Proceedings of the Fifth ACM Symposium on Computational Geometry (SCG),

Saarbrucken, West Germany, June 5-7, 1989, pp. 197–207.

Calculating Approximate Curve Arrangements Using Rounded Arithmetic *

Victor Milenkovic
Harvard University

We present here an algorithm for the curve arrangement problem: determine how a set of planar curves subdivides the plane. This algorithm uses rounded arithmetic and generates an approximate result. It can be applied to a broad class of planar curves, and it is based on a new definition of approximate curve arrangements. This result is an important step towards the creation of practical computer programs for reasoning about algebraic curves of high degree.

1 Introduction

In the area of computational geometry, as in other areas involving numerical computation, approximate solutions to problems are much cheaper but much less reliable than exact solutions. Consider as an example the *curve arrangement problem*: determine how a set of algebraic curves in the plane subdivides the plane into vertices, curved edges, and curved-boundary polygonal regions. The vertex locations are irrational quantities, but an *exact* curve arrangement algorithm can calculate them to any desired degree of accuracy and can correctly calculate the topology of the arrangement. Unfortunately, the cost of exact algorithms grows rapidly with the degree. If we implement the exact algorithm using rounded arithmetic and we allow the implementation to round nearby vertices to the same location, then the cost depends much less strongly on the degree. However, such an implementation lacks a theoretical foundation and will miss roots and generate inconsistent topologies. Still, approximate numerical computations are so much cheaper than exact computations that it is worthwhile to investigate the possibility of a theoretically well-founded approximate arrangement algorithm, even if that algorithm has greater symbolic complexity than the exact solution.

We describe here a rounded arithmetic, approximate solution to the curve arrangement problem for a broad set of curves. We make only three assumptions.

- Each curve is a convex (or concave) monotonic function.
- Using rounded arithmetic, it is possible to calculate approximately the distance of an arbitrary point to a curve.
- It is also possible to calculate approximately the intersection of the curve with an arbitrary line.

These assumptions will be stated more formally later, but it is clear that for any given set of algebraic curves, we can satisfy the first assumption by cutting the curve at horizontal and vertical tangents and points of inflection. We can perform the two types of calculation using standard approximation methods. If we assume that the distance calculation has unit cost and that the cost of calculating the line-curve intersection is L, then the algorithm has complexity,

$$\mathrm{O}(VL \log N + VC^2 L + VC^2 \log C),$$

where C is the number of curves, V is the number of intersections among the curves in the exact solution, and N is the number of bits of accuracy in the solution. In [14] we show that quadratic curves can be made to satisfy the three assumptions. For this case, we can implement the algorithm on any architecture whose floating point has at least $N + 17$ bits in the mantissa. In general we expect the number of bits required to be $N + 2 \log D + 17$ for curves of degree D.

The approximate curve arrangement algorithm treats the curves as "black boxes". In order to compare the approximate algorithm to the exact algorithm, we must make a few more assumptions. First we assume that the polynomial equations which define the curves have N-bit coefficients (input accuracy same as output accuracy). Furthermore, assume that the polynomials have degree D where $\log D$ is small in comparison to N. The number of vertices V in the exact case is about $D^2C^2/2$: D^2 intersections for every pair of curves. Assume that N-bit arithmetic operations have unit cost. Standard exact techniques using resultants appear to require at least D^7C^2 arithmetic operations. In comparison, under the assumptions of this paragraph, the approximate algorithm has complexity,

$$\mathrm{O}(D^4C^2 \log^2 N + D^4C^4 \log N + D^2C^4 \log C).$$

For fixed degree, the approximate algorithm seems to have no advantage, but for a fixed number of curves and increasing degree, the approximate solution is clearly better. Furthermore, we expect the cost of the optimal approximate

*Part of this research was done at Carnegie Mellon University and was supported by an IBM Manufacturing Research Fellowship. Authors' address: Aiken Computation Lab, Harvard University, Cambridge MA 02138.

solution to be

$$O(D^4C^2\log^2 N + D^2C^2\log C),$$

at least in the typical case. Approximate methods will almost certainly be the best route to a reliable and efficient solution to the curve arrangement problem.

The next three sections describe the approximate curve arrangement algorithm. Section 2 gives the high level curve arrangement algorithm in detail. This section includes a formal definition of the types of curves and the primitive operations required by the approximate curve arrangement algorithm, and it formally defines the output, an *approximate cylindrical decomposition* of the plane. Section 3 analyses the computational complexity of the algorithm and compares it with the exact algorithm. Finally, Section 4 gives an algorithm for a lower level primitive required by the high level algorithm. This primitive generates all intersections between a given pair of curves.

1.1 Background

In order to qualify as *robust* under the definitions in the paper, a geometric algorithm must be implementable using rounded ($N{+}c$)-bit arithmetic where N is the number of bits of accuracy in the output and c is constant or perhaps logarithmic in N. The implementation is not allowed to simulate high precision arithmetic. The author has also created line arrangement and plane arrangement algorithms that satisfy this definition [14].

For a discussion of the more general problem of improving the reliability geometric computations using rounded arithmetic, see [8, 9]. Problems addressed include polygon [19, 13] and polyhedron [10, 11, 20] modeling and the calculation of convex hulls [17] and other problems in plane geometry [15]. Karasick [11] has the most practical result to date, an ultra-reliable polyhedral modeling system that is provably *safe*: it will not generate a topologically inconsistent result. Greene and Yao [7] have addressed the problem of rounding the output of an exact algorithms, and rounding can also be accomplished using the techniques in [14] and [15].

The problem of avoiding singular cases (unexpected incidences) has been addressed by Edelsbrunner [6] and Yap [23]. These results are currently only applicable to exact implementations which have high cost. Sugihara [21] estimates that at least 80 times the input precision would be required to model natural quadric surfaces using exact arithmetic. However, Karasick, Lieber and Nackman [12] have recently demonstrated an exact Delaunay triangulation algorithm with empirical cost well below the worst case.

Exact calculations arrangements of algebraic curves or surfaces are based on the theory and techniques developed by Tarski [22] and Collins [4]. Based on these techniques, Schwartz and Sharir [18] described a method for computing all topological information about a set of algebraic surfaces in d dimensions. Arnon, Collins, and McCallum have developed and implemented exact algorithms for computing cylindrical algebraic decompositions in two [2, 3] and three [1] dimensions (see Definition 5 below). There is a considerable body of work in the area of computing the topology generated by a set of algebraic surfaces, but these are concerned with bounding the complexity as a function of dimension. We will only consider the two-dimensional problem here.

2 High Level Robust Curve Arrangement Algorithm

This section describes the heart of the robust curve arrangement algorithm. This portion is the main contribution of the research. The lower level portion in Section 4 consists of a set of numerical procedures, but the topological information is generated solely by the high level algorithm in this section.

2.1 General Definitions

Definition 1 *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two points in the plane. Define,*

$$\mathrm{R}(\mathbf{AB}) = [\mathbf{A}_x, \mathbf{B}_x] \times [\mathbf{A}_y, \mathbf{B}_y].$$

The rectangle $\mathrm{R}(\mathbf{AB})$ *has segment* $\mathbf{AB}$ *as one of its diagonals, and it is a* coordinate rectangle*: its sides are aligned with the* x *and* y *axes.*

Definition 2 *Given two real numbers* a *and* b*, we say that the sign of* a agrees *with the sign of* b *if,*

$$b \left\{ \begin{array}{c} = \\ > \\ < \end{array} \right\} 0 \textit{ implies } a \left\{ \begin{array}{c} = \\ \geq \\ \leq \end{array} \right\} 0.$$

Definition 3 (Monotonic Curve Segment) *A* monotonic curve segment *is a quadruple* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$ *such that:*

