Voronoi Diagrams over Dynamic Scenes

(Extended Abstract)

Thomas Roos*

Abstract

Given a finite set S of n points in the Euclidean plane $\mathbb{E}^2$, we investigate the change of the Voronoi diagram $VD(S)$ and its dual, the Delaunay triangulation $DT(S)$, under continuous motions of the underlying points.

It is the idea to use the topological dual of the Voronoi diagram that is shown to be locally stable under sufficiently small continuous motions, in opposite to the accompanying Voronoi diagram which is reconstructed from its dual only when it is needed. We present an efficient, numerically stable update algorithm for the topological structure of the Voronoi diagram in a dynamic scene, using optimal $O(\log n)$ time for each change. Furthermore, we develop fast algorithms for inserting and deleting points at the edge of the dynamic scene.

There are a lot of related problems in computational geometry, as for example the dynamic convex hull and the dynamic nearest neighbor problem, but also applications in motion planning and pattern recognition in dynamic scenes.

1 Introduction

One of the most fundamental data structures in computational geometry is the *Voronoi diagram*. In its most general form, the Voronoi diagram $VD(S)$ of a set S of n objects in a space E is a subdivision of this space into maximal regions, so that all points within a given region have the same nearest neighbor in S with regard to a general distance measure d. In fact the Voronoi diagram contains all of the proximity information defined by the given set in a powerful and computationally useful manner.

When Shamos and Hoey [ShHo 75] introduced the Voronoi diagram for a finite set of points in the Euclidean plane $\mathbb{E}^2$ into computational geometry, they improved a whole lot of worst-case bounds of related problems. Since then Voronoi diagrams in all variations appear as an increasingly interesting object of research (compare for example [Le 82], [Ed 86], [ChEd 87], [DrLe 78], [Fo 86], [Ya 87] and [Ro 89]).

Until now, only static Voronoi diagrams were studied. A first approach for the dynamization of Voronoi diagrams was presented by Gowda et al. [Go 83], who investigated an algorithm for inserting and deleting single points, each in linear time with the help of so-called Voronoi trees.

But when modeling *real dynamic scenes*, the parallel continuous motion of the points – together with a fast update of the Voronoi diagram – is desirable. And it turns out, that already under small continuous motions of the sites – in which case the above approach requires a total recalculation of the entire Voronoi diagram – an update of the Voronoi diagram needs much less time. The *main result* of the present work consists in the dynamization of the underlying objects.

*Lehrstuhl für Informatik I, Universität Würzburg, Am Hubland, D - 8700 Würzburg, Fed. Rep. of Germany. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under contract (No 88/10 - 1).

2 The Topological Structure of Voronoi Diagrams

This section summarizes the elementary definitions concerning classical Euclidean Voronoi diagrams, where closeness is defined by the Euclidean distance function d. Given a finite set $S := \{P_1, \ldots, P_n\}$ of $n \geq 3$ points in the Euclidean plane $\mathbb{E}^2$. First of all let $B(P_i, P_j)$ denote the perpendicular *bisector of P_i and P_j* and $v(P_i) := \{x \in \mathbb{E}^2 \mid \forall_{j \neq i}\ d(x, P_i) \leq d(x, P_j)\}$ the *Voronoi polygon of P_i*. The vertices of the Voronoi polygons are called *Voronoi points* and the bisector parts on the boundary are called *Voronoi edges*. Finally let $VD(S) := \{v(P_i) \mid P_i \in S\}$ denote the *Voronoi diagram of S*. The embedding of the Voronoi diagram provides a planar straight line graph that we call the *geometrical structure* of the underlying Voronoi diagram.

Now we turn our attention to the dual graph of the Voronoi diagram, the so-called *Delaunay triangulation* $DT(S)$. If S is in general position – i.e. no four points of S are cocircular and no three points of S are collinear – every bisector part in $VD(S)$ corresponds to an edge and every Voronoi point in $VD(S)$ to a triple in $DT(S)$. The use of the dual graph not only has numerically advantages, but also allows a clearer separation between geometrical and topological aspects.

