Guaranteed Quality Isotropic Tetrahedral Meshing of Arbitrary Geometric Domains

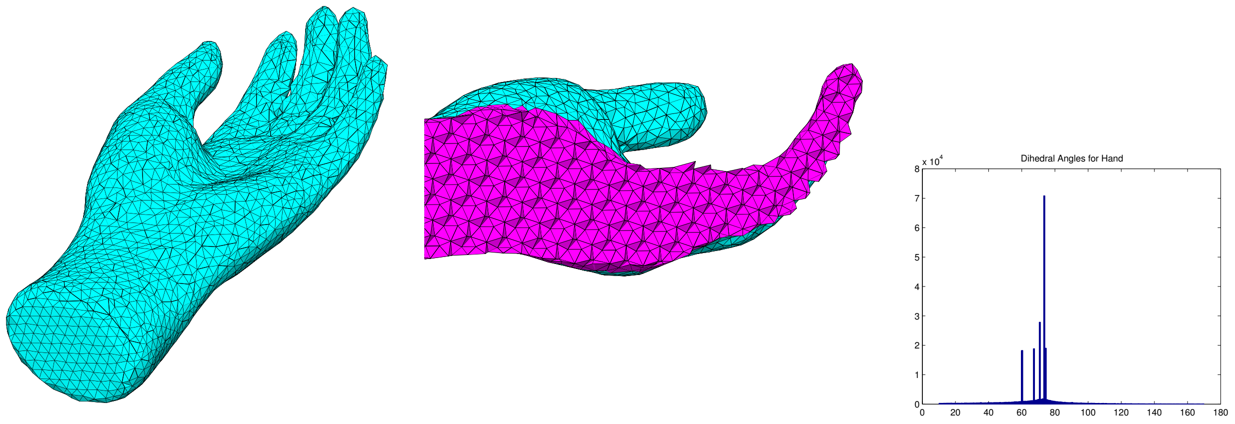

Figure 1: A tetrahedral mesh generated by our algorithm. On the left is a model of a hand and next to it is a cross section of the tetrahedra generated. Our algorithm produces isotropic meshes with no sizing variation across the domain. This mesh has over 8700 vertices and 40,000 tetrahedra. It took a total of 33 seconds to generate the signed distance transform and the mesh on a 3.06GHz Pentium IV with 1GB of RAM with an nVidia GeForce 7800 GTX graphics card. The histogram on the right shows the quality of the resulting mesh in terms of the dihedral angles of the tetrahedra. The meshes produced by our method are very close to being regular tetrahedral meshes. Quality measures in terms of aspect ratios (ratio of circumradius to inradius), and radius to edge ratios of the mesh are shown in Figure 2.

Abstract

Tetrahedralization of three-dimensional geometric domains find numerous applications in scientific computing, computer graphics, simulations and solid modeling. In this paper, we present a novel, automatic mesh generation algorithm of arbitrary geometric domains. We answer the question: *what is the best possible set of tetrahedra, in terms of their regularity, that can tile three-dimensional Euclidean space* , since it is well-known that it is not possible to do so with perfectly regular tetrahedra. We provide a very simple scheme to tile space based on the tetrahedral close packed structures of crystals. The quality of the tetrahedra in this tiling is extremely close to regular tetrahedra for almost all the typical quality measures used in the meshing literature. We further extend this technique to mesh arbitary geometric domains without significant degradation of quality. Our algorithm starts with our tiling scheme and then retains only those tetrahedra that lie completely inside or intersect the domain of interest. Intersecting tetrahedra are clipped with the domain boundary and retriangulated. One main advantage of our method is that it is significantly faster than most existing meshing algorithms since we avoid repeated Delaunay triangulation. We demonstrate the quality of the resulting isotropic meshes in a wide variety of examples.

1 Introduction

Simplicial mesh generation in three dimensions has numerous applications in scientific computing and visualization, simulations, computer graphics and solid modeling. Primarily, a simplicial mesh generation algorithm partitions a finite subset of $\mathbb{R}^3$ into small tetrahedra such that any two tetrahedra are either disjoint or intersect in a lower dimensional simplex. While the size requirement of each tetrahedra varies with the application, it is usually acceptable if all are of roughly the same size. This is called *isotropic meshing*. In this paper, we will focus mostly on isotropic meshing. Graded meshing, where the element sizing requirement varies across the domain, will be briefly addressed.

Perhaps, more crucial than the sizing requirement is the quality constraints imposed on the tetrahedra. The most popular measures [Bern et al. 1992] in three dimensions are *aspect ratio* (circumradius over inradius), *maximum dihedral angle* and *radius-to-edge ratio* (circumradius over shortest edge). While each of these measures are equivalent in two dimensions, it is not so in three dimensions.

There are some inherent mathematical limitations to the best possible meshes in three dimensions. It is well-known that $\mathbb{R}^3$ cannot be tiled with regular tetrahedra because its dihedral angle ($\arccos(1/3) = 70.53°$) is not a submultiple of 360° (see Figure 3 on the left). However, equilateral triangles do tile $\mathbb{R}^2$. Another issue that complicates meshing in three dimensions is dealing with the domain boundary. While it is possible to generate conforming triangulations respecting fairly general boundary constraints, this is not possible in three dimensions [Shewchuk 1998].