- $\mathbf{A}, \mathbf{B} \in \Re^2$ *are two points in the plane satisfying* $\mathbf{A}_x \leq \mathbf{B}_x$*;*
- $\gamma : [0,1] \rightarrow \Re^2$ *is a continuous, differentiable curve satisfying,* $\gamma(0) = \mathbf{A}$, $\gamma(1) = \mathbf{B}$, *and* $\forall t \in [0,1]$: $\gamma'_x(t) \geq 0$ *and* $\mathrm{sign}(\gamma'_y(t))$ *agrees with* $\mathrm{sign}(\mathbf{B}_y - \mathbf{A}_y)$*;*
- $\mathrm{F} : \mathrm{R}(\mathbf{AB}) \rightarrow \Re$ *is a differentiable function satisfying,*
  - $\forall t \in [0,1]$: $\mathrm{F}(\gamma(t)) = 0$,
  - $\forall \mathbf{P} \in \mathrm{R}(\mathbf{AB})$: $\mathrm{sign}\left(\frac{\partial \mathrm{F}}{\partial x}(\mathbf{P})\right)$ *agrees with* $\mathrm{sign}(\mathbf{A}_y - \mathbf{B}_y)$ *and* $\frac{\partial \mathrm{F}}{\partial y}(\mathbf{P}) \geq 0$.

$$\textit{If } \mathbf{A}_y \left\{ \begin{array}{c} < \\ > \\ = \end{array} \right\} \mathbf{B}_y,\ \gamma \textit{ is } \left\{ \begin{array}{c} \text{rising} \\ \text{falling} \\ \text{horizontal} \end{array} \right\}.$$

For example, γ could be any monotonic curve and $\mathrm{F}(x,y)$ could be the signed distance from γ. However, $\mathrm{F}(x,y)$ can in fact be any function acting “like” a signed distance (monotonic along horizontal and vertical lines).

For any $a \in [\mathbf{A}_x, \mathbf{B}_x]$, the vertical line $x = a$ will intersect the curve segment $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$. The intersection will either be a single point $\langle a, y(a) \rangle$ or a vertical segment $\langle a, y_{\mathrm{in}}(a) \rangle \langle a, y_{\mathrm{out}}(a) \rangle$. In the former case, we will define $y_{\mathrm{in}}(a) = y_{\mathrm{out}}(a) = y(a)$. In the latter case we assign the names $y_{\mathrm{in}}(a)$ and $y_{\mathrm{out}}(a)$ so that $\mathrm{sign}(y_{\mathrm{in}}(a) - y_{\mathrm{out}}(a))$ agrees with $\mathrm{sign}(\mathbf{A}_y - \mathbf{B}_y)$.

Definition 4 (Vertical Ordering) *Given a set of curve segments,*

$$\{\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i \rangle | i = 1, 2, \ldots, C\},$$

the vertical ordering *at* $x = a$ *is the total ordering on the set,*

$$\{y_{in,i}, y_{out,i} | i = 1, 2, \ldots, C\}.$$

In general $y_{\mathrm{in},i}$ and $y_{\mathrm{out},i}$ will be irrational, so we can only know their order, not their exact values.

Definition 5 (Cylindrical Decomposition) *The* cylindrical decomposition *of the plane generated by a set of curves*

$$\{\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i\rangle | i = 1, 2, \ldots, C\}$$

is a set of values $\{a_1, a_2, \ldots, a_K\}$ *such that,*

- $a_1 < a_2 < \cdots < a_K$;
- *for* $k = 1, 2, \ldots, K-1$, *the vertical ordering of the curves along the the line* $x = a$ *is constant on* $a \in (a_k, a_{k+1})$.

This is a variant of the standard definition of cylindrical decomposition [4]. Strictly speaking, the cylindrical decomposition is the arrangement of the set of curves segments union the set of vertical passing through the endpoints of the curve segments. Using this result we can find the arrangement (within any bounded region of the plane) of any set of unbounded monotonic curves. We can only know the values $\{a_1, a_2, \ldots, a_K\}$ implicitly.

2.2 Definitions for an Approximate Domain

The definitions in the preceding section used the concept of implicit values but were based on the assumption that these values could be calculated to any degree of accuracy if desired. This section provides definitions for the domain in which we can only compute values approximately.

Definition 6 (Coordinate Evaluation Function) *An* α-accurate coordinate evaluation function,

$$\text{coord-eval}(\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B}\rangle, x = a), \ (\mathbf{A}_x \le a \le \mathbf{B}_x),$$

takes a curve segment and a vertical line and returns a point, $\langle a, y(a)_{\mathrm{F}}\rangle$, *satisfying* $|\mathrm{F}(a, y(a)_{\mathrm{F}})| \le \alpha$.

Definition 7 *Let* $\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i\rangle$ *and* $\langle \mathrm{F}_j, \gamma_j, \mathbf{A}_j, \mathbf{B}_j\rangle$ *be two monotonic curve segments, and let* $\mathbf{R} \subseteq \mathrm{R}(\mathbf{A}_i\mathbf{B}_i) \cap \mathrm{R}(\mathbf{A}_j\mathbf{B}_j)$ *be some rectangle. We say,* γ_i *is* $\left\{\begin{array}{c}\text{ABOVE}\\ \text{BELOW}\\ \text{NEAR}\end{array}\right\}$ γ_j *if*

$$\forall \mathbf{P} \in \mathbf{R}: \left\{\begin{array}{c}\mathrm{F}_i(\mathbf{P}) \ge -\alpha\\ \mathrm{F}_i(\mathbf{P}) \le \alpha\\ |\mathrm{F}_i(\mathbf{P})| \le \alpha\end{array}\right\} \textit{ implies } \left\{\begin{array}{c}\mathrm{F}_j(\mathbf{P}) > \alpha\\ \mathrm{F}_j(\mathbf{P}) < -\alpha\\ |\mathrm{F}_j(\mathbf{P})| < \beta\end{array}\right\}.$$

Note that ABOVE and BELOW are symmetric relations but NEAR is not. It is definitely possible that none of these relations holds. The values α and β are positive real numbers which are parameters to the high level arrangement algorithm. Section 2.3 discusses these constants further.

Definition 8 (Complete Set of Approximate Intersections) *Given two monotonic curve segments* $\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i\rangle$ *and* $\langle \mathrm{F}_j, \gamma_j, \mathbf{A}_j, \mathbf{B}_j\rangle$, *a* complete set of approximate intersections *of* γ_i *with* γ_j *is a list of points,*

$$\mathbf{I}_{i,j,1}, \mathbf{I}_{i,j,2}, \ldots, \mathbf{I}_{i,j,K_{ij}} \in \mathrm{R}(\mathbf{A}_i\mathbf{B}_i),$$

such that,

$$\mathbf{I}_{i,j,1} = \mathbf{A}_i, \qquad \mathbf{I}_{i,j,K_{ij}} = \mathbf{B}_i,$$

and for $k = 1, 2, \ldots, K_{ij} - 1$,

- $\mathbf{I}_{i,j,k,x} \le \mathbf{I}_{i,j,k+1,x}$ *and* $\mathrm{sign}(\mathbf{I}_{i,j,k,y} - \mathbf{I}_{i,j,k+1,y})$ *agrees with* $\mathrm{sign}(\mathbf{A}_{i,y} - \mathbf{B}_{i,y})$,
- $|\mathrm{F}_i(\mathbf{I}_{i,j,k})| \le \alpha$,

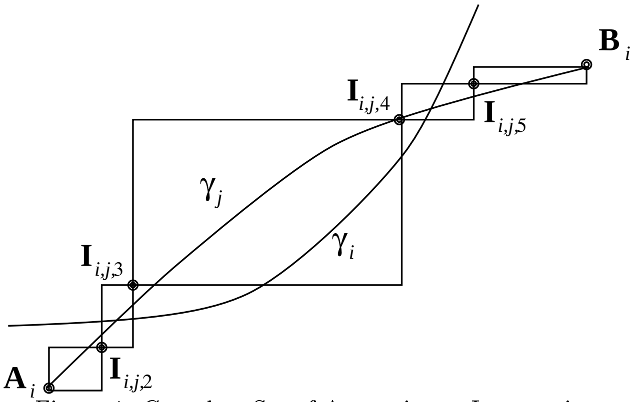

Figure 1: Complete Set of Approximate Intersections

- *if* $\mathbf{I}_{i,j,k}\mathbf{I}_{i,j,k+1}$ *is not horizontal or vertical,* γ_i *is either ABOVE, BELOW, or NEAR* γ_j *inside*

$$\mathrm{R}(\mathbf{I}_{i,j,k}\mathbf{I}_{i,j,k+1}) \cap \mathrm{R}(\mathbf{A}_j\mathbf{B}_j).$$

(See Figure 1.)

