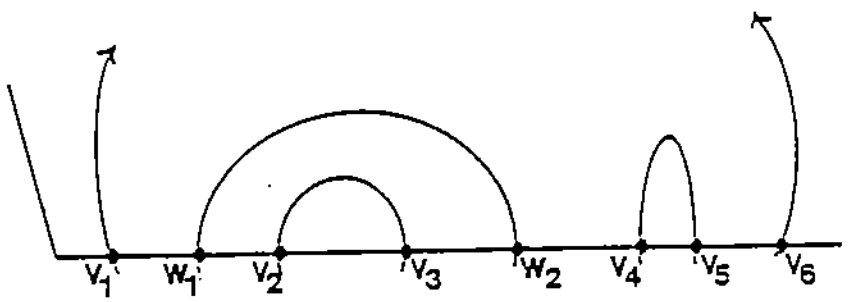

Figure 19: Partner computation when $S(W_1) = \emptyset$

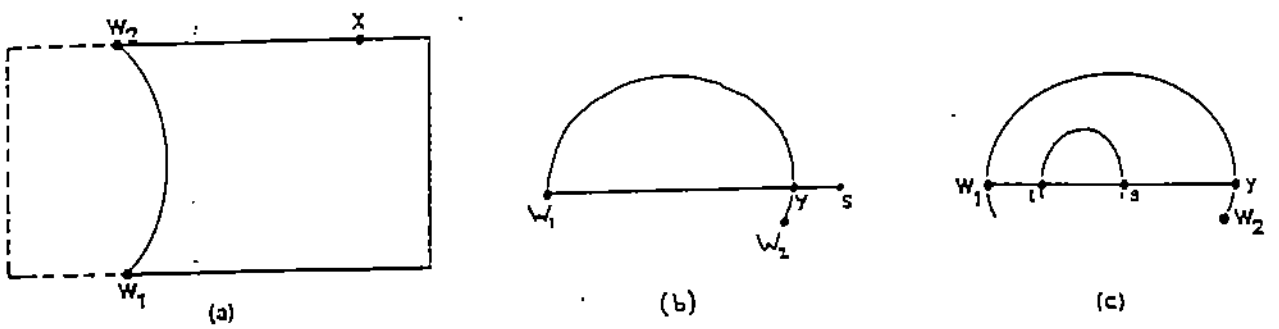

Figure 20: (a) $\widehat{W_1W_2}$ is dotted (b) $y \in \overline{W_1 s}$ (c) $s \in \overline{W_1 y}$

$$\#\{P \in \overline{W_1 t} \cap F : \text{ P faces s}\} + \#\{P \in \overline{ts} \cap F : \text{ P faces s}\} + 1$$

Moreover, by Lemma 2 ($\widehat{st}$ is convex),

$$\#\{P \in \overline{ts} \cap F : \text{ P faces s}\} =$$

$$\#\{P \in \overline{ts} \cap F : \text{ P faces t}\} =$$

$$\#\{P \in \overline{ts} \cap F : \text{ P faces } W_1\}$$

Upon cancelling terms in the above equation, we conclude that

$$\#\{P \in \overline{W_1 t} \cap F : \text{ P faces } W_1\} >$$

$$\#\{P \in \overline{W_1 t} \cap F : \text{ P faces s}\} =$$

$$\#\{P \in \overline{W_1 t} \cap F : \text{ P faces y}\}$$

But this contradicts condition (3) of Lemma 2 (convex segment $\widehat{W_1 W_2}$, $X = W_1$, $Y = y$). These contradictions lead us to conclude that $\overrightarrow{W_1 s}$ does not cross $\widehat{W_1 W_2} \setminus \{W_2\}$. In particular, by the definition of s', $\overline{W_1 s'}$ does not cross $\widehat{W_1 W_2} \setminus \{W_2\}$. Therefore, s' must either lie outside of W_1's tangent or on $\widehat{W_1 W_2}$(Figure 20(a)). Since s, as a member of S(W_1), lies on the strict inside of W_1's tangent, so must s'. Therefore, $s' \in \widehat{W_1 W_2}$ and $S'(W_1) \subset \widehat{W_1 W_2}$, as desired.

We now show that $W_2 \in S(W_1)$. $W_2 \in R(W_1)$ by Corollary 1, so it suffices to show that W_2 lies strictly inside of W_1's tangent. Suppose, for the sake of contradiction, that W_2 lies on W_1's tangent. By Lemma 3, W_1 must be a flex (whose tangent is a cell wall). Thus, the wall segment $\overline{W_1 W_2}$ is a subsegment of W_1's tangent and $S(W_1) \cap \overline{W_1 W_2} = \emptyset$ (by definition of $S(W_1)$). Therefore, $S'(W_1) \cap \overline{W_1 W_2} = \emptyset$. But $S'(W_1) \subset \widehat{W_1 W_2} = \overline{W_1 W_2}$. Thus, $S'(W_1) = \emptyset$, which contradicts our initial assumption. We conclude that W_2 does not lie on W_1's tangent. Since $\widehat{W_1 W_2}$ is a convex segment, W_2 lies on the inside of W_1's tangent, and thus on the strict inside.

