Discrete Comput Geom (2014) 52:476–491
DOI 10.1007/s00454-014-9629-y

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

Gary L. Miller · Donald R. Sheehy

Received: 10 September 2013 / Revised: 27 March 2014 / Accepted: 8 July 2014 /
Published online: 3 September 2014

Abstract We describe a new algorithm for computing the Voronoi diagram of a set of n points in constant-dimensional Euclidean space. The running time of our algorithm is $O(f \log n \log \Delta)$ where f is the output complexity of the Voronoi diagram and Δ is the spread of the input, the ratio of largest to smallest pairwise distances. Despite the simplicity of the algorithm and its analysis, it improves on the state of the art for all inputs with polynomial spread and near-linear output size. The key idea is to first build the Voronoi diagram of a superset of the input points using ideas from Voronoi refinement mesh generation. Then, the extra points are removed in a straightforward way that allows the total work to be bounded in terms of the output complexity, yielding the output sensitive bound. The removal only involves local flips and is inspired by kinetic data structures.

Keywords Voronoi diagram · Delaunay triangulation · Output-sensitive algorithms · Mesh generation · Kinetic data structures

1 Introduction

Voronoi diagrams and their duals, Delaunay triangulations, are ubiquitous in computational geometry, both as a source of interesting theory and a tool for applications [3,4].

G. L. Miller
Computer Science department, Carnegie Mellon University, 5000 Forbes Avenue,
Pittsburgh, PA 15213, USA
e-mail: glmiller@cs.cmu.edu

D. R. Sheehy
Computer Science & Engineering Department, University of Connecticut,
371 Fairfield Way, Unit 4155, Storrs, CT 06269-4155, USA
e-mail: don.r.sheehy@gmail.com

Many algorithms exist for planar Voronoi diagrams and optimal algorithms are known. In higher dimensions, the situation is complicated by the large gap between the best-case and worst-case output complexity. A Voronoi diagram of n points in $\mathbb{R}^d$ can have between $\Theta(n)$ and $\Theta(n^{\lceil d/2 \rceil})$ faces [22,29] (we suppress constant factors that only depend on d). This motivates the search for *output-sensitive* algorithms and analysis, where the time and space guarantees depend on both the input size n and the number of output faces f.

Computing Voronoi diagrams in $\mathbb{R}^d$ reduces to computing convex hulls in $\mathbb{R}^{d+1}$. This relationship is perhaps most clear in the dual, where the Delaunay triangulation is the projection of the lower hull of the input points lifted into $\mathbb{R}^{d+1}$ by the standard *parabolic lifting*:

$$(x_1, \ldots, x_d) \mapsto \left(x_1, \ldots, x_d, \sum_{i=1}^{d} x_i^2\right).$$

All of the previous literature on output-sensitive Voronoi diagrams in higher dimensions is directed at solving the more general problem of computing convex hulls.[1]

Seidel [28] gave an algorithm based on polytope shelling that runs in $O(n^2 + f \log n)$ time. The quadratic term is from the first phase of the algorithm that solves a d-dimensional linear program for each input point. Matousek and Schwartzkopf [24] showed how to exploit the common structure in these linear programs to improve the running time to $O(n^{2-2/(\lceil d/2 \rceil + 1)} \log^{O(1)} n + f \log n)$.

Another type of algorithm uses the dual notions of gift-wrapping [9,32] and pivoting [5]. Both approaches can enumerate the facets of a simple polytope in $O(nf)$ time, thus paying approximately linear time per face. Since Voronoi diagrams have at least n faces, these methods are not asymptotically faster than the LP-based methods except that they avoid the exponential dependence on the dimension inherent in such methods.

Chan [7] gave an algorithm based on gift-wrapping that runs in $O(n \log f + (nf)^{1-1/(\lceil d/2 \rceil + 1)} \log^{O(1)} n)$ time. A later work by Chan et al. [8] gave an algorithm that runs in $O((n + (nf)^{1-1/\lceil (d+1)/2 \rceil} + fn^{1-2/\lceil (d+1)/2 \rceil}) \log^{O(1)} n)$ time [8]. Even when $f = \Theta(n)$, this algorithm requires polynomial time per face.

Better bounds are known for 3- and 4-dimensional Voronoi diagrams where truly polylogarithmic time per face algorithms are known. Chan et al. [8] gave an algorithm that achieves $O(f \log^2 n)$ for $\mathbb{R}^3$. Amato and Ramos [2] gave an $O(f \log^3 n)$-time algorithm in $\mathbb{R}^4$.

Voronoi diagrams and Delaunay triangulations are used in mesh generation (see the recent book by Cheng et al. [12]). Extra vertices called Steiner points are added in a way that keeps the complexity down. Perhaps surprisingly, the number of vertices increases, but the total number of faces can decrease. Such a mesh can be constructed in $O(n \log \Delta)$ time using only $O(n \log \Delta)$ vertices [18], where Δ is the ratio of largest to smallest distances among pairs of input points. This was later improved to $O(n \log n)$

[1] To avoid confusion, when reporting running times of known algorithms, we always give the results as they apply to Voronoi diagrams rather than convex hulls.

time by a more complicated algorithm [25] and a relaxed definition of mesh quality, but we do not see how to use this fact for our application.

In this paper, we propose a new algorithm for constructing Voronoi diagrams that uses Voronoi refinement mesh generation as a preprocessing step. Then, it removes all of the Steiner points using a method derived from the field of kinetic data structures. We prove that at most $O(f \log \Delta)$ local changes occur during the removal process. Each local change requires only constant time to process. These local changes are ordered via a heap data structure which adds an extra factor of $O(\log(f \log \Delta)) = O(\log n + \log\log \Delta)$ to the running time. We assume that $\Delta < 2^{(n^{d+1})}$, for otherwise, the desired running time can be achieved by brute force. Under this assumption, $\log\log \Delta = O(\log n)$. Thus, the total running time is $O(f \log n \log \Delta)$.

Unlike previous work on output-sensitive Voronoi diagram construction, our algorithm does not use a reduction to the convex hull problem. Instead, it uses specific properties of the Voronoi diagram to get an improvement.

