Robust and Fast Algorithm for a Circle Set Voronoi Diagram in a Plane

Deok-Soo Kim[1], Donguk Kim[1], Kokichi Sugihara[2], and Joonghyun Ryu[1]

[1] Department of Industrial Engineering, Hanyang University
17 Haengdang-Dong, Sungdong-Ku, Seoul, 133-791, Korea
dskim@email.hanyang.ac.kr
{donguk, jhryu}@cadcam.hanyang.ac.kr

[2] Department of Mathematical Engineering and Information Physics, Graduate School of Engineering, University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113, Japan
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. Robust and fast computation of the exact Voronoi diagram of circle set is difficult. Presented in this paper is an edge-flipping algorithm that computes a circle set Voronoi diagram using a point set Voronoi diagram, where the points are the centers of circles. Hence, the algorithm is as robust as its counterpart of point set. Even though the theoretical worst-case time complexity is quadratic, the actual performance shows a strong linear time behavior for various test cases. Furthermore, the computation time is comparable to the algorithm of point set Voronoi diagram itself.

1 Introduction

Let $\mathbf{P} = \{\mathbf{p}_i \mid i = 1, 2, \ldots, n\}$ be the set of the centers $\mathbf{p}_i$ of circles $\mathbf{c}_i$ in a plane, and $\mathbf{C} = \{\mathbf{c}_i \mid i = 1,2,...,n\}$ be the set of circles $\mathbf{c}_i = (\mathbf{p}_i, r_i)$ where r_i is the radius of $\mathbf{c}_i$. VD($\mathbf{P}$) and VD($\mathbf{C}$) are the Voronoi diagrams for $\mathbf{P}$ and $\mathbf{C}$, respectively. Suppose that we want to compute the exact VD($\mathbf{C}$) where the radii of possibly intersecting circles are not necessarily equal. Several researches exist on this or related problems. Lee and Drysdale first considered Voronoi diagram for a set of non-intersecting circles [13], and suggested an O($n\log^2 n$) algorithm. They also reported another algorithm of $O(nc^{\sqrt{\log n}})$ [1,2]. Sharir reported an algorithm computing VD($\mathbf{C}$) in O($n\log^2 n$), where the circles may intersect [18]. Yap reported an O($n\log n$) time algorithm for line segments and circles [23]. While all of the above algorithms are based on the divide-and-conquer scheme, Fortune devised an O($n\log n$) time algorithm based on line sweeping [4]. Recently, Gavrilova and Rokne reported an algorithm to maintain the correct topology data structure of VD($\mathbf{C}$) when circles are dynamically moving [5]. Sugihara reported an approximation algorithm for VD($\mathbf{C}$) by sampling several points on the circles and computing the Voronoi diagram of these points [19].

In this paper, we present an algorithm that computes the Voronoi diagram of circle set correctly, robustly and efficiently. The robustness issue is the most important concern in this paper. The principle idea is as the following. Given a correct point set

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 718-727, 2001.

Voronoi diagram of the centers of circles, we compute the correct topology of VD(**C**) by flipping the edges of VD(**P**) of the centers. Then, we compute the equations of Voronoi edges.

It turns out that this approach works quite well. Since our approach is computing the correct topology of VD(**C**) by changing the topology of VD(**P**), our algorithm is as robust as a point set Voronoi diagram algorithm, provided that the decision can be made correctly. Note that the theory on the robustness issue for the point set Voronoi diagram has been well-established. Even though the theoretical worst-case time complexity is quadratic, the actual performance shows a strong linear time behavior for various test cases. In addition, the algorithm is quite simple to implement. The *edge* and *vertex* will be used in this paper to mean a Voronoi edge and a Voronoi vertex.

In this paper, we assume that the degrees of vertices of VD(**P**) as well as VD(**C**) are three, and VD(**P**) is represented in an efficient data structure such as a winged-edge data structure [14,17] and available a priori by a robust code such as [20,21,22] which is based on the exact computation strategy [6,20]. We also assume that the algorithm to compute the circumcircle(s) of three given circles, which is discussed in another paper [11], is available.

2 Edge Flipping

When an edge **e** in Fig. 1(a) is changed to to **e** in Fig. 1(b), it is called that **e** is *flipped* to **e** . Hence a flipping operation changes the pointers among the vertices, edges and generators appropriately.