The statement of the theorem has been verified if $S(W_1) \neq \emptyset$. Now suppose that $S(W_1) = \emptyset$. If W_1 is a refined singularity, then $W_2 \in S(W_1)$: $W_2 \in R(W_1)$ (as W_1's partner); W_2 does not lie on W_1's tangent (Lemma 3); and W_2 lies inside W_1's tangent (because $\widehat{W_1 W_2}$ is convex). This would contradict the $S(W_1) = \emptyset$ assumption, so W_1 cannot be a refined singularity. Therefore, W_1 must lie on a wall of the cell and T(W_1) is well-defined. If W_2 lies strictly inside W_1's wall (w.r.t. C), it also lies strictly inside W_1's tangent (Lemma 3). Therefore, if $W_2 \notin T(W_1)$, then $W_2 \in S(W_1)$. But $S(W_1) = \emptyset$, so $W_2 \in T(W_1)$.

Suppose that W_2 is not the closest member of $T(W_1)$ to W_1, and let $U \neq W_2$ be the closest. Since W_1 faces U, U must lie on $\overline{W_1W_2}$. By the proof used in Lemma 2, the nonsingular points of intersection of the curve with $\overline{W_1W_2}$ must pair up into couples. In particular, the endpoints on $\overline{W_1U} \subset \overline{W_1W_2}$ (all of which are nonsingular because of refinement) that face W_1 must pair with the equal number of endpoints on $\overline{W_1U}$ that face U. But U must also pair with an endpoint on $\overline{W_1U}$ that faces U, and there are no such endpoints remaining without a partner. This contradiction leads us to conclude that W_1's partner W_2 must be the closest element of $T(W_1)$ to W_1. ∎

5.5 Computation of Singularities and Flexes

The above convex decomposition of an algebraic curve requires the singularities and flexes of the curve, as well as their tangents. The singularities of a curve $f(x, y) = 0$ are the solution set of the system $\{f_x = 0, f_y = 0, f = 0\}$, while the points of inflection are the nonsingular intersections of the curve with its Hessian (the determinant of the matrix of double derivatives of the curve's equation) [26]. The restriction of points of inflection to flexes (see page 9) is straightforward [16]. The tangents of a singularity of the curve $f = 0$ can be found by translating the singularity to the origin. The equations of the tangents are the factors of the translated f's order form (the polynomial consisting of the terms of lowest degree) [26]. Finally, after the curve has been translated to projective space by homogenizing its equation to $f(x, y, z) = 0$ (where z is the homogenizing variable), the tangent of a flex P is $f_x(P)x + f_y(P)y + f_z(P)z = 0$ [26]. This completes our description of the convex decomposition of an algebraic curve.

6 Point location

The second key problem in the convex segment method of sorting is point location in the convex decomposition: given a point, identify the convex segment that contains it. This is an extension to the curved domain of the well-known problem of point location in a planar subdivision. We show how to locate points on both a convex segment and a general curve segment.

6.1 Point location I: On a convex segment

A decomposition is not very useful unless it is possible to locate points in it. In the case of sorting, point location is necessary to divide a set of points into convex segments for conquering. Since a convex segment is identified by its endpoints, finding the convex segment that contains a point is equivalent to finding the endpoints that bound this convex segment. Fortunately, this problem

is entirely analogous to finding the partners of a given endpoint as explained in Section 5.4, since both problems are instances of the more general question: "what are the two endpoints associated with a given point?" It is easy to locate a point in the proper cell, using well-known algorithms for point location in a planar subdivision [18,22].[8] If, as is often the case, a point lies in a cell with only one convex segment, then it is obvious what convex segment it belongs to. Otherwise, Theorem 4 and Lemma 4 can be used to locate a point on the proper convex segment.

Definition: A *connected component* of a curve is a maximal subset of the curve such that there exists a continuous path on the curve between any two points of the subset. For example, a hyperbola has two connected components. A type of connected component that requires special treatment is one that lies entirely inside of a cell, intersecting none of the walls (including artificial walls) of the cell partition. We call this a *nude* connected component since, unlike other connected components, it does not contain any endpoints of convex segments. Since it does not contain any singularities or flexes, a nude component is convex. It must also be closed (i.e., homeomorphic to a circle), otherwise it would intersect an artificial wall as it proceeded to infinity.

Theorem 4 *Consider a point x of curve F that lies in cell C and is not an endpoint of a convex segment.*[9] *Let* $S(x) = \{$*endpoints W in C |*

1. *x lies on the strict inside of W's tangent*
2. *W lies on the strict inside of x's tangent*
3. $\#\{P \in \overline{xW} \cap F : P \text{ faces } x\} = \#\{P \in \overline{xW} \cap F : P \text{ faces } W\}$
4. $\forall\, \alpha \in \overline{xW}$, $\#\{P \in \overline{x\alpha} \cap F : P \text{ faces } x\} \leq \#\{P \in \overline{x\alpha} \cap F : P \text{ faces } W\}$ }

If $S(x) = \emptyset$, then x lies on a nude connected component. Otherwise, let $S''(x) = \{\, W'' : W \in S(x) \,\}$, where W'' is the intersection of $x\bar{W}$ with the cell boundary. Let x_1 and x_2 be the two points of intersection of x's tangent with the cell boundary. x_1 and x_2 split the cell boundary into two halves, and every endpoint in $S''(x)$ lies on one of these halves. Let $S''_1, S''_2, \ldots, S''_p$ be the result of a sort of $S''(x)$ from x_1 to x_2. Then S_1 and S_p are partners and x lies on the convex segment $\widehat{S_1 S_p}$.

