Tetrahedral Meshing via Maximal Poisson-disk Sampling

Jianwei Guo[a], Dong-Ming Yan[a,*], Li Chen[b], Xiaopeng Zhang[a], Oliver Deussen[c,d], Peter Wonka[e]

[a]*National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*
[b]*School of Software, Tsinghua University, Beijing 100084, China*
[c]*University of Konstanz, Konstanz 78457, Germany*
[d]*Shenzhen Key Lab of Visual Computing and Visual Analytics / SIAT, Shenzhen 518055, China*
[e]*Visual Computing Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia*

Abstract

In this paper, we propose a simple yet effective method to generate 3D-conforming tetrahedral meshes from closed 2-manifold surfaces. Our approach is inspired by recent work on maximal Poisson-disk sampling (MPS), which can generate well-distributed point sets in arbitrary domains. We first perform MPS on the boundary of the input domain, we then sample the interior of the domain, and we finally extract the tetrahedral mesh from the samples by using 3D Delaunay or regular triangulation for uniform or adaptive sampling, respectively. We also propose an efficient optimization strategy to protect the domain boundaries and to remove slivers to improve the meshing quality. We present various experimental results to illustrate the efficiency and the robustness of our proposed approach. We demonstrate that the performance and quality (e.g., minimal dihedral angle) of our approach are superior to current state-of-the-art optimization-based approaches.

Keywords: Tetrahedral Mesh Generation, Maximal Poisson-disk Sampling, Sliver Removal, Mesh Optimization

1. Introduction

Mesh generation aims to approximate a given closed domain with simple discrete elements (e.g., triangle/quad in 2D and tetrahedron/pyramid/prism/hexahedron in 3D). It has numerous applications ranging from engineering to scientific research, e.g., simulation of mechanical parts or architecture, medical and biological data analysis, geographical science, computational fluid dynamics, and animation in computer graphics [1, 2]. Here we focus on 3D tetrahedral mesh generation.

Although many robust commercial software (e.g., Ansys) and open source packages exist for mesh generation (e.g., TetGen [3], CGALmesh [4], DelPSC [5], Gmsh [6], Qhull [7], etc.), research on tetrahedral meshing remains active because no available approach is able to fulfill all conflicting quality requirements of various applications.

There are many criteria to evaluate the quality of a tetrahedral mesh, e.g., the approximation quality, the gradation, the dihedral angle, and the radius ratio of a single tetrahedron. These criteria are typically in conflict with each other and are difficult to satisfy simultaneously. Among all the quality criteria, the dihedral angle is the most important for simulation because it is directly related to the condition number of the stiffness matrix. A single poorly shaped tetrahedron with near zero volume (called a *sliver*) can destroy the whole simulation [8]. Nevertheless, recent approaches still face the difficulty of completely removing slivers, even with careful treatment of the energy functions and with post-processing, such as simulated annealing or sliver exudation [9, 10, 11].

The main contribution of this paper is to propose a very simple algorithm that improves 3D tetrahedral mesh generation compared with existing algorithms that optimize complex objective functions. Our work is inspired by recent work in maximal Poisson-disk sampling (MPS) [12, 13], which generates uniformly and randomly distributed point

*Corresponding authors.
Email addresses: jianwei.guo@nlpr.ia.ac.cn (Jianwei Guo), yandongming@gmail.com (Dong-Ming Yan), chenlee@tsinghua.edu.cn (Li Chen), xpzhang@ia.ac.cn (Xiaopeng Zhang), Oliver.Deussen@uni-konstanz.de (Oliver Deussen), pwonka@gmail.com (Peter Wonka)

sets while exhibiting blue-noise properties. These extracted meshes from sampled point sets have very good geometric properties, e.g., angle bounds, edge-length bounds, etc. Our approach is a combination of randomized sampling and several effective optimization strategies that improve the lower/upper bounds of dihedral angles. Tetrahedra near the boundary surface are treated carefully such that the quality of the resulting tetrahedral mesh is provably protected. Our method works better than other greedy approaches for generating samples (e.g., Delaunay insertion). It is more efficient than optimization-based algorithms while at the same time it can improve the quality of the smallest dihedral angles.

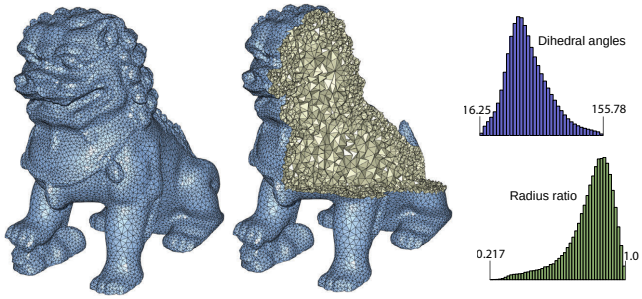

Figure 1: An example of our tetrahedral meshing result on the Lion model (input $100K$ triangles). Left and middle are the meshing result ($55K$ vertices and $300K$ cells) and a cutaway view, respectively. Right are the dihedral angle and radius ratio distributions.

2. Related Work

Delaunay mesh generation. There are three main categories of tetrahedral meshing algorithms: (i) advancing-front-based approaches [14, 15, 16], (ii) octree-based methods [17, 18, 19], and (iii) Delaunay-based meshing algorithms [20]. A complete review of these algorithms is beyond the scope of this paper. We refer the reader to a survey paper [21] and textbooks [22, 23, 24] (and references therein) for a more comprehensive introduction to tetrahedral meshing algorithms. In the following, we focus on Delaunay-based meshing algorithms, which are most closely related to our work.

Delaunay-based tetrahedral meshing algorithms have been shown to be the most successful in recent years [20]. From an algorithmic perspective, Delaunay meshing techniques can be further classified into two major types: Delaunay insertion/refinement-based methods [25, 4, 3] and optimization-based (or variational) approaches [9, 10, 26, 27, 28, 29, 11].

Delaunay insertion/refinement-based approaches usually start with an initial mesh and gradually insert new vertices at the center of a circumscribed sphere of the existing tetrahedra with the worst quality. The algorithm stops once all the user-specified criteria are satisfied, e.g., smallest dihedral angle, local edge length, approximation quality of the domain boundary, and so on. The framework of TetGen [3] uses a series of such approaches, including incremental Delaunay algorithms for inserting vertices, constrained Delaunay algorithms for inserting constraints and a Delaunay refinement algorithm for quality mesh generation. In summary, this type of meshing technique can usually provide theoretical guarantees of the angle bound, element size and approximation error. On the other hand, it is difficult to explicitly control the desired number of vertices. The Delaunay insertion method usually inserts more vertices than necessary to reach the desired quality criteria.

The optimization-based approaches start with an initial set of randomly sampled vertices, where the number of vertices is usually specified by the user. Then, different types of energy functions are proposed to characterize the desired meshing quality. A well-known optimization-based approach is *Centroidal Voronoi Tessellation* (CVT) [30] mesh generation [31, 29]. CVT-based algorithms optimize the compactness of the dual Voronoi cells instead of the shape of the tetrahedra, which does not suppress any poorly shaped elements. Alliez et al. [9] generalize *Optimal Delaunay Triangulation* (ODT) [32] for 2D mesh smoothing to 3D tetrahedral meshing. They propose to minimize the global mesh-dependent energy to update both vertex positions and connectivity. The work of Tournois et al. [10], called natural ODT (NODT), further extends the ODT energy to the domain boundaries. This extension ensures

