1A.4

Construction of Sparse Well-spaced Point Sets for Quality Tetrahedralizations

Ravi Jampani and Alper Üngör

Dept. of Computer & Information Science & Engineering
University of Florida
{rjampani,ungor}@cise.ufl.edu

Summary. We propose a new mesh refinement algorithm for computing quality guaranteed Delaunay triangulations in three dimensions. The refinement relies on new ideas for computing the goodness of the mesh, and a sampling strategy that employs numerically stable Steiner points. We show through experiments that the new algorithm results in sparse well-spaced point sets which in turn leads to tetrahedral meshes with fewer elements than the traditional refinement methods.

1 Introduction

We consider the following three dimensional geometric problem:

Problem 1. [Quality Steiner Triangulation] *Compute a small size triangulation of a given three dimensional domain such that all the tetrahedra in the triangulation are of good quality.*

The quality constraint is motivated by the numerical methods used in many engineering applications. Among various criteria, the following two are widely used to describe the goodness of a tetrahedron:

1. *Radius-ratio.* A tetrahedron is said to be *good* if its radius ratio (circumradius over inradius) is bounded from above.
2. *Radius-edge-ratio.* A tetrahedron is said to be *good* if its radius-edge ratio (circumradius over shortest edge length) is bounded from above.

An upper bound on the former criterion implies an upper bound on the latter, but not vice versa. Moreover, a bound on the radius ratio is equivalent to a bound on the dihedral angles of a tetrahedron. In this paper we mainly, focus on the former quality criterion. Point sets whose triangulation has bounded radius-edge ratio are also known as the well-spaced point set [41]. We can easliy meet the latter criterion using a sliver-removal algorithm [5, 12] as a postprocessing step to our algorithm. Under the quality constraint, our

objective is to make the triangulation size (the number of tetrahedra) as small as possible for its efficient use in applications.

The quality triangulation problem has been extensively studied both in two dimensions [1, 7, 11, 14, 28, 42] and three dimensions [3, 5, 12, 16, 19, 25, 30]. There are two main techniques that solve the mesh generation problem and provide theoretical guarantees: (i) Delaunay refinement and (ii) quadtree refinement. Among these Delaunay refinement method seems to be more popular due to its superior performance in practice. There are also a number of heuristic solutions addressing the mesh generation problem, such as the advancing front methods [20] and the sphere packing methods [18, 36]. While these algorithmic and heuristic methods are generally effective computing quality triangulations in two dimensions, it is hard to claim the same in three dimensions. There are two main reasons for the shortcomings in three dimensional mesh generation. First, algorithmic solutions for computing optimal triangulations remain as a major open research topic. There is no known algorithm for computing triangulations in three dimensions that maximize the minimum dihedral angle, minimize the maximum dihedral angle, or any other useful optimization criteria. Second, it is hard to come up with a point sampling strategy that leads to a good triangulation. Here, we address the second issue and give a Steiner point selection strategy effective in computing Delaunay meshes.

Delaunay refinement method involves first computing an initial Delaunay triangulation of the input domain, and then iteratively adding points called *Steiner points* to improve the quality of the triangulation. Traditionally, circumcenters of bad simplices are used as Steiner points [28, 30]. Recently, alternative techniques have been studied in two dimenions, with great benefits both in theory and in practice. We elaborate more on this in the next section.

In this paper, we present a new tetrahedral mesh refinement algorithm, which relies on the following main ideas:

1. We first present a new criterion to check whether a triangulation is good or not. We show that this new criterion is equivalent to the traditional way (a triangulation is good if all tetrehedra have good radius-edge ratio). The new criterion considers edges and the shape of their dual Voronoi facets.
2. Unlike the traditional algorithms, which insert one vertex at a time to fix a bad tetrahedron, we insert multiple Steiner vertices to fix all bad tetrahedra incident to a "loose" edge. Moreover, Steiner points used here are often different from the circumcenters (Voronoi vertices). The proposed strategy results in sparser well-spaced point sets and meshes with fewer tetrahedra.

2 Previous Work vs. Our Focus

In our review, we mainly focus on the Delaunay based mesh generation. The first generation of tetrahedral refinement algorithms [9, 30] came as extensions of the planar Delaunay refinement algorithms [7, 28]. These algorithms employed circumcenters of bad tetrahedra as Steiner points and provided upper bounds on the radius-edge ratio. Then, the tetrahedral meshing research progressed mainly in addressing two important challenges described below. Following those, we describe two other challenges, the latter of which is our focus in this paper.

Slivers

There are tetrahedra, called *slivers*, with bounded radius edge ratio but unbounded dihedral angles (and radius ratio). Slivers are both undesirable and ubiquitous in three dimensional (Delaunay) triangulations. Even when the point set is well-spaced slivers may result. Recently, several algorithms have been developed that are guaranteed to result in meshes with no slivers [3, 5, 12, 16, 19]. These can be grouped in three classes (i) those that employ weighted Delaunay triangulations [3, 5], (ii) those that rely on a perturbation on the Steiner points, and (iii) those that use a more structured Steiner point packing such as octree or lattice structures [16, 17, 25]. The theoretical guarantees provided by the first two classes are too small to be relevant in practice. The rigid structure of the Steiner point packing employed by the last class of algorithms leads to dense points and hence meshes with significantly large number of elements.