1.1 Contribution

We present the MeshVoronoi algorithm, a new, output-sensitive algorithm for computing Voronoi diagrams in $\mathbb{R}^d$. MeshVoronoi runs in $O(f \log n \log \Delta)$ time. When Δ is bounded by a polynomial function of n, such as is the case when input points have integer coordinates with $O(\log n)$ bits of precision, the running time is $O(f \log^2 n)$. Even the $O(n^{\lceil d/2 \rceil})$ worst-case inputs for Voronoi diagrams can be represented with polynomial spread [15]. We get an improvement over existing algorithms for inputs with polynomial spread in dimension $d > 3$ when $f = O(n^{2-2/\lceil d/2 \rceil})$.

The other advantage of the MeshVoronoi algorithm is that it is simple both to describe and to analyze.

1.2 Related Work

We will make use of the flip-based construction of weighted Delaunay triangulations similar to that presented by Edelsbrunner and Shah [14]. In that paper, the concern was to add a single point to a regular triangulation but when run backwards, it describes the removal of a single point by local flips. We are interested in removing all of the Steiner points of the mesh simultaneously. In this respect, the problem more resembles the Delaunay triangulation splitting problem for which Chazelle et al. gave a linear time algorithm for the plane [10]. Chazelle and Mulzer [11] gave an analogous result for splitting convex polytopes in $\mathbb{R}^3$.

Our algorithm may be viewed as a special case of a kinetic convex hull problem, and indeed, the main tools come directly from the literature on kinetic data structures [16]. In general, the kinetic convex hull problem is much harder than the instances arising in our algorithm and it is only because of the specific geometric structure of these instances that we are able to prove useful bounds.

Joswig and Ziegler [21] presented a different approach to output-sensitive convex hulls algorithms using homology computation. This algorithm is shown to be output-sensitive for simplicial polytopes like those produced for Delaunay triangulations of

points in general position. However, they do not improve bounds on the asymptotic running times in terms of n and f as their construction passes through several reductions.

Many low-dimensional algorithms for computing Voronoi diagrams depend on incremental construction, where the points are added one at a time. When the input can be degenerate, Bremner showed that incremental constructions cannot be output-sensitive [6]. Our algorithm does resemble an incremental algorithm. The main difference is that it adds *more* than just the input points in the first phase of the algorithm. The harder work is removing these extra points.

2 Background

2.1 Points and Distances in Euclidean Space

We will deal exclusively with the case of points in d-dimensional Euclidean space. The Euclidean norm of x is denoted $\|x\|$ and thus the Euclidean distance between two points x and y is $\|x - y\|$. For a point $x \in \mathbb{R}^d$ and a compact set $U \subset \mathbb{R}^d$, define $\mathrm{d}(x, U) := \min_{y \in U} \|x - y\|$. The *diameter* of a compact set is maximum distance among all pairs of points in the set. The *spread* Δ of a finite set of points is the ratio of the diameter to the smallest pairwise distance.

As noted above, we make the simplifying assumption that $\Delta < 2^{(n^{d+1})}$. This is similar to assuming an asymptotic floating point notation where coordinates are stored as floating point numbers with $O(\log n)$-bit words (see [17] for a more detailed treatment of this model). If $\Delta > 2^{(n^{d+2})}$, then $n \log \Delta > n^{d+3}$, which is enough time to test each $(d+1)$-simplex to see if it is Delaunay by brute force.

Alternatively, it is possible to simulate a $2^{O(n)}$ upper bound on the spread by decomposing the point set into a hierarchy of $O(n)$ point sets, each with such a bound on the spread (see Miller et al. [25,26] for an application of this approach to mesh generation). In this light, it is possible to interpret every occurrence of $\log \Delta$ as indicating $\min\{\log \Delta, n\}$. The assumed upper bound on the spread allows a clearer exposition of the main ideas of our approach without introducing the complexity of hierarchical point sets.

2.2 Voronoi Diagrams and Power Diagrams

Let $P \subset \mathbb{R}^d$ be a finite set of points. The *Voronoi diagram* of P is a cell complex in $\mathbb{R}^d$ of convex, polyhedral cells such that all points in a cell share a common set of nearest neighbors among the points of P. Formally, for a subset $S \subset P$, the Voronoi cell $\mathrm{Vor}_P(S)$ is defined as the set of all points x of $\mathbb{R}^d$ such that $\mathrm{d}(x, P) = \|x - y\|$ for all $y \in S$, i.e.

$$\mathrm{Vor}_P(S) := \{x \in \mathbb{R}^d : \mathrm{d}(x, P) = \|x - y\| \text{ for all } y \in S\}.$$

The relative interiors of these closed cells decompose $\mathbb{R}^d$. When $S = \{v\}$ is a singleton, we will abuse notation and refer to its Voronoi cell as $\mathrm{Vor}_P(v)$ instead of $\mathrm{Vor}_P(\{v\})$. For points in *general position* (no $k+3$ points lying on a common k-sphere), the affine

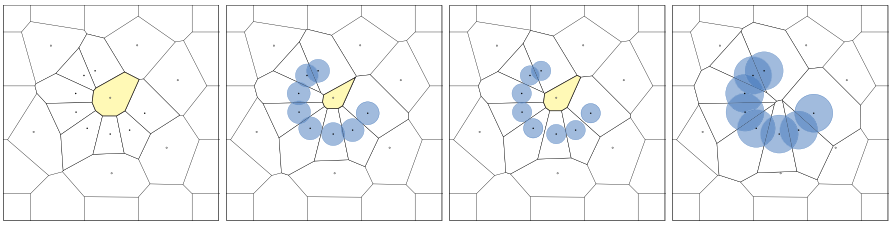

Fig. 1 A weighted Voronoi diagram with weights indicated by disks. As the weights of some points increase from *left* to *right*, some cells disappear from the diagram altogether

dimension of $\mathrm{Vor}_P(S)$ is $d - (|S| - 1)$. We write Vor_P to denote the set of nonempty cells $\mathrm{Vor}_P(S)$ for $S \subseteq P$. See the book by Edelsbrunner for a nice treatment of Voronoi diagrams [13].

