A unified approach towards computing Voronoi diagram, medial axis, Delaunay graph and \( \alpha \)-hull of planar closed curves using touching discs

Bharath Ram Sundar, Manojkumar Mukundan, Ramanathan Muthuganapathy*

*Advanced Geometric Computing Lab, Department of Engineering Design, Indian Institute of Technology Madras, India

ARTICLE INFO

Article history:
Received May 3, 2020

Keywords: Voronoi diagram, medial axis, Delaunay graph, \( \alpha \)-hull, freeform curves, bisectors, antipodal distance, touching disc

ABSTRACT

This paper proposes a unified approach towards computing geometry structures viz. Voronoi diagram, medial axis, Delaunay graph and \( \alpha \)-hull of planar closed curves. It initially presents an algorithm for computing the Voronoi diagram of a set of planar freeform closed curves without approximating the curves using points, lines or biarcs. The algorithm starts by computing the minimum antipodal discs (MADs) for all pairs of curves and these MADs are systematically processed to identify all branch points. The key feature of the algorithm is that it computes a branch point without computing any of the bisectors a priori. Local computations of Voronoi segments are then done using the identified pairs of the segments of curves. The theoretical foundation of the algorithm has been first laid for a set of convex curves and then extended to non-convex curves. It has also been shown that the developed algorithm for the Voronoi diagram can also be used to compute related structures such as medial axis, Delaunay graph and \( \alpha \)-hull. They have also been addressed without computing Voronoi edges/segments. Results of the implementation have been provided along with a detailed discussion of the algorithm.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Voronoi diagram and its close associates such as medial axis are typically grouped under the term ‘Skeletons’. They have been extensively studied and used in a wide range of applications such as animation, shape matching, surface reconstruction, dimensional reduction, morphing, mesh generation, etc. (please see [1] for details). Voronoi diagram for a set of entities (could be points/lines/curves) can be considered as a set of all points having more than one nearest entity [2]. Two computations that dominate in generating Voronoi diagram are:

- **Voronoi vertices** termed as branch points (a point at which at least three curves are equidistant).
- **Bisectors** (set of points that are equidistance from two curves) from which Voronoi edges (an edge between two Voronoi vertices) are computed.

In this paper, a touching disc method is proposed to find branch points of a set of planar curves without finding computationally expensive bisector intersections (A touching disc is essentially a disc tangent to at least one of the input curves). It is worth noting that not many algorithms for computing the Voronoi diagram of a set of freeform curves employ them as such without discretizing into points, lines, biarcs or splines. Accurate computation of branch points is possible as the input curves are not approximated. The algorithm finds not only the branch points but also the pairs of portions of input curves that con-

*Corresponding author
e-mail: bharathsram@gmail.com (Bharath Ram Sundar), manojatgec@gmail.com (Manojkumar Mukundan), emry01@gmail.com (Ramanathan Muthuganapathy)
tribute to a Voronoi edge. Interestingly, the very same algorithm developed for computing the Voronoi diagram for a set of non-convex curves can also be used for finding the medial axis for a multiply-connected domain; splitting the domain is not required to convert it into a simply-connected one as in the case of [3]. The information for the Delaunay graph and $\alpha$-hull has been derived simply as a by-product of the algorithm for computing the Voronoi diagram and does not require the actual computation of Voronoi segments. The following are the major contributions of this work:

- No pre-processing required in approximating the input curve(s) and no post-processing required in computing the Voronoi segment(s).
- Identifying branch points without computing the bisectors and thereby reducing the computational complexity.
- Computation of Delaunay graph and $\alpha$-hull have been achieved without the need to compute Voronoi diagram/Voronoi segments.
- A unified algorithmic approach for the computation of the Voronoi diagram, medial axis, Delaunay graph and $\alpha$-hull has been presented, perhaps for the first time.

The paper is organized as follows: Section 2 reviews the related works followed by Section 3 that provides a few preliminaries required for the description of the algorithm. The basic idea of the algorithm for a set of convex curves is described in Section 4. Section 5 presents the algorithm followed by Section 6 explaining the working of the algorithm with an example. Extension of Voronoi diagram to non-convex curves and subsequently to computing medial axis, Delaunay graph and $\alpha$-hull have been presented in Section 7. The results of the algorithm are reported and discussed in Section 8. Section 9 concludes the paper.

2. Related works

Bisectors for computing Voronoi diagram can be represented in closed form only when the two given boundary curves belong to one of the following: a point, a line, or a conic curve. For rational freeform planar curves as input, the bisector has been shown to be rational only for point/curve combination. For two planar curves, the bisector is algebraic but not rational [4]. For two planar closed curves, bisector consists of many portions (Figure 1(a) [5], and among them, only a part of the bisector segment shown in Figure 1(b) will contribute to the Voronoi diagram. Hence, it is important to find the pair of portions of two closed curves that contributes to the Voronoi diagram without any need of additional trimming. Currently, trimming based procedures takes a lot of post-processing such as monotone splitting, left-left and curvature constraints on the bisector before computing the branch point via a lower-envelope technique [5]. Intersection computation (including self-intersections) of bisectors has been employed in [6]. The degree of a bisector has been shown to be $4m - 2$ [7], for curves of degree $m$, resulting in a high degree polynomial even for low degree curves.

Hence, the process of computing intersection(s) of bisectors is also numerically intensive. For numerical tracing/differential geometry approaches, distance computation of a point on the Voronoi diagram with all other curves is employed, making it computationally expensive [8, 9, 10].

As in the case of Voronoi diagram, the computation of the branch point is the key for medial axis as well. Tracing procedure [9] requires numerical solving at each point on the medial axis to compute the branch point. Though it is well known that the medial axis can be obtained by simply flipping the normal at a point on the curve, not all Voronoi diagram algorithms can handle the medial axis, as self bisectors (equidistant points within the same curve [5]) are neglected while computing Voronoi diagram. For a single closed curve, the medial axis is typically made of self-bisectors.