As shown in Fig. 2, there are three possible configurations of an edge of VD(**P**) for the flipping test. An edge of VD(**P**) may have either two circumcircles, only one circumcircle or no circumcircle at the vertices of the edge. Fig. 2 shows the cases. When a circumcircle does not exist at a vertex, an inscribing circle actually exists at the given configuration.

2.1 Case I : Two Circumcircles

In Fig. 3, there are two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ on an edge $\mathbf{e}_1$. Let $\mathbf{CC}_i$ be a circumcircle about three generators corresponding to a vertex $\mathbf{v}_i$. When $\mathbf{e}_1$ is considered, the

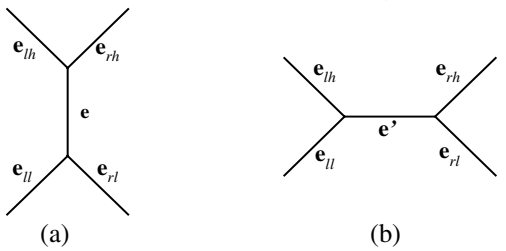


Fig. 1. Topology configuration after an edge flipping operation

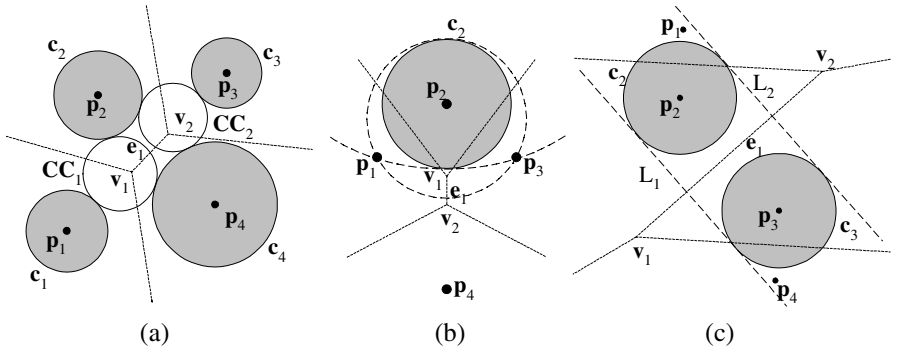


Fig. 2. Edge configurations

generator $\mathbf{c}_3$ is called a mating generator of $\mathbf{CC}_1$ and denoted as $\mathbf{M}_1$. When there exist circumcircles at the both ends of an edge, the circumcircles may or may not intersect with their mates.

Lemma 1. If both circumcircles do not intersect with their mates, the edge should not flip.

Proof. (Fig. 3(a)) The edge e_1 of VD(**P**) shown with dotted lines has two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$. The vertex $\mathbf{v}_1$ has three associated generators $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_4$, and the vertex $\mathbf{v}_2$ has three associated generators $\mathbf{p}_3$, $\mathbf{p}_4$ and $\mathbf{p}_2$. Let $\mathbf{CC}_1$ be a circumcircle to three circles $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$. From the definition of vertex $\mathbf{v}_1$ of VD(**P**), it can be determined that $\mathbf{CC}_1$ should be computed from $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$. Similarly, $\mathbf{CC}_2$ is a circumcircle to $\mathbf{c}_3$, $\mathbf{c}_4$ and $\mathbf{c}_2$. Note that we call $\mathbf{c}_3$ a mating generator of $\mathbf{CC}_1$. Since $\mathbf{CC}_1 \quad \mathbf{c}_3 =$ in the figure, any point inside or on $\mathbf{CC}_1$ is closer to $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$ than any point on $\mathbf{c}_3$. Similarly, $\mathbf{CC}_2 \quad \mathbf{c}_1 =$, and any point on $\mathbf{CC}_2$ is closer to $\mathbf{c}_2$, $\mathbf{c}_3$ and $\mathbf{c}_4$ than $\mathbf{c}_1$. Since same property holds for the centers of circles, the topology of VD(**P**) should be identical to the topology of VD(**C**). Therefore, the topology of VD(**P**) can be correctly used for the topology of VD(**C**) without any modification.

Lemma 2. If both circumcircles intersect with their mates, the edge should flip.

