Kinetic Data Structures for Clipped Voronoi Computations

Duru Türkoğlu*

Abstract

We consider the mesh refinement problem in the kinetic setting: given an input set of moving points, the objective is to design a kinetic data structure (KDS) for inserting additional so-called Steiner points so that the resulting output set yields a quality triangulation. Therefore, the selection of Steiner points plays a cruicial role, both in terms of the output itself and the quality of the triangulation. Although many Steiner point selection schemes have been devised, it is not straightforward how to adapt them in the kinetic setting; it may not even be possible to adapt some of these schemes.

In this paper, we design a KDS by extending a previously proposed query structure which has been employed for Steiner point selection in the dynamic setting. The key geometric property computed in these structures is the clipped Voronoi cells, a locally restricted version of the standard Voroni cells. Our KDS maintains these clipped Voronoi cells, where each query takes constant time to compute as well as to update. Hence, our KDS is responsive, and it is efficient processing a constant number of events for each query, and it is also local and compact.

1 Introduction

Mesh refinement is a fundamental step in scientific computing, where the goal is to construct a quality triangulation of a given set of input points. For most applications, a quality triangulation is one in which the minimum angle of every triangle is above some threshold. Until this quality criterion is satisfied, one needs to *refine* the trianguation by inserting additional *Steiner* points into the output, taking care to insert as few as possible. If the minimal output that admits a quality triangulation has m points, any output of size $O(m)$ is regarded as *size-optimal*.

Over the past twenty years, research has provided improved mesh refinement algorithms with theoretical guarantees. In chronological order, these guarantees were quality [4], size-optimality [8], efficient runtimes [5, 6], and efficient dynamic updates [1], each of them incorporating all of the previous guarantees. And most recently, Acar et al. developed a KDS for mesh refinement [2]. Their choice of Steiner points, however, only allowed bounded quality triangulations. Hence, mesh refinement problem in the kinetic setting still remains open for higher quality triangulations.

In the kinetic setting, the objective of mesh refinement is to maintain a set of moving Steiner points as well as the triangulation of the output set comprising input and Steiner points. Typically, algorithms used in practice maintain the moving mesh by a series of further refinements to fix low quality elements over time, and by remeshing occasionally to bound the size of the mesh. In this paper, we approach the problem using the kinetic data structures (KDS) framework introduced by Basch et al [3]. In this framework, the input points are provided with algebraic trajectories of constant degree and the computed geometric property is certified by a set of *certificates*, some of which may fail at a later time causing an *event*. Each event triggers a kinetic update for repairing both the geometric property itself and the set of certificates certifying it. As the trajectories of the input points are known in advance, the data structure maintains the desired geometric properties by processing these events in order of their failure times. In this framework effectiveness is measured by the following criteria: a KDS is called *responsive* if the kinetic updates are fast in the worst case, *efficient* if the number of events is small when compared to the number of intrinsic changes the geometric property has to go through in the worst case, *local* if each input point is associated with a small number of certificates, and *compact* if the total number of certificates and the size of the data structure is small.

Within the KDS framework, Steiner point selection becomes a hard problem; many that are suitable when input points do not move are hard to adapt effectively in the kinetic setting. One such example is the quadtree method, where the corners of the quadtree squares are inserted as Steiner points [4]. Another very commonly used such example is the circumcenters of low quality triangles [6, 8]. Aside from these very common examples, Üngör defines a more local type of Steiner points called off-centers [10], Hudson and Türkoğlu propose a local region to pick Steiner points from, a region called clipped Voronoi cells, one which includes off-centers [7], and Acar et al. choose Steiner points as geometric translations of input points at geometrically increasing distances [2]. In this paper, we use the kinetic mesh of Acar et al. as a point location structure and extend the clipped Voronoi computations of Hudson and Türkoğlu to the kinetic setting.