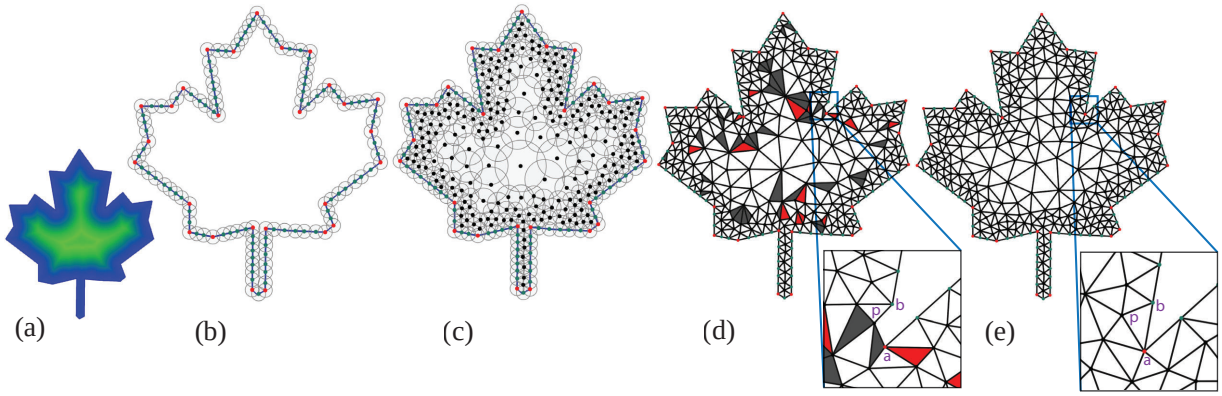

Figure 2: A 2D illustration of the MPS framework. (a) Input boundary and the sizing field defined in this domain, where warmer colors indicate a larger radius, and cooler colors indicate a smaller radius. (b) Boundary sampling and optimization with disks. The red points indicate sharp corners, while green points are the optimized samples. (c) Interior sampling (black points). (d) Regular triangulation. Triangles with the minimal angle smaller than 30° are shown in dark grey and triangles with undesirable angles by the user input are shown in red. The zoom-in view shows the non-conformal point, **p**, which causes the boundary part between **a** and **b** to disappear. (e) Samples and remeshing after optimization.

consistency of the energy function on the boundary and inside the domain. The number of badly shaped elements is reduced near the domain boundary. Recent work of Chen et al. [32] proposes an extended formulation for ODT that is able to generate graded meshes efficiently using quasi-Newton solvers. This type of algorithm generates higher-quality meshes in practice, but they are more time consuming and post-processing for mesh improvement is necessary to eliminate badly shaped elements [33, 34]. In addition, Jiao et al. [35] present a variational smoothing approach for optimizing surface and volume meshes simultaneously. They define energy functions based on conformal and isometric mappings between actual elements with ideal reference elements and develop a simple algorithm to minimize the energies.

From a boundary approximation perspective, Delaunay meshing can be classified into constrained meshing or conformal meshing. Our approach falls into the latter category, i.e., instead of interpolating the boundary, we approximate the boundary by resampling, which allows for higher tetrahedron quality at the boundary, especially when the quality of the input surface is very poor. Here, we present a simple method based on randomized sampling that is able to generate high-quality meshes that are comparable with state-of-the-art techniques, while being more efficient than variational approaches.

Maximal Poisson-disk sampling. Poisson-disk sampling techniques have been extensively studied in past decades [36], but most previous work cannot achieve maximality of the sampling. In 2006, Jones [37] first proposed an unbiased algorithm using the Voronoi diagram for gap detection/filling. Later, Ebeida et al. [38, 39] improved the efficiency of MPS by using a uniform grid for acceleration. Meanwhile, they showed that MPS point sets are able to generate high-quality triangulations [40] and Voronoi meshes [12] for the purpose of simulation. More recently, Yan and Wonka [13] and Guo et al. [41] extended MPS to mesh surfaces and adaptive sampling. They demonstrated that MPS can generate high-quality surface remeshing as well. In this paper, we further study the use of MPS for tetrahedral mesh generation. To the best of our knowledge, this is the first application of MPS to volumetric meshing.

3. Preliminaries and Overview

In this section, we first introduce the concept of maximal Poisson-disk sampling and then briefly describe the MPS framework for tetrahedral mesh generation.

3.1. Maximal Poisson-disk Sampling

Suppose $\mathbf{X} = \{(\mathbf{x}_i, r_i)\}$ is a Poisson-disk distribution in a compact domain Ω, where $\mathbf{x}_i$ is the position of the i^{th} point and r_i is the associated radius of $\mathbf{x}_i$. A *maximal Poisson-disk sampling* distribution should satisfy the following three properties: 1) *the minimal distance property* requires that any two disks are at least a minimum distance apart, i.e.,

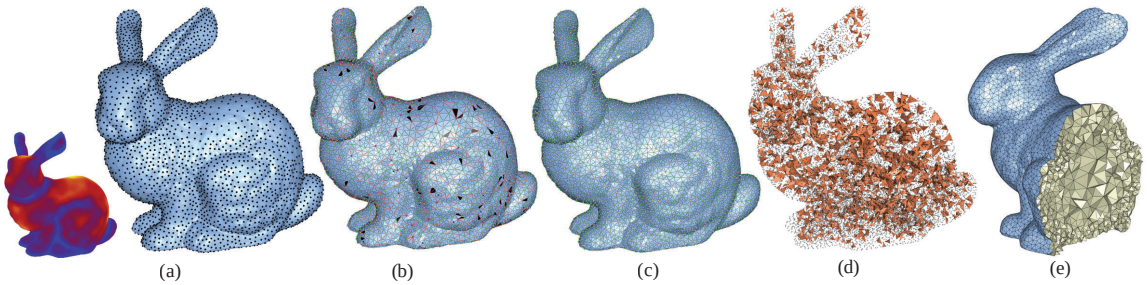

Figure 3: Overview of our algorithm. Given the input boundary mesh, we first generate surface samples (a) and extract a surface mesh (b). We optimize the boundary remeshing to improve its quality (c). After the boundary optimization, we perform volume sampling while showing slivers (d). Then we optimize the dihedral angles to remove these slivers. (e) is a cutaway view of the final mesh.

$\forall \mathbf{x}_i, \mathbf{x}_j \in P, \|\mathbf{x}_i - \mathbf{x}_j\| \geq \max(r_i, r_j)$; 2) *the bias-free property* requires that each point in the domain has a probability that is proportional to the sizing at this point to receive a sampling point; and 3) *the maximal sampling property* requires that the union of the disks covers the entire sampling domain, i.e., $\bigcup(\mathbf{x}_i, r_i) \supseteq \Omega$. Furthermore, if the radius, r_i, for all disks is a constant, then the sampling is classic uniform sampling [36]. Otherwise, it is adaptive sampling, where each r_i is defined with respect to an underlying sizing function, $\mu(\mathbf{x})$, defined over the sampling domain.

3.2. MPS Framework

Two recent and efficient MPS frameworks [39, 41] are useful for tetrahedral meshing. The 2D example in Figure 2 illustrates how to apply an MPS framework for mesh generation. There are four main steps in our sampling/meshing pipeline as follows:

1. Detect sharp corners and generate MPS on the boundary of the input domain. To avoid introducing vertices arbitrarily close to the domain boundary, we apply edge-length optimization to ensure that the boundary disks are deeply intersected [5].
2. Keep the boundary samples fixed and then sample the interior of the domain to achieve maximality.
3. Triangulate the sampled points using Delaunay or regular triangulation for uniform or adaptive sampling, respectively, and extract a mesh from the triangulation.
4. To improve the meshing quality, we perform optimization operations to remove bad elements and perform resampling/retriangulation locally. Multiple criteria are used to perform the mesh optimization.

4. Tetrahedral Meshing

Here, we seek to generate tetrahedral meshes from piecewise-smooth surfaces based on MPS. The details of each step are provided in the following pseudo-code (Algorithm 1).

