Precise Voronoi Cell Extraction of Free-form Rational Planar Closed Curves

Iddo Hanniel†, Ramanathan Muthuganapathy†,†, Gershon Elber†, Myung-Soo Kim†

†Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000, Israel

‡School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea

ABSTRACT
We present an algorithm for generating the Voronoi cells for a set of rational $C^1$-continuous planar closed curves, which is precise up to machine precision. Initially, bisectors for pairs of curves, $(C(t), C_i(r))$, are generated symbolically and represented as implicit forms in the $tr$-parameter space. Then, the bisectors are properly trimmed after being split into monotone pieces. The trimming procedure uses the orientation of the original curves as well as their curvature fields, resulting in a set of trimmed-bisector segments represented as implicit curves in a parameter space. A lower-envelope algorithm is then used in the parameter space of the curve whose Voronoi cell is sought. The lower envelope represents the exact boundary of the Voronoi cell.

Keywords: Voronoi cells, Skeleton, Free-form boundaries, Rationals, MAT

1. INTRODUCTION
Voronoi diagrams are one of the most extensively studied objects in computational geometry. Algorithms for generating Voronoi diagrams for rational entities typically preprocess the input curved boundaries into linear and circular segments. This preprocessing generates a Voronoi diagram that is approximate both in topology and geometry. Given a number of disjoint planar regions bounded by free-form curve segments $C_0(t), C_1(r_1), \ldots, C_n(r_n)$, their Voronoi diagram [1] is defined as a set of points that are equidistant but minimal from two different regions. The term ‘minimal’ ensures that for a point on the boundary curve, the corresponding point on the Voronoi diagram is the minimum in distance. This definition excludes self-Voronoi edges [2]. The Voronoi cell of a curve $C_0(t)$ is the set of all points closer to $C_0(t)$ than to $C_j(r_j), \forall j > 0$. The Voronoi diagram is then the union of the Voronoi cells of all the free-form curves. In this paper, the term Voronoi cell normally refers to the ‘boundary of the Voronoi cell’. A related entity to the Voronoi diagram, the Medial Axis Transform (MAT) or Skeleton, was introduced by Blum [3, 4] to describe biological shapes. The MAT can be viewed as the locus of the center and radius of a maximal ball as it rolls inside an object. Since their introduction, both Voronoi diagrams and MATs have been used in a wide variety of applications that primarily involve reasoning about geometry or shapes. Skeletons have been used in pattern and image analysis [5, 6], finite element mesh generation [7, 8], and path planning [9], to name a few.

Algorithms for generating Voronoi cells/diagrams have predominantly used linear or circular arc inputs [10, 11, 12]. Moreover, algorithms for generating Voronoi diagrams for rational entities typically preprocess the input curved boundaries into linear and circular segments [13], due to the difficulty in processing rationals directly. This preprocessing not only leads to the generation of artifacts and branches not present in the original Voronoi diagram, which requires a post-processing stage to remove them, but also it produces an approximated Voronoi diagram [14, 15].

A Voronoi diagram of $C^1$ planar curves consists of portions of bisector curves for some pairs of curves. In particular, the Voronoi cell of a curve $C_0(t)$ will involve portions of bisectors between a pair of curves in the set, one of which will be $C_0(t)$. However, generating the bisector for a pair of planar curves is trivial only if they are simple, such as straight lines or circular arcs. When the curve is a rational free-form curve, the bisector is rational only for a few special cases – a point and a rational curve in the plane [16] and two rational space curves for which the bisector is a rational surface [17]. However, for the case of coplanar curves (polynomial or rational), the bisector has been shown to be, in general, algebraic but not rational [16]. This has resulted in the need for numerical tracing of the bisector curves [18], which is computationally expensive. Alternatively, Elber and Kim [19] showed that the bisector for a pair of rational curves can be implicitly represented symbolically in the parametric space. The bisector so generated can be represented in an implicit form that can be used for further processing. Moreover, the bisectors can be accurately represented up to machine precision.

