Biting Spheres in 3D *

Xiang-Yang Li † Shang-Hua Teng Alper Üngör ‡

Abstract. *We present an efficient 3D meshing algorithm which combines the merits of two popular meshing methods, advancing front and sphere packing methods. In particular, it inherits the practicality and simplicity of advancing front and the size-and-quality guarantee of the sphere packing method. We prove that it generates a well-shaped mesh whose size is within a constant factor of the optimal possible for an arbitrary domain in 3D, extending the previous 2D results and algorithms to 3D.*

Keywords. unstructured mesh generation, mesh quality, advancing front, plastering, sphere packing, biting.

1 Introduction

Mesh generation, the problem of constructing a proper discretization of a continuous domain, is an essential step in numerical simulation of physical and engineering problems [24]. For problems with complex geometry boundaries and with solutions that change rapidly, a good unstructured meshing algorithm uses elements of properly chosen size and shape that adapt to the complex geometry and solution accuracy. In doing so, it generates meshes that are numerically sound and that are also as small as possible. Applications of mesh generation include computer-aided design (CAD), fluid flow simulations, and rocket science.

Several meshing methods such as those based on advancing front, Delaunay triangulations, and quadtrees/octrees have become popular due to their effectiveness in practice. All these methods, however, come with varying strengths. For example, advancing front [4, 12, 13] uses simple data structures and is efficient and relatively easy to implement. It offers a high quality point-placement strategy and achieves the boundary integrity. Unfortunately, it does not provide any general guarantee on the size and quality of the mesh it produces. On the other hand, more sophisticated methods such as quadtree/octree refinement [2, 17] and Delaunay triangulation based methods [8, 15, 18, 19] generate a well-shaped mesh whose size is within a constant factor of the optimal. Recently the authors of this paper developed a new meshing algorithm, the *biting method*, that combines the strengths of advancing front and these provably good meshing methods [11]. We have shown that the biting method generates well-shaped meshes for two dimensional (2D) domains. In this paper, we extend the results to three dimensions.

We first give a high level outline of the three dimensional (3D) biting method, and summarize our new contributions.

The biting method uses a specific type of Delaunay based methods, the *sphere packing* [15, 16, 23], in conjunction with advancing front. It first constructs a *well-spaced* point set by computing a sphere packing of the domain and then uses the Delaunay triangulation of this point set as the final mesh. In other words, the biting method uses advancing front to generate a sphere packing as an alternative to the three previous techniques. The first one applies *particle simulation* [20, 21, 22] to find a stable configuration of a set of energetic spheres. The second one uses quadtree/octree refinement to

*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. Supported in part by an NSF CAREER award (CCR-9502540), an Alfred P. Sloan Research Fellowship, and the U.S. Department of Energy through the University of California under Subcontract number B341494 (DOE ASCI). The emails of authors: xli2@cs.uiuc.edu, steng@cs.uiuc.edu, ungor@cs.uiuc.edu.

†Part of this work has been done during the author visit to the MCS department of Argonne National Laboratory, Argonne, Il.

‡Part of this work has been done during the author visit to the Parallel Computing Sciences Dept., Sandia National Laboratories, Albuquerque, NM, 87185

obtain an oversample of the input domain, and then applies a properly defined maximal independent set to create the sphere packing [10, 16, 23]. Third approach is based on the refinement of Delaunay triangulation proposed by Ruppert [18] and extended to 3D by Shewchuk [19].

This new advancing front based packing algorithm first finds a sphere packing of the boundary of the domain and then grows the packing towards the interior of the domain. Each time when a new sphere is added to the interior, a larger protection sphere is removed (bitten away) from the domain so that no future sphere will overlap with this one. It builds the sphere packing by adding spheres one at a time, or a layer at a time, in the same spirit as the standard advancing front method; our new method uses advancing front to construct a sphere packing instead of the mesh elements themselves. It is shown that biting method does generate a well-spaced point set, i.e. a point set whose Delaunay triangulation is well-shaped. The biting scheme can be made as practical as the standard advancing front methods [11]. In this paper, we extend biting method into 3D and prove that it generates well-shaped meshes whose size is within a constant factor of the optimal possible. Main challenge in this extension is protecting the boundary in order to avoid bad elements near the boundary. Element shapes near the boundary are often most critical for the numerical methods.

The paper is structured as follows. Section 2 discusses the control spacing which specifies the element sizes and point densities of a mesh. Section 3 reviews the standard advancing front and the sphere packing methods. Section 4 reviews the biting method in 2D and difficulties extending it to 3D. We will present two 3D biting algorithms which with different boundary handling techniques. Section 5 gives the size and quality guarantee of the generated mesh. We also introduce the mesh conformity concept in Section 5, and show that biting method generates a mesh which conforms well to a given control spacing function. We conclude with a discussion of future work in Section 6.

2 Control Spacing Function

The geometry of a domain Ω and a differential equation u to be solved in this domain defines a desired local spacing. This local spacing specifies expected element size in a given neighborhood or point densities near a point. This section reviews the way to determine the local spacing from the geometry of Ω and the numerical condition of u. The reader is referred to [11] for a complete discussion of this derivation.

To specify the local spacing of a mesh based on the domain geometry, Ruppert [18] introduced a concept called *local feature size*. In 3D given a piecewise linear complex (PLC [16]) S, the *local feature size* at a point $\boldsymbol{x}$, or simply $lfs(\boldsymbol{x})$, is the radius of the smallest disk centered at $\boldsymbol{x}$ that intersects two non-incident vertices, segments, or facets of S.

