Robustness of Algorithm for k-Gon Voronoi Diagram Construction

Zhenming Chen *   Jinhui Xu*

Abstract

In this paper, we investigate the robustness of a plane sweep algorithm for constructing the Voronoi diagram of a set of non-crossing line segments in 2-dimensional space using a distance metric induced by a regular k-gon. Based on the *degree* model [13], we show that the Voronoi diagram of a set of arbitrarily oriented segments can be constructed with degree 14 under certain k-gon metrics (e.g. $k = 6, 8, 12$). For some restricted inputs, the degree is reduced to 2. The algorithm is easy to be implemented and finds applications in VLSI layout.

1 Introduction

The Voronoi diagram of polygonal objects is a fundamental geometric structure with numerous applications in diverse areas. Excellent surveys on the background, construction and applications of Voronoi diagrams can be found in [1, 2, 3, 17]. However, robustness issues associated with the construction of this diagram have prevented many of its uses in practice [20, 9]. In this paper we investigate the algorithmic *degree* of constructing the Voronoi diagram of segments under a regular k-gon distance metric as it can provide a robust alternative to the ordinary Euclidean Voronoi diagram for certain orientations and values of k (e.g., k=8).

The *degree* of an algorithm was introduced by Liotta, Preparata, and Tamassia [12, 13] to model its robustness and it reflects the lowest precision (i.e., the minimum number of bits) to which arithmetic calculations have to be executed in order to guarantee the topological correctness of the flow of the algorithm. Our motivation comes from applications in VLSI layout as presented in [15]. Note that VLSI shapes are always in fixed orientations, often consisting only of axis parallel segments and segments of slope ± 1. Due to the enormous size of VLSI data robustness and speed of computational methods are criteria far more important than full accuracy.

The most basic operation in the construction of any Voronoi diagram is the so called *incircle test*. In the case of line segments and the Euclidean metric Burnikel showed [4, 5] that in-circle test can be computed with degree 40 but this is rather high for practical applications. As pointed out by Yap and Mehlhorn [20], algorithms with lower degree predicates can lead to better efficiency (since the higher the degree of a predicate the greater the slow down due to the need of using arithmetic filters). In [15] the use of the L_∞ (resp. L_1) metric was proposed as a practical solution to the high degree problem associated with the construction of the Voronoi diagram of segments. It was shown that in L_∞ the in-circle test for arbitrary line segments can be answered with degree 5 and the plane sweep paradigm [8] resulted in an algorithm of total degree 7. The algorithm simplifies to degree 1 in the case of segments in a few constant orientations which is typically the case in VLSI applications.

Between the L_1 and the L_2 metric, a family of metrics has been defined based on a regular k-gon, $k = 4, 6, 8, \ldots, \infty$ [19, 18]. The L_1 metric is derived for $k = 4$ and the L_2 metric is reached as k approaches ∞. A similar family of metrics between the L_∞ and the L_2 metric can similarly be defined. It is thus natural to ask whether the k-gon metric can provide a sharper than the L_∞/L_1, yet robust, approximation to the L_2 Voronoi diagram. The *octagon* metric derived by an *isothetic* octagon i.e., an octagon with axis parallel diagonals or axis parallel edges, can very appropriately capture proximity information in VLSI designs where shapes consist typically of edges in axis parallel, $\pi/4, 3\pi/4$ orientations. In this paper we study the degree i.e., the robustness of such construction.

The Voronoi diagram of segments, more general the Voronoi diagram of disjoint convex polygons, under general convex polygonal distances was studied in [14]. The case of points had been investigated earlier in

*Department of Computer Science and Engineering, State University of New York at Buffalo, Bell Hall 201, Buffalo, NY 14260, USA. E-mail: [zchen4, jinhui]@cse.buffalo.edu. The research of the first and third authors was supported in part by a faculty start-up fund from the CSE department, SUNY at Buffalo, and by an IBM faculty partnership award.

[19, 6]. In [14] the combinatorial properties of this diagram were established and a plane sweep method based on Fortune's plane sweep paradigm [8] to construct a compact representation was given. The compact representation was motivated by applications in motion planning and post-office location queries. Given the compact representation the actual Voronoi diagram can be retrieved in time proportional to its size. In this paper we give a similar, but simpler, plane sweep algorithm to construct the actual Voronoi diagram of time complexity $O(nk \log nk)$ using the k-gon metric, where n is the total number of segments. The asymptotic time complexity bound is slightly worse than the one in [14], i.e., $O(nk + n \log n \log k)$. In practice this slight asymptotic difference is not important (since in general k is a small constant, e.g., $k = 8$) and thus we use the $O(nk \log nk)$ algorithm in our analysis for simplicity.

