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 $C_0(t)$, $C_1(r_1)$, ... , $C_n(r_n)$, 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.

[∗]Corresponding author. Email: raman@cs.technion.ac.il

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. Lavender et al. [20] 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 [21] suggested an algorithm that is based on the tree structure, whereas Alt and Schwarzkopf [2] presented a randomized incremental algorithm. Nevertheless, both algorithms [21, 2] appear to have only theoretical implications. Ramamurthy and Farouki [14], and Ramanathan and Gurumoorthy [15] have used numerical tracing methods. [14] first generates the bisectors and then trims them, whereas [15] generates the trimmed segments of the medial axis directly.

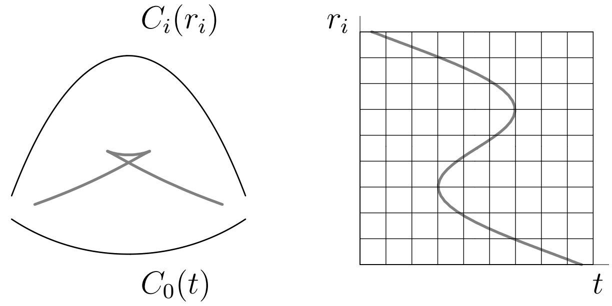
In this paper, the problem of generating a Voronoi cell of a planar free-form closed curve is addressed. The proposed algorithm directly uses the free-form rational curves that are not preprocessed into line segments or circular arcs. It is shown here that the symbolically generated bisector that is 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 [22], a well-known divide-and-conquer technique, is been used for generating the Voronoi cell [23] (to be precise, [23] generates medial axis). Though [23] has described an algorithm that uses lower envelope for generating medial axis transform of general curved objects, he approximates 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. All the algorithms and examples presented in this paper were implemented and created with the aid of tools available in the IRIT [24] 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. Results from our implementation and a discussion on the algorithm appear in Section 7. Finally, Section 8 concludes the paper.

2. BASIC IDEA

Let $C_0(t), C_1(r_1), ..., C_n(r_n)$ be the $n+1$ C^1-continous rational parametric planar closed curves. Without loss of generality, we may assume that the Voronoi cell is extracted for $C_0(t)$. The algorithm starts with the symbolic computation of a bivariate polynomial function, $\mathcal{F}_3(t, r_i)$, following [19], between two planar curves $C_0(t)$ and $C_i(r_i)$ so that the zero-set of $\mathcal{F}_3(t, r_i)$, $\mathcal{F}_3(t, r_i) = 0$, which is an implicit algebraic curve in the tr_i-plane, corresponds to the untrimmed bisector curve between $C_0(t)$ and $C_i(r_i)$. The formulation of $\mathcal{F}_3(t, r_i)$, 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 $\mathcal{F}_3(t, r_i)$ is shown in Figure 1(b).

A topology analysis algorithm [27], 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.



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

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

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

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 $C_1(r) = (x_1(r), y_1(r))$ be two planar C^1-continuous regular rational curves. Though the functions $\mathcal{F}_1(t, r)$ or $\mathcal{F}_2(t, r)$ in

[19] can be used, the bivariate function $\mathcal{F}_3(t,r)$ 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:

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

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

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

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):

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

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

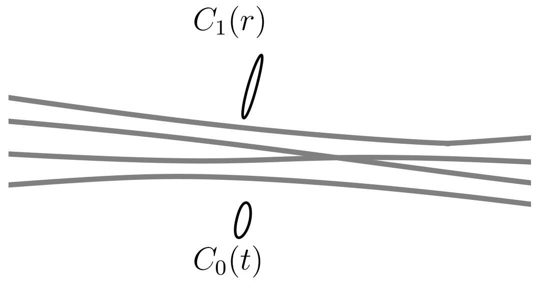
$$\begin{aligned} 0 &= \hat{\mathcal{F}}_3(t,r) \\ &= \left\langle P(t,r) - \frac{(C_0(t) + C_1(r)}{2}, C_0(t) - C_1(r)\right\rangle. \end{aligned} \tag{6}$$