Proof. (Fig. 3(b)) The point set is identical to Fig. 3(a) and the radii of the circles are different. Note that both $\mathbf{CC}_1$ and $\mathbf{CC}_2$ intersect with mates $\mathbf{c}_3$ and $\mathbf{c}_1$, respectively. The fact that $\mathbf{CC}_1$ intersects with the mate $\mathbf{c}_3$ means that $\mathbf{c}_3$ has a point on the circle which is closer to the vertex $\mathbf{v}_1$ than any point on the three associated circles $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$. This suggests that the topology of vertex $\mathbf{v}_1$, as was given in VD(**P**), cannot exist as a member of vertex set in VD(**C**). Similarly, the vertex $\mathbf{v}_2$ cannot be a member of vertex set of VD(**C**), since $\mathbf{CC}_2$ also intersects with $\mathbf{c}_1$. Therefore, the edge $\mathbf{e}_1$ cannot exist in VD(**C**) as the topological structure given in VD(**P**) because both end vertices of the edge should disappear simultaneously. On the other hand, $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$ define a valid new vertex $\mathbf{v}_1$ and $\mathbf{c}_1$, $\mathbf{c}_4$, and $\mathbf{c}_3$ define another valid vertex $\mathbf{v}_2$. Topologically connecting $\mathbf{v}_1$ and $\mathbf{v}_2$ with an edge creates a new Voronoi edge $\mathbf{e}_1$. Therefore, a new edge $\mathbf{e}_1$ should be born while the old edge $\mathbf{e}_1$ disappears, and this results in an edge flipping.

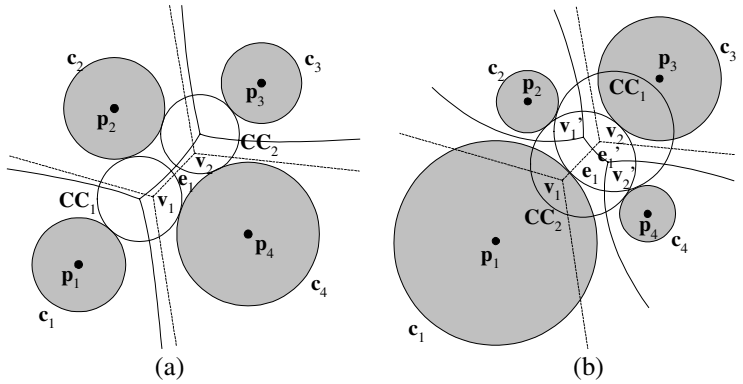


Fig. 3. Point sets in both figures (a) and (b) are identical, and therefore the point set Voronoi diagrams (shown with dotted lines) are identical as well. However, the corresponding circle set Voronoi diagrams (shown with solid curves) differ.

Between two circumcircles, it is possible that only one circumcircle intersects with its mating generator. Suppose that the circumcircles are $\mathbf{CC}_1$ and $\mathbf{CC}_2$ corresponding to $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively. Let $\mathbf{CC}_1 \quad \mathbf{M}_1$ and $\mathbf{CC}_2 \quad \mathbf{M}_2 =$. Since $\mathbf{CC}_1 \quad \mathbf{M}_1$, the topology of vertex $\mathbf{v}_1$ should be changed in the topology update process, while the topology of $\mathbf{v}_2$ should remain as it was given since $\mathbf{CC}_2 \quad \mathbf{M}_2 =$. Because of this small conflict, the current edge cannot be directly flipped. However, this conflict can be resolved by flipping another edge incident to the vertex $\mathbf{v}_1$ in a later step so that the topological structure of $\mathbf{v}_1$ is valid, while the topology of $\mathbf{v}_2$ remains at this moment. This observation provides the following lemma.

Lemma 3. If one circumcircles intersects with its mates, the edge should not flip.

2.2 Case II : One Circumcircle

Lemma 4. If one circumcircle exists, and the circumcircle intersects with its mate, the edge should flip.

Proof. (Fig. 4) As shown in Fig. 4(b) there is a case that no circumcircle, corresponding to vertex $\mathbf{v}_1$, exists to three generators $\mathbf{p}_1$, $\mathbf{c}_2$, and $\mathbf{p}_3$. Note that both dotted circles, associated to vertex $\mathbf{v}_1$, in the figure are not valid circumcircles, but circles inscribing $\mathbf{c}_2$. The fact that there is no circumcircle to three generator circles means the Voronoi vertex of three generator circles should disappear. In the given case, on the other hand, a circumcircle corresponding to vertex $\mathbf{v}_2$ exists and the circumcircle intersects with the mating generator $\mathbf{c}_2$. When this phenomenon happens an edge $\mathbf{e}_1$ should flip to $\mathbf{e}_1$.