Cheng *et al.* [Cheng et al. 1999] provide a classification of various degenerate tetrahedral shapes that are possible while triangulating a set of points. Their classification is based on whether the vertices are close to a line or plane. One of the latter cases is the *sliver* where the four points are almost cocircular - for example, they lie almost along the equator of their circumsphere (see Figure 3 on the

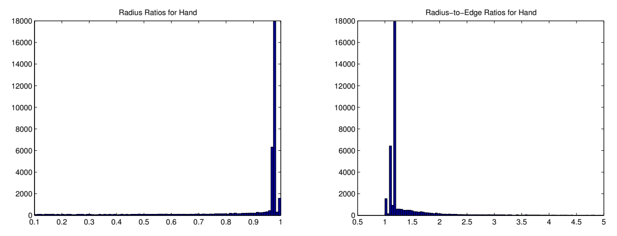

Figure 2: Histograms showing the quality of the tetrahedral mesh produced by our algorithm for the Hand model shown in Figure 1. The two histograms shows the quality in terms of aspect ratios (ratio of circumradius to inradius), and radius to edge ratios (in that order). The histograms show that the mesh is very close to being tiled with regular tetrahedra.

right). Previous approaches to meshing have tried to optimize one of these measures. One of the most common measures optimized is the radius-to-edge ratio. While they eliminate most of the bad cases, they do not get rid of slivers. On the other hand, aspect ratios and dihedral angles are good measures for all kinds of degeneracies.

Another issue that is important in mesh generation is the number of tetrahedra produced. The running time of the finite element method [Strang and Fix 1973] is a superlinear function of the number of tetrahedra in the mesh [Mitchell and Vavasis 2000]. Thus, there is significant penalty for meshes with a large number of tetrahedra.

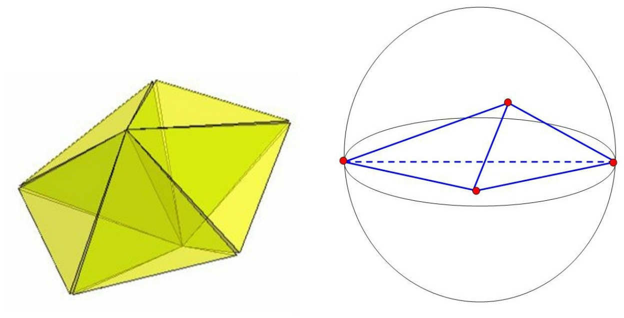

Figure 3: **Left:** Five regular tetrahedra sharing an edge. The gap between the faces are less than 1.5°. **Right:** Example of a sliver tetrahedron.

A combination of all these issues make the problem of tetrahedral mesh generation with boundary conditions very challenging. The first question, however, to ask is if we cannot tile space with regular tetrahedra, how close can we come? To answer this question, we resort to crystallographic literature. In order to understand the atomic structure of various complex transition alloys, they have investigated the possibilities of space filling by "almost regular" tetrahedra. Based on this study, we formulate a particular kind of space filling strategy that has surprisingly very nice properties.

Main Results: We present an algorithm to mesh arbitrary closed three-dimensional domains. We assume that there is an oracle that returns if a given point lies inside or outside the domain of interest. This allows us to mesh fairly general domains whose boundary can be polygonal meshes, implicit surfaces or even Boolean combinations of these domains. Figure 1 shows the mesh produced by our algorithm on the hand model. The resulting mesh has over 8700 vertices and 40,000 tetrahedra and took around 33 seconds to compute the signed distance field in the graphics hardware and the resulting mesh. The histograms in Figures 1 and 2 show that the tetrahedra in our meshes are close to being regular.

Our algorithm generates a special set of vertices obtained as a subset of a union of lattices. The Delaunay triangulation of this point set produces tetrahedra that are close to being regular. Tetrahedra that intersect the domain boundary are subdivided and the portion lying inside the domain is retriangulated. The quality of some of the boundary tetrahedra are worsened by this process. We perform a very simple local jittering of the generated boundary vertices to improve their quality.

Some of the main results of our algorithm are:

Tiling quality: The special distribution of points lets us generate tilings which are almost regular. All the tetrahedra in the tiling have dihedral angles that lie in the range $[60°, 74.2°]$. The lower bound for the range in a tiling must be $[60°, 72°]$ since either five or six tetrahedra much share en edge. It seems unlikely that our tiling pattern could be improved.

Generality of domain boundaries: We rely on an inside/outside classification test being available for the domain with respect to a point. This can usually be provided for general boundaries like polygonal meshes (through octrees, kd-trees or even signed distance transforms) those specified as the zero set of elementary functions, as well as Boolean combinations of such domains.

Efficiency: As opposed to variational methods [Alliez et al. 2005] or Delaunay refinement algorithms [Cheng et al. 1999], we do not recompute a retriangulation of most of the points specified. We do, however, perform retriangulation after small perturbations of the boundary vertices.

Implementation: We have a system implementation of our algorithm. It has been applied to complex polygonal meshes as well as some simple implicit surfaces and Boolean combinations. We measure the quality of the resulting simplicial mesh with respect to all three quality measures mentioned earlier: *aspect ratio, radius-to-edge ratio* and *dihedral angles*.

A limitation of this work is that we do not seek to optimize the number of tetrahedra produced inside the domain. However, by specifying a single sizing parameter we can control the size and number of simplices generated.

Organization: The rest of the paper is organized as follows. Section 2 briefly reviews the related previous work. Out tiling scheme is described in Section 3. Section 4 provides the details of how to use the tiling scheme to mesh a finite domain. We conclude in Section 6 after giving some results of applying our algorithm on different domains in Section 5.

2 Previous Work

In this section, we briefly review some of the existing mesh generation techniques. The problem of mesh generation has been studied extensively in the computational geometry, modeling and simulation, and visualization community. Detailed surveys of the mesh generation literature have been published [Bern and Plassmann 1999; Teng and Wong 2000; Eppstein 2001]. The interested reader can get more details about the current state of the art in these surveys and the references therein.

Meshing algorithms can be roughly classified into three main categories: Octree methods, Delaunay methods and Advancing front techniques.