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

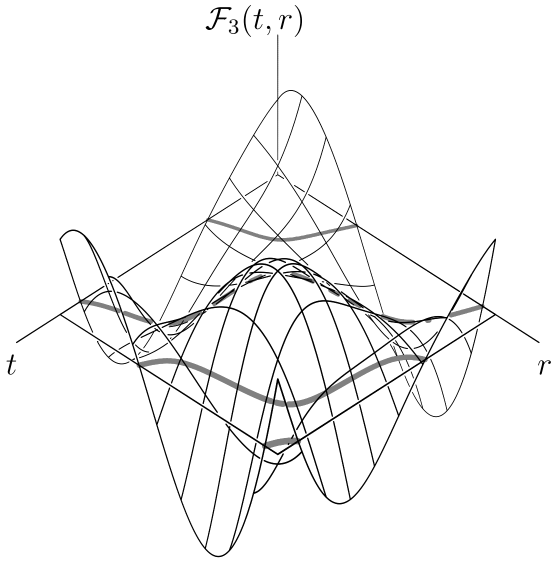
$$0 = \mathcal{F}_3(t,r) = (x_0'(t)y_1'(r) - x_1'(r)y_0'(t))\hat{\mathcal{F}}_3(t,r). \tag{7}$$

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 $(t, r, \mathcal{F}_3(t,r))$ generated by the bivariate function and its corresponding zero-set. It should be noted that $\mathcal{F}_3(t,r)$ represents the bisector with an accuracy that is bounded only by the machine's 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 $\mathcal{F}_3(t,r)$ and its zero-set, the bisectors.

Figure 2: The untrimmed bisector (a) and the zero-set of $\mathcal{F}_3(t,r)$ (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, $\mathcal{F}_3(t,r) = 0$, it is necessary to break it into monotone pieces, in the tr-space. Given the polynomial $\mathcal{F}_3(t,r)$ 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 $\mathcal{F}_3(t,r) = 0$ 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-

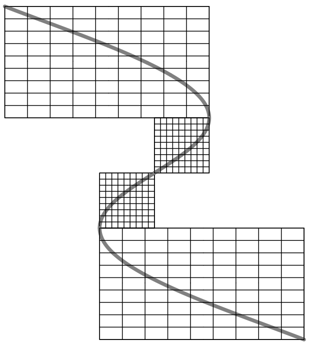


Figure 3: The monotone pieces of the zero-set of $\mathcal{F}_3(t,r)$ shown in Figure 1(b).

mon simultaneous solutions between $\mathcal{F}_3(t,r)=0$ and either $\frac{\partial \mathcal{F}_3}{\partial t}(t,r)=0$ or $\frac{\partial \mathcal{F}_3}{\partial r}(t,r)=0$. The multivariate solver of [26] is used to find these local extrema. Edge points are identified by computing all the intersections of $\mathcal{F}_3(t,r)=0$ 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 $\mathcal{F}_3(t,r)$ 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 $\mathcal{F}_3(t,r)$. 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.

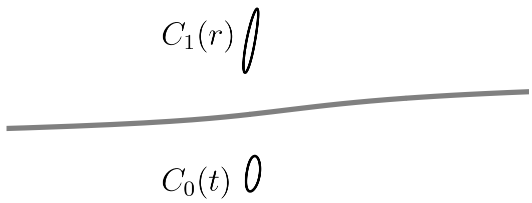


Figure 4: The bisector (in gray) between two closed C^1 planar curves after applying the LL-constraint (see also Figure 2(a)). The bisector extends to ∞ on both sides

Figure 2(a), in fact, shows the LL, LR, RL, and RR components of the bisector between the two C^1 planar closed curves.

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

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

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

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 their 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 [19], the intersection point satisfies the following equation:

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

where α and β represent the parameter on the normals of $C_0(t)$ and $C_1(r)$, respectively, whose values determine whether the intersection point is to the left 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 (refer to [19]):

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

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

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

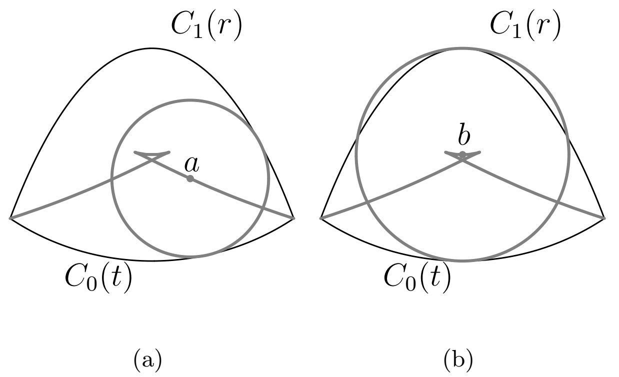


Figure 5: (a) The radius of the disk is smaller than the radius of curvatures at the footpoints of the curves. (b) The radius of the disk is greater than the radius of curvature of $C_1(r)$ at the footpoint implying the violation of the constraint.

