Precise Voronoi Cell Extraction of Free-form Planar Piecewise C^1-Continuous Closed Rational Curves

I. Hanniel, M. Ramanathan*, G. Elber
Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000, Israel
e-mail: {iddoh, raman, gershon}@cs.technion.ac.il

M.-S. Kim
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
e-mail: mskim@cse.snu.ac.kr

Abstract

We present an algorithm for generating Voronoi cells for a set of planar piecewise C^1-continuous closed rational curves, which is precise up to machine precision. The algorithm starts with the symbolically generated bisectors for pairs of C^1-continuous curve segments $(C(t), C_i(r))$. The bisectors are represented implicitly in the tr-parameter space. Then, they 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. The algorithm also supports piecewise C^1-continuous curves and generates the Voronoi cell of such input curves using additional point/curve bisector segments.

Keywords: Voronoi cells, Rational curves, Free-form boundaries, Skeleton, Medial Axis Transform.

1 Introduction

Voronoi diagrams are one of the most extensively studied objects in computational geometry. Algorithms for generating Voronoi diagrams for rational curves typically preprocess the input curved boundaries into linear and circular segments [Held 1998]. 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 $C_0(t)$, $C_1(r_1)$, ... , $C_n(r_n)$, their Voronoi diagram [Aurenhammer 1991] 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 [Alt and Schwarzkopf 1995]. 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, unless otherwise noted, 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 [Blum 1967; Blum 1973] 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 [Baja and Thiel 1994; Montanari 1969], finite element mesh generation [Armstrong 1994; Gursoy and Patrikalakis 1992], and path planning [O’Rourke 1993], to mention only a few.

Algorithms for generating Voronoi cells/diagrams have predominantly used linear or circular arc inputs [Kim et al. 1995; Srinivasan and Nackman 1987; Yap 1987]. Moreover, algorithms for generating Voronoi diagrams for rational entities typically preprocess the input curved boundaries into linear and circular segments [Held 1998], due to the difficulty in processing rational curves 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 produces an approximated Voronoi diagram [Ramamurthy and Farouki 1999; Ramanathan and Gurumoorthy 2003].

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 [Farouki and Johnstone 1994] and two rational space curves for which the bisector is a rational surface [Elber and Kim 1998b]. However, for the case of coplanar curves (polynomial or rational), the bisector has been shown to be, in general, algebraic but not rational [Farouki and Johnstone 1994]. This has resulted in the need for numerical tracing of the bisector curves [Farouki and Ramamurthy 1998], which is computationally expensive. Alternatively, Elber and Kim [Elber and Kim 1998a] showed that the bisector for a pair of rational curves can be represented implicitly and 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. Lavender et al. [Lavender et al. 1992] suggested a subdivision technique (based on interval arithmetic) to construct the global structure of the Voronoi diagram. This method is general in the sense that it can handle arbitrary set-theoretic objects in any dimensions. Chou [Chou 1995] suggested an algorithm that is based on a tree structure, whereas Alt and Schwarzkopf [Alt and Schwarzkopf 1995] presented a randomized incremental algorithm. Nevertheless, both algorithms [Alt and Schwarzkopf 1995; Chou 1995] appear to have only theoretical implications. Ramamurthy and Farouki [Ramamurthy and Farouki 1999], and Ramanathan and Gurumoorthy [Ramanathan and Gurumoorthy 2003] have used numerical tracing methods. [Ramamurthy and Farouki 1999] first generated the bisectors and then trim them, whereas [Ramanathan and Gurumoorthy 2003] generated the trimmed segments of the medial axis directly.

*Corresponding author.

In this paper, the problem of generating a Voronoi cell of planar free-form closed C^1-continuous curves is being addressed. Further, we also consider piecewise C^1-continuous curves, i.e., curves with finitely many C^1 discontinuities along them, extending our recently reported work [Hanniel et al. 2005]. To begin with, the algorithm is described assuming the curves are C^1 continuous since it involves only the computation of bisectors between pairs of curves. The extension of the algorithm for the presence of tangent discontinuity points is then described, involving additional bisectors between points and curves.

The proposed algorithm 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 represented as an implicit form in the parametric space between a pair of planar rational curves can be processed to extract the Voronoi cell of a closed curve. A lower envelope algorithm [Sharir and Agarwal 1995], a well-known divide-and-conquer technique, has been used for generating the Voronoi cell [Kao 1999] (to be precise, [Kao 1999] generated the medial axis). Though [Kao 1999] has described an algorithm that uses lower envelope for generating medial axis transform of general curved objects, he approximated the input curves using bi-arc splines in his implementation. It is shown here that the lower envelope algorithm can be used for processing rational curves directly. The output of the proposed approach is the boundary of a Voronoi cell. This boundary is not rational in general and hence is represented as a piecewise implicit form in the parametric spaces of the original rational curves. This introduced representation is precise up to machine precision, and while not exact, is probably amendable to exact arithmetic computations, a topic beyond the scope of this work.

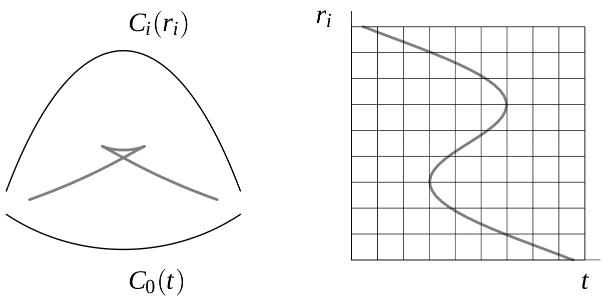
All the algorithms and examples presented in this paper were implemented and created with the aid of tools available in the IRIT [Elber 2002] solid modeling system, developed at the Technion, Israel.

The rest of this paper is structured as follows: Section 2 outlines the basic idea of this paper. The bivariate implicit representation of the bisector is described in Section 3. Splitting the function into monotone pieces is then described in Section 4, followed by the description of several bisector constraints in Section 5. Section 6 presents the algorithm for Voronoi cell extraction for a closed curve using a lower envelope algorithm. The extension of the Voronoi cell algorithm for piecewise C^1-continuous curves is described in Section 7. Results from our implementation and a discussion on the algorithm appear in Section 8. Finally, Section 9 concludes the paper.

2 Basic Idea

Let $C_0(t), C_1(r_1), ..., C_n(r_n)$ be a set of $(n+1)$ C^1-continuous rational parametric planar closed curves. Without loss of generality, we may assume that the Voronoi cell is extracted for the curve $C_0(t)$. When the curves are C^1- continuous, only the bisectors between pairs of curves, i.e. curve/curve bisectors (hereafter denoted as CCB) are involved, which are non-rational, in general [Farouki and Johnstone 1994]. The extension for piecewise C^1-continuous curves, discussed in Section 7, also involves the bisectors between a point and a curve, i.e. point/curve bisectors (hereafter denoted as PCB), which are rational [Farouki and Johnstone 1994].

The algorithm starts with the symbolic computation of a bivariate polynomial function , $\mathscr{F}_3(t, r_i)$, following [Elber and Kim 1998a], between two planar curves $C_0(t)$ and $C_i(r_i)$ so that the zero-set of $\mathscr{F}_3(t, r_i)$, $\mathscr{F}_3(t, r_i) = 0$, which is an implicit algebraic curve



(a) The bisector (in gray) between two rational C^1 planar curves.

(b) The zero-set in the domain of $\mathscr{F}_3(t, r_i)$, defines the bisector.

Figure 1: The bisector and the zero-set between two open C^1 rational curves $C_1(t)$ and $C_i(r_i)$.

in the tr_i-plane, corresponds to the untrimmed bisector curve between $C_0(t)$ and $C_i(r_i)$. The formulation of $\mathscr{F}_3(t, r_i)$, which is described in Section 3, is based on a symbolic substitution of polynomial/rational functions of t and r_i into a simple polynomial expression in the tr_i-plane. Figure 1(a) shows the bisector between two planar rational curves. The zero-set of its $\mathscr{F}_3(t, r_i)$ is shown in Figure 1(b).

In Figure 1(a), the CCB ends before it passes through the narrow portion of the two input curves. This early termination is due to the fact that the CCB measures the distances to the two input curves in the orthogonal directions to the curves' tangents. Moreover, near these narrow regions, the displayed CCB should connect to a PCB and then to a point-point bisector, which are not considered, as we assume C^1 geometry for now.

A topology analysis algorithm [Keyser et al. 2000], described in Section 4, is then used to decompose the bivariate function into tr_i-monotone pieces. This decomposition facilitates the effective manipulation of the bivariate function when processed 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 curves. 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. 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 [Sharir and Agarwal 1995] 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 for $C_0(t)$ is described in Section 6.

A similar approach is adopted if the input curves are piecewise C^1-continuous, considering the additional bisectors between points and curves. This is described in Section 7.