A typical approach of computing bisectors (and subsequently Voronoi diagram or medial axis) of planar curves by discretizing them into point-sets or straight lines creates unwanted edges (see Figure 1(c) [11], where small edges are present). To the best of our knowledge, there is no known relation between ‘amount of discretization’ and ‘the level of accuracy in computation’ in preserving the topology of branch points. An approximate algorithm for computing medial axis of planar domains based on the Domain Decomposition Lemma makes the localized computation of branch points easier [12]. In [11], a piecewise circular approximation of the boundary curve (which makes the computation of bisectors an easier task) along with a divide and conquer approach is employed in developing a fast and reliable algorithm for computing medial axis for simply-connected domain. In [3], though the boundary is segmented into spirals, a small number of touching disc computed to the original curve brings accuracy in topology. For this algorithm, the multiply-connected domain has to be converted into a simply-connected one by cutting through holes or inner boundaries.

Delaunay graph for a set of points has been defined as a dual of Voronoi diagram (i.e. two points in the input set are connected by a line if they share a Voronoi edge [13]). However, when the inputs are curves, it is not clear ‘which two points’ on the corresponding curves are to be connected. Hence, the Delaunay graph for curve inputs is more of a conceptual representation [14]. [15] deals with circles and [16] has implemented Apollonius graph for point sites which is effectively a weighted Voronoi graph for circles. These methods are based
on the Voronoi diagram of centers of circles which can not be extended directly to arbitrarily shaped curves. [14] deals with the Delaunay graph of smooth convex pseudo-circles, which relies on computing the Voronoi diagram first followed by the Delaunay graph. α-hull [13], a generalization of the convex hull, has been shown to be very useful in applications such as reconstruction [17]. For a set of points, the information for the α-hull can be obtained using its Delaunay graph [2]. However, as the Voronoi diagram by itself is quite intensive to compute for a set of curves, and as the Delaunay graph for this set is not well-defined, it is quite difficult to obtain information for the α-hull of a set of curves. In [14], the Delaunay graph of a set of ellipses has been attempted, with some points interior to each of the ellipses is being used to compute it. However, such interior points cannot be employed to compute α-hull as the empty circle property gets violated.

To the best of our knowledge, none of the previous works have combined the Delaunay graph and α-hull for curves, though the relationship between them is straight forward in the case of a point-set. In fact, though both Delaunay graph and α-hull can be derived from the Voronoi diagram, most (if not all) of the algorithms for the Voronoi diagram of curves are not amenable to compute them. This is because of the post processing cost involved in getting the exact portions of the input curves contributing to the Voronoi diagram that requires the computation of the connectivity information between the Voronoi diagram and the input set. The local minimum radius of each of the Voronoi segment is also needed to be computed.

To our knowledge, for inputs involving curves, computing only positive α-hull (i.e. intersection of all discs that contain all the curves) for a set of curves has been addressed in [18], using a set of enclosing circles and triple data.

3. Preliminaries

![Diagram showing antipodal discs, consistent TTDs, and branch discs.](image)

Fig. 2. Consistent MAD and TTD. In this paper, minimum antipodal lines and MADs are always shown in cyan, consistent TTDs in pink and BDs in red. Input curves are shown in green, unless specified otherwise.

Let the input be a set of disjoint planar freeform (parametric) closed \( C^1 \) continuous curves with no straight line portions.

Initially, it is assumed that the input curves are convex and an algorithm for computing Voronoi diagram for them has been presented assuming input curves are in general position (Section 7.1 discusses the extension for non-convex curves).

### Definition 1
When a disc is tangential to a curve, the point on the curve that just touches the disc is called **footpoint** and the disc is called **touching disc**.

### Definition 2
A touching disc is said to be **consistent** if outward normal of the curve and radius vector (from center to footpoint) of the disc are collinear and opposite at its footpoint.

### Definition 3
An **antipodal disc** between two curves is a touching disc to both these curves such that the outward normals at the two footpoints are collinear.

There can be many such antipodal discs between two curves and the one with minimum radius is called **Minimum Antipodal Disc** (MAD) and line joining the two footpoints of the MAD is called **minimum antipodal line**. Figure 2(a) shows all antipodal lines with minimum antipodal line in cyan. Distance between two closed curves is minimum only at minimum antipodal points [19].

### Definition 4
A **three touch disc** (TTD) is a touching disc having three footpoints, each one touching a different curve.

TTDs between three curves are shown in Figure 2(b), where consistent TTD is shown in pink color. Do note that there can be more than one consistent TTD for three curves (Figure 2(c)). Hereafter, we use the term TTD to represent consistent TTDs unless otherwise specified.

### Definition 5
A touching disc is said to be **empty** if the interior of the disc does not intersect any of the input curves.

### Definition 6
A **Voronoi disc** is an empty touching disc at any point on the Voronoi diagram.

### Definition 7
A **branch disc** (BD) is an empty TTD whose center is a **branch point**.

Figure 2(d) shows all TTDs; empty TTDs (BDs) and branch points are shown in red color.

### Definition 8
Radii of Voronoi discs along a Voronoi edge vary, and it can be expressed as a function called **Voronoi radius function**, \( R(u) \) where \( u \) parameterize a Voronoi edge.

**Lemma 1.** Voronoi radius function, \( R(u) \), attains local minima only at MADs between the two curves.

**Proof.** Consider a point \( B(u_i) \) on a Voronoi edge with Voronoi radius, \( R(u_i) \). It is known that the tangent to the Voronoi edge at a point is an angular bisector of the tangents at its footpoints (of the Voronoi disc at that point) on the corresponding curves [20] (Figure 3). For a small change \( \Delta u \) along the Voronoi edge, the new point \( B(u_i+\Delta u) \) and its footpoints can be assumed to be along their respective tangents. Let \( \Omega \) be the included angle between the radius vectors of the Voronoi disc at the footpoints. Normals at the new point \( B(u_i+\Delta u) \) are parallel to normals at...
Fig. 3. Voronoi radius function, $R(u)$ is local minima at MADs.

Fig. 4. Voronoi segment between $C_a$ and $C_b$ (in black) is shown bounded between Voronoi discs of $R_{max}$ (hereafter in red) and $R_{min}$ (hereafter in brown). Disc of $R_{max}$ in (a) is a MAD, whereas in (b) is a BD.

3.1. Voronoi segment

A Voronoi segment is defined in terms of Voronoi radius function, $R(u)$, which varies along a Voronoi edge. A Voronoi segment can be a Voronoi edge or portion of a Voronoi edge such that

1. one end of this segment corresponds to an $R(u)$ of local minimum (termed as $R_{min}$) and other end corresponds to a local maximum (termed as $R_{max}$)
2. for every Voronoi segment, $R(u)$ varies monotonically between $R_{min}$ and $R_{max}$.

