Generate Good Triangular Meshes, Conforming to Control Spacing Requirements

Xiang-Yang Li[1] **Shang-Hua Teng**[2] **Peng-Jun Wan** [3]

Abstract: To conduct numerical simulations by finite element methods, we often need to generate a high quality mesh, yet with a smaller number of elements. Moreover, the size of each of the elements in the mesh should be approximately equal to a given size requirement. Li *et al.* recently proposed a new method, named *biting*, which combines the strengths of advancing front and sphere packing. It generates high quality meshes with a theoretical guarantee. In this paper, we show that biting squares instead of circles not only generates high quality meshes but also has the following advantages. It is easier to generate high quality elements near the boundary with theoretical guarantee; it is very efficient time-wise; in addition, it is easier to implement. Furthermore, it provides simple and straightforward boundary protections in three dimensions.

keyword: Unstructured mesh generation, advancing front, biting, sphere packing, spacing function.

1 Introduction

In numerical simulations or computer graphics applications, we often need to decompose a domain into a collection of primitive elements. Not all meshes play equally well for numerical simulations. It is often the case that we are required to generate high quality meshes with small number of elements. In order to reduce the problem size, an *unstructured mesh* with a varying local topology and spacing is often used for problems with complex shape boundaries and with solutions that change rapidly. Over the years, several meshing methods have been developed to generate high quality triangular meshes in 2D or 3D. See Bern, Eppstein, and Gilbert (1990); Li and Teng (1998); Mitchell and Vavasis (1992); Ruppert (1992); Shewchuk (1998). Those based on advancing front, Delaunay triangulations, and quadtrees/octrees have become popular due to their effectiveness in practical applications. However, these methods do not come with equal strengths. For example, the advancing front method offers a high quality of vertex placement strategy and an integrity of the boundary. See Blacker (1991); Lohrer (1996); Lohrer and Parikh (1988). Unfortunately, it does not provide general guarantees on the size and quality of meshes it produces. Especially, it is hard or time consuming to process when the fronts meet each other or one front meets itself. On the other hand, more sophisticated methods such as quadtree/octree refinement, [Bern, Eppstein, and Gilbert (1990); Mitchell and Vavasis (1992)], and Delaunay based methods, [Chew (1997); Miller, Talmor, Teng, and Walkington (1995); Ruppert (1992); Shewchuk (1998)], guarantee to generate a well-shaped mesh such that the number of elements is within a constant factor of the optimal.

For numerical simulations, we are often required to generate a mesh whose element size is no larger than an element size specified by a function, namely, *control spacing function*. Some methods, such as Delaunay refinement, are not explicitly designed for generating a high quality mesh conforming to a spacing function. Recently, the authors had developed a new two-dimensional meshing algorithm called *biting*; see Li, Teng, and Üngör (2000). The algorithm combines the strengths of advancing front and these provably good meshing methods. It generates a high quality mesh conforming to a given control spacing function.

The biting algorithm uses the sphere packing [4] as the underlying structure, in conjunction with the advancing front method. The biting method first constructs a *well-spaced* vertex set by implicitly building a sphere packing of the domain and then uses the Delaunay triangulation of this vertex set as the final mesh. They use a new and efficient method to construct an almost tight sphere packing. At a high level, this new advancing-front-based packing algorithm first finds a sphere packing of the domain boundary and then grows the packing towards the interior of the domain. The packing is constructed implicitly as follows. For each point in the domain, it defines a biting sphere, with a radius proportional to the control spacing defined on it. The algorithm selects a corner vertex from the current domain boundary and removes its biting sphere from the domain. The domain boundary is updated and the algorithm is repeated until the domain is empty. They show that the set of *smaller spheres* centered at all biting centers is a sphere packing. In other words, the new method uses the advancing front to construct a sphere packing implicitly. Li, Teng, and Üngör (2000) showed that the Delaunay triangulation of the centers of all bitten spheres has small radius-edge ratio. Recall that, here the radius-edge ratio of an element is the ratio of its cir-

[1] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. And Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616.

[2] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. And Akamai Technologies Inc., 201 Broadway, Cambridge, MA 02319.

[3] Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616.

[4] In this paper, we use sphere as a more general terminology. It represents circle in two dimensions.

cumradius over the length of its shortest edge. The radius-edge ratio of a mesh is the largest radius-edge ratio of all of its elements. Furthermore, they showed that the size of each mesh element is within a small constant factor of the control spacing. Consequently, the number of mesh elements is within a small constant factor of any mesh that has small radius-edge ratio and conforms to the control spacing.

In this paper, we show that biting square (or cube in 3D) instead of sphere will also generate a high quality mesh whose size is within a constant factor of the optimal. In addition, the time complexity of two dimensional square-biting method is $O(n\log n)$, where n is the number of the output vertices. Furthermore, it is easier to implement this method than the previous sphere-biting method. It is also straightforward to extend the square-biting scheme to three dimensions without difficulty in protecting the domain boundary.

The rest of the paper is organized as follows. Section 2 introduces definitions that will be used in this paper. Section 3 presents the biting-square method and the proofs about the quality of the sphere packing that it generates. Section 4 gives the details of boundary protection to complete the proof of our main theorem. We analyze the time complexity of the square-biting method in Section 5. Section 6 gives some experimental results to show that the square-biting method actually generates well-shaped meshes. We conclude our paper in Section 7.

2 Preliminary

In this section, we review some definitions and results which are essential in presenting our algorithm and in proving the quality guarantees of the algorithm. Most of them can be found in many places, but for the completeness of the presentation, we still include them here.

2.1 Well-shaped and Well-conformed Mesh

In two dimensions, we assume that the input domain Ω is a *planar-straight-line graph* (PSLG), which is a collection of line segments and points in the plane, closed under intersection. If a control spacing function $f(x)$ over Ω is given, the generated mesh should conform well to $f(x)$ in addition to be well-shaped.

Quality measures. Different numerical methods and scientific problems may have different quality requirements on the underline meshes. Notice that, for very slowly varying functions, an essential quality requirement of the mesh is that all angles are not obtuse. On the other hand, another common shape criterion for mesh elements (triangle in 2D, tetrahedron in 3D) is the condition that the angles of each element are not too small, i.e., the aspect ratio of each element is bounded from above by a constant. See Babuška and Aziz (1976); Bern, Eppstein, and Gilbert (1990); Strang and Fix (1973). The aspect ratio of an element is often defined as the ratio of the radius of the smallest sphere containing the element to the radius of the largest sphere contained in the element. The aspect ratio of a mesh is defined as the largest aspect ratio of all of its elements. Unfortunately, until now, no method guarantees to generate three-dimensional Delaunay meshes with practically good small aspect ratio. An alternative but weaker quality measurement is to use the *radius-edge ratio* introduced by Miller, Talmor, Teng, and Walkington (1995, 1998).

In three dimensions, slivers are the only tetrahedra which have small radius-edge ratio but very large aspect ratio. Here, a sliver is a tetrahedron whose four vertices lie close to a plane and whose projection to that plane is a convex quadrilateral with no short edge. Notice that, in the recent breakthrough, Li (2000) developed a new refinement based algorithm that guarantees to generate meshes with bounded aspect ratio, i.e., sliver-free Delaunay meshes. However, the theoretical bound on the aspect ratio is large, although theoretically it is a constant. Experimental results are necessary to show the practical advantages of that algorithm.

Delaunay triangulation is widely used on mesh generation because it often provides a bridge to prove the theoretical quality guarantees of meshing algorithms. A simplicial mesh is Delaunay triangulation if the circumsphere of each of its elements does not contain any vertices inside it. The Delaunay triangulation of a set of two-dimensional vertices maximizes the minimum angle among all possible triangulations. Unfortunately, this is not true in three dimensions.

Based on the radius-edge ratio quality measure, we define well-shaped meshes as following.

Definition 2.1 [ρ-WELL-SHAPED MESH] *A mesh M is* ρ-well-shaped *if the maximum radius-edge ratio over all of its elements is bounded from above by* ρ.