Voronoi diagram generation algorithms that do not preprocess curved boundaries are relatively harder to find. Laven-
In this paper, the problem of generating a Voronoi cell of a planar free-form closed curve is addressed. The proposed algorithm directly uses the free-form rational curves that are not preprocessed into line segments or circular arcs. It is shown here that the symbolically generated bisector that is not minimal in distance to the corresponding boundary do not belong to the desired side. This constraint can further. Each monotone piece is subsequently subjected to several constraint checks described in Section 5. The constraints are applied based on the orientation of the two rational curves as well as their curvature fields. The orientation constraint determines the bisector’s side with respect to the curve. The object side is assumed to lie on the right hand side when the curve is traversed in the increasing direction of the regular parameterization. Application of the orientation constraint purges away portions of the untrimmed bisector that do not belong to the desired side. This constraint can also be flipped to obtain bisector portions that lie on the other side of the curve.

Following this, the resulting bisector portions are subjected to a curvature constraint. This constraint determines whether the radius of curvature of the input rational curve at a footpoint of the bisector is greater than the radius of curvature of the disk determined by the distance from the footpoint to the bisector. Note that the application of this constraint purges away some (but not all) points on the bisector that are not minimal in distance to the corresponding boundary curves.

The above process is repeated for all pairs of curves \( (C_0(t), C_i(r_i)) \). One feature of the algorithm described in this paper is the application of the lower envelope algorithm [22] to generate the Voronoi cell of \( C_0(t) \) with respect to \( C_i(r_i), \forall i > 0 \). The lower envelope algorithm here takes advantage of the correspondence between the bivariate function and the distance function that measures the distance from the footpoint to the corresponding point on the bisector. The lower envelope algorithm for the generation of the Voronoi cell at \( C_0(t) \) is described in Section 6.

3. **BIVARIATE FUNCTION**

In the coming sections, \( r \) will replace \( r_i \) for the purpose of clarity of representation. Let \( C_0(t) = (x_0(t), y_0(t)) \) and \( C_i(r) = (x_i(r), y_i(r)) \) be two planar \( C^1 \)-continuous regular rational curves. Though the functions \( F_1(t, r) \) or \( F_2(t, r) \) in monotone pieces. This decomposition facilitates the effective manipulation of the bivariate function when processed further. Each monotone piece is used to decompose the bivariate function into two rational \( C^1 \) planar curves.

![Figure 1: The bisector and the zero-set between two open \( C^1 \) rational curves \( C_1(t) \) and \( C_i(r) \).](image-url)
can be used, the bivariate function \( F_3(t, r) \) in [19] has been selected for the generation of the untrimmed bisector because it does not generate redundant branches, making it ideal for use in further processing. For completeness, the following formulation is taken, though not fully, from Elber and Kim [19]. [19] showed that a bisector point \( P \) must satisfy:

\[
\begin{align*}
\langle P - C_0(t), C_0'(t) \rangle &= 0, \\
\langle P - C_1(r), C_1'(r) \rangle &= 0, \\
\langle P - \frac{(C_0(t) + C_1(r))}{2}, C_0(t) - C_1(r) \rangle &= 0.
\end{align*}
\]

When the two tangents \( C_0'(t) \) and \( C_1'(r) \) are neither parallel nor opposite, the point \( P = (x, y) \) on the intersection of the two normal lines of the two curves has a unique symbolic solution for the following matrix equation; from Equations (1) and (2):

\[
\begin{bmatrix} C_0(t) \\ C_1(r) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \langle C_0(t), C_0'(t) \rangle \\ \langle C_1(r), C_1'(r) \rangle \end{bmatrix}.
\]

Using Cramer’s rule, we can generate a planar bivariate rational surface: \( P(t, r) = (x(t, r), y(t, r)) \), which is embedded in the \( xy \)-plane:

\[
\begin{align*}
x(t, r) &= \frac{x_0(t)x_0'(t) + y_0(t)y_0'(t)}{x_0(t)x_1'(r) + y_0(t)y_1'(r)}, \\
y(t, r) &= \frac{x_0(t)x_0(t) + y_0(t)y_0(t)}{x_1(r)x_1'(r) + y_1(r)y_1'(r)}.
\end{align*}
\]