The $lfs()$ function changes slowly within the domain. Formally, a function $f()$ is *Lipschitz with a coefficient* α if for any two points $\boldsymbol{x}$, $\boldsymbol{y}$ in the domain, $|f(\boldsymbol{x})-f(\boldsymbol{y})| \leq \alpha||\boldsymbol{x}-\boldsymbol{y}||$. Then the Lipschitz coefficient of lfs is bounded from above by 1 [18]. In addition, $lfs()$ is the maximum of all 1-Lipschitz functions over the same domain that satisfies the local spacing on the boundary [11].

Among several possible ways to describe the spacing function edge-length, $el()$, and nearest neighbor, $nn()$, are the most commonly used in the literature [18, 11]. $el(x)$ is the length of the longest edge of all mesh simplex elements that contain x. $nn(x)$ is the distance of x to the nearest mesh point if x is a mesh point and it is the distance to the second closest mesh point otherwise. For a well-shaped mesh, these two functions are related with each other [11].

The numerical condition is usually obtained from an *a priori* or an *a posteriori* error analysis based on an initial numerical simulation. It defines a *numerical spacing function*, denoted by $nsf(\boldsymbol{x})$, for each point $\boldsymbol{x}$ in the domain Ω. Computing the value of $nsf(\boldsymbol{x})$ for a point $\boldsymbol{x}$ is discussed in [11]. This function, however, may not satisfy the Lipschitz condition with a coefficient bounded by a constant. Li *et al.* [11] suggested to use the following 1-Lipschitz function to control the numerical spacing

$$\underline{nsf}(\boldsymbol{x}) = min(nsf(\boldsymbol{x}), min_{\boldsymbol{y}\in\Omega}(nsf(\boldsymbol{y})+||\boldsymbol{x}-\boldsymbol{y}||)).$$

This function is the best possible in the sense that for any 1-Lipschitz function g over the domain Ω, if $g(\boldsymbol{x}) \leq nsf(\boldsymbol{x})$ point-wise in Ω, then $g(\boldsymbol{x}) \leq \underline{nsf}(\boldsymbol{x})$ point-wise.

To capture both the geometric and the numerical requirements, a global spacing function, $gns()$

(geometric numerical spacing), is defined as

$$gns(\boldsymbol{x}) = min(lfs(\boldsymbol{x}), \underline{nsf}(\boldsymbol{x})).$$

This geometric numerical spacing function is also 1-Lipschitz [11].

Since exact computation of these spacing functions can be impractical, one can use interpolation of the discrete values on the vertices of a background mesh such as an octree decomposition of the domain.

3 Advancing Front and Sphere Packing Methods

Advancing front and sphere packing methods are surveyed in the original biting paper [11]. In this section, we review them briefly.

3.1 Advancing front methods

Advancing front methods construct a mesh of a domain by moving a front from its boundary towards its interior. It first generates an initial front typically by constructing a surface mesh for the boundary of the domain. It then creates new elements one at a time or a layer at a time and updates the front with these created faces [9, 12, 13]: In the one element-at-a-time model, it chooses a face of the current front and introduces a new mesh element with it as the base face. It can use another vertex on the front or insert a new Steiner point in the interior as the additional vertex of the new element. The base face and potentially some other faces on the front (if the additional vertex is an existing one) are removed from the front, and some faces of the new element are added to the front. This process is repeated until the front is empty, i.e., all fronts have merged upon each other and the domain is fully meshed.

The selection of the base face and the placement of the new mesh vertex are the two key steps of any advancing front method. These two steps must ensure that the new mesh element is valid and well-shaped and keep the front in good condition to allow the creation of graceful future elements. The faces of the clefts and the small faces are given priority to be picked as the base faces to satisfy these requirements.

Paving [4] is one of the popular advancing front methods for quadrilateral mesh generation. Attempts to extend the paving algorithm to three dimensions resulted a hexahedral meshing technique, *plastering* [5, 6]. As paving, plastering includes a number of basic operations (*row choice*, *closure check*, *row generation*, *smooth*, *seam*, *row adjustment*, *intersection*, and *cleanup*) to tightly control the moving front to ensure the mesh validity and quality.

Advancing front methods can be combined with Delaunay or quadtree/octree refinements. For instance, these refinement techniques can be used to generate a pretty-good domain decomposition of the input domain and then advancing front can be applied to get a mesh for each subdomain. We can also use quadtree/octree refinement to generate the set of points for the creation of the new elements.

3.2 Sphere packing methods

At a high level, the sphere-packing method fills an input domain with a set of spheres whose centers provide a good vertex set for a quality Delaunay mesh. It can be used to generate meshes for various quality conditions.

For example, Bern, Mitchell, and Ruppert [3] use sphere packing to triangulate an n-vertex polygonal region (potentially with holes) so that no element has angle larger than $\pi/2$. A similar sphere-packing based method has been developed by Bern and Eppstein [1] for quadrilateral meshes. Shimada and Gossard [21] have developed a sphere-packing method called *bubble mesh* to generate triangular meshes for two and three dimensions. Their packing scheme is based on the simulation of the particles that interact each other with repulsive/attractive forces. However, their method does not provide a theoretical bound on the time of the algorithm nor the quality of the mesh that they generate.