In the case of Voronoi diagram for a set of closed convex curves, maximum point always corresponds to a branch disc ($R_{max} = \infty$, when the end of the Voronoi segment extends to infinity), whereas minimum point can correspond to either a MAD (Figure 4(a)) or a branch disc (Figure 4(b)).

4. Basic idea of the Algorithm

The logic behind developing the proposed algorithm is to find all Voronoi discs, with locally extremum radius function, in the increasing order of their radii. During this process, the algorithm also pairs the discs corresponding to $R_{max}$ and $R_{min}$ for every Voronoi segment. These discs identify the pairs of portions of curves contributing to the Voronoi segment using their footpoints. Hence, trimming of bisectors can be avoided for finding Voronoi segments.

4.1. Directed edges

In this algorithm, all Voronoi segments are marked or identified using a concept called Directed Edges (DEs). A DE always corresponds to a unique Voronoi segment and vice versa. DE is a data structure used to track Voronoi segments and kept in a linked list. Information regarding the Voronoi disc corresponding to the $R_{min}$ of a Voronoi segment is used for defining its DE. For a Voronoi segment between two curves, this includes

1. footpoints of the disc of $R_{min}$ on each of the curves
2. ordered indices of the curves such that the order assigns a direction for a DE and this direction can be used to identify the corresponding Voronoi segment towards the right side of the DE.

As MAD in Figure 5(a) is empty, two Voronoi segments grow from the center of the MAD; a directed edge dentifies the one growing towards its right side. A directed edge, $DE_{ab}$ (Figure 5(b)), is shown as a solid line segment (in violet), connecting footpoint on $C_a$ to the footpoint on $C_b$ with an arrowhead (in red) at $C_b$. A Voronoi segment represented by a DE always grows towards its right side. Voronoi segment corresponding to $DE_{ba}$ is shown in Figure 5(c). In Figure 5(d), $DE_{ab}$ and $DE_{ba}$ are shown together along with the growing directions for Voronoi segments represented by them. As the footpoints of $DE_{ab}$ and $DE_{ba}$ are the same, they coincide and appear as a single solid line with double arrowhead.

4.2. Identifying Voronoi segments and branch points through directed edges

Fig. 5. Illustration of the basic idea. Hereafter, each red arrowhead in a violet line indicates a directed edge. Blue portions along the input curves indicate the portions of curves used for computing Voronoi segments. Black dashed lines indicate Voronoi segments extending towards the right side of corresponding DEs. Continuous black lines/curves indicate Voronoi segments, unless specified otherwise.
Definition 9. Two DEs, \(DE_{ab}\) and \(DE_{cd}\) are said to be consecutive if and only if curves, \(C_b\) and \(C_c\) are same (or \(C_a\) and \(C_d\) are same).

As input curves are assumed to be in general position, the maximum number of Voronoi segments for which the disc of \(R_{\text{max}}\) is the same is three, but the minimum is two in any case. In Figure 5(e), a BD (\(BD_{abc}\)) exists between the curves, \(C_a\), \(C_b\) and \(C_C\); center of this BD is towards the right side of \(DE_{ab}\), \(DE_{bc}\) and \(DE_{ca}\). Disc of \(R_{\text{max}}\) for the Voronoi segments represented by these three DEs are the same. These DEs can be seen deleted once corresponding Voronoi segments are computed (Figure 5(f)). In Figure 5(g), a BD (\(BD_{abc}\)) exists towards the right side of \(DE_{ab}\), \(DE_{bc}\). It can be noticed in Figure 5(h) that no BD (\(BD_{cba}\)) exists towards the right side of \(DE_{cb}\) and \(DE_{ba}\). Hence the idea is to use the presence of DEs to identify the BDs.

4.3. Finding MADs between all pairs of curves

As per the definition of Voronoi segment (Section 3.1) and Lemma 1, there can be two Voronoi segments, starting from the center of an empty MAD. In order to identify all empty MADs, MADs between all pairs of input curves are computed and sorted, as a list \(L\), in the increasing order of radii. Now, emptiness of these MADs are checked in the order in which it appears in the list \(L\). On finding an empty MAD, DEs are placed to mark the Voronoi segments. When two consecutive DEs exist, possibility of constructing TTDs is checked through triplets (which is explained below).

4.4. Formation of triplet

Introducing triplets restricts the number of TTDs to be formed on the way to find BDs.

Definition 10. A triplet, \(C_{ijk}\), for the curves \(C_i\), \(C_j\) and \(C_k\) is formed if directed edges \(DE_{ij}\) and \(DE_{jk}\):

- exist and have a TTD with center towards the right side of both the DEs; and

- are parametrically closest either in clockwise or anticlockwise direction along the curve \(C_j\).

\(DE_{ab}\) and \(DE_{bc}\) in Figure 5(g) form a triplet (\(C_{abc}\)), whereas \(DE_{cb}\) and \(DE_{ba}\) in Figure 5(h) do not form a triplet (\(C_{cba}\)). On the other hand, two different triplets are formed from three curves in Figure 2(e). Only the radius of the TTD corresponding to the triplet is inserted into \(L\) maintaining ascending order (let \(R_{ijk}\) denote the radius of the TTD for the triplet \(C_{ijk}\)). In Figure 6(a), \(DE_{ij}\) does not form a triplet with \(DE_{ji}\) as they are not parametrically close in either clockwise or anticlockwise direction along \(C_i\). It may be noted that parametric closeness ensures the formation of at most 4 triplets on identifying a new DE.

4.5. Computation of Voronoi diagram by processing the list, \(L\)

It may be noted that radius of TTDs identified through triplets are also maintained in the same list, \(L\), in the increasing order of radii. Each radius in \(L\) is processed in the same order to ascertain the emptiness of the corresponding disc. When the turn of a TTD (let it be \(TTD_{abc}\)) arrives for processing, if all or two of the DEs (\(DE_{ab}\), \(DE_{bc}\) and \(DE_{ca}\)) are present, \(TTD_{abc}\) is empty and is a BD. Voronoi segments for which disc of \(R_{\text{max}}\) is \(TTD_{abc}\) are computed immediately after identifying \(TTD_{abc}\) as empty; corresponding DEs are deleted so that they will not cause the formation of further triplet.