In practice, the number K_{ij} is chosen to be the minimum number such that this third condition is met. It is important to understand why this is the right way to define the set of intersections between γ_i and γ_j in a finite precision domain. In the exact, infinite precision domain, the two curves intersect in a discrete set of points. In the finite precision domain, however, the best we can say is that γ_i starts to one side of γ_j, approaches it and stays near to it for some distance (and crosses it at some unknown point), then stays on the other side of γ_j for a while, approaches it again, and so on. Thus we break up γ_i into curve segments $\mathbf{I}_{i,j,k}\mathbf{I}_{i,j,k+1}$ (approximate segments, actually, since $\mathbf{I}_{i,j,k}$ and $\mathbf{I}_{i,j,k+1}$ do not exactly lie on γ_i) such that each segment is definitely to one side of γ_j (ABOVE), definitely to the other (BELOW), or near γ_j for its entire length. Each exact intersection between the two curves becomes a segment of NEARness, so that if γ_i and γ_j are algebraic curves of degree D, then $K_{ij} \le 2D^2$.

Definition 9 (β-Approximation to a Curve) *A* β*-approximation to the curve* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B}\rangle$ *is a continuous, piecewise differentiable curve* $\tilde{\gamma}(t)$ *such that,*

- $\tilde{\gamma}(0) = \mathbf{A}$ *and* $\tilde{\gamma}(1) = \mathbf{B}$,
- $\forall t \in [0,1] : |\mathrm{F}(\tilde{\gamma}(t))| \le \beta$,
- $\forall t \in [0,1]$: *if* $\tilde{\gamma}'(t)$ *is defined then* $\tilde{\gamma}'_x(t) \ge 0$ *and* $\mathrm{sign}(\tilde{\gamma}'_y(t))$ *agrees with* $\mathrm{sign}(\mathbf{B}_y - \mathbf{A}_y)$.

The basic result of this paper is that if we are allowed to replace each γ_i with a β-approximation $\tilde{\gamma}_i$, then we can calculate a vertical ordering in finite precision.

Definition 10 (Approximate Cylindrical Decomposition) *An* approximate cylindrical decomposition *for a set of monotonic curves* $\{\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i\rangle | i = 1, 2, \ldots, C\}$ *is the cylindrical decomposition for some set of* β*-approximations* $\{\tilde{\gamma}_i | i = 1, 2, \ldots, C\}$.

Like γ_i, the curve $\tilde{\gamma}_i$ is implicit. Interestingly enough, the values $\{a_1, a_2, \ldots, a_K\}$ (see Definition 5), and the values $\{y_{\mathrm{in},i}, y_{\mathrm{out},i} | i = 1, 2, \ldots, C\}$, can be explicitly calculated in finite precision.

How good an approximation to γ_i is a β-approximation $\tilde{\gamma}_i$? Certainly $\tilde{\gamma}_i$ stays near to γ_i, at least according to the pseudo-distance function $\mathrm{F}_i(x, y)$. How closely this might correspond to Euclidean distance is a technical matter we will not discuss here. The unfortunate fact is that $\tilde{\gamma}_i$ is only piecewise differentiable, and if the set of curves $\{\gamma_i\}$ is algebraic, then the $\{\tilde{\gamma}_i\}$ generated by the robust curve arrangement algorithm will be only piecewise algebraic. This is the best we can do at present.

2.3 Overview of Robust Arrangement Algorithm

At this point we will first pursue the highest level goal and give an algorithm that constructs a β-approximation to the cylindrical decomposition generated by a set of monotonic curves, $\{\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i \rangle | i = 1, 2, \ldots, C\}$. This algorithm will assume we have available

- an α-approximate coord-eval function (Definition 6),
- a complete set of intersections of γ_i with γ_j for $1 \le i < j \le C$ (Definition 8).

Later, in Section 4, we will reduce the problem of calculating the complete set of intersections to the problem of finding an approximate intersection between an arbitrary line and a curve.

Before we go on with the high level algorithm, let us consider for a moment where the constants α and β come from, and how they depend on the arithmetic we are using. Let us assume that the monotonic curve segment $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$ is part of an algebraic curve defined by the polynomial function,

$$f(x, y) = \sum_{m+n \le D} a_{ij} x^i y^j,$$

where we restrict the domain to the unit square $[-1, 1] \times [-1, 1]$. Assume further that the coefficients $\{a_{ij}\}$ lie in the interval $[-1, 1]$ and have N bits of accuracy. What is N? It is in general hard to measure any physical quantity to better than one part in a million, and so for physical application of robust algorithms, we expect N to be no larger than 20.

If we really know only the first N bits of the coefficients, we really can only know $f(x, y)$ to accuracy of 2^{-N}. Hence we can set $\beta = 2^{-N}$ without sacrificing generality. We will see that $\alpha = \beta/136$, where the ratio 136 represents round-off error introduced while calculating the approximate intersection points.

There are several details left to be worked out, but it seems that these will only involve evaluating $f(x, y)$ or similarly complex polynomial functions to accuracy α, which can be accomplished using $(N + 2\log D + c)$-bit arithmetic where c is a small integer. We know at present that $(N + 17)$-bit arithmetic suffices for calculating arrangements of quadratic curves [14]. These preliminary results indicate that it will be possible to calculate cylindrical decompositions generated by curves of degree 500 to accuracy one part in a million using standard IEEE double precision (53-bit arithmetic). Thus we expect that robust arithmetic will be able to solve all reasonable problems ($N \le 20$, $D < 500$) using unit-cost arithmetic.

2.4 High Level Algorithm

This section presents an algorithm for constructing a β-approximation to the cylindrical decomposition generated by a set of monotonic curve segments (Definition 10). The algorithm has four steps. First we construct for each curve an α-approximation $\hat{\gamma}_i$. Second, we determine the cylindrical decomposition generated by the $\{\hat{\gamma}_i\}$, with some ambiguities allowed. Third, we construct a β-approximation $\tilde{\gamma}_i$ for each curve. Fourth, we compute an ambiguity-free cylindrical decomposition for the $\{\tilde{\gamma}_i\}$.

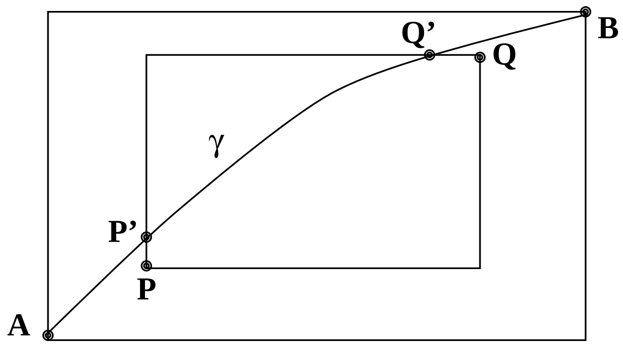

Figure 2: α-Segment $\mathbf{PQ}$

2.4.1 Lemmas

We first give a set of lemmas that will enable us to prove the correctness of the algorithm.

Lemma 1 *If* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$ *is a monotonic curve segment and if* $\mathbf{PQ} \in \mathrm{R}(\mathbf{AB})$ *is a horizontal or vertical line segment, then*

$$|\mathrm{F}(\mathbf{P})|, |\mathrm{F}(\mathbf{Q})| \le \alpha \textit{ implies } \forall \mathbf{V} \in \mathbf{PQ} : |\mathrm{F}(\mathbf{V})| \le \alpha,$$

$$|\mathrm{F}(\mathbf{P})|, |\mathrm{F}(\mathbf{Q})| \le \beta \textit{ implies } \forall \mathbf{V} \in \mathbf{PQ} : |\mathrm{F}(\mathbf{V})| \le \beta.$$

Proof: This lemma immediately follows from the fact that $\frac{\partial \mathrm{F}}{\partial x}$ and $\frac{\partial \mathrm{F}}{\partial y}$ have constant sign on $\mathrm{R}(\mathbf{AB})$.