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:

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

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

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

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:

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

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

$N_0^L(t)$ (resp. $N_1^L(r)$) 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 $N_0^L(t)$ (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

$$\frac{\big\langle P(t, r) - C_0(t), \hat{N}_0(t)/\kappa_0(t)\big\rangle}{1/\kappa_0(t)^2\big\langle \hat{N}_0(t), \hat{N}_0(t)\big\rangle} < 1, \tag{20}$$

$$\frac{\big\langle P(t, r) - C_1(r), \hat{N}_1(r)/\kappa_1(r)\big\rangle}{1/\kappa_1(r)^2\big\langle \hat{N}_1(r), \hat{N}_1(r)\big\rangle} < 1; \tag{21}$$

or

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

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

Figure 6 shows the portions of the bisector obtained after applying the curvature constraint (and after subjecting it to

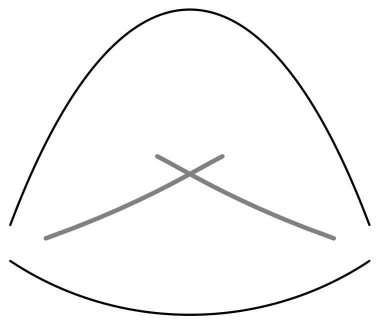


Figure 6: The bisector (in gray) after applying the curvature constraint, for the two curves shown in Figure 1(a).

the LL-constraint) for the untrimmed bisector shown in Figure 1(a). Note that 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 $\mathcal{F}_3(t, r_1)$ described in Section 3 and is followed by splitting $\mathcal{F}_3(t, r_1) = 0$ into monotone pieces described in Section 4. The process of trimming the zero-set of $\mathcal{F}_3(t, r_1)$ 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 $\mathcal{F}_3(t, r_1)$ 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 envelopes 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 [22, 28]. For the benefit of readers, the basic lower envelope algorithm is introduced in Section 6.1, following [22]. 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. 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

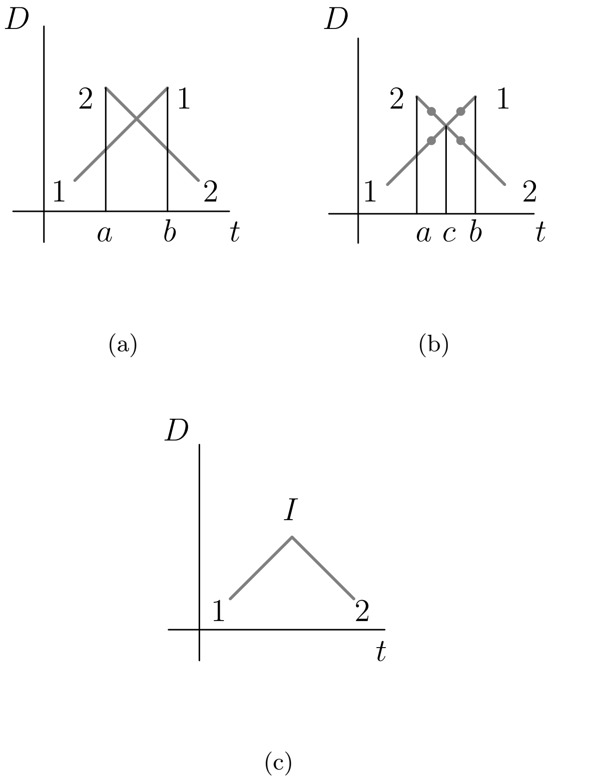


Figure 7: Illustration of the lower envelope algorithm.