For the set of curves shown in Figure 5(i), no DEs are defined using the footpoints of \(MAD_{ca}\) as it is not empty. However, \(DE_{ab}\) and \(DE_{bc}\) form a triplet, \(C_{abc}\), and \(TTD_{abc}\) gets inserted into the list \(L\). When the turn of this \(TTD_{abc}\) arrives for processing, as \(DE_{ab}\) and \(DE_{bc}\) are available, it is identified as a BD. This leads to the construction of Voronoi segments corresponding to \(DE_{ab}\) and \(DE_{bc}\) and deletion of \(DE_{ab}\) and \(DE_{bc}\) (Figure 5(j)). Now, Voronoi segment between \(C_a\) and \(C_C\) grows from the center of \(TTD_{abc}\) and extends towards infinity, and this Voronoi segment is identified by \(DE_{ca}\) defined using the footpoints of \(TTD_{abc}\) on \(C_a\) and \(C_C\) (Figure 5(j)).

4.6. MADs with used footpoints are not empty

While processing a MAD in the list, if any of the two footpoints of a MAD is already used for computing Voronoi segment, the MAD is not empty (by maximal disc property of the Voronoi diagram) and further emptiness check can be avoided.

4.7. Complexity reduction in finding emptiness of MADs

The emptiness of a MAD can be checked by computing the minimum distance of the center of the MAD to the other input curves. Nevertheless, such minimum distance computations are restricted by the following lemma.

Lemma 2. \(MAD_{ij}\) for which any of the footpoints is not used for computing a Voronoi segment is not empty if and only if it intersects with a curve which has a directed edge parametrically closest to the footprint of \(MAD_{ij}\).

Proof. Let \(MAD_{ij}\) with radius \(R_{ij}\) is not empty. \(MAD_{ij}\) is not empty proves that there exists a Voronoi disc corresponding to each of the footpoints of \(MAD_{ij}\). Consider the footpoint with one of the curves (let it be \(C_i\)) and let its corresponding Voronoi disc in the Voronoi diagram be \(D1\) of radius \(R_{D1}\).

\(D1\) belongs to some Voronoi segment with \(R_{\text{min}}\) and \(R_{\text{max}}\) such that \(R_{\text{min}} \leq R_{D1} \leq R_{\text{max}}\). There are two cases here.

Case 1: If \(R_{\text{max}} < R_{D1}\), then the entire portion of the Voronoi segment would have got processed and the footprint of \(MAD_{ij}\) would not be available. Hence the lemma.

Case 2: If \(R_{\text{max}} \geq R_{D1}\), the following are true. \(R_{ij} > R_{D1}\) (according to the assumption of non-empty MAD). By maximal disc property of the Voronoi diagram, \(MAD_{ij}\) is intersected by the curve with which \(D1\) has the other footprint, and let this curve be \(C_i\), \(R_{\text{min}} \in L\) and identifies the curves that contribute to \(D1\), through the corresponding directed edge. This directed edge must be parametrically closest to the footprint of \(MAD_{ij}\) with \(C_i\). This proves the second condition of the lemma.
The converse of the lemma is trivially true, and hence the lemma is proved.

As per the above lemma, emptiness of $MAD_{ij}$ can be checked with those curves which form triplets with $MAD_{ij}$. As at most four triplets can be formed with $MAD_{ij}$ (refer section 4.4), the emptiness of a MAD can be ensured by checking intersection with at most four of the other input curves.

5. The algorithm

The algorithm starts with an initialization where MADs between all pairs of curves are calculated and sorted, as a list $L$, in ascending order of radii. As the least radius MAD is empty, its directed edges are added to a linked list $D$. Algorithm 1 shows the initialization. Algorithm 2 shows handling MADs and formation of triplets. Algorithm 3 shows the steps involved in identifying all BDs (empty TTDs) and corresponding Voronoi segments. Using the above-mentioned algorithms as subroutines, Algorithm 4 computes the Voronoi diagram for a set of closed convex curves.

Algorithm 1 InitializeVD(CurveList)
1: Initialize $L$ as a list of all MADs sorted in ascending order of radii.
2: Get the first radius, $R_{ij}$ from $L$.
3: Add $DE_{ij}$ and $DE_{ji}$ (the smallest MAD is empty).
4: Remove the radius from the list.

Algorithm 2 ProcessMAD($L$)
1: Pick the first radius in $L$.
2: if the radius belongs to MAD (say $MAD_{jk}$) and the footpoints have not been used then
3: if (empty as per Lemma 2) then
4: Add $DE_{jk}$ and $DE_{kj}$.
5: Check the possibility for forming parametrically closest triplets for both $DE_{jk}$ and $DE_{kj}$.
6: Insert TTDs of available triplets appropriately in $L$.
7: end if
8: end if
9: Remove the radius from $L$.

Algorithm 3 ProcessTTD($L$)
1: Pick the first radius from $L$.
2: if the radius is of a TTD (say $TTD_{ijk}$) then
3: if all the three directed edges exist then
4: Compute Voronoi segments for pairs of curve segments from their DE footpoints to the footpoints of BD.
5: Delete $DE_{ij}$, $DE_{ik}$ and $DE_{ki}$.
6: else if if two of the DEs (say $DE_{ij}$ and $DE_{ik}$) exist then
7: Compute Vonoi segments for pairs of curve segments for $C_i, C_j$ and $C_i, C_k$ from their DE footpoints to the footpoints identified by TTD.
8: Delete $DE_{ij}$ and $DE_{ik}$
9: Add $DE_{ik}$.
10: Identify parametrically closest triplets for $DE_{ik}$.
11: Insert TTDs of available triplets appropriately in $L$.
12: else
13: Remove the radius from $L$.
14: end if
15: end if

Algorithm 4 GetCurvePortionsForVD(CurveList)
1: InitializeVD(CurveList)
2: while $L$ is not empty do
3: ProcessMAD(DiscList)
4: ProcessTTD(DiscList)
5: end while

5.1. Correctness proof of the algorithm

The correctness of the algorithm is proved through Lemmas 3, 4 and 5, provided in the appendix (Appendix A). The emptiness of a TTD is identified without any geometrical test (for curve containment) but rather with the mere existence of corresponding DEs (Lemma 5 proves this). Lemma 3 proves that the algorithm finds all branch discs. Thus, these lemmas together establish the correctness of the algorithm.