Definition 11 *An α-segment of a monotonic curve segment* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$ *is a pair of points,* $\mathbf{P}$, $\mathbf{Q}$ *such that,*

- $\mathbf{P}_x \le \mathbf{Q}_x$ *and* $\mathrm{sign}(\mathbf{P}_y - \mathbf{Q}_y)$ *agrees with* $\mathrm{sign}(\mathbf{A}_y - \mathbf{B}_y)$,
- $\mathrm{R}(\mathbf{PQ}) \subseteq \mathrm{R}(\mathbf{AB})$,
- $|\mathrm{F}(\mathbf{P})|, |\mathrm{F}(\mathbf{Q})| \le \alpha$.

Lemma 2 *If* $\mathbf{PQ}$ *is an α-segment of* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$, *then there exists a monotonic, piecewise differentiable curve* $\hat{\gamma}(t)$ *satisfying,* $\mathrm{F}(\hat{\gamma}(0)) = \mathbf{A}$, $\mathrm{F}(\hat{\gamma}(1)) = \mathbf{B}$, *and*

$$\forall t \in [0, 1] : |\mathrm{F}(\hat{\gamma}(t))| \le \alpha.$$

(See Figure 2.)

Proof: We do not know explicitly where $\gamma(t)$ intersects the sides of $\mathrm{R}(\mathbf{PQ})$; nevertheless, we can define $\mathbf{P}'$ to be the point at which $\gamma(t)$ intersects one of the two sides of $\mathrm{R}(\mathbf{PQ})$ containing $\mathbf{P}$, and we can define $\mathbf{Q}'$ to be the point at which $\gamma(t)$ one of the sides containing $\mathbf{Q}$. If $\gamma(t)$ misses $\mathrm{R}(\mathbf{PQ})$ altogether, we define $\mathbf{P}'$ and $\mathbf{Q}'$ to be the vertex of $\mathrm{R}(\mathbf{PQ})$ nearest to γ. In any case, we define $\hat{\gamma}(t)$ to be the curve that joins $\mathbf{P}$ to $\mathbf{P}'$, then follows $\gamma(t)$ from $\mathbf{P}'$ to $\mathbf{Q}'$, then joins $\mathbf{Q}'$ to $\mathbf{Q}$. It follows from Lemma 1 that along $\mathbf{PP}'$ and along $\mathbf{Q}'\mathbf{Q}$, $|\mathrm{F}(\hat{\gamma}(t))| \le \alpha$. It must be stressed that $\gamma(t)$ is implicit and $\hat{\gamma}(t)$ is doubly so; nevertheless, it exists.

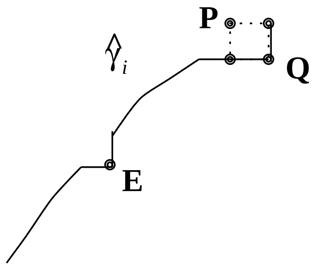

Figure 3: Extending $\hat{\gamma}_i$

2.4.2 Step 1: Constructing $\hat{\gamma}_i$

For each a curve $\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i \rangle$, define the set,

$$\mathrm{S}_i = \bigcup_{j=i+1}^{C} \{\mathbf{I}_{i,j,1}, \mathbf{I}_{i,j,2}, \ldots, \mathbf{I}_{i,j,K_{ij}}\},$$

where the $\{\mathbf{I}_{i,j,k}\}$ are as defined in Definition 8.

Set $\hat{\gamma}_i$ equal to $\mathbf{A}_i$.

While the current right endpoint of $\hat{\gamma}_i$ is not $\mathbf{B}_i$:

- Let $\mathbf{E}$ be the current right endpoint of $\hat{\gamma}_i$.
- Find $\mathbf{P} \in \mathrm{S}_i$ with the least $\mathbf{P}_x$ such that $\mathbf{P} \neq \mathbf{E}$ and $\mathbf{E}_x \leq \mathbf{P}_x$ and $\mathrm{sign}(\mathbf{E}_y - \mathbf{P}_y)$ agrees with $\mathrm{sign}(\mathbf{A}_{i,y} - \mathbf{B}_{i,y})$.
- Find $\mathbf{Q}$ with the least $\mathbf{Q}_y$ with the same properties.
- If $\mathbf{P} = \mathbf{Q}$:
  - if $\mathbf{EP}$ is horizontal or vertical, extend $\hat{\gamma}_i$ by the segment $\mathbf{EP}$;
  - if $\mathbf{EP}$ is not horizontal or vertical, extend $\hat{\gamma}_i$ by the curve $\hat{\gamma}$ implied by Lemma 2 as applied to $\mathbf{EP}$.
- If $\mathbf{P} \neq \mathbf{Q}$ (see Figure 3):
  - Extend $\hat{\gamma}_i$ by $\mathbf{E}\langle \mathbf{P}_x, \mathbf{Q}_y \rangle$ as in the preceding clause.
  - Extend $\hat{\gamma}_i$ by $\langle \mathbf{P}_x, \mathbf{Q}_y \rangle \langle \mathbf{Q}_x, \mathbf{Q}_y \rangle$.
  - Extend $\hat{\gamma}_i$ by $\langle \mathbf{Q}_x, \mathbf{Q}_y \rangle \langle \mathbf{Q}_x, \mathbf{P}_y \rangle$.

2.4.3 Step 2: Total Orders with Attachment

Now that we have the $\{\hat{\gamma}_i\}$, we can start to generate an approximate cylindrical decomposition. Let $\{a_1, a_2, \ldots, a_K\}$ be the sorted list of elements of the set,

$$\bigcup_{i=1}^{C} \{\mathbf{P}_x | \mathbf{P} \in \mathrm{S}_i\}.$$

Pick some open interval (a_k, a_{k+1}). There is a unique horizontal segment or α-segment $\mathbf{P}_i(k)\mathbf{Q}_i(k)$ of $\hat{\gamma}_i$ satisfying,

$$(a_k, a_{k+1}) \subset [\mathbf{P}_i(k)_x, \mathbf{Q}_i(k)_x].$$

Let,

$$\mathrm{G} = \{i | \mathbf{P}_i(k)\mathbf{Q}_i(k) \text{ is not horizontal}\}.$$

If $i, j \in \mathrm{G}$, $i < j$, then it can be shown that γ_i is either NEAR, ABOVE or BELOW γ_j inside $\mathrm{R}(\mathbf{P}_i(k)\mathbf{Q}_i(k))$. For these three cases we say that γ_i is NEAR, ABOVE or BELOW γ_j *above* (a_k, a_{k+1}).

We can construct a *total order with attachments* on G. This term refers to a list of sublists of indices in G:

$$((i_1, i_{1,1}, i_{1,2}, \ldots), (i_2, i_{2,1}, i_{2,2}, \ldots), \ldots, (i_L, i_{L,1}, i_{L,2}, \ldots)),$$

such that,

- the sublists partition G;
- $\forall 1 \leq g < h \leq L$: γ_{i_g} is BELOW γ_{i_h} above (a_k, a_{k+1});
- $\forall 1 \leq g \leq L$: γ_{i_g} is NEAR all the curves
$$\gamma_{i_{g,1}}, \gamma_{i_{g,2}}, \gamma_{i_{g,3}}, \ldots,$$
above (a_k, a_{k+1}).

We call this list of sublists a *total order with attachments* because there is a total order on the heads of the sublists, $\gamma_{i_1}, \gamma_{i_2}, \ldots, \gamma_{i_L}$, and because each curve γ_{i_g} has a list of curves, $\gamma_{i_{g,1}}, \gamma_{i_{g,2}}, \gamma_{i_{g,3}}, \ldots$, attached to it. This data structure represents as much as we can tell about the cylindrical decomposition generated by the $\{\hat{\gamma}_i\}$.

Constructing the total order with attachments is simple. We add the curves one at a time in any order. If the curve we are adding is NEAR any of the heads of the sublists, then it is added to the end of that sublist. Otherwise it becomes the head of a new sublist. Actually, the robust arrangement algorithm sweeps from left to right. We save a considerable amount of effort in constructing the total order above (a_k, a_{k+1}) from the one above (a_{k-1}, a_k).