Spacing function. A spacing function $f(x)$ is used to specify the ideal element size at every point of the domain Ω. The spacing function is typically defined from the geometry structure of the domain and/or the numerical system to be conducted on the domain. Let's first study what is the role of geometry structure of the domain in generating well-shaped meshes over it. Consider an input domain Ω and a well-shaped mesh M generated on Ω. The element of M could not have arbitrary size anywhere. For example, assume that one region of Ω has two closed vertices or segments. Then the local mesh elements in that region could not be too large. In other words, the geometrical structure of the domain more or less defines the element size of any well-shaped mesh that could be generated. Ruppert (1992) introduced the concept called *local feature size* function $\mathbf{lfs}(x)$ to capture this geometry condition. It is the radius of the smallest sphere centered at x that contains two non-incident features of the domain. Here the features are the input vertices and segments in two dimensions.

If the mesh is used for scientific computing, the numerical condition also determines the largest element size at each point such that the numerical error is not large. This element size specification is usually obtained from an *a priori* error analysis, or an *a posteriori* error analysis based on an initial numerical simulation. In other words, it defines a *numerical spacing functions*, denoted by $\mathbf{nsf}(x)$, for each point x in the domain Ω. The value of $nsf(x)$, from the interpolation viewpoint, is determined by the eigenvalues of the Hessian matrix H; see Strang and Fix (1973). The spacing of the mesh vertices, required by the accuracy of the discretization near x should depend on the reciprocal of the square root of the largest eigenvalues of H at x. Generally, the control spacing function of Ω is the combination of the local feature size $lfs(x)$ and the numerical condition $nsf(x)$.

To make it possible that we can generate a well-shaped mesh that conforms to a given control spacing $f(x)$, we expect to see some smoothness condition of $f(x)$. The following Lipschitz condition are often used to capture the smoothness of $f(x)$.

Definition 2.2 [α-LIPSCHITZ] *A function $f(x)$ is α-Lipschitz, if for any two points x, y of the domain*

$$|f(x) - f(y)| \leq \alpha ||x - y||.$$

Element size. Given a mesh M, without doubt, we need to describe the element size so we can measure its conformity to a given spacing function. There are several ways to describe the elements size of M. Edge length function $el(x)$ and nearest neighbor function $N(x)$ are two of the widely used ones. For each point $x \in \Omega$, $el(x)$ is the length of the longest edge of all mesh elements that contain x; while $N(x)$ is the distance of x to the second nearest mesh vertex in M. Notice that if x is a mesh vertex, then x itself is the nearest mesh vertex of x.

Given the elements size specification $f(x)$, the generated mesh should conform well to it in addition to be well-shaped. In the ideal mesh M, the elements size derived at any vertex of M should be within a small constant factor of the control spacing $f(x)$. The smaller the constant, the better conformed the control spacing $f(x)$. Thus, we define the conformity of a mesh as following.

Definition 2.3 [Conformity] *Let x be a vertex of mesh M; let $N(x)$ be its nearest neighbor value derived from M. We call $c(x) = \min(\frac{N(x)}{f(x)}, \frac{f(x)}{N(x)})$ the conformity of vertex x.*

Then the conformity of a mesh M is defined as following.

Definition 2.4 [γ-Well-Conformed Mesh] *A mesh is γ-well-conformed if every vertex of the mesh has conformity at least γ.*

In this paper, we consider the following problem. Given a domain Ω and an α-Lipschitz spacing function $f(x)$, construct a well-shaped mesh M that conforms well to $f(x)$.

2.2 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 high quality Delaunay mesh. It can be used to generate meshes for various quality conditions.

Shimada and Gossard (1995) 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 particles that interact with each other under repulsive/attractive forces. Here particle is typically a sphere with radius proportional to the control spacing at its center for isotropic meshing. They first define a proximity-based force among the spheres, and then find a stable configuration by moving, inserting, or deleting spheres. However, their method does not provide any theoretical bound on the time of the algorithm nor the quality of meshes that the algorithm generates.

Miller, Talmor, Teng, and Walkington (1995, 1998) designed a sphere-packing based meshing method which combines two well-known methods: quadtree and Delaunay triangulation. First, the algorithm applies a balanced quadtree refinement to approximate the spacing function $f(x)$. Second, it oversamples a set of points in the domain to define a set of overlapping spheres. Then, it computes a maximal independent set (MIS) of non-overlapping spheres to obtain a sphere packing. Finally, it computes the Delaunay triangulation of the centers of these spheres. Notice that, to generate high quality meshes, it needs very dense sample points initially; on the other hand, dense sample implies high complexity of the algorithm.

Let $B(x,r)$ denote the sphere centered at point x with radius r. Suppose $f(x)$ is the desired element size function. Miller, Talmor, Teng, and Walkington (1995, 1998) introduced the following definition to capture the quality of a sphere packing.

Definition 2.5 [β-Packing] *A set S of spheres is a β-packing with centers P of Ω with respect to a spacing function $f(x)$ if*

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

The following structure theorem by Miller, Talmor, and Teng (1998) states that β-sphere packing implies a well-shaped mesh.

Theorem 2.1 *For any positive constant β, there exists a constant ρ depending only on β such that if $f(x)$ is a spacing function of Lipschitz constant 1 over a domain Ω and S is a β-packing with respect to $f(x)$, then the Delaunay triangulation M of the centers of S is a ρ well-shaped mesh; in addition, for each point p in Ω, $N_M(p) = \Theta(f(p))$, where the constant in Θ depends only on β.*

3 Biting to Generate Mesh

Li, Teng, and Üngör (2000) present a new scheme, called the *biting method*, which combines the strengths of advancing front and sphere packing. It uses advancing front to generate a quality sphere packing rather than the mesh itself. The Delaunay triangulation of the centers of the spheres is then used to define the final mesh. They show that the biting method constructs a well-shaped Delaunay mesh whose size is optimal up to a constant factor. In this paper, we show that biting *squares* instead of *spheres* will also generate well-shaped meshes. Moreover, we show that the time complexity of the new algorithm is $O(n \log n)$, where n is the number of the mesh vertices generated. Furthermore, the square biting method can be easily extended to three dimensions. When post refinement or coarsening is needed, our algorithm can also refine or coarsen the previous generated mesh.

3.1 Biting Scheme

The basic idea of the biting method is to use the advancing front method to construct a well-spaced vertex set with respect to the spacing function. The input domain boundary is set as the initial advancing front. The biting method selects a vertex of the advancing front, and removes a square centered at it from the remaining interior domain. The removed square is called the *biting-square*. That vertex is added as a new mesh vertex, and the boundary between the union of biting-squares and the remaining interior domain defines the new front. The above steps are repeated until the advancing front is empty. The Delaunay triangulation of the biting centers is the final mesh. The following Figures 1, 2, 3, 4, 5 give a snapshot of the biting scheme. The input is a PSLG domain with a hole inside. Notice that the biting-squares centered at interior points are aligned with axis.

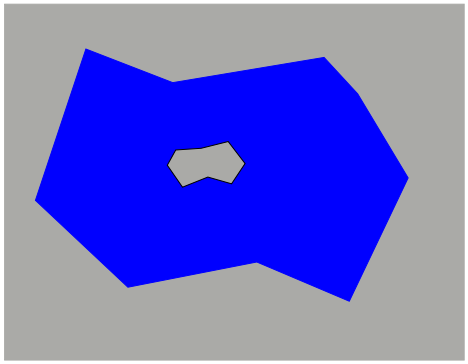


Figure 1 : The initial PSLG domain with a hole inside.

Let $\Pi(x,r)$ be a square centered at point x with edge length $2r$. Note that the orientation of a square is not denoted here. Let $B(x,r)$ be a sphere centered at point x with radius r. A biting-square at a point x is $\Pi(x, c_b f(x))$, where c_b is a constant that will be decided later.

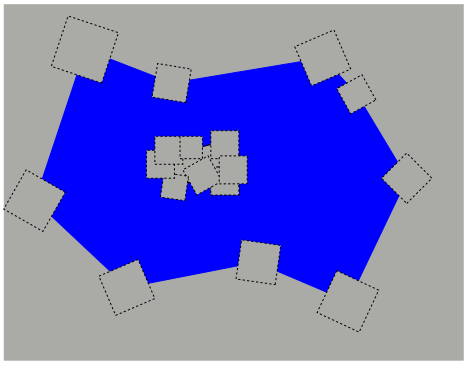


Figure 2 : Biting only on the vertices of the polygon.