Miller *et al.* [15, 16] have designed a sphere-packing based meshing method which combines two well-known methods, quadtree and Delaunay refinements. We use their definition of sphere packing. Let $B(\boldsymbol{x}, r)$ denote the sphere centered at point $\boldsymbol{x}$ with radius r.

Definition 3.1 (β-Packing, [15]) *Let β be a positive real constant. A set S of spheres is a β-packing with centers P of Ω with respect to a spacing function $f()$ if*

- *For each point p of P, $B(p, f(p)/2) \in S$;*

- *The interiors of any two spheres s_1 and s_2 in S do not overlap; and*
- *For each point $q \in \Omega$, there is a sphere in S that overlaps with $B(q, \beta f(q)/2)$.*

In this paper, we measure the quality of a tetrahedra element by the *radius-edge ratio* as defined in [15, 16]. Based on this quality measure, we can define well-shaped meshes as the following.

Definition 3.2 (α-well-shaped mesh) *A mesh M is* α-well-shaped *for a constant $\alpha > 1$ if the minimum radius-edge ratio over all of its elements is bounded from above by α.*

Our method is similar to the method of Miller *et al.* [14] and uses their structure theorem on the equivalence of β-sphere packing and well-shaped meshes.

Theorem 3.1 (Sphere Packing, [15]) *The following two statements are true.*

(a) For any positive constant β, if $f()$ is a spacing function of Lipschitz constant 1 over a domain Ω and S is a β-packing with respect to $f()$, then the Delaunay triangulation M of the centers of S is a good-aspect-ratio mesh; in addition, for each point $\boldsymbol{p}$ in Ω, $nn_M(\boldsymbol{p}) = \Theta(f(\boldsymbol{p}))$, where the constant in Θ depends only on β.

(b) if M is a good-aspect-ratio mesh, then there exists a constant β such that the set of spheres

$$S = \{B(\boldsymbol{p}, nn_M(\boldsymbol{p})/2) : \forall \boldsymbol{p} \in M\},$$

is a β-packing with respect to $\frac{1}{2}nn_M()$.

Note that a good-aspect-ratio mesh is not necessarily a well-shaped mesh as it may include slivers. However, we use the recent algorithm by Cheng *et al.* [7] to eliminate slivers and generate a well-shaped mesh. Their algorithm uses weighted Delaunay triangulation and removes the slivers in a good-aspect-ratio mesh. Through the rest of this paper, we will use the terms good-aspect-ratio and well-shaped meshes interchangeably based on this sliver result.

4 Biting Scheme

This section starts with a review of biting method and its approach to boundary protection in 2D. Extending this approach to 3D is not straightforward. After discussing the challenging problem of boundary protection in 3D, two algorithms for generating a sphere packing protecting the boundaries are given.

4.1 Biting scheme in 2D

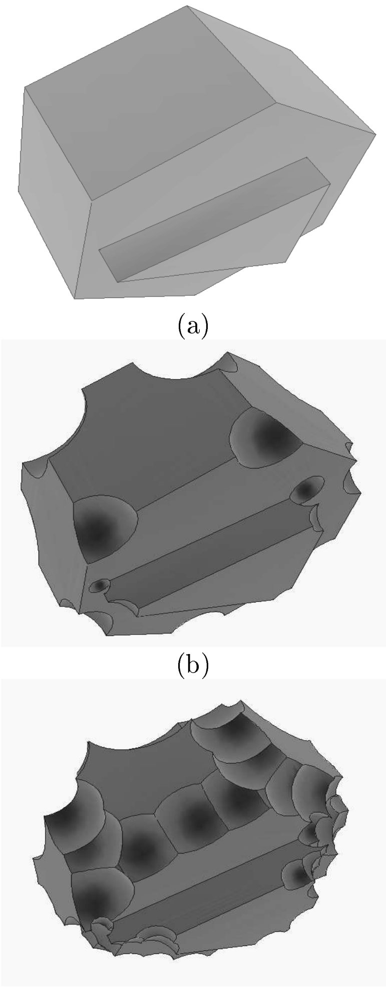
Biting method was introduced as a practical mesh generation method that constructs a well-shaped mesh using a sphere packing built by advancing front. Size and quality guarantee of the algorithm was proven in [11]. We also provided an angle bound for 2D domains discussing the boundary protection of the domain.

The biting method moves a front from the boundary of the domain to the interior and adds new mesh points in the process. These mesh points are chosen such that we can define a sphere packing with them as the centers. At each step, we place a new point on the current front rather than place it in the interior. Each time we add a point, we remove a sphere, the *biting sphere* from the remaining interior domain. The boundary between the union of biting spheres and the remaining interior domain defines the new front. This process is called *biting*. A biting sphere at a point $\boldsymbol{x}$ is $B(\boldsymbol{x}, c_b h(\boldsymbol{x}))$, where c_b is a constant. Note that the biting spheres are different from the packing spheres due to some technical reasons that are discussed in [11]. If the input domain has a small angle, we can cut off that small angle first to approximate the domain. Hence through the rest of the paper, we assume that the input domain does not have an acute input angle.

4.2 Biting scheme in 3D

Biting idea to construct a sphere packing applies to 3D as well. Going one dimension higher, we pack spheres rather than circles (2D spheres). Complication, however, arises in protecting the boundary.

