A dynamic sampling approach towards computing Voronoi diagram of a set of circles

Manojkumar Mukundan, Ramanathan Muthuganapathy*

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

Abstract

Discretizing the input curves into points or lines is a way of constructing an approximate Voronoi diagram. Sampling the input curves into points requires a fine sample density to obtain a reasonable topological and geometrical accuracy of the Voronoi diagram. In this work, it is shown that an accurate computation of the Voronoi diagram for a set of circles employing a very coarse sample set is possible. The approach proposes the selection of varying number of sample points on each circle dynamically, using the Delaunay graph of center points of circles. We then show that this approach can be used to identify neighborhood information of the circles. The touching disc method is then used to construct the Voronoi diagram with an algorithmic complexity of $O(n \log n)$. The work also demonstrates the robustness and theoretical correctness of the proposed algorithm by considering inputs in non-general position and for a large number of circles of the order of $\mathbf{10^5}$.

Keywords: Voronoi diagram, Circles, Touching disc, Dynamic sampling, Delaunay graph, Antipodal disc

1. Introduction

For a set of geometric entities (such as points, lines, curves, or circles), the Voronoi diagram can be considered as the locus of a point that is equidistant from more than one entity. A two-dimensional Voronoi diagram tessellates the plane into regions called Voronoi regions such that there is a Voronoi region corresponding to every entity in the given set. Each Voronoi region is a collection of points that are nearest to the corresponding geometric entity than any other entity in the given set Aichholzer et al. (2009). Application of Voronoi diagram can be found in image segmentation and feature extraction, motion planning, collision avoidance, determining the optimal deployment of infrastructure in town planning, mesh generation, dimensional reduction, surface reconstruction etc. Cornea et al. (2007). Voronoi diagram of circles, also known as additively weighted Voronoi diagram, can be used for estimating the size of a wire bundle Sugihara et al. (2004); Ryu et al. (2020), sensor deployment Mahboubi and Aghdam (2013) etc.

It is well known that the Voronoi diagram of circles of varying radii cannot be the same as that of the point set representing centers of these circles. Many algorithms are currently available which are capable of finding the Voronoi diagram of circles which are not only varying in radii but intersecting also. Lee and Drysdale has proposed an $O(n log^2 n)$ algorithm for computing Voronoi diagram for a set of non-intersecting circles, where n is the number of circles in the input set Lee and Drysdale (1981). Sharir also proposed an $O(n \log^2 n)$ algorithm for a set of circles, which may be intersecting Sharir (1985). An $O(n^2)$ algorithm by D.-S. Kim *et al.* finds Voronoi diagram of intersecting circles, with an edge flipping technique, after finding Voronoi diagram of centers of circles Kim et al. (2001a,b). A sweepline algorithm by Jin et al.

*Corresponding author

Email addresses: manojatgec@gmail.com (Manojkumar Mukundan), emry01@gmail.com (Ramanathan Muthuganapathy)

(2006) computes the Voronoi diagram of a set of circles of varying radii, where the intersection of circles and circles completely contained in other circles are permitted. As the order of intersection can be of $O(n^2)$, this algorithm has a worst case time complexity of $O(n^2 \log n)$, whereas for non-intersecting input, this becomes $O(n \log n)$. A topology oriented incremental algorithm by Lee et al. (2016) shows a linear performance even though the complexity is of $O(n^2)$. Algorithm of Sundar et al. (2020) can be used to find the Voronoi diagram of a set of non-intersecting circles in the $O(n^2)$ time complexity using a touching disc approach.

An approximate Voronoi diagram can be obtained by computing the Voronoi diagram of a point set sampled from the boundary of the input circles. However, this method requires a very fine sample density to bring accuracy in topology of the computed Voronoi diagram Sugihara (1993). Theoretically, an $O(n \log n)$ algorithmic complexity can be achieved by employing the same number of samples for all of the input circles. Nevertheless, computational time increases as the number of samples per circle increases. Moreover, it is difficult to have a priori knowledge on a sample density sufficient to accurately capture the topology of the Voronoi diagram. Sample based algorithm by Li et al. (2019) for finding Voronoi diagram of spheres requires restarting with an increased sampling density when the topology is not captured correctly. An adaptive subdivision of bounding box has been employed by Wang et al. (2020) for finding the additively weighted Voronoi diagram in 3D.

Most of the algorithms for computing the Voronoi diagram are not capable of dealing with degenerate inputs, where Voronoi vertices equidistant from more than three circles meet during the course of computation. Usually symbolic perturbations are used in order to address degenerate input set Edelsbrunner and Mücke (1990); Yap (1990); Sugihara (1992). The analysis of predicates is used by Emiris and Karavelas (2006) for the exact computation of additively weighted Voronoi diagrams in the plane. In the work of Devillers et al. (2015), a qualitative symbolic perturbation method is used, where a sequence of perturbation parameters is used to perturb the input objects. Algorithm by Karavelas and Yvinec (2002) has a complexity of $O(n \log n)$ in computing the Voronoi diagram of a set of non-intersecting circles but requires perturbing the input set for addressing degeneracies in input. Algorithms for computing the Voronoi diagram by Sugihara and Iri (1992) and Lee et al. (2016) handle degeneracies by following a topology oriented incremental approach.

In this paper, a sampling-based algorithm is proposed for the exact computation of the Voronoi diagram for a set of circles. Delaunay graph (dual of Voronoi diagram) computed for the centers of the input circles is utilized to dynamically collect a bare minimum number of samples from each circle. The dynamically collected samples are then used to establish the neighborhood details for computing the Voronoi diagram using a touching disc method. In this paper, it is also demonstrated that the developed algorithm is capable of addressing degeneracies and perform exact computation of Voronoi diagram without employing any perturbation of the input set. Following are the major contributions of this work:

- A dynamic sampling approach to the exact computation of Voronoi diagram of circles of $O(n \log n)$ complexity.
- Proposed a strategy to control the sampling rate and discussed its theoretical correctness for the exact computation.
- Handles circles in non-general position without perturbing the input set.
- A programmable correctness-check to verify the topological and geometrical accuracy of the computed Voronoi diagram.

The remainder of this paper is organised as follows. Sections 2 and 3 explain the terminologies used in this paper and a brief description of the touching disc method, respectively. Section 4 discusses the idea of sampling. The proposed dynamic sampling is then explained in Section 5. Section 6 presents the algorithm, followed by Section 7, where the correctness of the algorithm is discussed. Handling inputs in non-general position is explained in Section 8. Section 9 explains how the algorithm deals with intersecting and hidden circles. In Section 10, the results of the algorithm are presented and discussed. Finally, Section 11 concludes the paper.

2. Preliminaries

Let $\mathcal{C} = \{c_i | i = 1, 2, ...n\}$ represents a set of input circles, where n is the size of $\mathcal{C}$ and $c_i = c_i(p_i, r_i)$ represents a circle with center at $p_i = p_i(x_i, y_i)$ and radius $r_i \geq 0$ in $\mathbb{R}^2$. $P = \{p_i | i = 1, 2, ...n\}$ represents set of center points of all input circles. Let $\mathcal{VD}(\mathcal{C})$ is the Voronoi diagram, in $\mathbb{R}^2$, for the set of circles $\mathcal{C}$, computed in Euclidean metric. The input circles are assumed to be non-intersecting. However, the proposed algorithm can also handle inputs containing intersecting circles and identifies circles that are completely contained in another circles.

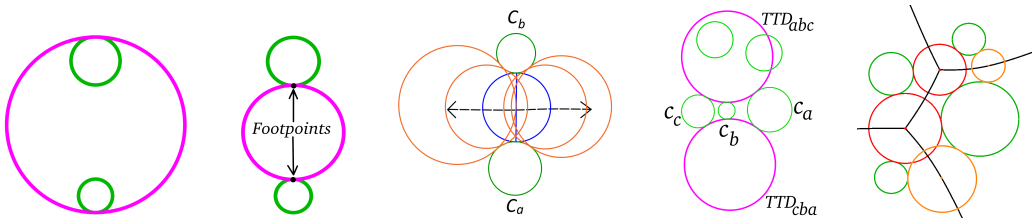
Definition 1. ***Touching disc*** *is a disc that is tangential to a set of circles and doesn't contain any of them. The point at which the touching disc touches a circle is called* ***footpoint*** *(Figure 1(a)-(b)).*

Definition 2. ***Antipodal Disc (AD)*** *between two circles is the touching disc with minimum radius and having antipodal footpoints (i.e., they are diametrically opposite).*

AD between circles c_a and c_b is represented as AD_{ab} (blue disc in Figure 1(c)).

Definition 3. *A touching disc to three circles is called* ***Three Touch Disc (TTD).***

There can be two TTDs between c_a, c_b and c_c, represented as TTD_{abc} and TTD_{cba} (Figure 1(d)).



(a) (b) (c) (d) (e)

Figure 1: Touching discs (in this paper, input circles are shown in green color, unless specified otherwise). (a) Not a touching disc as it contains the circles to which it is tangent. (b) A touching disc and footpoints. (c) A few touching discs between c_a and c_b, among which AD_{ab} is shown in blue. (d) Two TTDs exist between c_a, c_b, and c_c. TTD_{abc} is not empty as it contains\intersects one input circle. (e) Discs shown in red and orange are Voronoi discs, where discs in red are branch discs, the center of which is a branch point. Voronoi edges are shown in black color.

Definition 4. *A touching disc of a set of circles is said to be* ***empty*** *if it does not intersect or contain any of the other circles from $\mathcal{C}$.*

TTD_{abc} in Figure 1(d) is not empty as it either contains/intersects at least one circle other than c_a, c_b and c_c. TTD_{cba} is empty.