6. Working of the algorithm 4 with an example

6.1. Start of the algorithm

Figure 7 describes construction of Voronoi diagram for a set of 5 convex curves. Algorithm 1 initializes list $L$ as a list of all MADs sorted in ascending order of radii (Figure 7(a)).

$L = \{R_{24}, R_{12}, R_{23}, R_{14}, R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{45}\}$.

As the MAD with least radius is empty, radius $R_{24}$ is removed after adding $DE_{24}$ and $DE_{42}$ (Figure 7(b)). Hence, after executing Algorithm 1,

$L = \{R_{12}, R_{23}, R_{14}, R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{45}\}$.

6.2. Processing other radii

From $L$, $R_{12}$ is processed next (Figure 7(c)). Emptiness of $MAD_{12}$ is checked by forming triplet with curve $C_4$ (Figure 7(d)). $MAD_{12}$ has been found to be empty, directed edges $DE_{12}$ and $DE_{21}$ are added (Figure 7(e)). $R_{12}$ is removed from the list and hence $L = \{R_{23}, R_{14}, R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{45}\}$. Continuing with the illustration, $C_4$ (Figure 7(e)) is a triplet from $DE_{12}, DE_{21}$. Radius of $TTD_{421}$, $R_{421} (= 8.69)$ is added to $L$ appropriately. Hence, the list is...
\[ R_{23}, R_{14}, R_{421}, R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{45} \].

Now, emptiness of \( \text{MAD}_{23} \) is checked by forming triplets \((C_3, C_2, C_4)\) (Figure 7(f)) and \((C_1, C_2, C_3)\) (Figure 7(g)). As \( \text{MAD}_{23} \) is found empty, \( \text{DE}_{23} \) and \( \text{DE}_{32} \) are added (Figure 7(h)). Now \( L \) is \( \{R_{14}, R_{421}, R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{45}\} \). DEs with parametrically closest footpoints on the curve, \( C_2 \) contribute triplets \( C_{324} \) and \( C_{123} \). Hence, \( R_{324} \) (= 30.39) and \( R_{123} \) (= 44.69) are inserted to \( L \). The list at this stage is \( \{R_{14}, R_{421}, R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{324}, R_{45}, R_{123}\} \). Processing of \( R_{14} \) leads to Figure 7(i), showing all the directed edges. The list is \( \{R_{221}, R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{324}, R_{45}, R_{123}\} \).

Now, \( R_{421} \) is processed. As all the three directed edges \( \text{DE}_{42}, \text{DE}_{21}, \text{DE}_{14} \) exist, the disc is classified as \( BD_{121} \) and its center is a branch point. Corresponding Voronoi segments are then computed (black segments in Figure 7(j)) and \( R_{421} \) is removed from the list. Also, the directed edges \( \text{DE}_{42}, \text{DE}_{21}, \text{DE}_{14} \) are removed. Figure 7(j) shows the unprocessed directed edges (single arrowheads showing \( \text{DE}_{24}, \text{DE}_{41}, \text{DE}_{12} \)). \( L \) is \( \{R_{34}, R_{13}, R_{35}, R_{15}, R_{25}, R_{324}, R_{45}, R_{123}\} \). As \( \text{MAD}_{34} \) as well as the subsequent \( \text{MAD}_{13} \) are found to be not empty (Figures 7(k) and 7(l)), no directed edges are added between \( C_3, C_4 \) and \( C_1, C_3 \). Now, \( L = \{R_{35}, R_{15}, R_{25}, R_{324}, R_{45}, R_{123}\} \).

\( \text{MAD}_{35} \) (Figure 7(m)) and \( \text{MAD}_{15} \) are empty and hence \( R_{351} \) (= 26.34) (Figure 7(n)) is added to the list. \( L \) is \( \{R_{25}, R_{351}, R_{324}, R_{45}, R_{123}\} \). \( \text{MAD}_{25} \) is found to be empty and hence directed edges are added (Figure 7(o)). \( R_{235} \) (= 24.87) and \( R_{125} \) (= 24.99) are now added. \( L \) is \( \{R_{325}, R_{125}, R_{351}, R_{324}, R_{45}, R_{123}\} \). \( R_{325} \) belongs to a TTD and since \( \text{DE}_{23}, \text{DE}_{35}, \text{DE}_{42} \) exist, this TTD is classified as \( BD \). Voronoi segments are computed and then, the directed edges and \( R_{325} \) are removed (Figure 7(p)). Similar to \( R_{325}, R_{125} \) is processed (Figure 7(q)). List at this point is \( \{R_{351}, R_{324}, R_{45}, R_{123}\} \).

Since the footpoints of \( R_{351} \) has been used in the formation of Voronoi diagram (while processing \( R_{325} \) and \( R_{125} \), this disc is rejected without processing further. For \( R_{324} \) (Figure 7(r)), since directed edges \( \text{DE}_{32}, \text{DE}_{24} \) exist, \( TTD_{324} \) is classified as \( BD_{324} \). However, \( \text{DE}_{34} \) is now added using the footpoints of \( BD_{324} \) on \( C_3 \) and \( C_4 \). This is because, the only Voronoi segment between \( C_3 \) and \( C_4 \) starts from the center of \( BD_{324} \) and grows towards the right side of \( \text{DE}_{34} \), the addition of which can contribute to further triplets (though in this illustration, there are no further triplets from \( \text{DE}_{34} \)).
Computed Voronoi segments are shown in Figure 7(s). \( L \) is \( \{R_{45}, R_{23}\} \). However, the footpoints of the MAD$_{45}$ have already been used in the computation of Voronoi segments (Figure 7(t)) and hence this MAD is rejected without further processing. For \( R_{123} \), none of the directed edges DE$_{12}$, DE$_{23}$ and DE$_{31}$ are available (implying that they have already been used for the computation of Voronoi segments), and hence this TTD is rejected without processing for BD.

### 6.3. Termination of the algorithm 4

The algorithm 4 terminates when all MADs are processed and no further triplet is available for processing the corresponding TTD. At this juncture, the unused portions of the curves (which will contribute to the convex hull of the curves) (green ones in Figure 7(t)) are used to complete the Voronoi diagram. Figure 7(u) shows the final Voronoi diagram.