Octree method: With this method, cubes containing the geometric domain are recursively subdivided until a desired resolution is reached. Irregular cells are then created where the voxels intersect the surface. Tetrahedra are generated from both the irregular cells on the boundary and the internal regular voxels [Yerry and Shephard 1984; Shephard and Georges 1991]. Bern *et al.* [Bern et al. 1990; Bern et al. 1994] use quadtrees to subdivide a two dimensional domain until each input point satisfies a *well separated* condition. The quadtree cells are then warped and triangulated to generate meshes of provable quality. Mitchell *et al.* [Mitchell and Vavasis 2000] extend this algorithm to three dimensions using octrees with guaranteed aspect ratio of all the tetrahedra. One disadvantage of octree methods is that the resulting meshes change as the orientation of the cubes in the octree structure is changed. Our method is close in spirit to the octree methods. Instead of tiling the domain with voxels and then triangulating them, we tile the space with tetrahedra. Hence our algorithm inherits most of the drawbacks of octree method as well.

Delaunay methods: By far the most popular of the meshing techniques are those utilizing the Delaunay criterion. The Delaunay criterion, sometimes called the ”empty sphere” property states that any node must not be contained in the interior of the circumsphere of any tetrahedra within the mesh. A typical approach is to first mesh the boundary of the geometry to provide an initial set of vertices. The first step is to compute the Delaunay triangulation of the boundary vertices. Nodes are then inserted incrementally into the existing mesh, redefining the simplicial mesh. Different methods use different strategies to choose interior nodes to insert.

One approach is to define new vertices at element circumcenters as proposed by Chew[15] and Ruppert[16]. When a specific order of insertion is followed, this technique is often referred to as ”Guaranteed Quality” as triangles can be generated with a minimum bound on any angle in the mesh. Delaunay refinement techniques, in general, offer optimality in terms of quality measures like radius-to-edge ratio and are also asymptotically optimal in the number of tetrahedral elements they generate. As mentioned before, however, optimizing radius-to-edge ratios can introduce slivers. Sliver exudation and weighted Delaunay refinement [Cheng et al. 1999; Cheng and Dey 2003] have been proposed to handle slivers as part of the refinement process.

Another approach that has gained lot of traction recently is the use of Centroidal Voronoi tessellations [Du and Wang 2003]. The main idea here is to minimize a quadratic energy functional such that the resulting meshes are dual to optimal Voronoi diagrams. Du *et al.* [Du and Wang 2003] show that the functional is minimized when the vertices are positioned at the centroids of their own Voronoi cell. Through a subtle modification of the energy functional, Chen *et al.* [Chen and Xu 2004] argue that this energy measures the quality of the mesh tetrahedra, not their dual Voronoi cell. Alliez *et al.* [Alliez et al. 2005] provide an algorithm to consistently minimize the energy functional proposed by Chen *et al.* [Chen and Xu 2004]. Freitag *et al.* [Freitag and Ollivier-Gooch 1996] use local swapping and smoothing techniques like Laplacian smoothing to improve a given tetrahedral mesh.

Advacing front methods: Another class of 2D and 3D mesh generation algorithms is the advancing front, or moving front method. In this method, the tetrahedra are built progressively inward from the triangulated surface. An active front is maintained where new tetrahedra are formed. A sizing function can also be defined in this method to control element sizes. Li *et al.* [Li et al. 1999; Li et al. 2000] use an advancing front technique combined with sphere packing to mesh the domain. They provide guaranteed quality of tetrahedra based on radius-to-edge ratios.

2.1 Tetrahedral Close Packing

Tetrahedra have interesting connections to sphere packings [Conway and Sloane 1998], certain special tilings of space like foams and froths [Sullivan 2000] and complex alloy structures. It is well known that the maximum number of spheres that can be arranged in $\mathbb{R}^3$ such that all sphere touch each other is four. The dual structure leads to a regular tetrahedron. Crystallographers have studied transition metal alloys in which the atoms are arranged as nearly regular tetrahedra [Shoemaker and Shoemaker 1986]. Frank and Kasper [Frank and Kasper 1958; Frank and Kasper 1959] observed that studying sphere packings and their dual tetrahedral network can explain the crystalline structure of transition metal alloys. These are called the *Frank-Kasper phases* and belong to a class of tetrahedral close-packed (tcp) structures. Sullivan [Sullivan 2000] provides a mathematical definition of tcp structures as follows: triangulations whose edges have valence (number of faces adjacent to an edge) five or six, but no two edges of a triangle have valence six. Eppstein *et al.* [Eppstein et al. 2004] use this definition to list various tilings of space with near regular tetrahedra.

3 Near Regular Tetrahedral Tiling of Space

We shall now describe our tiling of space with near regular tetrahedra. We start by providing some definitions and cover relevant background material.

3.1 Definition and Background

We start by defining a lattice. In simple terms, a lattice in $\mathbb{R}^3$ is a regular placement of points in an infinite integer grid defined by three basis vectors, $(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$. Any point $\mathbf{p}$ in the lattice can be expressed as $a\mathbf{v_1} + b\mathbf{v_2} + c\mathbf{v_3}$, where $a, b, c \in \mathbb{Z}$. In general, the basis vectors need not be mutually orthogonal, but we will assume that they are linearly independent. Let $\mathbf{V}$ be the matrix whose column i is the vector $\mathbf{v_i}$. Then,

Definition 3.1. *A lattice L in three dimensions is a discrete subgroup of $\mathbb{R}^3$. $L = \mathbf{V}\mathbb{Z}^3$.*

A canonical integer grid in three dimensions can be specified by the identity matrix. Lattices of this type are referred to as *cubic lattices*. Lattices are invariant to scaling and rigid transformations (rotation and translation). So, in general, it is possible to express a lattice of the form $\alpha\mathscr{R}\mathbf{V}\mathbb{Z}^3 + \mathbf{t}$, where α is the scaling parameter, $\mathscr{R}$ is a rotation matrix and $\mathbf{t}$ is a translation vector.