The Voronoi diagram of P, denoted Vor_P, has a natural dual called the *Delaunay diagram*, denoted Del_P. For point sets in general position, the Delaunay diagram is an embedded simplicial complex called the *Delaunay triangulation*. The Voronoi/Delaunay duality is realized combinatorially by inverting the posets of the corresponding cell complexes, identifying each k-face of the Voronoi diagram with a $(d-k)$-simplex of the Delaunay triangulation. Among other things, this implies that $|\mathrm{Vor}_P| = |\mathrm{Del}_P|$.

If the points P have real-valued weights $w : P \to \mathbb{R}$ and the Euclidean distance from $p \in P$ to $x \in \mathbb{R}^d$ is replaced with the power distance $\pi_p(x) = \|x-p\|^2 - w(p)^2$, then the Voronoi diagram becomes a *weighted Voronoi diagram*, also known as a *power diagram* [13]. The dual is still well-defined and is known as the *weighted Delaunay diagram* (or the *weighted Delaunay triangulation* when it is a simplicial complex). Figure 1 shows a sequence of weighted Voronoi diagrams for a set of points with different weights.

The convex hull of a set of points S is the boundary of the convex closure of the points. If S is finite, then the convex hull of S is a cell complex of polyhedral faces. If S is in general position, then it is a simplicial complex. We write $\mathrm{ch}(S)$ to denote the set of faces of the convex hull of S, so it makes sense, for example, to write $\mathrm{ch}(S) \subset \mathrm{Del}_S$.

Weighted Delaunay diagrams are the projections of the lower faces of convex polytopes in one dimension higher. In fact, interpreting the power distance to the origin as a height function for the points lifted into $\mathbb{R}^{d+1}$, gives the weighted Delaunay diagram as the projection of the lower convex hull of these lifted points. In particular, setting all weights to 0 gives the (unweighted) Delaunay diagram as a convex hull in $\mathbb{R}^{d+1}$. Thus, there is a strong connection between the problems of computing convex hulls, Delaunay triangulations, and Voronoi diagrams.

2.3 Orthoballs and Encroachment

A weighted point p *encroaches* $B = \mathrm{ball}(c, r)$ if

$$\pi_p(c) < r^2,$$

and it is *orthogonal* to B if

$$\pi_p(c) = r^2.$$

For a collection of $d+1$ weighted points in $\mathbb{R}^d$ (not all on a hyperplane), the *orthoball* is the unique ball orthogonal to each of the weighted points, and its center and radius are called the *orthocenter* and *orthoradius* respectively. For unweighted points, a point encroaches a ball if its in the interior, and it is orthogonal if it is on the boundary. The orthocenter of unweighted points is called the *circumcenter* and its center and radius are called the *circumcenter* and *circumradius* respectively. Again, the book by Edelsbrunner gives a nice treatment of orthoballs and encroachment [13].

If a weighted point encroaches the orthoball of a simplex σ then we say that σ is *encroached*. For points in general position, the weighted Delaunay triangulation is the unique triangulation in which no simplex is encroached by any weighted vertex. In the unweighted case, this corresponds to the property that no vertex lies in the interior of the circumball of any Delaunay simplex.

2.4 Flips in Triangulations

Bistellar flips are a useful way to make local changes in triangulations [23]. This operation takes a collection of $d+2$ vertices S for which a triangulation of the convex hull $\mathrm{ch}(S)$ is a subcomplex of the larger triangulation, and replace its interior with a new triangulation. In $\mathbb{R}^2$, there are three classes of flips: those that swap the diagonal of a quadrilateral, those that introduce a new vertex in a triangle by splitting it into three, and those that remove a vertex incident to exactly three triangles. These are called respectively $(2, 2)$-, $(1, 3)$-, and $(3, 1)$-flips, where the numbers correspond to the number of d-simplices removed and inserted respectively. In d dimensions, there are similar (i, j)-flips for nonnegative integers i, j such that $i + j = d + 2$, where each replaces i old d-simplices with j new d-simplices.

There is also a geometric interpretation of a flip in which the two different triangulations of the $d+2$ points can be seen as the projections of the respectively upper and lower faces of the convex hull of the points after lifting them into one dimension higher.

Given a point set P and any two weight functions w_1 and w_2 on P, the weighted Delaunay triangulation for w_1 can be transformed into the weighted Delaunay triangulation for w_2 by a sequence of bistellar flips. This process of local changes is at the heart of incremental constructions [14]. Any continuous change in the weights of a set of points causes a change in the corresponding weighted Delaunay triangulation that can be realized by a sequence of bistellar flips (see Fig. 2). In such a case, the flips happen exactly at those moments when the diagram is no longer a triangulation. For unweighted points, this happens when some set of $d + 2$ points lie on a common $(d - 1)$-sphere. This can be tested by a linear predicate. That is, $d + 2$ points $p_1, \ldots, p_{d+2}$ lie on a $(d - 1)$-sphere if and only if

$$\det \begin{bmatrix} p_1 & \cdots & p_{d+2} \\ \|p_1\|^2 & \cdots & \|p_{d+2}\|^2 \\ 1 & \cdots & 1 \end{bmatrix} = 0.$$

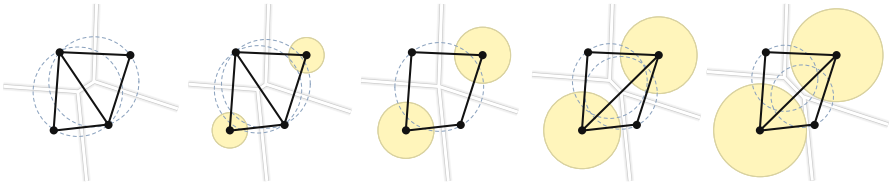

Fig. 2 From *left* to *right*, increasing the weight of two points causes a flip in the weighted Delaunay triangulation. The *dotted circles* indicate the orthoballs of the weighted Delaunay triangles. The *center figure* indicates the exact weight when the flip occurs as the orthoballs of all four possible triangles coincide

These represent the degenerate configurations of points. As before, for weighted points, we replace the norm with the power distance to find that $d+2$ weighted points form a degenerate configuration if and only if

$$\det \begin{bmatrix} p_1 & \cdots & p_{d+2} \\ \|p_1\|^2 - w(p_1)^2 & \cdots & \|p_{d+2}\|^2 - w(p_{d+2})^2 \\ 1 & \cdots & 1 \end{bmatrix} = 0.$$

This determinant test is a consequence of the lifting definition of the weighted Delaunay triangulation.