#### 7. Voronoi diagram of non-convex curves, medial axis, Delaunay graph and \( \alpha \)-hull

### 7.1. Extension of the Voronoi algorithm to non-convex curves

In the case of non-convex curves, a Voronoi disc can touch an input curve at more than one point and hence self-bisectors have to be computed (A self-bisector can be defined as a bisector segment, where the footpoints for a point on the segment are from the same curve). This implies that MAD and BD have to be solved within the same curve itself. In such a scenario, it is convenient to split a non-convex curve into portions that will not contribute multiple footpoints. The high curvature points of the curve aid in such a division (a high curvature point along a curve is a point at which curvature is a local maximum).

**Definition 11.** A disc associated with the high curvature (positive) point on an input curve is called High Curvature Disc (HCD) (Figure 8). Here, positive means that the outward normal of the curve and radius vector of the HCD at the footpoint are in the opposite direction.

HCD has only one footpoint and its center is also a starting point of a self-bisector segment (Figure 8).

If the input curve is split at all its high curvature points, then none of the split curves will have a Voronoi disc touching at two points of the same split curve [21].

Thus, by splitting an input curve at all its high curvature points, it can be made sure that no curve from the split curves will contribute two or more footpoints to any Voronoi segment. The radius of the corresponding HCD is added to the radius list, \( L \).

However, some of the HCDs need not be empty and the curve portions to the left and right of the HCD footpoint contribute to the final Voronoi diagram. Thus, it is important to ascertain the emptiness of HCDs. The emptiness of an HCD is checked by computing the minimum distance of the center of the HCD to input curves. Nevertheless, the number of such minimum distance computations for an HCD$_{ij}$ can also be restricted by lemma 2 (by replacing MAD$_{ij}$ in lemma 2 with HCD$_{ij}$).

Thus the HCDs are handled as in Algorithm 5.

### Algorithm 5: ProcessHCD(\( L \))

1. Pick the first radius in \( L \).
2. if the radius belongs to HCD (say HCD$_{jk}$) and the footpoint has not been used then
3. if empty according to Lemma 2 then
4. Add DE$_{jk}$.
5. Identify TTDs similar to Algorithm 2.
6. Insert available TTDs appropriately in \( L \).
7. end if
8. end if
9. Remove the radius from \( L \).

### Algorithm 6: ProcessXAD(\( L \))

1. Pick the radius from \( L \).
2. if the radius is of a XAD and the footpoints are not used then
3. Get parametrically closest directed edges to the footpoints of \( XAD_{ij} \) as DE$_{ij}$ and DE$_{ji}$
4. Compute Voronoi segments for pairs of curve segments identified by DE$_{ij}$ and DE$_{ji}$ from their footpoints to the footpoints identified by XAD$_{ij}$
5. Delete edges DE$_{ij}$ and DE$_{ji}$
6. end if
7. Remove the radius from \( L \).

#### 7.1.1. Maximum antipodal disc

Also, between the concave portions of curves, the antipodals that are locally maximum can be empty (see Figures 9(a) and 9(b)). In that case, the Voronoi radius function attains local maxima and the empty disc will have two Voronoi segments joining at its center (Figure 9(c)). Though the center is not a branch point, it will mark the end of the corresponding Voronoi segments. So, maximum antipodal discs (XADs) are introduced in the non-convex case in order to make the definition of a Voronoi segment consistent with that of the convex case. These XADs will be dealt with as in Algorithm 6, to generate the Voronoi segments.
Algorithm 7 GetCurvePortionsForVD(CurveList)

1: InitializeVD(CurveList)
2: Split the input curves at high curvature points.
3: Compute all HCDs and XADs and insert into \( L \) keeping the increasing order of radii.
4: while \( L \) is not empty do
5: \hspace{1em} ProcessMAD(\( L \)) (Algorithm 2)
6: \hspace{1em} ProcessHCD(\( L \)) (Algorithm 5)
7: \hspace{1em} ProcessTTD(\( L \)) (Algorithm 3)
8: \hspace{1em} ProcessXAD(\( L \)) (Algorithm 6)
9: end while

7.1.2. Modifications

In Algorithm 4, GetCurvePortionsForVD(CurveList), there are only two discs to handle: MAD and TTD. To handle non-convex curves, this has to be modified accordingly by including HCD and XAD. MAD and HCD are checked for emptiness using Lemma 2. HCD and XAD are handled as shown in ProcessHCD(\( L \)) and ProcessXAD(\( L \)) respectively. Thus the Algorithm 4 in Section 6 is modified appropriately to handle non-convex curves (see Algorithm 7).

7.2. Medial axis

Voronoi diagram is typically computed outside the set of curves, where a curve is split at HCDs. It can also be recalled that, for a point on the computed Voronoi diagram, the radius vector and the outward normal at a corresponding footpoint are collinear and opposite to each other. However, the medial axis is computed interior to the curve, where, a point on the medial axis, the radius vector and the inward normal (i.e., flip the normal direction) of a corresponding footpoint are collinear and opposite to each other. Hence, the algorithm for the Voronoi diagram (Algorithm 7) can be suitably modified to compute the medial axis. For example, the definition for ‘consistent disc’ can be modified to compare the radius vector with inward normal at the footpoint (refer Definition 2 in Section 3). In general, the medial axis can be obtained by employing Algorithm 7. The proposed algorithm is capable of computing the medial axis for not just a single curve but also for curves representing an outer boundary along with several inner boundaries (without any pre-processing for converting it into a simply-connected one).

7.3. Delaunay graph

The algorithm developed for the Voronoi diagram of curves gives pairs of exact portions of curves, such that, bisectors that contribute to the Voronoi diagram exist only between the identified pairs as Voronoi segments. The pair is represented as StructureEdge in Algorithm 8 and is shown graphically in Figure 10. Each curve portion is identified using its corresponding curve parameters, say Curve1.Parameter1 and Curve1.Parameter2 (\( \gamma \) and \( \beta \) respectively in Figure 10). Also, the radii of the discs at the start and end of a Voronoi segment are obtained \( (R_{\text{min}} \text{ and } R_{\text{max}}) \) from the touching disc information. For example, for a set of convex curves, \( R_{\text{min}} \) comes from MAD or BD and \( R_{\text{max}} \) typically from BD (refer Section 3.1). For Voronoi segments that extend up to infinity, \( R_{\text{max}} \) is taken as an arbitrarily large value.