We now introduce a *one - point - compactification* to simplify the following examinations and algorithms. Therefore we consider the modified basic set $S' := S \cup \{\infty\}$ and obtain the extended Delaunay triangulation

$$DT(S') = DT(S) \cup \{(P_i, \infty) \mid P_i \in S \cap \partial CH(S)\}$$

i.e. in addition to the Delaunay triangulation $DT(S)$, every point on the boundary of the convex hull $\partial CH(S)$ is connected to ∞. We call the underlying graph of the extended Delaunay triangulation $DT(S')$ the *topological structure* of the Voronoi diagram. We obtain the following characterization for triples in $DT(S')$:

$$\begin{array}{lcl} \{P_i, P_j, P_k\} \in DT(S') & \Longleftrightarrow & v(P_i, P_j, P_k) \text{ is a Voronoi point in } VD(S). \\ \{P_i, P_j, \infty\} \in DT(S') & \Longleftrightarrow & P_i \text{ and } P_j \text{ are neighboring points of } S \text{ on} \\ & & \text{the boundary of the convex hull } \partial CH(S). \end{array}$$

As $DT(S')$ is a complete triangulation of the extended plane $\overline{\mathbb{E}^2}$ – i.e. every triple is bounded by exactly three edges and every edge belongs to exactly two triples – *Euler's polyhedron formula* implies that the number of of edges and triples of the topological structure $DT(S')$ of the Voronoi diagram $VD(S)$ remains linear. Furthermore it is easy to see, that the hardest part of constructing a Voronoi diagram is to determine its topological structure, because the geometrical structure of a Voronoi diagram can be derived from it by a simple flow of the existent Delaunay triples in $DT(S')$ in linear time. In addition, the geometrical structure is determined only locally by its topological structure, namely in the neighborhood of the corresponding Voronoi point. This implies the possibility of a local update of the Voronoi diagram after a local change of one or more points in S.

3 Topological Events

In this section we consider the case of *continuously moving points* and investigate those situations where the topological structure of the Voronoi diagram changes. Therefore, given a finite set $S := \{P_1, \ldots, P_n\}$ of $n \geq 3$ continuous curves in the Euclidean plane $\mathbb{E}^2$, with $P_i : \mathbb{R} \to \mathbb{E}^2$, $t \mapsto P_i(t)$, under the following *assumptions*:

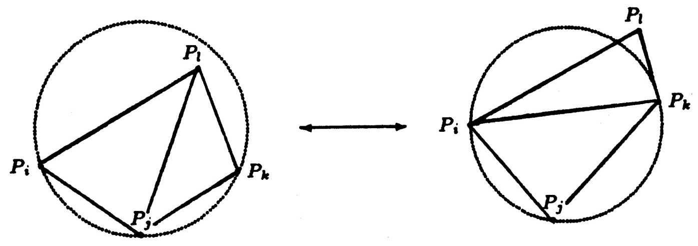
A *The points move without collisions, i.e.* $\forall_{i \neq j}\ \forall_{t \in \mathbb{R}}\ P_i(t) \neq P_j(t)$

B *There exists a moment $t_0 \in \mathbb{R}$ where $S(t_0)$ is in general position.*

Our first basic theorem describes the local stability of the topological structure $DT(S'(t_0))$.

Theorem 1 For a finite set S of points in general position, the topological structure of the Voronoi diagram is *locally stable* under sufficiently small continuous motions of the sites. However, the geometrical structure changes only in the neighborhood of the moving points.

Therefore we have seen, that the loss of general position of the points S is necessary for changing the topological structure $DT(S')$. Now the question for sufficient conditions arises. With this intention, we proceed with an investigation of the *elementary changes* of the topological structure of a Voronoi diagram. We show that they can be characterized as so-called *SWAPs* of adjacent triples in $DT(S')$, except of degenerated cases.



As well, the transition is equivalent to a *fusion* and disappearance of the two original Voronoi points in the moment of cocircularity, while the two dual Voronoi points are generated. To summarize our results, we present our second theorem.

Theorem 2 Elementary changes in the topological structure of the Voronoi diagram $VD(S)$ are characterized by SWAPs of adjacent triples in $DT(S')$, except of degenerated cases.