All the above diagrams fall under the umbrella of abstract Voronoi diagrams [10] and known techniques for the construction of such diagrams such as divide and conquer or randomized incremental construction can be applied once the computation of a bisector is available. For simplicity though plane sweep is often preferable in practice as long as the associated robustness issues can be resolved. Note that plane sweep methods are not available for abstract Voronoi diagrams and often demonstrate greater robustness problems.

2 k-gon Voronoi Diagram

This section discusses the concept of k-gon metric, the bisectors between points and segments, the plane sweeping method for constructing the k-gon Voronoi Diagram, and the associated events.

In Minkowski geometry, any convex set M can be used to define a *convex distance function* between any two points p, q as the amount that M needs to be scaled to include $q - p$. A special case is derived for M being a regular k-gon, termed the k-gon metric. The k-gon distance between two points p, q is the *size* of the smallest k-gon touching p and q. Depending on how we define the *size* of a k-gon, the k-gon distance can be an upper or a lower bound to the Euclidean distance. The *radius* of a regular k-gon P is the distance from the center of P to one of its vertices. The *diameter* is twice the radius of P. The *width* of P is the distance between two parallel sides of P. The *size* of P can be naturally defined either in terms of the diameter or the width of P. Defining the size of P as the diameter results in a family of metrics whose unit circles are inscribed in the ordinary unit circle, generalize the L_1 metric, and give an upper bound to the Euclidean distance. Restricting the orientations of the diameters of P to be in λ-geometry i.e., to form angles $i\pi/\lambda$ with the x-axis where λ is an integer that divides or is divided by 360 and $\lambda \geq 2$, we derive the well known λ-metric that has been extensively used in problems stemming out of VLSI routing.

Based on the k-gon metric, one can define the k-gon bisector of a pair of geometric objects, such as a pair of points, a point and a segment, two segments, a point and a line, and a segment and a line. Figure 1 gives some examples of different bisectors. A good property of these bisectors is that each of them consist of $O(k)$ line segments and no parabolic arc.

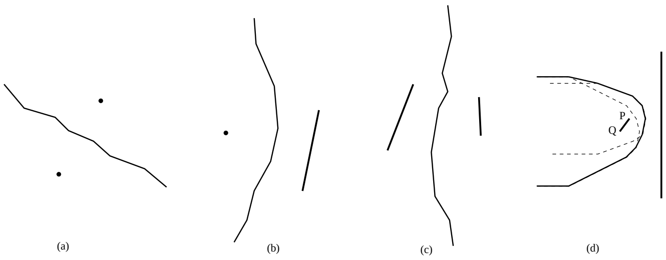

Figure 1: Bisectors of different pairs of geometric objects under the octagon metric.

Using these bisectors, we can define a k-gon Voronoi diagram for a set of geometric objects. Let G be a set of non-crossing planar segments. The Voronoi diagram of G, denoted by $V(G)$, is a partition of the plane into regions, called *Voronoi cells*, such that each cell is associated with an site of G. A point q lies in the cell corresponding to a site p_i if and only if the distance from q to p_i is smaller than the distance from q to any other site; $d(q, p_i) < d(q, p_j)$ for each $p_j \in G, j \neq i$. In $V(G)$, *Voronoi edge* is part of bisector between its two neighboring sites, and *Voronoi vertex* is the point equidistance to three or more sites.

A simple and efficient way for constructing the Voronoi diagram $V(G)$ is to use the plane sweeping method in [7, 8, 16]. Due to the special property of the k-gon metric, our plane sweeping algorithm has some interesting features which enable us to achieve an efficient and robust solution for certain values of k. Below we briefly discuss them.

First of all, we notice that during the process of plane sweeping, we are actually tracking the topological change of a beachline, which is the bisector between sweepline and the sites to the left of the sweepline. The bisector between a point v and the sweepline L is symmetric about a horizontal line due to the symmetry of the regular polygon, and contains two unbounded horizontal ray with the largest and smallest y-coordinates. For a segment s, its bisector with the sweepline L can be derived from the bisectors of its two endpoints with L. Let P_1 and P_2 be two regular k-gons of radius r and centered at the two endpoints v_1 and v_2 of s respectively. Assume that the k vertices of P_1 and P_2 are all labeled counterclockwise starting at their rightmost ones. Then, there exist two numbers j and $j + k/2$ such that connecting the two j^{th} vertices and the two $(j + k/2)^{th}$ vertices of the two regular k-gons results in a convex hull of the two regular polygon, and is the locus of equidistance from the segment s. We call the number j and $j+k/2$ as the *index of tangency*, it's easy to see the index of tangency can be determined by the slope of the input segment.