Input Constraints

Computing triangulations of three dimensional domains is a major challenge in itself even without enforcing any quality measure or output size requirement. This is mainly due to the input boundary constraints. For instance, there are polyhedra which cannot be triangulated unless Steiner points are allowed [2, 27, 29]. Delaunay triangulation of the vertices of an input domain may not conform the domain boundary (facets) in general. The conforming and constrained Delaunay triangulation algorithms [8, 26, 33, 35, 39] address this issue. These algorithms are integrated within the Delaunay refinement framework. Earlier Delaunay refinement algorithms had strong constraints on the input type which they can handle. For instance, the dihedral angles between the input facets are assumed to be reasonably large [30]. Later, algorithms that can handle smaller input angles and larger classes of inputs are studied. [4, 32]

Time efficiency

The original Delaunay refinement algorithm has quadratic time complexity [28]. This compares poorly to the time-optimal quadtree refinement algorithm

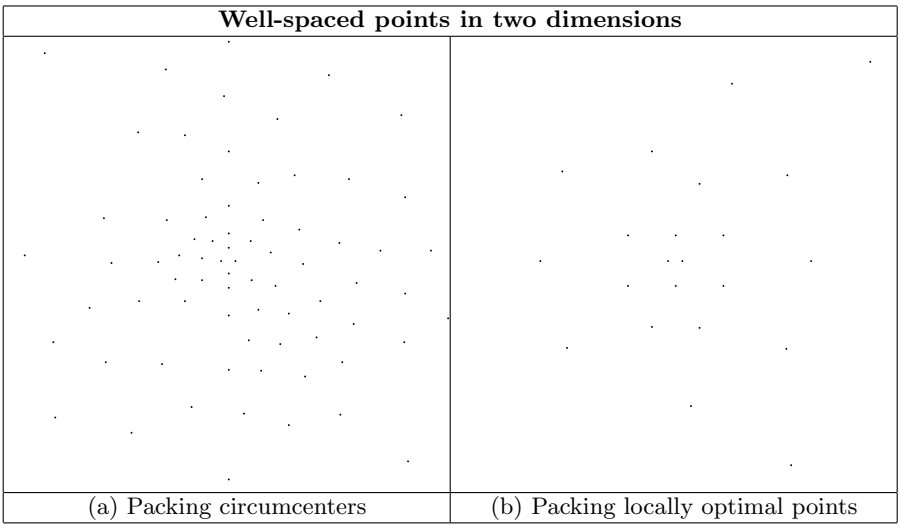

Fig. 1. Vertices of the output of the planar Delaunay refinement algorithms are shown for a constraint angle of 30°. Input in this case is a pair of points surrounded by a large empty space. The traditional circumcenter insertion packs many more Steiner points (a) than the locally optimal Steiner point placement strategy (b).

of Bern *et al.* [1] which runs in $O(n \log n + m)$ time, where m is the minimum size of a good quality mesh. The first improvement was given by Spielman *et al.* [37, 38] as a consequence of their parallelization of the Delaunay refinement algorithm. Their algorithm runs in $O(m \log m \log^2(L/h))$ time (on a single processor), where L is the diameter of the domain and h is the smallest feature in the input. Later, Miller [21] further improved this describing a new sequential Delaunay refinement algorithm with running time $O((n \log(L/h) + m) \log m)$. Then, Har-Peled and Üngör [14] presented the first time-optimal Delaunay refinement algorithm, which runs in $O(n \log n + m)$ time. This algorithm, originally presented in two dimensions has a natural extension to three dimensions for point sets. More recently, alternative time-efficient Delaunay refinement algorithms for higher dimensionsional meshing have been introduced [15]. The implications and benefits of the time-efficient algorithms in practice have yet to be explored. We should note that the Steiner point placement algorithm we propose here complements the time-optimal algorithm framework proposed in [14], and promises to be a good algorithm both in theory and practice.

Point Packing

Our focus in this study is on creating good packings of the Steiner points. We want the output point sets to be well-spaced [41], as the Delaunay triangulation of such point sets has bounded radius-edge ratio. On the other hand, having the packing sparse leads to meshes with fewer elements. Recent

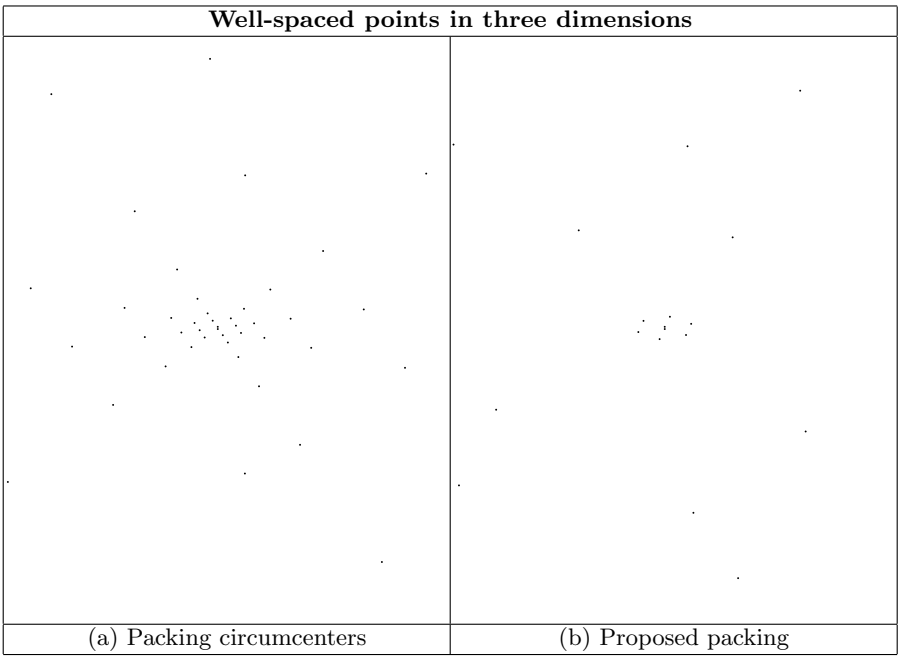

Fig. 2. A careful packing results in sparse well-spaced point sets in three dimensions. Here input is a pair of points inside a cube. We zoomed in to illustrate the difference in our packing strategy (b) with respect to the traditional iterative circumcenter packing (a). See Figure 6 for the complete view of this data set.