Definition 5. *A* ***Voronoi disc*** *is a touching disc that is empty and has more than one footpoint.*

Definition 6. *A* ***Voronoi edge*** *between two circles is the locus of the center of a Voronoi disc, which has footpoints with these input circles.*

Definition 7. *An empty TTD is called a* ***branch disc****, and its center is called a* ***branch point****.*

Among the Voronoi discs shown in Figure 1(e) (in red and orange color), the discs shown in red color are branch discs.

Definition 8. *The* ***Voronoi cell*** *of a circle $c_i \in \mathcal{C}$ is defined as $\mathcal{VC}_i = \{p \in \mathbb{R}^2 | d(p, p_i) - r_i \leq d(p, p_j) - r_j\}, i \neq j$, where $d(p, q)$ is the Euclidean distance between the points $p, q \in \mathbb{R}^2$.*

Equality defines Voronoi edges and Voronoi vertices (or branch points), which form the boundary of Voronoi cells. Circles c_i and c_j are said to be neighbors, and $\mathcal{VC}_i$ and $\mathcal{VC}_i$ are said to be adjacent Voronoi cells, if $\mathcal{VC}_i$ and $\mathcal{VC}_i$ share Voronoi edge(s).

Definition 9. *The radius of the Voronoi disc varies along a Voronoi edge, and it is expressed as a function called* ***Voronoi radius function*** $R(u)$*, where* u *is a parameter representing a point on the Voronoi edge. Voronoi disc for which* $R(u)$ *reaches a local maximum is called* ***disc of*** R_{max}*, whereas that of local minimum is called* ***disc of*** R_{min}*.*

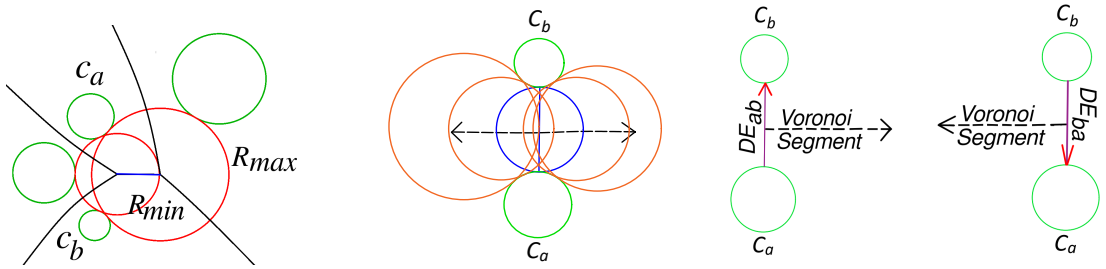
Definition 10. *A Voronoi edge or a portion of a Voronoi edge bounded by the discs of* R_{max} *and* R_{min}*, between which* $R(u)$ *varies monotonically, is called a* ***Voronoi segment****.*

In Figure 2(a), one of the Voronoi segments is highlighted in blue color, and corresponding discs of R_{max} and R_{min} are shown in red color. As $\mathcal{VD}(\mathcal{C})$ is the union of all Voronoi segments, every Voronoi segment can be identified in the touching disc method using the disc of R_{min} corresponding to each of the Voronoi segments.

2.1. Directed edges

The existence of a Voronoi segment between two circles, c_a and c_b, is represented by placing a Directed Edge (DE).

A DE is graphically represented as a straight line segment with an arrowhead such that no point of its Voronoi segment lies at the left side of the DE. The endpoints of a DE are the footpoints of the *disc of* R_{min}, of the Voronoi segment, with c_a and c_b. If $\mathbb{DE}(C)$ is the set of all DEs, and $\mathbb{VS}(C)$ the set of all Voronoi segments, there exists a bijection from $\mathbb{DE}(C)$ to $\mathbb{VS}(C)$. A data structure is used to define a DE, which holds information such as the coordinates of its endpoints, direction, radius and center of the *disc of* R_{min}.



(a) (b) (c) (d)

Figure 2: Voronoi segments and directed edges (DEs). (a) The Voronoi segment between c_a and c_b, bounded between the discs of R_{min} and R_{max}, is highlighted in blue color. (b) AD_{ab} between c_a and c_b acts as the *disc of* R_{min} for two Voronoi segments (in dashed line). (c) Voronoi segment corresponding to DE_{ab}, and (d) Voronoi segment corresponding to DE_{ba}.

In Figure 2(b), touching discs between input circles c_a and c_b keep growing from AD_{ab} in two directions. Hence, it acts as the *disc of* R_{min} for two Voronoi segments (Please refer to Definition 10). These Voronoi segments are represented by the DEs, DE_{ab} and DE_{ba}, which are shown separately in Figures 2(c) and (d), along with their corresponding Voronoi segment. In Figure 2(b)-(d), the *disc of* R_{min} is an empty AD, whereas, in Figure 2(a), a Voronoi segment is highlighted in blue color for which the *disc of* R_{min} is an empty TTD (Section 3.2 illustrates how a DE is placed for such a Voronoi segment). It may be noted that *disc of* R_{max} is always a branch disc; for the Voronoi segment extending to infinity, it is assumed that the branch point is at infinity and the *disc of* R_{max} is of radius ∞. In Figures 2(c) and (d), the center of the *disc of* R_{max} of the Voronoi segment is at infinity and lies at the right side of the DE.

3. A brief description of touching disc method

Initially, it is assumed that ADs between all pairs of input circles are computed. These ADs are sorted in the ascending order of radii and inserted into a list $\mathbb{L}$. Now, each of these ADs is tested for emptiness in the order of its radius. Once found empty, DEs are placed to mark corresponding Voronoi segments.

Definition 11. *Two DEs,* DE_{ab} *and* DE_{cd} *are said to be consecutive if and only if either* c_b *and* c_c *or* c_a *and* c_d *represent the same circle .*

3.1. Parametric closeness of DEs

If we consider a circle c_a in the input set, there can be many DEs connected to c_a during the course of computation. Let $p(x(t), y(t))$ be a point on the circle c_a, where $t \in [0, 1]$ parameterize the circle. DEs which are parametrically close to the point p are those DEs that have one of their endpoints on c_a and closer to the point p along c_a in either clockwise or anticlockwise direction than the endpoint of any other DEs on c_a. So, at a particular instance of computation, there can be a maximum of two DEs parametrically close to the point p; one in the clockwise direction and the second in the anticlockwise direction. Parameter t is used to measure the parametric closeness of DEs.

Parametric closeness of two DEs can be checked if and only if they have footpoint on a common circle (Refer to Figure 3 for parametric closeness of DEs). When two consecutive DEs, DE_{ab} and DE_{bc}, are found parametrically close with respect to their endpoint on c_b, the possibility of constructing TTD_{abc} is checked. TTD_{abc} is possible if and only if the center of a TTD between c_a, c_b and c_c lies at the right side of both DE_{ab} and DE_{bc}.

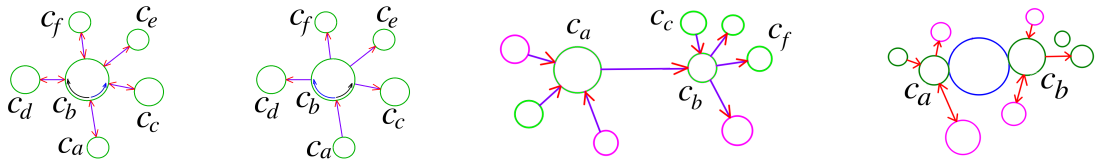


Figure 3: The parametric closeness of DEs (DEs are shown arbitrarily for the explanation). (a) DEs connected to c_b. DE_{bd} and DE_{db} are parametrically close to both DE_{ab} and DE_{ba} (clockwise, along c_b). DE_{bc} and DE_{cb} are also parametrically close to both DE_{ab} and DE_{ba} (anticlockwise, along c_b). (b) DE_{bd} and DE_{bc} are parametrically close and consecutive to DE_{ab}. (c) Circles (in pink) form TTDs with c_a and c_b on introducing DE_{ab}. TTD_{abc} will not be formed, as DE_{bc} is not present. TTD_{abf} will not be formed, as DE_{ab} and DE_{bf} are not parametrically close. (d) The parametrically close DEs to the footpoints of AD_{ab} (in blue) identifies the circles in pink color.

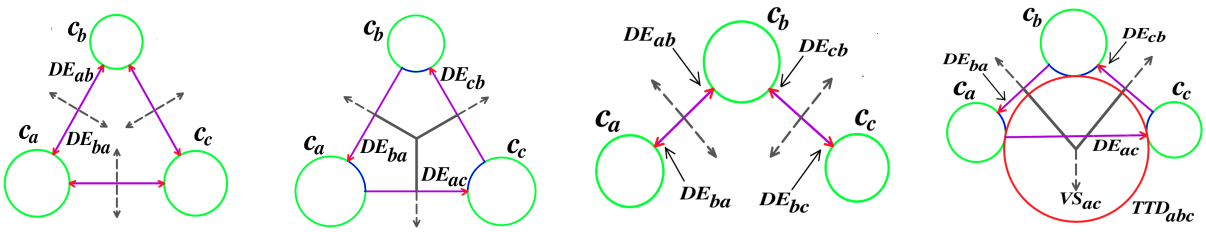
3.2. Processing TTDs and computing Voronoi segments

Parametric closeness restricts the number of TTDs to be formed for finding branch discs. When a TTD is formed, it is also inserted into the same list $\mathbb{L}$, maintaining the order of radii. When the turn of a TTD (TTD_{abc}) arrives for processing, if all or at least two of its DEs (DE_{ab}, DE_{bc}, and DE_{ca} are DEs of TTD_{abc}) are present, it is empty, and TTD_{abc} will be identified as a branch disc (Figure 4(a)). Once a branch disc is identified, Voronoi segments of its DEs for which TTD_{abc} is the *disc of* R_{max} are computed. These DEs are deleted immediately after computing its Voronoi segments (Figure 4(b)). If one of the three DEs is absent (let it be DE_{ca}), it indicates that there is no Voronoi segment between c_a and c_c for which TTD_{abc} is the *disc of* R_{max} (Figure 4(c)). In this case, after computing Voronoi segments of DE_{ab} and DE_{bc}, DE_{ab} and DE_{bc} are deleted, and DE_{ac} is inserted to identify the Voronoi segment between c_a and c_c for which TTD_{abc} is the *disc of* R_{min}. Here, DE_{ac} is defined using the footpoints of TTD_{abc} on c_a and c_c (Figure 4(d)). The algorithm stops once all the discs in the list $\mathbb{L}$ are addressed.