Substituting the expressions for \( x(t, r) \) and \( y(t, r) \) into Equation (3), we get:

\[
0 = F_3(t, r) = \langle P(t, r) - \frac{(C_0(t) + C_1(r))}{2}, C_0(t) - C_1(r) \rangle.
\]

Equation (6) is then multiplied by the denominator of \( P(t, r) \) to yield \( F_3(t, r) \),

\[
0 = F_3(t, r) = (x_0(t)y_1'(r) - x_1'(r)y_0(t))\tilde{F}_3(t, r).
\]

Once Equation (7) is solved, points on the bisector can be computed from Equations (5). The bisector curve has a considerably lower degree when represented in a parameter space (see [19] for further details), compared to the conventional representation of the bisector curve in the \( xy \)-plane [25].

Figure 2(a) shows the untrimmed bisector of two closed curves and Figure 2(b) shows the surface \( (t, r, F_3(t, r)) \) generated by the bivariate function and its corresponding zero-set. It should be noted that \( F_3(t, r) \) represents the bisector with an accuracy that is bounded only by the machine’s precision.

4. SPLITTING INTO MONOTONE PIECES

In order to manipulate and effectively use (i.e., compute lower envelopes) the zero-set of the bivariate function, \( F_3(t, r) = 0 \), it is necessary to break it into monotone pieces, in the \( tr \)-space. Given the polynomial \( F_3(t, r) \) in the \( tr \)-domain, the problem is to decompose the zeros of the function into monotone pieces within the domain, while obtaining the connectivity between the monotone pieces. This process results in branches of the function delimited by appropriate \( tr \)-domain points, such that when one traverses the branch from one endpoint to the other, the branch is increasing/decreasing in both \( t \) and \( r \). The topology analysis algorithm presented in [27] has been implemented to split \( F_3(t, r) = 0 \) into monotone branches in both \( t \) and \( r \).

The topology analysis algorithm in [27] uses the turning points (local maxima and minima) and edge points as the input. The turning points are isolated by finding the com-
mon simultaneous solutions between $F_3(t, r) = 0$ and either $rac{∂F_3}{∂r}(t, r) = 0$ or $\frac{∂F_3}{∂t}(t, r) = 0$. The multivariate solver of [26] is used to find these local extrema. Edge points are identified by computing all the intersections of $F_3(t, r) = 0$ with the boundaries of the domain of interest. Recursive subdivision in the $tr$-space is used to find the connectivity between all turning and edge points. Each subdivision involves taking a vertical or horizontal line and finding all intersections of the curve with that line. The edge points are further classified depending on which border they hit.

The algorithm proceeds by analyzing the pattern of turning points and edges points in the region, and either finding the connections between them, or subdividing the region. For further details, please refer to [27].

Figure 3 shows the monotone pieces (bounded by a box) obtained by the application of the topology analysis algorithm for the zero-set of of $F_3(t, r)$ shown in Figure 1(b). The algorithm splits the given input function at the middle whenever two or more turning points are detected. Hence, the two monotone regions that are visible in the middle region of Figure 3. This, however, does not affect our algorithm in any way.

5. APPLYING CONSTRAINTS

In this section, two constraints are described to trim the individual bisectors represented as the zeros of the bivariate function $F_3(t, r)$. The constraints are based on the orientation of the input curves (Section 5.1) and their curvature fields (Section 5.2).

5.1 Orientation Constraint

The orientation constraint uses the direction of the parameterization of the input rational curves. The right hand side is assumed to belong to the interior of the object while traversing the curve along the increasing direction of parameterization. The orientation constraint purges away points on the untrimmed bisector that do not lie on the desired side — that is, a left-left constraint generates $tr$-points corresponding to bisector points that lie to the left of both curves. Hence it can also be termed as LL-constraint where LL implies Left-Left. In a similar manner, we can construct the other orientation constraints as LR/RL/RR constraints.