In order to provide quality guarantee, one must ensure the quality of the elements incident to the boundary. For that purpose we can avoid the placement of mesh vertices too close to the boundary, in other words, to protect the boundary. In 2D this was achieved by checking intersection of the sphere



(a) (b) (c)

Figure 1: A snapshot of the *biting* scheme: (a) initial PLC domain; (b) bites on the vertices of the domain; (c) bites on the edges of the domain.

to be bitten with the spheres already bitten on the boundary. This is just to make sure no intersection occurs close to the boundary, as the intersection is most likely to be selected as a mesh vertex. If such an intersection occurs, we move the center of the biting sphere to a *midpoint* between the two already bitten spheres on the boundary. This just moves the intersection point away from the boundary (see [11] for details). This idea can not be easily extended to 3D because the notion of midpoint in 3D is not as clear as in 2D.

We provide two biting methods as they differ in their ways to handle the boundary. The first method uses classical advancing front techniques to construct a one layer inside the boundary to protect it and then start biting from that layer on. The second method starts biting directly from the boundary but avoids choosing the center of the bites (mesh vertices) which is close to the initial input boundary.

4.2.1 Peeling

First boundary protection method for biting, *peeling*, applies a classical advancing front that makes sure that there are no bad elements created along this layer. Then biting is applied to the interior without worrying about the boundary protection. We can use plastering [5] to construct the one layer that protects the boundary. The algorithm can be described formally as follows.

Algorithm `Biting via peeling`

1. [**Construct Spacing**]: Compute the control spacing function $h()$ of Ω by combining the local feature size and the numerical condition;
2. [**Boundary Protection**]: Apply plastering to construct a one layer around the boundary such that all the elements are well shaped. Plastering method as it is introduced [5] places hexahedral elements. However we are interested in the point set rather than the elements. Hence we can just consider the mesh vertices placed. Set the initial front for the next step as the interior faces of these hexahedras.
3. [**Interior Biting**]: Choose a vertex $\boldsymbol{x}$ on the front and remove its biting sphere.

   Modify the front by introducing the sphere caps of the new biting sphere and removing the intersection of it with the front.

   Repeat until the advancing front is empty.
4. Construct the Delaunay triangulation of the centers of the biting spheres as the final mesh.

4.2.2 Leveling

This approach starts with 2D boundary protection, biting from vertices and edges first. We continue biting spheres centered on the surface of the domain until the whole surface is bitten, without considering the intersection of spheres unlike the 2D boundary protection. Instead we pay attention not to choose the sharp noses that are too close to the boundary as the center of the biting sphere when we continue biting the interior region.

For protecting the input boundary, in the following leveling approach, we make sure that all interior mesh vertices are far enough from the boundary. In other words, the mesh has the following property: for every vertex $\boldsymbol{x}$, $h_{\boldsymbol{x}} \geq cf(\boldsymbol{x})$, where c is a constant to specified later, $h_{\boldsymbol{x}}$ is the distance of $\mathbf{x}$ to the closest boundary.

Algorithm `Biting via leveling`

1. [**Construct Spacing**]: Compute the control spacing function $h()$ of Ω by combining the local feature size and the numerical condition;

2. Let the boundary of the domain be the initial front, (see Figure 1 (a));

3. [**Vertex Protection**]: Bite all the input vertices by removing their biting spheres from the interior of the domain, (see Figure 1 (b));

   Modify the front which becomes a set of segments and sphere caps. Segments are represented by the endpoints and sphere caps are represented by the centers of the biting spheres.

4. [**Edge Protection**]: Bite spheres centered on the input boundary edges: choose a vertex $\boldsymbol{x}$ on the front and remove its biting sphere. Whenever possible, we choose $\boldsymbol{x}$ on the intersection of some bitten spheres with the initial boundary edge, (see Figure 1 (c)).

   We apply the same boundary edge protection as in 2D [11].

   Modify the front by introducing the sphere caps of the new biting spheres and removing the intersection of it with the front.

   Repeat until all initial input boundary edges are bitten;

5. [**Face Biting**]: Bite spheres centered on the input boundary faces: choose a vertex $\boldsymbol{x}$ on the front and remove its biting sphere. Whenever possible, we choose $\boldsymbol{x}$ on the intersection of some bitten spheres with the initial boundary face. This step essentially is same as 2D Biting. In our previous work, the quality of the 2D mesh has already been proven [11]. We have to pay attention, however, not to introduce problems for the 3D mesh. In other words, after all the boundary faces are bitten, intersection of any three biting sphere should be far from boundary as this intersection is a candidate mesh vertex in our scheme.

6. [**Boundary Protection**] After biting the input boundaries, let $\boldsymbol{x}$ be the current sharp nose for the candidate of the biting center. Note that here $\boldsymbol{x}$ is not on a boundary; it is the intersection of at least three spheres centered on the input boundary face of the domain. If $h_{\boldsymbol{x}} \geq cf(\boldsymbol{x})$, then just bite the point $\boldsymbol{x}$. Otherwise, we level it up to a new point $\boldsymbol{y} = \boldsymbol{x} + c_b/(1 + \alpha c_b) f(\boldsymbol{x})\overline{\boldsymbol{u}}$, where $\overline{\boldsymbol{u}}$ is the normal direction of the closest boundary (see Figure 2). Then we bite the sphere centered at point $\boldsymbol{y}$ instead of $\boldsymbol{x}$. As to be discussed later, with appropriate value of c_b, we can guarantee that nose is bitten and the point $\boldsymbol{y}$ is far enough from the boundary.