2.4.4 Intersection with Horizontal or Vertical Lines

Before we go on to Step 3, we observe that for any value $a \in [\mathbf{A}_{i,x}, \mathbf{B}_{i,x}]$ we can find an explicit intersection between the vertical line $x = a$ and the curve $\hat{\gamma}_i(t)$. Most of the cases are easy: if $x = a$ passes through a horizontal segment or through the endpoint of an α-segment or if it contains a vertical line segment, the intersection is obvious.

In the hard case, $x = a$ passes through an α-segment $\mathbf{P}_i\mathbf{Q}_i$ of γ_i.

- Let $\mathbf{I} = \text{coord-eval}(\gamma_i, x = a)$.
- If $\mathbf{I} \in \mathrm{R}(\mathbf{P}_i\mathbf{Q}_i)$, then we are done.
- If $\mathbf{I}_y > \max(\mathbf{P}_{i,y}\mathbf{Q}_{i,y})$, then set $\mathbf{I}_y$ equal to $\max(\mathbf{P}_{i,y}\mathbf{Q}_{i,y})$.
- If $\mathbf{I}_y < \min(\mathbf{P}_{i,y}\mathbf{Q}_{i,y})$, then set $\mathbf{I}_y$ equal to $\min(\mathbf{P}_{i,y}\mathbf{Q}_{i,y})$.

Modify $\hat{\gamma}_i$ by splitting $\mathbf{P}_i\mathbf{Q}_i$ into α-segments $\mathbf{P}_i\mathbf{I}$ and $\mathbf{IQ}_i$ (see Figure 4).

This algorithm generates an explicit intersection point. Since $\hat{\gamma}_i$ is an approximation, we are free to modify it. Intersections with vertical lines are analogous.

2.4.5 Step 3: Constructing $\tilde{\gamma}_j$

A vertical ordering (Definition 4) is a total order, not a total order with attachments. We need to construct an approximation $\tilde{\gamma}_j$ for each γ_j so that each pair of curves $\tilde{\gamma}_i$ and $\tilde{\gamma}_j$ can be unambiguously ordered along each vertical line. Recall that $\tilde{\gamma}_j$ is required to be a β-approximation to γ_j (Definition 5). Naturally we set $\tilde{\gamma}_j(0)$ equal to $\mathbf{A}_j$. We know that,

$$\mathbf{A}_{j,x} = a_l \text{ for some } a_l \in \{a_1, a_2, \ldots, a_K\}.$$

Let us assume we have extended $\tilde{\gamma}_j$ to $x = a_k$, and the current rightmost endpoint $\mathbf{E}$ of $\tilde{\gamma}_j$ is $\langle a_k, b_k \rangle$. The following procedure extends it to $x = a_{k+1}$.

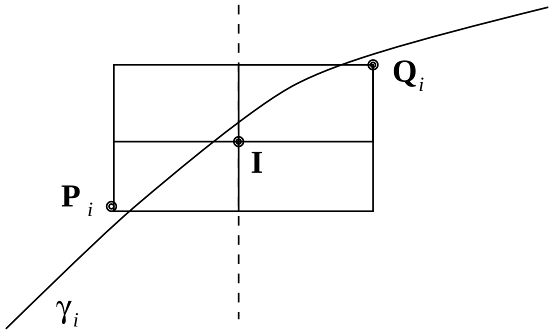

Figure 4: Intersection of a Vertical Line with $\hat{\gamma}_i$

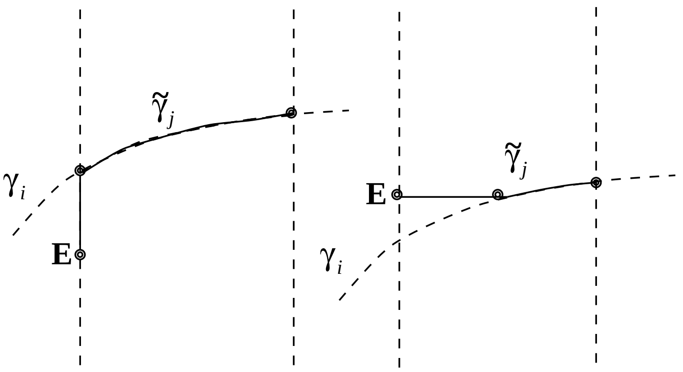

Figure 5: Extending $\tilde{\gamma}_i$

- If $\hat{\gamma}_j$ is horizontal above (a_k, a_{k+1}), extend $\tilde{\gamma}_j$ horizontally to $\langle a_{k+1}, b_k \rangle$.
- Otherwise find the index j in the total order with attachments above (a_k, a_{k+1}). Let i be the head of the sublist to which j belongs $(i \leq j)$.
  - Let $\mathbf{I}(a_k)$ be the intersection of $x = a_k$ with $\hat{\gamma}_i$. Similarly $\mathbf{I}(a_{k+1})$.
  - If $b_k \leq \mathbf{I}(a_k)_y$, extend $\tilde{\gamma}_j$ vertically to $\mathbf{I}(a_k)$ and then append the portion of $\hat{\gamma}_i$ joining $\mathbf{I}(a_k)$ to $\mathbf{I}(a_{k+1})$ (left half of Figure 5).
  - If $b_k \geq \mathbf{I}(a_{k+1})_y$, extend $\tilde{\gamma}_j$ horizontally to $\langle a_{k+1}, b_k \rangle$.
  - If $\mathbf{I}(a_k)_y < b_k < \mathbf{I}(a_{k+1})_y$, then let $\mathbf{Q}$ be the intersection of the line $y = b_k$ with $\hat{\gamma}_i$. Extent $\tilde{\gamma}_j$ horizontally to $\mathbf{Q}$ and then append the portion of $\hat{\gamma}_i$ joining $\mathbf{Q}$ to $\mathbf{I}(a_{k+1})$ (right half of Figure 5).

Note that in many cases we add an α-segment of $\hat{\gamma}_i$ to $\tilde{\gamma}_j$. This is always all right because the NEARness condition assures that all points $\mathbf{P}$ of $\hat{\gamma}_i$ along this segment satisfy $|\mathrm{F}_j(\mathbf{P})| \leq \beta$.

The procedure above assumes that γ_i and γ_j are both rising (Definition 3). If γ_j is rising and γ_i is falling, we must not add a segment of $\hat{\gamma}_i$ to $\tilde{\gamma}_j$. Instead we do the following (Figure 6).

- If $b_k \geq \mathbf{I}(a_k)_y$, extend $\tilde{\gamma}_j$ horizontally across (a_k, a_{k+1}).
- If $b_k < \mathbf{I}(a_k)_y$, we add a vertical segment to $\mathbf{I}(a_k)_y$ and then extend $\tilde{\gamma}_j$ horizontally across (a_k, a_{k+1}).

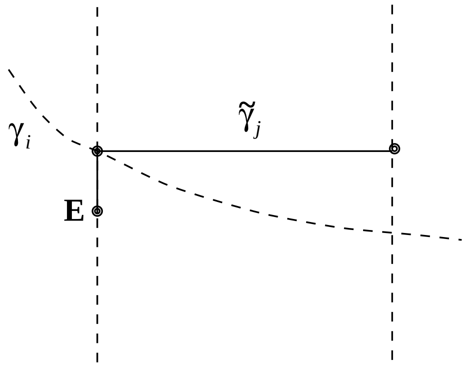

Figure 6: Special Case: γ_i Falling, γ_j Rising

2.4.6 Step 4: Generating the Cylindrical Decomposition

It is straightforward to show that $\tilde{\gamma}_j$ is a β-approximation to γ_j. Furthermore, the α-segments appearing in the final shapes of the set of curves $\{\tilde{\gamma}_i\}$ are separated in the following sense: if $x = a$ intersects an α-segment of γ_i and an α-segment of γ_j, then γ_i will be ABOVE or BELOW γ_j above $x = a$, not NEAR. Thus no two of the α-segments intersect.

Let $\mathbf{PQ}$ be some horizontal segment which is part of some $\tilde{\gamma}_j$. We can construct the intersection of $\mathbf{PQ}$ with $\tilde{\gamma}_i$ as follows.