2.5 Voronoi Aspect Ratios and Voronoi Refinement

The *in-radius* of a Voronoi cell $\mathrm{Vor}_P(q)$ is the radius of the largest ball centered at q contained in $\mathrm{Vor}_P(q)$. The *out-radius* of $\mathrm{Vor}_P(q)$ is the radius of the smallest ball centered at q that contains all of the vertices of $\mathrm{Vor}_P(q)$. Such a ball contains all of $\mathrm{Vor}_P(q)$ for bounded Voronoi cells. The *aspect ratio* of the Voronoi cell of q is the ratio the out-radius over the in-radius, denoted $\mathrm{aspect}_P(q)$. We say a set of points M is τ-*well-spaced* if for all $v \in M$, $\mathrm{aspect}_M(v) \leq \tau$.

Voronoi diagrams of well-spaced points have many nice properties. The most relevant for our purposes is that no d-dimensional Voronoi cell has more than $2^{O(d)}$ facets (faces of codimension 1) [27, Theorem 4.2].

For any set of n points P, there exists a τ-well-spaced superset M for any constant $\tau > 2$. Moreover, such a superset with

$$|M| = O(n \log \Delta) \tag{1}$$

can be computed in $O(n \log n + |M|)$ time [25]. The process of adding points to improve the Voronoi aspect ratio is called *Voronoi refinement*. It is perhaps more widely known in its dual form, Delaunay refinement. The extra points added are called *Steiner points*.

The analysis of Voronoi refinement depends on a function called the *Ruppert feature size*, defined for all $x \in \mathbb{R}^d$ as the distance to the second nearest input point.

$$\mathbf{f}_P(x) := \max_{p \in P} \mathrm{d}(x, P \setminus \{p\}).$$

One defines the feature size with respect to a set M similarly and denotes it $\mathbf{f}_M$. Voronoi refinement produces a τ-well-spaced set of points M such that for each vertex $v \in M$,

$$\mathbf{f}_P(v) \leq K\mathbf{f}_M(v),$$

where $K = \frac{2\tau}{\tau-2}$ (see [18] or [30, Theorem 3.3.2]). This bound relates the feature size of the points in M to the feature size induced by the input points P.

Let Ω be a convex bounding region containing P; generally a box or a ball is used. Throughout, we assume that the diameter of Ω is only a constant factor larger than the diameter of P. The total number of vertices in the output M is determined up to constant factors by the *feature size measure* of Ω, defined as

$$\mu(\Omega) := \int_{\Omega} \frac{\mathrm{d}x}{\mathbf{f}_P(x)}.$$

For a wide class of inputs, the feature size measure is $O(n)$ [31]. For general inputs, the bound of $O(n \log \Delta)$ in (1) above is well-known and can also be derived as a corollary to Lemma 4 below.

2.6 Sparse Voronoi Refinement

The meshing algorithm that we use is called Sparse Voronoi Refinement (SVR) and is due to Hudson et al. [18]. SVR is able to avoid the worst case complexity of Voronoi diagrams because it guarantees that the intermediate state is always a well-spaced point set and thus has a Voronoi diagram of linear complexity.

For the case of point sets in a bounding box, the SVR algorithm is easy to describe. It is an incremental construction that proceeds by alternating between two phases called *break* and *clean*. In the break phase, the algorithm finds a Voronoi cell $\mathrm{Vor}(r)$ that contains an input point that has not yet been inserted. It then attempts to add a Steiner point q at the vertex of $\mathrm{Vor}(r)$ that is farthest from r. If there is an (uninserted) input point p more than a (user-defined) constant times closer to q than to r, then p is added instead. This is called *yielding* to p. The clean phase repeatedly finds any cell $\mathrm{Vor}(r)$ with $\mathrm{aspect}(r) > \tau$ and adds attempts to add a Steiner point q at the vertex of $\mathrm{Vor}(r)$ that is farthest from r. As with the break phase, we yield to nearby uninsterted inputs when attempting to insert Steiner points. The clean phase continues until all cells have aspect ratio at most τ.

The running time of SVR is $O(n \log \Delta + |M|)$. The $n \log \Delta$ term comes from the point location data structure which associates each point with the Voronoi cell that contains it. When attempting to add a point, the nearby Voronoi cells are checked to see if there are nearby input points to yield to.

Acar et al. [1] developed an efficient implementation of SVR in 3-d. It can also be efficiently parallelized [19]. Recently, Miller et al. [25] showed that a variation of SVR runs in $O(n \log n + |M|)$ time by using a more complex point location scheme.

3 Algorithm

In this section, we describe the MESHVORONOI algorithm. It has three phases: a preprocessing phase where the input points are placed in a bounding box and meshed, a removal phase that eliminates most of the Steiner points by incremental flipping, and a cleanup phase that removes the last remaining Steiner points on the boundary. The main data structure is a heap that stores the facets that are scheduled for removal by flipping. We call this the *flip heap*.

Throughout the algorithm, there is a global time parameter t. There will be a superset M of the input points P. At time t, M_t denotes the set M with weights, where the weight of a point p at time t is given as

$$w(p,t) := \begin{cases} \sqrt{t} & \text{if } p \in \mathrm{P}, \\ 0 & \text{otherwise.} \end{cases}$$

Using this weighting scheme, there exists a sufficiently large t such that for all $t' > t$, $\mathrm{Del}_{M_t} = \mathrm{Del}_{M_{t'}}$. We use $M_\star$ to refer to M_t, where t is some such sufficiently large value.