3 Bivariate Function - Curve/Curve Bisector

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 $C_1(r) = (x_1(r), y_1(r))$ be two planar C^1-continuous regular rational curves. Though the functions $\mathscr{F}_1(t,r)$ or $\mathscr{F}_2(t,r)$ in [Elber and Kim 1998a] can be used, the bivariate function $\mathscr{F}_3(t,r)$ in [Elber and Kim 1998a] 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 [Elber and Kim 1998a]. [Elber and Kim 1998a] showed that a bisector point P must satisfy:

$$\left\langle P - C_0(t), C_0'(t) \right\rangle = 0, \tag{1}$$

$$\left\langle P - C_1(r), C_1'(r) \right\rangle = 0, \tag{2}$$

$$\left\langle P - \frac{(C_0(t) + C_1(r)}{2}, C_0(t) - C_1(r) \right\rangle = 0. \tag{3}$$

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):

$$\begin{bmatrix} C_0'(t) \\ C_1'(r) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \langle C_0(t), C_0'(t) \rangle \\ \langle C_1(r), C_1'(r) \rangle \end{bmatrix}. \tag{4}$$

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:

$$x(t,r) = \frac{\begin{vmatrix} x_0(t)x_0'(t) + y_0(t)y_0'(t) & y_0'(t) \\ x_1(r)x_1'(r) + y_1(r)y_1'(r) & y_1'(r) \end{vmatrix}}{\begin{vmatrix} x_0'(t) & y_0'(t) \\ x_1'(r) & y_1'(r) \end{vmatrix}},$$

$$y(t,r) = \frac{\begin{vmatrix} x_0'(t) & x_0(t)x_0'(t) + y_0(t)y_0'(t) \\ x_1'(r) & x_1(r)x_1'(r) + y_1(r)y_1'(r) \end{vmatrix}}{\begin{vmatrix} x_0'(t) & y_0'(t) \\ x_1'(r) & y_1'(r) \end{vmatrix}}. \tag{5}$$

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

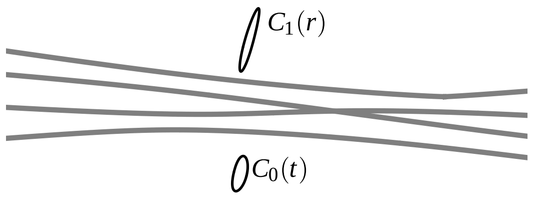
$$\begin{aligned} 0 &= \hat{\mathscr{F}}_3(t,r) \\ &= \left\langle P(t,r) - \frac{(C_0(t) + C_1(r)}{2}, C_0(t) - C_1(r) \right\rangle. \end{aligned} \tag{6}$$

Equation (6) is then multiplied by the denominator of $P(t,r)$ to yield $\mathscr{F}_3(t,r)$,

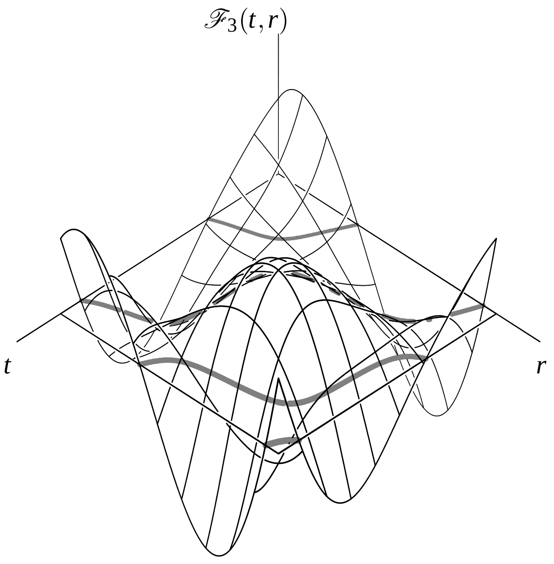
$$0 = \mathscr{F}_3(t,r) = (x_0'(t)y_1'(r) - x_1'(r)y_0'(t))\hat{\mathscr{F}}_3(t,r). \tag{7}$$

Once Equation (7) is solved, points on the bisector can be computed from Equation (5). The bisector curve has a considerably lower degree when represented in a parameter space (see [Elber and Kim 1998a] for further details), compared to the conventional representation of the bisector curve in the xy-plane [Hoffmann and Vermeer 1991].

Figure 2(a) shows the untrimmed bisector of two closed curves and Figure 2(b) shows the surface $(t, r, \mathscr{F}_3(t,r))$ generated by the bivariate function and its corresponding zero-set. It should be noted that $\mathscr{F}_3(t,r) = 0$ represents the bisector with an accuracy that is bounded only by the machine precision.



(a) The untrimmed bisector (in gray) of two C^1 closed planar curves (in black). The bisectors extend to ∞ on both sides.



(b) The bivariate function $\mathscr{F}_3(t,r)$ and its zero-set.

Figure 2: The untrimmed bisector (a) and the zero-set of $\mathscr{F}_3(t,r)$ (b) between two C^1 closed planar curves $C_0(t)$ and $C_1(r)$.

4 Splitting into Monotone Pieces

In order to manipulate and effectively use (i.e., compute lower envelopes) the zero-set of the bivariate function, $\mathscr{F}_3(t,r) = 0$, it is necessary to break it into monotone pieces, in the tr-space. Given the polynomial $\mathscr{F}_3(t,r)$ 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

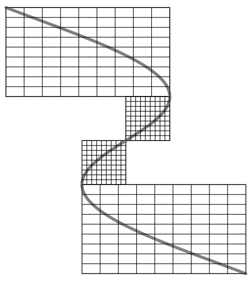


Figure 3: The monotone pieces of the zero-set of $\mathscr{F}_3(t,r)$ shown in Figure 1(b).

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 [Keyser et al. 2000] was implemented to split $\mathscr{F}_3(t,r) = 0$ into monotone branches in both t and r.

The topology analysis algorithm in [Keyser et al. 2000] uses the turning points (local maxima and minima) and edge points as the input. The turning points are isolated by finding the common simultaneous solutions between $\mathscr{F}_3(t,r) = 0$ and either $\frac{\partial \mathscr{F}_3}{\partial t}(t,r) = 0$ or $\frac{\partial \mathscr{F}_3}{\partial r}(t,r) = 0$. The multivariate solver of [Elber and Kim 2001] is used to find these local extrema. Edge points are identified by computing all the intersections of $\mathscr{F}_3(t,r) = 0$ 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 [Keyser et al. 2000].

Figure 3 shows the monotone pieces (bounded by a box) obtained by the application of the topology analysis algorithm for the zero-set of $\mathscr{F}_3(t,r)$ shown in Figure 1(b). The algorithm splits the given input function at the midpoint whenever two or more turning points are detected. Hence, two monotone curve segments are visible in the middle region of Figure 3. This, however, does not affect our algorithm in any manner.

5 Applying Constraints

In this section, two constraints are described to trim the individual bisectors represented as the zero-set of the bivariate function $\mathscr{F}_3(t,r)$. 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 2(a), in fact, shows the LL, LR, RL, and RR components of the bisector between the two C^1 planar closed curves.

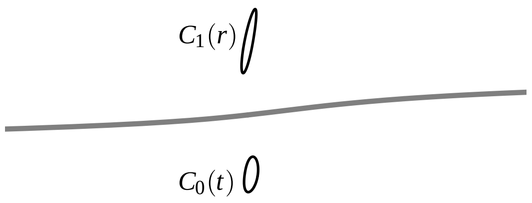


Figure 4: The bisector (in gray) between two closed C^1 planar curves after applying the LL-constraint (see also Figure 2(a)). The bisector extends to ∞ on both sides.

The LL-constraint for two regular C^1 rational curves is reduced to the following two equations:

$$\langle P(t,r) - C_0(t), N_0^L(t)\rangle > 0, \tag{8}$$
$$\langle P(t,r) - C_1(r), N_1^L(r)\rangle > 0, \tag{9}$$

where $N_0^L(t)$ is the normal field defined by $(-y_0'(t), x_0'(t))$. $N_1^L(r)$ has a similar expression and $P(t,r)$ is as defined in Equation (5). Note that $N_0^L(t)$ and $N_1^L(r)$ do not flip directions at inflection points.

The LL-constraint can also be formulated in the following way. Since a point on the bisector is obtained by the intersection of normal lines at the footpoints of the rational curves [Elber and Kim 1998a], the intersection point satisfies the following equation:

$$C_0(t) + \alpha N_0^L(t) = C_1(r) + \beta N_1^L(r), \tag{10}$$

where α and β represent the parameter on the normals of $C_0(t)$ and $C_1(r)$, respectively, whose signs determine whether the intersection point is to the left or right of the curves. Positive values for both α and β imply that the intersection point is to the left of both curves. Thus, the LL-constraint can also be written as follows:

$$\alpha(t,r) > 0, \tag{11}$$
$$\beta(t,r) > 0, \tag{12}$$

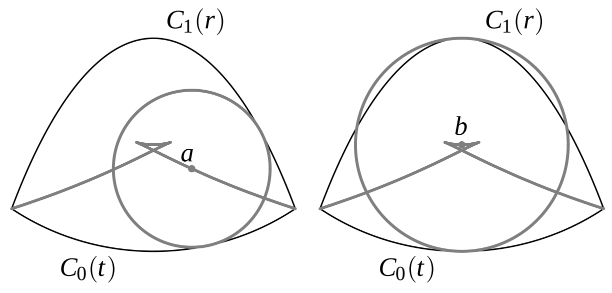
where α and β are given by (see [Elber and Kim 1998a]):

$$\alpha = \alpha(t,r) = \frac{\begin{vmatrix} x_1(r) - x_0(t) & y_1'(r) \\ y_1(r) - y_0(t) & -x_1'(r) \end{vmatrix}}{\begin{vmatrix} -y_0'(t) & y_1'(r) \\ x_0'(t) & -x_1'(r) \end{vmatrix}},$$
$$\beta = \beta(t,r) = \frac{\begin{vmatrix} -y_0'(t) & x_1(r) - x_0(t) \\ x_0'(t) & y_1(r) - y_0(t) \end{vmatrix}}{\begin{vmatrix} -y_0'(t) & y_1'(r) \\ x_0'(t) & -x_1'(r) \end{vmatrix}}. \tag{13}$$

Figure 4 shows the portions of the bisector of Figure 2(a) after applying the LL-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



(a) (b)

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.