**Figure 5:** The monotone pieces of the zero-set of $F_3(t, r)$ shown in Figure 1(b).

The LL-constraint for two regular planar closed curves after applying the LL-constraint (see also Figure 2(a)). The bisector extends to $\infty$ on both sides.

The constraints are based on the orientation constraints as LR/RL/RR constraints.

**Figure 4:** The bisector (in gray) between two closed $C^1$ planar curves after applying the LL-constraint.

In this section, two constraints are described to trim the input functions $F_3(t, r)$ by the application of the topology analysis algorithm for the zero-set of $F_3(t, r) = 0$ with the boundaries of the domain of interest. Recursive subdivision in the $tr$-space is used to find the connectivity between all turning and edge points. Each subdivision involves taking a vertical or horizontal line and finding all intersections of the curve with that line. The edge points are further classified depending on which border they hit.

The algorithm proceeds by analyzing the pattern of turning points and edges points in the region, and either finding the connections between them, or subdividing the region. For further details, please refer to [27].

Figure 3 shows the monotone pieces (bounded by a box) obtained by the application of the topology analysis algorithm for the zero-set of of $F_3(t, r)$ shown in Figure 1(b). The algorithm splits the given input function at the middle whenever two or more turning points are detected. Hence, the two monotone regions that are visible in the middle region of Figure 3. This, however, does not affect our algorithm in any way.

5. APPLYING CONSTRAINTS

In this section, two constraints are described to trim the individual bisectors represented as the zeros of the bivariate function $F_3(t, r)$. The constraints are based on the orientation of the input curves (Section 5.1) and their curvature fields (Section 5.2).

5.1 Orientation Constraint

The orientation constraint uses the direction of the parameterization of the input rational curves. The right hand side is assumed to belong to the interior of the object while traversing the curve along the increasing direction of parameterization. The orientation constraint purges away points on the untrimmed bisector that do not lie on the desired side — that is, a left-left constraint generates $tr$-points corresponding to bisector points that lie to the left of both curves. Hence it can also be termed as LL-constraint where LL implies Left-Left. In a similar manner, we can construct the other orientation constraints as LR/RL/RR constraints.

**Figure 5:** The monotone pieces of the zero-set of $F_3(t, r)$ shown in Figure 1(b).

The LL-constraint for two regular planar closed curves after applying the LL-constraint (see also Figure 2(a)). The bisector extends to $\infty$ on both sides.

The constraints are based on the orientation constraints as LR/RL/RR constraints.

**Figure 4:** The bisector (in gray) between two closed $C^1$ planar curves after applying the LL-constraint.
Figure 5: (a) The radius of the disk is smaller than the radius of curvatures at the footpoints of the curves. (b) The radius of the disk is greater than the radius of curvature of $C_1(r)$ at the footpoint implying the violation of the constraint.

5.2 Curvature constraint

The curvature constraint is based on the computation of the curvature fields of the input rational curves and disk at a point on the bisector. This constraint eliminates some (though not all) points on the bisector that do not satisfy the minimal distance requirement of the Voronoi diagram. This constraint, in the form of a lemma, is as follows:

**Lemma 1.** The radius of curvature of the disk at any point on the Voronoi diagram/cell should be less than or equal to the minimum of the local radius of curvature of the boundary segments.

For the proof of the lemma, please refer to [21].

Figure 5 illustrates the curvature constraint. Figure 5(a) shows that the radius of the disk is smaller than the radius of curvature of the curves at the footpoints, implying that the curvature constraint is fully satisfied and the center of the disk (marked a in the figure) becomes a possible valid point on the Voronoi diagram. Violation of this constraint implies that the radius of the disk at that bisector point is greater than the radius of curvature of a curve at the footpoint. Figure 5(b) shows a disk that violates the curvature constraint. Consequently, the center of such a disk (marked b in the figure), though it is a point on the bisector, is not a point on the Voronoi diagram, as it violates the minimality of the distance. Hence, points such as b have to be purged away.