research indicated that a more careful packing strategy than the traditional circumcenter packing has great benefits in two dimensional mesh generation [42, 13]. Thanks to the sparsity of the generated point sets, the meshes tend to be a factor of two or more smaller than the meshes generated by iterative circumcenter insertions. This implies substantial reduction not only in triangulation time, but also in the running time of the subsequent application algorithms. Moreover, the new Steiner point packing strategies help us design software that are effective for stronger constraint angles than the traditional methods. The original Delaunay refinement algorithm of Ruppert is proven to terminate with size-optimal quality triangulations for $\alpha \leq 20.7°$. In practice, it generally works for $\alpha \leq 34°$ and fails to terminate for larger constraint angles. The new variant of the Delaunay refinement algorithm generally terminates for constraint angles up to $42°$. Figures 1 and 2 illustrate the benefits of a locally optimal Steiner packing in two dimensions, and potential benefits of the proposed packing in three dimensions. At a recent workshop, the second author sketched ideas to illustrate the potential benefits in three dimensions [43].

2.1 Standard Delaunay refinement

We review the general framework of the three dimensional Delaunay refinement algorithm. We refer the reader to [10, 30] for further details.

In three dimensions, a collection Ω of vertices, segments, and facets is called a *piecewise linear complex* (PLC) if (i) all lower dimensional elements on the boundary of an element in Ω also belong to Ω, and (ii) if any two elements intersect, then their intersection is a lower dimensional element in Ω [24]. We first compute the Delaunay triangulation of the set of vertices of the input PLC Ω. Then, we add new points (i) to recover the edges and facets that are not conformed by the Delaunay triangulation and (ii) to improve the quality of the triangulation. A point is said to *encroach* upon a simplex if it is inside the smallest sphere that contains the simplex. A tetrahedron is considered *bad* if its radius-edge ratio is larger than a pre-specified constant $\beta \geq 2$. We maintain the Delaunay triangulation as we add new points using the following rules.

1. If a segment is encroached upon, we add its midpoint.
2. If a facet is encroached upon, we add its circumcenter unless Rule 1 applies.
3. If a tetrahedron is of bad quality, we add its circumcenter unless Rule 1 or 2 applies.

Shewchuk [30, 31] showed that this algorithm is guaranteed to compute tetrahedral meshes with bounded radius-edge ratio. That is, the output vertex set is well-spaced. He also showed that the point set has good grading with respect to input feature size distribution. The key component of the termination guarantee and the correctness of the Delaunay refinement algorithms is that a Steiner point insertion does not introduce features smaller than the existing ones. In this paper, we show that this property holds for the our Steiner point packing strategy. The termination property and the correctness of the algorithm follows, replacing the corresponding lemmas in the standard Delaunay refinement framework. For simplicity, we do not repeat the entire framework, and refer to [10, 30] for the structure of the complete proof of correctness for Delaunay refinement algorithms.

3 Preliminaries for our Algorithm

We assume the user is familiar with the definitions of Delaunay triangulations and Voronoi diagrams [10]

3.1 Edges and their Voronoi Facets

Consider a pair of points p and q in the mesh that are connected to each other in the Delaunay triangulation. (See Figure 3.) Let $Vor(pq)$ denote the Voronoi facet dual to the Delaunay edge pq.

Definition 1. *Let S be the set of spheres that have radius $\beta|pq|$ and go through the points p and q. The disk formed by the centers of the spheres in S is called the* **core disk** *of pq, denoted as $CoreDisk(pq)$.*

Note that $CoreDisk(pq)$ is coplanar with $Vor(pq)$.

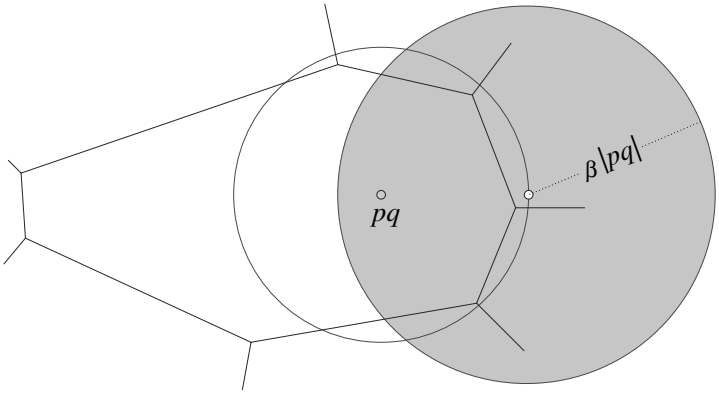

Fig. 3. Let pq be an edge (orthogonal to the view plane) in a 3D triangulation. The Voronoi face dual of the edge pq is shown as a polygonal region. The shaded region represents a sphere with radius $\beta|pq|$ that goes through points p and q. The centers of all such spheres form a circle, i.e. $CoreDisk(pq)$.

Definition 2. *A Delaunay edge pq is said to be* **short** *if it is the shortest edge of a tetrahedron incident to it.*

Definition 3. *A Delaunay edge pq that is not on the boundary is said to be* **bad** *if $Vor(pq) \nsubseteq CoreDisk(pq)$. A boundary Delaunay edge is* **bad** *if there exists a vertex of $Vor(pq)$ that is not in $CoreDisk(pq)$.*

We prove below that the Delaunay triangulation of a point set includes a bad tetrahedron if and only if it includes a bad edge. We propose a local sampling strategy which introduces several Steiner points at once to ensure $Vor(pq) \subseteq CoreDisk(pq)$ after the insertion for every Delaunay edge pq. Our sampling strategy should be locally optimal maximizing the smallest new pairwise point distance.

