A New Approach to Output-Sensitive Voronoi Diagrams and Delaunay Triangulations

Gary L. Miller and Donald R. Sheehy

Abstract

We present an algorithm for computing Voronoi diagrams and Delaunay triangulations of point sets in $\mathbb{R}^d$. We also give an output-sensitive analysis, proving that the running time is at most $O(m \log n \log \Delta)$, where n is the input size, m is the output size, and the spread Δ is the ratio of the diameter to the closest pair distance. For many realistic settings, the spread is polynomial in n, in which case we have the only known algorithm that is within a poly-logarithmic factor of optimal for the entire range of output sizes and any fixed dimension.

1 Introduction

Delaunay refinement starts with the Delaunay triangulation of a set of points P and then proceeds to add extra points called *Steiner points* to improve the *quality* of the Delaunay simplices (see for example [4]). Here quality could take different definitions depending on the application, and we call the output a *quality mesh* on a *well-spaced* superset of P. This simple procedure has a worst-case running time that is at least the cost of building Del_P, the Delaunay triangulation of the input points, since that is the first step. In 2006, the Sparse Voronoi Refinement (SVR) algorithm of Hudson, Miller, and Phillips reordered the priorities of the standard algorithm and proved a nearly optimal bound of the running time for any fixed dimension d [5]. In particular, they showed that for many inputs, one can compute the Delaunay triangulation of a well-spaced superset of the input in less time than it would take to compute the Delaunay triangulation of the input alone. This is only possible because of the large gap between the best-case and worst-case complexity of Delaunay triangulations as a function of $n := |P|$ [8]. The quality condition guarantees that the refined output lies close to the best-case, i.e. all vertices touch only a constant number of Delaunay simplices.

In this paper, we turn this story around and explore the reverse question: If computing a Delaunay triangulation of P is no longer a prerequisite for computing a quality mesh, then might it be possible to use the quality mesh to efficiently compute the Delaunay triangulation of P? Indeed, we give a simple algorithm that removes all of the Steiner points by a simple local flipping routine, leaving behind Del_P. We show how to characterize the flips in terms of the intersection of two Voronoi diagrams. Then, we bound the total number of combinatorial changes throughout the algorithm by bounding these intersections, which can be done using standard tools from the mesh generation literature.

2 Related Work

Previous output sensitive methods for Voronoi diagrams in higher dimensions were based on computing convex hulls. The shelling approach of Seidel achieves $O(n^2 + m \log n)$ running time [7]. This was improved slightly to lessen the quadratic preprocessing by Matousek and Schwartzkopf [6]. Another approach is a "gift-wrapping" algorithm due to Chan [2]. Later improvements by Chan et al. give an $O(m \log^2 n)$ for Voronoi diagrams in $\mathbb{R}^3$ [3]. Similarly, an $O(m \log^3 n)$ algorithm was given for Voronoi diagrams in $\mathbb{R}^4$ by Amato and Ramos[1].

3 The Algorithm

There are three phases to the algorithm. In the first phase, a quality mesh is constructed using SVR within a bounding box containing the points. This mesh has $O(n \log \Delta)$ vertices and $O(n \log \Delta)$ simplices. The second phase uses local flips to remove the Steiner points. These flips are ordered to maintain a weighted Delaunay triangulation with the weights of all input points increasing uniformly from 0 to ∞. The potential flips are stored in a heap and ordered according

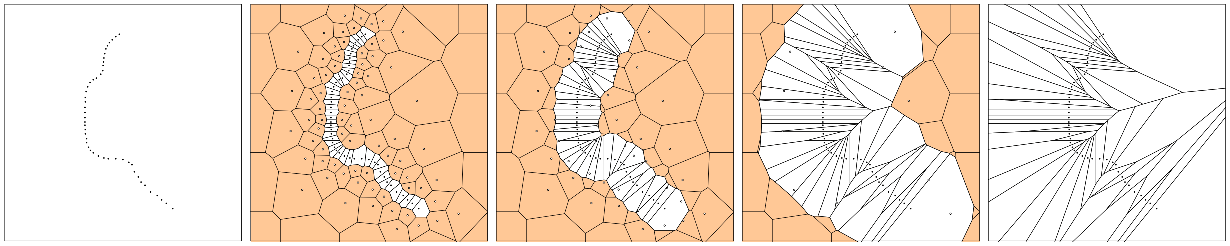

Figure 1: An illustration of the algorithm from left to right. Starting with a point set, it is extended to a quality mesh. Then the weights of the input points are increased until the extra cells disappear.

to the weights of the input points when the flip will occur. In the third and final phase, the boundary vertices are removed. These vertices must be handled separately because they appear in the weighted Delaunay triangulation for all possible weight assignments to the points.

4 Analysis

When do flips happen? The key to the analysis is to bound the number of flips that occur in the transformation from Vor_M to Vor_P. We do this by observing that a flip happens at time t exactly if there is a degenerate $(d+2)$-tuple of points under the standard lifting

$$p \mapsto (p, \|p\|^2 - w_p^2),$$

where the weight w_p is t for input points and 0 for Steiner points. That is, all $d+2$ lifted points lie on a common hyperplane. This hyperplane in $\mathbb{R}^{d+1}$ has a dual point using the duality:

$$(x_1, \ldots, x_{d+1}) \Leftrightarrow y_d = 2x_1y_1 + \cdots + 2x_dy_d - x_d.$$

For such collections of points, the dual point projects to the *orthocenter* of the weighted points, the center of a sphere that intersects tangentially each the spheres with center p and radius w_p. By partitioning these points into two sets I and S depending on whether they are input points or Steiner points, one can easily compute that the distance from the orthocenter to each point of I is the same. Similarly, the distance from the orthocenter to each point of S is the same. Thus, the orthocenter is the intersection of a face of Vor_P and a face of Vor_M. Since all flips are characterized this way, we can bound the number of flips by bounding the number of intersections between these two Voronoi diagrams.

Counting flips. In many problems, it is quite challenging to bound the number of flips, but several factors make it possible for our algorithm. First, the radius of the Voronoi cell of a Steiner point in a quality mesh is proportional to its distance to the nearest input point. Second, the Voronoi cells of Steiner points in Vor_M have only a constant number of faces. So, by an easy packing argument, we get that the number of face-face intersections between Vor_P and Vor_M is at most $O(m \log \Delta)$. That is, each $(d-k)$-face of Vor_P can only intersect $O(\log \Delta)$ k-faces of Vor_M. So the total number of flips is $O(m \log \Delta)$ and the total running time is $O(m \log n \log \Delta)$, where the extra log term comes from the heap operations needed to order the flips.

References

[1] Nancy M. Amato and Edgar A. Ramos. On computing voronoi diagrams by divide-prune-and-conquer. In *Symposium on Computational Geometry*, pages 166–175, 1996.

[2] Timothy M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete & Computational Geometry*, 16(4):369–387, 1996.

[3] Timothy M. Chan, Jack Snoeyink, and Chee-Keng Yap. Primal dividing and dual pruning: Output-sensitive construction of four-dimensional polytopes and three-dimensional voronoi diagrams. *Discrete & Computational Geometry*, 18(4):433–454, 1997.

[4] Siu-Wing Cheng, Tamal K. Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.

[5] Benoît Hudson, Gary Miller, and Todd Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.

[6] Jiří Matoušek and Otfried Schwarzkopf. Linear optimization queries. In *Symposium on Computational Geometry*, pages 16–25, 1992.

[7] Raimund Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *STOC: ACM Symposium on Theory of Computing*, 1986.

[8] Raimund Seidel. On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 181–185, 1987.