Proof: If x does not lie on a nude component, then $S(x) \neq \emptyset$, since it will contain the two endpoints of x's convex segment. (One can also quite easily establish the converse: if x lies on a

[8] Artificial boundaries are ignored when locating points in a cell: a point is considered to lie in an artificially closed cell C as long as it lies in the open cell associated with C.

[9] If x is an endpoint of a convex segment, then Theorem 3 can be used to determine x's partner in C, and thus its convex segment in C.

nude component, then $S(x) = \emptyset$.) The rest of the proof is similar to the proof of Theorem 3, and the interested reader is referred to [16]. ∎

Example 6.1 *In Figure 21(a), $S(x) = \emptyset$ and x lies on a nude component.*

Consider the cell of Figure 15 that contains the convex segments $\widehat{W_1W_2}$ and $\widehat{W_3W_4}$. W_1 does not satisfy condition (2) of $S(x)$ and W_2 does not satisfy condition (3). Thus, $S(x) = \{W_3, W_4\}$ and x must lie on $\widehat{W_3W_4}$.

Consider the cell partition of Figure 3. $S(P_1) = \{W_1, W_2, W_5, W_6\}$, which does not resolve the question of P_1's convex segment. Let x_1 and x_2 be the two points of intersection of P_1's tangent with the cell boundary. The sort of $S''(P_1)$ from x_1 to x_2 is W_1, W_6, W_5, W_2, so P_1 must lie on $\widehat{W_1W_2}$.

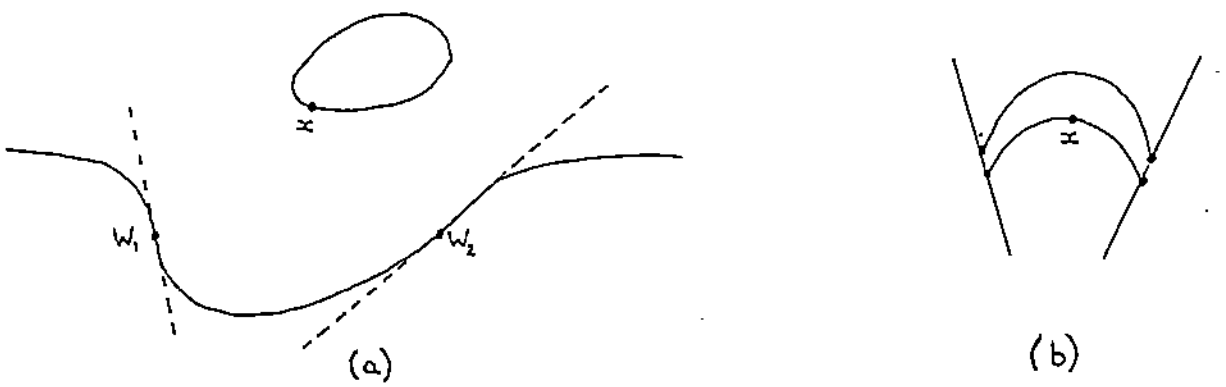

Figure 21: (a) x lies on a nude component (b) two overlapping segments

If there is only one nude component in a cell, then Theorem 4 can successfully locate a point on this convex segment. However, if there is more than one nude component in the cell, then the following lemma must be used to distinguish these nude components.

Lemma 4 *Let P and Q be points that lie on nude components of a curve and in the same cell. P and Q lie on the same nude component if and only if Q lies in $S(P)$, where $S()$ is as in Theorem 4.*

Proof: Let P and Q lie on nude components M and N, respectively. If $M = N$, then P and Q lie on the same convex segment, so $Q \in S(P)$ by Lemma 2. Suppose that $M \neq N$. Nude components do not intersect, since they do not contain any singularities. Therefore, there are only three cases

to consider: M lies inside N, N lies inside M, and neither lies inside the other. In all three cases, it is straightforward to show that Q violates one of the conditions of S(P). ∎

Point location can be made faster through two observations, both of which make use of the endpoint pairings already computed. The idea is to eliminate endpoints from $S(x)$ in Theorem 4 faster. First, as soon as the endpoint W is eliminated, W's partner can also be eliminated, since the two desired endpoints are partners. Second, by convexity, the curve segment between an endpoint W_1 and its partner W_2 lies on one side of $\overline{W_1W_2}$. Thus, if x does not lie on the appropriate side of $\overline{W_1W_2}$, then both W_1 and W_2 can be eliminated. These observations should be used along with conditions (1-2) to eliminate as many endpoints as possible from S(x) (in the best case, leaving only two). Conditions (3-4) should only be used when absolutely necessary, because they involve the expensive solution of an equation of degree n (where n is the degree of the curve F). Fortunately, the only time that conditions (3-4) will be needed to locate a point on a convex segment is for a point that lies on one of two overlapping convex segments in the same cell, as in Figure 21(b): x lies inside all four endpoint's tangents and all four endpoints lie inside x's tangent. Experience with algebraic curves (e.g. Lawrence's catalog of algebraic curves [19]), combined with experimental evidence, indicates that this situation is very rare: a wall of the cell partition will almost always separate overlapping parts of the curve. Therefore, a point can usually be located on a convex segment very cheaply.