3.3. Emptiness check of ADs

It may be noted that TTDs are also inserted into the same list $\mathbb{L}$. When the turn of a particular AD in the list $\mathbb{L}$ arrives for processing, many TTDs (appearing in the list before this AD) might have been identified as branch discs leading to its computation of Voronoi segments. An AD for which footpoints are already used for the computation of the Voronoi segment is not empty (by maximum disc property of Voronoi diagram).

For a set of circles, $C = \{c_a, c_b, c_c, ..\}$, if the radius of AD_{ab} is not greater than that of AD_{ac}, c_c will not intersect AD_{ab}. Hence, it is enough to consider those ADs which have been processed before AD_{ab} for identifying the circles which are intersecting AD_{ab}. For a set of circles in Figure 3 (d), let the DEs that exist while processing AD_{ab} are red. Since DEs come into existence in the order of radii of their corresponding



(a) (b) (c) (d)

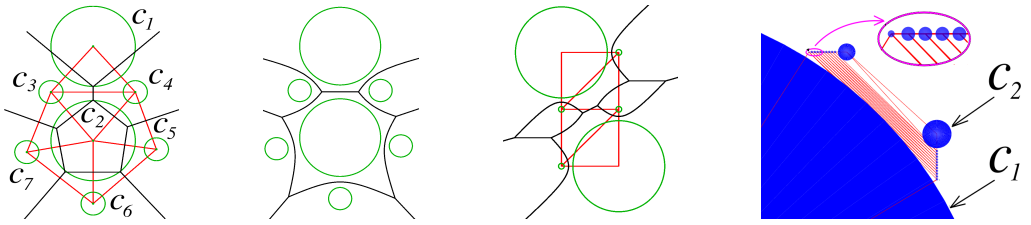
Figure 4: Processing TTDs. (a) As AD between each pair of circles is empty, two DEs are defined using footpoints of each AD; in the figure, these DEs appear as a single line segment (in magenta color) with two arrowheads (in red). Each arrowhead corresponds to a DE. (b) DE_{ab}, DE_{bc} and DE_{ca} are deleted after computing its Voronoi segments (shown in firm dark line). (c) As AD_{ac} is not empty, DEs are not placed using its footpoints. (d) DE_{ab} and DE_{bc} are deleted. DE_{ac} is introduced using the footpoints of TTD_{abc} (in red), and the corresponding Voronoi segment is also shown (with a dashed line labeled as VS_{ac}). In (a) and (b), the portion of circles used for computing Voronoi segments is highlighted in blue color.

discs of R_{min}, radii of discs of R_{min} corresponding to all existing DEs cannot be greater than the radius of AD_{ab}. Now, assume footpoints of AD_{ab} are not used for computing Voronoi segments (Otherwise, AD_{ab} will not be empty by maximum disc property of Voronoi diagram).

There can be a maximum of four circles that are parametrically close to the footpoints of AD_{ab} (as shown in pink color in Figure 3 (d)), and let these circles form a set C_1. Assume none of the circles in C_1 intersect AD_{ab}. If C_2 is the set of circles that are actually intersecting AD_{ab}, C_2 has circle or circles parametrically close to AD_{ab}. However, this contradicts our initial assumption of the set of parametrically close circles. Hence, it is enough to check emptiness with circles that have parametrically close DEs with AD_{ab}.

4. Complexity reduction by sampling

Considering ADs between all pairs of circles leads to $O(n^2)$ ADs. This can be reduced by considering neighborhood details available from the Delaunay graph of points representing centers of the input circles. Delaunay graph is the dual of Voronoi diagram where two points are connected by a Delaunay edge if they share a Voronoi edge between them. Let $\mathbb{D}_1 = \mathbb{D}_1(P)$ represents the Delaunay graph of center points of all input circles. In Figure 5(a), the center points of c_1 and c_2 are not neighbors in $\mathbb{D}_1$. Figure 5(b) shows the Voronoi diagram of the circles, where c_1 and c_2 have become neighbors. Here, it can be seen that AD between c_1 and c_2 is empty. In order to list this AD in the list $\mathbb{L}$ of touching disc method, 2-ring neighborhood details from $\mathbb{D}_1$ are to be used (neighbor's neighbor). Figure 5(c) shows that 3-ring neighborhood details (neighbor's neighbor's neighbor) are to be used to list the AD between the two largest circles. Again, Figure 5(d) shows a case where the number of neighborhood rings required is more than three as it appears that c_1 and c_2 are neighbors in the Voronoi diagram of circles.



(a) (b) (c) (d)

Figure 5: The Voronoi diagram (in black color) of circles is not the same as the Voronoi diagram of centers of circles (Delaunay graphs of centers of circles are shown in red color). (a) Voronoi diagram of centers of circles is shown in black color. (b)-(c) Larger circles are neighbors in the Voronoi diagram of circles. (d) It appears that c_1 and c_2 are neighbors in the Voronoi diagram of circles (filled circles in blue) even though they are not so in the Voronoi diagram of the centers of the circles.

4.1. Delaunay graph of sample points

As the Delaunay graph of center points of input circles does not give any desired neighborhood details, the next option is to select sample points from the circles and construct the Delaunay graph of these sample points. Let $S_i = \{s_j | j = 1, 2..m\}$, represents a set of m sample points selected for the circle c_i (by equally subdividing the circle). Hence, $S = S_1 \cup S_2 \cup ....S_n$ represents the set of sample points from all circles. Figure 6(a) shows sample points taken for a set of circles. Here, $m = 20$ as 20 samples are taken from each circle.

If $\mathbb{D}_2(S)$, represents Delaunay graph of all sample points, the list of ADs,
$\mathbb{L} = \{AD_{ij} | \exists$ *at least one Delaunay edge in* $\mathbb{D}_2(S)$ *connecting samplepoints* s_p *and* s_q *such that* $s_p \in S_i$ *and* $s_q \in S_j\}$.

Figure 6(b) shows $\mathbb{D}_2(S)$ (in red color) for sample points corresponding to $m = 8$. Corresponding list of ADs,

$\mathbb{L} = \{AD_{12}, AD_{13}, AD_{14}, AD_{15}, AD_{17}, AD_{23}, AD_{24}, AD_{25}, AD_{26}, AD_{27}, AD_{37}, AD_{45}, AD_{56}, AD_{67}\}$.

It is difficult to fix a value for m as it may cause oversampling for smaller radii circles and undersampling for bigger radii circles, and hence we are proposing a dynamic sampling approach.

5. Dynamic sampling

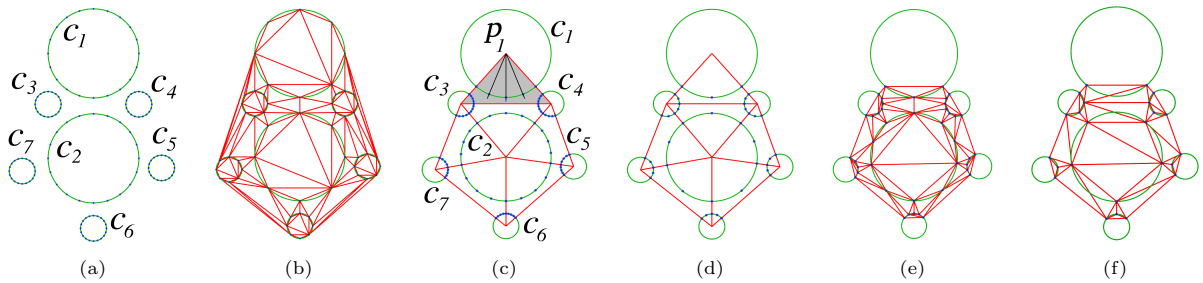
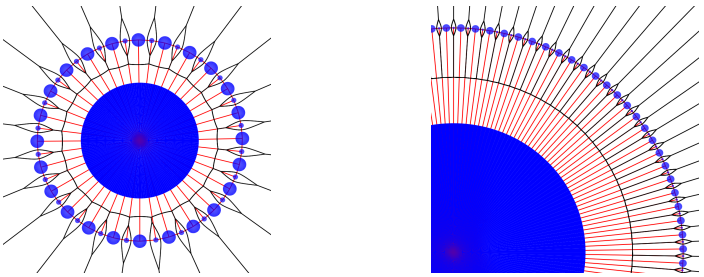


Figure 6: For capturing neighborhood details for the touching disc method, dynamic sampling requires fewer samples than uniform sampling. (a) Uniform sampling (20 samples per circle). (b) Delaunay graph of sample points (8 uniform samples per circle). (c) Dynamic sampling with $N_s = 3$. Face incident to p_1 is shown in grey color. (d) Dynamic sampling with $N_s = 1$. (e) Delaunay graph of sample points in (d). (f) Delaunay graph of sample points with $N_s = 0$.