Algorithm 1: Tetrahedral meshing based on MPS

Input: 3D domain, Ω, bounded by a surface $\mathcal{M}$, the minimal sampling radius, r_{min}
Output: tetrahedral mesh T

1: Build a uniform grid, G, to voxelize Ω and design a sizing field inside the domain
2: Generate samples on surface $\mathcal{M}$, and perform boundary optimization
3: Sample the interior of Ω while conforming the boundary
4: Optimize bad dihedral angles

The input of our algorithm is a 3D domain, Ω, bounded by a 2-manifold watertight triangular mesh, $\mathcal{M} = \{f_i\}_{i=1}^m$, where f_i is a triangle facet of the boundary $\mathcal{M}$. Similar to the 2D case, the core of our tetrahedral meshing algorithm consists of MPS in the input domain and a series of mesh optimization operations for improving the meshing quality.

However, there are still some difficulties that make tetrahedral meshing more difficult than its 2D counterpart. First, MPS always leads to high-quality triangulations in the 2D domain. In 3D, even well-spaced vertices could create degenerate 3D elements (e.g., slivers). Besides, dealing with boundary conformity is also more difficult in 3D. In the 2D case, there always exists a 2D triangulation that conforms to a set of non-intersecting line segment constraints, but this is no longer true in 3D [9]. Figure 3 shows the pipeline of the proposed framework in 3D.

4.1. Initialization

Initially, a minimal sampling radius, r_{min}, is specified by the user. In addition, a uniform grid, G, is built to voxelize the 3D domain, Ω. The cell size of G equals $\frac{r_{min}}{\sqrt{3}}$, which guarantees that each grid cell can contain at most one sample in the context of Poisson-disk sampling. The grid cells are classified into three types: boundary cells (intersecting the boundary surface), interior cells, and exterior cells. This grid is useful in two ways. First, it is used to accelerate the sampling operations during the MPS process. In this case, each grid cell stores a flag (initialized as 'true') to indicate whether or not this cell is valid. Here, a valid cell means that it is not fully covered by a sample. Further, we utilize the grid G to design a sizing field in the 3D domain for graded meshing, which we describe later.

4.2. Surface Sampling

At the beginning, we perform MPS only on the boundary surface. The purpose of this step is to approximate the domain boundary by a new surface mesh with better quality and to make the final tetrahedral mesh conform with it. We apply a surface sampling strategy that includes two stages: dart throwing with grid and gap filling. Dart throwing continually chooses a random triangle facet, f_i, from $\mathcal{M}$ and generates a random sample in this facet. To make it unbiased, the probability of selecting the triangle, f_i, is proportional to its area, A_i. Then we reject or accept this new sample based on whether or not it conflicts with any previously accepted samples. The conflict check is fast because we need to check only the sample's local neighboring boundary cells (in the uniform case, it is a $5 \times 5 \times 5$ template of cells). Once a sample is accepted, the flags of the corresponding cells fully covered by this sample are marked as "false". After observing k (e.g., $k = 300$) consecutive rejections, we switch to the gap-filling stage. In this stage, we clip the triangle facets in $\mathcal{M}$ by the spheres of the samples and collect the gap regions, as described in [13]. Then we rethrow darts only in these gaps to generate new samples. This process of gap computation and dart throwing repeats until there are no gaps. The result of this step is shown in Figure 3(b).

In addition, the input boundary may include sharp edges and feature vertices, especially in mechanical models. To handle these features, we implement a feature-aware sampling step before the surface sampling. Specifically, the features are either provided by the user as 1D curved skeletons or automatically detected by our algorithm. The feature vertices (e.g., corners, cusps) are directly added to the sampling set. Then, we perform 1D MPS to get samples on the creases that are composed by a chain of sharp edges. These feature samples remain fixed in all later steps.

Boundary optimization. Once the surface sampling is done, we extract the surface mesh using the algorithm described in [42]. The extracted mesh might contain undesirable elements, such as irregular vertices whose degrees are less than 5 or larger than 7, and elongated triangles with long edges and small angles (less than 30°). All such elements will result in interior samples that are too close to the boundary, which further result in the tetrahedra with bad shapes. To make the surface remeshing as compact as possible, we propose an approach that interleaves between the valence, angle and edge length optimization. Take the edge length optimization as an example. We iteratively remove the vertices of the long edges whose length is larger than $\frac{\sqrt{3}}{2}(r_1 + r_2)$, where r_1 and r_2 are the sampling radii of the vertices. Then we resample the domain boundary until there are no long edges or until the maximal iteration number is reached. The valence and angle optimization are performed in the same way as in [13, 41]. Figure 3(c) shows the result of boundary optimization.

4.3. Volume Sampling

Next, we generate samples in the interior of the domain. However, if we sample the volume directly after surface sampling, it will generate many slivers near the boundary regions that are difficult to remove in the later steps. Besides, it is difficult to maintain the conformation of the boundary of the final tetrahedral mesh (Figure 4 (c) shows a non-conforming case), since we use Delaunay triangulation instead of constrained Delaunay triangulation to extract the final mesh. To protect the input boundary, we relax the maximal property of MPS near the boundary regions by adding some "virtual samples". Before introducing our approach in 3D, we first use a simple 2D illustration to make

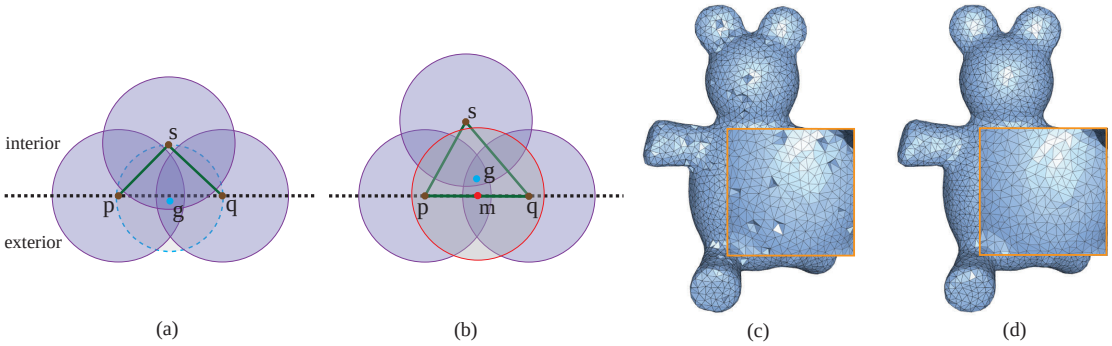

Figure 4: (a) and (b) illustrate our boundary-protection approach. (c) and (d) show the comparison of tetrahedral meshing results without and with our boundary-protection approach, respectively.

the concept of "virtual sample" clear. As shown in Figure 4 (a), the black dashed line is one edge of the input boundary, points **p** and **q** are two boundary samples and they are locally maximal (there is no gap between **p** and **q**). If a new sample is generated at point **s**, a triangle $\triangle$**spq** will be extracted. However, since the circumcenter, **g**, of $\triangle$**spq** is outside the domain, this triangle will be discarded, resulting in the boundary part of **pq** not being maintained in the final mesh. In our approach, we insert a "virtual sample", **m**, at the middle point of **pq**. Then, a newly generated sample, **s**, should not conflict with **m** (Figure 4 (b)). In this case, the circumcenter, **g**, of $\triangle$**spq** will be inside the domain, and the edge, **pq**, is kept. To generalize this approach to a 3D volume, we compute the restricted Voronoi diagram (RVD) [42] of the surface samples and put a "virtual sample" at each restricted Voronoi vertex. These "virtual samples" will not be added to the sample set. They are only used to mark the grid cells as invalid, which are fully covered by themselves. Therefore, we cannot generate samples in these cells any more. Figure 4 (c) and Figure 4 (d) show a comparison of meshing results without and with our boundary protection approach.