7. [**Interior Biting**]: After all the noses that are created by the spheres whose center are on the boundary are either bitten (with or without leveling), we start the interior biting. Choose a vertex $\boldsymbol{x}$ on the front and remove its biting sphere. Note that leveling is not needed for interior biting;

   Modify the front by introducing the sphere caps of the new biting sphere and removing the intersection of it with the front.

   Repeat until the advancing front is empty.

8. Construct the Delaunay triangulation of the centers of the biting spheres as the final mesh.

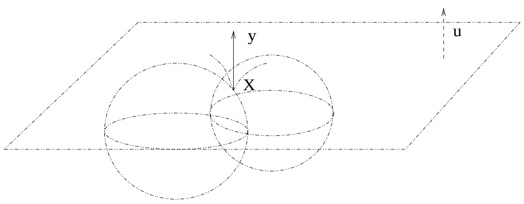


Figure 2: Leveling: If the candidate biting point $\boldsymbol{x}$ is too close to the boundary, then we choose point $\boldsymbol{y}$ by lifting $\boldsymbol{x}$ along the normal direction of the boundary hyper-plane.

The basic idea of the biting method is to first compute the control spacing function of the mesh. We then try to find a point set by constructing a sphere packing with respect to the spacing function. For making sure that the biting process results in a sphere packing, we use the following simple idea: at every step, we choose a center on the advancing front and remove its biting sphere. The removal of its biting sphere ensures that the future packing spheres will not intersect with the packing sphere of this center. The Delaunay triangulation is then used to generate the mesh from the resulting sphere packing.

Usually, the advancing front is represented as a circular list of already placed points. In our method, it is represented as a set of spherical caps and boundary segments. We choose the next Steiner point on the front itself unless it requires leveling. In other words, the front itself is a subset of the feasible region for the selection of new mesh vertices, making it easier to choose the next point.

The intersection of three caps or two caps and a boundary face provides a good candidate for a new Steiner point, whose biting sphere will reduce the interior. *This is actually the way we take a bite on a biscuit or an apple. We center our bite more or less around a sharpest nose. Then bite after bite, we eat away the boundary of the food and move to its interior.*

5 Size and Quality Guarantee

It is shown that the biting method generates a well-shaped mesh in 2D [11]. Moreover, the size of this mesh is within a constant factor of the optimal. These statements extends to 3D, by considering sphere packing rather than circle packing. The first statement is proven by showing that the points placed by the biting method is well-spaced, i.e., they are centers of a β-packing with respect to a 1-Lipschitz spacing function. As it is shown by Talmor [23] even well-spaced point set do not prevent slivers. We use the sliver exudation algorithm of Cheng et al. [7] to get a well-shaped mesh from the good-aspect-ratio mesh. The size optimality follows from the fact that the spacing function is maximal.

5.1 Good Sphere Packing

The spheres that are used for biting are different from the spheres that are used for packing. For each point that the biting scheme generates, we would like to define a *packing sphere*. Because the biting spheres defined by our scheme overlap among themselves, the packing sphere of a point is chosen to be smaller than its biting sphere. Let us focus on a particular point $\boldsymbol{x}$. From the specification of the biting scheme, the biting sphere at $\boldsymbol{x}$ is $B(\boldsymbol{x}, c_b h(\boldsymbol{x}))$, where $c_b < 1$ is a positive constant. We now choose another positive constant $c_p < c_b$, and define the packing sphere at $\boldsymbol{x}$ to be $B(\boldsymbol{x}, c_p h(\boldsymbol{x}))$.

In principle, in the biting scheme, c_b is the largest constant that satisfies the constraint of the control spacing function so that the resulting mesh has an optimal size. Similarly, c_p is the largest constant to ensure no two packing spheres overlap so that the gaps among the packing spheres are minimized. The relation between c_b and c_p is discussed in the original paper [11]. Based on their relation it is shown that for appropriate biting constant c_b, we can set the packing constant $c_p = \frac{c_b}{2+\alpha c_b}$. Then similarly we have the following theorem

Theorem 5.1 *The biting scheme constructs $\beta = \frac{1+\alpha c_b}{1-\alpha c_b}$-packing with respect to the spacing function $\frac{2c_b}{2+\alpha c_b} f()$.*

This together with Theorem 3.1 is used to prove the following theorem.

Theorem 5.2 ([11]) *The biting method generates a well-shaped mesh.*

Note that both biting methods, peeling and leveling, will generate well-shaped meshes. Peeling is actually the combination of the plastering and the interior biting. The interior biting generates a good sphere packing [11]. The plastering method also generates a good one layer point placement to protect the boundary.

Leveling is the pure biting method which protects the boundary by leveling up any biting center in addition to generates well spaced and well conformed mesh vertices.

5.2 Radius-Edge Ratio

We have shown that the biting scheme generates a well shaped mesh. However, the constant bound on the radius-edge ratio so derived may be too large. In this section, we provide a better analysis of the upper bound of the radius-edge ratio.

In our analysis, we divide the tetrahedral elements into two subsets: the first subset contains all elements that are visible from their circumcenters and the second subset contains all elements that are not completely visible from their circumcenters, i.e., elements that are close to the boundary. Assume that the spacing function $f()$ used by the biting scheme is α-Lipschitz. Let $t = \alpha c_b$. We use t to control the quality and the size of the mesh generated by the biting scheme. In our proof, we assume $t \le 1/3$, although for a practical scheme, we can set t to $1/2$. The following lemma bounds the radius edge ratio of the tetrahedra in the first subset.