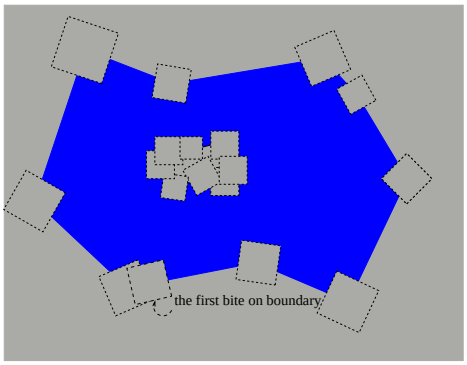


Figure 3 : The first bite of a non-original vertex.

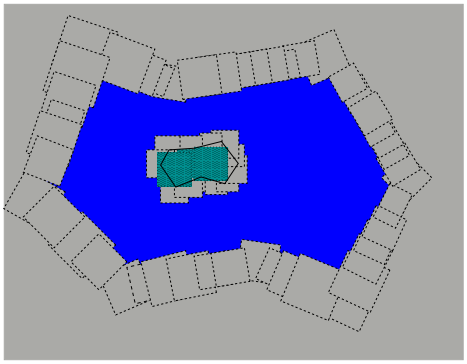


Figure 4 : Biting a layer of the boundary.

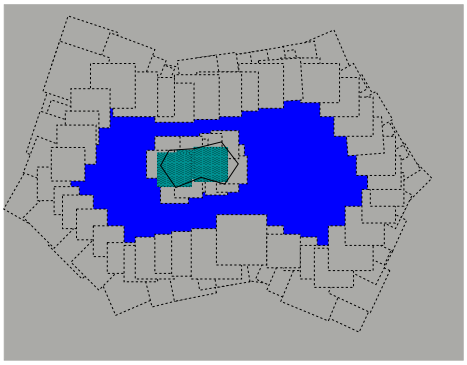


Figure 5 : Biting a layer in the interior of the domain.

Usually, the advancing front is represented as a circular list of already placed mesh vertices. In our method, we always choose the next Steiner point on the front itself, i.e., the front itself is a subset of a feasible region to select new mesh vertices. Consequently, it is easier to choose the next mesh vertex than classic advancing front methods. Observe that the packing-sphere will never appear in our biting method. We only remove the biting-squares from the domain. We will prove that the centers of all biting-squares define a sphere packing by carefully choosing the sizes of the packing-spheres. The following is a formal description of the biting-square method.

Algorithm: BITING SQUARE

1. **[Initial Front:]** Let the boundary of the domain be the initial front; see Figure 1;
2. **[Vertex Protection:]** Bite all the input vertices by removing their biting squares from the interior of the domain; see Figure 2. The orientation of the square is decided by the two incident edges of the vertex. See Figure 7. Update the advancing front after removing the biting-square.
3. **[Edge Protection:]** Bite squares centered on the input boundary: choose a vertex x on the front and remove its biting-square; see Figures 3 and 4. The biting-square is aligned with the boundary edge. See Figure 6. Update the advancing front.

   Repeat until all initial input boundaries are bitten.
4. **[Interior Biting:]** Choose a vertex x on the front and remove its biting-square; see Figures 5. The biting-square is aligned with the axes. Update the advancing front.

   Repeat until the advancing front is empty.
5. **[Delaunay Triangulation:]** Construct the Delaunay triangulation of the biting centers as the final mesh.

Notice that, after every biting, the intersection points of current biting-square with previous biting-squares are the candidates of biting centers later. Hence for protecting the boundary, we align the biting-square centered at a boundary vertex along the boundary. Thus the introduced new candidate biting points are not too close to the boundary compared with its control spacing requirement. See Figure 6 for illustrations of boundary biting alignments.

However, it is impossible to align all bitings with the boundary edges. For example, for an input vertex v that is incident by two non-perpendicular boundary edges vu_1 and vu_2, it is impossible to align the biting-square centered at v with vu_1 and vu_2. Let line vx be the line that divides the angle $\angle u_1vu_2$ into equal half. We use the following criteria to select the orientation of the biting-square centered at v. If the angle $\angle u_1vu_2 \leq 135^o$, or the angle $\angle u_1vu_2 \geq 225^o$, the diagonal of the biting-square is then aligned with vx. Otherwise, $135^o \leq \angle u_1vu_2 \leq 225^o$, then one side of the biting-square is aligned with vx. See Figure 7 for the orientation illustrations. By a simple geometry computation, the above orientation maximizes the angle formed by the boundary edges and intersected side edge of the biting-square, i.e., the angle $\angle vpq$ in the Figures.

4 Quality Guarantee of Biting

In this section we show that the biting-square method generates well-shaped meshes. Moreover, the mesh size is within a constant factor of the optimal. For the first statement we prove that the points placed by the biting method are well-spaced, i.e., they are centers of a β-packing with respect to a 1-Lipschitz spacing function. The size optimality then follows from the fact that the spacing function is well-conformed.

4.1 β-*Sphere Packing*

For each mesh vertex generated by the biting scheme, we define a *packing-sphere* centered at it. Observe that the biting-squares generated by the biting scheme overlap among themselves. Therefore, the packing-sphere of a mesh vertex is chosen to be smaller than its biting-square. Let us focus on a particular point x. From the specification of the biting scheme, the biting-square centered at x is $\Pi(x, c_b f(x))$, where $c_b < 1$ is a positive constant. We now choose another positive constant $c_p < c_b$, and define the packing-sphere at x to be $B(x, c_p f(x))$. See Figure 8 for an illustration of biting-square and the packing-sphere defined at a mesh vertex. The following lemma proved by Li, Teng, and Üngör (2000) implies the relation that need to be satisfied by the biting constant c_b and the packing constant c_p.

Lemma 4.1 *[Li, Teng, and Üngör (2000)] Assume that spacing function $f(x)$ is α-Lipschitz, and $||x-y|| \geq \frac{2\gamma}{1-\alpha\gamma}\min(f(x), f(y))$, where $\alpha\gamma < 1$. Then the interior of two spheres $B(x, \gamma f(x))$ and $B(y, \gamma f(y))$ do not overlap.*

Thus the biting-square $\Pi(x, c_b f(x))$ is like a protecting square of x: it prevents any point whose packing-sphere potentially overlaps with that of point x from being chosen. Notice that the biting scheme works for any control spacing function $f(x)$ with Lipschitz condition. Our first goal is to show that the biting scheme generates a good sphere packing. Let S_b be the set of biting-squares generated by the biting scheme, and S_p be the set of corresponding packing-spheres defined as above. Lemma 4.1 implies that if the biting constant satisfies that $c_b \geq \frac{2c_p}{1-\alpha c_p}$ and $\alpha c_p < 1$, then the interior of any two packing-spheres do not overlap. Which is stated by the following lemma.

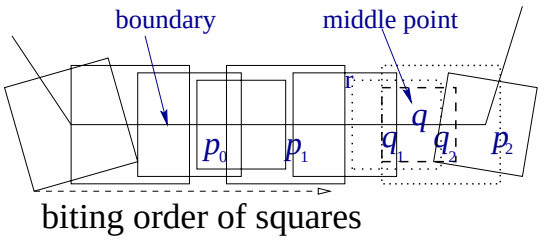


Figure 6 : The biting centered at vertices of input boundary.