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 radii of curvature of the boundary segments.*

For the proof of the lemma, please refer to [Chou 1995].

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:

$$D_0(t,r) = ||P(t,r) - C_0(t)||, \tag{14}$$
$$D_1(t,r) = ||P(t,r) - C_1(r)||. \tag{15}$$

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 larger than the (positive) curvature of one of the curves at the footpoint, we have:

$$1/D_0(t,r) > \left(sign\left\langle C_0''(t), N_0^L(t)\right\rangle\right)\kappa_0(t), \tag{16}$$
$$1/D_1(t,r) > \left(sign\left\langle C_1''(r), N_1^L(r)\right\rangle\right)\kappa_1(r), \tag{17}$$

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:

$$\left\langle P(t,r) - C_0(t), \hat{N}_0(t)/\kappa_0(t)\right\rangle < \left\langle \hat{N}_0(t)/\kappa_0(t), \hat{N}_0(t)/\kappa_0(t)\right\rangle, \tag{18}$$
$$\left\langle P(t,r) - C_1(r), \hat{N}_1(r)/\kappa_1(r)\right\rangle < \left\langle \hat{N}_1(r)/\kappa_1(r), \hat{N}_1(r)/\kappa_1(r)\right\rangle. \tag{19}$$

$N_0^L(t)$ (resp. $N_1^L(r)$) can either be in the same or opposite direction with respect to $\hat{N}_0(t)$ (resp. $\hat{N}_1(r)$). If they are in the opposite direction, the left-hand side of inequalities (18) and (19) will be negative and, therefore, always hold. If $N_0^L(t)$ (resp. $N_1^L(r)$) 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

$$\frac{\left\langle P(t,r) - C_0(t), \hat{N}_0(t)/\kappa_0(t)\right\rangle}{1/\kappa_0(t)^2\left\langle \hat{N}_0(t), \hat{N}_0(t)\right\rangle} < 1, \tag{20}$$
$$\frac{\left\langle P(t,r) - C_1(r), \hat{N}_1(r)/\kappa_1(r)\right\rangle}{1/\kappa_1(r)^2\left\langle \hat{N}_1(r), \hat{N}_1(r)\right\rangle} < 1; \tag{21}$$

or

$$\left\langle P(t,r) - C_0(t), \kappa_0(t)\hat{N}_0(t)\right\rangle < 1, \tag{22}$$
$$\left\langle P(t,r) - C_1(r), \kappa_1(r)\hat{N}_1(r)\right\rangle < 1, \tag{23}$$

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

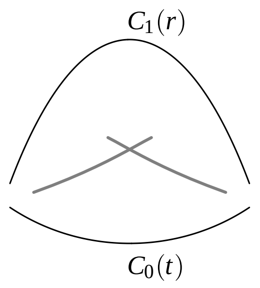


Figure 6: The bisector (in gray) after applying the curvature constraint, for the two curves shown in Figure 1(a).

Figure 6 shows the portions of the bisector obtained after applying the curvature constraint (and after subjecting it to the LL-constraint) for the untrimmed bisector shown in Figure 1(a). Some points on the bisector that do not correspond to the minimality of the distance condition of the Voronoi diagram have been removed. However, further processing to remove all remaining points that do not correspond to the minimality of the distance is required. This is achieved by applying the lower envelope algorithm that is described in the following section.

6 Voronoi Cell Extraction

In this section, the extraction of the Voronoi cell for a C^1-continuous closed curve $C_0(t)$, with respect to the other C^1-continuous closed curves $C_i(r_i)$, $\forall\ i > 0$, is described. The extraction starts with the computation of the untrimmed bisectors using the bivariate function $\mathscr{F}_3(t, r_1)$ described in Section 3 and is followed by splitting $\mathscr{F}_3(t, r_1) = 0$ into monotone pieces described in Section 4. The process of trimming the zero-set of $\mathscr{F}_3(t, r_1)$ using the LL-constraint described in Section 5.1 is then carried out. Subsequently, the curvature constraints described in Section 5.2 trim the zero-set of $\mathscr{F}_3(t, r_1)$ further. The above process is repeated for all pairs of curves $C_0(t)$ and $C_i(r_i)$, $\forall\ i > 1$. The lower envelope algorithm (described in this section) is used at the end to compute the Voronoi cell of $C_0(t)$.

The problem of constructing the Voronoi cell is reduced to the computation of the lower envelope in the following way: Given a curve $C_0(t)$ and the set of curves $C_i(r_i)$, let $D_i(t)$ denote the distance function $D_i(t, r_i)$, described in Section 5.2. Then, the lower envelope of the set $\{D_i(t)\}, i = 1, \ldots, n$, identifies the Voronoi cell of $C_0(t)$. In other words, the problem of computing the Voronoi region of $C_0(t)$ is reduced to the problem of computing the lower envelope of a set of distance functions, $D_i(t)$, over the t-domain of $C_0(t)$. Points on the Voronoi cell are then computed by mapping the points on the lower envelope onto the tr_i-domain.

The concept of computing the lower envelopes has been extensively used for arrangements of line segments [Halperin 1997; Sharir and Agarwal 1995]. For the benefit of readers, the basic lower envelope algorithm is introduced in Section 6.1, following [Sharir and Agarwal 1995]. The description of the basic predicates needed to implement the algorithm over rational curves is then presented in Section 6.2, followed by the details of using the predicates to generate the Voronoi cell, in Section 6.3.

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. Thereafter, 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 also 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.

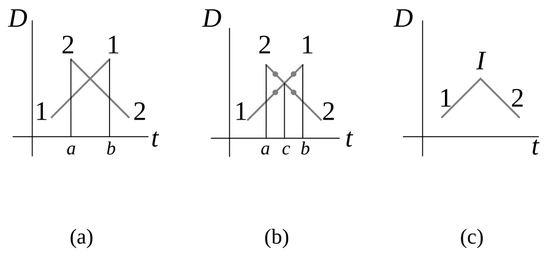


Figure 7: Illustration of the lower envelope algorithm.

The lower envelope algorithm for line segments is illustrated in Figure 7(a). Consider the two segments 1-1 and 2-2, as 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 parametric region is ab; hence they are split at parameters a and b. Further processing is required only for the portions where the overlapping of parameters occurs. The portion of overlapping parametric regions is tested for intersections. If intersection points exist, they are again split at these 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 Rational Curves

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 of each of these functions may differ. 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 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 parameters 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 value.

- A function for splitting a t-monotone curve into two subcurves at a new t-parameter.

For line segments or 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.

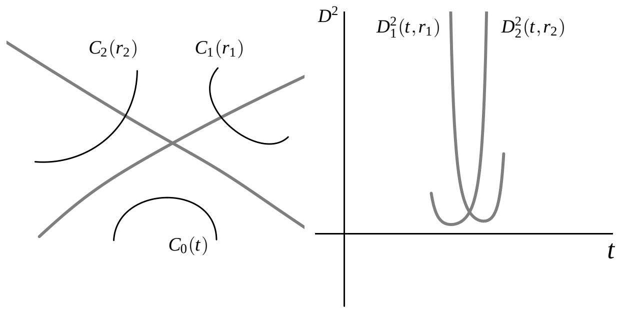
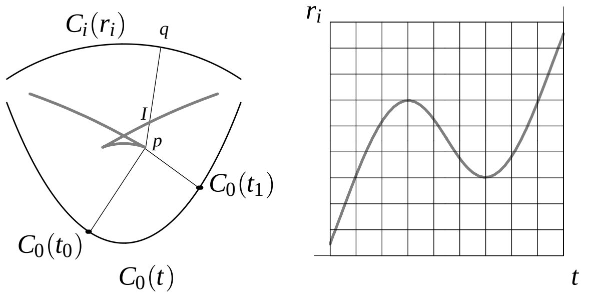


Figure 8: (a) The bisectors (in gray) between the pairs $(C_0(t),C_1(r_1))$ and $(C_0(t),C_2(r_2))$ of C^1 planar curves. (b) The squared distance functions $D_1^2(t,r_1)$ and $D_2^2(t,r_2)$ of the bisectors of the respective pairs of curves.



(a) The bisector (in gray) between two rational C^1 planar curves.

(b) The zero-set in the domain of $\mathscr{F}_3(t,r_i)$, defines the bisector.

Figure 9: The bisector of a non-convex C_0 curve can have t-parameters corresponding to non-minimal points.