Even though a circumcircle exists, it is possible that the existing circumcircle does not intersect with the mating generator circle. Obviously, the edge should not flip in this case and therefore the following lemma results.

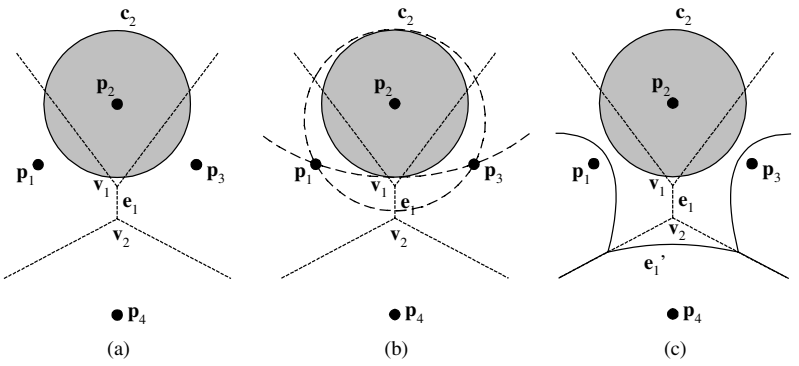


Fig. 4. A case that only one circumcircle exists and the existing circumcircle intersects with the mating generator.

Lemma 5. Only one circumcircle exists, and the circumcircle does not intersect with its mate, the edge should not flip.

2.3 Case III : No Circumcircle

It is even possible that an edge does not yield any valid circumcircles. Instead, only inscribing circles are defined by the circle generators. In this case, the edge does not flip as stated by the following lemma.

Lemma 6. When no circumcircle exists, the edge should not flip.

3 Special Cases Due to Convex Hull

While the above six lemmas guarantee the robust and correct transformation from the topology of VD(**P**) to that of VD(**C**), there could be a few special cases that need careful treatment. Let CH(**A**) be the convex hull of set **A**.

A flipping operation does not change the cardinality of topology while the generator stays inside of CH(**P**). Since it is viewed as a continual process, there could be an incident that four generators are circumcircular and one edge disappears. However, it is assumed that this case, which can be handled by extending the proposed algorithm, does not exist in this paper.

As the radius of a generator increases, a number of interesting and tricky problems may occur. The edges and vertices of VD(**P**) may sometimes disappear, and new edges and vertices, which were not in VD(**P**), are created when certain conditions are satisfied. Both cases, which have a direct relationship with the convex hulls of both generator sets, are elaborated in this section.

Similarly to a Voronoi diagram of a point set, a Voronoi region of $\mathbf{c}_i$ of VD(**C**) is infinite if and only if $\mathbf{c}_i$ CH(**C**) . Due to this observation, a Voronoi region defined by generators interior to CH(**C**) always defines a bounded region. Since

CH(**P**) and CH(**C**) may have different generators in their boundaries, there may be changes of bounded and unbounded regions in both Voronoi diagrams. This process involves the changes of the cardinality as well as the structure of the topology of Voronoi diagrams.

Suppose that a point **p** was a vertex of CH(**P**), and located interior to CH(**C**). Then, as will be discussed soon, one unbounded region of VD(**P**) becomes a bounded one in VD(**C**). This causes changes in the number of vertices and edges, too. The number of edges is increased by one, and so is the number of vertices. Similar phenomenon exists in the opposite case. In other words, when a point **p** was interior to CH(**P**) and the circle **c**, which corresponds to **p**, intersects with the boundary of CH(**C**), a bounded region now becomes an unbounded infinite region and creates one new vertex as well as one new edge. If there is no change between the generator sets that lie on the boundaries of CH(**P**) and CH(**C**), the number of edges, and therefore vertices, of VD(**C**) is identical to that of VD(**P**). The details of these cases related to the convex hulls are not discussed here.

4 Edge Geometry

Once the topology of VD(**C**) is fixed, it is necessary to compute the edge equations of VD(**C**) to complete the construction. The equation of a Voronoi edge of Voronoi diagram of circles is either a part of line or hyperbola [2,10]. The cases of parabolic and elliptic arcs do not occur in our problem. Persson and Held represented the edge equations using a parametric curve obtained by solving the intersection equations of the offset elements of generators [8,9,16]. In their representation, both line and hyperbola are represented in different forms. On the other hand, Kim used a rational quadratic Bézier curve to represent the edges. In this representation, any type of bisectors, for example, line, parabola, hyperbola, or ellipse, can be represented in a unified form, and hence used in this paper, too.