To sample the interior of the domain, we utilize the uniform grid, G, as the implicit data structure for efficient 3D sampling, as inspired by [39]. The sampling process also has two steps. In the first step, we repeatedly choose a random valid grid cell that is not outside of the domain. Then, a random sample is generated in this cell. This new sample is accepted if it is inside the domain and does not conflict with previous samples (including the boundary samples). In the second step, the background grid is refined, and each valid cell is subdivided into eight smaller cells, which are referred to as fragments. We collect the fragments that are not fully covered into a flat array and perform the dart throwing on it. This process repeats until the maximal sampling is achieved. The result of this step is shown in Figure 3(d).

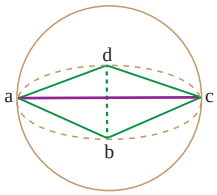

Figure 5: An example of a sliver with 4 near-cocircular vertices.

4.4. Mesh Optimization

Finally, given the sampled point set, we can extract the tetrahedral mesh by using 3D Delaunay triangulation, while discarding the Delaunay tetrahedra whose circumscribed sphere centers are outside the domain. However, the quality of this tetrahedral mesh is usually not sufficient to fulfill all quality requirements. We improve the mesh quality through optimizing the sample locations, both on the boundary surface and the domain volume.

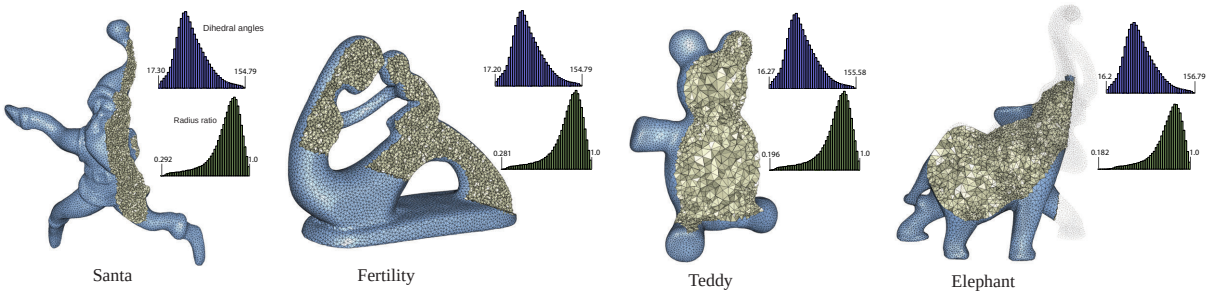

Figure 6: Tetrahedral meshing results of our algorithm. The two images of the left (Santa and Fertility) are uniform meshes, and the two images of the right (Teddy and Elephant) are graded meshes.

Dihedral angle optimization. Although the meshing results based on MPS in 2D and on surfaces exhibit nice properties like the triangle's angle bound $[30°, 120°]$, such properties are not evident in our tetrahedral mesh for dihedral angles. Slivers with very small dihedral angles may exist (see Figure 3(d)). Further, traditional sliver perturbation [10] or the sliver exudation method [33] is not suitable for our case, because it breaks the Poisson-disk sampling criteria or has little effect. To remove slivers, we use a similar optimization framework as above. For each sliver, the two vertices of the longest edge are first removed (one sliver and its longest edge, **ac**, are displayed in Figure 5). Then, we collect the gap fragments caused by removing these vertices and refill them using the method described in Section 4.3. The optimization step terminates when there are no slivers.

4.5. Graded Mesh Generation

To generate a graded 3D mesh (corresponding to adaptive sampling), we design a sizing field in the whole domain that indicates the local sampling radius. A sizing function, $\mu(\mathbf{x})$, is first defined over the boundary surface, which is also used for adaptive boundary sampling. Then, it is further extrapolated to the grid cells in the interior of the domain by using a fast marching construction method [9]. As a result, the elements will be denser on the boundary where high resolution is generally desired, especially in the high-curvature regions. In the interior of the domain, the elements will be coarser and vary in size.

In our approach, we adopt the local feature size (lfs) as the boundary sizing function, $\mu(\mathbf{x})$, but other functions can also be used with the same framework. We also set a maximal sampling radius, $r_{max} = \lambda\, r_{min}$ (in default $\lambda = 8$), to avoid very big radii. The adaptive sampling process is similar to the uniform case, except for some differences in the details. In the dart-throwing stage, we evaluate the radius, r_i, for each newly generated sample, s_i. Then, we perform a conflict check using a subset of $\lceil \frac{2\, r_i}{r_{min}/\sqrt{3}} \rceil^3$ grid cells. In the mesh extraction stage, we adopt regular triangulation instead of Delaunay triangulation, where each sample, s_i, is associated with a weight, $w_i = r_i^2$. Instead of using the traditional Euclidean distance, the distance function used by the regular triangulation is the power distance, represented by $d(\mathbf{s}_i, w_i, \mathbf{s}_j, w_j) = \|\mathbf{s}_i - \mathbf{s}_j\|^2 - w_i - w_j$.

5. Experimental Results

Our algorithm is implemented in C++ using the CGAL library [43] for 3D Delaunay/regular triangulation. The results shown in this paper are obtained on an Intel i7 3.07 GHz desktop with 12 GB memory.

Tetrahedral meshing. To verify the robustness and the validity of the proposed approach, we ran our algorithm on various input domains with arbitrary topologies. Intuitively, our algorithm worked well because Poisson-disk samplers are well distributed, leading to well-shaped tetrahedra. Figures 1 and 6 illustrate the tetrahedral meshing results, as well as the histograms of the dihedral angle and radius ratio distributions. The radius ratio, κ, is defined as $\kappa = \frac{3r_{in}}{r_{cir}}$, where r_{in} and r_{cir} are the inscribed and circumscribed sphere radius of a tetrahedron, respectively. Furthermore, our algorithm can preserve the features of the input domain. Two examples of feature preserving are illustrated in Figure 7.

In the mesh optimization stage, we set the default value of the desired minimal dihedral angle to $16° \sim 18°$. The optimization usually converges within $30 \sim 50$ iterations. Figure 8 shows the convergence behavior of the mesh

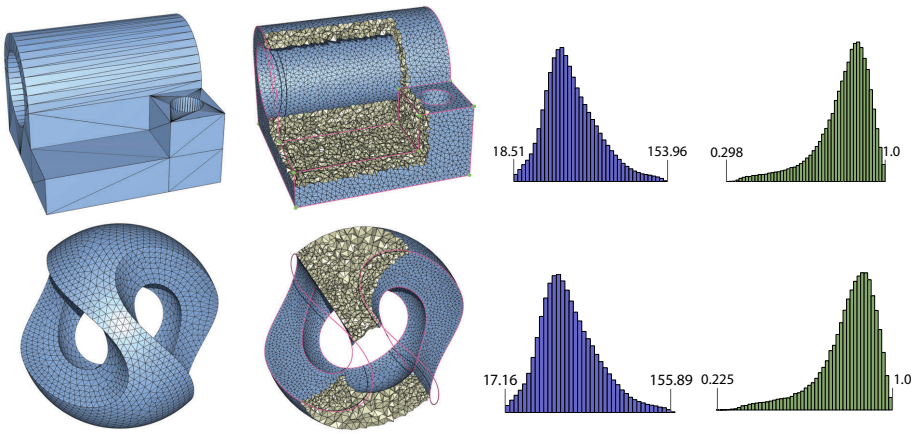

Figure 7: Tetrahedral meshing results on the Joint (uniform) and Sculpt (adaptive) models with sharp features preserved. Left: input boundary surfaces. Middle: cutaway view of the meshing results, while showing the features (green points are the feature vertices and pink lines are the sharp creases). Right: dihedral angle distributions (blue) and radius ratio distributions (green).

optimization process. Though we do not have a formal theoretical proof of the convergence, our algorithm works well in practice with various input domains.