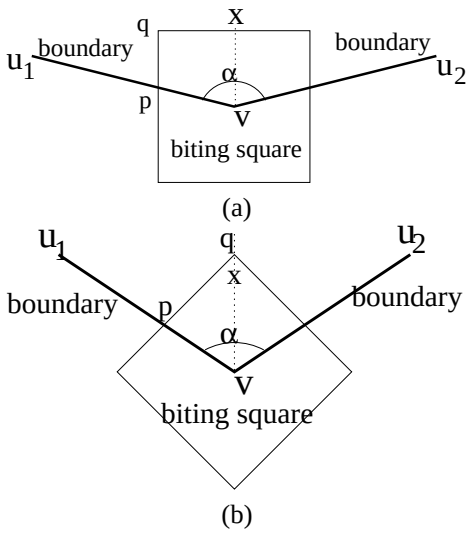


Figure 7 : The orientation of the biting-square: (a) if $135^o \leq \alpha \leq 225^o$, one side of the biting-square is aligned with *vx*. (b) if $\alpha < 135^o$ or $\alpha \geq 225^o$, then the diagonal edge of the biting-square is aligned with *vx*.

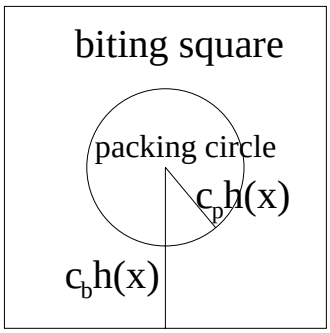


Figure 8 : The biting-squares and packing-spheres centered at point *x*.

Lemma 4.2 [NO OVERLAP] *If the packing constant satisfies* $c_p = \frac{c_b}{2+\alpha c_b}$*, then there is no overlap among the packing-spheres* S_p*.*

PROOF. Let's consider any two packing-spheres $B(x, c_p f(x))$ and $B(y, c_p f(y))$ defined by two biting vertices x and y. Observe that either x is bitten before y, or y is bitten before x, which is implied by the biting scheme. In other words, either x is outside of the biting-square centered at y or y is outside of the biting-square centered at x. This simple observation implies that $||x-y|| \geq c_b \min(f(x), f(y))$. From $c_p = \frac{c_b}{2+\alpha c_b}$, we have $c_b = \frac{2c_p}{1-\alpha c_p}$ and $\alpha c_p = \frac{\alpha c_b}{2+\alpha c_b} < 1$. The lemma then follows from Lemma 4.1. □

In other words, the packing-spheres defined at the centers of all bitten squares satisfy the first two conditions of sphere packing. It remains to show that for any point y in the domain, there is a sphere from S_p that intersects the sphere which is β factor of the packing-sphere centered at y, where β is a constant.

Lemma 4.3 [NO LARGE GAP] *For any point* $y \in \Omega$*, there is a sphere in* S_p *that overlaps with* $B(y, \beta c_p f(y))$*, where* $\beta = \frac{2\sqrt{2}-1+\sqrt{2}\alpha c_b}{1-\sqrt{2}\alpha c_b}$*. In other words, there is no large gap at* y*.*

PROOF. The biting scheme guarantees that point y in the domain Ω is covered by at least one biting-square of S_b. Let $\Pi(x, c_b f(x))$ be the biting-square that covers y. Then $||y-x|| \leq \sqrt{2} c_b f(x)$. Because spacing function $f(x)$ is α-Lipschitz, $f(y) \geq (1-\sqrt{2}\alpha c_b) f(x)$. Noting that $c_p = 2c_b/(2+\alpha c_b)$, we have

$$\begin{aligned}
& \beta c_p f(y) + c_p f(x) \\
= & \frac{2\sqrt{2}-1+\sqrt{2}\alpha c_b}{1-\sqrt{2}\alpha c_b} c_p f(y) + c_p f(x) \\
= & \frac{c_b}{2+\alpha c_b}\left(\frac{2\sqrt{2}-1+\sqrt{2}\alpha c_b}{1-\sqrt{2}\alpha c_b} f(y) + f(x)\right) \\
\geq & \frac{c_b}{2+\alpha c_b}\left(\frac{2\sqrt{2}-1+\sqrt{2}\alpha c_b}{1-\sqrt{2}\alpha c_b}(1-\sqrt{2}\alpha c_b) f(x) + f(x)\right) \\
= & \frac{c_b}{2+\alpha c_b}\left((2\sqrt{2}-1+\sqrt{2}\alpha c_b) f(x) + f(x)\right) \\
= & \sqrt{2} c_b f(x) \\
\geq & ||x-y||.
\end{aligned}$$

Hence for point y, there is a sphere in S_p that overlaps with the sphere $B(y, \beta c_p f(y))$. In other words, there is no large gap at y. □

Consequently, we know that the biting method implicitly generates a β-sphere packing. From the β-packing definition 2.5, S_p is β-packing with respect to spacing function $2c_p f(x)$. We summarize it by the following theorem.

Theorem 4.4 [β-SPHERE PACKING] *The set of spheres* S_p *is a* $\beta = \frac{2\sqrt{2}-1+\sqrt{2}\alpha c_b}{1-\sqrt{2}\alpha c_b}$*-packing with respect to the spacing function* $\frac{2c_b}{2+\alpha c_b} f(x)$*.*

Then it follows from Theorems 2.1 and 4.4 that the Delaunay triangulation of the centers of S_p is a well-shaped mesh, i.e., the radius-edge ratio of each of its elements is bounded from above by a constant.

Theorem 4.5 [WELL-SHAPED MESH] *The square biting method generates meshes whose radius-edge ratio is bounded from above by a constant.*

4.2 The Spacing Conformity and the Size Guarantee

In this section, we show that the nearest neighbor value of any point x in the domain is related to the control spacing function $f(x)$ by a constant factor. This relation enables us to show that the biting scheme generates a mesh whose size is within a constant factor of any competing mesh.

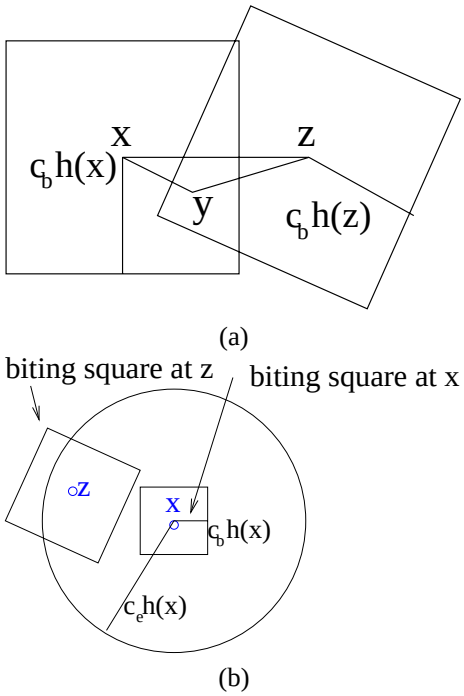


Figure 9 : The biting-squares and packing-spheres. (a) a point y is covered by at least two biting-squares, (b) for any biting-square $\Pi(x, c_b f(x))$, there must exist a biting-square centered at z, such that z is contained inside the sphere $B(x, c_e f(x))$, where $c_e = 2\sqrt{2}c_b/(1-\sqrt{2}\alpha c_b)$.

Recall that the required spacing function $f(x)$ is α-Lipschitz. Let $c_e = 2\sqrt{2}c_b/(1-\sqrt{2}\alpha c_b)$. We define sphere $B(x, c_e f(x))$ as the extension sphere of the biting-square. From Lemma 4.1, we know that the two biting-squares $\Pi(x, c_b f(x))$ and $\Pi(y, c_b f(y))$ do not intersect if $||x-y|| \geq c_e f(x)$ and $\sqrt{2}\alpha c_b < 1$. In other words, there is at least one other mesh vertex generated inside the sphere $B(x, c_e f(x))$ other than x. It implies that the nearest neighbor value $N(x)$ can not be too large. We have the following lemma to bound the nearest neighbor value $N(x)$.