It is known that a conic arc can be converted into a rational quadratic Bézier curve form which is defined as

$$(t) = \frac{w_0(1 \quad t)^2\mathbf{b}_0 + 2w_1t(1 \quad t)\mathbf{b}_1 + w_2t^2\mathbf{b}_2}{w_0(1 \quad t)^2 + 2w_1t(1 \quad t) + w_2t^2} \qquad t \quad [0,1] \tag{1}$$

where $\mathbf{b}_0$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the control points, and w_0, w_1 and w_2 are the corresponding weights. It is known that a rational quadratic Bézier curve (t) representation of a conic curve can be computed if two points $\mathbf{b}_0$ and $\mathbf{b}_2$ on the curve, two tangents of the curve at $\mathbf{b}_0$ and $\mathbf{b}_2$, and another passing point **p** on the curve are known [3]. Among these five conditions, two points $\mathbf{b}_0$ and $\mathbf{b}_2$ are already known since the bisector should pass through two vertices of a Voronoi edge. Another passing point on the bisector can be obtained trivially as a point on the line segment defined by the centers of two generator circles and equidistant from two circles. Two last conditions of tangent vectors can be obtained by the following lemma which can be proved without much difficulty.

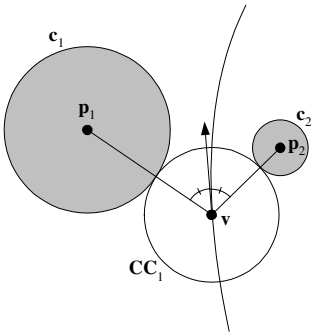
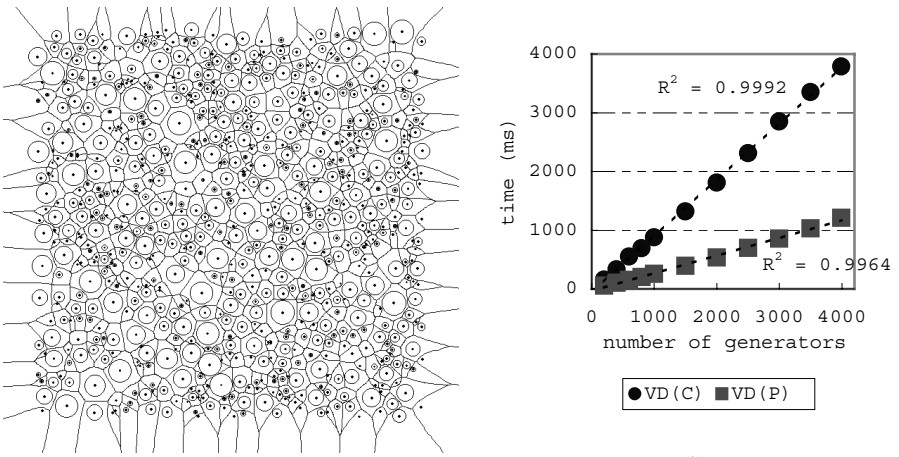


Fig. 5. A tangent vector on a bisector.

Lemma 7. Let a bisector (t) be defined between two circles $\mathbf{c}_1$ and $\mathbf{c}_2$. Then, the tangent line of (t) at a point $\mathbf{v}$ is given by an angle bisector of $\mathbf{p}_1\mathbf{v}\mathbf{p}_2$, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the centers of $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively.