Let R_c denote the radius of the $CoreDisk(pq)$. Then, $R_c = \sqrt{\beta^2 - 1/4}|pq|$.

Theorem 1. *Delaunay triangulation of a point set has bad tetrahedra if and only if there is a bad edge in the triangulation.*

Indeed we can prove the following stronger statement, which considers only the short edges. This modification plays a role in the efficiency of the sampling algorithm for fixing the bad edges. That is we only need to address the short edges.

Theorem 2. *Delaunay triangulation of a point set has bad tetrahedra if and only if there is a bad short edge in the triangulation.*

Proof. If there exists an edge pq whose Voronoi cell is not contained inside the CoreDisk then clearly there is a bad tetrahedron (which is the dual of the Voronoi vertex that is outside the CoreDisk). For the other direction, consider a bad tetrahedron with the shortest edge pq. Then, its dual can be shown not to be contained.

4 Proposed Refinement Algorithm

Our algorithm should be seen as a proper extension of the offcenter algorithm presented in two dimensions [42]. The extension, however, is not straightforward. In two dimensions, an edge is incident to two triangles. This makes it easy to compute Steiner points such that an existing "short" edge is guaranteed to be neighbour to two good triangles. The main challenge in three dimensions is there could be a varying number of tetrahedra incident to an edge. That there are very many ways to pack points around a small feature so that it is surrounded by good tetrahedra. Our algorithm (presented as Algorithm 1) first detects a collection of consecutive bad tetrahedra incident to a short edge and then introduces potentially multiple Steiner points so that the edge is incident to a set of good tetrahedra (or it dissappears from the triangulation). We should note that we always maintain the Delaunay property of the triangulation.

Algorithm 1

```
Compute the Delaunay triangulation of the input
while ∃ a bad short edge pq
    Compute the connected components of Vor(pq) − CoreDisk(pq)
    Select the connected component with the largest arc boundary
    Sample Steiner points on this component
    Recompute the Delaunay triangulation of the modified point set
```

Traditional Delaunay refinement algorithms and implementations consider the triangulation as a list of tetrahedra, and maintain the list of bad tetrahedra in a priority queue. We can transform Algorithm 1 to Algorithm 2 to make

our packing strategy immediately compatible with the existing Delaunay refinement framework. The two algorithms are equivalent as a consequence of Theorem 2.

Algorithm 2

```
Compute the Delaunay triangulation of the input
while ∃ a bad tetrahedron pqrs with shortest edge pq
    Compute the connected components of Vor(pq) − CoreDisk(pq)
    Select the connected component with the largest arc boundary
    Sample Steiner points on this component
    Recompute the Delaunay triangulation of the modified point set
```

4.1 Connected Components

Let pq be an edge incident to a bad tetrahedron. Connected components of $Vor(pq) - CoreDisk(pq)$ can be computed through a linear scan on the polygonal chain boundary of $Vor(pq)$. (See Figure 4 (left).) Each connected component can be characterized by the intersection points of the boundary of $Vor(pq)$ and the boundary of $CoreDisk(pq)$. In our notation, we use ∂ to denote the boundary of a region. If $\partial Vor(pq) \cap \partial CoreDisk(pq) = \varnothing$, then there is one connected component. Unless $Vor(pq) \cap CoreDisk(pq) = \varnothing$, this component is a polygonal region with a hole, whose outer boundary is $\partial Vor(pq)$ and interior boundary is $\partial CoreDisk(pq)$. One can detect whether $Vor(pq) \cap CoreDisk(pq) = \varnothing$. In practice this case occurs rarely, thanks to the strategy we implement for prioritizing the handling of bad tetrahedra. If $\partial Vor(pq) \cap \partial CoreDisk(pq) \neq \varnothing$, then each connected component can be represented by two intersection points a and b, and a polygonal sub-chain $v_1, v_2, \ldots, v_k$ of $\partial Vor(pq)$. (See Figure 4.)

We call the angle of the arc that bounds a connected component the *span angle* of the connected component. The span angle is a real number in the interval $(0, 2\pi]$. The span angle is 2π, either when $\partial Vor(pq) \cap \partial CoreDisk(pq) = \varnothing$ or when $a = b$.

4.2 Sampling the Connected Components

Making sure that the sampling is sufficiently sparse is important for the termination guarantee of the Delaunay refinement algorithm. The distance between the sampling points should be at least $|pq|$. On the other hand, it is desirable to have a dense enough sampling so that the new Voronoi dual of pq is contained inside the core. Here we sketch a strategy that computes the sampling points for a given connected component specified by its spanning angle γ.

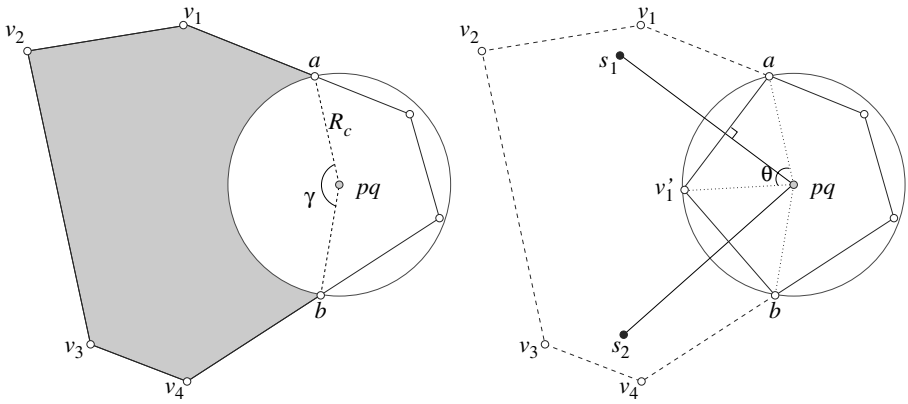

Fig. 4. An edge pq whose Voronoi CoreDisk difference consists of a single connected component is shown in (left). The span angle of the component is γ. A two sampling is shown in (right) for a sampling angle of $\gamma/2$. After the insertion of sample points s_1 and s_2, the Voronoi dual of pq, $Vor(pq)$, is modified to include new vertices a, b, and v_1'.