- Compute coord-eval$(\gamma_i, y = \mathbf{P}_y)$ to generate a point $\mathbf{I}$.
- If $\mathbf{I} \in \mathbf{PQ}$, split $\mathbf{PQ}$ into $\mathbf{PI}$ and $\mathbf{IQ}$.

The resulting set of $\{\tilde{\gamma}_i\}$ intersect only at explicit locations. The cylindrical decomposition is easily derived from the curves.

3 Computational Complexity

In this section we examine the computational complexity of the robust arrangement algorithm and the standard algorithm for computing cylindrical decompositions.

3.1 Robust Algorithm

We can put an upper bound on the cost of the robust algorithm by examining each component in turn. Wherever possible, we will estimate the typical cost we expect for the component.

The first component is the cost of computing the complete set of approximate intersections (Definition 8) for each pair of curves γ_i and γ_j. Let L be the cost of computing the approximate intersection of an arbitrary line with γ_i (see Definition 14). Section 4 will show that the cost of computing a single approximate intersection point of γ_i and γ_j is $\mathrm{O}(L \log N)$, where N is the number of bits of precision in the output. The total cost of computing the intersections is $\mathrm{O}(VL \log N)$, where V is the total number of vertices.

Sorting all the intersection points for Step 1 of the algorithm costs $\mathrm{O}(V \log C)$. Constructing all the $\hat{\gamma}_i$ costs $\mathrm{O}(V)$.

In Step 2, the length of the list $a_1, a_2, \ldots, a_K$ is no more than $2V$. Constructing the total order with attachments above (a_k, a_{k+1}) costs $\mathrm{O}(C \log C)$, although we really expect that creating it from the total order above (a_k, a_{k+1}) to be constant in the typical case (in the worst case we might have to reinsert all the elements of a sublist). Hence the cost of Step 2 is $\mathrm{O}(VC \log C)$, worst case, $\mathrm{O}(V)$ typical case.

Computing the intersection of a vertical or horizontal line with $\hat{\gamma}_i$ costs $\mathrm{O}(L + \log C)$ if we know which interval (a_k, a_{k+1}) we are dealing with. Constructing each $\tilde{\gamma}_j$ involves at most four line intersections per interval. Hence the total cost of constructing all the $\tilde{\gamma}_j$ is $\mathrm{O}(VCL + VC \log C)$. With certain optimizations and in the typical case, this cost is probably $\mathrm{O}(VL)$.

Finally, Step 4 involves computing the intersection of every horizontal or vertical segment with each curve $\tilde{\gamma}_i$. The maximum number of segments is $2VC$ and the cost of computing each intersection is $\mathrm{O}(L+\log C)$. If every segment intersects every other curve, the cost is $\mathrm{O}(VC^2L+VC^2 \log C)$. Typically, we expect only V segments and a constant number of intersections per segment. Typical cost is expected to be $\mathrm{O}(VL)$.

Adding everything up, the worst case cost is

$$\mathrm{O}(VL \log N + VC^2 L + VC^2 \log C).$$

The typical cost is expected to be

$$\mathrm{O}(VL \log N + V \log C).$$

It is reasonable to assume that for algebraic curves of degree N,

$$L = \mathrm{O}(D^2 \log N), \qquad V = C^2 D^2/2.$$

Worst case under these assumptions:

$$\mathrm{O}(D^4 C^2 \log^2 N + D^4 C^4 \log N + D^2 C^4 \log C).$$

Typical:

$$\mathrm{O}(D^4 C^2 \log^2 N + D^2 C^2 \log C).$$

3.2 Exact Techniques

Algebraic curves are the zeros of polynomials in two variables. The input to a algebraic curve arrangement algorithm would be a set of polynomials. We cannot explicitly represent the curves, but by manipulating the coefficients of the polynomials, we can compute the cylindrical decomposition of the plane generated by the curves. This section examines the standard techniques for this problem and estimates their cost.

We are given a set of polynomials

$$\{f_i(x, y) | i = 1, 2, \ldots, C\}.$$

Let us assume that these polygons are of degree D and have coefficients which are N-bit integers. The standard technique constructs a set of *resultants*

$$\{r_{ij}(x, y) | 1 \leq i < j \leq C\}.$$

These resultants are polynomials of degree D^2 with $2DN$-bit coefficients. The union of the roots of these resultants will be the set $\{a_1, a_2, \ldots, a_K\}$ in the cylindrical decomposition (Definition 5) generated by the corresponding algebraic curves. Generating the cylindrical decomposition is at least as hard as sorting the set of roots of all the resultants.

Each resultant will have $2DN$-bit coefficients and be of degree D^2. The total number of degree D^2, $2DN$-bit polynomials is $2^{2D^3 N}$. At least $2^{2D^3 N}/D^2$ of these are irreducible [16], and so there are at least this many possible values for the roots of the resultants. It is easy to show that the roots must lie in the bounded interval $[-D^2 2^{DN}, D^2 2^{DN}]$. Hence roots can be as close as $D^4 2^{DN}/2^{2D^3 N}$. Thus distinguishing the roots may require calculating them with $D^3 N$ bits of precision. Assuming that N-bit arithmetic has unit cost, it requires at least D^5 operations to evaluate each a_k and hence $D^7 C^2$ to evaluate all of them.

Unfortunately, we cannot be sure that this worst case is attainable because the resultants are not independent of one another. Also, we assume that in order to distinguish two roots, we must evaluate the resultant at a rational value between the two roots. There are ways of determining the order of the roots without evaluation [5], but these seem to be as expensive. Still, cost proportional to $D^7 C^2$ is plausible, and for fixed C and large D, it is highly likely that the robust curve arrangement algorithm has lower computational complexity than the exact algorithm.

4 Low Level Algorithms

The high level curve arrangement algorithm of Section 2 requires a complete set of approximate intersection (Definition 8) for each pair of curves. This section reduces the problem of finding complete sets of intersections to simpler problems. First, we assume that we can compute the approximate intersection of an arbitrary line and a monotonic curve segment. Second, we assume that each curve segment is convex and that we can compute approximate tangent lines. For the set of algebraic curves, these are reasonable assumptions, and therefore this section shows that the robust curve arrangement algorithm is very likely to apply to algebraic curves.

We will first formally define an approximate line-curve intersection and approximate tangent lines. Assuming primitive operations that satisfy these definitions, Section 4.2 shows how to compute a *single* approximate intersection between two curves. Section 4.3 then goes on to show how to use the single-intersection algorithm to generate complete sets of intersections.

4.1 Definitions

This section describes three primitive operations that we assume that we can perform on the curves: the *evaluation primitive*, the *normal vector primitive*, and the *line intersection primitive*.

For each monotone curve segment $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$ we make a number of assumptions. First, we assume that

$$|\mathbf{A}_x|, |\mathbf{A}_y|, |\mathbf{B}_x|, |\mathbf{B}_y| \leq 1$$

and that

$$\forall \mathbf{P} \in \mathrm{R}(\mathbf{AB}) : \left| \frac{\partial \mathrm{F}}{\partial x}(\mathbf{P}) \right|, \left| \frac{\partial \mathrm{F}}{\partial y}(\mathbf{P}) \right| \leq 1.$$

In addition, we assume that each monotone curve segment is convex: the curvature of γ has constant sign. Finally, we assume that for $|\kappa| < 200$, the curve defined by $\mathrm{F}(x, y) = \kappa\alpha$ is also a monotone curve segment (this is the same α that appears in Definition 8).