6.3 Voronoi Cell via Lower Envelope

Given a tr_i-monotone curve $\mathscr{F}_3(t,r_i)=0$, for every t value there is a single corresponding r_i value. Furthermore, there is a correspondence between $\mathscr{F}_3(t,r_i)=0$ and $D_i(t)$. If $\mathscr{F}_3(t,r_i)=0$ 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_i value and hence a single corresponding distance value. Therefore, we can apply the lower envelope algorithm to the distance functions defined over monotone bisectors of the form $\mathscr{F}_3(t,r_i)=0$, 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 $D_i(t,r_i)$ may be used instead. Figure 8(b) shows the squared distance functions $D_1^2(t,r_1)$ and $D_2^2(t,r_2)$ for the bisectors 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 $\mathscr{F}_3(t,r_i)$ and $D_i^2(t,r_i)$ functions.

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

$$||D_i(t,r_i)||^2 = ||D_j(t,r_j)||^2, \tag{24}$$
$$\mathscr{F}_3(t,r_i)=0, \tag{25}$$
$$\mathscr{F}_3(t,r_j)=0. \tag{26}$$

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 tr_i-monotone $\mathscr{F}_3(t,r_i)=0$ segments) and then comparing the function values of $D_i^2(t,r_i)$ and $D_j^2(t,r_j)$ at the respective parameter values.

To split a $D_i^2(t)$ function into two pieces at a given t-parameter, all that is needed is to split its corresponding tr_i-monotone implicit curve $\mathscr{F}_3(t,r_i)=0$ at that parameter.

The result of the lower envelope algorithm is a list of tr_i-monotone implicit curves $\mathscr{F}_3(t,r_i)=0$ that are split at t-parameters of equidistant bisector points. The union of the curves $\mathscr{F}_3(t,r_i)=0$ represents the Voronoi cell of $C_0(t)$.

The above lower envelope algorithm results in the Voronoi cell of $C_0(t)$ when $C_0(t)$ is convex. In the case of a convex $C_0(t)$, for *every* t-parameter, the footpoints corresponding to the points on the trimmed bisector are also the minimal distance point of $C_0(t)$ (see Figure 1). However, when $C_0(t)$ is non-convex, this may not be the case. There may be some t-intervals that have no corresponding lower envelopes. For example, the footpoint (such as $C_0(t_0)$ in Figure 9(a)) of the untrimmed bisector point (p in Figure 9(a)) does not correspond to the minimal distance with respect to $C_0(t)$. In other words, there exist points such as $C_0(t_1)$ (in Figure 9(a)) closer to p than its footpoint $C_0(t_0)$.

A footpoint $C_0(t_0)$ of p that does not correspond to minimal distance with respect to $C_0(t)$ implies that p is inside the Voronoi region of $C_0(t)$ (with respect to $C_i(r_i)$) and not on its boundary. Let r_{i0} denote the parameter of $C_i(r_i)$ that corresponds to t_0 (and to p), and let q be the point $C_i(r_{i0})$ (see Figure 9(a)). Clearly, q is outside the Voronoi region of $C_0(t)$. Moreover, as the trimmed bisector is continuous and divides the plane into two Voronoi regions, the line segment pq connecting p with q intersects the boundary of the Voronoi region (i.e., the trimmed bisector) in at least one point (such as I in Figure 9(a)). The intersection point I also has q as its foot point, since the line segment pq is on the normal line to $C_i(r_{i0})$. Thus, the parameter r_{i0} corresponding to t_0 also corresponds to at least one other t-parameter with a smaller distance to q (see Figure 9(a)).

Therefore, repeating the lower envelope procedure for the r_i-parameter eliminates the t-parameters that do not correspond to minimal distance. In fact, by applying the lower envelope in t we trim away all portions of the untrimmed bisector that are inside the Voronoi region of $C_i(r_i)$ and by applying the lower envelope in r_i we trim away all portions of the untrimmed bisector that are inside the Voronoi region of $C_0(t)$. Hence, the trimmed bisector for a pair of curves $C_0(t)$ and $C_i(r_i)$ is the result of applying the lower envelope procedure both in t and in r_i.

7 The Voronoi Cell for Piecewise C^1-Continuous Curves

When the curve is only piecewise C^1-continuous, bisectors for point/curve pairs have to be considered as well. The bisector of a point/curve pair is rational [Farouki and Johnstone 1994]. In this section, the bisector generation for a point/curve pair is described first (Section 7.1). Trimming of the PCB is then described using the orientation (Section 7.2.1) and curvature constraints (Section 7.2.2), reminiscent of the CCB trimming case (Sections 5.1 and 5.2). The application of these constraints to PCB is discussed in Section 7.2.3. The minimum distance/radial lower envelope of these trimmed bisectors, is described in Section 7.3. Finally, cases where either $C_0(t)$ or $C_i(t)$, $i > 0$, are piecewise C^1-continuous are discussed in Sections 7.4 and 7.5, respectively. In what follows, let $P = (p_x, p_y)$ denote a tangent discontinuity point and $C(t) = (x(t), y(t))$ denote a C^1 continuous curve segment.

7.1 Point/Curve Bisector

A point B on a PCB has to satisfy the following two equations:

$$\left\langle B - C(t), C^{'}(t) \right\rangle = 0, \tag{27}$$

$$\left\langle B - \frac{C(t)+P}{2}, C(t) - P \right\rangle = 0. \tag{28}$$

Equation (27) indicates that the vector from the bisector point B to its footpoint on the curve must be orthogonal to the tangent of the curve at the footpoint. Equation (28) indicates that the bisector point B has to be equidistant to both P and $C(t)$.

Equations (27) and (28) are linear in B. Hence, they can be solved symbolically using Cramer's rule to identify the bisector points $B(t) = (b_x(t), b_y(t))$ as follows:

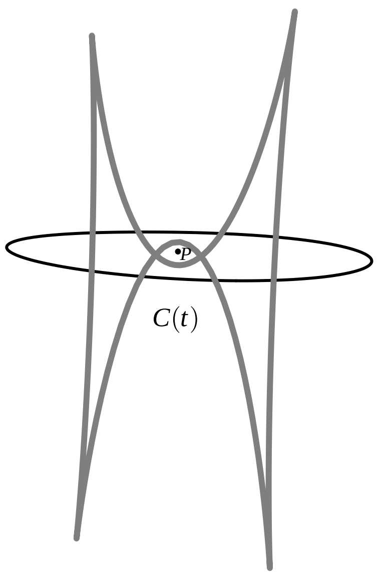
$$b_x(t) = \frac{\begin{vmatrix} x_(t)x'(t) + y(t)y'(t) & y'(t) \\ (\frac{p_x + x(t)}{2})(x(t) - p_x) + (\frac{p_y + y(t)}{2})(y(t) - p_y) & y(t) - p_y \end{vmatrix}}{\begin{vmatrix} x'(t) & y'(t) \\ x(t) - p_x & y(t) - p_y \end{vmatrix}},$$

$$b_y(t) = \frac{\begin{vmatrix} x'(t) & x_(t)x'(t) + y(t)y'(t) \\ x(t) - p_x & (\frac{p_x + x(t)}{2})(x(t) - p_x) + (\frac{p_y + y(t)}{2})(y(t) - p_y) \end{vmatrix}}{\begin{vmatrix} x'(t) & y'(t) \\ x(t) - p_x & y(t) - p_y \end{vmatrix}}. \tag{29}$$

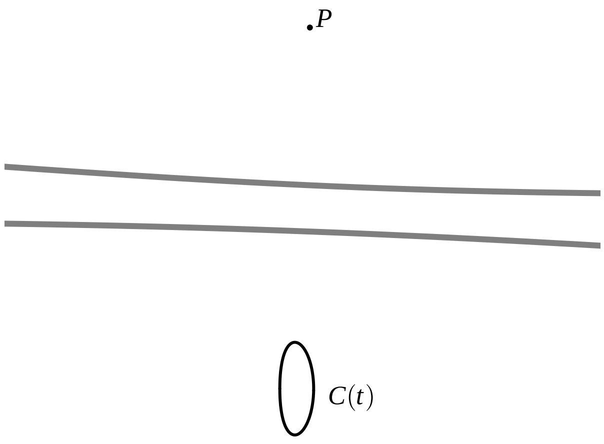
Clearly, Equation (29) representing $B(t)$ is rational, provided $C(t)$ is rational. Figures 10(a) and 10(b) shows two examples of untrimmed bisectors (in gray) between a point P and a closed C^1 continuous curve, $C(t)$.

7.2 The Point/Curve Bisector's Constraints

In this section, two constraints, similar to the ones applied to bisectors of pairs of curves, are described to trim the individual PCB. The constraints are based on the orientation of the input curve and its curvature field. The constraints can directly be applied to the obtained PCB without the monotone splitting that was required in the CCB case. The function B is monotone, following the single parameter t, from the curve $C(t)$. Hence, the bisector function B will be denoted $B(t)$.



(a)



(b)

Figure 10: Two examples of a bisector (in gray) between a planar closed rational curve $C(t)$ and a point P. In (a), the point is inside the closed curve, whereas in (b), P is outside $C(t)$.