3.1 Checking Potential Flips

A set S of $d+2$ points forms a *potential flip* at time t if $\mathrm{ch}(S) \subset \mathrm{Del}_{M_t}$. A potential flip can be identified with any of its interior facets. For example, in the plane, $(2,2)$-flips are usually associated with the edge in the interior of a convex quadrilateral that will get replaced during the flip. In general, the facet representing a potential flip stands for the $d+2$ points comprising the two simplices sharing that facet.

The weights of the points of M vary with the time parameter t. For $p \in M$, the power distance from $x \in \mathbb{R}^d$ to p at time t is given as

$$\pi_p(x,t) := \|p-x\|^2 - w(p,t)^2 = \begin{cases} \|p-x\|^2 - t & \text{if } p \in \mathrm{P}, \\ \|p-x\|^2 & \text{otherwise.} \end{cases}$$

So, for $d+2$ points $p_1, \ldots, p_{d+2}$ comprising a potential flip, the time associated with the flip is the value of t such that

$$\det \begin{bmatrix} p_1 & \cdots & p_{d+2} \\ \pi_{p_1}(0,t) & \cdots & \pi_{p_{d+2}}(0,t) \\ 1 & \cdots & 1 \end{bmatrix} = 0.$$

The determinant on the left hand side is a linear function of t, so it is easy to solve for t. When evaluating a potential flip, the algorithm simply checks if the time associated with the flip is before or after the current time. This approach to viewing linear, geometric predicates as polynomial functions of time is well established in the area of kinetic data structures [16]. In our case, only one row of the matrix is changing with time, and the change is linear, so there is no need to solve higher degree polynomial systems as is common in other kinetic data structures problems. Moreover, there will only be a flip if there are both Steiner points and input points.

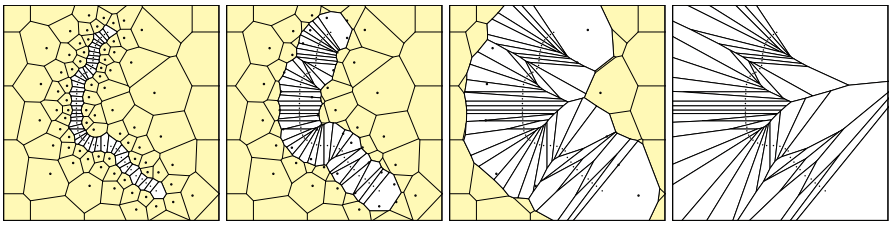

Fig. 3 An illustration of the algorithm from *left* to *right*. Starting with a point set, it is extended to a well-spaced superset. The cells of the Steiner points are *shaded*. Then, the weights of the input points are increased until the extra cells disappear

3.2 Preprocessing

The first step in the algorithm is to add a constant number of points to form a bounding region around the input. This can be done so that the total complexity of the convex hull of the augmented point set is constant. Second, the Sparse Voronoi Refinement algorithm adds $O(n \log \Delta)$ Steiner points to produce a τ-well-spaced superset M, for a constant $\tau > 2$ (choosing $\tau = 3$ is reasonable) [1,18]. Third, the facets of the Delaunay triangulation of M are each checked for a potential flip and added to the flip heap accordingly.

3.3 Flipping Out Steiner Points

We maintain the weighted Delaunay triangulation through a sequence of incremental flips induced by the changes in weights. While the flip heap is nonempty, we pop the next potential flip. The time t is set to be the time of this potential flip. If the flip is still valid, i.e. all of the relevant facets are still present in the weighted Delaunay triangulation at time t, then we perform the flip. Otherwise, we do nothing and continue. If any new facets are introduced, we check them for potential flips and add them to the flip heap accordingly. Then we loop. Figure 3 illustrates the sequence of Voronoi diagrams as the algorithm proceeds.

3.4 Postprocessing

To complete the construction, the algorithm removes all boundary vertices and all incident Delaunay simplices. In Sect. 4.1, we prove that only Del_P remains.

4 Analysis

4.1 Boundary Issues

The weighting will not remove all of the Steiner points. However, as the following lemma shows, only the boundary vertices will remain.

Lemma 1 (Only boundary vertices remain) *If $q \in \mathrm{Del}_{M_t}$ for some Steiner point q and all $t \geq 0$, then q is a boundary vertex.*

Proof Let p be any input point and let q be any non-boundary Steiner point. Let

$$t_\star = \max_{x \in \mathrm{Vor}_M(q)} \|p - x\|^2.$$

Such a maximum exists because the Voronoi cells of non-boundary vertices are compact. Suppose for contradiction that $q \in \mathrm{Del}_{M_t}$ for some $t > t_\star$. Since $q \in \mathrm{Del}_{M_t}$, there must exist some point $y \in \mathrm{Vor}_{M_t}(q)$. It follows that $y \in \mathrm{Vor}_M(q)$ as well and so $\|p - y\|^2 \leq t_\star$. We now observe that

$$\pi_p(y, t) = \|p - y\|^2 - t \leq t_\star - t < 0 \leq \|q - y\|^2 = \pi_q(y, t),$$

contradicting the assumption that $y \in \mathrm{Vor}_{M_t}(q)$. □

We need to show that removing the boundary vertices in a naive way as a post-processing phase gives the correct output. For this, it will suffice to show the following lemma.

Lemma 2 (Induced Subcomplex) *There exists $t_\star$ such that for all $t > t_\star$, Del_P is an induced subcomplex of Del_{M_t}.*

Proof For all $t \geq 0$, and each $p \in P$, we have $p \in \mathrm{Vor}_{M_t}(p)$. That is, input points are always their own nearest neighbors as weights increase. This implies that the vertices of Del_P are all present in Del_{M_t}.

Suppose for contradiction that some simplex σ is in $\mathrm{Del}_P \setminus \mathrm{Del}_{M_t}$ for some $t > r^2$, where r is the circumradius of σ. Then there must be some Steiner point q encroaching the orthoball of σ at time t. Let x be the circumcenter of σ. Since σ is composed only of input points, x is also the orthocenter of σ for all times t. For all $p \in \sigma$, we have

$$\pi_p(x, t) = \|p - x\|^2 - t = r^2 - t < 0 \leq \|q - x\|^2 = \pi_q(x, t).$$

It follows that q does not encroach the orthoball of σ at time t. This contradiction implies that $\mathrm{Del}_P \subset \mathrm{Del}_{M_{t_\star}}$ as long as $t_\star$ is greater than the largest squared circumradius of any simplex in Del_P.