In this connection the *original advantage* of the one - point - compactification becomes apparent. Both cases can be treated likewise in the extended dual graph $DT(S')$ by simple SWAPs of diagonal edges of adjacent triples. In the following we call a pair of adjacent triples in $DT(S')$ a *quadrilateral.*

Up to now we left one difficulty out of consideration, namely the cases, where more than four points in $S(t)$ are cocircular or more than three points in $S(t)$ are collinear at the same time, or in other words where at least two adjacent quadrilaterals swap at the same time.[1] In this case we can use the linear time algorithm presented in [Ag 87] to retriangulate the interior of the convex polygon described by the cocircular points at a moment $t + \varepsilon$. However, it is necessary to select $\varepsilon > 0$ in such a way, that the moment of retriangulation precedes the next topological event. In the case of neighboring collinear points on the boundary of the convex hull the same effect appears.

4 Dynamic Scenes

In this section we present an algorithm for the update of the topological structure of a Voronoi diagram under continuous motions of the points in S. In the previous section topological events are characterized by moments of cocircularity or collinearity of neighboring points. Therefore we demand, that the zeros of the functions $INCIRCLE(\ldots)$ and $CCW(\ldots)$ introduced by [GuSt 85] are calculable.[2] We can restrict ourselves to the following additional assumption, that is achieved, for example, in the case of piecewise straight curves :

C *The zeros of the function $INCIRCLE(P_i, P_j, P_k, P_l)$ are calculable in constant time.*

[1] Quadrilaterals that are not adjacent may swap in no particular order, because they do not affect each other.

[2] The functions are defined as follows

$$INCIRCLE(P_i,P_j,P_k,P_l) := \begin{vmatrix} x_{P_i} & y_{P_i} & x_{P_i}^2 + y_{P_i}^2 & 1 \\ x_{P_j} & y_{P_j} & x_{P_j}^2 + y_{P_j}^2 & 1 \\ x_{P_k} & y_{P_k} & x_{P_k}^2 + y_{P_k}^2 & 1 \\ x_{P_l} & y_{P_l} & x_{P_l}^2 + y_{P_l}^2 & 1 \end{vmatrix} \quad \text{and} \quad CCW(P_i,P_j,P_k) := \begin{vmatrix} x_{P_i} & y_{P_i} & 1 \\ x_{P_j} & y_{P_j} & 1 \\ x_{P_k} & y_{P_k} & 1 \end{vmatrix}$$

Assumption (C) implies that each quadrilateral generates at most a constant number of topological events. Now we already proceed with a coarse sketch of the algorithm :

Preprocessing :

1. Compute the topological structure $DT(S'(t_0))$ of the starting position.
2. For every existing quadrilateral in $DT(S'(t_0))$ calculate the potential topological events.
3. For the set of the potential topological events build up a balanced SWAP - tree.

Iteration :

1. Determine the next topological event and decide whether it's a SWAP or a RETRIANGULATION.
2. Process the topological event and do an update of the SWAP - tree.

Next we look closer to the individual steps of the algorithm and their runtime- and storage requirements. Using the divide & conquer algorithm for the Delaunay triangulation presented in [GuSt 85][3] and a balanced SWAP - tree, where the topological events are stored according to their temporal appearance, the entire preprocessing step requires optimal $O(n \log n)$ time and $O(n)$ space.

To determine the next topological event in the first iteration step, we use a simple minimum query in the tree which requires $O(\log n)$ time. Furthermore if we assume, that the number of cocircular and collinear points in the degenerated cases remains constant (anything else is completely unlikely), then the decision can be done in constant time. Before we analyze the second iteration step, we firstly notice that each SWAP destroys only four quadrilaterals while other four quadrilaterals are generated. Therefore all we have to do is to delete the four destroyed quadrilaterals and their corresponding topological events in the SWAP - tree and to insert the four new ones. In the case of a degenerated situation there is also only a constant number of destroyed and newly generated quadrilaterals. Altogether the total amount of time used by the second iteration step is $O(\log n)$. We can prove, that this result is already optimal under linear motions of the points, by presenting a worst case example where the topological events requires $\Theta(\log n)$ time each. Additionally we have seen, that only the really necessary topological events are performed. On the other hand there are at most $\binom{n+1}{4} \in O(n^4)$ quadrilaterals, each of which generates at most a constant number of topological events.

Theorem 3 Given a finite set $S(t)$ of n continuous curves under the assumptions (A), (B) and (C). The motion of the points requires optimal $O(n \log n)$ preprocessing time and $O(n)$ storage. Every topological event that appears can be treated in optimal $O(\log n)$ time. Furthermore there are at most $O(n^4)$ topological events during the entire flow of the points. During the linear motion of one point $P_i \in S$ there are at most $O(n)$ topological events.