*Department of Computer Science, University of Chicago

Our contribution in this paper is a first step towards solving the kinetic mesh refinement problem for achieving arbitrary quality triangulations. Building on the result of Hudson and Türkoğlu [7], we design a KDS for computing clipped Voronoi cells so that a meshing algorithm can use it for picking Steiner points to construct the output mesh. Given a point v, our KDS provides the following procedures pertaining to v: ApproximateNN(v) to compute an approximation of the distance from v to its nearest neighbor, ClippedVoronoi(v, β) to compute the clipped Voronoi cell of v, and AddVertex(p, v) to insert into the output a Steiner point p near v. We prove in Theorem 3 that all of the above procedures run in constant time, create a constant number of certificates, and hence can be updated in constant time upon a certificate failure. Our KDS reduces the problem of kinetic mesh refinement of arbitrary quality to suitable choice of Steiner points within local neighborhoods.

2 Definitions

In this section we provide preliminary definitions for the rest of the paper. To distinguish mesh points from any point in space we refer to mesh points as vertices.

Definition 1 *Given a vertex v, $\mathrm{NN}(v)$ is the distance from v to its nearest neighbor, and $\mathrm{Vor}(v)$ is the Voronoi cell of v consisting of all points x such that for any vertex $u \neq v$, $|ux| \geq |vx|$. Then v is called ρ-well-spaced [9] if $\mathrm{Vor}(v)$ is inside the ball centered at v with radius $\rho\,\mathrm{NN}(v)$, and a mesh is called ρ-well-spaced if every vertex in the mesh is ρ-well-spaced.*

Using the fact that the dual of a Voronoi diagram yields a Delaunay triangulation, one can observe that well-spacedness is equivalent to large minimum angles in the output triangulation; the lower the parameter ρ, the larger the minimum angles.

Definition 2 *[7] The β-clipped Voronoi cell of a vertex v, $\mathrm{Vor}^{\beta}(v)$, is the intersection of $\mathrm{Vor}(v)$ and the ball centered at v with radius $\beta\,\mathrm{NN}(v)$. For any point $x \in \mathrm{Vor}^{\beta}(v)$, the ball centered at x with radius $|vx|$ is a witness[1] ball empty of vertices, and the witness region of $\mathrm{Vor}^{\beta}(v)$ is the union of witness balls.*

Figure 1 depicts the above definitions of well-spacedness, clipped Voronoi cells, witness balls and witness regions.

Definition 3 *[8] Given a point x in space, the local feature size of x, $\mathrm{lfs}(x)$, is the distance from x to its second-nearest input vertex, and a size-conforming mesh is one in which for every output vertex v, $\mathrm{NN}(v) \in \Theta(\mathrm{lfs}(v))$; Ruppert proves that being size-optimal is equivalent to being size-conforming.*

[1] The origial term used in [7] is "certificate", however, we use the term "witness" to avoid confusion with KDS certificates.

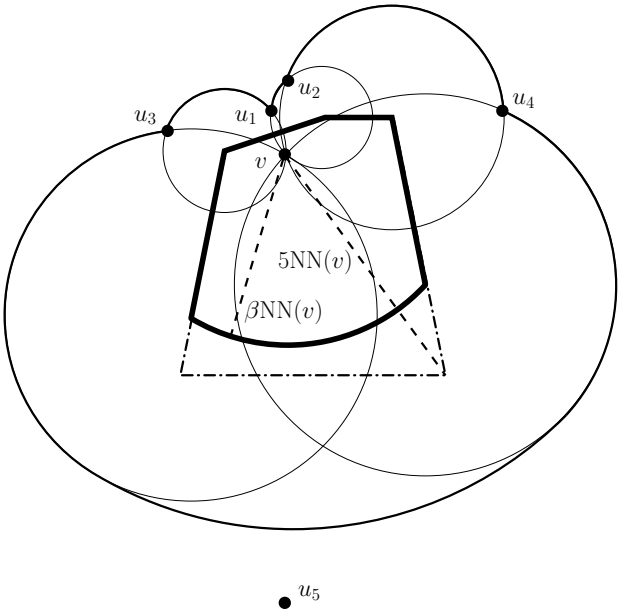

Figure 1: A vertex v and its neighbors u_1 through u_5. Farthest point in $\mathrm{Vor}(v)$ is at $5\,\mathrm{NN}(v)$ distance, thus, v is 5-well-spaced but not β-well-spaced. As a result, $\mathrm{Vor}^{\beta}(v)$ is shown with a thick boundary and its witness balls and witness region with a thin boundary.