Secondly, we use a big enough axis-aligned bounding box to hold all the input segments and contain the topological structure of $V(G)$. The sweepline L sweeps across the bounding box from left to right, and the Voronoi diagram $V(G)$ is incrementally constructed during this sweeping process. Let G_t be the portion of segments or subsegments (i.e., the left portion of those segments in G intersected by L) to the left of L at time t. To compute the Voronoi diagram $V(G_t)$ of G_t, we dynamically maintain the set $B(G_t)$ of bisectors of L and the segments in G_t. The lower envelop (looking horizontally) of $B(G_t)$ is called the *beachline*, and contains two types vertices (called *breakpoints*): intersections of segments in the bisector of a site and intersections of segments in bisectors of two different sites. For the first order Voronoi diagram, it is sufficient to keep track of the change of the beachline, determine the vertices and edges of $V(G_t)$, and finalize them as part of $V(G)$. During the sweeping, two types of events (following [7, 16]), *site event* and *circle event*, need to be handled.

A *site event* occurs at the moment when sweepline reaches the leftmost point of a segment. Initially, the new segment introduces a degenerated bisector, which is a horizontal ray, into the beachline. Later on, this degenerated bisector grows into a full k-segment bisector. Due to the growth of these bisectors, two adjacent breakpoints on the beachline could merge into one breakpoint and form a *circle event*. At that time, three segments meet at a common point. (Note that if there are more than three segments meeting at a common point, a degenerate case will be introduced, and the duplicate vertices can be merged into one in the post-processing [8].) A circle event makes a breakpoint and the incident segment out of the beachline and thus can be removed from further consideration. During the sweeping, a k-segment bisector is created if and only if a site event occurs, and a segment in the beachline is removed if and only if a circle event occurs.

3 Algorithm for k-Gon Voronoi Diagram

Our plane-sweep algorithm uses two main data structures, one for the *beachline* and the other for the *event list*. Based on the y-monotone property, the beachline is maintained by a balanced binary tree T with each node corresponding to a breakpoint. The event list is stored in a priority queue, called *event queue* Q. The priority of each event is the x-coordinate when the event will occur. For a site event, the node is simply the x-coordinate of the endpoint of the corresponding line segment. For a circle event, the data structure stores the corresponding segments and breakpoints involved and a flag indicating which breakpoint is going to be deleted.

The plane sweeping algorithm first determines the indices of tangency for each input segment and initializes the event queue Q with all the site events. Then it repeatedly extracts an event from Q until Q is empty. For a site event, it first searches T to locate the intersection point of the degenerated bisector with the beachline. The intersection point becomes a new breakpoint of the beachline. Then, it generates two new potential circle events for the newly added breakpoint and its two neighboring breakpoints. The generated circle events could override (or be overridden by) some existing circle events in Q associated with the involving breakpoints, thus proper updating is needed to remove those *false alarm* events. For a circle event, the algorithm marks the intersection point as a Voronoi vertex, removes the breakpoint and segment from the beachline, generates a new circle event for the remaining breakpoint and its new neighbor in the beachline, and removes possible false alarm events related to the new circle event.

The algorithm runs in $O(nk \log(nk))$ time and $O(nk)$ space for n input segments. The reason is the follows. Each bisector has $O(k)$ bisector segments and there are at most $O(nk)$ breakpoints in the wavefront. Thus, an insertion or deletion operation in the tree T takes

$O(\log(nk))$ time. Further, for the event queue, there are n site events, and at most $O(nk)$ circle events. Thus the running time of each priority queue operation takes $O(\log nk)$ time. The major cost for each event is to have $O(1)$ operations on the two data structures and thus can be bounded by $O(\log(nk))$ time. The total time of the algorithm is therefore $O(nk\log(nk))$.

4 Degree Analysis

To analyze the degree, we consider an input segment $PQ : (x_p, y_p) - (x_q, y_q)$ with $x_p > x_q$ and $y_p > y_q$ (other cases are similarly handled). The formula of bisector segment between input segment and the sweepline $L : x = t$ can be either

$$y = A_i(x - x_g - d_g B_i) + y_g + d_g C_i, \text{or}$$

$$D_i x - d_x y + (d_x \sin i\theta - d_y \cos i\theta) d_p + d_x y_p - D_i x_p = 0,$$