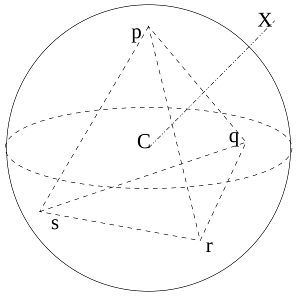


Figure 3: The circumcenter of the tetrahedra is inside the domain.

Lemma 5.3 *Let $\triangle_{\boldsymbol{pqrs}}$ be a tetrahedra element of the first subset. Let l be the length of the shortest edge of $\triangle_{\boldsymbol{pqrs}}$. Let R be the radius of the circumsphere $\mathcal{C}$ of $\triangle_{\boldsymbol{pqrs}}$. Then*

$$\frac{R}{l} \le \frac{1}{1-2t}. \quad (1)$$

Proof: : Let $\boldsymbol{c}$ be the circumcenter of $\triangle_{\boldsymbol{pqrs}}$. Assume $\boldsymbol{c} \in B(\boldsymbol{x}, c_b f(\boldsymbol{x}))$. Then $||\boldsymbol{c} - \boldsymbol{x}|| \le c_b f(x)$. Because the mesh is a Delaunay triangulation, $\boldsymbol{x}$ is not in the interior of circumsphere of $\triangle_{\boldsymbol{pqrs}}$, i.e., $||\boldsymbol{x} - \boldsymbol{c}|| \ge R$. Thus $f(\boldsymbol{x}) \ge R/c_b$. Because $f()$ is α-Lipschitz, $f(\boldsymbol{c}) \ge f(\boldsymbol{x}) - \alpha||\boldsymbol{c} - \boldsymbol{x}||$. Also because $||\boldsymbol{c} - \boldsymbol{x}|| \le c_b f(x)$, $f(\boldsymbol{c}) \ge (1 - \alpha c_b) f(\boldsymbol{x}) \ge (1 - \alpha c_b) R/c_b$.

Let's assume, without loss of generality, that $||\boldsymbol{p} - \boldsymbol{q}|| = l$, and $B(\boldsymbol{p}, c_b f(\boldsymbol{p}))$ is bitten before $B(\boldsymbol{q}, c_b f(\boldsymbol{q}))$, implying $l \ge c_b f(\boldsymbol{p})$. Because $f()$ is α-Lipschitz, $f(\boldsymbol{c}) \le f(\boldsymbol{p}) + \alpha R \le l/c_b + \alpha R$. Using $f(\boldsymbol{c}) \ge (1 - \alpha c_b) R/c_b$, we have $(1-t)R/c_b \le (l + tR)/c_b$. Then the lemma follows. $\square$

For the tetrahedras in the second subset, we can apply the similar approach to prove that the radius-edge ratio is not worse than this. Notice that for every vertex $\boldsymbol{x}$, $h_{\boldsymbol{x}} \ge c f(\boldsymbol{x})$, where c is a constant to specified in the next subsection, $h_{\boldsymbol{x}}$ is the distance of $\mathbf{x}$ to the closest boundary.

5.3 Boundary Protection

We show that the mesh vertices generated by the biting scheme are not too close to the boundary with respect to its control spacing. For the peeling method, it is the result of the nice boundary protection of the plastering method [5]. For the leveling method, it is guaranteed by our scheme. However, we need determine the constant c in the leveling scheme. Notice that the biting sphere centered on point $\boldsymbol{y}$ leveled from point $\boldsymbol{x}$ should cover the point $\boldsymbol{x}$. See scheme leveling 4.2.2. Otherwise, point $\boldsymbol{x}$ is still remain a candidate for the biting center.

Lemma 5.4 (leveling factor) *If $||\boldsymbol{y} - \boldsymbol{x}|| = c_b/(1 + \alpha c_b) f(x)$, then the sphere $B(\boldsymbol{y}, c_b f(\boldsymbol{y}))$ covers point $\boldsymbol{x}$.*

Proof: : Noting that the control spacing $f()$ is α-Lipschitz, $c_b f(\boldsymbol{y}) \ge c_b f(\boldsymbol{x}) - \alpha c_b ||\boldsymbol{y} - \boldsymbol{x}||$. From $||\boldsymbol{y} - \boldsymbol{x}|| = c_b/(1 + \alpha c_b) f(x)$, we have $c_b f(\boldsymbol{y}) \ge c_b/(1 + \alpha c_b) f(x) = ||\boldsymbol{y} - \boldsymbol{x}||$. Then the lemma follows, i.e., $\boldsymbol{x}$ is in $B(\boldsymbol{y}, c_b f(\boldsymbol{y}))$. $\square$

Then we can choose $c_b/(1 + 2\alpha c_b)$ as the constant c in the leveling scheme.

Lemma 5.5 (Boundary Protection) *After leveling, for every vertices $\boldsymbol{y}$ of the mesh, $h_{\boldsymbol{y}} \ge f(\boldsymbol{y}) c_b/(1 + 2\alpha c_b)$.*

Proof: We just need prove for the vertices $\boldsymbol{y}$ leveled up from some point $\boldsymbol{x}$ which are close to the boundary. From lemma 5.4, we know that $||\boldsymbol{y} - \boldsymbol{x}|| = c_b/(1 + \alpha c_b) f(\boldsymbol{x})$. From the α-Lipschitz condition, we have $f(\boldsymbol{y}) \le f(\boldsymbol{x}) + \alpha||\boldsymbol{y} - \boldsymbol{x}||$, which implies that $f(\boldsymbol{y}) \le (1 + 2\alpha c_b)/(1 + \alpha c_b) f(\boldsymbol{x})$. So

$$\begin{aligned} h_{\boldsymbol{y}}/f(\boldsymbol{y}) &\ge ||\boldsymbol{y} - \boldsymbol{x}||/f(\boldsymbol{y}) \\ &\ge c_b/(1 + 2\alpha c_b). \end{aligned}$$

$\square$

Hence for all vertices in the mesh generated by leveling, we show that they are not too close to the boundary.