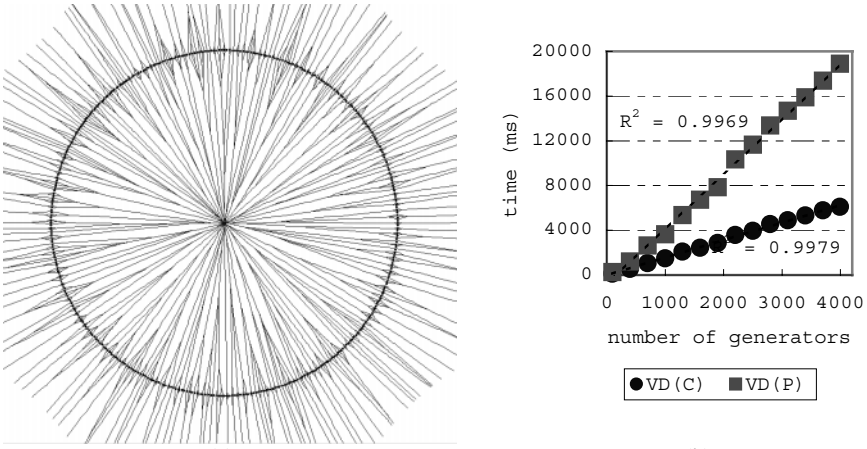
5 Implementation and Experiments

The proposed algorithm has been implemented and tested on MSVC++ on Intel Celeron 300MHz processor. Fig. 6 and Fig. 7 show two examples. In Fig. 6, 800 random circles are generated and do not intersect each other and have different radii. And In Fig. 7, the 400 non-intersecting circles with different radii generated on a large circle. Fig. 6(a) and Fig. 7(a) show results, and Fig. 6(b) and Fig. 7(b) show the computation time taken by a number of generator sets with varying cardinality. In the figure, the computation time taken by a code to compute the Voronoi diagram of point sets is denoted by VD(**P**), and the time taken by our code to compute the Voronoi



(a) (b)

Fig. 6. (a) Voronoi diagram of 800 random circles. (b) The computation time taken to compute the Voronoi diagram of point sets, VD(**P**), and our code to compute the Voronoi diagram of circle sets, VD(**C**).



(a) (b)

Fig. 7. (a) Voronoi diagram of 400 random circles on a large circle. (b) The computation time taken by a code to compute the Voronoi diagram of point sets, VD(**P**), and our code to compute the Voronoi diagram of circle sets, VD(**C**).

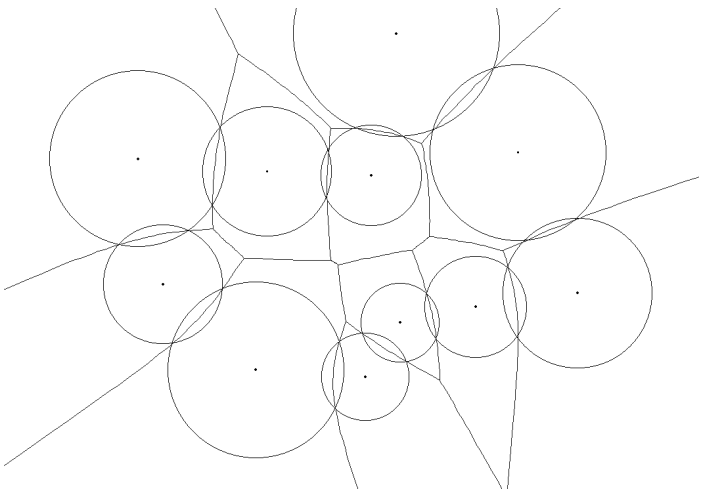


Fig. 8. An example when the generators intersect each other.

diagram of circle sets is denoted by VD(**C**). The point sets are the centers of circles generated at random, in this example. Note that the time denoted by VD(**C**) does not include the time taken by a preprocessing, which is actually the time denoted by VD(**P**). Therefore, the actual computation time to compute VD(**C**) from a given circle set is the accumulation of both computation times.

Comparing VD(**C**) with VD(**P**), it can be deduced that VD(**C**) is not as big as it might have been expected. Through experiences, there are cases that VD(**C**) is even much smaller than VD(**P**). Also, note that the correlation coefficient shown in the figure suggests that the average running behavior is a strong linear one. We have experimented with many other cases, and all the cases shows similar linear pattern.

Based on these experiments we claim that the proposed algorithm is very efficient and robust. Even though the worst-case scenario, which will given $O(n^2)$ time performance, is theoretically possible, it is difficult to expect to face such a case in reality. Fig. 9 shows that our algorithm works for the cases that the circles intersect each other. Fig. 9a shows the result of preprocessing which is the Voronoi diagram of point set, and 9b shows the Voronoi diagram of circle set.

6 Conclusions

Presented in this paper is an algorithm to compute the exact Voronoi diagram of circle set from the Voronoi diagram of point set. Even though the time complexity of the proposed algorithm is $O(n^2)$, the algorithm is quite fast, produces exact result, and robust.