This completes our description of techniques that are needed for sorting by the convex segment method. We digress for a moment to show how the theory that we have developed can be used to solve two important problems (although they are not needed for sorting): locating a point on an arbitrary segment and deciding whether two points lie on the same connected component.

6.2 Point location II: On an arbitrary segment

Once it is known how to locate a point on a convex segment of a curve's convex decomposition, it is straightforward to solve the more general problem of locating a point on an arbitrary segment of the curve. Recall that every endpoint of a convex segment in our (normalized) convex decomposition has exactly two partners. Therefore, every convex segment has a unique predecessor and successor, and it is trivial to order the convex segments. Consider a segment $\widehat{AB}$ of curve C and a point P on C. To decide if P lies on $\widehat{AB}$, we compute the convex segments of C's decomposition that contain P, A, and B (say C_p, C_a, and C_b, respectively). Then, P lies on $\widehat{AB}$ if and only if C_p lies in between C_a and C_b. If P lies on the same convex segment as A and/or B, then the decision requires more subtlety. For example, if P lies on the same convex segment $\widehat{EF}$ as A (but not B), then the decision

is made by sorting P, A, E, and F along $\widehat{EF}$, using Theorem 1: $P \in \widehat{AB}$ if and only if the order is E, P, A, F (resp., E, A, P, F) and $\widehat{AB}$ leaves A towards E (resp., F). (A method for deciding if $\widehat{AB}$ leaves A towards E or F is described in a footnote on page 7.) In short, point location on an arbitrary segment is easily reducible to point location on a convex segment.

6.3 Curves with many connected components

It should now be clear that the convex segment method can sort points on any algebraic curve. In particular, it can sort points that are strewn over several connected components of a curve, with no more difficulty than sorting points on a single component. This is another advantage of the convex segment method over the parameterization method, because it is not clear how the latter method could deal with points on several components, even if we allow nonrational parameterizations. Would each connected component have a separate parameterization? If so, how would the single equation of a curve produce several independent parameterizations? If not, how would one determine the range of parameter values that is associated with each connected component?

A very useful test for a curve with several components is whether two points lie on the same connected component. For example, with this capability it is reasonable to define an edge of a solid model as a particular connected component of a multi-component curve, since the test allows you to restrict intersections with the curve to this connected component. The following lemma shows that our decomposition of the curve into convex segments makes it simple to perform this test. (Lemma 4 can be used for points on nude components.)

Lemma 5 *Let P and Q be points of a curve, not both of which lie on a nude component. Let P and Q lie on convex segments $\widehat{AB}$ and $\widehat{CD}$, respectively.*[10] *P and Q lie on the same connected component if and only if $A \equiv C$, where $v \equiv w$ if and only if $\widehat{vw}$ is a convex segment of our cell partition or $v \equiv z$ and $w \equiv z$ for some z.*

Two other decompositions of an algebraic curve, Collins' cylindrical algebraic decomposition [11,4] and Canny's stratification [8], can also be used to separate a curve into connected components and thus decide whether two points lie on the same connected component.

[10] If P (resp., Q) lies on a nude component, then A and B (resp., C and D) are null symbols.

6.4 Broad comparison of methods

Let us compare the convex segment method of sorting with the others that were mentioned in Section 3. Like the brute-force tracing method, the convex segment method leaps from one point to another along the curve (viz., from an endpoint to its partner). However, its jumps are large while the tracing method's jumps must be very small. Moreover, once the partner of each convex segment endpoint of the cell partition has been computed (which can be done once and for all in a preprocessing step), each jump of the convex segment method can be done very quickly; whereas, the tracing method must grope for some time (by applying Newton's method) to find the destination of each jump. In short, the convex segment method makes large, bold jumps while the tracing method makes small, timid ones.

The convex segment method is similar to the parameterization method because they both reduce the sorting problem to an easier one. The parameterization method observes that the sorting of points on a line is simple and tries to unwind the curve into a line by parameterizing it. Rather than trying to reduce the entire problem, the convex segment method divides the problem up into many smaller ones (viz., the sorting of points on a convex segment). We shall see that the many small reductions of the convex segment method can be done more quickly than the single, large reduction of the parameterization method.

The convex segment method incorporates preprocessing, since the convex decomposition of a curve can be done at any time. As a result, the actual sorting is usually very efficient. One might consider the parameterization of a curve to be preprocessing, but the subsequent runtime steps (solving for the parameter value of each point) are usually more expensive than those for the convex segment method (following pointers, locating points, and sorting convex segments).

7 Complexity

In this section, we analyze the complexity of the convex segment method of sorting. We base our complexity analysis on the RAM model, where basic arithmetic operations are of unit cost [3].