5.4 Spacing Conformity

Given the control spacing specification, the generated mesh should conform well to it in addition to the fact that the mesh should be well-shaped. In the ideal mesh, the spacing function defined for any mesh vertex should be within a constant factor of the control spacing. The smaller the constant, the better conformed the spacing. Thus, we define the conformity of the mesh as following, which captures how well the mesh conform to the given control spacing.

Definition 5.1 (Conformity) *Let $\boldsymbol{x}$ be a mesh vertices of the mesh; let $nn(\boldsymbol{x})$ be its nearest neighbor value. If $nn(\boldsymbol{x}) \leq f(\boldsymbol{x})$, let $c(\boldsymbol{x}) = nn(\boldsymbol{x})/f(\boldsymbol{x})$, otherwise $c(\boldsymbol{x}) = f(\boldsymbol{x})/nn(\boldsymbol{x})$. Then $c(\boldsymbol{x})$ is defined as the conformity of vertex $\boldsymbol{x}$.*

Definition 5.2 [**γ-Well-Conformed Mesh**] *A mesh is γ-well-conformed if every vertex of the mesh has conformity at least γ.*

In order to prove the size optimality, we show that the nearest neighbor value of any point in the domain is related to the control spacing function $f()$ by a constant factor. The proof of the above statement requires different arguments for the mesh vertices and any other point in the domain as stated by two lemmas in [11]. Similar to [11], we have for any mesh vertex $\boldsymbol{x}$, $c_b(1-\alpha c_b)f(\boldsymbol{x}) \leq nn(\boldsymbol{x}) \leq \frac{2c_b}{1-\alpha c_b} f(\boldsymbol{x})$, which implies the following theorem

Theorem 5.6 *The biting methods generates a mesh with conformity $\frac{1}{c_b(1-\alpha c_b)}$.*

Moreover, the following lemma show that the edge length function *el* is also within a small constant factor of the spacing function.

Lemma 5.7 *For any mesh vertices $\mathbf{x}$ of the mesh M generated by our biting method, we have*

$$el\,(\mathbf{x}) \leq \frac{2c_b}{1-2\alpha c_b} f\,(\mathbf{x})\,.$$

where $el\,(\mathbf{x})$ is the length of the longest edge incident to $\mathbf{x}$.

Proof: We consider any mesh vertex $\mathbf{x}$ of the mesh M generated by our biting method. Let $\nabla\mathbf{xyzw}$ be any tetrahedra of the mesh M such that edge $\mathbf{xy}$ is the longest edge incident to $\mathbf{x}$. Let $\mathbf{c}$ be the center of circumcircle of tetrahedra $\nabla\mathbf{xyzw}$. Let R be the radius of its circumcircle.

If point $\mathbf{c}$ is inside the circle $B(\mathbf{x}{,}c_b f(\mathbf{x}))$, then we have

$$el\,(\mathbf{x}) = |\mathbf{x}-\mathbf{y}| \leq 2R \leq 2c_b f(\mathbf{x}).$$

Otherwise, we know that point $\mathbf{c}$ is covered by at least one biting circle, say $B(\mathbf{q}{,}c_b f(\mathbf{q}))$. Notice that from the Delaunay triangulation property, we know that $|q-\mathbf{c}| \geq R$. Then $c_b f(\mathbf{q}) \geq |q-\mathbf{c}|$. As $f()$ is 1-Lipschitz function, we have $f(\mathbf{c}) + \alpha\,|q-\mathbf{c}| \geq f(\mathbf{q})$. Thus $c_b\,(f(\mathbf{c}) + \alpha\,|q-\mathbf{c}|) \geq c_b f(\mathbf{q}) \geq |q-\mathbf{c}|$, which implies that

$$f(\mathbf{c}) \geq \frac{1-\alpha c_b}{c_b}\,|q-\mathbf{c}| \geq \frac{1-\alpha c_b}{c_b} R.$$

From $f(\mathbf{c}) \leq f(\mathbf{x}) + \alpha R$, we have

$$f(\mathbf{x}) \geq \frac{1-2\alpha c_b}{c_b} R.$$

Thus

$$el\,(\mathbf{x}) = |\mathbf{x}-\mathbf{y}| \leq 2R \leq \frac{2c_b}{1-2\alpha c_b} f(\mathbf{x}).$$

Then the lemma follows. □

Then using the size optimality lemma of Miller *et al.* [14] the following theorem can be established.

Theorem 5.8 ([11]) *Size of the mesh generated by the* biting *method is within a constant factor of the optimal possible.*

Notice that unlike the theorem proved by Ruppert [18] or [14], the hidden constant by our biting method is much small, thanks to Lemma 5.7.

6 Conclusion

This paper presents a quality tetrahedral mesh generation scheme. As it is introduced in [11] our scheme, the biting method, first applies some variations of an advancing front method to generate a quality sphere packing; it then constructs the final mesh by Delaunay triangulation. Therefore, it is as simple as and as practical as the advancing front methods.