To show that Del_P is an induced subcomplex and complete the proof, we observe that because Del_P and Del_{M_t} are embedded simplicial complexes covering the convex closure of P, there can be no simplices in Del_{M_t} containing only vertices of P that are not already included in Del_P. □

Finally, the last important fact to check is that there is not too much extra work to put the points in a bounding domain. In meshing, this slack between the bounding box and the input is called scaffolding [20].

Lemma 3 (Bounded Scaffolding) *The number of simplices removed in the final step of the algorithm is $O(|\mathrm{Del}_P|)$. That is, $|\mathrm{Del}_{M_\star} \setminus \mathrm{Del}_P| = O(|\mathrm{Del}_P|)$.*

Proof Any simplex $\sigma \in \mathrm{Del}_{M_\star} \setminus \mathrm{Del}_P$ can be written as a disjoint union $\sigma = S \sqcup T$ where $S \in \mathrm{ch}(M)$ and $T \in \mathrm{ch}(P)$. By construction, $\mathrm{ch}(M)$ has a constant number of vertices and thus $|\mathrm{ch}(M)|$ is also a constant. Since $\mathrm{ch}(P) \subseteq \mathrm{Del}_P$, we have $|\mathrm{ch}(P)| = O(|\mathrm{Del}_P|)$. It follows that there can be at most $|\mathrm{ch}(M)| \cdot |\mathrm{ch}(P)| = O(|\mathrm{Del}_P|)$ such simplices. □

4.2 Running Time Analysis

In this section, we will bound the number of flips by volume packing arguments which are much easier to formulate for Voronoi diagrams. Consequently, we change from the Delaunay perspective to the Voronoi perspective.

Theorem 1 (Running Time) *Given n points $P \subset \mathbb{R}^d$ in general position, the* MeshVoronoi *algorithm constructs the Voronoi diagram of P in $O(f \log \Delta \log n)$ time, where f is the number of faces of* Vor_P.

Before proceeding to the proof of the running time guarantee, we will first bound the number of combinatorial changes of the weighted Voronoi diagram and thus also the number of heap operations. These are the main technical points of the analysis.

We first prove a relatively standard packing bound on the number of Voronoi cells of Vor_M that can intersect the Voronoi cell of an input point.

Lemma 4 (Packing) *Let M be a τ-well-spaced superset of P satisfying the feature size condition that $\mathbf{f}_P(q) \leq K\mathbf{f}_M(q)$ for all $q \in M$ for some constants τ and K. Moreover, assume that M is contained in a bounding region Ω such that* $\mathrm{diameter}(\Omega) \leq c\,\mathrm{diameter}(P)$ *for a constant c. Let p be any point of P. Then, the number of points $q \in M$ such that* $\mathrm{Vor}_M(q) \cap \mathrm{Vor}_P(p) \neq \emptyset$ *is* $O(\log \Delta)$.

Proof The proof will be by a volume packing argument. Let M_p denote the set of points $q \in M \setminus \{p\}$ such that $\mathrm{Vor}_M(q) \cap \mathrm{Vor}_P(p) \neq \emptyset$. For any $q \in M_p$ and $x \in \mathrm{Vor}_M(q) \cap \mathrm{Vor}_P(p)$, we derive the following bound on the distance between p and q.

$$\begin{aligned}
\|p - q\| &\leq \|q - x\| + \|p - x\| && \text{[by the triangle inequality]}\\
&\leq \|q - x\| + \mathbf{f}_P(x) && [x \in \mathrm{Vor}_P(p)]\\
&\leq 2\|q - x\| + \mathbf{f}_P(q) && [\mathbf{f}_P \text{ is 1-Lipschitz}]\\
&\leq 2\tau\mathbf{f}_M(q) + \mathbf{f}_P(q) && [\text{since } M \text{ is } \tau\text{-well-spaced}]\\
&\leq (2\tau + K)\mathbf{f}_M(q) && \text{[by the feature size condition]}
\end{aligned}$$

Define $\gamma := \frac{1}{4\tau+2K}$ so that the preceding bound may be written as

$$\gamma\|p - q\| \leq \mathbf{f}_M(q)/2.$$

We partition the set M_p into geometrically growing spherical shells A_i, where for each integer i,

$$A_i := \{q \in M_p \mid (1+\gamma)^{i-1} \leq \|p - q\| \leq (1+\gamma)^i\}.$$

For each $q \in M_p$, define the ball $B_q := \text{ball}(q, \gamma \| p - q \|)$. The definition of B_q and the bound on $\| p - q \|$ imply

$$B_q \subseteq \text{ball}(q, \mathbf{f}_M(q)/2) \subset \text{Vor}_M(q),$$

and so the balls B_q are pairwise interior disjoint. For $q \in A_i$, we further get that $B_q \subset \text{ball}(p, (1+\gamma)^{i+1})$. Thus,

$$\begin{aligned} \text{Vol}(\text{ball}(p, (1+\gamma)^{i+1})) &\geq \text{Vol}\Big(\bigsqcup_{q \in A_i} B_q \Big) \\ &\geq |A_i| \text{Vol}(\text{ball}(q, \gamma(1+\gamma)^{i-1})). \end{aligned}$$

It follows that $|A_i| \leq (1+\gamma)^{2d}/\gamma^d$ for all i. The nearest point of M_p to p has distance at least $\mathbf{f}_P(p)/K$ from p by the feature size condition. If δ is the minimum pairwise distance among points of P, then $\delta/K \leq \mathbf{f}_P(p)/K$ So, for $i < \lfloor \log_{1+\gamma}(\delta/K) \rfloor$, A_i is empty. The farthest point of M_p to p has distance at most diameter(Ω). So, for $i > \lceil \log_{1+\gamma}(\text{diameter}(\Omega)) \rceil$, A_i is empty. Thus, there are at most

$$\lceil \log_{1+\gamma}(\text{diameter}(\Omega)) \rceil - \lfloor \log_{1+\gamma}(\delta/K) \rfloor = O(\log \Delta)$$

nonempty sets A_i. This completes the proof as we have shown that M_p can be decomposed into $O(\log \Delta)$ sets, each of constant size. □

4.2.1 Counting Flips

The main challenge in the analysis is to bound the number of flips that happen in the transformation from the Voronoi diagram of M to the Voronoi diagram of P. The key to bounding this number is to observe that each such flip is witnessed by the intersection of a k-face of Vor_M and a $(d-k)$-face of Vor_P for some k. Thus, we count these intersections instead. This intuition is made precise in the following lemmas. The first bounds the number of flips performed by the algorithm. The latter bounds the number of potential flips considered by the algorithm, since not all potential flips are performed.