Definition 4 *A mesh refinement algorithm is bottom-up if it incrementally ensures well-spacedness of the vertices in the order of their local feature size. For a given constant γ, a bottom-up algorithm inserts a Steiner vertex u, when every mesh vertex v with $\mathrm{NN}(v) < \gamma\,\mathrm{NN}(u)$ is ρ-well-spaced.*

Our data structure requires a bottom-up meshing algorithm that outputs a size-conforming mesh to ensure local searches and fast runtimes.

3 Data Structure and Implementation

Hudson and Türkoğlu use quadtrees for point location as they provide cruicial properties to guarantee fast runtimes. However, quadtrees do not adapt well to the kinetic setting, and we instead use the kinetic mesh of Acar et al. as our point location structure [2]. Their KDS picks Steiner points from points they call *satellites*, pre-defined geometric translations of input vertices at geometrically increasing distances. They maintain the Voronoi diagram of their output vertex set and their Voronoi cells, or *cells* in short so as to distinguish from the Voronoi cells of our definition, provide properties similar to those of the quadtree squares, and our guarantees depend on them:

(A) Each cell has a constant number of neighbors.

(B) The size of each cell is in proportion to the local feature size of the points in the cell.

The size of a cell can be described by the nearest neighbor distance of its node with respect to other nodes in the diagram, and the well-spacedness of the node guarantees that the whole cell is contained within a ball of constantly larger radius, less than 9/2 times larger to be precise. As shown in Lemma 3.6 [2], the nearest neighbor is at distance $> 2^{\ell-2}$, where ℓ, defined as the *rank* of the node, is already computed by the data structure. Using the rank information of the nodes, we implement our procedures as follows:

- ApproximateNN(v) returns the distance $2^{\ell-2}$, where ℓ is the rank of v. This procedure does not create any certificates.
- AddVertex(p, v) starts at the cell of the node that contains v and iteratively advances by moving the search to the neighbor node closest to p. Finally, it reaches the cell that contains p as none of its neighbor nodes is closer to p, and inserts p in that cell as a Steiner vertex.

We certify this procedure by the distance comparisons between the final cell and its neighbors which create a set of what we call *cell* certificates. Upon the failure of a cell certificate, we update this search by restarting it from the node of the failure.

- ClippedVoronoi(v, β) performs in two stages; in the first stage it collects information about nearby vertices. It starts at the cell of the node that contains v and explores the region by moving outward along neighboring cells, in increasing order of their nodes' distances from v. It continues the search to find the nearest neighbor of v and to determine $\mathrm{NN}(v)$. Then it continues the search to explore other vertices within $2\beta\,\mathrm{NN}(v)$ distance of v. It stops exploring further at a cell if one of the two stopping conditions is satisfied:

  1. The distance from v to the cell is too large.
  2. The size of the cell is too small.

  Once the exploration of the nearby vertices is over, it proceeds to the second stage to determine actual Voronoi neighbors among the vertices within $2\beta\,\mathrm{NN}(v)$ distance. It removes a vertex u from the set if there is no witness ball incident to v and u empty of vertices or if the radius of the smallest one is larger than $\beta\,\mathrm{NN}(v)$ (Figure 2). It returns the remaining set as $\mathrm{Vor}^{\beta}(v)$.

To make the implementation of the procedure ClippedVoronoi(v, β) more precise, we introduce two constants λ_1 and λ_2 for the stopping conditions. In the first condition, we define the distance from v to a cell with node p to be too large if $|vp| > (2\beta+\lambda_1)\,\mathrm{NN}(v)$ (Lemma 1). And in the second condition, we define the size of a cell to be too small if the rank of its node is less than $\ell - \lambda_2$, where ℓ is the rank of v (Lemma 2).