The sampling angle

We choose an angle called the *sampling angle*, denoted by θ. If $\gamma < \pi/2$ then $\theta = \gamma$ and we use only one sample point. Otherwise, we use $l = \lfloor 4\gamma/\pi \rfloor$ sample points. In this case we subdivide the arc ab of the connected component into equal pieces of angle $\theta = \gamma/l$. So, whenever $\gamma \geq \pi/4$, we have $\theta \in [\pi/4, \pi/2)$.

Location of the sampling points

The number of points and the sampling angle determines the exact location of where we expect the new Voronoi vertices of $Vor(pq)$. These will be $a, v_1', \ldots, v_{l-1}', b$, where l is a positive integer. See Figure 4 (right). For each expected Voronoi edge, there is a unique point in the plane of $Vor(pq)$ which leads to the formation of the edge. We take each such point as the sampling points. The sampling point is on a ray originating from the midpoint of pq and orthogonal to the projected Voronoi edge. It is of distance $(x + \cos(\theta/2)R_c)$ from the midpoint of pq, where $x = \sqrt{(\cos^2(\theta/2)(\beta^2 - 1/4) + 1/4}\ |pq|$. This calculation is given in Lemma 1 in more detail.

Numerical errors and perturbation

Roundoff errors in the calculation of the sample point coordinates might lead to undesired result, e.g., the projected Voronoi vertices might fall slightly outside the $CoreDisk(pq)$. To avoid this we perturb the sample point coordinates toward the midpoint of pq.

Sampling points outside the connected components

It is also possible that sample points might land outside $Vor(pq)$. To prevent inserting points outside the $Vor(pq)$, we compute the intersection e of the ray ms_i and the $\partial Vor(pq)$. If e is far enough from the $CoreDisk(pq)$, we can stil use this sample point together with the other sample points and guarantee that distance in between the sampling points is sufficiently large. Otherwise, we choose to use one sample point at a time.

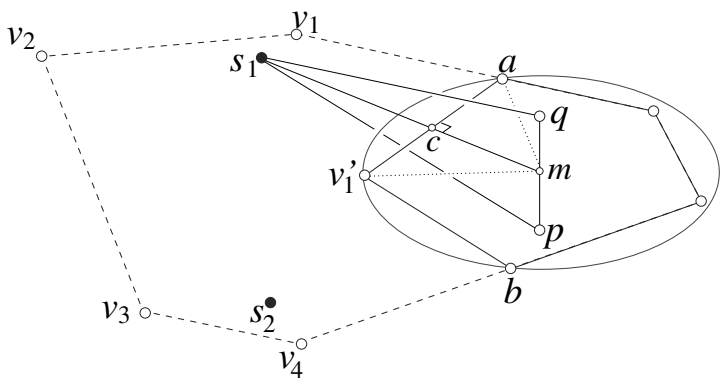

Fig. 5. The exact locations of the sampling points depend on the span angle sampling angle and the radius-edge ratio constraint β.

5 Analysis

Lemma 1. *Let $s_1, \ldots, s_l$ be the set of sampling points inserted at the same iteration. The distance between each of the sampling points s_i, $i = 1, \ldots, l$, and existing vertices is at least $\beta|pq|$.*

Proof. Each sampling point belongs $Vor(pq)$. The nearest neighbours of a point in this region are p and q. Moreover, the sampling points are in $Vor(pq) - CoreDisk(pq)$. Observe that for all $x \in (Vor(pq) - CoreDisk(pq))$, we have $xp > \beta|pq|$ Hence, $|ps_i| > \beta|pq|$ for every sampling point s_i.

We can indeed specify the distance $|ps_i|$ in terms of sampling angle θ, and β. Recall that $R_c = \sqrt{\beta^2 - 1/4}|pq|$. Let x be the distance of p to the new Voronoi edge dual to pqs_i. See Figure 5. By the Pythagoras Theorem,

$$x = \sqrt{(\cos^2(\theta/2)(\beta^2 - 1/4) + 1/4}\ |pq|.$$

Then, applying the same theorem once more, we get

$$ps_i = \sqrt{(x + \cos(\theta/2)R_c)^2 + |pq|^2/4}.$$

It is easy to verify that this value is greater than $|pq|$ for sufficiently large β and θ values. □

Lemma 2. *Let $s_1, \ldots, s_l$ be the set of sampling points inserted at the same iteration where $l \geq 2$. The distance between any two sampling points s_i, and s_j, where $i, j \in \{1, \ldots, l\}$ and $i \neq j$, is at least $\beta|pq|$.*

Proof. The distance between any two sampling points is at least $s_i s_j = 2\sin(\theta/2)(x + \cos(\theta/2)R_c)$. (Note this is the case when either $j = i + 1$, or $i = 1$ and $j = l$.) For $\theta = [\pi/4, \pi/2)$ and $\beta > 2$, it is easy to verify that $s_i s_j > \beta|pq|$.

Integrating the above Lemmas with the Delaunay refinement analysis framework [10, 30] we conclude with the following theorem.