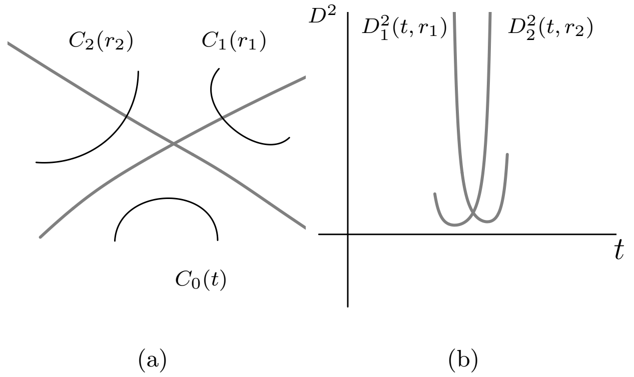


Figure 8: (a) The bisectors (in gray) between the pairs $(C_0(t), C_1(r_1))$ and $(C_0(t), C_2(r_2))$ of C^1 planar curves. (b) The squared distance functions $D_1^2(t, r_1)$ and $D_2^2(t, r_2)$ of the bisector of the respective pairs of curves.

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 $\mathcal{F}_3(t, r_i) = 0$ function, for every t value there is a single corresponding r_i value. Furthermore, there is a correspondence between $\mathcal{F}_3(t, r_i) = 0$ and $D_i(t)$. If $\mathcal{F}_3(t, r_i) = 0$ 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 on the distance functions defined over monotone bisectors of the form $\mathcal{F}_3(t, r_i) = 0$, 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 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 $\mathcal{F}_3(t, r_i)$ and $D_i^2(t, r_i)$ functions.

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

$$||D_i(t, r_i)||^2 = ||D_j(t, r_j)||^2, \tag{24}$$

$$\mathcal{F}_3(t, r_i) = 0, \tag{25}$$

$$\mathcal{F}_3(t, r_j) = 0. \tag{26}$$

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 t-monotone $\mathcal{F}_3(t, r_i) = 0$ 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 subcurves at a given t-parameter, all that is needed is to split its corresponding t-monotone $\mathcal{F}_3(t, r_i) = 0$ implicit function at that parameter.

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

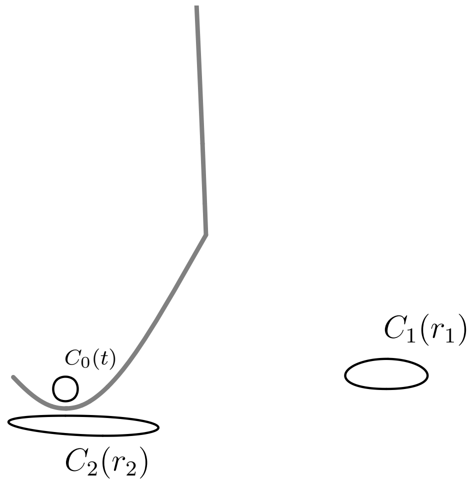


Figure 9: Voronoi cell (in gray) of C_0(t). $Deg_{max} = 5$, $CP_{max} = 5$ **and** $Deg\mathcal{F}_{3max} = 18$.

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 Deg), number of control points used (denoted as CP) and the degree of $\mathcal{F}_3(t, r)$ (denoted as $Deg\mathcal{F}_3$) 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 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).

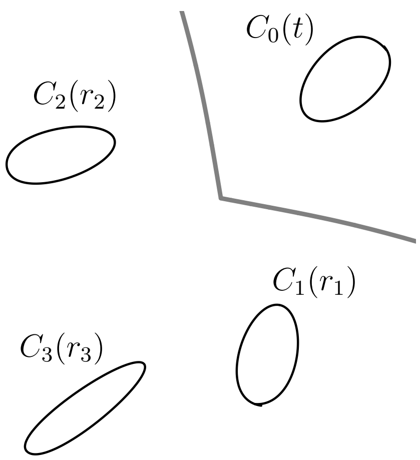


Figure 10: Voronoi cell (in gray) of C_0(t). $Deg = 5$, $CP = 5$ **and** $Deg\mathcal{F}_3 = 18$.

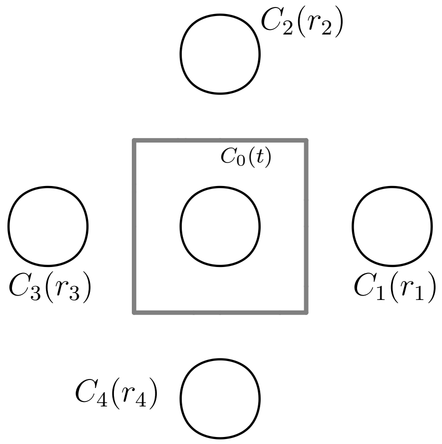


Figure 11: Voronoi cell (in gray) of C_0(t). All curves are identical, resulting in degenerated bisector curves as lines. $Deg = 3$, $CP = 7$ **and** $Deg\mathcal{F}_3 = 10$.