To generate a Delaunay graph (as in [14]), the touching discs themselves give information about which pairs of curves share a Voronoi segment (curve portions in blue in Figure 10). As the inputs are curves, a representative point (in this paper, centroid of each curve) has been employed as the corresponding point representation. Two representative points are connected if the two corresponding curves share touching discs that would generate a Voronoi segment between them. The uniqueness of this method is that the Delaunay graph can be obtained via touching discs, implying that the graph can be computed without actually computing the Voronoi diagram of curves.

7.4. \( \alpha \)-hull

The \( \alpha \)-hull is the union of all Voronoi discs with radius \( \alpha \) (\( \alpha \)-discs). Lemma 3 in [22] states that the centers of the \( \alpha \)-disc have to lie on the Voronoi diagram for the \( \alpha \)-hull. In practice, only \( \alpha \)-discs are computed and not the union of them.

From the obtained Delaunay graph, \( \alpha \)-discs for curves for a given \( \alpha \) can be directly computed. For computing the \( \alpha \)-discs, it is known a priori that a disc of radius \( \alpha \) will fit only in places where the StructureEdge satisfies \( R_{\text{min}} < \alpha < R_{\text{max}} \) (similar to Lemma 3 of [13]). Thus, \( \alpha \)-discs are obtained by solving for touching discs.

8. Results and Discussion

8.1. Results

Algorithm 7 has been implemented using IRIT geometric kernel [23]. MADs and Touching discs (for their constraint equations, please see [18, 19]) have been solved using the multivariate constraint solver in IRIT. The concept of directed edges is implemented using linked list. All the freeform (parametric) curves are represented using non-uniform rational B-splines (NURBS). For testing, curves up to degree 4 have been employed. Figure 11(a) shows the Voronoi diagram, BDs and the left-over directed edges (which are present only in the curves belonging to the convex hull, indicating the completeness of the algorithm). Figures 11(b) to 11(d) show Voronoi diagram with directed edges (not included the Voronoi diagram for the green segments of the curves). Figure 11(f) shows a result where one end of the Voronoi segment extends to \( \infty \) for almost all segments. Figures 11(e), 11(g) and 11(h) show the Voronoi diagram, where one of the curves in the input is very small.
Voronoi diagram for a set of non-convex curves has been shown in Figure 12(a), where the input curves have been split using high curvature points. Figure 12(b) shows the Voronoi diagram if the input curves were not split, where the self-bisectors did not form part of the Voronoi diagram. Additional results for a set of curves that also include non-convex curves are shown in Figure 12(c) to Figure 12(e). For more non-convex curves, results have been shown in the form of medial axis (Figure 13). Figures 13(a) to 13(c) show the result of medial axis for non-convex curves. Figure 13(d) shows that the algorithm can compute the medial axis for a single closed convex curve as well. Figure 13(e) shows medial axis for multiple non-intersecting set of curves (whose Voronoi diagram is shown in Figure 12(c)). Figure 13(f) demonstrates that the algorithm is capable of generating medial axis for a multiply-connected object having multiple inner boundaries.

Delaunay graph for a set of curves is shown in Figure 14(a) using which \(\alpha\)-discs for the set of curves for different radii values have been computed (Figures 14(b) to 14(e)).

8.2. Discussion

8.2.1. Comparison of computational complexity with other approaches.

In this paper, no computation of bisector was employed to compute the branch points unlike the \(O(n^2)\) bisectors computed in [5] and [6] (where \(n\) is the number of input curves). Only local computation of Voronoi segments has been employed, effectively eliminating computationally intensive processing on bisectors. Given a curve of degree \(m\), TTD or MAD has a degree of \(m + (m - 1)\), a substantially lower degree polynomial compared to its bisector (which is \(4m - 2\) [7]). HCD and XAD can be computed using differential property (curvature) at a point in the curve, which are also straight forward computations. Finding the center point of a TTD to the left or right of a directed edge can be done using a simple area predicate [2]. Emptiness of TTD is decided using the existence of directed edges and not by any curve containment check. For the emptiness check of MAD, HCD, and XAD, though curve containment has been used, it has been shown that the number of
such computations have been restricted to a few (see Lemma 2). Insertion and deletion on a list is a straight forward operation. Hence the algorithm in this paper largely revolves around the computation of MAD and TTD, a lesser intensive computation than that has been employed in other works [5, 6, 9, 10].

To compute the Delaunay graph, the touching discs themselves give the pairs of curves for which an edge to be made in the graph. A simple structure (StructureEdge, see Algorithm 8) has been introduced to process the Delaunay graph. In fact, the StructureEdge also stores the radii information from which α-discs can be computed for a given α. Without this information, computationally intensive additional processing on the Voronoi distance function is needed to compute the radii information.

8.2.2. Robustness

The topology of the Voronoi diagram depends on capturing the branch points robustly and correctly. As opposed to using step sizes as in [9] or intersection of bisectors (which captures topology well with more intensive computation) as in [6], the proposed approach uses a simple directed edge existence. Figure 11(f) indicates two branch discs that are very close to each other are captured correctly. Figures 11(a) show the result where a curve is very small, showing that the algorithm is quite robust for small curves as well. It can be noted that the input curves have not been approximated by lines or arcs. Even with approximations such as the one with biarc [11] do not appear to guarantee topological correctness of the computed Voronoi diagram (see Figure 12 and the discussions therein in [3]). It has also been noted in [3] that, for achieving correct topology, approximation with very high precision might be needed for spiral arc discretization as well. An exact implementation such as in [14], is another possible approach for improving robustness.

8.2.3. Algorithmic complexity

Overall computation has been dominated by the algorithm for the Voronoi diagram. The major portion of the computation of the Voronoi diagram is taken in the first step of the initialization of the algorithm, which is computing all MADs. For a given n curves, this process takes $O(n^2)$, since it considers all pairs of curves. During triplet formation, it is to be noted that, there can be at most four triplets for a MAD (as only the parametrically closest curves on clockwise/anti-clockwise direction are used). It has to be emphasized that not all triples (combination of three curves) from the set of curves are considered for constructing TTDs. Hence, the algorithm’s complexity at this point will be $O(4 \times n^2)$. As computing the emptiness of discs is done geometrically, the overall algorithmic complexity will be $O(n^2)$, comparable to existing algorithms, whose complexity typically is $O(n^2)$ (for example, please see [9]). There have been $O(n \log n)$ algorithms, like [3] and [11], achieved through a divide and conquer approach. However, the same approach will not work in computing medial axis for a multiply-connected region (involving large number of holes) and Voronoi diagram for a large number of closed curves.