Theorem 3. *The proposed Delaunay refinement algorithm outlined in Algorithm 2 terminates with a guaranteed quality tetrahedral mesh.*

6 Experiments

Our implementation of the proposed algorithm relies on a beta version of the `Pyramid` software (provided by Jonathan Shewchuk). The current version is not very robust and was a limitation conducting comprehensive experimental study. We should emphasize that the results presented in this section are quite preliminary. Nevertheless, we believe the study is representative of the strength of the new packing strategy. We give a comparison with the circumcenter insertion algorithm as implemented in the `Pyramid` software. In our implementation, we handle bad tetrahedra starting with the ones with the shortest edges first.

Data Sets

We ran our experiments on various data sets including the following:

1. *Helix* data set consists of 1008 points forming a double helix. The two helices are very close to each other forming small features. See this model in Figure 7.
2. *Cube1* consists of ten points two of which are located unit distance from each other and at the center of a cube of side length 10^5 units. See this model in Figure 6.
3. *Cube8* is a cube that has eight small features, each one is close to a corner of the cube. See this model in Figure 6.
4. *Random Points* consists of $10,000$ points spread uniformly at random inside a cube.

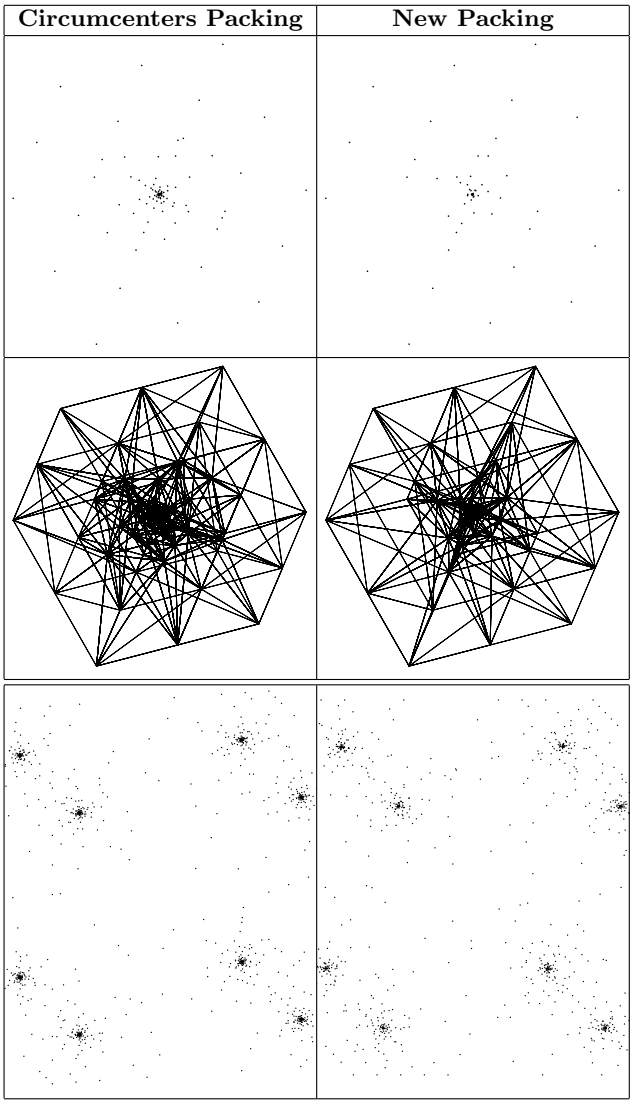

Fig. 6. Output mesh vertex set of the previous and the new Delaunay refinement algorithm for the Cube1 (top row) and Cube8 (bottom row) data sets. Delaunay triangulation is also shown for the Cube1 data set (middle row). See Table 1 for the statistics.

5. *Ellipsoid* consists of 10,000 points spread uniformly at random on an ellipsoid which is inside a cube formed by 8 additional points. See this data set in Figure 7.