Lemma 4.6 *For each vertex x of the mesh generated by the biting-square method, $N(x)$ satisfies that*

$$c_b(1-\sqrt{2}\alpha c_b) f(x) \leq N(x) \leq \frac{2\sqrt{2}c_b}{1-\sqrt{2}\alpha c_b} f(x). \quad (1)$$

PROOF. We first show that the nearest neighbor value $N(x)$ is not too small. Let y be its nearest mesh vertex. First, if vertex y is not contained inside the interior of the biting-square $\Pi(x, c_b f(x))$, then $N(x) \geq c_b f(x)$. If y is inside square $\Pi(x, c_b f(x))$, then vertex y must be bitten before x. We have

$$c_b f(y) \leq ||y-x|| \leq \sqrt{2}c_b f(x).$$

Because $f(x)$ is α-Lipschitz, $f(y) \geq (1-\sqrt{2}\alpha c_b) f(x)$. Hence,

$$N(x) = ||y-x|| \geq c_b(1-\sqrt{2}\alpha c_b) f(x).$$

We then show that the nearest mesh vertex y can not be too far away. By Lemma 4.1, if y is not contained inside the sphere $B(x, c_e f(x))$, the biting-square centered at y does not overlap with that of x. Hence, the boundary of $\Pi(x, c_b f(x))$ is not covered by any other biting-squares. See Figure 9 (c). The lemma then follows from property that every point in the domain is covered by at least one biting-square. In other words, $N(x) \leq c_e f(x)$. □

The above lemma implies that the generated mesh conforms well to the given control spacing $f(x)$.

Theorem 4.7 *The mesh generated by the* biting *method has* $\min(c_b(1-\sqrt{2}\alpha c_b), \frac{1-\sqrt{2}\alpha c_b}{2\sqrt{2}c_b})$ *conformity to the control spacing.*

We now show that the nearest neighbor value of each non-mesh point in the domain is also linearly related to its control spacing value.

Lemma 4.8 *Assume point $y \in \Omega$ is not a mesh vertex. Then $N(y)$ defined by the mesh generated by the biting method satisfies*

$$\frac{c_b f(y)}{2+2\sqrt{2}\alpha c_b} \leq N(y) \leq \frac{(3\sqrt{2}-2\alpha c_b)c_b f(y)}{(1-\sqrt{2}\alpha c_b)^2}. \quad (2)$$

PROOF. First, there is at least one biting-square, say $\Pi(x, c_b f(x))$, that covers point y. As proved by the previous lemma, we know that there is at least one mesh vertex z other than x that is inside sphere $B(x, c_e f(x))$. See Figure 9 (c). Then $||z-y|| \leq ||z-x|| + ||x-y|| \leq c_e f(x) + \sqrt{2}c_b f(x) = (c_e + \sqrt{2}c_b) f(x)$. So the nearest neighbor value at y satisfies

$$N(y) \leq max(||x-y||, ||z-y||) \leq (c_e + \sqrt{2}c_b) f(x).$$

Then the inequality $f(y) \geq f(x) - \alpha||x-y|| \geq (1-\sqrt{2}\alpha c_b) f(x)$ implies that

$$N(y) \leq (\sqrt{2}c_b + c_e) f(y)/(1-\sqrt{2}\alpha c_b).$$

We then show that the nearest neighbor value $N(y)$ is also not too large. Recall that point y is covered by biting-squares. There are two cases: the first case is that only one biting-square contains y, and the second case is that two or more biting-squares contain y.

In the first case, let us assume that y is covered by biting-square $\Pi(x, c_b f(x))$. Thus $||y-z|| > c_b f(z)$ holds for any other mesh vertex z, i.e., $y \notin \Pi(z, c_b f(z))$. Because $f(x)$ is α-Lipschitz,

$$f(y) \leq f(z) + \alpha||y-z|| \leq (\alpha + 1/c_b)||y-z||.$$

Then for any mesh vertex z, if $y \notin \Pi(z, c_b f(z))$, then $||y-z|| \geq \frac{c_b}{1+\alpha c_b} f(y)$. Therefore, by the definition of the nearest neighbor function, $N(y) \geq min_{z \neq x}(||y-z||)$. This implies that

$$N(y) \geq \frac{c_b}{1+\alpha c_b} f(y).$$

In the second case, assume that biting-squares $\Pi(x, c_b f(x))$ and $\Pi(z, c_b f(z))$ contain y. See Figure 9 (b). Further assume that $\Pi(x, c_b f(x))$ is bitten before $\Pi(z, c_b f(z))$. Hence $||x-z|| \geq c_b f(x)$. Because $f(x)$ is α-Lipschitz and $y \in \Pi(x, c_b f(x))$, we have $f(y) \leq (1+\sqrt{2}\alpha c_b) f(x)$. By the triangle inequality,

$$\begin{aligned} & max(||y-x||, ||y-z||) \\ \geq & \quad \tfrac{1}{2}||x-z|| \\ \geq & \quad c_b/2 f(x) \\ \geq & \quad c_b/(2+2\sqrt{2}\alpha c_b) f(y). \end{aligned}$$

Therefore, the second smallest distance from y to the set of mesh vertices whose biting-squares contain y is at least $c_b/(2+2\alpha c_b) f(y)$. In addition, from the analysis of the first case, the smallest distance from y to the set of mesh vertices whose biting-squares do not contain y is at least $c_b/(1+\alpha c_b) f(y)$. Thus,

$$N(y) \geq c_b/(2+2\sqrt{2}\alpha c_b) f(y).$$

Consequently,

$$\frac{c_b}{2+2\sqrt{2}\alpha c_b} f(y) \leq N(y) \leq \frac{\sqrt{2}c_b + c_e}{1-\sqrt{2}\alpha c_b} f(y),$$

and the lemma follows from $c_e = 2\sqrt{2}c_b/(1-\sqrt{2}\alpha c_b)$. □

Some tedious manipulation yields that the number of the mesh vertices of a well-shaped triangular mesh is linearly related to the integral of $1/N(y)^2$ over all $y \in \Omega$. See the results by Miller, Talmor, Teng, and Walkington (1998). The $N()$ function deduced from the mesh generated by the biting method is within a constant factor of $f(x)$ implies the following theorem. Notice that the constant depends on the quality of the competing mesh.

Theorem 4.9 *Size of the mesh generated by the biting method is within a constant factor of any well-shaped mesh that conforms well to the given control spacing function $f(x)$.*

4.3 The Radius-edge Ratio of the Mesh

Theorem 4.5 shows that the biting scheme generates well-shaped meshes. However, the constant bound on the radius-edge ratio so derived may be too small. In this section, we provide a better analysis of the bound of the radius-edge ratio of all mesh elements.

Our analysis is mainly concentrated on two-dimensional domain. In our analysis, as by Li, Teng, and Üngör (2000), we divide the triangle elements into two subsets: the first subset contains all elements whose circumcenters are inside the domain and the second subset contains all other elements. Our aim is to derive a direct analysis on the radius-edge ratio of all mesh elements generated. We first study the triangles in the first subset.

Lemma 4.10 *Let $\triangle pqr$ be a triangle of the first subset. Let l be the length of the shortest edge of $\triangle pqr$; R be the radius of the circumcircle C of $\triangle pqr$. Then the radius-edge ratio of $\triangle pqr$ satisfies*

$$\frac{R}{l} \leq \frac{\sqrt{2}}{1-2\sqrt{2}\alpha c_b}. \tag{3}$$

PROOF. Let c be the circumcenter of $\triangle pqr$. Assume that c is covered by biting-square $\Pi(x, c_b f(x))$. Then $||c-x|| \leq \sqrt{2}c_b f(x)$. Because the mesh is a Delaunay triangulation, x is not in the interior of circumcircle of $\triangle pqr$, i.e., $||x-c|| \geq R$. Thus $f(x) \geq R/(\sqrt{2}c_b)$. Because $f(x)$ is α-Lipschitz, $f(c) \geq f(x) - \alpha||c-x||$. Also because $||c-x|| \leq \sqrt{2}c_b f(x)$, we have

$$f(c) \geq (1-\sqrt{2}\alpha c_b) f(x) \geq (1-\sqrt{2}\alpha c_b) R/(\sqrt{2}c_b).$$

Without loss of generality, assume edge pq is the shortest edge of the triangle, i.e., $||p-q|| = l$. Also assume that $\Pi(q, c_b f(p))$ is bitten before $\Pi(q, c_b f(q))$, which implies that $l \geq c_b f(p)$. Because $f(x)$ is α-Lipschitz, $f(c) \leq f(p) + \alpha R \leq l/c_b + \alpha R$. Using $f(c) \geq (1-\sqrt{2}\alpha c_b) R/(\sqrt{2}c_b)$, we have

$$(1-\sqrt{2}\alpha c_b) R/(\sqrt{2}c_b) \leq f(c) \leq (l + \alpha c_b R)/c_b.$$

The lemma then follows. □

We now study the triangle elements in the second subset. Consider a triangle $\triangle_{x_1 x_2 x_3}$ from the second subset. Let C be its

circumcircle. Let c be the center of C; R be the radius of C. Assume vertices p_1 and p_2 are the two closest mesh vertices on a boundary that separates $\triangle x_1x_2x_3$ and c. Here, we also assume that p_1p_2 is the closest boundary segment to $\triangle x_1x_2x_3$. Note that p_1 and/or p_2 may be one of the vertices of triangle $\triangle x_1x_2x_3$. See Figure 10 for an example.