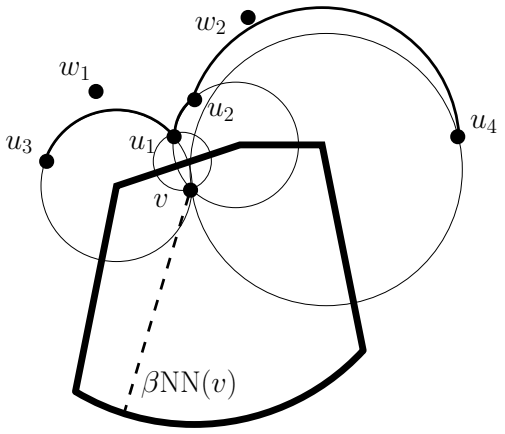

Figure 2: Setting of Figure 1. $\mathrm{Vor}^{\beta}(v)$ is represented by $\{u_1, u_2, u_3, u_4\}$. Smallest witness balls incident to each Voronoi neighbor u_i and v is depicted. For the vertices w_1 and w_2 within $2\beta\,\mathrm{NN}(v)$ distance there are no witness balls. Each of the circumcircles defined by the triplets $\{v, w_1, u_i\}$ and $\{v, w_2, u_i\}$ contains at least one other vertex.

Finally, we summarize the certificates and the kinetic update of ClippedVoronoi(v, β). In the first stage, distance comparisons create what we call *search* certificates. And, in the second stage, distance to circle comparisons create what we call *voronoi* certificates. Upon the failure of any one certificate, we restart that stage from scratch.

4 Proofs

In this section, we first prove that there exists constants λ_1 and λ_2 for our procedures; we then prove that our KDS is effective. Our proofs rely on the assumptions that the mesh refinement algorithm that employs our KDS is bottom-up and that its output is size-conforming.

The underlying summary of the proofs in this section is that local feature size is a smooth function and working with size-conforming structures also achieve similar smoothness properties. The first Lemma is based on the property that lfs cannot get too large within a locality.

Lemma 1 *There exists a constant λ_1 such that if the procedure* ClippedVoronoi(v, β) *stops exploring a cell because of the first condition, the distance from v to any point in that cell is greater than $2\beta\,\mathrm{NN}(v)$.*

Proof. For any point x within $2\beta\,\mathrm{NN}(v)$ distance of v, using the triangle inequality, we know that there

exist two output vertices within $O(\mathrm{NN}(v))$ distance of x, namely v and the nearest neighbor of v. And, since we assume that the output is size-conforming, we have $\mathrm{lfs}(x) \in O(\mathrm{NN}(v))$. Then condition (B) implies that the cell that includes x has size bounded by $O(\mathrm{NN}(v))$. Let λ_1 be the constant hidden in the big-Oh notation and let p be the node of that cell. Using the triangle inequality we have $|vp| \leq |vx| + |xp|$ or $|vp| \leq (2\beta + \lambda_1)\,\mathrm{NN}(v)$. Proof follows from the contrapositive argument. □

The below Lemma is based on the property that lfs stays large within large empty balls. In other words, if lfs is too small in some local neighborhood, then there must be vertices in between to support smoothness, and well-spacedness is sufficient to ensure that.

Lemma 2 *Assuming that for every mesh vertex u with $\mathrm{NN}(u) < \gamma\,\mathrm{NN}(v)$ is ρ-well-spaced, there exists a constant λ_2 such that the union of the cells explored by the procedure* ClippedVoronoi(v, β) *covers the witness region of* $\mathrm{Vor}^{\beta}(v)$.