7.2.1 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* constraint generates points corresponding to bisector points that lie to the left of the curve. Hence it can also be termed as L-constraint where L implies Left. In a similar manner, we can construct the other orientation constraint as R-constraint.

The L-constraint for a regular rational curve is reduced to the following equation:

$$\left\langle B - C(t), N^L(t) \right\rangle > 0, \qquad (30)$$

where $N^L(t)$ is the normal field of $C(t)$ defined as $(-y'(t), x'(t))$.

7.2.2 Curvature Constraint

As in the CCB case, this constraint eliminates some (though not all) points on the bisector that do not satisfy the minimal distance requirement of the Voronoi diagram. The curvature constraint (as in the curve/curve case) can be reduced to the following equation:

$$\left\langle B - C(t), \kappa(t)\hat{N}(t) \right\rangle < 1, \qquad (31)$$

where $\kappa(t)$ is the curvature function of $C(t)$ and $\hat{N}(t)$ denotes its unit normal vector field.

7.2.3 Applying the Constraints

In the case of CCBs, the constraints are applied on the zero-set of the bivariate function $\mathscr{F}_3(t,r)$. Since a PCB is a rational function, the constraints can directly be applied in the bisector function $B(t)$ that was obtained symbolically (Equation (29)).

Zeros in a PCB are defined to be the points on the PCB where the sign of the constraint equations changes. Poles, where the PCB has points at infinity, are also included in this definition. In the case of a PCB, when the given point P lies outside the given curve $C(t)$, the zeros and poles are equivalent, i.e., the sign change of the PCB occurs only at the poles where the difference vector $C(t) - P$ is parallel to the tangent $C'(t)$. Hence, the zeros of the L-constraint occur when the denominator of Equation (29) vanish:

$$\begin{vmatrix} x'(t) & y'(t) \\ x(t) - p_x & y(t) - p_y \end{vmatrix} = 0. \qquad (32)$$

The zeros for the curvature constraint can be obtained by solving the following equation:

$$\left\langle B - C(t), \kappa(t)\hat{N}(t) \right\rangle - 1 = 0. \qquad (33)$$

The zeros of the curvature are the cusp points in the bisector function $B(t)$ (see Equation (29) and Figure 10(a)). The cusp points can be identified by equating the first derivative of $B(t)$ to zero, i.e., $B'(t) = 0$. The degree of $B'(t)$ is less than the degree of Equation (33) and hence more amenable to implementation.

The bisector function is then subdivided at the zeros. The subdivided portions are subjected to the constraints (Equations (30) and (31)) at the mid-parameter values. The subdivided portions that do not satisfy the constraints are purged away.

Figures 11(a) and 11(b) show the resulting bisector after applying the constraints (their untrimmed version is shown in Figures 10(a) and 10(b)).

7.3 Minimum Distance / Radial Lower Envelope

The trimmed bisector shown in Figure 11(a) contains some redundant segments that do not correspond to the minimum that the Voronoi cell definition demands. The following procedure can be used to obtain the true minimum.

- Identify the intersection points between the remaining portions of the bisector curves and split them at these points.
- At the mid-parameter values, check whether the footpoint of the bisector point is the minimum (in distance) to the curve $C(t)$. Portions of the bisector that do not correspond to the minimum distance are purged away.

In the above procedure, the routine ‘identifying the minimum distance between a point and a curve’ is applied only at the mid-parameter values and hence it is reasonably efficient.

Figure 12 shows the trimmed bisector after applying the ‘minimum distance’ procedure (the untrimmed version is shown in Figure 10(a)).

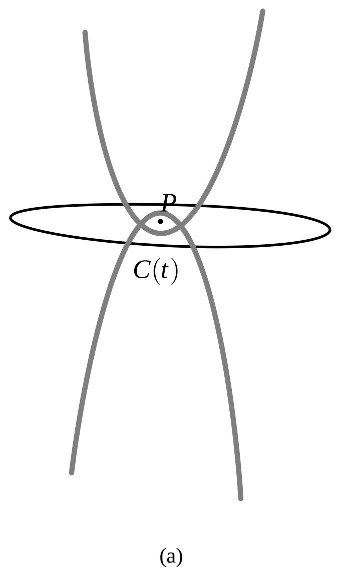
Alternatively, the lower envelope described in Section 6.1 can be applied radially around the tangent discontinuity point. Such a lower envelope will be termed *radial lower envelope*. In the radial lower envelope, the distance function is defined over the angle θ between a point $B_i(r_i)$ and P, similar to the definition of the distance function over the t-domain in Section 6.1 (refer to Figure 13). In Figure 13, the angle θ is shown to be measured from the horizontal line PL. However, in general, any reference line could be used. The basic functions needed for the radial lower envelope, defined using the angle θ and the curves $B_i(r_i)$, are as follows:

- Finding the intersections of the distance functions corresponds to computing the intersections of the B_i curves (such as I_1, I_2, etc. in Figure 13) and sorting them radially, i.e., according to the angle θ.
- Comparing two distance functions at a given θ-parameter corresponds to finding the points of B_i and B_j that intersect the ray emanating from P with angle θ and comparing their distances to P.
- Splitting a curve at a given θ-parameter corresponds to splitting it at its intersection point with the θ-angle ray emanating from P.

Figure 14 shows one application of a radial lower envelope. Figure 14(a) shows the trimmed PCBs. Application of a radial lower envelope results in the bisector segments shown in Figure 14(b). Further trimming of the resultant bisector is possible, to keep it within the bounds of the two limiting normals of the curve at the tangent discontinuity location, P. This step is shown in Figure 14(c).

7.4 Voronoi Cell When Only $C_0(t)$ is Piecewise C^1-Continuous

Assume curve $C_0(t)$ is the only piecewise C^1-continuous curve in the input set of curves, and $C_i(t)$, $i > 0$ are all C^1 continuous. To obtain the Voronoi cell of $C_0(t)$, the following procedure is applied.



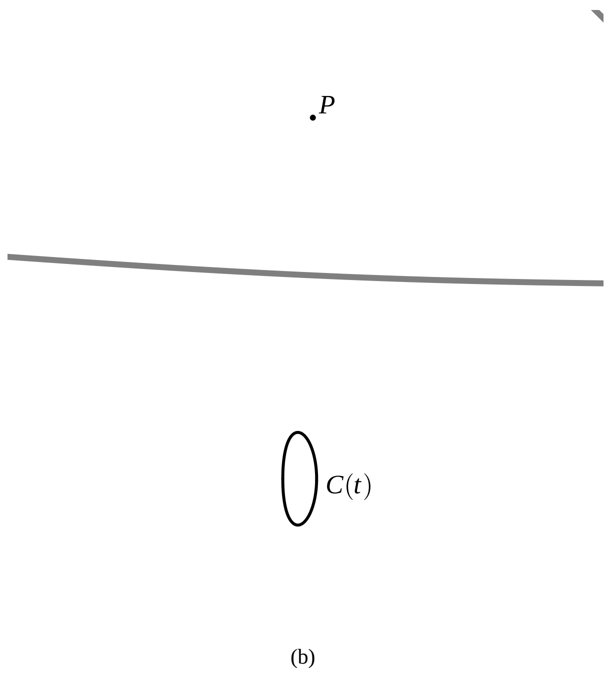


Figure 11: The bisector (in gray) between a planar closed rational curve $C(t)$ and a point P after applying the constraints (see also Figures 10(a) and 10(b)).

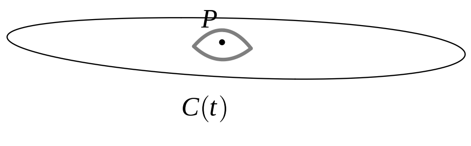


Figure 12: The trimmed bisector (in gray) between a planar closed rational curve $C(t)$ and a point P.

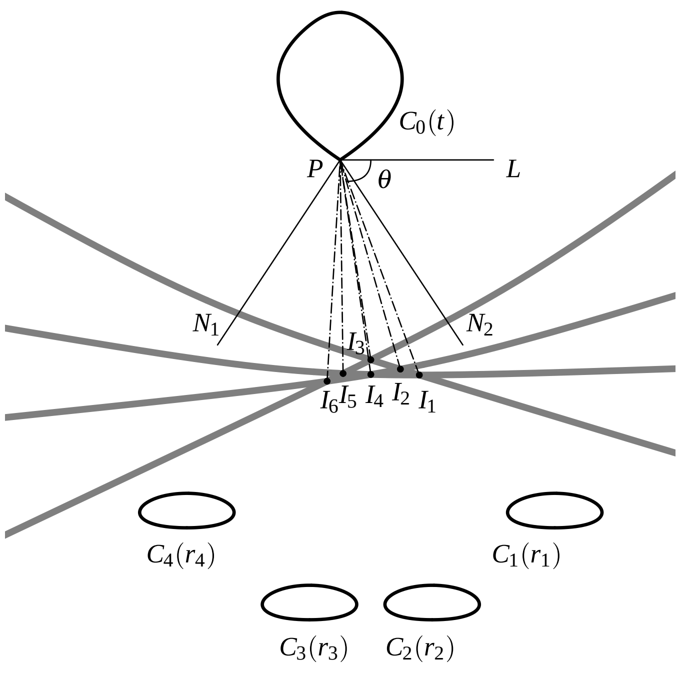


Figure 13: Radial lower envelope.