Definition 12 (Evaluation Primitive) *Given a monotonic curve segment* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B} \rangle$ *and a point* $\mathbf{P} \in \mathrm{R}(\mathbf{AB})$*, the* evaluation primitive *gives a value* $\mathrm{F}(\mathbf{P})_\mathrm{F}$ *that satisfies*

$$|\mathrm{F}(\mathbf{P})_\mathrm{F} - \mathrm{F}(\mathbf{P})| \leq \alpha.$$

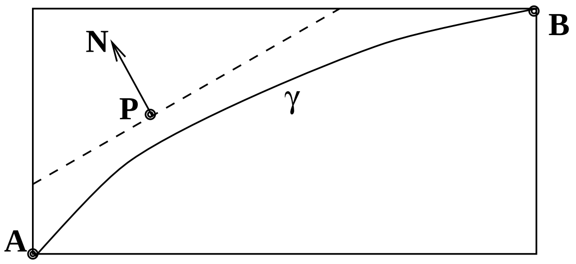

Figure 7: Approximate Normal to Convex

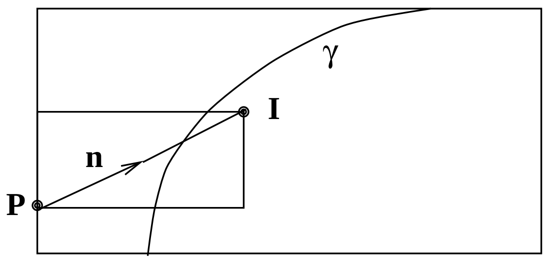

Figure 8: Line Intersection Primitive

In other words, we cannot know the value of $\mathrm{F}(x, y)$, only an approximation with accuracy α. For the values of α we are concerned with, we assume that we need $(N+2\log D+c)$-bit arithmetic to achieve this accuracy (c is a constant).

Definition 13 (Normal Vector Primitive) *Given a monotonic curve segment* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B}\rangle$ *and a point* $\mathbf{P} \in \mathrm{R}(\mathbf{AB})$, *the* normal vector primitive *returns an approximate unit normal vector* $\vec{\mathbf{N}}(\mathrm{F}, \mathbf{P})$ *such that if* γ *is* $\left\{ \begin{array}{c} \textit{convex} \\ \textit{concave} \end{array} \right\}$ *then* $\forall \mathbf{Q} \in \mathrm{R}(\mathbf{AB})$: $(\mathbf{Q}-\mathbf{P}) \cdot \vec{\mathbf{N}}(\mathrm{F}, \mathbf{P}) \geq 0$ *implies*

$$\left\{ \begin{array}{c} \mathrm{F}(\mathbf{Q}) \geq \mathrm{F}(\mathbf{P}) - \alpha \\ \mathrm{F}(\mathbf{Q}) \leq \mathrm{F}(\mathbf{P}) + \alpha \end{array} \right\}$$

(see Figure 7).

Definition 14 (Line Intersection Primitive) *The* line intersection primitive *takes four inputs:*

- *a monotone curve segment* $\langle \mathrm{F}, \gamma, \mathbf{A}, \mathbf{B}\rangle$,
- *a coordinate rectangle (see Definition 1)* $\mathbf{R} \subseteq \mathrm{R}(\mathbf{AB})$,
- *a point* $\mathbf{P}$ *on the boundary of* $\mathbf{R}$.
- *a unit normal vector* $\vec{\mathbf{n}}$ *defining a line* $\mathbf{L}$ *passing through* $\mathbf{P}$.

The operator returns ABOVE, BELOW, ABOVE-INTER *or* BELOW-INTER *with the following meanings.*

- ABOVE*:* $\forall \mathbf{Q} \in \mathbf{R} : (\mathbf{Q}-\mathbf{P}) \cdot \vec{\mathbf{n}} \geq 0 \rightarrow \mathrm{F}(\mathbf{Q}) \geq -\alpha$.
- BELOW*:* $\forall \mathbf{Q} \in \mathbf{R} : (\mathbf{Q}-\mathbf{P}) \cdot \vec{\mathbf{n}} \leq 0 \rightarrow \mathrm{F}(\mathbf{Q}) \leq \alpha$.
- ABOVE-INTER*:*

  $$\exists \mathbf{I} \in \mathbf{R} : |(\mathbf{I}-\mathbf{P}) \cdot \vec{\mathbf{n}}| \leq \alpha \textit{ and } |\mathrm{F}(\mathbf{I})| \leq \alpha$$

  and

  $$\forall \mathbf{Q} \in \mathrm{R}(\mathbf{PI}) : (\mathbf{Q}-\mathbf{P}) \cdot \vec{\mathbf{n}} \geq 0 \rightarrow \mathrm{F}(\mathbf{Q}) \geq -\alpha$$

  (see Figure 8).

- BELOW-INTER*:*

  $$\exists \mathbf{I} \in \mathbf{R} : |(\mathbf{I}-\mathbf{P}) \cdot \vec{\mathbf{n}}| \leq \alpha \textit{ and } |\mathrm{F}(\mathbf{I})| \leq \alpha$$

  and

  $$\forall \mathbf{Q} \in \mathrm{R}(\mathbf{PI}) : (\mathbf{Q}-\mathbf{P}) \cdot \vec{\mathbf{n}} \leq 0 \rightarrow \mathrm{F}(\mathbf{Q}) \leq \alpha.$$

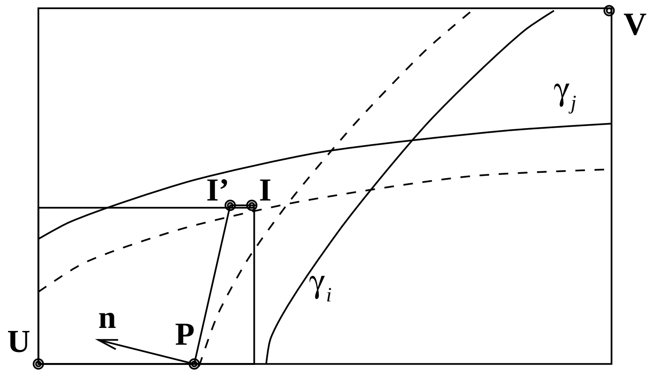

Figure 9: Line Casting

If the operator returns ABOVE-INTER *or* BELOW-INTER, *it also generates a point* $\mathbf{I}$ *which satisfies the appropriate condition.*

This definition rigorously defines the notion that $\mathbf{I}$ is the intersection point between $\mathbf{L}$ $(= \langle \mathbf{P}, \vec{\mathbf{n}}\rangle)$ and γ in $\mathbf{R}$ and nearest to $\mathbf{P}$. Figure 8 illustrates the relation among $\mathbf{I}$, $\mathbf{P}$, $\vec{\mathbf{n}}$, γ, and $\mathbf{R}$ in the case that the primitive returns ABOVE-INTER.

4.2 Calculating Intersections

Starting with the three input primitives we create a *single-vertex intersection operator* which determines whether two curves intersect inside a coordinate rectangle.

Definition 15 (Single-Vertex Intersection Operator) *A* single-vertex intersection operator *takes as input two curves* $\langle \mathrm{F}_i, \gamma_i, \mathbf{A}_i, \mathbf{B}_i\rangle$ *and* $\langle \mathrm{F}_j, \gamma_j, \mathbf{A}_j, \mathbf{B}_j\rangle$ *and a coordinate rectangle* $\mathbf{R} \subseteq \mathrm{R}(\mathbf{A}_i\mathbf{B}_i) \cap \mathrm{R}(\mathbf{A}_j\mathbf{B}_j)$. *It either returns a point* $\mathbf{I} \in \mathbf{R}$ *such that,*

$$|\mathrm{F}_i(\mathbf{I})|, |\mathrm{F}_j(\mathbf{I})| \leq 10\alpha,$$

or it asserts one the following relations inside $\mathbf{R}$*:*

- γ_i *is* ABOVE γ_j,
- γ_i *is* BELOW γ_j.

The algorithm for the single-vertex operator uses a technique called *line casting.* Figure 9 illustrates this technique for curves γ_i and γ_j and rectangle $\mathrm{R}(\mathbf{UV})$. These dotted curves are the zeros of $\mathrm{F}_i(x, y) - 3\alpha$ and $\mathrm{F}_j(x, y) + 3\alpha$. Using two calls to coord-eval (Definition 6), we generate a point $\mathbf{P}$ on the boundary of $\mathrm{R}(\mathbf{UV})$ satisfying $|\mathrm{F}_i(\mathbf{P}) - 3\alpha| \leq \alpha$. Let $\vec{\mathbf{n}} = \vec{\mathbf{N}}(\mathrm{F}_i, \mathbf{P})$. Call the line intersection primitive with $\mathbf{P}$, $\vec{\mathbf{n}}$ and $\mathrm{F}_i(x, y) - 3\alpha$. In this case, the output is a point $\mathbf{I}$ and the assertion BELOW-INTER.