If the largest circle in Figure 5(d) is of radius infinity, it is difficult to assign a value for m as it causes oversampling for smaller circles and an increase in computation time. Such a situation demands a varying number of samples corresponding to the radius of the circle. In order to address such variation in the number of sample points, initially, a Delaunay graph $\mathbb{D}_1 = \mathbb{D}_1(P)$ for the center points of circles is computed. Now, for each circle c_i, sample points are placed at the points where c_i intersects with the Delaunay edges of $\mathbb{D}_1(P)$ incident at p_i. The remaining sample points shown in the Figure 6(c) are placed as follows. Let, $F_i = \{f_{ij} | j = 1, 2..k_i\}$ represents the set of incident faces at p_i in $\mathbb{D}_1$, where k_i is the number of faces incident at p_i. Consider radial lines from p_i by dividing the intended angle of f_{ij} at p_i into $N_s + 1$ equal intervals, where N_s is number of angular sampling. Now, a sample point each can be placed at the intersecting point of the circle c_i with each of the N_s radial lines. Even though the value of N_s is static, dynamic sampling is effected when it is combined with the Delaunay graph of centers of input circles. In Figure 6(c), for $N_s = 3$, 20 samples are placed for circle c_2, whereas only 5 samples are placed for c_1. Thus, a dynamic variation in sample points can be seen between c_1 and c_2, even though they are of the same radius. Figure 6(d) shows dynamic sampling with $N_s = 1$. Figure 6(e) shows the Delaunay graph, $\mathbb{D}_2(S)$, where S is the sample points collected through dynamic sampling corresponding to $N_s = 1$. Neighborhood details from this $\mathbb{D}_2(S)$ are used for listing ADs, in the same way as explained for uniform sampling. Figure 6(f) shows the Delaunay graph of sample points collected through dynamic sampling corresponding to $N_s = 0$. Here, it can be seen that there is no Delaunay edge in $\mathbb{D}_2(S)$ connecting the two largest circles.

Dynamic sampling with an angular sampling of one requires only three sample points for the largest circle shown in Figure 5(d). For the input sets, as shown in Figure 7, dynamic sampling helps to place more sample points for the circle at the center. Details of dynamic sampling for Figure 7 are shown in Table 1.



(a) (b)

Figure 7: Dynamic sampling selects more samples from the circle at the center. In (a) and (b), considering the Delaunay graph of centers of circles (shown in red color), the largest circle has all the remaining circles as its neighbors.

Table 1: Sample details for input sets shown in Figure 7 ($N_s = 3$).

Figure number	**Input size**	**Sample size for the largest circle**	**Sample size for each of the other circles**	**Average samples per circle**
Figure 7(a)	49	192	9	12.735
Figure 7(b)	199	792	9	12.935

5.1. Complexity is independent of the number of samples in dynamic sampling approach

Let N is the total number of samples taken through dynamic sampling. For each circle, one sample is placed on the Delaunay edge (of $\mathbb{D}_1$) incident to the center of the circle. As the number of Delaunay edges is linear with respect to the number of sites, the number of samples placed along the Delaunay edges is also linear with respect to the number of circles.

Next, samples are placed by considering radial lines by equally subdividing the Delaunay faces (of $\mathbb{D}_1$) incident to the center of the circle. Here, the number of radial lines (decided by the number of angular sampling, N_s) considered for each Delaunay face is the same. As the number of Delaunay faces is also linear with respect to the number of sites and N_s is a constant, number of samples placed by subdividing the Delaunay faces is also linear. So the total number of samples obtained through dynamic sampling is linear with respect to the number of circles and complexity of the algorithm is independent of number of samples. In dynamic sampling, the number of samples is controlled by the parameter N_s.

6. The algorithm

Algorithm 1 handles the initialization procedure, where a list of ADs is prepared. This list is prepared based on the neighborhood details obtained through the Delaunay graph $\mathbb{D}_1(P)$ and $\mathbb{D}_2(S)$. A value for the number of angular sampling, N_s is set as either one or three during the initialization of the algorithm. Processing of ADs is done by Algorithm 2 and that of TTDs is done by Algorithm 3. Finally, Algorithm 4 computes the Voronoi diagram of a set of circles using Algorithm 1, Algorithm 2, and Algorithm 3 as subroutines.

Algorithm 4 terminates when all the ADs are processed, and no further TTDs are available for processing. The DEs remaining at this point cannot form further TTDs, and they connect circles on the convex hull of the input set. The unused portion of the circles (to which the leftover DEs connect) can be used to complete the Voronoi diagram.

Algorithm 1 *InitializeVD*($CircleList$)

1: Compute Delaunay graph $\mathbb{D}_1$ for the centers of input circles.
2: For each of the circles c_i, place a sample point at the point along each of the edges incidents to p_i that intersects with c_i.
3: Set number of angular sampling, N_s
4: Consider radial lines from p_i by making $N_s + 1$ angular subdivision of the intended angle of each of the faces f_{ij} at p_i, where f_{ij} is the face incident to p_i in $\mathbb{D}_1$. Place a sample point each along these radial lines that intersect with the circle c_i.
5: Compute Delaunay graph $\mathbb{D}_2$ for the set of sample points $\mathbb{S} = S_1 \cup S_2 \cup ..S_n$, where S_i is the set of sample points from the circle c_i.
6: Compute AD_{ij} between c_i and c_j, if there exists a Delaunay edge in $\mathbb{D}_2(S)$, connecting sample points s_p and s_q such that $s_p \in S_i$ and $s_q \in S_j$.
7: Initialize $\mathbb{L}$ as a list of all ADs obtained in step 6, sorted in ascending order of radii.
8: Get the first disc, AD_{ij} from $\mathbb{L}$.
9: Add DE_{ij} and DE_{ji} (as the smallest AD is empty).
10: Remove the disc from the list.

Algorithm 2 *ProcessAD*($\mathbb{L}$)

1: Pick the first disc in $\mathbb{L}$.
2: **if** the disc is a AD (say AD_{jk}) and the footpoints have not been used **then**
3: Find C_1 as the set of circles connected by DEs which are parametrically close to the footpoints of AD_{jk}.
4: **if** ($C_1 = \emptyset$ or circles in C_1 do not intersect AD_{jk}) **then**
5: Add DE_{jk} and DE_{kj}.
6: Check for parametrically closest DEs to form TTDs for both DE_{jk} and DE_{kj}.
7: Insert TTDs, if any, to $\mathbb{L}$ maintaining increasing order of radii.
8: **end if**
9: **end if**
10: Remove the disc from $\mathbb{L}$.

7. The correctness of the algorithm

The correctness of the touching disc method depends on identifying all empty ADs. With the dynamic sampling technique, ADs between all pairs of circles are not listed; rather, the pairs are decided based on the neighborhood details obtained through the Delaunay graph of sample points.

However, missing an empty AD always does not lead to a wrong Voronoi diagram. Assume AD_{ac} is empty and is missed from listing in L. For identifying an empty TTD, two DEs are enough (for TTD_{abc}, let the available DEs be DE_{ab} and DE_{bc}), and DE_{ac} will be placed while processing TTD_{abc}, using the footpoints of the TTD_{abc} on c_a and c_c. Here, DE_{ac} identifies two Voronoi segments; each of them otherwise would have been represented by DE_{ac} and DE_{ca}, where DE_{ac} and DE_{ca} would have used the footpoints of AD_{ac} as their endpoints.

Let $\mathcal{VD}(\mathcal{C})$ is the actual Voronoi diagram, and $\mathcal{VD}^*(\mathcal{C})$ is the one computed by Algorithm 4, for a set of circles $\mathcal{C}$. $\mathcal{VD}^*(\mathcal{C})$=$\mathcal{VD}(\mathcal{C})$ is proved through the following lemmas, assuming that $\mathcal{C}$ is a set of non-intersecting circles. Later, Section 9 explains how the algorithm handles the case of intersecting circles. Due to the instability in computing topology in the neighborhood of the zero-length Voronoi segment (Section 8.2), it is also assumed that the input circles are in general position.

Lemma 1. *During the execution of Algorithm 4 for computing $\mathcal{VD}(\mathcal{C})$, at least one DE is created connecting to $c_i \in \mathcal{C}$.*

Proof. Dynamic sampling collects samples from each circles, $c_i \in \mathcal{C}$. As the Delaunay graph is connected, at least a sample point from c_i creates a Delaunay edge by connecting a sample point from another circle, $c_j \in \mathcal{C}$. Among all ADs, between c_i and $c_j \in \mathcal{C}$, the smallest is empty. Hence, at least one empty AD_{ij} will be identified and corresponding DEs will be placed.

Algorithm 3 $ProcessTTD(\mathbb{L})$

Pick the first disc from $\mathbb{L}$.
if the disc is a TTD (say TTD_{ijk}) **then**
  Delete those DEs (if any) of TTD_{ijk} for which any of the endpoints are covered for computing the Voronoi segment.
  if all the three directed edges exist **then**
    Compute Voronoi segments for DE_{ij}, DE_{jk} and DE_{ki}.
    Delete DE_{ij}, DE_{jk} and DE_{ki}
  else if two of the DEs (say DE_{ij} and DE_{jk}) exist **then**
    Compute Voronoi segments for DE_{ij} and DE_{jk}.
    Delete DE_{ij} and DE_{jk}
    Add DE_{ik}.
    For DE_{ik}, identify parametrically closest DEs for computing TTDs.
    Insert TTDs, if any, appropriately in $\mathbb{L}$.
  else
    Remove the disc from $\mathbb{L}$.
  end if
end if

Algorithm 4 $ComputeVD(CircleList)$

$InitializeVD(CircleList)$
while $\mathbb{L}$ is not empty **do**
  $ProcessAD(\mathbb{L})$
  $ProcessTTD(\mathbb{L})$
end while

□

Lemma 2. *At the end of Algorithm 4, for computing the Voronoi diagram of a set of non-intersecting circles, no leftover DEs other than that form a continuous cycle of consecutive DEs exists.*

Proof. $\mathcal{VD}(\mathcal{C})$ consists of Voronoi segments extending to infinity, and DEs of such Voronoi segments are not deleted at the end of Algorithm 4 as they do not form TTDs. The convex hull of $\mathcal{C}$, $Hull(\mathcal{C})$, consists of straight line segments and arcs of circles Rappaport (1992). Each straight line segment of $Hull(\mathcal{C})$ can be assumed as an arc of a circle: of infinite radius; tangent to two circles in $\mathcal{C}$; and does not intersect or contain any circle $c_i \in \mathcal{C}$. While traversing the convex hull in the anticlockwise direction, the centers of such tangent circles lie in the exterior of $Hull(\mathcal{C})$. Hence, the straight line segments of $Hull(\mathcal{C})$ identify the pairs of circles connected by leftover DEs that form a cycle of consecutive DEs. □