To formulate the constraint, a distance function $D(t, r)$ is introduced that corresponds to the distance between the bisector point and the footpoint (actually to both footpoints as they are equidistant from the bisector point) and is defined as follows:

$$D_0(t, r) = ||P(t, r) - C_0(t)||,$$  \hspace{1cm} (14)

$$D_1(t, r) = ||P(t, r) - C_1(r)||.$$  \hspace{1cm} (15)

The distance functions correspond to the radius of the disk that is tangent to the footpoints, and either $D_0(t, r)$ or $D_1(t, r)$ can be used. Since the curvature of the disk of a bisector point is smaller than the (positive) curvature of one of the curves at the footpoint, we have:

$$1/D_0(t, r) > \langle \text{sign}(C_0''(t), N_0(t))\rangle \kappa_0(t), \hspace{1cm} (16)$$

$$1/D_1(t, r) > \langle \text{sign}(C_1''(r), N_1(r))\rangle \kappa_1(r), \hspace{1cm} (17)$$

where the sign function denotes the sign of the expression within the brackets and $\kappa_0(t)$ and $\kappa_1(r)$ are the curvature functions of the respective curves.

Constraints (16) and (17) appeared to be too slow in our preliminary implementation since they are applied over all the points on the bisector. Moreover, they are not rational expressions as they contain square roots and hence must be squared. In contrast, vector functions $\hat{N}_0(t)/\kappa_0(t)$, $\hat{N}_1(r)/\kappa_1(r)$, $\kappa_0(t)\hat{N}_0(t)$, and $\kappa_1(r)\hat{N}_1(r)$ are rational, provided $C_0(t)$ and $C_1(r)$ are rational curves, where $\hat{N}_0(t)$ and $\hat{N}_1(r)$ denote the unit normal vectors. Then, the curvature constraints can be reformulated in the following alternate way:

$$\langle P(t, r) - C_0(t), \hat{N}_0(t)/\kappa_0(t) \rangle < \langle \hat{N}_0(t)/\kappa_0(t), \hat{N}_0(t)/\kappa_0(t) \rangle,$$  \hspace{1cm} (18)

$$\langle P(t, r) - C_1(r), \hat{N}_1(r)/\kappa_1(r) \rangle < \langle \hat{N}_1(r)/\kappa_1(r), \hat{N}_1(r)/\kappa_1(r) \rangle.$$  \hspace{1cm} (19)

$\hat{N}_0(t)$ (resp. $\hat{N}_1(r)$) can either be in the same or the opposite direction with respect to $\hat{N}_0(t)$ (resp. $\hat{N}_1(r)$). If they are in the opposite directions, the left hand side of inequalities (18) and (19) will be negative and, therefore, always hold. If $\hat{N}_0(t)$ (resp. $\hat{N}_1(r)$) is in the same direction as $\hat{N}_0(t)$ (resp. $\hat{N}_1(r)$), then these inequalities will hold only when the radius of the disk is smaller than the radius of curvature $1/\kappa_0(t)$ (resp. $1/\kappa_1(r)$) of the boundary curve (see Figure 5).

The curvature constraint also implies that a point on the Voronoi cell cannot be closer to its footpoint than the evolute point corresponding to that footpoint, since $C_0(t) + \hat{N}_0(t)/\kappa_0(t)$ (resp. $C_1(r) + \hat{N}_1(r)/\kappa_1(r)$) traces the evolute of a curve $C_0(t)$ (resp. $C_1(r)$) and is rational, provided $C_0(t)$ (resp. $C_1(r)$) is rational.

The curvature constraints (18) and (19) can be reduced to

$$\langle P(t, r) - C_0(t), \hat{N}_0(t)/\kappa_0(t) \rangle < 1,$$  \hspace{1cm} (20)

$$\langle P(t, r) - C_1(r), \hat{N}_1(r)/\kappa_1(r) \rangle < 1;$$  \hspace{1cm} (21)

or

$$\langle P(t, r) - C_0(t), \kappa_0(t)\hat{N}_0(t) \rangle < 1,$$  \hspace{1cm} (22)

$$\langle P(t, r) - C_1(r), \kappa_1(r)\hat{N}_1(r) \rangle < 1,$$  \hspace{1cm} (23)

since $\langle \hat{N}_0(t), \hat{N}_0(t) \rangle = \langle \hat{N}_1(r), \hat{N}_1(r) \rangle = 1$.