- Split the curve $C_0(t)$ at the tangent discontinuity points.
- Compute the CCBs and apply the curve/curve trimming procedure and the lower envelope to get the bisector segments of the curve/curve case (Figure 15(a)).
- Compute the PCBs and apply the point/curve trimming procedure and the radial lower envelope to get the bisector segments of the point/curve case (Figure 15(b)).
- The Voronoi cell for piecewise C^1-continuous $C_0(t)$ is the union of the above two sets of bisector segments (Figure 15(c)).

Whenever both CCB and PCB exist and in order to distinguish between them, the CCB is shown in gray and the PCB in black, as in Figures 15 and 16).

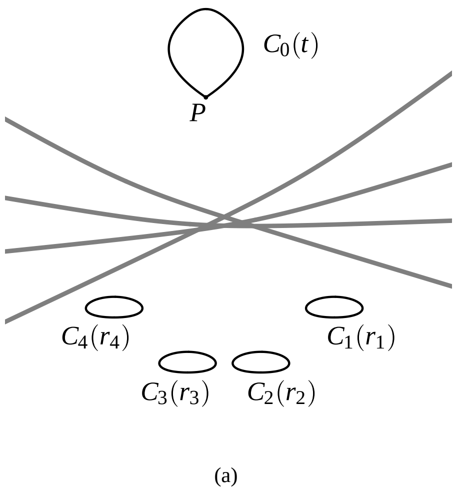
7.5 Voronoi Cell when $C_i(r_i)$ is Piecewise C^1-Continuous

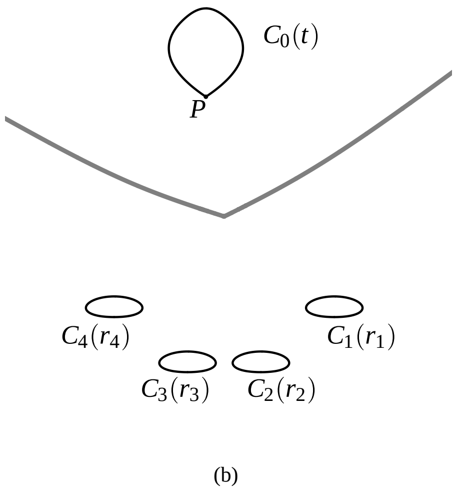
Computing the Voronoi cell for piecewise C^1-continuous curves $C_i(r_i)$ amounts to computing the lower envelope of the trimmed CCBs and the trimmed PCBs (induced by the tangent discontinuity points of $C_i(r_i)$). Hence the lower envelope routine described in Section 6.1 needs to be extended to handle PCBs as well.

The bisector curve $B_i(t)$ derived in Section 7.1 is parameterized with the parameter t of $C_0(t)$. Therefore, the squared distance function:

$$D_i^2(t) = \langle B_i(t) - C(t), B_i(t) - C(t) \rangle, \qquad (34)$$

is well defined over the t-domain. Such a squared distance function, originating from a PCB, can easily be computed at a given t-parameter and the distance can be compared to another squared





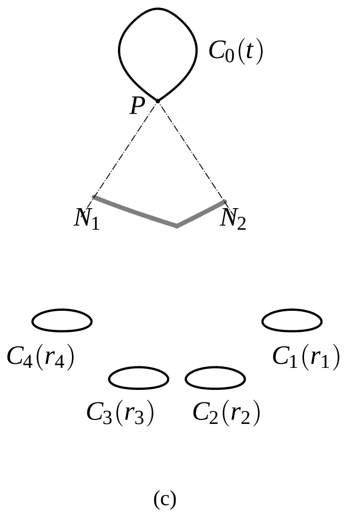
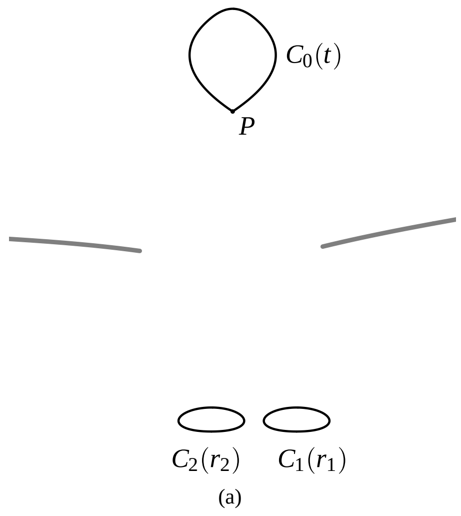
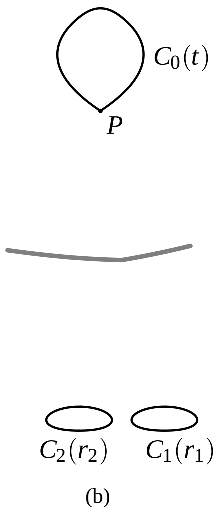


Figure 14: (a) The trimmed bisector (in gray) of the PCBs. (b) Portion of the bisector after applying the radial lower envelope. (c) Trimmed radial lower envelope using the bounding normals of the point.





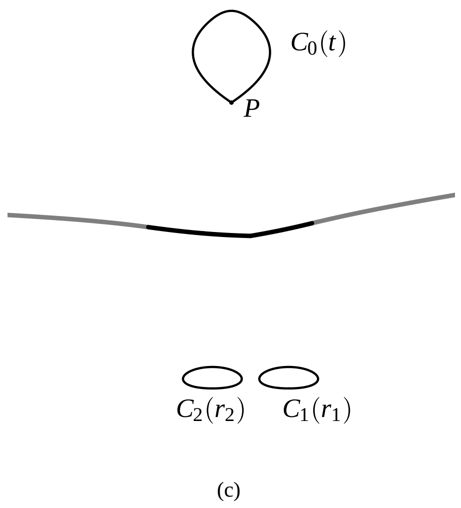


Figure 15: (a) The portion of the Voronoi cell due to CCBs. (b) The portion of the Voronoi cell due to PCBs. (c) The complete Voronoi cell of a piecewise C^1-continuous $C_0(t)$.

distance value at the same t-parameter, either originating from a PCB (computed in the same way) or from a CCB (computed as described in Section 6.3). In a similar way, splitting a function $D_i^2(t)$ originating from a PCB, at a given t-parameter corresponds to splitting the bisector curve $B_i(t)$ at that parameter.

In identifying the intersection points of the bisectors, corresponding to the intersections of the squared distance functions, we distinguish three cases.

- Intersections between two CCBs.
- Intersections between a CCB and a PCB.
- Intersections between two PCBs.

Section 6.3 described how the first case is carried out. The second case, the intersections between a CCB $B_i(t, r_i)$ (represented by the implicit function $\mathscr{F}_3(t, r_i)$) and a PCB $B_j(t)$, can be identified using the following equations (in t and r_i):

$$D_i^2(t, r_i) = D_j^2(t), \tag{35}$$

$$\mathscr{F}_3(t, r_i) = 0, \tag{36}$$

where $D_j^2(t)$ is the univariate squared distance function defined in Equation (34) and $D_i^2(t, r_i)$ is the bivariate squared distance function described in Section 6.3. The third case, intersections between two PCBs, can be identified by solving the univariate equation (in t):

$$D_i^2(t) = D_j^2(t). \tag{37}$$

Alternatively, it can be solved using a regular curve/curve intersection procedure for two rational bisector curves $B_i(t)$ and $B_j(t)$.

Thus, the complete algorithm for computing the Voronoi cell for piecewise C^1-continuous curves $C_i(r_i)$ is:

- Split the curves $C_i(r_i)$ at the tangent discontinuity points.
- Compute the trimmed CCBs corresponding to the C^1-continous portions of C_i.
- Compute the trimmed PCBs corresponding to the tangent discontinuity points of C_i.
- Apply the lower envelope algorithm to the resulting set of PCBs and CCBs.

Figure 16 shows such an example.