Let $\mathcal{P}_H$ be the set of points in the circles that remains unused, for computing Voronoi segments, at the end of Algorithm 4 (Figure 8(a)). All points in $c_i \in \mathcal{C}$ that are not in $\mathcal{P}_H$ are used for computing Voronoi segments, and corresponding DEs are deleted. However, these Voronoi segments define a region $\mathcal{VC}_i^*$, and corresponding DEs of these Voronoi segments identify a set of circles, $\mathcal{B}_i^*$ (Figure 8(b)). $\mathcal{VC}_i^*$ and $\mathcal{VC}_j^*$ are adjacent if they share Voronoi segments. $\mathcal{VC}_i^*$ and $\mathcal{VC}_j^*$ are disjoint sets of points as Voronoi segments of c_i and c_j are constructed as a connected and planar graph using DEs. By recursively extending the above procedure to all $c_j \in \mathcal{B}_i^*$, we get a set of disjoint regions, $\mathcal{R}^* = \{\mathcal{VC}_1^*, \mathcal{VC}_2^*, ...\mathcal{VC}_n^*\}$, that divide the entire plane.

Let $\mathcal{VC}_i$ is the Voronoi cell of c_i corresponding to $\mathcal{VD}(\mathcal{C})$, and $\mathcal{B}_i$ is the set of neighboring circles. If the equality in the definition of Voronoi cell (Definition 8) is removed, Voronoi cells become disjoint. Hence, if HP_{ij} is the set of all points nearest to c_i than to c_j, we can define a Voronoi cell, $\mathcal{VC}_i = \bigcup_{i \neq j} HP_{ij}$. Here, HP_{ij} and HP_{ji} are separated by the bisector between c_i and c_j. Hence, $\mathcal{VC}_i$ can be considered as the lower envelope of bisectors between c_i and the remaining circles in $\mathcal{C}$. In the touching disc method, $\mathcal{VC}_i^*$ is

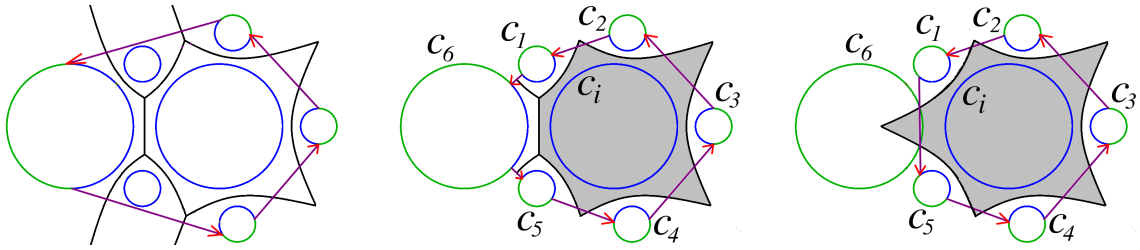


Figure 8: The correctness of the algorithm. The portion of circles used for computing Voronoi segments is highlighted in blue color. (a) Leftover DEs form a cycle and connect the circles in the convex hull. (b) All DEs connected to c_i are deleted after computing Voronoi segments defining the boundary of $\mathcal{VC}_i^*$ (in grey). Neighbors of c_i, $\mathcal{B}_i^* = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. The remaining Voronoi segments of $\mathcal{VC}_j^*$ of $c_j \in \mathcal{B}_i^*$ will be computed using the unused portion of c_j, making $\mathcal{VC}_i^*$ and $\mathcal{VC}_j^*$ as disjoint sets. (c) DEs after computing Voronoi segments of c_i, if $c_6 \notin \mathcal{B}_i^*$. Corresponding $\mathcal{VC}_i^*$ is shown in grey.

computed by constructing Voronoi segments between c_i, and $c_j \in \mathcal{B}_i^*$. Hence, to prove $\mathcal{VC}_i^* = \mathcal{VC}_i$ and the correctness, it is enough to prove $\mathcal{B}_i = \mathcal{B}_i^*$.

Lemma 3. *The Voronoi diagram computed by Algorithm 4 satisfying Lemma 1 and Lemma 2 is correct so that the computed Voronoi cells are disjoint, and adjacent/nonadjacent Voronoi cells in $\mathcal{VD}(\mathcal{C})$ remain adjacent/nonadjacent in the computed Voronoi diagram.*

Proof. If $c_m \in \mathcal{B}_i \setminus \mathcal{B}_i^*$, $\mathcal{VC}_i \subset \mathcal{VC}_i^*$ (Figure 8(c)). However, in the touching disc method, if $c_m \notin \mathcal{B}_i^*$, $c_i \notin \mathcal{B}_m^*$; and this implies, $\mathcal{VC}_m \subset \mathcal{VC}_m^*$. However, this leads to $\mathcal{VC}_i^* \cap \mathcal{VC}_m^* \neq \emptyset$. As it contradicts the set of disjoint regions, $\mathcal{R}^*$, $\mathcal{B}_i \setminus \mathcal{B}_i^* = \emptyset$; implies, adjacent Voronoi cells in $\mathcal{VD}(\mathcal{C})$ remain adjacent in the computed Voronoi diagram.

Now, if $c_m \in \mathcal{B}_i^* \setminus \mathcal{B}_i$, Voronoi segment between c_m and c_i forms a boundary for $\mathcal{VC}_i^*$, and leads to $\mathcal{VC}_i \subset \mathcal{VC}_i^*$. However, if $c_m \in \mathcal{B}_i^*$, $c_i \in \mathcal{B}_m^*$, and hence, $\mathcal{VC}_m \subset \mathcal{VC}_m^*$, implies, $\mathcal{VC}_m^* \cap \mathcal{VC}_i^* \neq \emptyset$. As it contradicts the set of disjoint regions, $\mathcal{R}^*$, $\mathcal{B}_i^* \setminus \mathcal{B}_i = \emptyset$; implies nonadjacent Voronoi cells in $\mathcal{VD}(\mathcal{C})$ remain nonadjacent in the computed Voronoi diagram. Thus, $\mathcal{B}_i = \mathcal{B}_i^*$, and the lemma.

□

DEs formed during the execution of the algorithm can be stored with $O(n)$ memory space, and leftover DEs can be found in $O(n)$ complexity at the end of the algorithm. Hence, a programmable correctness check of the computed Voronoi diagram is possible.

8. Addressing input circles in non-general position

The existence of Voronoi vertices (branch points) equidistant from more than three circles is a degeneracy for algorithms following an exact computation of the Voronoi diagram. Input set having such degeneracy is considered as in non-general position. During the course of computing the Voronoi diagram, Voronoi vertices of degree more than three may happen. Figure 9(a) shows a Voronoi diagram for a set of input circles, along with branch discs. All DEs which come into existence during the course of finding $\mathcal{VD}(\mathcal{C})$ are shown. Voronoi segment identified by DE_{ac} is shown in blue color. If the length of this Voronoi segment approaches zero (as shown in Figure 9(b)), TTD_{abc} and TTD_{acd} tend to coincide. If these branch discs coincide, a single branch point exists, which is equidistant from all four input circles. The Voronoi segment of DE_{ac} will be computed while processing TTD_{acd}, and if the distance between centers of the *disc of* R_{max} and *disc of* R_{min} (Definition 9) is zero or a user defined "near-zero", that implies the four circles bring a vertex of degree four. Hence, our approach allows Voronoi segments of zero length and thereby addresses the problem of Voronoi vertices of degree more than three without any special conditioning of the input set.

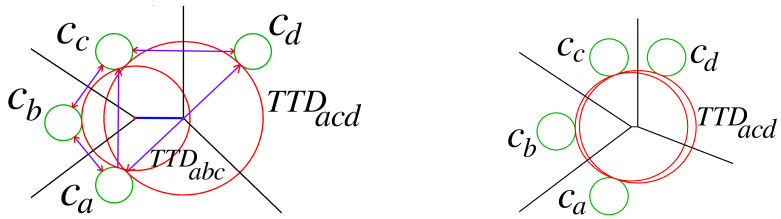


Figure 9: A Voronoi vertex equidistant from four circles is identified by two coinciding branch discs. (a) Line with an arrowhead each at both ends represents two coinciding DEs. The Voronoi segment of DE_{ac} is highlighted in blue color. (b) As the length of the Voronoi segment of DE_{ac} approaches zero, TTD_{abc} tends to coincide with TTD_{acd}.

8.1. The number of TTDs formed is linear if all circles are equidistant from a vertex

Enforcing parametric closeness for consecutive DEs restricts the number of TTDs that form during the execution of the algorithm. The way TTDs are formed in the case of non-general position input can be explained with a simple case. A non-general position input, where all the input circles are equidistant from a branch point is shown in Figure 10. For ease of explanation, all the 12 circles considered here are of the same radius. The formation of blue color DEs shown in Figure 10(a) leads to the formation of 12 TTDs (for example, DE_{ab} forms TTD_{abc} by pairing with DE_{bc} and TTD_{lab} by pairing with DE_{la}). Even though these TTDs are coinciding, the algorithm constructs it repeatedly on finding consecutive DEs. Here, the formation of TTDs is split into two stages. The first stage is completed once all the blue color DEs (as shown in Figure 10(a)) and associated 12 TTDs are formed.