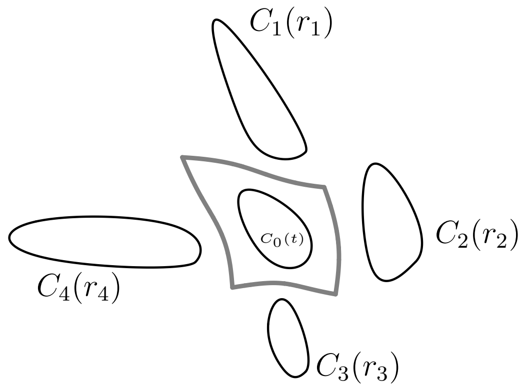


Figure 12: Voronoi cell (in gray) of C_0(t). $Deg = 3$, $CP = 7$ **and** $Deg\mathcal{F}_3 = 10$.

7.1 Discussion

The following are the major steps of our algorithm:

1. formulate the bivariate function $\mathcal{F}_3(t, r)$;
2. split the zero-sets of $\mathcal{F}_3(t, r)$'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 $\mathcal{F}_3(t, r)$ 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

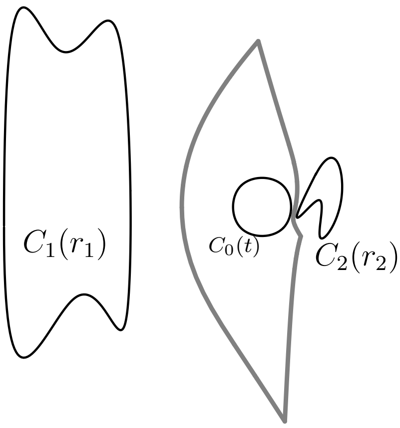


Figure 13: Voronoi cell (in gray) of C_0(t). $Deg_{max} = 3$, $CP_{max} = 8$ **and** $Deg\mathcal{F}_{3max} = 10$**.**

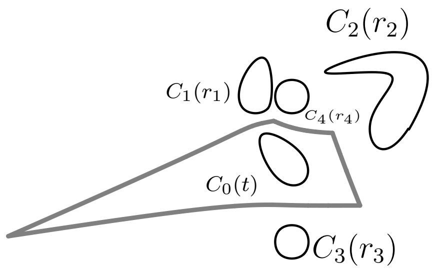


Figure 14: Voronoi cell (in gray) of C_0(t). $Deg = 3$, $CP = 7$ **and** $Deg\mathcal{F}_3 = 10$**.**

Intel Pentium 4 1.8GHz computer with 256MB RAM using the IRIT [24] 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 $\mathcal{F}_3(t, r)$ into monotone pieces involves the computation of turning points, which can be computed to arbitrary precision. Though the splitting procedure described in [27] 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 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 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 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 $\mathcal{F}_3(t, r)$ can be used to generate self-bisectors (bisectors of a curve with itself), the presented 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 rational [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-continous 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 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 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 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 piecewise implicit forms in a tr-parameter space that is arbitrarily precise. The presented approach can also be implemented using exact arithmetic [29], using packages such as bignums [30], and using long floats [31].

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.

9. REFERENCES

[1] **F. Aurenhammer**, "A survey of fundamental geometric data structure", *ACM Computing Surveys*, Volume 23, Number 3, September 1991, pp 345–405.

[2] **H. Alt and O. Schwarzkopf**, "The Voronoi diagram of curved objects", *11th Symposium on Compuational Geometry*, 1995, pp 89–97.

[3] **H. Blum**, "A transformation for extracting new descriptors of shape", in *Models for the Perception of Speech and Visual Form*, ed. Walthen Dunn, MIT Press, 1967, pp 362–381.

[4] **H. Blum**, "Biological shape and visual science (Part I)", *Journal of Theoretical Biology*, Volume 38, 1973, pp 205–287.

[5] **G. S. Baja and E. Thiel**, "(3-4)-Weighted skeleton decomposition for pattern representation and description", *Pattern Recognition*, Volume 27, Number 8, 1994, pp 1039–1049.

[6] **U. Montanari**, "Continuous skeletons from digitized images", *Journal of the Association for Computing Machinery*, Volume 16, Number 4, October 1969, pp 534–549.