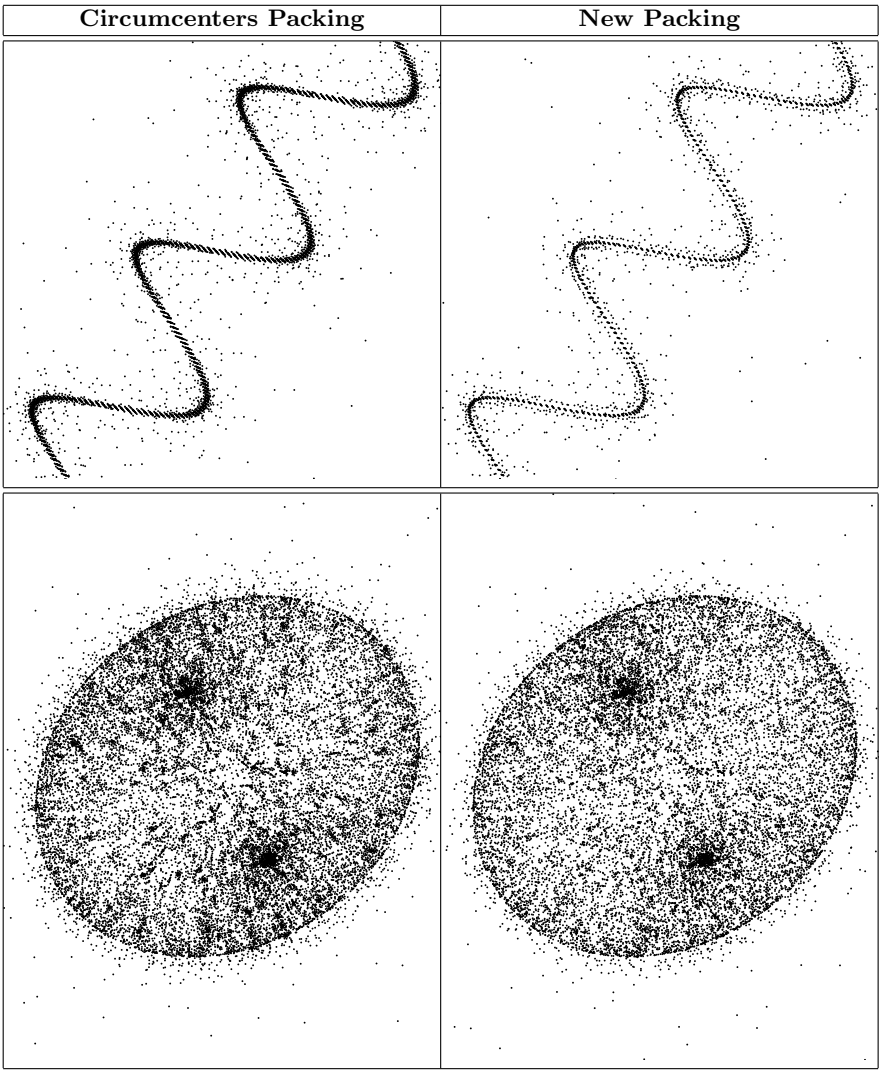

Fig. 7. Output of the previous and the new Delaunay refinement algorithm for the Helix (zoomed in) and the Ellipsoid data sets. The radius-edge ratio constraint in all triangulations is 4. See Table 1 for the statistics.

Table 1 presents a summary of our experiments on two data sets. The first block of four columns provide information about the input data sets. The fifth column shows the imposed radius-edge ratio constraint β. Columns six and seven list the output size of the circumcenter insertion algorithm in the number of vertices and in the number of tetrahedra, respectviely. Columns eight and nine list these quantities for the new algorithm. The last column

Data Set				Quality	CC algorithm		New algorithm		Imp.
name	v	e	f	R/l	v	t	v	t	%
Cube1	12	12	6	1.5	596	3,420	570	3,280	4.3
				2	265	1,440	220	1,196	16.9
				4	153	730	79	330	48.3
				8	133	607	51	195	61.6
Cube8	32	12	6	1.5	2,164	12,016	1,952	10,803	9.7
				2	1,042	5,531	881	4,537	15.4
				4	610	2,836	360	1,605	40.9
				8	483	2,074	233	924	51.7
Ellipsoid	10,008	12	6	1.5	69,075	431,908	60,861	379,718	11.8
				2	39,710	247,031	32,394	200,960	18.4
				4	21,396	131,553	17,256	103,380	19.3
				8	15,202	89,501	13,328	75,238	12.3
Helix	1,008	12	6	1.5	50,527	310,743	38,965	239,330	22.8
				2	26,259	157,163	16,596	99,521	36.7
				4	12,763	71,812	5,938	37,738	53.4
				8	8,727	49,998	3,923	37,738	55.0
Random	1,008	12	6	1.5	27,851	176,846	25,884	164,301	7.0
				2	17,743	113,743	16,572	106,420	6.5
				4	11,463	75,257	11,118	73,484	3.0
				8	10,377	68,595	10,326	68,202	0.4

Table 1. The number of vertices (v) and the number of tetrahedra (t) in the output is listed for the proposed algorithm in comparison to the circumcenter insertion (CC) algorithm. The last column reports the percentage improvement in the number of output vertices.

lists the percentage improvement in the number of output vertex size. In some cases, the percentage improvement listed seems insignificant. Note that in such cases, the input size is very large compared to the Steiner vertices inserted by each of the algorithm. If we report the improvements in the number of Steiner points, the percentages listed would be much larger. For instance, the last entry would be 13.7% instead of 0.4%. We get similar improvements for other data sets that are not listed here.

7 Discussions

We proposed a new Delaunay refinement algorithm for computing quality-guaranteed three dimensional tetrahedral meshes. It differs from previous algorithms on several fronts:

- Unlike the traditional algorithms, we handle potentially more than one bad tetrahedra at a time.
- We insert multiple Steiner points at each iteration. While this seems to be a good match within a parallel Delaunay refinement framework such

as [38], our sequential strategy should not be confused with a parallel algorithm.
- The point set of the output is not only well-speced but also significantly sparse compared to the output of the previous algorithms.

Further research is needed to fully utilize the benefits of the approach proposed here. We plan to complete a robust implementation of the three dimensional meshing software and run more comprehensive experiments. We would like to explore the potential benefits of our approach on computing sliver-free meshes.

Acknowledgements

Thanks to Jonathan Shewchuk for providing the `Pyramid` software.

References

1. M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. *J. Comp. System Sciences* 48:384–409, 1994.
2. B. Chazelle, Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm, *SIAM Journal on Computing* 13: 488–507, 1984.
3. S.-W. Cheng and T. K. Dey. Quality meshing with weighted Delaunay refinement. *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, 137–146, 2002.
4. S.-W. Cheng, T.K. Dey, E.A. Ramos, and T. Ray. Quality Meshing of Polyhedra with Small Angles. *Int. J. Computational Geometry & Applications* 15(4): 421–461, 2005.
5. S.-W. Cheng, T.K. Dey, H. Edelsbrunner,M.A. Facello, and S.-H. Teng. Sliver exudation. *Proc. 15th ACM Symp. Comp. Geometry*, 1–13, 1999.
6. S.-W. Cheng and Sheung-Hung Poon. Three-Dimensional Delaunay Mesh Generation. 419-456 *Discrete & Computational Geometry*, 36(3): 419–456, 2006.
7. L.P. Chew. Guaranteed-quality triangular meshes. TR-89-983, Cornell Univ., 1989.
8. D. Cohen-Steiner, E. C. de Verdiére, and M. Yvinec. Conforming Delaunay triangulations in 3d. *Proc. 18th ACM Symp. Comp. Geometry*, 199–208, 2002.
9. T.K. Dey, C. L. Bajaj, and K. Sugihara. On good triangulations in three dimensions. *Int. J. Computational Geometry & Applications* 2(1):75–95, 1992.
10. H. Edelsbrunner. *Geometry and Topology for Mesh Generation.* Cambridge Univ. Press, 2001.
11. H. Edelsbrunner and D. Guoy. Sink insertion for mesh improvement. *Proc. 17th ACM Symp. Comp. Geometry*, 115–123, 2001.
12. H. Edelsbrunner, X. Li, G.L. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Üngör, and N. Walkington. Smoothing and cleaning up slivers. *Proc. 32nd ACM Symp. on Theory of Computing*, 273–277, 2000.
13. H. Erten and A. Üngör. Delaunay Refinement with Locally Optimal Steiner Points. *(to appear) In Proc. EUROGRAPHICS Symposium on Geometry Processing, Barcelona, Spain* July 2007.