Figure 6 shows the portions of the bisector obtained after applying the curvature constraint (and after subjecting it to
6.1 General Lower Envelope Algorithm

Given a set of \( t \)-monotone curves, they are initially divided into two subsets of size at most \( \lceil n/2 \rceil \), recursively computing the lower envelope for each of the subsets. Then, these subenvelopes are merged back to obtain the overall lower envelope. The termination condition of the recursion is a single \( t \)-monotone curve that is the lower envelope of itself.

The main part of the algorithm is the merging step, which is performed in a sweep-like manner. Let us assume that we have two lists of curves, which are themselves lower envelopes that are to be merged. The merging process of the two lists starts from the leftmost \( t \) parameter value. A sweep-like algorithm, along the \( t \)-domain, identifies the set of intervals \([a, b]\) where the two lists overlap. Therefore, we end up with merges of the intervals \([a_i, b_i]\) in the \( t \)-domain where both curves are defined. All the intersection points of the curves from the two lists lying on the interval \([a_i, b_i]\) are identified. The \( t \) parameter values, where intersections occur, are identified. Between a pair of intersection points, the portions that belong to the merged lower envelope have to be identified. This can be done by comparing the \( D_i(t) \)-values of the two curves at the middle \( t \)-parameter between the intersection points. Because of continuity, the curve with the smaller \( D_i(t) \)-value at the mid-parameter has smaller \( D_i(t) \)-values over the interval \([a, b]\) and, therefore, belongs to the merged lower envelope.

The lower envelope algorithm is illustrated for line segments in Figure 7(a). Consider the two segments 1-1 and 2-2, as is shown in the figure. The first step involves the identification of overlapping \( t \)-parametric regions and the splitting of the two segments at these parametric values. For example, for the two curves 1-1 and 2-2 shown in Figure 7(a), the overlapping parameter region is \( ab \) and hence they are split at parameters \( a \) and \( b \). Further processing is required only for the portions where the overlapping of parameters occur. The portion of overlapping parametric regions is tested for intersections. If intersection points exist, they are again split at those parameter values (e.g., \( c \) in Figure 7(b)). The split portions between intersection points are then tested to identify the ones with minimal distance and to eliminate other portions. The \( D_i(t) \) comparison check is performed only at the mid-parameter value between the intersections (marked as dots in Figure 7(b)). The result of the merging process in the example is shown in Figure 7(c).

6.2 Lower envelopes of rationals

Analyzing the algorithm, it is easy to see that it requires only a few basic geometric operations. However, when rational curves are involved, the possible computation each of these functions may differ. Then, the main functions needed are

- A function that identifies all \( t \)-parameters where intersections occur. Formulation of the intersection parameters can vary from one problem to another and also
is a correspondence between $F$ and $D$. Therefore, we can apply the lower envelope algorithm.

Figure 7: Illustration of the lower envelope algorithm.

depends on the tools that are available for that particular problem. For example, in the case of a Voronoi cell, equating two distance functions, computed symbolically, will determine the $t$ parameter of the intersections.

- A function that compares the $D_i(t)$-values of two curves at a given $t$-parameter. In the Voronoi cell determination, this function amounts to a comparison of the distance function values at the corresponding middle $t$-parameter values.
- A function for splitting a $t$-monotone curve into two subcurves at a new $t$-parameter.

For line segments or for univariate functions, these basic functions can easily be implemented. In the following section, the details of generating the Voronoi cell via lower envelopes is described.

6.3 Voronoi Cell via Lower Envelope

Given a $t$-monotone $F_3(t, r_1) = 0$ function, for every $t$ value there is a single corresponding $r_1$ value. Furthermore, there is a correspondence between $F_3(t, r_1) = 0$ and $D_i(t)$. If $F_3(t, r_1) = 0$ is monotone over an interval $t \in [a, b]$, then $D_i(t)$ is well defined over $[a, b]$ since for every $t$ value there is a single $r_1$ value and hence a single corresponding distance value. Therefore, we can apply the lower envelope algorithm on the distance functions defined over monotone bisectors of the form $F_3(t, r_1) = 0$, provided we can compute the needed basic predicates for $D_i(t)$ as described in Section 6.2.