7.1 Complexity of convex decomposition

Theorem 5 *A curve of order n (a curve whose defining polynomial is degree n) can be decomposed into convex segments in* $O(\alpha[n^2] + n^2\alpha[MAX * n] + n^6\alpha[n])$ *time, where* $\alpha[n]$ *is the time required to find the real roots of a univariate polynomial equation of degree n, and* MAX *is the maximum number of quadratic transformations that are necessary to decompose any singularity of the curve*

into simple points.[11]

Proof:

Computation of singularities, flexes. Consider the curve $f(x,y) = 0$ of order n. Its singularities are found by solving the simultaneous system of equations $\{f_x = 0, f_y = 0, f = 0\}$. One method is to use resultants [26]. The resultant of two polynomials with respect to the variable x_n is a polynomial whose roots are the projection onto the hyperplane $x_n = 0$ of the intersections of the two polynomials. Let X (resp., Y) be the real roots of the resultant of f_x and f_y with respect to y (resp., x), which is a univariate polynomial in x (resp., y) of degree $O(n^2)$. Since singularities at infinity are not of interest, those roots in X (resp., Y) that cause the terms of highest degree of $\{f_x = 0, f_y = 0\}$ to simultaneously vanish are not of interest. (The terms of highest degree of a polynomial are intimately related to its solutions at infinity, since they dominate the polynomial as solutions get large.) Therefore, before computing the roots of the resultant, the GCD of the leading term polynomials of f_x and f_y is computed and divided out of the resultant, all in $O(n\ log^2\ n)$ time [3]. Now X (resp., Y) is the collection of abscissae (resp., ordinates) of the finite-solution set of $\{f_x = 0, f_y = 0\}$. X (and Y) can be computed in $O(n^4\ log^3\ n + \alpha[n^2])$ time, since the resultant of a pair of polynomials of degree at most n in r variables can be computed in $O(n^{2r}\ log^3\ n)$ time [2]. The singularities of the curve are $\{\ (x,y)\ :\ x \in X,\ y \in Y$ and $f(x,y) = f_x(x,y) = f_y(x,y) = 0\ \}$. This pairwise substitution takes $O(n^6)$ time, since X and Y are each of size $O(n^2)$ and the evaluation of an equation of degree n requires $O(n^2)$ time. Hence, all singularities of the curve can be computed in $O(\alpha[n^2] + n^6)$ time. With similar techniques, the flexes can also be computed in $O(\alpha[n^2] + n^6)$ time.

Computation of their tangents. Recall that the tangents at a singularity (a,b) are computed by translating the singularity to the origin and factoring the polynomial consisting of the terms of lowest degree of the translated f(x,y) into linear factors. (For example, the lines $x - y = 0$ and $x + y = 0$ are the tangents of the curve $x^3 - x^2 + y^2 = 0$.) A translation is simply a linear substitution $x_t = x - a,\ y_t = y - b$, which takes $O(n^4)$ time for a bivariate equation of order n. The factorization of a homogeneous bivariate polynomial is equivalent to the solution of a univariate polynomial. Therefore, the computation of the tangents at a singularity requires $O(n^4 + \alpha[n])$ time. A curve of order n has at most $O(n^2)$ singularities [26], so all of the tangents at singularities can be computed in $O(n^6 + n^2\alpha[n])$ time. The computation of the tangent at a flex is easier, only involving the $O(n^2)$ operation of bivariate (or homogeneous trivariate) polynomial evaluation (Section 5.5). A curve of

[11] MAX is 1 if each singularity has distinct tangents, and MAX will usually be 1 or 2 in geometric modeling applications.

order n also has at most $O(n^2)$ flexes [26], so all of the tangents at flexes can be computed in $O(n^4)$ time.

Computation of intersections of singularity/flex tangents with curve. The intersections of the singularity/flex tangents with the curve are needed to create the convex decomposition. Consider the number of tangents. There are at most $O(n^2)$ tangents at flexes. A curve of order n has at most $\frac{(n-1)(n-2)}{2}$ double points, where a singularity of multiplicity t counts as $\frac{t(t-1)}{2}$ double points and has $O(t)$ tangents [26]. Consequently, there are $t/\frac{t(t-1)}{2} < 2$ tangents per double point, or at most $O(n^2)$ singularity tangents. The intersection of a tangent with the curve involves a linear substitution and a solution of the resulting polynomial, thus $O(n^4 + \alpha[n])$ time or $O(n^6 + n^2\alpha[n])$ for all tangents. Note that the $O(n^2)$ tangents generate $O(n^3)$ endpoints on the curve, since each tangent intersects the curve in at most n points (Bezout's Theorem).

Refinement of singularities and infinite segments. A singularity of multiplicity t is refined into $O(2t)$ endpoints, meaning $2t/\frac{t(t-1)}{2} \leq 4$ refined endpoints per double point, or a total of $O(n^2)$ refined endpoints at singularities. Thus, the number of endpoints of convex segments (and the number of convex segments) remains $O(n^3)$ after refinement. Consider the time that is required to refine the singularities. Each singularity is translated to the origin and subjected to quadratic transformations (perhaps translating the singularity back to the origin after certain quadratic transformations). $O(n^2)$ quadratic transformations are sufficient to reduce all of the singularities to simple points, since the singularities of a curve of order n account in total for $O(n^2)$ double points and the application of each quadratic transformation removes at least one double point, in a global amortized counting [1]. We have seen that the translation of a curve requires $O(n^4)$ time, amounting to a total $O(n^6)$ translation time. Each quadratic substitution $x = x_1$, $y = x_1y_1$ takes $O(n^2)$ time (there are $O(n^2)$ terms in the original equation of the curve). Therefore, all of the quadratic transformations take $O(n^4)$ time.