[7] **H. N. Gursoy and N. M. Patrikalakis** "An automatic coarse and finite surface mesh generation scheme based on medial axis transform: Part 1 Algorithms", *Engineering with Computers*, Volume 8, 1992, pp 121–137.

[8] **C. G. Armstrong**, "Modeling requirements for finite element analysis", *Computer Aided Design*, Volume 26, Number 7, July 1994, pp 573–578.

[9] **J. O'Rourke**, *Computational Geometry in C*, Cambridge University Press, 1993.

[10] **D. Kim, I. Hwang and B. Park** "Representing the Voronoi diagram of a simple polygon using rational quadratic Bèzier curves", *Computer-Aided Design*, Volume 27, Number 8, 1995, pp 605–614.

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

[12] **V. Srinivasan and L. R. Nackman**, "Voronoi diagram for multiply-connected polygonal domains I: Algorithm", *IBM Journal of Research and Development*, Volume 31, Number 3, May 1987, pp 361–372.

[13] **M. Held**, "Voronoi diagram and offset curves of curvilinear polygons", *Computer-Aided Design*, Volume 30, Number 4, 1998, pp 287–300.

[14] **R. Ramamurthy and R. Farouki**, "Voronoi diagram and medial axis algorithm for planar domains with curved boundaries I: Theoretical foundations", *Journal of Computational and Applied Mathematics*, Volume 102, 1999, pp 119–141.

[15] **M. Ramanathan and B. Gurumoorthy**, "Constructing medial axis transform of planar domains with curved boundaries", *Computer-Aided Design*, Volume 35, Number 7, June 2003, pp 619–632.

[16] **R. Farouki and J. Johnstone**, "The bisector of a point and a plane parametric curve", *Computer Aided Geometric Design*, Volume 11, Number 2, 1994, pp 117–151.

[17] **G. Elber and M. Kim**, "The Bisector surface of rational space curves", *ACM Transaction on Graphics*, Volume 17, Number 1, January 1998, pp 32–39.

[18] **R. Farouki and R. Ramamurthy**, "Specified-precision computation of curve/curve bisectors", *International Journal of Computational Geometry and Applications*, Volume 8, Number 5 and 6, 1998, pp 599–617.

[19] **G. Elber and M. Kim**, "Bisector curves for planar rational curves", *Computer-Aided Design*, Volume 30, Number 14, 1998, pp 1089–1096.

[20] **D. Lavender, A. Bowyer, J. Davenport, A. Wallis and J. Woodwark**, "Voronoi diagrams of set theoretic solid sodels", *IEEE Computer Graphics and Applications*, September 1992, pp 69–77.

[21] **J. J. Chou**, "Voronoi diagrams for planar shapes" , *IEEE Computer Graphics and Applications*, March 1995, pp 52–59.

[22] **M. Sharir and P. K. Agarwal**, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, 1995.

[23] **Ju-Hsein Kao**,"*Process planning for additive/subtractive solid free-form fabrication using medial axis transform*", Ph.D. Thesis, Department of Mechanical Engineering, Stanford University, CA, June 1999.

[24] **G. Elber**, *IRIT 9.0 Users's Manual*, Technion, 2002, http://www.cs.technion.ac.il/∼irit.

[25] **C. Hoffmann and P. Vermeer**, "Eliminating extraneous solutions in curve and surface operation", *International Journal of Computational Geometry and Applications*, Volume 1, Number 1, 1991, pp 47–66.

[26] **G. Elber and M-S Kim**, "Geometric constraint solver using multivariate rational spline functions", *Proceedings of the Symposium on Solid Modeling and Applications*, Ann Arbor, Michigan, 2001, pp 1–10.

[27] **J. Keyser, T. Culver, D. Manocha and S. Krishnan**, "Efficient and exact manipulation of algebraic points and curves", *Computer-Aided Design*, Volume 32, Number 11, 15 September 2000, pp 649–662.

[28] **D. Halperin**, *Handbook of Discrete and Computational Geometry*, In Jacob E. Goodman and Joseph O'Rourke, editors, CRC Press LLC, Boca Raton, FL, 1997.

[29] **C. Yap**, "Towards exact geometric computation", *Computational Geometry: Theory and Applications*, Volume 7, Number 1, 1997, pp 3–23.

[30] **http://www.swox.com/gmp/**

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