where $\theta = 2\pi/k$, $g \in \{p, q\}$,

$$A_i = \frac{\sin(i+1)\theta\ (1+\cos i\theta) - \sin i\theta\ (1+\cos(i+1)\theta)}{\cos(i+1)\theta\ (1+\cos i\theta) - \cos i\theta\ (1+\cos(i+1)\theta)},$$

$$B_i = \frac{\cos i\theta}{1+\cos i\theta},$$

$$C_i = \frac{\sin i\theta}{1+\cos i\theta},$$

$$D_i = d_y(1+\cos i\theta) - d_x \sin i\theta,$$

$$d_g = t - x_g,$$

$$d_x = x_p - x_q, \text{and}$$

$$d_y = y_p - y_q.$$

In the above equations, $\sin(i\theta)$ and $\cos(i\theta)$, $0 \le i < k$, are in general radicals, seemingly suggesting that the degree is high. To accurately analyze the degree, we consider the case of $k = 8$. In the octagon metric, all the $\sin(i\theta)$ and $\cos(i\theta)$ are multiple of $\sqrt{2}$. Let $\beta = \sqrt{2}$ (Note that in general β can be any radical of the form $\frac{c_1 + c_2\sqrt{s}}{c_3 + c_4\sqrt{s}}$ for some rational numbers c_1, c_2, c_3, c_4 and s) and using the following simple rewriting rules:

1. $\beta \pm \beta \longrightarrow \beta$,
2. $\beta\beta \longrightarrow \beta$,
3. $\beta^{-1} \longrightarrow \beta$,
4. $\beta\alpha^n \longrightarrow \alpha^{2n}$,
5. $\beta\alpha^m \pm \beta\alpha^n \longrightarrow \beta\alpha^{\max\{m,n\}}$.

In the rewriting rule, we use α to represent the input variable whose degree is 1. In our algorithm, the coordinate x_g,y_g of point g, are input variables with degree 1. Use the rewriting rule, we can first rewrite A_i, B_i, C_i, D_i as $A_i \longrightarrow \beta$, $B_i \longrightarrow \beta$, $C_i \longrightarrow \beta$, $D_i \longrightarrow \alpha\beta$. We notice that A_i can be rewrite as β because all the $\sin(i\theta)$ and $\cos(i\theta)$ are multiple of same radical. In general, if we choose other k with the same property such as $k = 6, 12$, the degree of the algorithm will be the same as $k = 8$. If k does not have such a property, the degree will be higher.

In our plane sweeping algorithm, 4 types of predicates need to be answered exactly to ensure the accuracy: (BP means a breakpoint)

(A) $\alpha < \tan(i\pi/k)$ (for comparing the slope of input segment with the slope of a side of the regular polygon).

(B) $p <_y BP$ (for comparing the y-value of a point p with a breakpoint in the wavefront and determining the initial position of a bisector line segment).

(C) $p <_x BP$ (for comparing the x-value of a point p with the x-value of a break point and inserting a new spike event into the event queue).

(D) $BP_1 <_x BP_2$ (for comparing the x-value of two break points and inserting a new spike event into the event queue).

Using these rewriting rules, we can show (details are left to the full version) that Predicate (D) has the highest degree when three segments intersect at a common point. Note that in the octagon metric case, we only have one radical number which can be computed in an off-line fashion (and thus treated as a constant) to a precision which ensures the correctness of the exact computation. For example, if the desired precision is b bits, and the degree of the algorithm is d, then the radical number can have a precision with $d \times b + O(1)$ bits.

For Predicate (D), we let the three intersecting segments be

$$a_i x + b_i y + c_i t = d_i,$$

$$a_j x + b_j y + c_j t = d_j, \text{and}$$

$$a_k x + b_k y + c_k t = d_k.$$

Solve this linear system, we have

$$x = \begin{vmatrix} b_i & b_j & b_k \\ c_i & c_j & c_k \\ d_i & d_j & d_k \end{vmatrix} \Bigg/ \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{vmatrix} \qquad (1)$$

$$y = -\begin{vmatrix} a_i & a_j & a_k \\ c_i & c_j & c_k \\ d_i & d_j & d_k \end{vmatrix} \Bigg/ \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{vmatrix} \qquad (2)$$

$$t = \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ d_i & d_j & d_k \end{vmatrix} \Bigg/ \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{vmatrix} \qquad (3)$$

Use rewriting rules, we can get $x = \frac{\alpha^4\beta}{\alpha^3\beta}$, $y = \frac{\alpha^4\beta}{\alpha^3\beta}$. Plug these into Predicate (D), we have $\frac{\alpha^4\beta}{\alpha^3\beta} - \frac{\alpha^4\beta}{\alpha^3\beta} \quad \rightarrow \quad \alpha^7\beta \quad \rightarrow \quad \alpha^{14}$.