The second stage is split into a number of levels such that further TTDs formed are accounted into each level. The second stage starts with the 12 DEs (in blue color) and the 12 TTDs (formed in the first stage). While processing TTD_{abc}, if DE_{ab} and DE_{bc} are available, TTD_{abc} is labelled as a branch disc and DE_{ac} is introduced (shown in red color in Figure 10(b)). Introduction of every red color DEs causes formation of two TTDs by pairing with two adjacent DEs. These red color DEs together with newly formed TTDs constitute the first level of the second stage (Figure 10(c)). Figure 10(d) shows the number of DEs in the second level, which is half of the number of DEs in the first level.

The above case can be generalized for a set of n input circles, and again for ease of explanation, it is assumed that n is sufficiently large and $n = 2^k$, where k is a positive integer. All circles are assumed to be of the same radius.

Number of TTDs formed in the first stage = n.

Number of TTDs formed in the first level of the second stage = n.

Number of TTDs formed in the second level of the second stage = $n/2$.

Now, in each of the subsequent level, the number of TTDs present are half of the previous level.

Thus the total number of TTDs formed = $n + n + n/2 + n/4 + n/8 + ... = 3n$.

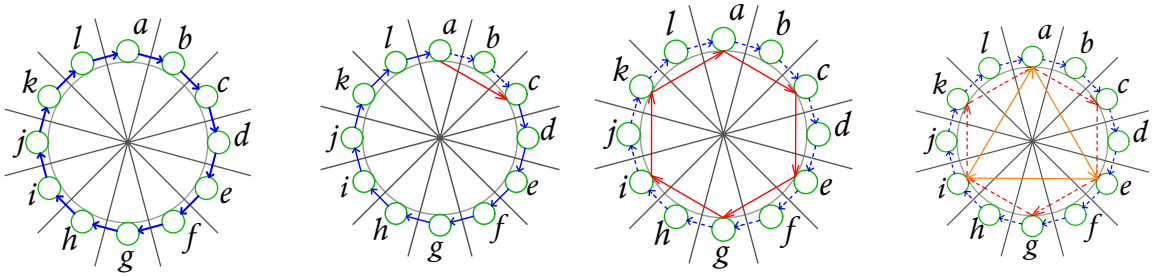
This upper limit of $3n$ for the number of TTDs for which the center is at equidistant from n circles will be same irrespective of n is even or odd.

8.2. Topology in the neighborhood of the zero-length Voronoi segment in non-general position

Each DE in Figure 10(a) forms two TTDs. For example, DE_{ab} forms TTD_{lab} and TTD_{abc}. If TTD_{abc} is processed before TTD_{lab}, DE_{ab} will not be available while processing TTD_{lab} and TTD_{lab} will not be identified as a branch disc. Hence, in the neighborhood of the zero-length Voronoi segment, the topology may be unstable.

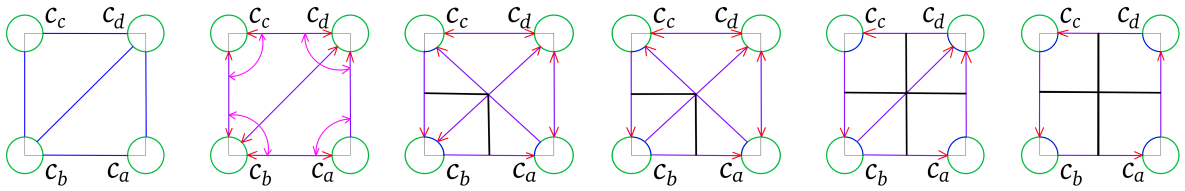
8.3. Handling crossing DEs in non-general position

Consider four circles of uniform radii with centers on a uniform grid, where AD between each pair of circles is empty. However, ADs included in the list, L based on the Delaunay graph of sample points are shown in Figure 11(a). As these ADs are empty, corresponding DEs are introduced, as shown in Figure



(a) (b) (c) (d)

Figure 10: DEs for which disc of R_{max} is the same (in ash color) are shown. (a) DEs in the first stage. Each of the blue color DEs leads to the formation of two TTDs. (b) DE_{ac} (in red color) leads to the formation of TTD_{acd} and TTD_{lac}. (c) DEs formed in the first level of the second stage (red color) lead to 12 TTDs (d) DEs formed in the second level of the second stage (Orange color) lead to the formation of 6 TTDs. In (b)-(d), deleted DEs are shown in dashed lines.



(a) (b) (c) (d) (e) (f)

Figure 11: Computing Voronoi diagram for a set of uniform circles with centers placed at uniform grid points (portions of circles used for computing Voronoi segment (in thick black line) can be seen highlighted in blue color). (a) Pairs of circles between which ADs are identified using dynamic sampling are connected using blue lines. (b) As all ADs are empty, DEs are placed. (c) DE_{ac} introduced after processing TTD_{abc} can be seen crossing DEs . (d) After processing TTD_{bcd}. (e) After processing TTD_{cda}. (f) After processing TTD_{dab}, DEs remaining are those for which Voronoi segments are extending to infinity.

11(b). These DEs cause the formation of four TTDs (DEs pairing to form TTDs are shown using arrow-headed curves in pink color). Let the sequential order in which these TTDs appear in the list $\mathbb{L}$ are: TTD_{abc}, TTD_{bcd}, TTD_{cda} and TTD_{dab}. When the turn of TTD_{abc} arrives for processing, it is identified as a branch disc leading to the introduction of DE_{ac} which crosses DE_{bd} and DE_{db} (Figure 11(c)). Next, while processing TTD_{bcd}, only two DEs (DE_{cd} and DE_{db}) are present, with the endpoint of DE_{db} on c_b already covered for computing Voronoi segment. Hence, DE_{db} is to be deleted, resulting in one DE (DE_{cd}) left for TTD_{bcd}. Hence, TTD_{bcd} is not identified as a branch disc (Figure 11(d)). Now, as the turn of TTD_{cda} arrives, it is identified as a branch disc as all three DEs exist, with endpoints not covered for the computation of Voronoi segment (Figure 11(e). For TTD_{dab}, only DE_{bd} remains, and that too will be deleted as both of its endpoints are covered for the computation of Voronoi segments (Figure 11(f)). Thus crossing DEs are handled by checking their endpoints.

9. Handling intersecting and hidden circles

The radius of AD between two circles for which x_i, x_j are center points and r_i, r_j are radii is computed as,

$R_{ij} = \|x_i - x_j\| - (r_i + r_j)$.

Even though the definition of AD is not applicable for two intersecting circles, the algorithm computes the radius of AD using the above equation and gets a negative value for R_{ij}. Assuming AD between two intersecting circles is empty, two DEs are placed using the footpoints identified along the line passing through the centers of the circles (Figure 12(a)). The algorithm uses such DEs for constructing TTDs using consecutive DEs (Figure 12(b)). Similar to non-intersecting circles, footpoints of DEs and TTD are used to

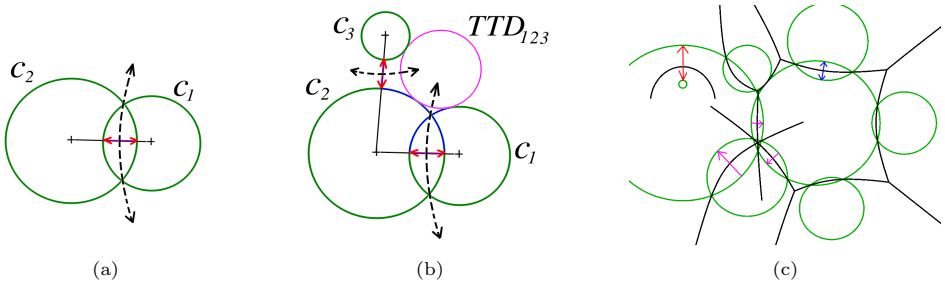


Figure 12: DEs and TTDs for intersecting circles. (a) Footpoints of DEs (DE_{12} and DE_{21}) are computed using the line passing through the centers of the circles. (b) Consecutive DEs, DE_{12} and DE_{23} form TTD_{123}. The portion of circles used for computing Voronoi segments while processing TTD_{123} is highlighted in blue color. (c) DEs in pink and red are leftover DEs at the end of Algorithm 4. Leftover DEs that form a cycle are not shown. Voronoi segments of all leftover DEs can be seen computed to a certain length, assuming they are extending to infinity.

identify the portion of circles (highlighted in blue color) that can be used to compute the Voronoi segments. As TTDs are constructed following its definition, TTD between three mutually intersecting circles in Figure 12(c) is not possible; corresponding DEs (in pink color) are left at the end of Algorithm 4. However, DEs shown in blue are not left at the end of the algorithm as they create TTDs, and Voronoi segments of these DEs are computed while processing this TTD. A circle that is completely contained in another circle is considered as a hidden circle/site Karavelas and Yvinec (2002). Hidden circles can be easily identified using the value of R_{ij}, and their DEs (shown in red color in Figure 12(c)) are also left at the end of the algorithm. In Figure 12(c), Voronoi segments of leftover DEs are computed to a certain length, assuming they extend to infinity (leftover DEs that form a cycle are not shown). The Voronoi diagram constructed external to all circles consists of a disjoint set of Voronoi cells, and leftover DEs that exist other than that form a closed cycle correspond to intersecting circles, not forming TTDs, and hidden circles. Hence, the correctness proof remains valid for the Voronoi diagram defined by the DEs, which are not left at the end of Algorithm 4.

10. Results and discussion

We implemented the algorithm in C++ using CGAL The CGAL Project (2020)(used only for computing Delaunay triangulation of a point set). ADs can be constructed using the distance between centers of the circles and radii of circles. TTDs are constructed using the algorithm of Gavrilova and Rokne (2003). The exact computation of points on the Voronoi edges can be computed using the 2-dimensional version of the bisector equations in Hu et al. (2017).