The algorithm uses the point set Voronoi diagram of the centers of circles as an initial solution, and finds the correct topology of the Voronoi diagram of circle set by flipping the appropriate edges of the point set Voronoi diagram. Then, the edge equations are computed. Because our algorithm uses a point set Voronoi diagram, which has been studied extensively in its robustness as well as performance, the proposed algorithm is as robust as a point set Voronoi diagram.

Acknowledgements

The first author was supported by Korea Science and Engineering Foundation (KOSEF) through the Ceramic Processing Research Center (CPRC) at Hanyang University, and the third author was supported by Torey Science Foundation, Japan.

References

1. Drysdale, R.L.III, Generalized Voronoi diagrams and geometric searching, Ph.D. Thesis, Department of Computer Science, Tech. Rep. STAN-CS-79-705, Stanford University, Stanford CA (1979).
2. Drysdale, R.L.III, and Lee, D.T, Generalized Voronoi diagram in the plane, Proceedings of the 16th Annual Allerton Conference on Communications, Control and Computing, Oct. (1978) 833-842.
3. Farin, G., Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide, 4th edition, Academic Press, San Diego (1996).
4. Fortune, S., A sweepline algorithm for Voronoi diagrams, Algorithmica, Vol. 2 (1987) 153-174.
5. Gavrilova, M. and Rokne, J., Swap conditions for dynamic Voronoi diagram for circles and line segments, Computer Aided Geometric Design, Vol. 16 (1999) 89-106.
6. Gavrilova, M., Ratschek, H. and Rokne, J., Exact computation of Delaunay and power triangulations, Reliable Computing, Vol. 6 (2000) 39-60.

7. Hamann, B. and Tsai, P.-Y., A tessellation algorithm for the representation of trimmed NURBS surfaces with arbitrary trimming curves, Computer-Aided Design, Vol. 28, No. 6/7 (1996) 461-472.
8. Held, M., On the Computational Geometry of Pocket Machining, LNCS, Springer-Verlag (1991).
9. Held, M., Lukács, G. and Andor, L., Pocket Machining Based on Contour-Parallel Tool Paths Generated by Means of Proximity Maps, Computer-Aided Design, Vol.26, No. 3 (1994) 189-203.
10. Kim, D.-S., Hwang, I.-K. and Park, B.-J., Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves, Computer-Aided Design, Vol. 27, No. 8 (1995) 605-614.
11. Kim, D.-S., Kim, D., Sugihara, K., Ryu, J., Apollonius tenth problem as a Point Location Problem, (Submitted to ICCS 2001).
12. Kim, D.-S., Kim, D., and Sugihara, K., Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry, (Submitted to Computer Aided Geometric Design).
13. Lee, D.T. and Drysdale, R.L.III, Generalization of Voronoi diagrams in the plane, SIAM J. COMPUT., Vol. 10, No. 1, February (1981) 73-87.
14. Mäntylä, M., An introduction to solid modeling, Computer Science Press (1988).
15. Okabe, A., Boots, B. and Sugihara, K., Spatial Tessellations Concepts and Applications of Voronoi Diagram, John Wiley & Sons (1992).
16. Persson, H., NC machining of arbitrarily shaped pockets, Computer-Aided Design, Vol. 10, No. 3 (1978) 169-174.
17. Preparata, F.P. and Shamos, M.I. Computational Geometry An Introduction Springer-Verlag (1985).
18. Sharir, M., Intersction and closest-pair problems for a set of planar discs, SIAM J. COMPUT., Vol. 14, No. 2 (1985) 448-468.
19. Sugihara, K., Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams, Graphical Models and Image Processing, Vol. 55, No. 6 (1993) 522-531.
20. Sugihara, K., Experimental study on acceleration of an exact-arithmetic geometric algorithm, Proceedings of the IEEE International Conference on Shape Modeling and Applications (1997) 160-168.
21. Sugihara, K. and Iri, M., Construction of the Voronoi diagram for one million generators in single-precision arithmetic, Proc. IEEE 80 (1992) 1471-1484.
22. Sugihara, K., `http://www.simplex.t.u-tokyo.ac.jp/~sugihara/`, (2000).
23. Yap, C.K., An O(*n*log*n*) algorithm for the Voronoi diagram of a set of simple curve segments, Discrete Comput. Geom., Vol. 2 (1987) 365-393.