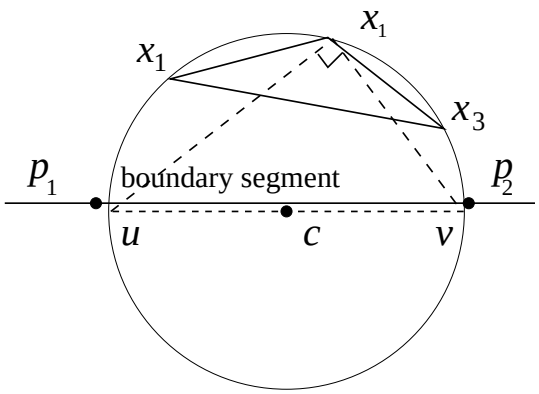


Figure 10 : A triangle $x_1x_2x_3$ of the second subset. The circumcenter c is separated from $x_1x_2x_3$ by segment p_1p_2.

From the Delaunay triangulation property, we know that there is no mesh vertex inside the circumcircle C. It implies that biting-squares centered at p_1 and p_2 should cover the segment p_1p_2. Let r be the intersection point of $\Pi(p_1, c_b f(p_1))$ with $\Pi(p_2, c_b f(p_2))$ that is inside the domain. It is simple to show that angle $\angle p_1x_ip_2$ is obtuse for $i = 1,2,3$. Observe that $\angle p_1x_ip_2 \leq \angle p_1rp_2$. Therefore, angle $\angle p_1rp_2$ is also obtuse. Without loss of generality, assume that p_1 is bitten before p_2.

From the biting scheme, we know that only the following biting scenarios are possible. Vertex p_2 is on the intersection of biting-square $\Pi(p_1, c_b f(p_1))$ with the boundary segment containing p_1p_2, as illustrated by Figure 11 (a) and (b). Or vertex p_2 is on the intersection of biting-square $\Pi(p_3, c_b f(p_3))$ with the boundary segment containing p_1p_2 and biting-squares centered at p_1 and p_2 overlaps, as illustrated by Figure 11 (c). Or vertex p_1 is an input corner vertex, and p_2 is the intersection point of biting-square $\Pi(p_1, c_b f(p_1))$ with the boundary segment containing p_1p_2, which is illustrated by Figure 11 (d), (e) and (f).

Recall that the angle $\angle p_1rp_2$ is obtuse if segment p_1p_2 separates a triangle $\triangle x_1x_2x_3$ and its circumcenter. By checking all possible configurations of p_1 and p_2, only case (c) is possible to produce an obtuse angle p_1rp_2. See Figure 11. It then remains to study the situation illustrated by the Figure 11 (c).

Lemma 4.11 *The radius-edge ratio of any generated Delaunay triangle whose circumcenter is not inside the domain is at least* $R/l \leq \frac{t^2-2t+2}{6t^2-8t+2}$, *where* $t = \alpha c_b$.

PROOF. We assume that the spacing at vertex p_1 is less than that of p_2. We first show that length of p_1p_2 is within a small

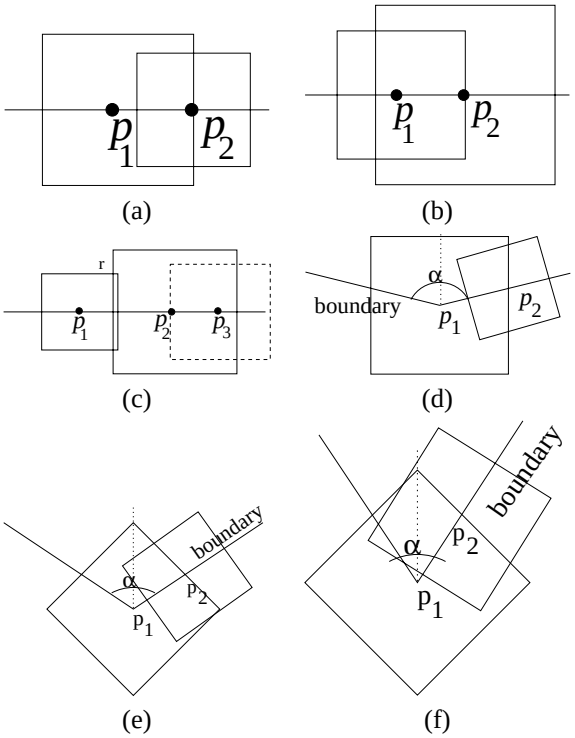


Figure 11 : The configurations of adjacent biting centered at vertices of input boundary. Cases (a): the spacing at p_2 is less than that of p_1; (b): the spacing at p_2 is larger than that of p_1; (c): p_2 is the intersection of $\Pi(p_3, c_b f(p_3))$ with the boundary segment; (d): p_1 is input vertex, and the input angle at p_1 is larger than 135^o. (e) and (f): p_1 is input vertex, and the input angle at p_1 is less than 135^o.

constant factor of $f(p_1)$. Because two biting-squares overlap, then Lemma 4.1 implies that $||p_1 - p_2|| \leq \frac{2c_b}{1-\alpha c_b} f(p_1)$. The

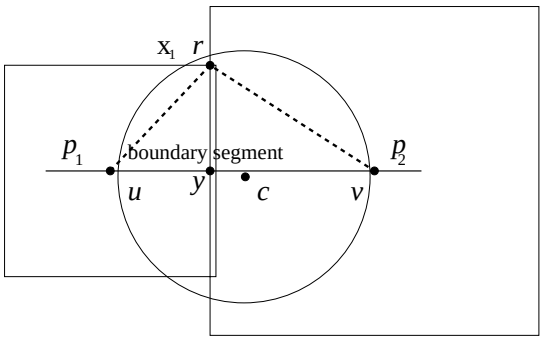


Figure 12 : The example of two biting vertices p_1 and p_2 on a boundary segment that separates a triangle and its circumcenter.

above Figure 12 illustrates the proof that follows. Let r be the intersection point of biting-squares centered at p_1 and p_2, which is in the interior of the domain. Observe that r must be inside the circumcircle of triangle $\bigtriangledown x_1x_2x_3$, as shown in the Figure 12. Otherwise, the vertices of triangle $\bigtriangledown x_1x_2x_3$ can not be generated by biting. Let h_r be the distance of point r to boundary segment p_1p_2. We have $h_r = c_b f(p_1)$. Recall that the angle $\angle p_1 r p_2$ is obtuse. Then it is simple to show that $||p_1 - p_2|| \geq 2h_r$. For convenience, let $E = ||p_1 - p_2||/2$. Then $h_r \geq (1 - \alpha c_b)E$ and $h_r < E$.