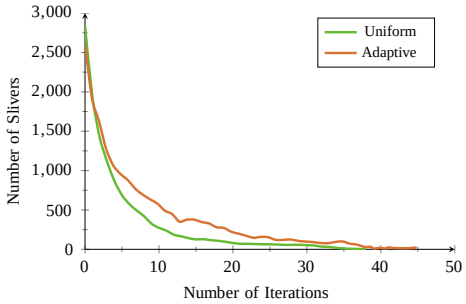

Figure 8: Number of slivers in relation to the number of iterations. The statistics are obtained on the Santa (uniform with $136K$ tetrahedra) and Elephant (adaptive with $133K$ tetrahedra) models.

Quality comparison. Next, we compare our approach with current state-of-the-art techniques, including the interleaved Delaunay refinement and optimization method (NODT) [10], Centroidal Voronoi Tessellation (CVT) [29], revisited optimal Delaunay triangulaiton (RODT) [11] and a Delaunay-insertion-based appraoch – TetGen [3]. Note that for TetGen, we provided as input the boundary of our optimized mesh for fair comparison, while NODT is implemented in CGAL with sliver perturbation.

Tables 1 and 2 list the mesh quality comparison results for uniform and adaptive meshing, respectively. Here θ_{min} and θ_{max} are the minimal and maximal dihedral angles of a tetrahedron, κ_{min} and $\bar{\kappa}$ are the minimal and average radius ratios. d_H is the Hausdorff distance (divided by the diagonal length of the input mesh bounding box) between the input boundary and remeshing result. $\theta_{<10^\circ}$ and $\theta_{<18^\circ}$ are the percentages of dihedral angles smaller than 10° and 18°, respectively. The best results are highlighted using bold font. Both of these tables suggest that our method achieves best min/max dihedral angles, θ, and the minimal radius ratios, κ_{min}. Our $\bar{\kappa}$ is not as good as NODT, CVT, RODT, while our $\theta_{<18^\circ}$ is comparable with NODT method. The reason is that we generate the point set by employing the randomized sampling, while the other three methods tend to generate more regular structures. The histogram distributions of dihedral angles and radius ratios shown in Figures 10 and 11 also confirm this observation. However, randomized meshes are preferred in many simulation applications [40], such as fracture simulations [44] and fluid simulation [45]. Additional comparison results are provided in the supplemental materials.

Performance. We now evaluate the performance of our algorithm. The processing time mainly contains three com-

Table 1: Comparison of uniform tetrahedral meshing qualities. $|\mathbf{X}|$ is the number of sampled points, $|\tau|$ is the number of generated tetrahedra.

Model	Method	$\|\mathbf{X}\|$	$\|\tau\|$	θ_{min}	θ_{max}	κ_{min}	$\bar{\kappa}$	d_H	$\theta_{<10^\circ}(\%)$	$\theta_{<18^\circ}(\%)$
Sphere	TetGen	4.8K	25K	11.80	159.91	0.197	0.795	0.31	**0**	0.46
	CVT	5.0K	26K	12.24	161.67	0.236	0.910	0.32	**0**	0.1
	RODT	5.0K	26K	6.16	169.47	0.133	**0.915**	0.32	7.5×10^{-3}	0.02
	NODT	4.9K	26K	17.60	152.36	0.320	0.882	**0.28**	**0**	0.002
	Ours	4.6K	25K	**19.02**	**149.91**	**0.346**	0.821	0.32	**0**	**0.0**
Fertility	TetGen	42K	241K	6.84	165.73	0.104	0.782	0.31	3.7×10^{-4}	0.35
	CVT	41K	227K	7.43	162.37	0.187	**0.894**	**0.26**	0.06	0.16
	NODT	40K	216K	15.82	156.76	0.217	0.879	0.32	**0**	**0.01**
	Ours	41K	230K	**17.20**	**154.97**	**0.281**	0.808	0.34	**0**	0.04
Santa	TetGen	37K	192K	7.47	165.98	0.086	0.781	0.27	1.2×10^{-4}	0.29
	CVT	38K	203K	8.45	169.61	0.193	**0.911**	0.25	0.04	0.19
	NODT	39K	210K	15.05	158.41	0.208	0.863	**0.24**	**0**	**0.01**
	Ours	39K	209K	**17.30**	**154.79**	**0.292**	0.810	0.33	**0**	0.03
Joint	TetGen	20K	103K	8.79	161.20	0.181	0.789	0.23	3.3×10^{-4}	0.38
	CVT	21K	106K	10.01	165.24	0.191	**0.886**	**0.20**	**0**	0.12
	NODT	21K	105K	16.13	156.42	0.205	0.868	0.22	**0**	0.008
	Ours	21K	107K	**18.51**	**153.96**	**0.298**	0.815	0.27	**0**	**0.0**

Table 2: Comparison of graded tetrahedral meshing qualities.

Model	Method	$\|\mathbf{X}\|$	$\|\tau\|$	θ_{min}	θ_{max}	κ_{min}	$\bar{\kappa}$	d_H	$\theta_{<10^\circ}(\%)$	$\theta_{<18^\circ}(\%)$
Botijo	TetGen	39K	199K	7.15	167.15	0.089	0.773	0.32	9.2×10^{-4}	0.38
	CVT	38K	171K	4.94	171.10	0.085	0.768	0.32	0.12	0.98
	NODT	37K	185K	14.63	156.25	0.197	**0.874**	**0.31**	**0**	**0.02**
	Ours	38K	195K	**16.31**	**155.70**	**0.201**	0.796	0.33	**0**	0.11
Elephant	TetGen	55K	291K	8.72	162.23	0.148	0.772	**0.28**	1.2×10^{-4}	0.26
	CVT	56K	289K	4.17	169.89	0.073	0.742	0.33	0.02	0.89
	NODT	56K	294K	13.36	157.38	**0.192**	**0.881**	0.30	**0**	**0.01**
	Ours	57K	301K	**16.20**	**156.79**	0.182	0.796	0.31	**0**	0.13
Sculpt	TetGen	24K	128K	10.98	157.66	0.149	0.783	**0.18**	**0**	0.34
	CVT	23K	127K	5.31	168.54	0.085	0.761	0.28	0.21	0.96
	NODT	24K	128K	16.13	157.11	0.178	**0.869**	0.24	**0**	**0.01**
	Ours	23K	127K	**17.16**	**155.89**	**0.225**	0.802	0.19	**0**	0.05

ponents: sampling, Delaunay/regular triangulation, and mesh optimization. With the uniform grid as the implicit

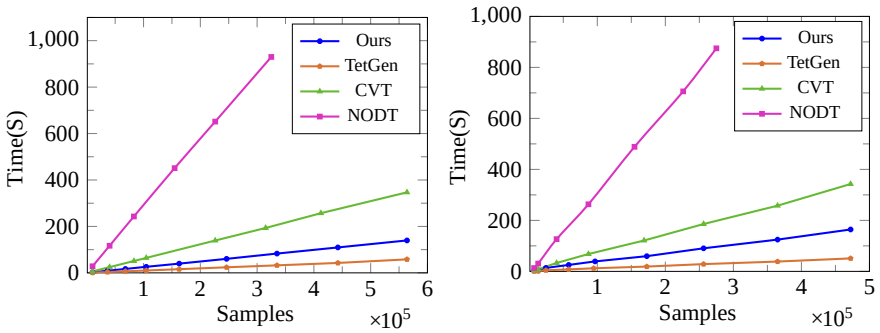

Figure 9: Performance comparison of our method with tetGen, CVT and NODT. Left: uniform tetrahedral mesh; right: graded tetrahedral mesh. The running time of our algorithm also contains building the uniform grid.