- If $\mathrm{F}_j(\mathbf{I})_{\mathrm{F}} \leq 2\alpha$, then $\mathbf{I}$ is the output of the single-vertex intersection operator.
- Otherwise, recurse with rectangle $\mathrm{R}(\mathbf{IV})$.

Lemma 3 *For each step of the recursion, if we return* $\mathbf{I}$, *then* $|\mathrm{F}_i(\mathbf{I})|, |\mathrm{F}_j(\mathbf{I})| \leq 10\alpha$. *Otherwise, there exists a polygonal curve joining* $\mathbf{P}$ *to* $\mathbf{I}$ *separating the curve* $\mathrm{F}_i(x, y) = \alpha$ *from* $\mathrm{F}_j(x, y) = -\alpha$ *inside* $\mathrm{R}(\mathbf{UI})$.

Proof: The first statement follows from the termination condition. To prove the second part, let $\mathbf{I}'$ be the other intersection with the boundary of $\mathrm{R}(\mathbf{UI})$ of the line defined by $\mathbf{P}$ and $\vec{\mathbf{n}}$ ($\mathbf{I}'$ is implicit). Every point $\mathbf{W}$ on the path $\mathbf{PI}'$, $\mathbf{I}'\mathbf{I}$ satisfies

$$\mathrm{F}_i(\mathbf{W}) > \alpha \text{ and } \mathrm{F}_j(\mathbf{W}) < \alpha.$$

We see from the lemma that as we recurse, we generate a path,

$$\mathbf{P}_1\mathbf{I}_1'\mathbf{I}_1\mathbf{P}_2\mathbf{I}_2'\mathbf{I}_2 \cdots \mathbf{P}_k\mathbf{I}_k'\mathbf{I}_k$$

that separates the curve $\mathrm{F}_i(x, y) = \alpha$ from $\mathrm{F}_j(x, y) = -\alpha$ inside $\mathrm{R}(\mathbf{UI}_k)$. If we reach the other side of $\mathrm{R}(\mathbf{UV})$, we return BELOW; otherwise we return $\mathbf{I}_k$. Either way we satisfy Definition 15.

It can be shown that for curves that meet with angle θ, this algorithm converges in $\mathrm{O}(\theta^{-1} \log N)$ steps. For small θ (tangent curves), in order to obtain the same computational complexity, we replace F_j by some linear combination, $\lambda\mathrm{F}_i + \mu\mathrm{F}_j$. The single-vertex intersection algorithm is given in its entirety in [14].

4.3 Multiple Intersections

If we examine Definition 8, we see that the single-vertex intersection operator comes close to generating the complete list of intersection points. Certainly, we can generate rectangles $\mathrm{R}(\mathbf{UI})$ such that γ_i is ABOVE or BELOW γ_j inside $\mathrm{R}(\mathbf{UI})$. The question remains as to how we can generate α-segments of γ_i that are NEAR to γ_j. The full details of the complete intersection algorithm are given in [14], but the following lemma indicates the central idea.

Lemma 4 *If inside rectangle* $\mathbf{R}$ γ_i *is BELOW the curve generated by* $\mathrm{F}_j(x, y) - 10\alpha$ *and ABOVE the curve generated by* $\mathrm{F}_j(x, y) + 10\alpha$, *then* γ_i *is NEAR to* γ_j *inside* $\mathbf{R}$ *(assuming* $\beta \geq 11\alpha$*).*

The proof follows directly from the definitions of ABOVE, BELOW and NEAR. So we see that generating and proving an interval of NEARness is simply a matter of making two calls to the single-vertex intersection operator. It turns out that in order to handle all possible cases, it is necessary to set $\beta = 136\alpha$.

5 Conclusion

There remains a considerable amount of work to be done in the area of creating practical solutions to the curve arrangement problem. First of all, we must work out the numerical details of algorithms for the three input primitives for the set of algebraic curves with degree higher than two. Also, for fixed degree D we must make progress towards establishing a more reasonable $\mathrm{O}(C^2 \log C)$ upper bound on the cost of the algorithm.

Nevertheless, we have taken an important first step. The definition of the set of valid input curves and the necessary primitive provides a useful abstraction. More importantly, the definition of an approximate arrangement makes it possible to study a very broad new class of approximate geometric algorithms.

References

[1] D. S. Arnon, G. E. Collins, and S. McCallum. An adjacency algorithm for cylindrical algebraic decompositions of three-dimensional space. *J. Symbolic Computation*, 5:163–187, 1988.

[2] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decompositions i: the basic algorithm. *SIAM J. Comp.*, 13(4):865–877, 1984.

[3] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decompositions ii: an adjacency algorithm for the plane. *SIAM J. Comp.*, 13(4):878–889, 1984.

[4] G. E. Collins. *Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. Lecture Notes in Computer Science, No. 35*, Springer-Verlag, New York, 1975.

[5] M. Cost and M. F. Roy. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *J. Symbolic Computation*, 5:121–129, 1988.

[6] Herbert Edelsbrunner and Ernst Peter Mucke. *Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms.* Technical Report UIUCDCS-R-87-1393, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, December 1987.

[7] Daniel H. Greene and F. Frances Yao. Finite-resolution computational geometry. In *27th Annual Symposium on the Foundations of Computer Science*, pages 143–152, IEEE, October 1986.

[8] C. Hoffmann and J. Hopcroft. Towards implementing robust geometric computations. In *Proceedings of the Symposium on Computational Geometry*, ACM, 1988.

[9] Christoph M. Hoffmann. *The Problem of Accuracy and Robustness in Geometric Computation.* Technical Report CSD-TR-771, Computer Sciences Department, Purdue University, April 1988.

[10] Christoph M. Hoffmann, John E. Hopcroft, and Michael S. Karasick. *Robust Set Operations on Polyhedral Solids.* Technical Report 87-875, Department of Computer Science, Cornell University, Ithaca, New York 14853-7501, October 1987.

[11] Michael Karasick. *On the Representation and Manipulation of Rigid Solids.* PhD thesis, McGill University, August 1988.

[12] M. Karsick, D. Lieber, and L. Nackman. *Efficient Delaunay Triangulation using Rational Arithmetic.* IBM Research Report RC 14455, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, To appear in 1989.

[13] Victor Milenkovic. Verifiable implementations of geometric algorithms using finite precision arithmetic. *Artificial Intelligence*, 37:377–401, 1988.

[14] Victor J. Milenkovic. *Verifiable Implementations of Geometric Algorithms using Finite Precision Arithmetic.* Technical Report CMU-CS-88-168, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, July 1988.

[15] Thomas Ottmann, Gerald Thiemt, and Christian Ullrich. Numerical stability of geometric algorithms. In *Proceedings of the Symposium on Computational Geometry*, ACM, 1987.

[16] M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comp.*, 9(2):273–280, May 1980.

[17] D. Salesin, J. Stolfi, and L. Guibas. Epsilon geometry: building robust algorithms from imprecise computations. In *Proceedings of the Symposium on Computational Geometry*, ACM, 1989.

[18] J. Schwartz and M. Sharir. On the 'piano movers' problem, ii. general techniques for computing topological properties of real algebraic manifolds. *Adv. Appl. Math.*, 4:298–301, 1983.

[19] Mark Segal and Carlo H. Sequin. Consistent calculations for solids modeling. In *Proceedings of the Symposium on Computational Geometry*, pages 29–37, ACM, June 1985.

[20] Mark Segal and Carlo H. Sequin. Partitioning polyhedral objects into non-intersecting parts. *IEEE Computer Graphics and Applications*, 8(1), January 1988.

[21] K. Sugihara. An approach to error-free solid modeling. In *IMA Summer Program on Robotics*, Institute for Mathematics and Applications, University of Minnesota, 1987.

[22] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, 1948. 2nd edition, 1951.

[23] C. Yap. A geometric consistency theorem for a symbolic perturbation theorem. In *Proceedings of the Symposium on Computational Geometry*, ACM, 1988.