We then show that the circumradius R of triangle $\triangle_{x_1x_2x_3}$ is not too large. Observe that the circumcircle of $\triangle_{x_1x_2x_3}$ does not contain vertices p_1 and p_2 inside. Some tedious computation yields that $R \leq (h_r^2 + E^2)/(2h_r)$. For the sake of easy presentation, let $t = \alpha c_b$. Because $h_r \geq (1 - \alpha c_b)E$ and $h_r < E$, we then have

$$R \leq \frac{t^2 - 2t + 2}{2 - 2t} E.$$

It remains to show that the shortest edge of triangle $\bigtriangledown x_1x_2x_3$ is not too short. Assume that the shortest edge is x_1x_3 and vertex x_1 is bitten before x_3. Then the length l of this shortest edge satisfies $l \geq c_b f(x_1) \geq c_b(f(p_1) - \alpha||x_1 - p_1||)$. From $\angle p_1x_1p_2$ is obtuse, we have $||p_2 - p_1|| > ||x_1 - p_1||$. It follows that

$$l \geq c_b f(p_1) - t||p_1 - p_2|| \geq (1 - 3t)E.$$

Then we are in the position to claim that the radius-edge ratio of triangle $\triangle_{x_1x_2x_3}$ is bounded from above by a constant, i.e.,

$$R/l \leq \frac{t^2 - 2t + 2}{6t^2 - 8t + 2}.$$

The similar proof follows when the spacing at vertex p_1 is not less than that of p_2. □

Consequently, we know that the meshes generated by biting scheme have small radius-edge ratio. Notice that, the bound of the radius-edge ratio for triangles whose circumcenters are not inside the domain can be improved by the following observation. After biting the squares centered at p_1 and p_2, the region left in the circumcircle of $\bigtriangledown x_1x_2x_3$ is too small to fill three bitings centered at x_1, x_2 ad x_3. It implies that p_1 and p_2 are the vertices of the triangle $\bigtriangledown x_1x_2x_3$.

4.4 Numerical Robustness

To construct the Delaunay triangulation of the biting centers, we often need to check if all edges are locally Delaunay. Let's consider an edge pq of the mesh, and let r, s be two mesh vertices that form two triangles pqr and pqs with edge pq. Edge pq is locally Delaunay, if the circumcircle of triangle pqr does not contain vertex s inside. The operation to check if an edge pq is locally Delaunay is often called *in-circle* test. Due to the roundoff error, the in-circle test is not always consistent, which in turn will cause the Delaunay kernel to result in a non-valid triangulation. Several solutions have been investigated to overcome these problems; see George and Borouchaki (1998). They either slightly perturb all points, or introduce a threshold value in comparisons, or perform exact computations by using the integer-type coordinate system. To address this problem, we propose a new method that generates high quality mesh by combining it with our biting method.

Let φ be a positive constant less than 1. Assume pq is the current edge to be checked, and $\bigtriangledown pqr$, $\bigtriangledown pqs$ are the two incident triangles. Let $B(c_r, R_r)$, and $B(c_s, R_s)$ be the circumcircles of $\bigtriangledown pqr$, $\bigtriangledown pqs$ respectively. We call them the φ-*circumcircles* of the triangles. For convenience, let $\alpha_\tau(x) = ||x - c||/R$, where c, R are the center and radius of the circumcircle of an element τ. See George and Borouchaki (1998). Then the following definition is introduced by Li, Teng, and Üngör (1999).

Definition 4.1 [φ-LOCALLY-DELAUNAY] *Edge pq is called* φ*-locally-Delaunay if r is not in the interior of circle $B(c_s, \varphi R_s)$, and s is not in the interior of circle $B(c_r, \varphi R_r)$, i.e., $\alpha_{\bigtriangledown pqs}(r) \geq \varphi$, and, $\alpha_{\bigtriangledown pqr}(s) \geq \varphi$.*

The above Figure 13 gives an example of edge pq that is φ-locally Delaunay. Then we define the φ-Delaunay mesh as following.

Definition 4.2 [φ-DELAUNAY] *A mesh is called* φ*-Delaunay, if all edges are* φ*-locally-Delaunay. In other words, the* φ*-circumcircle of each triangle is empty.*

Hence the traditional Delaunay triangulation is 1-Delaunay under this definition. To make a mesh φ-Delaunay, we check each edge of the mesh, if it does not satisfy the φ-local-Delaunay property, we flip the edge. Notice that, there may have many φ-Delaunay triangulations for a given point set.

Recall that the Delaunay triangulation of any two-dimensional point set maximizes the minimum angle among all possible triangulations. However, the φ-Delaunay triangulation of an arbitrary two-dimensional point set can not guarantee that the minimum angle is within a constant factor of the minimum angle generated by the Delaunay triangulation. The next theorem shows that any φ-Delaunay triangulation of any point set generated by the biting method is a well-shaped mesh.

Theorem 4.12 [WELL-SHAPED φ-DELAUNAY] *Let M be a φ-Delaunay triangulation of the point set generated by biting domain Ω according to an α-Lipschitz spacing $f(x)$ using biting constant c_b. Then the radius-edge ratio of the mesh is at most a constant that depends on α, c_b, φ.*

PROOF. We show it for the triangles whose circumcenters are inside the domain. We will prove that

$$R/l \leq \frac{\sqrt{2}}{(1-\sqrt{2}\alpha c_b)\varphi - \sqrt{2}\alpha c_b}.$$

Let c be the circumcenter of $\triangle pqr$; let R be the circumradius of $\triangle pqr$. Assume that the circumcenter c is covered by biting-square $\Pi(x, c_b f(x))$. Noticed that $||x-c|| \geq \varphi R$ because of the φ-Delaunay triangulation property. Similar to Lemma 4.10, we have

$$f(c) \geq (1-\sqrt{2}\alpha c_b)\varphi R/(\sqrt{2}c_b),$$

and

$$f(c) \leq l/c_b + \alpha R.$$

Then the theorem follows, if $\varphi \geq \frac{\sqrt{2}\alpha c_b}{1-\sqrt{2}\alpha c_b}$ and $\alpha c_b < \sqrt{2}/4$.

When the circumcenter is not inside the domain, the analysis is similar to the proof that the Delaunay triangulation of the generated vertices has small radius-edge ratio. The details of the proofs are omitted here. □

5 The Complexity of Biting

In this section, we show that the time complexity of biting square scheme is $O(n \log n)$, where n is the number of the output vertices.

We assume that the advancing front is represented by linked list of edges. The list of edges are in counterclockwise direction and there is no self-intersection among edges. Each advancing front edge has pointers to the two vertices and the two edges that are incident to it. In addition, each vertex on advancing front has pointers to the two edges that are incident to it. For the sake of easy presentation, we will use (st, ed, pe, se) to denote the four pointers of an edge, where *st* and *ed* are the *source* and *destination* vertex respectively; *pe* and *se* are the *previous* and *successive* edge respectively. In addition, we use (pe, se) to denote the two pointers of a vertex, where *pe* and *se* are the previous and successive incident edges of this vertex. For example, the advancing front of the domain illustrated in Figure 14 (a) is represented by the linked edges v_1v, vv_5, v_5v_4, v_4v_3, v_3v_2 and v_2v_1. Edge vv_5 is represented by (v, v_5, v_1v, v_5v_4). Vertex v has pointers to two edges v_1v and vv_5.

We then analyze the time complexity of the algorithm by studying the complexity of every stage of the biting. Observe that, there are no intersections among biting-squares centered on input vertices, because the constant $c_b < 1/2$ and the spacing function $f(x)$ is no more than the local feature size function $lfs(x)$. Therefore, to remove the biting-square centered at an input vertex v, we only need update the two incident edges of the vertex v for constructing the new advancing front. For example, after removing the biting-square $\Pi(v, c_b f(v))$ from the domain illustrated by Figure 14 (a), the new advancing front is updated as follows. Two new edges pq and qr are created. The edge v_1v is modified to (v_1, p, v_2v_1, pq), i.e., the destination vertex is set to p and the successive edge is set to pq. Edge vv_5 is modified similarly. Observe that we do not need to modify other edges to construct the new advancing front. It implies that we can bite the square centered at each input vertex at constant time. The similar arguments hold for the biting of vertices on the input boundaries. See the following Figure 14 for an illustration of removing a biting square and update the advancing front. In both cases, the new advancing front can be updated in constant time.

We then study how to bite the squares centered at interior points efficiently. The following observation is important to speed up the algorithm. The orientation of the biting-squares does not affect the theoretical bounds of the qualities of the generated mesh. Recall that we always align the biting-squares centered at interior points with the axis in the biting scheme. Notice that the major cost of biting an interior point x lies in computing the intersections of the biting-square $\Pi(x, c_b f(x))$ with the previous advancing front, as well as in constructing the new advancing front. We prove that these can be done in $O(\log n)$ time, where n is the number of vertices of advancing front before biting point x.