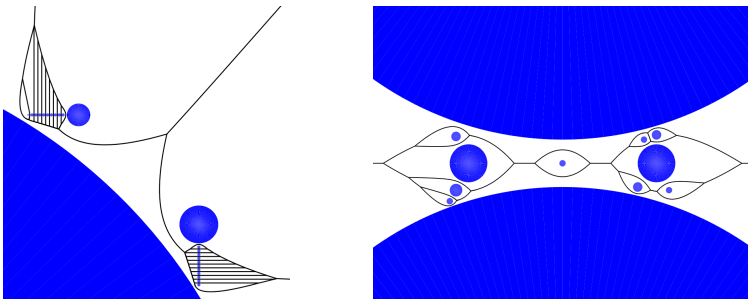


Figure 13: Results of the implementation. Voronoi diagram for (the input circles in blue color as filled circles): (a) the input set shown in Figure 5(d); and (b) a configuration given in Lee et al. (2016).

The effect of dynamic sampling can be seen in Table 1, which exhibits variation in sample size among input circles. For cases, as shown in the configuration of Figure 7, with an angular sampling of three, the

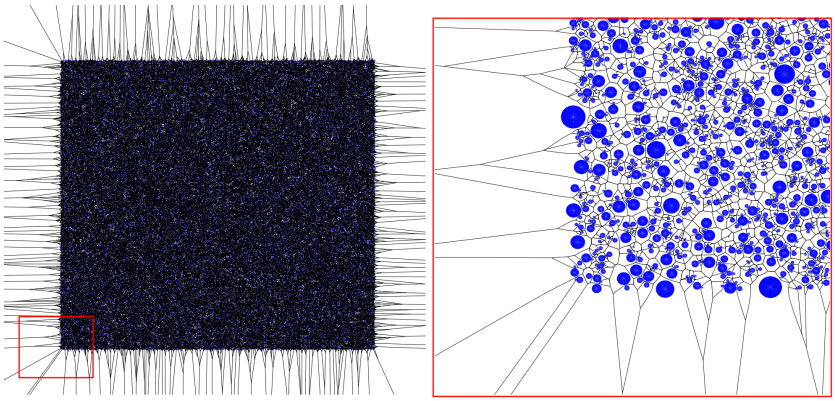


Figure 14: Voronoi diagram of 1×10^5 random circles and zoom-up of boxed region (input circles in blue color).

average number of samples per circle has an upper bound of thirteen. Figure 13(a) is a case where multi-ring neighborhood details are required from the Delaunay graph of center points of input circles for preparing the list of empty ADs. However, with a dynamic sampling, the number of samples used for the largest circle is only three, and the Voronoi diagram is computed correctly, with an angular sampling of one. Figure 13(b) shows the Voronoi diagram for a configuration adapted from Lee et al. (2016), where convex Voronoi cells, multiple Voronoi edges existing between same pairs of circles, and Voronoi cells having only two edges are captured correctly. Figure 14 shows the Voronoi diagram for 1×10^5 circles, where the centers of the circles are randomly placed in a domain where x and y coordinates range as $100 \leq x \leq 200$ and $100 \leq y \leq 200$. Results obtained against degenerate test cases are shown in Figure 15.

10.1. Robustness

The ability to handle non-general position input brings robustness to the proposed algorithm. In Figure 15(a) and (b), all circles in the input set are tangent to a common reference circle; circles are of non-uniform radii in Figure 15(a), whereas all 100 circles in the input set are of equal radii in Figure 15(b). Figure 15(c) shows a zoom-up of the center of the Voronoi diagram given in Figure 15(b), where the vertices are marked in red color. Due to the numerical artifact in producing the input set, multiple vertices are obtained in the center as opposed to a single one (similar to the result in Lee et al. (2016)). It may be noted that each zero length Voronoi edge simply adds to the degree of a Vertex beyond three and will not pose any hindrance, to the algorithm, against the correct computation of the Voronoi diagram. Hence, the Voronoi vertex with a degree of more than three is not an unexpected case for the proposed algorithm. In Figures 15(d)-(e), circles of equal radii are placed at uniform grid points. Figures 15(b)-(f) are test cases used in Lee et al. (2016) for the degeneracy cases.

In Figure 15(f), centers of the circles are placed at uniform grid points, and the result is presented in terms of the dual structure called the quasi-triangulation Kim et al. (2010). Here, an edge connecting centers of two circles indicates the existence of a Voronoi edge between them. The result is correct as all circles are connected with the triangular mesh, and every grid cell is divided into two triangles with no edge crosses another Lee et al. (2016). Constructing the dual structure is a straightforward task as DEs specify the pairs of circles having a Voronoi edge between them. DEs corresponding to zero length Voronoi segment ensures triangulation of the test case. Correctness check of the algorithm ensures topological and geometrical consistency of the computed Voronoi diagram, where topological consistency means partitioning the plane into n regions corresponding to n number of input circles and geometrical consistency means each of the input circles resides inside the corresponding Voronoi region with no two regions overlap.

Due to the error in input set, the circles may be intersecting or there can be a hidden circle of relatively low radius. Even though the input is supposed to be non-intersecting, our algorithm is capable of handling intersecting and hidden circles (Please refer to Figure 12(c) and Section 9). As branch points are computed using TTDs, computation of the intersection of bisectors can be avoided, which is more prone to numerical

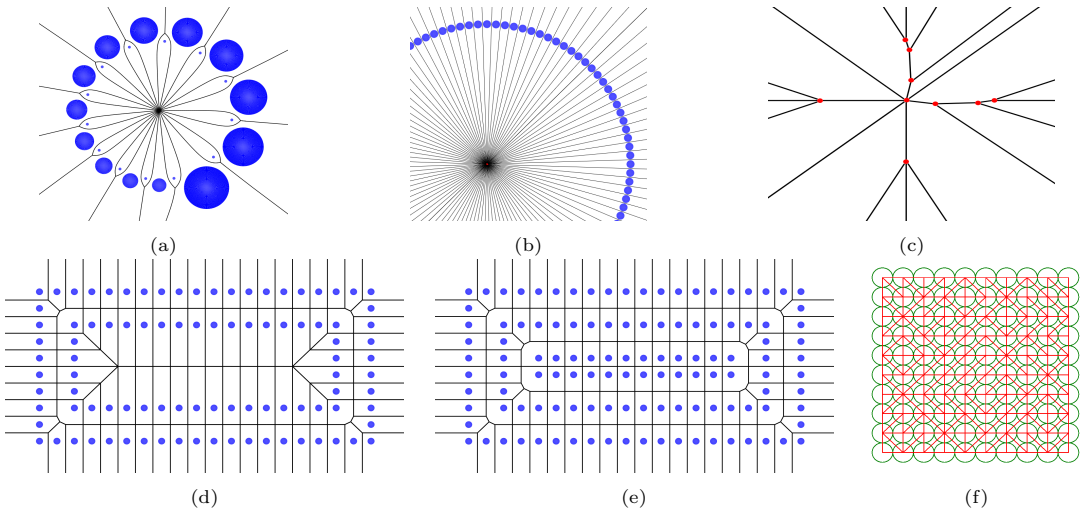


Figure 15: Results obtained against degenerate cases. (a)-(b) Voronoi diagram for circles that are tangent to a common reference circle. In (b), all 100 circles are of identical radii. (c) Zoom-up of the center of the Voronoi diagram shown in (b). (d)-(f) Voronoi diagram of identical circles placed at grid points. (f) Quasi-triangulation of circles placed at grid points. Circles in each column are of identical radii. The radius of a circle in the first left column is 1.0001, and the radii of circles in the right column are bigger than those of its left column by 0.0001.

errors. Dynamic sampling with $N_s = 1$, has correctly computed Voronoi diagram for all the test cases used in this paper.

10.2. Comparison of our approach with other methods

10.2.1. Algorithmic complexity

In our approach, each of the ADs considered are based on neighborhood details obtained from the Delaunay graph of sample points, and the number of samples created is of $O(n)$. Hence, the number of ADs computed by our algorithm is of $O(n)$ and so are TTDs (Please refer to Section 5.1 and Section 8.1 and see Figure 16(a)). Thus, the algorithmic complexity of the touching disc method based on dynamic sampling is decided by the algorithmic complexity in constructing the Delaunay graph of sample points. For randomly generated input sets, with $N_s = 1$, the average samples per set is approximately $12n$ (Figure 16(a)). For the configuration shown in Figure 7, it is $13n$ for $N_s = 3$. Hence, the overall algorithmic complexity of our algorithm in the worst case is $O(n\,log\,n)$.

Jin et al. (2006) compute the Voronoi diagram with a worst case time complexity of $O(n\,log\,n)$. However, the performance of the algorithm for non-general position input has not been demonstrated. Lee et al. (2016) address the issue of degeneracies in the input set but the algorithmic complexity is $O(n^2)$. Sample-based algorithm of Sugihara (1993) computes an approximate Voronoi diagram of circle set and requires fine samples at the expense of computational time. Our algorithm uses a bare minimum number of samples and follows an exact computation to address degeneracies in the input set to obtain a topologically consistent and geometrically accurate Voronoi diagram. The algorithm in Li et al. (2019) may require a restart where as ours do not.