Some of the well-known crystal structures like body-centered cubic (bcc) lattice and face-centered cubic (fcc) lattice can be expressed as the union of multiple lattices. Let $\mathbf{I}$ denote the 3×3 identity matrix. Then, $BCC = \{\mathbf{I}\mathbb{Z}^3\} \cup \{\mathbf{I}\mathbb{Z}^3 + (0.5, 0.5, 0.5)\}$.

3.2 Tiling Scheme

We are now ready to provide our tiling scheme by choosing the vertices. We start by choosing the lattice $L = 0.5\mathbf{I}\mathbb{Z}^3$. We classify nodes in this lattice into four colors, $\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}$ [Conway and Torquato 2006]. A node $\mathbf{p} = (x, y, z)$ has color $\mathbf{i}$ if

$$2(x+y+z) \equiv i \mod 4$$

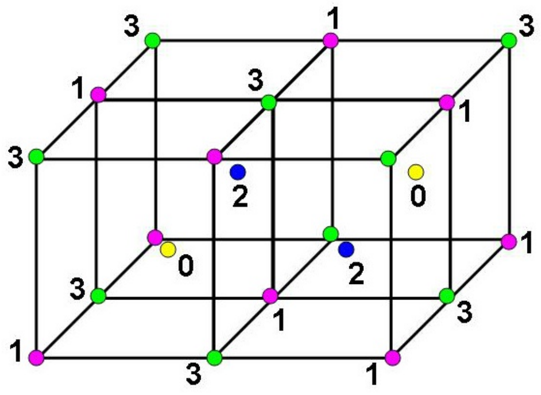

Figure 4: Four coloring of the lattice $0.5\mathbf{I}\mathbb{Z}^3$. Color $\mathbf{0}$ is yellow, color $\mathbf{1}$ is magenta, $\mathbf{2}$ is blue and $\mathbf{3}$ is green. The blue and yellow nodes are at the center of the cells, while the green and magenta nodes are at the cell vertices.

Figure 4 shows the coloring scheme. Given this coloring, we replace nodes of color $\mathbf{2}$ (blue) and $\mathbf{3}$ (green) by nodes of color $\mathbf{2.5}$ (cyan) which are located at the midpoints of the shortest line segments connecting nodes of color $\mathbf{2}$ and $\mathbf{3}$. This is shown in Figure 5.

Collect all nodes of color $\mathbf{0}$, $\mathbf{1}$ and $\mathbf{2.5}$. The Delaunay triangulation of this point set tiles space with almost regular tetrahedra (see Figure 6. Incidentally, this is one of the tetrahedral close-packed structures. The dual Voronoi cells of this triangulation all have 16 faces (12 pentagonal and 4 hexagonal). Another interesting observation is that all the vertex coordinates are rational.

Figure 7 shows the histogram of three quality measures of the spatial tiling (see Figure 6) produced by the vertex positioning scheme. Statistics of the tiling are shown in Table 1. Notice that the histograms are heavily concentrated near the values for a regular tetrahedron. There are exactly five values of dihedral angles in the tiling and all of them range between 60° and 74.2° (the dihedral angle for a regular tetrahedron is 70.53°). The normalized radius ratios are equally remarkable. All the tetrahedra have values greater than 0.97.

4 Meshing Algorithm

In this section, we will describe our algorithm to mesh arbitrary three-dimensional domains. Let the domain of interest be Ω whose boundary is $\partial\Omega$. We will assume throughout the algorithm discussion that $\partial\Omega$ is an orientable, compact surface enclosing a volume. Our algorithm is divided into several steps.

Algorithm 1 is a high level overview of our algorithm. We will now go through the details of the various steps.

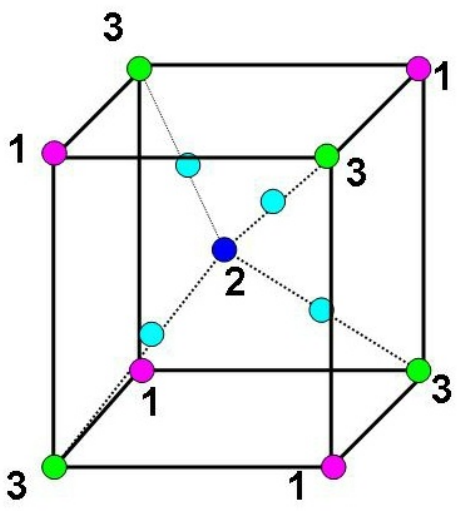

Figure 5: Nodes of color $\mathbf{2}$ and $\mathbf{3}$ are replaced by nodes of color $\mathbf{2.5}$ (cyan in the figure) which are positioned at the midpoints of the shortest line segments connecting nodes of color $\mathbf{2}$ and $\mathbf{3}$.

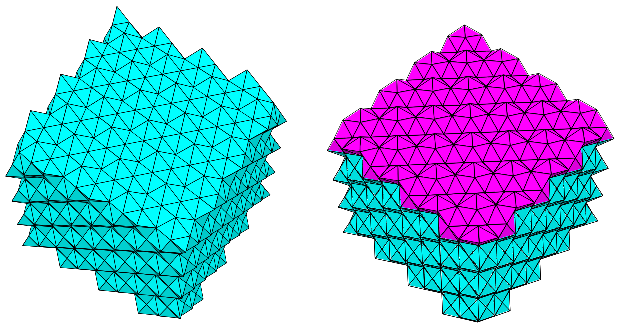

Figure 6: **Left:** The image shows the tiling of space with the tetrahedra specified by the above scheme. **Right:** The image on the right shows the cross section of the tiling.

4.1 Computing Signed Distance Fields

We use the signed distance fields for many purposes. The primary purpose is that it allows us to answer inside/outside queries on the domain. This test allows us to classify points and tetrahedral elements which are repeatedly used throughout the algorithm. Secondly, it greatly speeds up the intersection computation of individual tetrahedra with the domain boundary. There are several methods to compute the signed distance field both in software and hardware [Mauch 2000]. With the recent advances in programmable graphics hardware, it is possibly to scan convert and compute the distance transform for very complex meshes in a matter of a few seconds [Sud et al. 2006]. The grid resolution used in the distance field computation determines the accuracy of various queries. In practice, computing a $100 \times 100 \times 100$ grid suffices for reliable querying.