8.2.4. Unified approach

In general, most approaches that compute the Voronoi diagram can also generate medial axis (at least, theoretically). To the best of our knowledge, none of the previous works dealt with Voronoi diagram, medial axis, Delaunay graph and α-hull in unison for curves. Other algorithms will require post-processing such as identifying the radius of antipodal disc etc. Unified approach is a unique feature of the proposed algorithm and is made possible because the computations have been based on touching discs and not on bisectors. Such an approach also enabled a structure StructureEdge (see Algorithm 8) to be formulated directly from the touching discs. Only a few works, like [15] and [14], deal with Delaunay graph of curves but they too restrict the input to either circles or require the computation of Voronoi diagram.

8.2.5. Limitations and Future work

Although the theoretical nature of the developed algorithm makes it amenable for application to polygons or curves with $C^1$ discontinuities, application to such entities have not been considered. In the future, additional touching discs such as pseudo-TTD which creates the shape locally at its footpoints can be considered. Also, lines piercing the shape locally at their endpoints (which are vertices of polygons or $C^1$ discontinuities) can be considered as pseudo-antipodal. Extension of touching discs to three dimensions is also under consideration.

9. Conclusion

The paper presents an approach where the structures, Voronoi diagram, medial axis, Delaunay graph and α-discs can be computed in one go for a set of curves as input. The developed algorithm has also been demonstrated through implementation. It has been shown that bisector computation is not required to compute the branch points, effectively reducing computational complexity to a large extent. Moreover, empty TTDs and thereby BDs are identified without any curve containment check. Theoretical bases and observations ensure the correctness of the algorithm. Possible future works have been indicated.

References

Appendix A: Lemmas showing the correctness of the algorithm

Lemma 3. All empty TTDs are identified by the algorithm.

Proof. Let $TTD_{abc}$ be an empty disc. Hence, the center of $TTD_{abc}$ is a branch point at which Voronoi segments between each pair of $C_a$, $C_b$ and $C_c$ meet Voronoi segment towards this branch point may start from the center of a MAD or a BD (as explained in section 3.1).

Case 1: All Voronoi segments contributing to the branch point start from the center of MAD (Figure 15(a)).

All MADs are checked for emptiness in the increasing order of radii. On finding empty, its DEs will be inserted into the list $\mathbb{D}$ of DEs and radii of triplets formed from these DEs are inserted into $\mathbb{L}$ keeping the increasing order of radii. Now, consider those DEs which contribute to the triplet for $TTD_{abc}$. There can be no smaller empty TTD whose center acts as the branch point for the Voronoi segments from these DEs (in that case, $TTD_{abc}$ can not be empty) (Figure 15(b)). Hence, these DEs remain there till the turn of $TTD_{abc}$ arrives for processing. Hence, the lemma.

Case 2: Voronoi segment contributing to the branch point starts from center of a BD (Figure 15(c)).

This BD cannot be of radius more than the radius of $TTD_{abc}$, so immediately after identifying this BD, corresponding DE will be inserted into $\mathbb{D}$. Since $TTD_{abc}$ is empty, this DE will remain there till $TTD_{abc}$ is taken for processing. Now, the problem has been shifted to this BD which is also an empty TTD. This BD will be identified if it belongs to Case 1. Otherwise, it is a recursion of Case 2. At every stage of recursion, radius of BD introducing a DE is getting reduced, so recursion is finite and at some stage, case 1 is met as there exist BD/BDs of smallest radius among all BDs. This proves the lemma.

Corollary 1. All empty TTDs (BDs) will appear in the list $\mathbb{L}$, and by the time a TTD’s turn arrives for processing, corresponding DEs will be present, in the list $\mathbb{D}$ of DE, in order to identify its emptiness.

Lemma 4. All empty TTDs (BDs) are identified by the algorithm in the increasing order of their radii.

Proof. Let there be two empty TTDs, $D_1$ and $D_2$ such that radius of $D_1$ is greater than that of $D_2$. Assume $D_1$ is inserted into $\mathbb{L}$ before $D_2$ gets identified and inserted into $\mathbb{L}$. However, Voronoi disc corresponding to the $R_{min}$ of each of the Voronoi segments towards $D_2$ is of radius not larger than that of $D_2$. Hence, these discs contribute DEs for the formation of triplet of $D_2$ before the turn of $D_1$ arrives for processing. This leads to the insertion of $D_2$ into the list $\mathbb{L}$, placed before $D_1$. Consequently, turn of $D_2$ arrives before $D_1$ and hence the lemma.

Lemma 5. As the turn of $TTD_{abc}$ in the list $\mathbb{L}$ arrives, if two of the corresponding DEs (say $DE_{ab}$ and $DE_{bc}$) are present, then $TTD_{abc}$ is empty and is a BD for the curves $C_a$, $C_b$ and $C_c$.

Proof. The presence of the DEs in the list $\mathbb{D}$ indicates that the construction of Voronoi segments towards the right side of these DEs has not been completed.
Consider the Voronoi segment identified by $DE_{ab}$ which grows through centers of Voronoi discs till a Voronoi disc touches the third curve. This third curve will not be $C_e$ (let it be $C_d$) if $TTD_{abc}$ is not empty. If so, there exists a $TTD_{abd}$ towards the right side of $DE_{ab}$ and smaller than $TTD_{abc}$ (Figure 15(b)). Therefore, the turn of $TTD_{abd}$ (which is now an empty TTD) arrives before the turn of $TTD_{abc}$, and this leads to the construction of Voronoi segments corresponding to $TTD_{abd}$ and deletion of $DE_{ab}$. Thus, $DE_{ab}$ will not be available by the time $TTD_{abc}$ is taken for processing if $TTD_{abc}$ is not empty. Hence, the lemma.

In a similar way the emptiness of a TTD with three DEs can also be proved.