8 Results and Discussion

We implemented the proposed algorithm for extracting the Voronoi cell of a closed curve using the IRIT [Elber 2002] modeling environment. In this section, results from some test cases are presented. The input rational curves are represented as Bézier or B-spline curves. At no time are these (piecewise) rational curves approximated into line segments, circular arcs, or a different representation. Hence, we are working with only a few (piecewise) rational input entities as opposed to a large number of (linear and/or circular) input generators, which may be the case when approximation schemes are employed. 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 *Deg*), the number of control points used (denoted as *CP*) and the degree of $\mathscr{F}_3(t, r)$ (denoted as *Deg*$\mathscr{F}_3$) computed as given in [Elber and Kim 1998a] are also

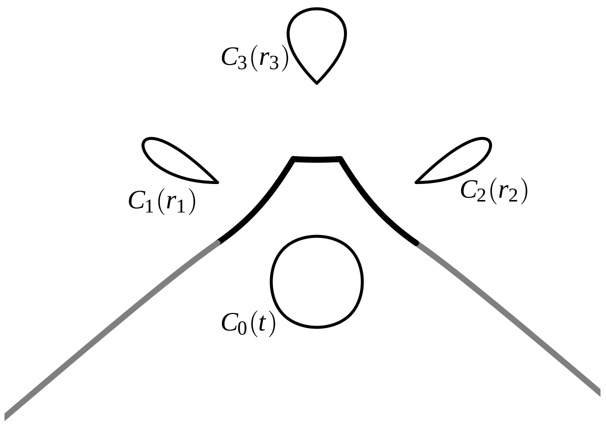


Figure 16: Voronoi cell for piecewise C^1-continuous $C_i(r_i)$'s. $C_0(t)$ has $Deg = 3$, and $CP = 7$, $C_1(r_1)$, $C_2(r_2)$ and $C_3(r_3)$ have $Deg = 3$, $CP = 4$. $Deg\mathscr{F}_3 = 10$

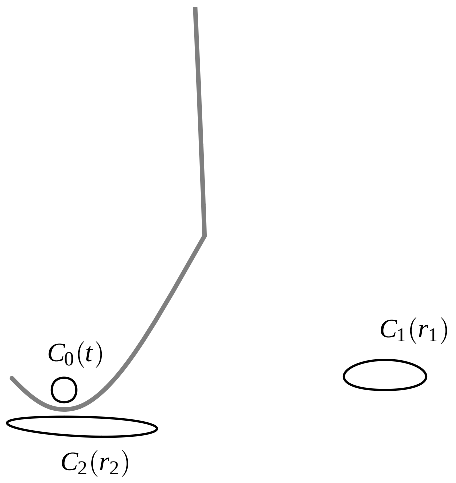


Figure 17: Voronoi cell (in gray) of $C_0(t)$. $C_0(t)$ has $Deg = 3$, and $CP = 7$, $C_1(r_1)$ and $C_2(r_2)$ have $Deg = 5$, $CP = 5$. $Deg\mathscr{F}_{3max} = 18$.

provided for each of the figures. They can sometimes denote the maximum value of the input curves which can be identified by the tag *max*. Figure 17 shows the Voronoi cell of a closed curve $C_0(t)$. Figure 18 shows the Voronoi cell of $C_0(t)$, where the input geometry consists of four closed curves. Figures 19 and 20 show two more test cases where all objects have convex profiles. The algorithm is demonstrated for non-convex curves as well, in Figures 21 to 24. In Figures 23 and 24, the curve $C_0(t)$ is non-convex. A Voronoi cell for piecewise C^1-continuous curves $C_i(t)$, $i > 0$, is shown in Figure 25. Another result for piecewise C^1-continuous curves $C_i(t)$, $i > 0$, is presented in Figure 16.

8.1 Discussion

The following are the major steps of our algorithm:

1. formulate the bivariate function $\mathscr{F}_3(t, r)$;
2. split the zero-sets of $\mathscr{F}_3(t, r)$'s into monotone pieces;
3. apply the trimming of CCBs i.e. in $\mathscr{F}_3(t, r)$'s;

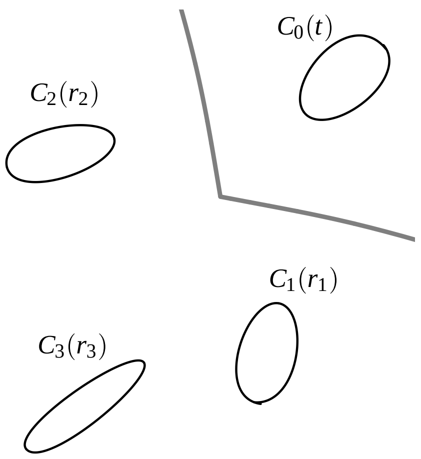


Figure 18: Voronoi cell (in gray) of C_0(t). All the input curves have $Deg = 5$, and $CP = 5$. $Deg\mathscr{F}_3 = 18$.

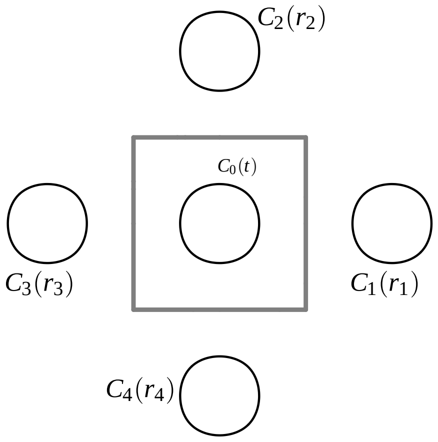


Figure 19: Voronoi cell (in gray) of C_0(t). All the input curves have $Deg = 3$, and $CP = 7$. Here, the bisector curves degenerate into lines. $Deg\mathscr{F}_3 = 10$.

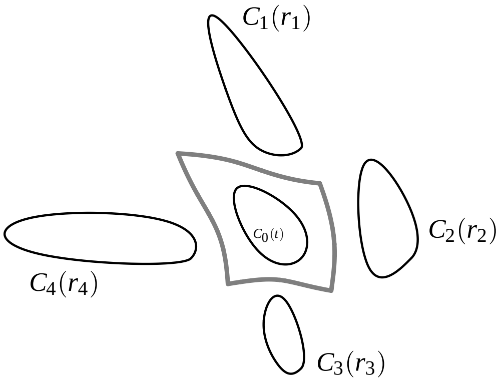


Figure 20: Voronoi cell (in gray) of C_0(t). All the input curves have $Deg = 3$, $CP = 7$. $Deg\mathscr{F}_3 = 10$.

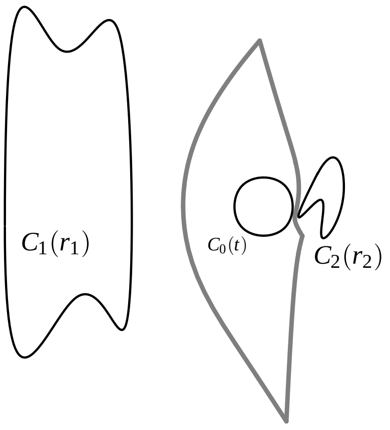


Figure 21: Voronoi cell (in gray) of C_0(t). All the input curves have $Deg = 3$. However, $C_0(t)$ and $C_2(r_2)$ have $CP = 7$, whereas $C_1(r_1)$ has $CP = 8$. $Deg\mathscr{F}_{3max} = 10$.

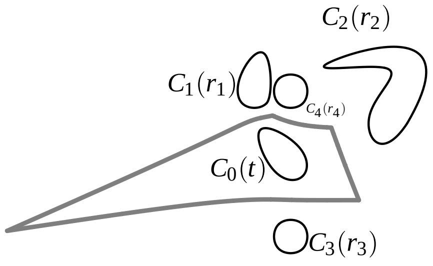


Figure 22: Voronoi cell (in gray) of C_0(t). All the input curves have $Deg = 3$, $CP = 7$. $Deg\mathscr{F}_3 = 10$.

4. compute PCBs, for piecewise C^1-continuous curves;
5. apply trimming of PCBs;
6. compute the lower envelope, extracting the Voronoi cell; and
7. compute radial lower envelope(s), if $C_0(t)$ is piecewise C^1-continuous.

Our experimental results indicate that all the above steps are reasonably efficient. Computation of the Voronoi cell took from a few seconds to a few minutes. All the experiments were carried out on an Intel Pentium 4 1.8GHz computer with 256MB RAM using the IRIT [Elber 2002] environment.

Even though the algorithm has been implemented using floating point arithmetic, it has been found to be reasonably robust. Splitting the zero-set of a bivariate function $\mathscr{F}_3(t,r)$ into monotone pieces involves the computation of turning points, which can be computed to arbitrary precision. Though the splitting procedure described in [Keyser et al. 2000] uses exact arithmetic, our experiments indicate that the splitting worked well with floating point arithmetic in all cases that have been tested so far. Exact arithmetic 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 computation 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 considered precise up

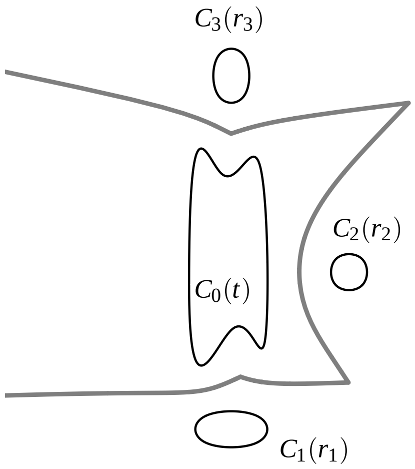


Figure 23: Voronoi cell (in gray) of a non-convex C_0(t). All the input curves have $Deg = 3$. However, $C_0(t)$ has $CP = 8$, whereas the other curves have $CP = 7$. $Deg\mathscr{F}_3 = 10$.

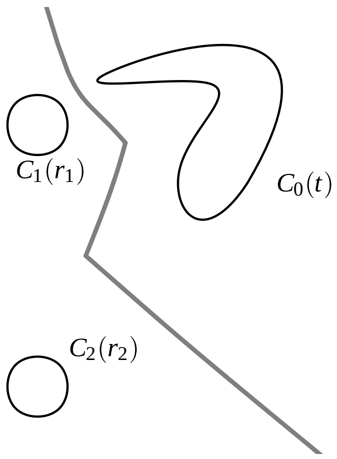


Figure 24: Voronoi cell (in gray) of a non-convex C_0(t). All the input curves have $Deg = 3$, $CP = 7$. $Deg\mathscr{F}_3 = 10$.