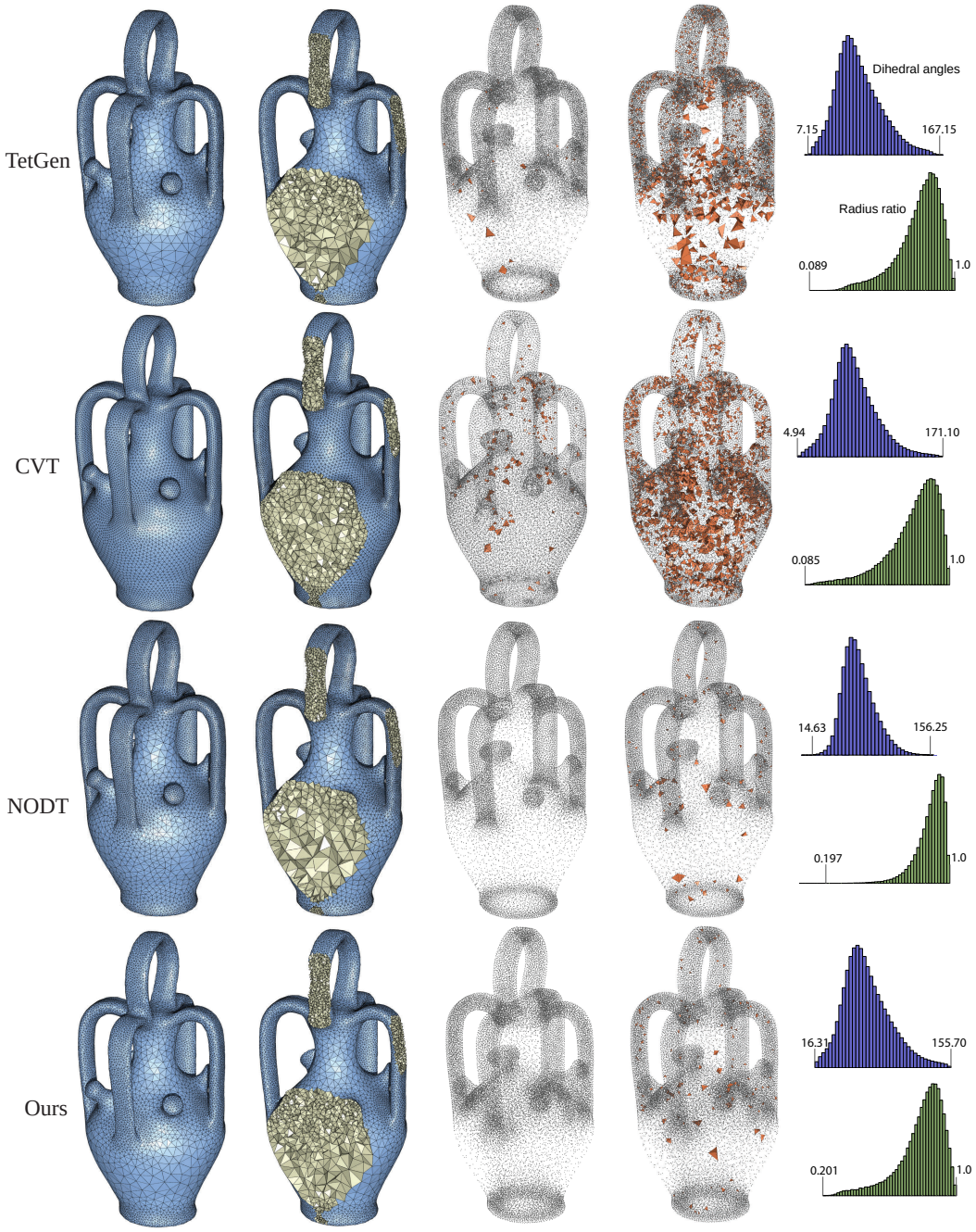

Figure 10: Comparison between different algorithms in generating a graded tetrahedral mesh. For each row, from left to right we show the meshing result, its cutaway view, slivers with dihedral angles smaller than 10° and 18°, and distributions of dihedral angles (blue) and radius ratios (green).

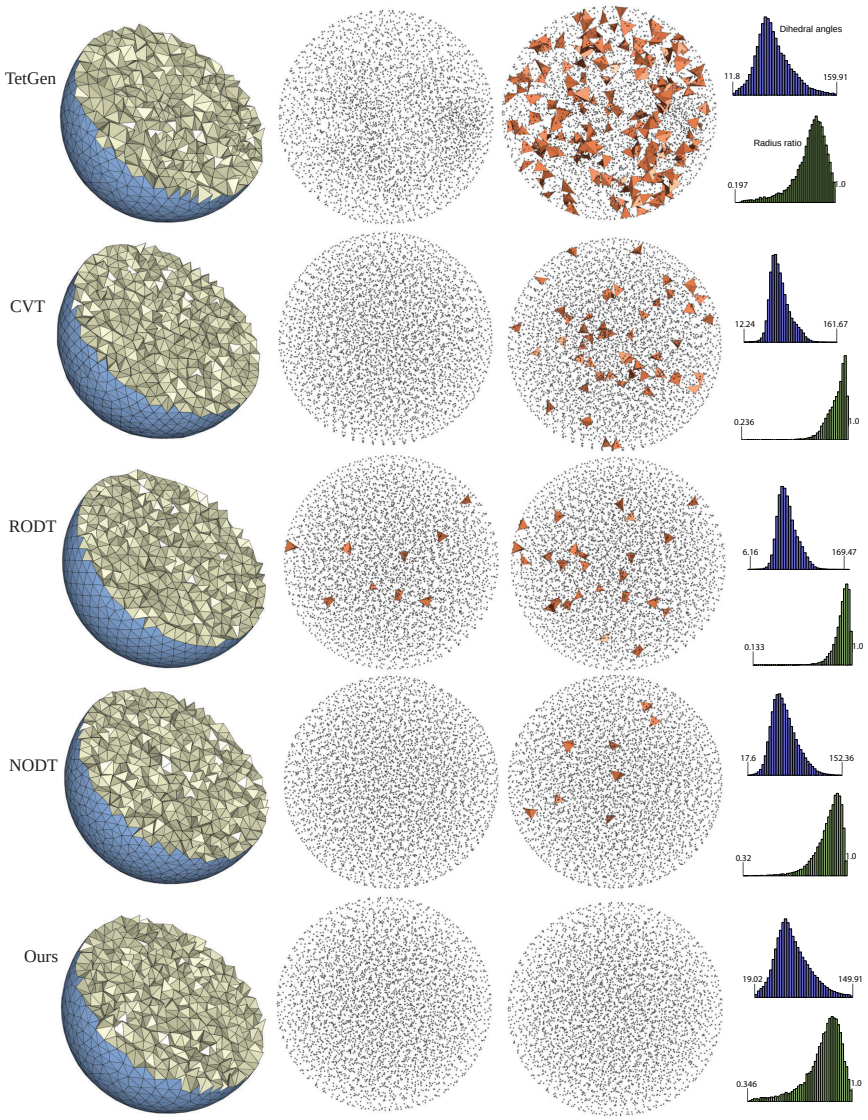

Figure 11: Comparison of our uniform tetrahedral mesh with those produced by other algorithms. For each row, from left to right we show the cutaway view of meshing result, slivers with dihedral angles smaller than 10° and 18°, and distributions of dihedral angles (blue) and radius ratios (green).

flat-array data structure, the sampling stage could achieve good performance. In the Delaunay/regular triangulation, we build a full triangulation once after the surface sampling. Then, in the later mesh extraction and optimization steps, we update dynamically for point removal and insertion. Note that in adaptive sampling, the resolution of the sub-grid used for conflict checking is much larger than that used for uniform sampling. Therefore, it reduces the performance of the graded mesh generation.

Figure 9 presents a comparison of the time our algorithm consumes with that of previous methods. All the timing curves are computed on the Botijo model, with an increasing number of sample points. In all these algorithms, the dihedral angle bound is set to 15 degrees. As illustrated in this figure, our algorithm is reasonably fast. Though TetGen is the most efficient method, its meshing results contain many slivers. In summary, our algorithm is capable of generating high-quality tetrahedral meshes, while maintaining good efficiency.