Proof. Given a cell at which ClippedVoronoi(v, β) stops exploration, it is sufficient to prove that the cell does not intersect the witness region of $\mathrm{Vor}^{\beta}(v)$. If the reason to stop is the first condition, by Lemma 1, we know that the cell lies at a distance more than $2\beta\,\mathrm{NN}(v)$ away from v, thus it cannot intersect the witness region. If the reason to stop is the second condition, for any point p in the witness region, we know that p lies in a ball of radius at least $\mathrm{NN}(v)/2$. Therefore the premise of this Lemma satisfies the premise of Theorem 3 in [7]. As a result, we have $\mathrm{lfs}(p) \in \Omega(\mathrm{NN}(v))$ or $\mathrm{lfs}(p) \in \Omega(2^{\ell})$. This means that there exists a constant λ_2 large enough such that by condition (B), for any cell of rank less than $\ell - \lambda_2$, the local feature size of any point in the cell is less than $\mathrm{lfs}(p)$, which is to say those cells cannot intersect the witness region. □

Since lfs is a smooth function and since we ensure that the search does not extend to neighborhoods that have small local feature sizes, packing arguments prove that our procedures take constant time.

Theorem 3 *In our KDS,* ClippedVoronoi(v, β) *maintains* $\mathrm{Vor}^{\beta}(v)$ *and* AddVertex(p, v) *maintains insertion of a Steiner vertex p in constant time.*

Proof. The proof for ClippedVoronoi(v, β) relies on two facts: the search explores only a constant number of cells and each cell contains a constant number of vertices. For the first fact we use a packing argument: Lemma 1 bounds λ_1 which in turn proves that the search does not go beyond a distance of $O(\mathrm{NN}(v))$, and on the other hand, Lemma 2 bounds λ_2 which in turn proves that each cell explored is of size at least $\Omega(\mathrm{NN}(v))$. For the second fact we use another packing argument: for a given cell, condition (B) states that the size of the cell is bounded by $O(\mathrm{lfs}(p))$ for any given point p inside the cell, and since the mesh is size-conforming, any vertex p inside the cell has an empty ball of size $\Omega(\mathrm{lfs}(p))$.

The proof for AddVertex(p, v) is simpler: p must lie within the cells explored in the ClippedVoronoi(v, β) call, thus the search for p involves only a subset of those cells. □

5 Conclusion

In this paper, we showed how to support a class of mesh refinement algorithms with a kinetic data structure that could help find appropriate Steiner points in constant time. For simplicity, we have chosen to explain the algorithms in a purely theoretical fashion, however, we can employ several improvements that will make the constants in our algorithms practical.

References

[1] U. A. Acar, A. Cotter, B. Hudson, and D. Türkoğlu. Dynamic well-spaced point sets. In *SCG '10: Proceedings of the 26th Annual Symposium on Computational Geometry*, pages 314–323, 2010.

[2] U. A. Acar, B. Hudson, and D. Türkoğlu. Kinetic mesh refinement in 2d. In *Proceedings of the Twenty-seventh Annual Symposium on Computational Geometry*, SoCG '11, pages 341–350, 2011.

[3] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999.

[4] M. Bern, D. Eppstein, and J. R. Gilbert. Provably Good Mesh Generation. *Journal of Computer and System Sciences*, 48(3):384–409, 1994.

[5] S. Har-Peled and A. Üngör. A time-optimal Delaunay refinement algorithm in two dimensions. In *21st Symposium on Computational Geometry*, pages 228–236, 2005.

[6] B. Hudson, G. L. Miller, and T. Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, 2006. Long version in Carnegie Mellon University Tech. Report CMU-CS-06-132.

[7] B. Hudson and D. Türkoğlu. An efficient query structure for mesh refinement. In *Canadian Conference on Computational Geometry*, pages 115–118, 2008.

[8] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[9] D. Talmor. *Well-Spaced Points for Numerical Methods*. PhD thesis, Carnegie Mellon University, Pittsburgh, August 1997.

[10] A. Üngör. Off-centers: A new type of Steiner point for computing size-optimal quality-guaranteed Delaunay triangulations. In *LATIN*, pages 152–161, 2004.