Lemma 5 (Flip Bound) *Given n points $P \subset \mathbb{R}^d$, the* MeshVoronoi *algorithm performs $O(f \log \Delta)$ flips, where $f = |\text{Vor}_P|$.*

Proof A flip occurs exactly when the weights cause some pair of adjacent simplices to encroach on each other. That is, the two simplices share a common orthoball. Let U be the $d+2$ vertices comprising these simplices and let c be the center of their orthoball. Let $k+1$ be the number Steiner points in U. There must be at least one Steiner point and one input point in U in order for there to be a flip, so $0 \leq k \leq d$. The weights of the input points in U are all equal and thus these points are all also equidistant from c. So, c is contained in a k-face of Vor_P. The Steiner points of U have weight 0, so they are equidistant from c and closer to c than any of the input points. So, c is contained in a $(d-k)$-face of Vor_M, where M is the τ-well-spaced superset

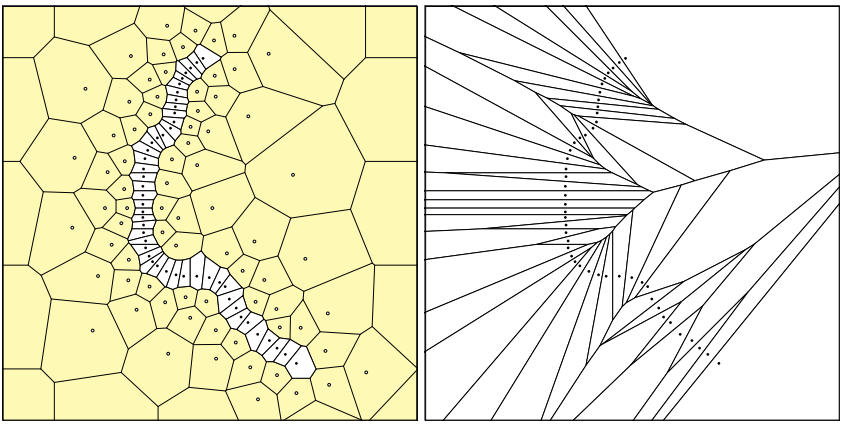

Fig. 4 To bound the number of flips we need only count the number of intersection between the two Voronoi diagrams, Vor_M on the *left* and Vor_P on the *right*

of P constructed by the algorithm. Thus, the point c is the intersection of a k-face F of Vor_P and a $(d-k)$-face G of Vor_M.

Now, to bound the total number of flips it will suffice to prove that each k-face of Vor_P intersects at most $O(\log \Delta)$ faces of dimension $d-k$ in Vor_M. Let F be any k-face of Vor_P and let $\mathrm{Vor}_P(p)$ be a d-face of Vor_P containing F. According to Lemma 4, there are at most $O(\log \Delta)$ points $q \in M$ such that $\mathrm{Vor}_M(q)$ intersects $\mathrm{Vor}_P(p)$. The d-faces of Vor_M have only a constant number of faces each, so there can be only $O(\log \Delta)$ faces of Vor_M intersecting $\mathrm{Vor}_P(p)$ and thus only $O(\log \Delta)$ such faces of dimension $d-k$ intersecting F. □

Thus, the key fact in the analysis is that the number of flips per Voronoi cell of Vor_P is bounded by the number of cells of Vor_M it intersects. See Fig. 4 for an example.

Lemma 6 (Potential Flip Bound) *Given n points $P \subset \mathbb{R}^d$, the* MeshVoronoi *algorithm sees $O(f \log \Delta)$ potential flips, where $f = |\mathrm{Del}_P|$.*

Proof At the start of the algorithm, there is one potential flip for each facet of Del_M. By (1), this is at most $O(n \log \Delta)$ potential flips. During the rest of the algorithm, there are at most $\binom{d+2}{d} = O(d^2)$ new potential flips each time a real flip occurs, one for each new facet that appears. By Lemma 5, this is $O(f \log \Delta)$. □

4.2.2 The Main Result

We are now ready to prove the running time guarantee, Theorem 1.

Proof of Theorem 1 It will suffice to bound the running time of each phase of the algorithm. The preprocessing takes $O(n \log \Delta)$ from the running time of Sparse Voronoi Refinement. Seeding the heap also takes $O(n \log \Delta)$ time as each facet is checked in constant time and the amortized cost of heap insertion is $O(1)$. There are at most two heap operations for each potential flip, one insertion and one deletion. The total

number of potential flips is $O(f \log \Delta)$ as shown in Lemma 6. Deleting the minimum element from a heap with $O(f \log \Delta)$ elements requires $O(\log(f \log \Delta)) = O(\log n)$ time. Thus, the total time for all heap operations is $O(f \log \Delta \log n)$ as desired. □

5 Conclusion

The algorithm we have presented is a direct combination of Delaunay mesh generation and kinetic data structures. The output-sensitive running time depends on the log of the spread. This is the usual cost of doing a geometric divide and conquer. It remains an interesting question if it is possible to replace the $\log \Delta$ term with a $\log n$ by some more combinatorial divide and conquer. The other log-term coming from the heap operations may also permit some improvement as it is clear that many flips are geometrically independent and so their ordering is not strict.