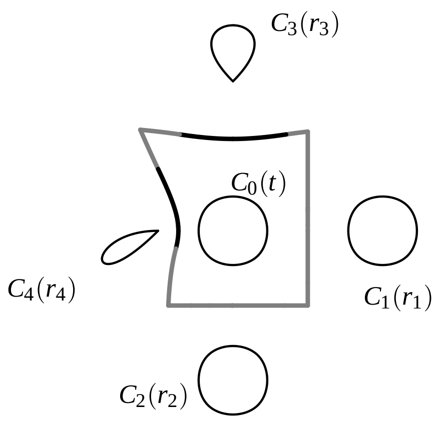


Figure 25: Voronoi cell (in gray) of C_0(t) for some piecewise C^1-continuous $C_i(r_i)$'s. All the input curves have $Deg = 3$. $CP = 7$ for $C_0(t)$, $C_1(r_1)$ and $C_2(r_2)$. $C_3(r_3)$ and $C_4(r_4)$ have $CP = 4$. $Deg\mathscr{F}_3 = 10$.

to machine precision. Moreover, the algorithm does not require a post-processing stage to remove the artifacts that would have been created by an approximation of the rational curves. The algorithm generates 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 18, 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.

The medial axis [Ramamurthy and Farouki 1999; Ramanathan and Gurumoorthy 2003] for a set of curves connected at the end points and forming a closed domain can be generated using the presented Voronoi cell generation procedure. Making one of the curves in the set as $C_0(t)$, in turn, will generate the medial axis. Figure 26 shows the medial axis transform (MAT) of a closed domain consisting of free-form curves using the Voronoi cell algorithm described in this paper. Computation of each Voronoi cell took around ten seconds. However, it is clear that, not all the curves are to be used as $C_0(t)$ in order to generate the medial axis. A more efficient generation scheme of the medial axis of freeform planar curves is currently under investigation.

Since the definition of a Voronoi diagram precludes self-Voronoi edges (traditionally, a Voronoi diagram is defined only for distinct entities), no self-bisectors are considered in this work. Yet, the bivariate function $\mathscr{F}_3(t,r)$ can be used to generate self-bisectors (bisectors of a curve with itself), and the presented algorithm can also be extended to generate self-Voronoi edges. This work can also be used to generate Voronoi diagrams or MATs for domains bounded by piecewise C^1-continuous rational curves and vertices that are concave or reflex. These ideas are currently under investigation.

9 Conclusions

This paper presented an algorithm for generating the Voronoi cells of a set of rational planar closed curves. It is shown that the symbolically computed bisectors in suitable parameter space for pairs of rational curves can be used to generate the Voronoi cells of pla-

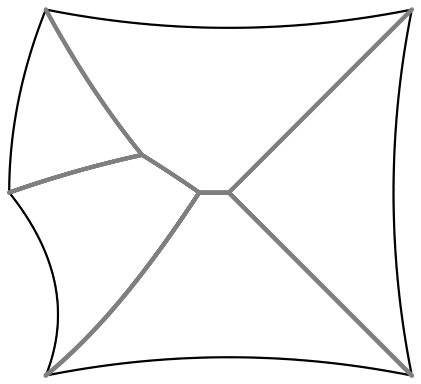


Figure 26: Medial axis transform (MAT) of a closed domain consisting of free-form curves. The input curves are quadratic Bézier curves.

nar 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 segments, thereby eliminating the post-processing stage required 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 a Voronoi diagram and medial axis of an object bounded by piecewise rational curves. This is currently under investigation.

In conclusion, it should be noted that the proposed algorithm does not approximate the bisector segments and the output is the boundary of a Voronoi cell that is represented as a piecewise implicit form in tr-parameter space that is arbitrarily precise. The presented approach might also be adapted to the use of exact arithmetic [Yap 1997], using packages such as bignums [GMP 2002], and using long floats [CORE 2004].

Acknowledgments

This research was supported in part by the Israel Science Foundation (grant No. 857/04), in part by an European FP6 NoE grant 506766 (AIM@SHAPE), and in part by the Korean Ministry of Information and Communication (MIC) under the program of IT Research Center on CGVR.

References

ALT, H., AND SCHWARZKOPF, O. 1995. The Voronoi diagram of curved objects. *11th Symposium on Compuational Geometry*, 89–97.

ARMSTRONG, C. G. 1994. Modeling requirements for finite element analysis. *Computer Aided Design 26*, 7 (July), 573–578.

AURENHAMMER, F. 1991. A survey of fundamental geometric data structure. *ACM Computing Surveys 23*, 3 (September), 345–405.

BAJA, G. S., AND THIEL, E. 1994. (3-4)-weighted skeleton decomposition for pattern representation and description. *Pattern Recognition 27*, 8, 1039–1049.

BLUM, H. 1967. A transformation for extracting new descriptors of shape. *Models for the Perception of Speech and Visual Form*, 362–381.

BLUM, H. 1973. Biological shape and visual science (Part I). *Journal of Theoretical Biology 38*, 205–287.

CHOU, J. J. 1995. Voronoi diagrams for planar shapes. *IEEE Computer Graphics and Applications* (March), 52–59.

CORE, 2004. http://cs.nyu.edu/exact/core/.

ELBER, G., AND KIM, M.-S. 1998. Bisector curves for planar rational curves. *Computer-Aided Design 30*, 14, 1089–1096.

ELBER, G., AND KIM, M.-S. 1998. The bisector surface of rational space curves. *ACM Transaction on Graphics 17*, 1 (January), 32–39.

ELBER, G., AND KIM, M.-S. 2001. Geometric constraint solver using multivariate rational spline functions. In *Proceedings of the Symposium on Solid Modeling and Applications*, D. Anderson and K. Lee, Eds., ACM, 1–10.

ELBER, G. 2002. *IRIT Users's Manual*, 9.0 ed., June.

FAROUKI, R., AND JOHNSTONE, J. 1994. The bisector of a point and a plane parametric curve. *Computer-Aided Geometric Design 11*, 2, 117–151.

FAROUKI, R., AND RAMAMURTHY, R. 1998. Specified-precision computation of curve/curve bisectors. *International Journal of Computational Geometry and Applications 8*, 5 and 6, 599–617.

GMP, 2002. GNU multiple precision arithmetic library. http://www.swox.com/gmp/.

GURSOY, H. N., AND PATRIKALAKIS, N. M. 1992. An automatic coarse and finite surface mesh generation scheme based on medial axis transform: Part 1 Algorithms. *Engineering with Computers 8*, 121–137.

HALPERIN, D. 1997. *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL.

HANNIEL, I., RAMANATHAN, M., ELBER, G., AND KIM, M.-S. 2005. Precise Voronoi cell extraction of freeform rational planar closed curves. In *Proceedings of the Symposium on Solid and Physical Modeling*, V. Shapiro and L. Kobbelt, Eds., ACM, 51–59.

HELD, M. 1998. Voronoi diagram and offset curves of curvilinear polygons. *Computer-Aided Design 30*, 4, 287–300.

HOFFMANN, C., AND VERMEER, P. 1991. Eliminating extraneous solutions in curve and surface operation. *International Journal of Computational Geometry and Applications 1*, 1, 47–66.

KAO, J.-H. 1999. *Process planning for additive/subtractive solid free-form fabrication using medial axis transform*. PhD thesis, Department of Mechanical Engineering, Stanford University.

KEYSER, J., CULVER, T., MANOCHA, D., AND KRISHNAN, S. 2000. Efficient and exact manipulation of algebraic points and curves. *Computer-Aided Design 32*, 11 (15 September), 649–662.

KIM, D., HWANG, I., AND PARK, B. 1995. Representing the Voronoi diagram of a simple polygon using rational quadratic bèzier curves. *Computer-Aided Design 27*, 8, 605–614.

LAVENDER, D., BOWYER, A., DAVENPORT, J., WALLIS, A., AND WOODWARK, J. 1992. Voronoi diagrams of set theoretic solid models. *IEEE Computer Graphics and Applications* (September), 69–77.

MONTANARI, U. 1969. Continuous skeletons from digitized images. *Journal of the Association for Computing Machinery 16*, 4 (October), 534–549.

O'ROURKE, J. 1993. *Computational Geometry in C*. Cambridge University Press.

RAMAMURTHY, R., AND FAROUKI, R. 1999. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries I: Theoretical foundations. *Journal of Computational and Applied Mathematics 102*, 119–141.

RAMANATHAN, M., AND GURUMOORTHY, B. 2003. Constructing medial axis transform of planar domains with curved boundaries. *Computer-Aided Design 35*, 7 (June), 619–632.

SHARIR, M., AND AGARWAL, P. K. 1995. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press.

SRINIVASAN, V., AND NACKMAN, L. R. 1987. Voronoi diagram for multiply-connected polygonal domains I: Algorithm. *IBM Journal of Research and Development 31*, 3 (May), 361–372.

YAP, C. K. 1987. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete Computational Geometry*, 2, 365–393.

YAP, C. 1997. Towards exact geometric computation. *Computational Geometry: Theory and Applications 7*, 1, 3–23.