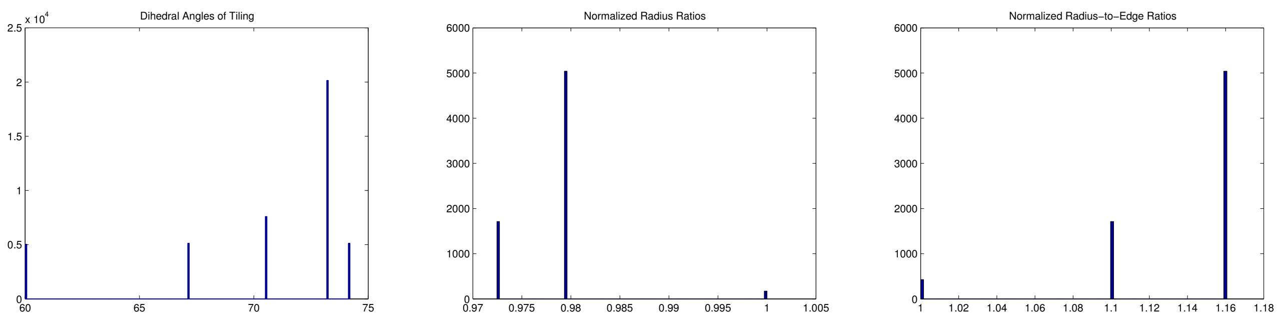

Figure 7: From left, the image shows the histogram of all the dihedral angles in the mesh (notice that there are only 5 values of the dihedral angles ranging from 60° to 74.2°), the histogram of the radius ratios (all tetrahedra have a quality measure > 0.97) and the histogram of radius-to-edge ratios (values range from 1.0 to 1.16), respectively. The radius ratio and radius-to-edge ratios are normalized so that it is 1.0 for regular tetrahedra.

Quality Measure	Minimum	Maximum	Average	Std. Deviation
Min. Dihedral Angle	60.0°	70.53°	64.4°	4.54°
Max. Dihedral Angle	70.53°	74.21°	72.86°	1.36°
Normalized Aspect Ratio	0.972	1.0	0.982	0.01
Normalized Radius-to-Edge Ratio	1.0	1.16	1.11	0.06

Table 1: Table showing the statistics of various quality measures used to measure our spatial tiling. The corresponding values for a regular tetrahedron is 70.53° for dihedral angle and 1.0 for the normalized aspect and radius-to-edge ratios. The table clearly shows that our tiling comes very close to being regular.

Algorithm 1 TetrahedralMeshing($\partial\Omega$, s)

Input: Domain boundary $\partial\Omega$ and specified element size s **Output:** Tetrahedral mesh M in the interior of Ω

```
$\partial\Omega$ = ReadInputBoundary()
ComputeSignedDistanceField($\partial\Omega$)
$N$ = EstimateNumberofPointSamples($\partial\Omega$, $s$)
GeneratePointSamplesFromTiling($N$)
ComputeTilingMesh()
ClassifyMeshTetrahedra($\Omega$) as {inside, outside, intersecting}
Discard outside tetrahedra
Mesh $M$ = inside tetrahedra
$\mathscr{I}$ = set of intersecting tetrahedra
while $\mathscr{I} \neq \emptyset$ do
    Pick $t \in \mathscr{I}$
    $R = t \cap \partial\Omega$
    $R'$ = retriangulate $R$
    $M = M \cup R'$
end while
Return $M$
```

4.2 Estimating Number of Points

The number of points to be generated inside the domain is a function of the sizing parameter s and the domain volume. For isotropic meshing, s is a constant. We estimate the domain volume by performing a quasi-Monte Carlo sampling [Niederreiter 1992] and estimating the volume by querying the signed distance field. This is a standard approach and is easily extended to sizing fields that vary across the domain [Alliez et al. 2005]. We use this estimate to determine the number of copies of a single lattice block to produce the vertices of the tiling. The details of the point and tiling mesh generation were discussed in Section 3.2. We generate a tiling of the space such that it contains the domain of interest.

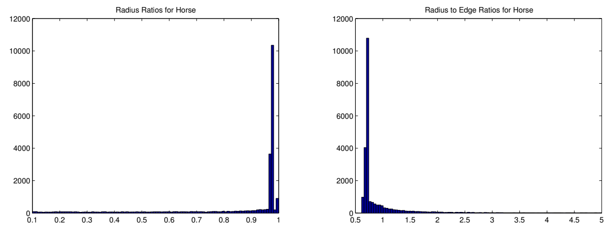

Figure 9: This image shows the distribution of the aspect ratio (circumradius to inradius) and the radius-to-edge ratio of the simplices in the tetrahedral mesh produced by our algorithm on the model of the horse shown in Figure 8.

4.3 Tetrahedral Classification

Once the tetrahedral tiling is produced, we have to classify tetrahedra as either belonging inside, outside or intersecting the domain. In order to classifiy tetrahedra reliably, we assume that the sign of the distance field at the vertices and circumcenter of the tetrahedra is sufficient to determine if it intersects the domain or not. While this condition need not necessarily be satisfied in domains with fairly complex topology, this assumption is quite reasonable for most practical considerations. Our heuristic is as follows.