During the reduction of a singularity to simple points, each quadratic transformation can increase the degree of the curve's equation, since x^iy^j becomes $x^i(x^{j-d}y^j) = x^{i+j-d}y^j$, where d is the multiplicity of the singularity.[12] In other words, the degree of the polynomial can increase by $O(j)$, where j is the highest degree of y in any term of the polynomial undergoing quadratic transformation. Since $j = n$ for the polynomial of the original curve and the y-degree of every term remains invariant under quadratic transformation (and does not increase under translation

[12]It might appear that x^iy^j should become $x^i(x^iy^j)$. However, redundant factors must be removed from the polynomial. For example, $x^2 - y^3 = 0$ becomes $1 - xy^3 = 0$, not $x^2 - x^3y^3 = 0$. The equation of a curve with a singularity of multiplicity d at the origin has no terms of degree less than d, so a factor of x^d can always be removed.

of the curve either), the degree of the polynomial can only increase by $O(n)$ with each quadratic transformation. Therefore, by the end of the reduction of a singularity to simple points, the curve's equation can be of degree $O(MAX * n)$.

Finally, after a quadratic transformation where the multiplicity of the singularity drops, one computes the intersections of the new curve of order i with the y-axis, which takes $\alpha[i]$ time. Again, since this is computed after at most $O(n^2)$ quadratic transformations, the total time taken by all of the intersection computations is at most $O(n^2\alpha[MAX * n])$ time. We conclude that a (pessimistic) bound on the time for refining the convex segment endpoints at singularities is $O(n^6 + n^2\alpha[MAX * n])$. There are at most two infinite segments, which are comparatively simple to refine.

Pairing endpoints. Consider the time required to compute the partners of the $O(n^3)$ endpoints. The dominating expense is the computation of the set $R(W_1)$ of Theorem 3 for each endpoint W_1. It takes $O(k\alpha[n])$ time to compute $R(W_1)$ for an endpoint in a cell with k endpoints, $O(k^2\alpha[n])$ time to compute $R(W_1)$ for every endpoint in a cell with k endpoints, and $O(\sum k_i^2\alpha[n])$ time to compute $R(W_1)$ for every endpoint in every cell, where k_i is the number of endpoints in cell C_i and the sum is over all cells C_i. Since $\sum k_i = O(n^3)$, $O(\sum k_i^2\alpha[n]) = O(n^6\alpha[n])$. Therefore, partner computation takes $O(n^6\alpha[n])$ time. ∎

It must be emphasized that the n of the above analysis is the order of the curve. This makes the analysis fundamentally different from those that we are familiar with, such as $O(n \log n)$ for sorting numbers (where n is the *number of points*) or $O(n \log\log n)$ for triangulating a simple polygon (where n is the *number of edges* of the polygon). (For example, in the above analysis, n is the constant 1 for all polygons.) As a result, the complexity of an operation such as the convex decomposition of an algebraic curve can be misleading, since it is very easy (although wrong) to compare it with familiar complexities of discrete (rather than continuous) algorithms such as number sorting or polygon manipulation.

It should also be noted that the above analysis is pessimistic. The worst case time will be reached only by the most pathological curves: the time to decompose curves that arise in practice in geometric modeling is much more reasonable. For example, a typical endpoint will lie on the boundary of a single-segment cell and its partner will be computed in $O(1)$, not $O(k\alpha[n])$, time. This observation has been borne out in practice, with the testing of the algorithms on various curves (see Section 8). The efficiency will be even further improved by the fact that the singularities and flexes, which are important to other geometric algorithms, may already be available in many cases.

7.2 Complexity of sorting

We now consider the complexity of sorting points along a curve after its convex decomposition is available. This sorting is usually very efficient, because the traversal of a curve by convex segments has been reduced to the traversal of a doubly linked list, and it is usually simple to find the points on each convex segment. Once again, the following worst-case analysis is unrealistically pessimistic for geometric modeling applications.

Theorem 6 *After the curve has been decomposed into convex segments, m points on a plane algebraic curve of order n can be sorted by the convex segment method in $O(mn^3\alpha[n] + m \log m)$ time. If the curve does not have overlapping segments (see page 28), then m points can be sorted in $O(mn^3 + m \log m)$ time.*

Proof: The dominating expense of sorting is to locate every point on a convex segment, since the convex segments are already implicitly sorted (by endpoint pairing) and the sorting of points along a convex segment is simple (by Theorem 1, it is equivalent to the $O(k \log k)$ operation of finding and sorting a set of angles). A point can easily be located in the proper cell of the cell partition. A vector of size $O(n^2)$ is associated with each of the m points and each cell: this vector specifies the side (inside or outside) of each singularity/flex tangent that the point or cell lies on. A point lies in a cell if and only if their two vectors match.[13] Therefore, the only potentially challenging step is locating the convex segment in the cell that contains the point. In the worst case, it requires $O(k\alpha[n])$ time to compute the set $S(x)$ of Theorem 4 for a point in a cell with k endpoints, since the intersection of line segments with the curve is required. There are $O(n^3)$ endpoints, so point location requires $O(n^3\alpha[n])$ time per point and $O(mn^3\alpha[n])$ time for all points.[14] After adding $O(m \log m)$ time for sorting the points along the convex segments, the convex segment method requires worst-case $O(mn^3\alpha[n] + m \log m)$ time to sort m points by traversing $O(n^3)$ convex segments. If the curve does not have overlapping segments, then curve-line intersection can be avoided in the computation of the set $S(x)$, thus dropping the $\alpha[n]$ factor. ∎

[13] The vector of a cell need not, and will not, be complete. Only the entries for the cell's walls are necessary.