Unfortunately, a rational representation of $D_i(t)$ is not available. The square of the distance function $D_i(t, r_j)$ may be used instead. Figure 8(b) shows the squared distance functions $D_i^2(t, r_1)$ and $D_i^2(t, r_2)$ for the bisector between pairs of planar $C^1$ rational curves $(C_0(t), C_1(r_1))$ and $(C_0(t), C_2(r_2))$ shown in Figure 8(a). Then, the required basic predicates can be computed symbolically using $F_3(t, r_1)$ and $D_i^2(t, r_i)$ functions.

The intersection points can be identified using the following set of equations ($i \neq j$):

\[ ||D_i(t, r_j)||^2 = ||D_j(t, r_i)||^2, \]

\[ F_3(t, r_1) = 0, \]

\[ F_3(t, r_j) = 0. \]

Comparison of two squared distance functions, $D_i^2(t)$ and $D_j^2(t)$, at a given $t$ parameter is performed by initially obtaining the corresponding $r_i$ and $r_j$ parameter values (a single solution for $t$-monotone $F_3(t, r_1) = 0$ segments) and then comparing the function values of $D_i^2(t, r_1)$ and $D_j^2(t, r_j)$ at the respective parameter values.

To split a $D_i^2(t)$ function into two subcurves at a given $t$-parameter, all that is needed is to split its corresponding $t$-monotone $F_3(t, r_1) = 0$ implicit function at that parameter.

The result of the lower envelope algorithm is then a list of $t$-monotone $F_3(t, r_1) = 0$ implicit curves that are split at $t$-parameters of equidistant bisector points. The union of these $F_3(t, r_1) = 0$ functions represents the Voronoi cell of $C_0(t)$.

7. RESULTS AND DISCUSSION

We have implemented the proposed algorithm for extracting the Voronoi cell of a closed curve using the IRIT [24].
modeling environment. In this section, results from some test cases are presented. In what follows, the closed curve for which the Voronoi cell is to be generated is denoted as $C_0(t)$ and the other closed curves are denoted appropriately as $C_i(r_i), \forall i > 0$. The degree of the input curves (denoted as $\text{Deg}$), number of control points used (denoted as $\text{CP}$) and the degree of $F_3(t,r)$ (denoted as $\text{Deg}F_3$) computed as given in [19] are also provided for each of the figures. They can sometimes denote the maximum value of the input curves which can be identified by the tag $\text{max}$. Figure 9 shows the Voronoi cell of a closed curve $C_0(t)$. Figure 10 shows the Voronoi cell of $C_0(t)$, where the input geometry consists of four closed curves. Figures 11 and 12 show two more test cases where all objects have convex profiles. The algorithm works well for non-convex curves as well (see Figures 13 and 14).

7.1 Discussion

The following are the major steps of our algorithm:

1. formulate the bivariate function $F_3(t,r)$;
2. split the zero-sets of $F_3(t,r)$’s into monotone pieces;
3. apply the trimming constraints; and
4. compute the lower envelope.