- If the absolute distance value at the circumcenter is greater than the circumradius and the circumcenter lies inside the domain, then the tetrahedron is classified **inside**.
- If the absolute distance value at the circumcenter is greater than the circumradius and the circumcenter lies outside the domain, then the tetrahedron is classified **outside**.
- If neither is true, detect zero crossing of the distance field along any of the six edges. If there is one, classify tetrahe-

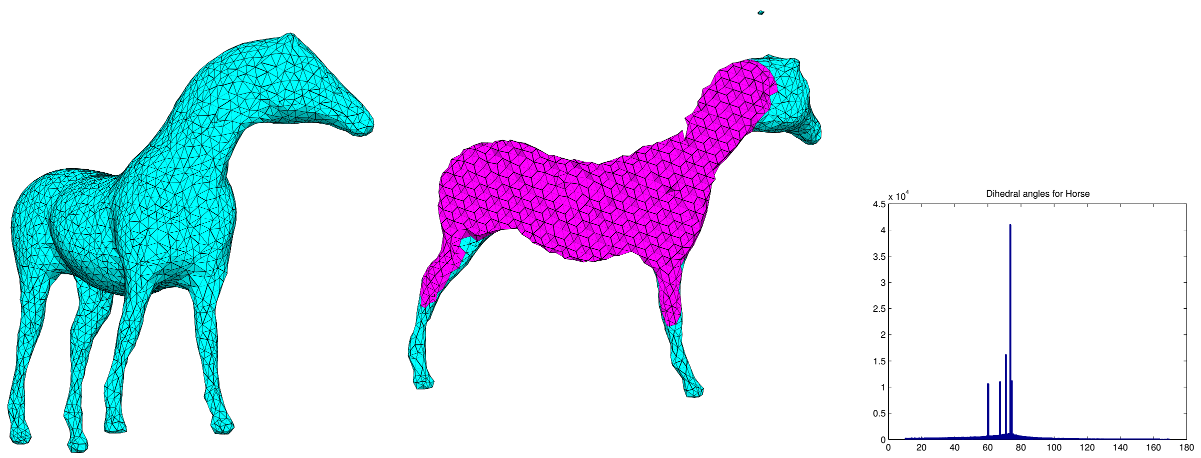

Figure 8: From left, this image shows the tetrahedral mesh produced by our algorithm on the model of the horse. The figure to its right is a cross section of the simplices generated. This mesh has close to 6000 vertices and 26,000 tetrahedra. It took a total of 19 seconds to generate the signed distance transform and the mesh on a 3.06GHz Pentium IV with 1GB of RAM with an nVidia GeForce 7800 GTX graphics card. The histogram on the right shows the distribution of the dihedral angles in the mesh. The histogram for aspect ratios and radius-to-edge ratios are shown in Figure 9.

dron as **intersecting**.

- Otherwise, classify tetrahedron as **inside** or **outside** depending on the orientation of the circumcenter.

4.4 Tetrahedral Intersection and Retriangulation

The next step in the algorithm is to clip the intersecting tetrahedra with the domain boundary. We use the signed distance field again for this purpose. We use linear interpolation along the simplicial edges to determine the intersection point. This is akin to the Marching Tetrahedra method commonly used in volume visualization. Figure 11 (top row) shows the various ways in which a tetrahedron can intersect the domain boundary (provided the assumption that intersection is determined by sign change along edges). The bottom row shows the polyhedra that remain after they are clipped by the domain boundary. The shapes are convex and are not very difficult to triangulate. We observe here that this final step of the algorithm which performs the retriangulation deteriorates the mesh quality that is resulted. The number of such simplices is a very small fraction (surface area vs. volume) of the entire mesh. Our experiments also bear this fact out.

Before returning the final mesh, we try to improve the quality of the mesh by only trying to perturb the boundary vertices slightly. We search in the local neighborhood of the tangent plane to see if any improvement is possible. This is similar to the "boundary vertex jittering" peformed in Alliez *et al.* [Alliez et al. 2005].

5 Results and DIscussion

We have implemented our algorithm and tested it on various domains. It is fairly simple to implement since software for most of the heavy lifting like Delaunay triangulation and distance transforms are available. We used Qhull (`http://www.qhull.org`) to perform the Delaunay triangulation. We used the software of Sud *et al.* [Sud et al. 2006] which computes the signed distance fields on the graphics hardware. Prior to obtaining this code, we use a software-based scan conversion method [Mauch 2000] to evaluate distance fields.

Our algorithm can handle large complex meshes and produce quality meshes in a matter of seconds. The speed of our algorithm can be attributed to the fact that we do not incur the cost of repeated Delaunay triangulations. The quality of the resulting meshes is evident in the histogram plots. Figure 8 shows the mesh produced by our algorithm on the horse model. This mesh has around 6000 vertices and 26,000 tetraheda and it took 19 seconds to run on an 3.06GHz Pentium IV with 1GB of RAM and a nVidia GeForce 7800 GTX graphics card.

Model	# Verts	# Tets	Time
$Hand$	8738	39994	33
$Horse$	5851	25404	19
$Bunny(Midres)$	10774	51548	31
$GradedSphere$	8095	40178	22
$Socket$	10667	48503	35
$Boolean$	875	4231	8

Table 2: Performance of our algorithm on various input domains. All the timing measurements are reported on a 3.06 GHz Pentium IV with 1 GB of RAM and an nVidia 7800 GTX graphics card. The timing includes the time to generate the signed distance field and is measures in seconds. For large datasets, the distance field computation dominates the total time. Hand is the model in Figure 1, Horse is the model in Figure 8, Bunny (Midres) is the bunny model on the top right of Figure 10, Graded Sphere is the model in Figure 12, Socket is the model on the left of Figure 13 and Boolean is the model to its right in Figure 13.