Another possible direction of future work is to exploit hierarchical meshes to keep the number of Steiner points linear [25]. However, it is not known how to leverage this into an improvement over the bounds presented here. At best it replaces the $\log \Delta$ with $\min\{\log \Delta, n\}$.

Going forward, we hope to extend the methods here to the convex hull problem. Likely, this will require significant new ideas, but it may be possible to achieve a similar output-sensitive running time of $O((f + n) \log \Delta \log n)$ for computing the convex hull of n points with f faces.

Acknowledgments GLM was partially supported by the NSF Grant CCF-1065106. DRS partially supported by the European project No. 255827 (CG-Learning).

References

1. Acar, U.A., Hudson, B., Miller, G.L., Phillips, T.: SVR: Practical engineering of a fast 3D meshing algorithm. In: Proceedings of the 16th International Meshing Roundtable, pp. 45–62 (2007)
2. Amato, N.M., Ramos, E.A.: On computing Voronoi diagrams by divide-prune-and-conquer. In: Proceedings of the 12th Annual Symposium on Computational Geometry, pp. 166–175 (1996)
3. Aurenhammer, F.: Voronoi diagrams-a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**(3), 345–405 (1991)
4. Aurenhammer, F., Klein, R., Lee, D.T.: Voronoi Diagrams and Delaunay Triangulations. World Scientific Publishing Company, Singapore (2013)
5. Avis, D., Fukuda, K.: A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. Discrete Comput. Geom. **8**(1), 295–313 (1992)
6. Bremner, D.: Incremental convex hull algorithms are not output sensitive. Discrete Comput. Geom. **21**(1), 57–68 (1999)
7. Chan, T.M.: Output-sensitive results on convex hulls, extreme points, and related problems. Discrete Comput Geom **16**(4), 369–387 (1996)
8. Chan, T.M., Snoeyink, J., Yap, C.K.: Primal dividing and dual pruning: output-sensitive construction of four-dimensional polytopes and three-dimensional Voronoi diagrams. Discrete Comput. Geom. **18**(4), 433–454 (1997)
9. Chand, D.R., Kapur, S.S.: An algorithm for convex polytopes. J. ACM **17**(1), 78–86 (1970)
10. Chazelle, B., Devillers, O., Hurtado, F., Mora, M., Sacristán, V., Teillaud, M.: Splitting a Delaunay triangulation in linear time. Algorithmica **34**, 39–46 (2002)
11. Chazelle, B., Mulzer, W.: Computing hereditary convex structures. Discrete Comput. Geom. **45**(4), 796–823 (2011)
12. Cheng, S.W., Dey, T.K., Shewchuk, J.R.: Delaunay Mesh Generation. CRC Press, Boca Raton (2012)

13. Edelsbrunner, H.: Geometry and Topology for Mesh Generation. Cambridge University Press, Cambridge (2001)
14. Edelsbrunner, H., Shah, N.R.: Incremental topological flipping works for regular triangulations. Algorithmica **15**, 223–241 (1996)
15. Erickson, J.: Nice point sets can have nasty Delaunay triangulations. Discrete Comput. Geom. **30**(1), 109–132 (2003)
16. Guibas, L.: Kinetic data structures. In: Mehta, D.P., Sahni, S. (eds.) Handbook of Data Structures and Applications, chap. 23, pp. 23-1–23-18. CRC Press, Boca Raton (2005)
17. Har-Peled, S., Mendel, M.: Fast construction of nets in low dimensional metrics, and their applications. SIAM J. Comput. **35**(5), 1148–1184 (2006)
18. Hudson, B., Miller, G., Phillips, T.: Sparse Voronoi Refinement. In: Proceedings of the 15th International Meshing Roundtable, pp. 339–356. Birmingham, Alabama (2006). Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132
19. Hudson, B., Miller, G.L., Phillips, T.: Sparse Parallel Delaunay Refinement. In: 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 339–347. San Diego (2007)
20. Hudson, B., Miller, G.L., Phillips, T., Sheehy, D.R.: Size complexity of volume meshes vs. surface meshes. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1041–1047 (2009)
21. Joswig, M., Ziegler, G.M.: Convex hulls, oracles, and homology. J. Symb. Comput. **38**(4), 1247–1259 (2004)
22. Klee, V.: On the complexity of d-dimensional Voronoi diagrams. Archiv der Mathematik **34**, 75–80 (1980)
23. Lawson, C.L.: Properties of *n*-dimensional triangulations. Comput Aided Geom. Des. **3**, 231–246 (1986)
24. Matoušek, J., Schwarzkopf, O.: Linear optimization queries. In: Proceedings of the 8th Annual Symposium on Computational Geometry, pp. 16–25 (1992). http://doi.acm.org/10.1145/142675.142683
25. Miller, G.L., Phillips, T., Sheehy, D.R.: Beating the spread: Time-optimal point meshing. In: Proceedings of the 27th ACM Symposium on Computational Geometry, pp. 321–330 (2011)
26. Miller, G.L., Sheehy, D.R., Velingker, A.: A fast algorithm for well-spaced points and approximate Delaunay graphs. In: Proceedings of the 29th ACM Symposium on Computational Geometry, pp. 289–298 (2013)
27. Miller, G.L., Talmor, D., Teng, S.H., Walkington, N.: On the radius-edge condition in the control volume method. SIAM J. Numer. Anal. **36**(6), 1690–1708 (1999)
28. Seidel, R.: Constructing higher-dimensional convex hulls at logarithmic cost per face. In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, pp. 404–413 (1986)
29. Seidel, R.: On the number of faces in higher-dimensional Voronoi diagrams. In: Proceedings of the 3rd Annual Symposium on Computational Geometry, pp. 181–185 (1987)
30. Sheehy, D.R.: Mesh generation and geometric persistent homology. Ph.D. thesis, Carnegie Mellon University (2011)
31. Sheehy, D.R.: New bounds on the size of optimal meshes. Comput. Graph. Forum **31**(5), 1627–1635 (2012)
32. Swart, G.: Finding the convex hull facet by facet. J. Algorithm. **6**(1), 17–48 (1985)