Structural analysis. To demonstrate that our meshing method is useful in real-world engineering applications, we

(a) TetGen (b) CVT (c) NODT (d) Ours

Figure 12: Structural analysis with the same force settings (arrows in (a)). The top row shows the stress map with color coding (zoom-in figures show a slightly different view), and the bottom row shows the interior structure.

perform structural analysis on the generated tetrahedral meshes as shown in Figure 12. The stress maps are computed by using the OOFEM library [46], an object-oriented finite element solver. First, an isotropic linear elastic material model is assigned to each mesh. After the user's exterior forces (defined on the boundary facets of some specific tetrahedra) are included, a non-linear static analysis is performed. The output data provide a quantitative description of the stress over all tetrahedra, which is visualized using a continuous stress map. As demonstrated in this figure, the stress distribution over our meshing structure is more uniform than over the others, indicating that our mesh can resist more forces.

Comparison to sliver removal approaches. There are several approaches that are able to improve the meshing quality by suppressing slivers as a post-processing step. Two well-known representative approaches are sliver exudation [33] and sliver perturbation [10]. The former tunes a weighted Delaunay triangulation that is free of slivers (minimal dihedral angles smaller than 5° in their method) based on a weight reassignment computation. This method eliminates slivers without adding new points or changing the positions of current points, while updating their connectivity. Sliver perturbation performs a gradient ascent search over the radius of the circumscribed sphere of a sliver as well as a gradient descent over the sliver's volume.

To justify the validity of our mesh optimization strategy, we compare our results with sliver exudation [33] and sliver perturbation [10]. For a fair comparison, we first generate an initial tetrahedral mesh using our method without any optimization. Then, we compare with sliver exudation and sliver perturbation for improvement on this mesh. Figure 13 shows the comparison results using the Bimba model, where a graded mesh is generated. We observe that sliver exudation has a limited effect, while sliver perturbation cannot always remove the slivers for a given threshold. Our strategy is able to generate meshes with larger minimal dihedral angles than these two state-of-the-art approaches.

Limitations. The main limitation of our proposed framework is that we do not have a theoretical proof of the lower bound of the dihedral angle that we can achieve. Our algorithm cannot guarantee the generation of a pleasant result if the required minimal dihedral angle bound is too large (e.g., 18°). In addition, the proposed method has similar difficulties as previous methods (ODT, CVT, Delaunay refinement, etc) in handling thin-sheet regions. We cannot generate a valid mesh if the sampling density does not meet the local feature size of the domain. Figure 14 shows such an example. Finally, although sampling has been demonstrated to be useful in a variety of graphics applications including remeshing and modeling, exploring the relationship between blue-noise properties and mesh generation is still an open problem.

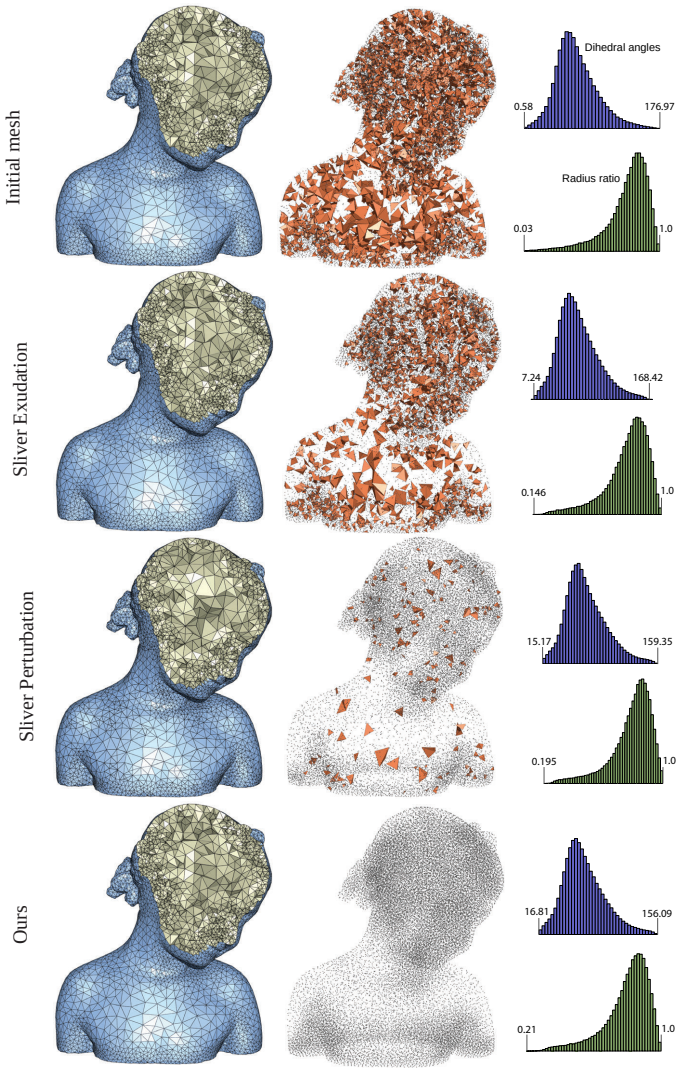

Figure 13: Comparison with sliver exudation [33] and sliver perturbation [10] on the Bimba model. Left: cutaway view of the initial mesh and the optimized mesh; Middle: sliver tetrahedra for a dihedral angle of 17 degrees; Right: dihedral angle distributions (blue), and radius ratio distributions(green).

6. Conclusion

In this paper, we introduced a simple yet efficient method for high-quality tetrahedral mesh generation that is superior to the current state-of-the-art approaches. Our approach is based on the maximal Poisson-disk sampling framework, while taking boundary preservation and mesh grading into account. Several optimization steps are performed to greatly improve the meshing quality. We demonstrated the validity of our algorithm by meshing a variety of complex domains even with sharp features, as well as comparing it with the state-of-the-art approaches. In the future, we plan to extend our approach to anisotropic mesh generation. We are also interested in meshing implicit surfaces such as isosurfaces.

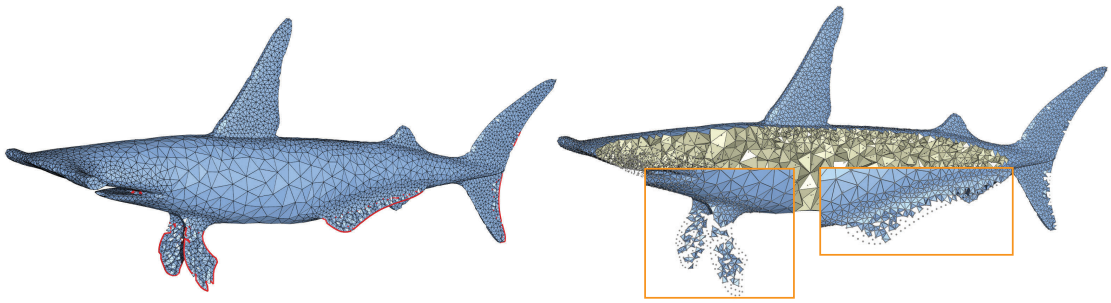

Figure 14: Our algorithm cannot guarantee a pleasant result when the minimal radius is larger than the local feature size of the domain. Left: surface remeshing after the boundary sampling stage. The non-manifold edges are shown in red. Right: tetrahedral meshing result with two zoom-in views.

Acknowledgements

This work was partially funded by the National Natural Science Foundation of China (Nos: 61372168, 61331018, 61272225, 61572274 and 61271431), the China National 863 Program (No. 2015AA016402), and Foreign 1000 Talent Plan (WQ201344000169).