Perhaps one could wonder that using an initial distribution of points like the one generated here could be used in conjunction with other methods to produce better quality meshes. We experimented with a few mesh generation software like Tetgen (`http://tetgen.berlios.de/`), QMG [Mitchell and Vavasis 2000] and GRUMMP [Freitag and Ollivier-Gooch 1996]. However, all the meshed produced as a result of perturbing our initial point set were of significantly lower quality. It appears that the tiling approach is a special case that one does not arrive easily through local

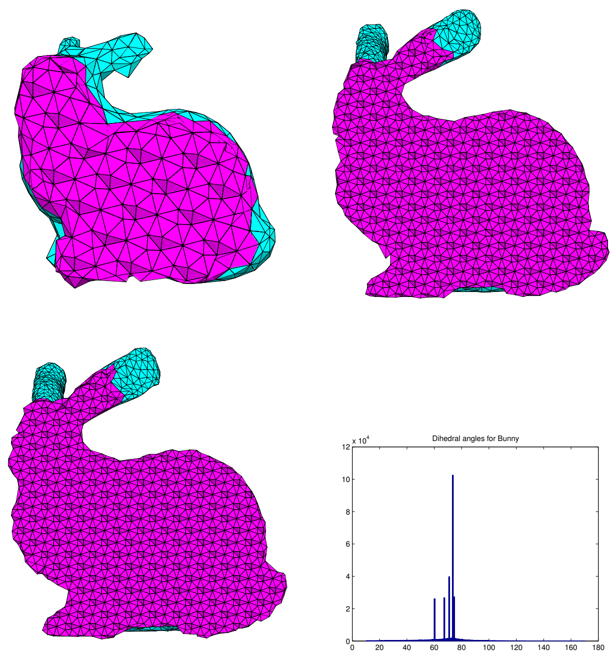

Figure 10: Sizing variation: The user can control the size of the mesh elements using a parameter. This parameter is directly proportional to the dimensions to the simplices produced by the meshing algorithm. In this image, the parameter is decreased. With increasing complexity, the bunny has 1194 verts and 4868 tets; the second one has 10774 verts and 51548 tets; and the final one has 13009 verts and 62887 tets. The histogram on the right shows the distribution of the dihedral angles for the mesh in the top right.

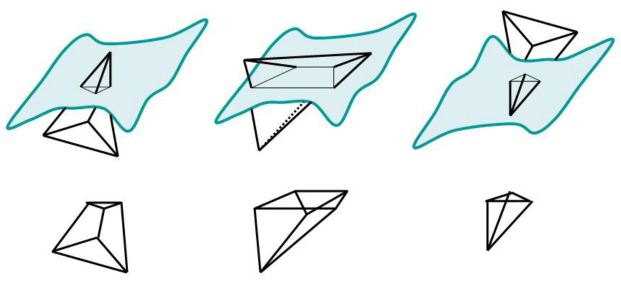

Figure 11: **Top Row:** The image shows various ways a tetrahedron can intersect the domain boundary. **Bottom Row:** These the clipped parts of the tetrahedron that remain in the domain. As is observable, the remaining polyhedra are convex and can be triangulated easily.

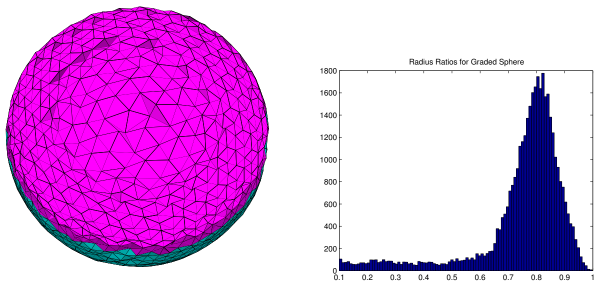

Figure 12: **Left:** The image shows the cross section of a graded mesh based on the local feature size. The mesh elements are becoming smaller as we approach the boundary of the domain. **Right:** The image on the right shows the histogram of the normalized radius ratios of all the tetrahedra in the mesh.

search techniques. That is probably why meshing in three dimensions is so difficult. On a related note, the question of meshing a cube with tetrahedra such that all dihedral angles are strictly less than 90° is still open. The related problem in two dimensions has been solved.

There are many limitations to our approach. One of the most important aspects of mesh generation is the ability to control the sizing function and vary it along the mesh. We experimented a little with some heuristics to control the quality degradation, but none were promising. Figure 12 shows the cross section of a graded mesh of a sphere and the distribution of the normalized radius ratios.

The algorithm as described in Section 4 is oblivious of the presense of sharp edges and corners. However, we do handle sharp features specially. The user can specify vertices in the boundary mesh that have to be fixed. The vertices will be retained in the final mesh. The only special case we handle is if the boundary tetrahedra intersect in that region of the domain. Figure 13 (left) shows the mesh produced when the boundary has sharp features. We do not show the quality histogram because of lack of space. However, the mesh is of much lower quality than for smooth surfaces.

We can also mesh domains by an implicit specification like a set operation between two domains. In Figure 13 (right) we show a toy example of meshing the difference between a truncated cylinder and sphere. The domains are specified by their implicit equation.

The tiling scheme presented in Section 3.2 starts with the lattice $L = 0.5\mathbf{I}\mathbb{Z}^3$. However, we can also apply a rigid transformation to this lattice to suit our domain of interest. For polyhedral domains, we use the vertices in the mesh boundary to compute a 3×3 covariance matrix $\mathscr{C}$. We then perform a simple principal component analysis (PCA) on this matric to obtain an orthogonal frame $\mathscr{R}$ aligned with the domain. The vertices obtained by the tiling scheme are transformed by this rotation matrix $\mathscr{R}$. The rest of the algorithm remains the same.

6 Conclusion

In this paper, we have presented a new algorithm to mesh arbitrary geometric domains in three dimensions. We show that a particular set of points generated from an initial lattice produces close to regular tetrahedra which tile space. We also presented an algorithm that uses this initial tiling to mesh arbitary geometric domains without significant degradation of quality. One main advantage of our method is that it is significantly faster than most existing meshing algorithms since we avoid repeated Delaunay triangulation. We have demonstrated the quality of the resulting isotropic meshes in a wide variety of examples.