[14] Observe the worst-case pessimism of this analysis. It is unlikely that there are $O(n^3)$ real endpoints, since many of the n intersections of a singularity/flex tangent with the curve will be complex. It is extremely unlikely that all of these endpoints are in the same cell and that none of these endpoints would be eliminated by the cheap $O(1)$ conditions of Theorem 4.

8 Execution times

This section presents execution times for the sorting of some representative curves by the convex segment and parameterization methods. These empirical results are a good complement to the complexity analysis of Section 7, since they capture the expected case, rather than the worst case, behaviour of the methods. The source code was written in Common Lisp and execution times are in seconds on a Symbolics Lisp Machine, not including time for disk faults and garbage collection. Times for the convex segment method are the average of twelve trials, while times for the parameterization method are the average of three trials. Preprocessing time is the time required to create the cell partition and find the partners of all of the endpoints. Five curves are examined: two rational cubic and three non-rational quartic.

We do not consider the time required to find a parameterization of the curve or to find the flexes and singularities of the curve. Each of these computations is a preprocessing step that is entirely independent of sorting, and often the parameterization, singularities, and flexes of a curve will already be available. Moreover, the computation of a curve's parameterization is of approximately the same complexity as the computation of a curve's singularities and flexes, so our comparison of sorting methods should not be biased.

The first example illustrates the superiority of the convex segment method: even when the preprocessing time is added to the sorting time, it is more efficient. Also notice that the rate of growth of the convex segment method is much smaller. The inferiority of the tracing method (see end of Section 3) is obvious from this example, and we do not consider it further.

Example 8.1 *A semi-cubical parabola*
Equation of the curve: $27y^2 - 2x^3 = 0$
Preprocessing time: 0.27 seconds
Parameterization: $\{x(t) = 6t^2,\ y(t) = 4t^3 \ :\ t \in (-\infty, +\infty)\}$

number of sortpoints	*1*	*2*	*6*
convex segment	*.01*	*.03*	*.03*
convex segment + preprocessing	*.28*	*.30*	*.30*
parameterization	*.47*	*.63*	*1.04*
tracing	*3.14*	*2.89*	*4.77*

The second example illustrates the tradeoff between a very fast sort that requires preprocessing (convex segment method) and a moderately fast sort that does not require preprocessing (parameterization method).

Example 8.2 *Folium of Descartes*

Equation of the curve: $x^3 + y^3 - 15xy = 0$

Preprocessing time: 2.81 seconds

Parameterization: $\{x(t) = \frac{15t}{1+t^3},\ y(t) = \frac{15t^2}{1+t^3} \ :\ t \in (-\infty, +\infty)\}$

number of sortpoints	*1*	*2*	*5*	*9*
convex segment	*0.01*	*0.01*	*0.05*	*0.04*
convex segment + preprocessing	*2.82*	*2.82*	*2.85*	*2.85*
parameterization	*1.01*	*1.07*	*1.76*	*3.17*

The remaining three curves are non-rational, so they are only sorted with the convex segment method.

Example 8.3 *Devil's Curve (with several connected components)*

Equation of the curve: $y^4 - 4y^2 - x^4 + 9x^2 = 0$

Preprocessing time: 2.20 seconds

number of sortpoints	*1*	*4*	*7*
convex segment	*0.09*	*0.09*	*0.10*
convex segment + preprocessing	*2.29*	*2.29*	*2.30*

Example 8.4 *Limacon*

Equation of the curve: $x^4 + y^4 + 2x^2y^2 - 12x^3 - 12xy^2 + 27x^2 - 9y^2 = 0$

Preprocessing time: 4.62 seconds

number of sortpoints	*2*	*5*	*8*
convex segment	*.09*	*.30*	*.55*
convex segment + preprocessing	*4.70*	*4.92*	*5.17*

Example 8.5 *Cassinian oval*

Equation of the curve: $x^4 + y^4 + 2x^2y^2 + 50y^2 - 50x^2 - 671 = 0$

Preprocessing time: 5.36 seconds

number of sortpoints	*2*	*4*	*6*
convex segment	*.14*	*.17*	*.19*
convex segment + preprocessing	*5.50*	*5.53*	*5.55*

9 Comparison of sorting methods

In this section, we consider the relative merits of the parameterization and convex segment methods of sorting. Certain curves cannot, or should not, be sorted by the parameterization method: curves that do not possess a rational parameterization and curves for which a rational parameterization cannot be efficiently obtained. Therefore, the convex segment method is often the only viable way to sort points along a curve.