References

[1] B. M. Klingner, B. E. Feldman, N. Chentanez, J. F. O'Brien, Fluid animation with dynamic meshes, ACM Trans. on Graphics (Proc. SIGGRAPH) 25 (3) (2006) 820–825.
[2] R. Ando, N. Thürey, C. Wojtan, Highly adaptive liquid simulations on tetrahedral meshes, ACM Trans. Graph. 32 (4) (2013) 103:1–103:10.
[3] H. Si, Tetgen, a Delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Softw. 41 (2) (2015) 11:1–11:36.
[4] C. Jamin, P. Alliez, M. Yvinec, J.-D. Boissonnat, CGALmesh: a generic framework for delaunay mesh generation, ACM Trans. on Math. Softw.
[5] S.-W. Cheng, T. K. Dey, J. A. Levine, A practical Delaunay meshing algorithm for a large class of domains, in: 16th Intl. Meshing Roundtable, 2007, pp. 477–494.
[6] C. Geuzaine, J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities 79 (11) (2009) 1309–1331,.
[7] C. B. Barber, D. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Transactions on Mathematical Software 22 (4) (1996) 469–483.
[8] J. R. Shewchuk, What is a good linear element? interpolation, conditioning, and quality measures, in: 11th Intl. Meshing Roundtable, 2002, pp. 115–126.
[9] P. Alliez, D. Cohen-Steiner, M. Yvinec, M. Desbrun, Variational tetrahedral meshing, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24 (3) (2005) 617–625.
[10] J. Tournois, C. Wormser, P. Alliez, M. Desbrun, Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation, ACM Trans. on Graphics (Proc. SIGGRAPH) 28 (3) (2009) 75:1–75:9.
[11] Z. Chen, W. Wang, B. Lévy, L. Liu, F. Sun, Revisiting optimal Delaunay triangulation for 3D graded mesh generation, SIAM Journal on Scientific Computing 36 (3) (2014) A930–A954.
[12] M. Ebeida, S. Mitchell, Uniform random voronoi meshes, in: W. Quadros (Ed.), Proceedings of the 20th International Meshing Roundtable, Springer Berlin Heidelberg, 2012, pp. 273–290.
[13] D.-M. Yan, P. Wonka, Gap processing for adaptive maximal Poisson-disk sampling, ACM Trans. on Graphics 32 (5) (2013) 148:1–148:15.
[14] J. Schöberl, NETGEN An advancing front 2D/3D-mesh generator based on abstract rules, Comput Visual Sci. 1 (1997) 41C52.
[15] W.-Y. Choi, D.-Y. Kwak, I.-H. Son, Y.-T. Im, Tetrahedral mesh generation based on advancing front technique and optimization scheme, International Journal for Numerical Methods in Engineering 58 (12) (2003) 1857–1872. doi:10.1002/nme.840.
[16] Y. Ito, A. M. Shih, B. K. Soni, Reliable isotropic tetrahedral mesh generation based on an advancing front method, in: IMR, 2004, pp. 95–106.
[17] F. Labelle, J. R. Shewchuk, Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles, ACM Trans. on Graphics (Proc. SIGGRAPH) 26 (3).
[18] J. Bronson, J. A. Levine, R. Whitaker, Lattice cleaving: A multimaterial tetrahedral meshing algorithm with guarantees, IEEE Transactions on Visualization and Computer Graphics 20 (2) (2014) 223–237.
[19] Z. Yu, J. Wang, Z. Gao, M. Xu, M. Hoshijima, New software developments for quality mesh generation and optimization from biomedical imaging data, Computer Methods and Programs in Biomedicine 113 (1).
[20] S.-W. Cheng, T. K. Dey, J. R. Shewchuk, Delaunay Mesh Generation, CRC Press, 2012.
[21] S. J. Owen, A survey of unstructured mesh generation technology, in: INTERNATIONAL MESHING ROUNDTABLE, 1998, pp. 239–267.
[22] M. Bern, P. Plassmann, Mesh Generation, Elsevier Science, 2000.
[23] P. J. Frey, P.-L. George, Mesh Generation: Application to Finite Elements, Hermes Science, 2001.
[24] H. Edelsbrunner, Geometry and Topology for Mesh Generation, Cambridge University Press, 2001.

[25] L. P. Chew, Guaranteed-quality Delaunay meshing in 3D, in: Proceedings of the Thirteenth Annual Symposium on Computational Geometry, 1997, pp. 391–393.
[26] J. Dardenne, S. Valette, N. Siauve, N. Burais, R. Prost, Variational tetraedral mesh generation from discrete volume data, The Visual Computer (Proc. CGI 2009) 25 (5) (2009) 401–410.
[27] E. Vanderzee, A. N. Hirani, D. Guoy, E. Ramos, Well-centered triangulation, SIAM J. Sci. Comput. 31 (6) (2010) 4497C4523.
[28] D. Yan, W. Wang, B. Lévy, Y. Liu, Efficient computation of 3D clipped Voronoi diagram, in: Advances in Geometric Modeling and Processing – GMP, 2010, pp. 269–282.
[29] D.-M. Yan, W. Wang, B. Lévy, Y. Liu, Efficient computation of clipped Voronoi diagram for mesh generation, Computer-Aided Design 45 (4) (2013) 843–852.
[30] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi tessellations: applications and algorithms, SIAM Review 41 (1999) 637–676.
[31] Q. Du, D. Wang, Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations, International Journal for Numerical Methods in Engineering 56 (9) (2003) 1355–1373.
[32] L. Chen, Mesh smoothing schemes based on optimal delaunay triangulations, in: IMR, 2004, pp. 109–120.
[33] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, S.-H. Teng, Sliver exudation, J. ACM 47 (5) (2000) 883–904.
[34] B. M. Klingner, J. R. Shewchuk, Agressive tetrahedral mesh improvement, in: 16th Intl. Meshing Roundtable, 2007, pp. 3–23.
[35] X. Jiao, D. Wang, H. Zha, Simple and effective variational optimization of surface and volume triangulations, Engineering with Computers 27 (1) (2011) 81–94.
[36] A. Lagae, P. Dutré, A comparison of methods for generating Poisson disk distributions, Computer Graphics Forum 27 (1) (2008) 114–129.
[37] T. R. Jones, Efficient generation of Poisson-disk sampling patterns, Journal of Graphics Tools 11 (2) (2006) 27–36.
[38] M. S. Ebeida, A. Patney, S. A. Mitchell, P. M. K. Andrew Davidson, J. D. Owens, Efficient maximal Poisson-disk sampling, ACM Trans. on Graphics (Proc. SIGGRAPH) 30 (4) (2011) 49:1–49:12.
[39] M. S. Ebeida, S. A. Mitchell, A. Patney, A. A. Davidson, J. D. Owens, A simple algorithm for maximal Poisson-disk sampling in high dimensions, Computer Graphics Forum (Proc. EUROGRAPHICS) 31 (2) (2012) 785–794.
[40] M. S. Ebeida, S. A. Mitchell, A. A. Davidson, A. Patney, P. M. Knupp, J. D. Owens, Efficient and good Delaunay meshes from random points, Computer-Aided Design 43 (11) (2011) 1506–1515.
[41] J. Guo, D.-M. Yan, X. Jia, X. Zhang, Efficient maximal Poisson-disk sampling and remeshing on surfaces, Computers & Graphics 46 (6-8) (2015) 72–79.
[42] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, W. Wang, Isotropic remeshing with fast and exact computation of restricted Voronoi diagram, Computer Graphics Forum 28 (5) (2009) 1445–1454.
[43] CGAL, CGAL, Computational Geometry Algorithms Library, http://www.cgal.org (2015).
[44] J. Bolander, S. Saito, Fracture analyses using spring networks with random geometry, Engineering Fracture Mechanics 61 (5) (1998) 569–591.
[45] F. de Goes, C. Wallez, J. Huang, D. Pavlov, M. Desbrun, Power particles: An incompressible fluid solver based on power diagrams, ACM Trans. on Graphics (Proc. SIGGRAPH) 34.
[46] B. Patzák, D. Rypl, Object-oriented, parallel finite element framework with dynamic load balancing, Advances in Engineering Software 47 (1) (2012) 35–50.