Biting method overcomes the difficulty that the standard advancing front methods have in the region where the fronts meet. By constructing sphere packings, it generates provably good mesh.

The advancing front method generates a quality sphere packing more efficiently than previously proposed methods such as those that use the particle simulation [21] or maximal independent set (MIS) of oversampled points [15]. The biting scheme is often more efficient than the paving method because it explicitly maintains the set of candidates for new mesh points.

In our ongoing implementation of the biting algorithm, we use the biting of squares instead of spheres. The first reason is that computing the intersection of cubes is easier than that for spheres; the second reason is to maintain the advancing front resulted from the intersection of cubes is easier than to maintain that resulted from the intersection of the spheres. Notice that the biting cubes is the approximation of the biting sphere.

References

[1] Bern, M., and Eppstein, D. Quadrilateral meshing by circle packing. In 6^{th} *International Meshing Roundtable* (1997), pp. 7–20.

[2] Bern, M., Eppstein, D., and Gilbert, J. Provably good mesh generation. In *Proc. 31st IEEE Symp. Foundations of Computer Science* (1990), pp. 231–241. To appear in *J. Comp. System Science*.

[3] Bern, M., Mitchell, S., and Ruppert, J. Linear-size nonobtuse triangulation of polygons. In *Proc. 10th ACM Symp. Comp. Geometry* (1994), pp. 221–230.

[4] Blacker, T. D. Paving: a new approach to automated quadrilateral mesh generation. *Int. Jour. for Numerical Methods in Eng 32* (1991), 811–847.

[5] Blacker, T. D., and Meyers, R. Seams and wedges in plastering: A 3d hexahedral mesh generation algorithm. *Engineering With Computers 2* (1993), 83–93.

[6] Canann, S. *Plastering and Optismoothing: New Approaches to Automated, 3D Hexahedral Mesh Generation and Mesh Smoothing.* PhD thesis, Brigham Young University, Provo, UT, 1991.

[7] Cheng, S.-W., Dey, T. K., Edelsbrunner, H., Facello, M. A., and Teng., S.-H. Sliver exudation. In *Proc.* 15^th *ACM Symp. Comp. Geometry* (1999).

[8] Chew, L. Guaranteed-quality delaunay meshing in 3d (short version). In *Proc.* 13^{th} *ACM Symp. Comp. Geometry* (1997), pp. 391–393.

[9] Frey, P. J., Borouchaki, H., and George, P. L. Delaunay tetrahedralization using an advancing-front approach. In 6^{th} *International Meshing Roundtable* (1997).

[10] Li, X.-Y., Teng, S.-H., and Üngör, A. Simultaneous refinement and coarsening: adaptive meshing with moving boundaries. In 7^{th} *International Meshing Roundtable* (1998), pp. 201–210.

[11] Li, X.-Y., Teng, S.-H., and Üngör, A. Biting: Advancing front meets sphere packing. In 2^{nd} *Symp. on Trends in Unstructured Mesh Generation* (1999).

[12] Löhner, R. Progress in grid generation via the advancing front technique. *Engineering with Computers 12* (1996), 186–210.

[13] Löhner, R., and Parikh, P. Three dimensional grid generation by the advancing-front method. *Int. J. Numer. Meth. Fluids 8* (1988), 1135–1149.

[14] Miller, G. L., Talmor, D., and Teng, S.-H. Data generation for geometric algorithms on non-uniform distributions. *International Journal of Computational Geometry and Applications* (1998).

[15] Miller, G. L., Talmor, D., Teng, S.-H., and Walkington, N. A Delaunay based numerical method for three dimensions: generation, formulation, and partition. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing* (Las Vegas, Nevada, 29 May–1 June 1995), pp. 683–692.

[16] Miller, G. L., Talmor, D., Teng, S.-H., Walkington, N., and Wang, H. Control volume meshes using sphere packing: generation, refinement, and coarsening. In 5^{th} *International Meshing Roundtable* (Oct. 1996), pp. 47–61.

[17] Mitchell, S., and Vavasis, S. Quality mesh generation in three dimensions. In *Proc. 8th ACM Symp. Comp. Geom.* (1992), pp. 212–221.

[18] Ruppert, J. A new and simple algorithm for quality 2-dimensional mesh generation. In *Proc. 4th ACM-SIAM Symp. on Disc. Algorithms* (1993), pp. 83–92.

[19] Shewchuk, J. R. Tetrahedral mesh generation by delaunay refinement. In 14^{th} *Annual ACM Symposium on Computational Geometry* (1998), pp. 86–95.

[20] Shimada, K. *Physically-based Mesh Generation:Automated Triangulation of surfaces and Volumes via Bubble Packing.* PhD thesis, MIT, 1993.

[21] Shimada, K., and Gossard, D. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere-packing. In 3^{rd} *Symp. on Solid Modeling and Appl* (1995), pp. 409–419.

[22] Shimada, K., Yamada, A., and Itoh, T. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In 6^{th} *International Meshing Roundtable* (1997), pp. 375–390.

[23] Talmor, D. *Well-Spaced Points for Numerical Methods.* PhD thesis, Carnegie Mellon, 1997.

[24] Teng, S.-H., and Wong, C. Unstructured mesh generation: Theory, practice, and perspectives. *International Journal of Computational Geometry and Applications* (1999).