In VLSI layout, the input segments are often rectilinear or oriented with slope ± 1. In this case, $x = \frac{\alpha\beta}{\beta}$, $y = \frac{\alpha\beta}{\beta}$, and the degree of Predicate (D) is $\frac{\alpha\beta}{\beta} - \frac{\alpha\beta}{\beta} \quad \rightarrow \quad \alpha\beta \quad \rightarrow \quad \alpha^2$.

Theorem 1. *For a set of non-crossing segments with arbitrary orientation, the plane-sweeping algorithm for computing the octagon Voronoi diagram has degree* 14*. If the orientation of the segments is either rectilinear,* $\pi/4$*, or* $3\pi/4$*, then the degree is* 2.

Acknowledgement We would like to thank Evanthia Papadopoulou for valuable contributions.

References

[1] F. Aurenhammer, "Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure," *ACM Computer Surveys*, 23:345–405, 1991.

[2] F. Aurenhammer and R. Klein, "Voronoi Diagrams," in J. Sack and G. Urrutia (eds.) *Handbook of Computational Geometry*, pp. 201–290, Elsevier Science Pub., 2000

[3] Mark de Berg, Marc van Kreveld, Mark Overmars, Otfried Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer-Verlag, 2000.

[4] C. Burnikel, "Exact Computation of Voronoi Diagrams and Line Segment Intersections," Ph. D. Dissertation, Universität des Saarlandes, Saarbrücken, Germany, 1996.

[5] C. Burnikel, K. Mehlhorn, and S. Schirra, "How to Compute the Voronoi Diagram of Line Segments: Theoretical and Experimental Results," *Proc. ESA*, LNCS 855, pp. 227–239, 1994.

[6] L. P. Chew and R. L. Drysdale, "Voronoi diagrams based on convex distance functions. In *Proc. ACM Symp. Comp. Geom.*, pages 235-244, 1985.

[7] F. Dehne and R. Klein, "The Big Sweep: On the Power of the Wavefront Approach to Voronoi Diagrams," *Algorithmica*, 17:19–32, 1997.

[8] S. J. Fortune, "A Sweepline Algorithm for Voronoi Diagrams," *Algorithmica*, 2:153–174, 1987.

[9] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap, "A Core Library For Robust Numeric and Geometric Computation", *ACM Symp. on Comp. Geometry*, 1999.

[10] R. Klein, Concrete and Abstract Voronoi Diagrams. Number 400 in LNCS. Springer-Verlag, 1989.

[11] R. Klein, D. Wood, Voronoi diagrams based on general metrics in the plane, *Proc. 5th Symp. on Theoretical Computer Science*, 1988, 281-291, LNCS 294, Springer Verlag.

[12] G. Liotta, F.P. Preparata, and R. Tamassia, Robust Proximity Queries: An illustration of degree-driven algorithm design, SIAM Journal on Computing, 28,3,1998,864-889.

[13] G. Liotta, F.P. Preparata, and R. Tamassia, "Robust Proximity Queries in Implicit Voronoi Diagrams," Technical Report CS-96-16, Center for Geom. Comput., Comput. Sci. Dept., Brown Univ.,1996.

[14] Michael McAllister, David Kirkpatrick, Jack Snoeyink, A Compact Piecewise-Linear Voronoi Diagram for Convex Sites in the Plane. *Discrete & Computational Geometry*, 15:73–105, 1996.

[15] E. Papadopoulou and D. T. Lee, "Critical Area Computation via Voronoi Diagram," *IEEE T rans. on Computer-Aided Design*, 18:463–474, 1999.

[16] E. Papadopoulou and D. T. Lee, "The L_∞ Voronoi Diagram and Applications to VLSI Layout and Manufacturing", *International Journal of Computational Geometry and Applications*, Vol. 11, No. 5, 2001, 503-528.

[17] F. P. Preparata and M. I. Shamos. Computational Geometry - An Introduction. Springer-Verlag, New York, 1985.

[18] M. Sarrafzadeh and C.K. Wong, "Hierarchical Steiner Tree Construction in Uniform Orientation," *IEEE Tran. on Comp.-Aided Design*, Vol. 11, No. 8, 1992, pp. 1095-1103.

[19] P. Widmayer and Y. F. Wu and C. K. Wong, "On some distance problems in fixed orientations," *SIAM J. Comput.*, 16:728–746, 1987.

[20] Chee Yap and Kurt Mehlhorn, "Towards Robust Geometric Computation", *Fundamentals of Computer Science Study Conference*, July 25-27, Washington DC.