References

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN,

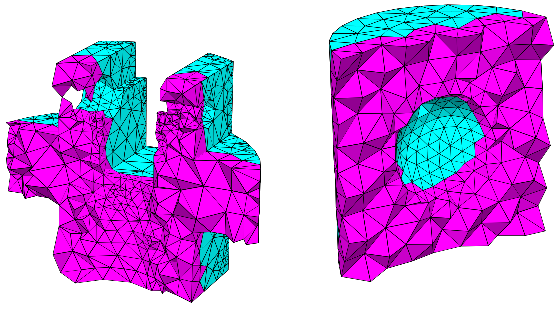

Figure 13: **Left:** The image shows the cross section of a CAD model with sharp features. The user is allowed to specify special points that are retained in the final mesh. **Right:** Boolean operations: The image on the right is a simple illustration of the generality. By specifying two implicit surfaces and their difference, we are able to mesh the correct domain without too much trouble.

M. 2005. Variational tetrahedral meshing. *ACM Trans. Graph. 24*, 3, 617–625.

BERN, M., AND PLASSMANN, P. 1999. *Mesh Generation, Chapter 6 in Handbook of Computational Geometry*. Elsevier Science.

BERN, M. W., EPPSTEIN, D., AND GILBERT, J. R. 1990. Provably good mesh generation. In *IEEE Symposium on Foundations of Computer Science*, 231–241.

BERN, M. W., EDELSBRUNNER, H., EPPSTEIN, D., MITCHELL, S. L., AND TAN, T. S. 1992. Edge insertion for optional triangulations. In *LATIN '92: Proceedings of the 1st Latin American Symposium on Theoretical Informatics*, Springer-Verlag, London, UK, 46–60.

BERN, M., EPPSTEIN, D., AND GILBERT, J. 1994. Provably good mesh generation. *J. Comput. Syst. Sci. 48*, 3, 384–409.

CHEN, L., AND XU, J. 2004. Optimal delaunay triangulations. *Journal of Computational Mathematics 22*, 2, 299–308.

CHENG, S. W., AND DEY, T. K. 2003. Quality meshing with weighted delaunay refinement. *SIAM J. Computing 33*, 69–93.

CHENG, S., DEY, T. K., EDELSBRUNNER, H., FACELLO, M. A., AND TENG, S. 1999. Sliver exudation. In *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*, ACM Press, New York, NY, USA, 1–13.

CONWAY, J. H., AND SLOANE, N. J. A. 1998. *Sphere-packings, lattices, and groups*. Springer-Verlag New York, Inc., New York, NY, USA.

CONWAY, J. H., AND TORQUATO, S. 2006. Packing, tiling and covering with tetrahedra. *Personal Communication*.

DU, Q., AND WANG, D. 2003. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal of Numerical Methods in Engineering 56*, 9, 1355–1373.

EPPSTEIN, D., SULLIVAN, J. M., AND UNGOR, A. 2004. Tiling space and slabs with acute tetrahedra. *Comput. Geom. Theory Appl. 27*, 3, 237–255.

EPPSTEIN, D. 2001. Global optimization of mesh quality. *Tutorial at 10th Int. Meshing Roundtable, Newport Beach.*

FRANK, F., AND KASPER, J. 1958. Complex alloy structures regarded as sphere packings. i. definitions and basic principles. *Acta Crystallographica 11*, 3, 184–190.

FRANK, F., AND KASPER, J. 1959. Complex alloy structures regarded as sphere packings. ii. analysis and classification of representative structures. *Acta Crystallographica 12*, 7, 483–499.

FREITAG, L. A., AND OLLIVIER-GOOCH, C. 1996. A comparison of tetrahedral mesh improvement techniques. *Proc. 5th Int. Meshing Roundtable, Sandia National Laboratories*, 87–106.

LI, X., TENG, S., AND UNGOR, A. 1999. Biting spheres in 3d. *Proc. 8th Int. Meshing Roundtable, South Lake Tahoe*, 85–95.

LI, X., TENG, S., AND UNGOR, A. 2000. Biting: Advancing front meets sphere packing. *International Journal of Numerical Methods in Engineering 49*, 61–81.

MAUCH, S. 2000. A fast algorithm for computing the closest point and distance transform.

MITCHELL, S. A., AND VAVASIS, S. A. 2000. Quality mesh generation in higher dimensions. *SIAM J. Comput. 29*, 4, 1334–1370.

NIEDERREITER, H. 1992. *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

SHEPHARD, M. S., AND GEORGES, M. K. 1991. Three-dimensional mesh generation by finite octree technique. *International Journal of Numerical Methods in Engineering 32*, 709–749.

SHEWCHUK, J. R. 1998. Tetrahedral mesh generation by delaunay refinement. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, ACM Press, New York, NY, USA, 86–95.

SHOEMAKER, D., AND SHOEMAKER, C. 1986. Concerning the relative numbers of atomic coordination types in tetrahedrally close packed metal structures. *Acta Crystallographica 42*, 3–11.

STRANG, G., AND FIX, G. J. 1973. *An Anslysis of the Finite Element Method.* Prentice Hall.

SUD, A., GOVINDARAJU, N., GAYLE, R., AND MANOCHA, D. 2006. Interactive 3d distance field computation using linear factorization. In *Proc. ACM Symposium on Interactive 3D Graphics and Games (I3D) (To Appear).*

SULLIVAN, J. 2000. New tetrahedrally close-packed structures, in: Foams, emulsions and their applications. *Proceedings of Eurofoam*, 111–119.

TENG, S., AND WONG, C. W. 2000. Unstructured mesh generation: Theory, practice, and perspectives. *Int. J. Comput. Geometry Appl. 10*, 3, 227–266.

YERRY, M. A., AND SHEPHARD, M. S. 1984. Three-dimensional mesh generation by modified octree technique. *International Journal of Numerical Methods in Engineering 20*, 1965–1990.