14. S. Har-Peled and A. Üngör. A time-optimal Delaunay refinement algorithm in two dimensions. *Proc. ACM Symposium on Computational Geometry*, 228–236, 2005.
15. B. Hudson, G.L. Miller, and T. Phillips. Sparse Voronoi Refinement. *Proc. 15th Int. Meshing Roundtable*, 339–356, 2006.
16. F. Labelle. Sliver Removal by Lattice Refinement. *Proc. ACM Symposium on Computational Geometry*, 347–356, 2006.
17. F. Labelle and J.R. Shewchuk. Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles *Proc. ACM SIGGRAPH* 2007.
18. X.-Y. Li, S.-H. Teng, and A. Üngör. Biting: Advancing front meets sphere packing. *Int. J. Numer. Meth. Eng.* 49:61–81, 2000.
19. X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshed in 3D. *Proc. ACM-SIAM Symp. on Discrete Algorithms*, 28–37, 2001.
20. R. Löhner. Progress in grid generation via the advancing front technique. *Engineering with Computers* 12:186–210, 1996.
21. G.L. Miller. A time efficient Delaunay refinement algorithm. *Proc. ACM-SIAM Symp. on Disc. Algorithms*, 400–409, 2004.
22. G.L. Miller, S. Pav, and N. Walkington. When and why Ruppert's algorithm works. *Proc. 12th Int. Meshing Roundtable*, 91–102, 2003.
23. G.L. Miller, S. Pav, and N. Walkington. Fully Incremental 3D Delaunay Refinement Mesh Generation. *Proc. 11th Int. Meshing Roundtable*, 75–86, 2002.
24. G.L. Miller, D. Talmor, S.-H. Teng, N. Walkington, and H. Wang. Control volume meshes using sphere packing: Generation, refinement, and coarsening. *Proc. 5th Int. Meshing Roundtable*, 47–61, 1996.
25. S. Mitchell and S. Vavasis. Quality mesh generation in three dimensions. *Proc. 8th ACM Symp. Comp. Geometry*, 212–221, 1992.
26. M. Murphy, D. M. Mount, and C. W. Gable. A point-placement strategy for conforming Delaunay tetrahedralization. *Proc. 11th ACM-SIAM Symp. on Discrete Algorithms*, 67–74, 2000.
27. M.S. Paterson and F.F. Yao, Binary partitions with applications to hidden-surface removal and solid modeling, *Proc. 5th ACM Symp. Computational Geometry*, 23–32, 1989.
28. J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. *Proc. 4th ACM-SIAM Symp. on Disc. Algorithms*, 83–92, 1993.
29. E. Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder, *Math. Annalen*, 98: 309–312, 1928.
30. J.R. Shewchuk. *Delaunay Refinement Mesh Generation.* Ph.D. thesis, Carnegie Mellon University, 1997.
31. J.R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. *Proc. 14th Annual ACM Symposium on Computational Geometry*, 86–95, 1998.
32. J.R. Shewchuk. Mesh generation for domains with small angles. *Proc. 16th ACM Symposium on Computational Geometry*, 1–10, 2000.
33. J.R. Shewchuk. Sweep algorithms for constructing higher-dimensional constrained Delaunay triangulations. *Proc. 16th ACM Symposium on Computational Geometry*, 350–359, 2000.
34. J.R. Shewchuk. Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery. *Proc. 11th Int. Meshing Roundtable*, 193–204, 2002.
35. J.R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications* 22(1–3):21–74, 2002.

36. K. Shimada. *Physically-based Mesh Generation: Automated Triangulation of Surfaces and Volumes via Bubble Packing.* Ph.D. thesis, MIT, 1993.
37. D.A. Spielman, S.-H. Teng, and A. Üngör. Parallel Delaunay refinement: Algorithms and analyses. *Proc.* 11*th Int. Meshing Roundtable*, 205–217, 2002.
38. D.A. Spielman, S.-H. Teng, and A. Üngör. Parallel Delaunay refinement: Algorithms and analyses. *Int. J. Comput. Geometry Appl.* 17(1): 1-30 (2007)
39. H. Si. On Refinement of Constrained Delaunay Tetrahedralizations. *Proc.* 11*th Int. Meshing Roundtable*, 2006.
40. D.A. Spielman, S.-H. Teng, and A. Üngör. Parallel Delaunay refinement with off-centers. *Proc. EUROPAR*, LNCS 3149, 812–819, 2004.
41. D. Talmor. *Well-Spaced Points for Numerical Methods.* Ph.D. thesis, Carnegie Mellon University, 1997.
42. A. Üngör. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. *Proc. LATIN*, LNCS 2976, 152–161, 2004.
43. A. Üngör. Quality meshes made smaller. *Proc. 20th European Workshop on Computational Geometry*, 5–8, 2005.