Our experimental results indicate that all the above steps are reasonably efficient. Computation of the symbolic function $F_3(t,r)$ took from a fraction of a second to a few seconds. Computation of the monotone pieces and application of the constraints took from several seconds to a minute. The lower envelope algorithm took from a few seconds to a few minutes. All the experiments were carried out on an
Fably robust. Splitting the zero-set of a bivariate function
Even though the algorithm has been implemented using
the IRIT [24] environment.
Intel Pentium 4 1.8GHz computer with 256MB RAM using
approximation of the rational curves. The algorithm generates
remove the artifacts that would have been created by an ap-
sidered precise to within the machine precision. Moreover,
the data that are generated on the V oronoi cell can be con-
approximating them by linear or circular arcs. As a result,
The input rational free-form curves are used directly without
have been tested so far. Exact arithmetic tends to slow down
worked well with floating point arithmetic in all cases that
exact arithmetic, our experiments indicate that the splitting
sion. Though the splitting procedure described in [27] uses
viable for pairs of rational curves can be used
be preprocessed and approximated with linear or circular
also shows that the input rational curves do not need to
be preprocessed and approximated with linear or circular
segments, thereby eliminating the post-processing stage re-
quired to trim the artifacts generated by this preprocessing.
Another promising important advantage of this approach is
the option of applying the same basic strategy to generate
Voronoi diagram and medial axis of an object bounded by
the option of applying the same basic strategy to generate
Voronoi cells of planar curves by employing
nal [16] and the bisector between two rational curves can be
represented implicitly as shown in [19], this work can also
be used to generate Voronoi diagrams or MATs for domains
bounded by piecewise $C^1$ rational curves and vertices that
are concave or reflex. This is currently under investigation.

8. CONCLUSIONS
This paper presented an algorithm for generating the Voronoi
cells of a set of $C^1$-continuous rational planar closed curves.
It is shown that the symbolically computed bisectors in suit-
able parameter space for pairs of rational curves can be used to
generate the Voronoi cells of planar curves by employing
a lower envelope algorithm. The approach in this paper
also shows that the input rational curves do not need to
be preprocessed and approximated with linear or circular
Voronoi cells that are topologically correct and geometrically
accurate to whatever precision desired.

Not all bisectors, generated between pairs of closed curves,
are going to play a role in generating the Voronoi cell of some
closed curve. For example, for the closed curve $C_0(t)$ in
Figure 10, the Voronoi cell of $C_0(t)$ does not contain parts
of the bisector generated for the pair $C_0(t)$ and $C_3(r_3)$, even
though the untrimmed bisector is generated for them and
processed by the algorithm.

Since the bivariate function $F_3(t,r)$ can be used to gener-
ate self-bisectors (bisectors of a curve with itself), the pre-
sented algorithm can also be used to generate self-Voronoi
edges. Since the definition of a Voronoi diagram precludes
self-Voronoi edges (traditionally, Voronoi diagram is defined
only for distinct entities), it is not shown in the current
work. However, the self-Voronoi edges will be useful when
constructing the MAT of free-form curves. Since the bisector
between a point and a rational curve is shown to be ratio-

9. REFERENCES

Acknowledgments
This research was supported in part by the Israel Science
Foundation (grant No. 857/04) and in part by an European
FP6 NoE grant 506766 (AIM@SHAPE), and in part by the
Korean Ministry of Science and Technology (MOST) under
the Korean-Israeli Binational Research Grant.

Figure 13: Voronoi cell (in gray) of $C_0(t)$. $\deg_{\text{max}} = 3$, $CP_{\text{max}} = 8$ and $\deg F_{\text{max}} = 10$.

Figure 14: Voronoi cell (in gray) of $C_0(t)$. $\deg = 3$, $CP = 7$ and $\deg F_3 = 10$.

Even though the algorithm has been implemented using
floating point arithmetic, it has been found to be reason-
ably robust. Splitting the zero-set of a bivariate function
$F_3(t,r)$ into monotone pieces involves the computation of
turning points, which can be computed to arbitrary preci-
isation tends to slow down the algorithm considerably. It is also possible that alternate
algorithms that split a curve into monotone pieces could be
used. The lower envelope algorithm uses symbolic computa-
tion of distance functions and hence it is fast, reliable and
robust.

The input rational free-form curves are used directly without
approximating them by linear or circular arcs. As a result,
the data that are generated on the Voronoi cell can be con-
sidered precise to within the machine precision. Moreover,
the algorithm does not require a post-processing stage to
remove the artifacts that would have been created by an ap-
proximation of the rational curves. The algorithm generates
M. Ramanathan and B. Gurumoorthy

[12] V. Srinivasan and L. R. Nackman


[31] http://cs.nyu.edu/exact/core/