We extend this algorithm with a simple reversible algorithm for inserting and deleting points at the edge of the dynamic scene – or in other words – outside a relevant window. It is the idea to insert the point sufficiently distant from the convex hull $CH(S)$, so that the Delaunay triangulation $DT(S)$ remains stable and only some extended dual edges must be changed. With that, inserting and deleting points at the edge of the dynamic scene requires optimal $O(k)$ time where $k := |S \cap \partial CH(S)|$ is the number of points on the boundary of the convex hull of S.

Naturally we can use other existing algorithms for inserting points at arbitrary places (compare [GuSt 85]) which uses $O(n)$ time. However, only complicated linear-time algorithms for deleting points are known so far (see [Ag 87]). Therefore it might be better to remove a point at an arbitrary place in two steps by using the previous algorithms. At first, we move this point at the edge of the scene where afterwards it can be removed easily.

[3] This algorithm is exceptionally suitable for this task, because the used quad-edge data structure makes possible a fast derivation of the geometrical structure of the Voronoi diagram and the accompanying convex hull.

References

[Ag 87] A. Aggarwal, L. Guibas, J. Saxe and P. Shor, *A Linear Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon*, Proc. of the 19th Annual ACM Symposium on Theory of Computing, New York City, 1987, pp 39 - 45

[ChEd 87] B. Chazelle and H. Edelsbrunner, *An Improved Algorithm for Constructing k th - Order Voronoi Diagrams*, IEEE Transactions on Computers, Nov. 1987, Vol. C-36, No. 11, pp 1349 - 1354

[DrLe 78] R.L. Drysdale III and D.T. Lee, *Generalized Voronoi Diagrams in the Plane*, Proc. 16 th Annual Allerton Conference on Communications, Control and Computing, Oct. 1978, pp 833 – 842

[Ed 86] H. Edelsbrunner, *Edge-Skeletons in Arrangements with Applications*, Algorithmica 1986, Vol. 1, pp 93 - 109

[Ed 87] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, EATCS Monographs in Computer Science, Springer - Verlag, Berlin - Heidelberg, 1987

[Fo 86] S. Fortune, *A Sweepline Algorithm for Voronoi Diagrams*, Proc. 2 nd Annual ACM Symp. Computational Geometry, Yorktown Heights, 1986, pp 313 – 322

[Go 83] I.G. Gowda, D.G. Kirkpatrick, D.T. Lee and A. Naamad, *Dynamic Voronoi Diagrams*, IEEE Trans. on Information Theory, Vol. IT-29, No. 5, Sept. 1983, pp 724 - 731

[GuSt 85] L. Guibas and J. Stolfi, *Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams*, ACM Transactions on Graphics, Vol. 4, No. 2, April 1984, pp 74 - 123

[HaDe 59] H. Hadwiger und H. Debrunner, *Kombinatorische Geometrie in der Ebene*, Monographies de L'Enseignement Mathématique, No. 2, Université Genève, 1960

[Le 82] D.T. Lee, *On k-Nearest Neighbor Voronoi Diagrams in the Plane*, IEEE Transactions on Computers, Vol. C-31, No. 6, June 1982, pp 478 - 487

[No 88] H. Noltemeier, *Computational Geometry and its Applications*, Proceedings Workshop CG '88, Universität Würzburg, März 1988, LNCS 333, Springer Verlag, 1988

[PrSh 85] F.P. Preparata and M.I. Shamos, *Computational Geometry – An Introduction*, Springer - Verlag, New York, 1985

[Ro 88] T. Roos, *Voronoi Diagramme*, Diplomarbeit, Universität Würzburg, 1988

[Ro 89] T. Roos, *k - Nearest - Neighbor Voronoi Diagrams for Sets of Convex Polygons, Line Segments and Points*, Proceedings 15th Intern. Workshop on Graph-Theoretic Concepts in Computer Science WG89, LNCS 411, Springer - Verlag, Berlin - Heidelberg - New York

[ShHo 75] M.I. Shamos and D. Hoey, *Closest - Point Problems*, Proc. 16 th Annual Symp. on FOCS, 1975, pp 151 – 162

[Ya 87] C.K. Yap, *An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments*, Discrete & Computational Geometry, 1987, Vol. 2, pp 365 – 393