10.2.2. Computational Complexity

Our approach is based on dynamic sampling and hence requires very few samples even for large circles (see Figure 5(d)). Figure 16(a) shows that for randomly generated sets of circles, the average number of samples used per circle is approximately 12 for $N_s = 1$. Hence, the computational time required for the algorithm is decided by the time taken for computing the Delaunay graph of sample points, which is approximately 12 times of the time required to do the same for the center point of input circles. This is significantly less than the computation of sample points used by Sugihara (1993), where unwanted edges

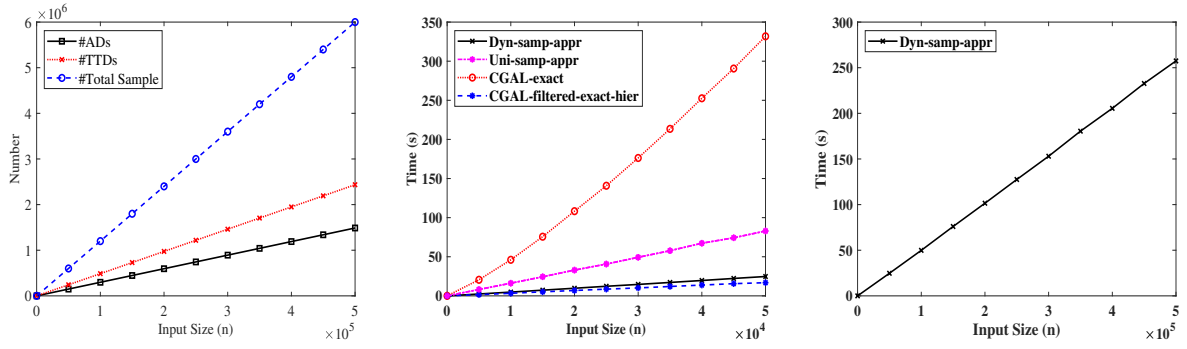


Figure 16: Results for random circles. (a) Variation of the number of ADs, TTDs and the total number of sample points (for $N_s = 1$) with the size of the input set. (b)-(c) Computation time comparison with CGAL-exact and CGAL-filtered-exact-hierarchy.

are also created in the Voronoi diagram of samples, demanding a trimming. On the contrary, no post-processing is required for our algorithm as no extraneous edges exist as in Lee et al. (2016). Figure 16(b) and (c) shows a time comparison of our algorithm with CGAL library (Computation environment: Intel Core i5-2400 CPU @ 3.10GHz × 4; 8GB RAM; Ubuntu 18.04.5 LTS [64bit]). In Figure 16(b) and (c), Dyn-samp-appr represents the implementation of our algorithm with a dynamic sampling approach with $N_s = 1$, and Uni-samp-appr represents the implementation with a uniform sampling approach with 50 samples per circles; CGAL-exact represents the implementation of Karavelas and Yvinec (2002) with exact number type; and CGAL-filtered-exact-hierarchy maintains a hierarchy of Voronoi diagram to accelerate an incremental construction of Voronoi diagram Devillers (1998). Here, the exact number type in CGAL-exact brings accuracy at the cost of computation time, whereas accuracy in topology and geometry is assured by our algorithm (Please refer to Section 7) at a lesser computation time, which is comparable to that of CGAL-filtered-exact-hierarchy. However, in both the CGAL codes, input in non-general positions is addressed by perturbing the input set. Time taken for uniform sampling approach is obviously more as more samples are required (For configurations in Figure 7(a) and (b), 192 and 792 samples per circles, respectively are ideally required).

Unlike the algorithm of Kim et al. (2001a,b), no special treatment is required around the boundary of the convex hull of the circles as leftover DEs easily circumvent this problem (convex hull of the circle set can be computed using leftover DEs). The topology oriented algorithm of Lee et al. (2016) orients about three cases, whereas our algorithm is consistent in dealing with TTDs to get the topology, which makes the algorithm easier to implement. The computation of touching discs (ADs and TTDs) is linear and does not take much time. Moreover, the algorithm is free of time consuming computations such as intersection of bisectors to obtain branch points (degree of a TTD has been shown to be lesser than that of a bisector Sundar et al. (2020)). Segment-wise computation of Voronoi edges progresses on identifying each of the empty TTDs avoiding any merging operations as in Sharir (1985). Finally, the correctness of the result with respect to topology and geometry of the Voronoi diagram also can be checked in linear time using leftover DEs (Please refer to Section 7). Hence, the algorithm is practically valid with a programmable correctness check on both topology and geometry and the computational complexity mentioned against computing both topology and geometry of the Voronoi diagram.

10.3. Limitations and future work

The computation time of the dynamic sampling approach mostly depends on the time taken for computing the Delaunay graph of sample points. However, this method is a good candidate for finding the Voronoi diagram for closed planar curves as the approximate computation of the Voronoi diagram by discretizing the input curves requires a large set of points to decently find the geometry and topology of the Voronoi diagram. Using touching disc method, a coarse sample is enough to provide the list of ADs required to

compute the Voronoi diagram. Hence, the future work is to develop an $O(n \log n)$ algorithm for computing the Voronoi diagram of a set of planar curves and the medial axis of a multiply-connected planar curves, using touching disc method. Moreover, extending this method to its 3D counterpart is also considered as a future work.

11. Conclusion

An algorithm for computing the Voronoi diagram for a set of circles was presented. The proposed dynamic sampling approach enabled the algorithm to achieve a complexity of $O(n \log n)$. It was also shown that the inputs in non-general position can be handled without employing any perturbation technique. The algorithm was demonstrated to be robust while handling large input data and precisely computed the Voronoi diagram without using any bisector intersection for finding branch points. Possible future works have also been mentioned.

References

Aichholzer, O., Aigner, W., Aurenhammer, F., Hackl, T., Jüttler, B., Rabl, M., 2009. Medial axis computation for planar free-form shapes. Comput. Aided Des. 41, 339–349.

Cornea, N.D., Silver, D., Min, P., 2007. Curve-skeleton properties, applications, and algorithms. IEEE Transactions on Visualization and Computer Graphics 13, 530.

Devillers, O., 1998. Improved incremental randomized delaunay triangulation, in: Proceedings of the fourteenth annual symposium on Computational geometry, pp. 106–115.

Devillers, O., Karavelas, M., Teillaud, M., 2015. Qualitative symbolic perturbation: a new geometry-based perturbation framework. INRIA , 34.

Edelsbrunner, H., Mücke, E.P., 1990. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Transactions on Graphics (tog) 9, 66–104.

Emiris, I.Z., Karavelas, M.I., 2006. The predicates of the Apollonius diagram: algorithmic analysis and implementation. Computational Geometry 33, 18–57.

Gavrilova, M.L., Rokne, J., 2003. Updating the topology of the dynamic Voronoi diagram for spheres in Euclidean d-dimensional space. Computer Aided Geometric Design 20, 231–242.

Hu, Z., Li, X., Krishnamurthy, A., Hanniel, I., McMains, S., 2017. Voronoi cells of non-general position spheres using the GPU. Computer-Aided Design and Applications 14, 572–581.

Jin, L., Kim, D., Mu, L., Kim, D.S., Hu, S.M., 2006. A sweepline algorithm for Euclidean Voronoi diagram of circles. Computer-Aided Design 38, 260–272.

Karavelas, M.I., Yvinec, M., 2002. Dynamic additively weighted Voronoi diagrams in 2D, in: European Symposium on Algorithms, Springer. pp. 586–598.

Kim, D.S., Cho, Y., Sugihara, K., 2010. Quasi-worlds and quasi-operators on quasi-triangulations. Computer-Aided Design 42, 874–888.

Kim, D.S., Kim, D., Sugihara, K., 2001a. Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology. Computer Aided Geometric Design 18, 541–562.

Kim, D.S., Kim, D., Sugihara, K., 2001b. Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry. Computer Aided Geometric Design 18, 563–585.

Lee, D.T., Drysdale, III, R.L., 1981. Generalization of Voronoi diagrams in the plane. SIAM Journal on Computing 10, 73–87.

Lee, M., Sugihara, K., Kim, D.S., 2016. Topology-oriented incremental algorithm for the robust construction of the Voronoi diagrams of disks. ACM Transactions on Mathematical Software (TOMS) 43, 1–23.

Li, X., Krishnamurthy, A., Hanniel, I., McMains, S., 2019. Edge topology construction of Voronoi diagrams of spheres in non-general position. Computers & Graphics 82, 332–342.

Mahboubi, H., Aghdam, A.G., 2013. An energy-efficient strategy to improve coverage in a network of wireless mobile sensors with nonidentical sensing ranges, in: 2013 IEEE 77th Vehicular Technology Conference (VTC Spring), pp. 1–5. doi:10.1109/VTCSpring.2013.6691875.

Rappaport, D., 1992. A convex hull algorithm for discs, and applications. Computational Geometry 1, 171–187.

Ryu, J., Lee, M., Kim, D., Kallrath, J., Sugihara, K., Kim, D.S., 2020. Voropack-d: Real-time disk packing algorithm using Voronoi diagram. Applied Mathematics and Computation 375, 125076.

Sharir, M., 1985. Intersection and closest-pair problems for a set of planar discs. SIAM Journal on Computing 14, 448–468.

Sugihara, K., 1992. A simple method for avoiding numerical errors and degeneracy in Voronoi diagram construction. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 75, 468–477.

Sugihara, K., 1993. Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams. CVGIP: Graphical Models and Image Processing 55, 522–531.

Sugihara, K., Iri, M., 1992. Construction of the Voronoi diagram for "one million" generators in single-precision arithmetic. Proceedings of the IEEE 80, 1471–1484.

Sugihara, K., Sawai, M., Sano, H., Kim, D.S., Kim, D., 2004. Disk packing for the estimation of the size of a wire bundle. Japan Journal of Industrial and Applied Mathematics 21, 259–278.

Sundar, B.R., Mukundan, M.K., Muthuganapathy, R., 2020. A unified approach towards computing Voronoi diagram, medial axis, Delaunay graph and α-hull of planar closed curves using touching discs. Computers & Graphics 89, 131 – 143. URL: http://www.sciencedirect.com/science/article/pii/S0097849320300613, doi:https://doi.org/10.1016/j.cag.2020.05.010.

The CGAL Project, 2020. CGAL User and Reference Manual. 5.2 ed., CGAL Editorial Board.

Wang, P., Yuan, N., Ma, Y., Xin, S., He, Y., Chen, S., Xu, J., Wang, W., 2020. Robust computation of 3D Apollonius diagrams, in: Computer Graphics Forum, Wiley Online Library. pp. 43–55.

Yap, C.K., 1990. Symbolic treatment of geometric degeneracies. Journal of Symbolic Computation 10, 349–370.