For those curves that can be sorted in either way, the convex segment method is generally far more efficient than the parameterization method at the actual sorting of the points. However, the parameterization method does not have the expense of preprocessing that the convex segment method does. Therefore, when only a few points need to be sorted (over the entire lifetime of the curve) and the sorting of these points must be done soon after the definition of the (rational) curve, the parameterization method will usually be the method of choice. (However, we have seen an example where the convex segment method is superior to parameterization even when we include preprocessing time.) The expense of preprocessing will be warranted whenever sorting time is a valuable resource, as in a real-time application, or when the number of points that will be sorted is large. The convex segment method will also be preferable when the curve is defined long before it is ever sorted (as with a complex solid model that requires several days, weeks, or even months to develop), since the preprocessing can be done at any time that processing time becomes available before the sort. We conclude that the convex segment method is an effective new method for sorting points along an algebraic curve, and that in many situations it is either the only or the best method.

10 Conclusions

We have developed a new method of sorting points along an algebraic curve that is superior to the conventional methods of sorting. Many curves that could not be sorted, or that could only be sorted slowly, can now be sorted efficiently. The development of our new method has also illustrated how an algebraic curve can be decomposed into convex segments, how to locate points on segments of algebraic curves, and how to decide whether two points lie on the same connected component.

This work is one of the first solutions of a computational geometry problem that is applicable to curves of arbitrary degree. Methods are usually restricted to curves/surfaces of some specific or bounded degree, such as polygons/polyhedra or quadrics. The creation and manipulation of curves and surfaces is of major importance to geometric modeling. A sophisticated geometric modeling system should offer a rich collection of tools to aid this manipulation. This paper has been an examination of one of these tools. The progress of geometric modeling depends upon the development of more tools and upon the extension of more computational geometry algorithms from polygons to curves and surfaces of higher degree.

11 Acknowledgements

This work formed part of the thesis of J. Johnstone, who is grateful for the guidance of his advisor, John Hopcroft.

References

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

[2] C. Bajaj and A. Royappa, *Note on an Efficient Implementation of Sylvester's Resultant for Multivariate Polynomials*, Technical Report CSD-TR-718, Dept. of Computer Science, Purdue University, 1987.

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

[4] D. S. Arnon, *Topologically reliable display of algebraic curves*, J. Computer Graphics, 17(1983), pp. 219–227.

[5] T. Asano, T. Asano, and H. Imai, *Partitioning a polygonal region into trapezoids*, Res. Mem. RMI84-03, Dept. Math. Eng. and Instrumentation Physics, Univ. of Tokyo, 1984.

[6] D. Avis and G. T. Touissant, *An efficient algorithm for decomposing a polygon into star-shaped components*, Pattern Recognition, 13(1981), pp. 395-398.

[7] C. Bajaj, C. Hoffmann, J. Hopcroft, and B. Lynch, *Tracing surface intersections*, Computer Aided Geometric Design, 1988, to appear.

[8] J. F. Canny, *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1987.

[9] B. Chazelle and D. P. Dobkin, (1985). *Optimal convex decompositions*, in Computational Geometry, G. T. Toussaint, ed., North-Holland, New York, 1985, pp. 63-133.

[10] B. Chazelle and J. Incerpi, *Triangulation and shape-complexity*, ACM Trans. on Graphics, 3(1984), pp. 135-152.

[11] G. E. Collins, *Quantifier elimination for real closed fields by cylindrical algebraic decomposition*, Proc. 2nd GI Conference on Automata Theory and Formal Languages, Lecture Notes in Computer Science 35, Springer, Berlin, 1975, pp. 134–183.

[12] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[13] M. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan, *Triangulating a simple polygon*, Info. Proc. Lett., 7(1978), pp. 175–80.

[14] S. Hertel and K. Mehlhorn, *Fast triangulation of simple polygons*, Proc. FCT'83, Borgholm, Lecture Notes in Computer Science, Springer, Berlin, 1983, pp. 207–218.

[15] C. Hoffmann and J. Hopcroft, *The potential method for blending surfaces and corners*, in Geometric Modeling: Algorithms and New Trends, G. Farin, ed., SIAM, Philadelphia, 1987, pp. 347–365.

[16] J. Johnstone, *The sorting of points along an algebraic curve*, Technical Report 87-841, Ph.D. thesis, Dept. of Computer Science, Cornell University, Ithaca, NY, 1987.

[17] J. M. Keil, *Decomposing polygons into simpler components*, Technical Report 163/83, Ph.D. thesis, Dept. of Computer Science, Univ. of Toronto, 1983.

[18] D. Kirkpatrick, *Optimal search in planar subdivisions*, this Journal, 12(1983), pp. 28–35.

[19] J. D. Lawrence, *A Catalog of Special Plane Curves*, Dover, New York, 1972.

[20] W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, 1979.

[21] J. O'Rourke, *The complexity of computing minimum convex covers for polygons*, Proc. 20th Annual Allerton Conf. on Comm. Control and Comput., 1982, pp. 75–84.

[22] F. Preparata and M. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[23] J. R. Sack, *An $O(n \log n)$ algorithm for decomposing simple rectilinear polygons into convex quadrilaterals*. Proc. 20th Annual Allerton Conf. on Comm. Control and Comput., 1982, pp. 64–74.

[24] R. E. Tarjan and C. J. Van Wyk, *An $O(n \log\log n)$-time algorithm for triangulating a simple polygon*, this Journal, 17(1988), pp. 143–178.

[25] S. B. Tor and A. E. Middleditch, *Convex decomposition of simple polygons*, ACM Trans. on Graphics, 3(1984), pp. 244–265.

[26] R. J. Walker, *Algebraic Curves*, Springer-Verlag, New York, 1950.