Then we first show how to find the edges of previous advancing front that intersect the current biting-square $\Pi(x, c_b f(x))$. Notice that we can find all intersected edges if we find all vertices of current advancing front that are inside the biting-square $\Pi(x, c_b f(x))$. Recall that square $\Pi(x, c_b f(x))$ is aligned with axis. Hence, the question reduces to report all vertices of the advancing front that are inside a coordinates-aligned square.

Notice that there is no any order requirement for interior bitings to guarantee the theoretical bound on the radius-edge ratio of the mesh. Therefore, we can bite special vertex in current advancing front if it can speed up the algorithm. For example, if we select the vertex with the largest y value, then we can

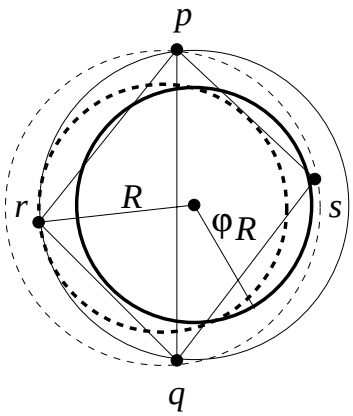


Figure 13 : The shared edge pq of two triangles pqr and pqs are φ-locally Delaunay. The dashed circles are the circumcircle and φ-circumcircle of triangle $\bigtriangledown pqs$. And the solid circles are the circumcircle and φ-circumcircle of triangle $\bigtriangledown pqr$. Notice that the edge pq is not locally Delaunay under classic Delaunay definition.

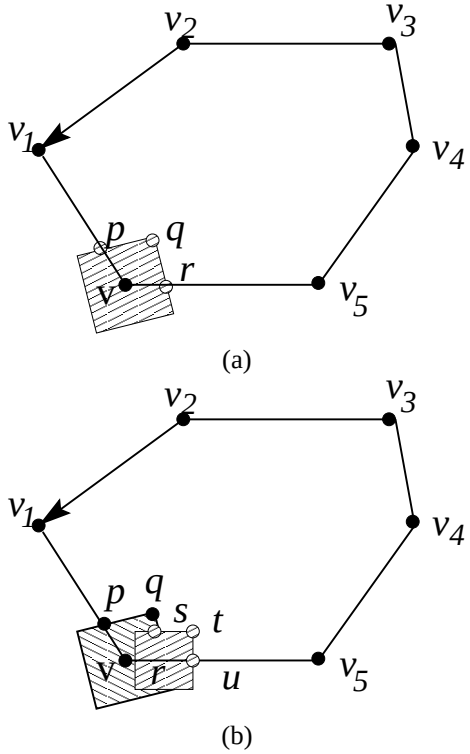


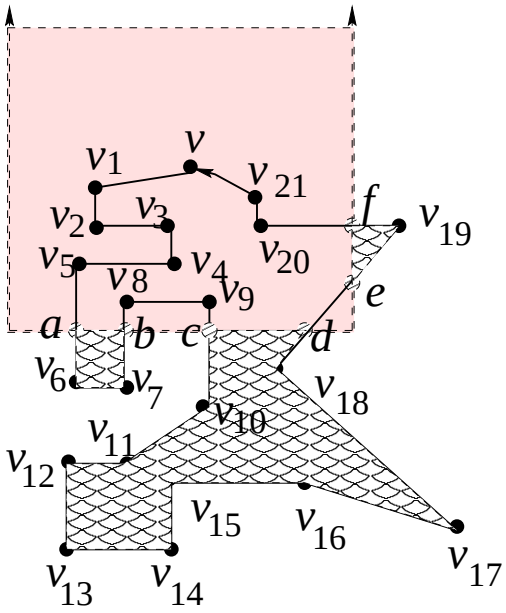
Figure 14 : Remove biting-squares centered at a vertex: case (a): the vertex is an input vertex; case (b): the vertex is on a boundary segment. We use the solid points (such as v_1, v_2) to denote the vertices of previous advancing front and use the shaded points (such as p, q of the left figure) to denote the new vertices in the updated advancing front. For the left figure, the new advancing front after $\Pi(v, c_b f(v))$ is removed is polygon $pqrv_5v_4v_3v_2v_1$. Here, the points p, q, r are the vertices generated due to removing of the biting-square $\Pi(v, c_b f(v))$. For the right figure, the new advancing front after $\Pi(r, c_b f(r))$ is removed is $pqstuv_5v_4v_3v_2v_1$.

use the priority search tree to report all points in a three-sided rectangle (the top side of the rectangle is open). As showed in McCreight (1985), the priority search tree can be built in $O(n \log n)$ time, the report time is $O(\log n + k)$, and the space requirement is $O(n)$. Here k is the number of points to be reported. More importantly, the priority search tree can be updated in $O(\log n)$ time per deletion of a vertex and per insertion of a vertex. Let's first assume that there is only a constant number of vertices of the advancing front that are inside the biting-square to be reported. The proof is given later in this section. Then we can compute all edges in the current advancing front that intersect with the edges of $\Pi(x, c_b f(x))$. Hence only a constant number of queries from the priority search tree is necessary for constructing the new advancing front after biting x. Recall that each query costs $O(\log n)$ time. However, we have to update the priority search tree also. Notice that there is only a constant number of new intersection points introduced. And there is only a constant number of vertices that are covered by the biting square need to be deleted from the current advancing front. Therefore, there is only a constant number of deletions and insertions that are needed to update the priority search tree. This can be done in $O(\log n)$ time also.

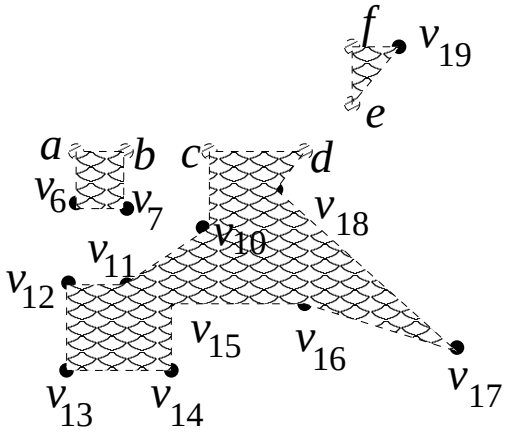
Observe that for updating the new advancing front, just finding the intersection points and all segments that form the advancing front is not sufficient. We have to connect the segments, for example, by linked lists to form the polygonal boundaries of the remaining domain. We show that the new advancing front can be re-linked in constant time after all segments had been computed.

Assume that we want to bite vertex v, and had computed all edges in current advancing fronts which intersect with the edges of $\Pi(v, c_b f(v))$. For the sake of easy presentation, we first assume that the current advancing front is represented by one polygon P. Assume that each edge has one unique index. And the indices of edges are numbered in monotonically increasing counterclockwise order. Let e_i, $1 \le i \le m$ be the edges of the polygon P that intersect the sides of biting-square centered at v; let v_i, $1 \le i \le m$ be the corresponding intersection points. If one edge e intersects the biting-square twice, we can view the edge e as two imaginary edges such that each one intersects the biting-square once. Notice that m is bounded from above by a constant by Lemma 5.2. Then in constant time, we can sort the intersected edges $\{e_i \mid 1 \le j \le m\}$ in increasing order according to their indices. Furthermore, we sort the intersection points v_i, $1 \le i \le m$, by counterclockwise order on the four edges of $\Pi(v, c_b f(v))$. This can also be done in constant time. Let $\widetilde{v_i}$ be the sorted result.

For example, let's see what all above discussion means by studying the biting of the domain illustrated in Figure 15 (a). The advancing front is represented by polygon $vv_1v_2 \ldots v_{19}v_{20}v_{21}$. Edge v_iv_{i+1} has index i. Here let v_0 denote v. Recall that the priority search tree will return all vertices inside the biting-square centered at v, i.e., v_1, v_2, v_3, v_4, v_5, v_8,



A configuration before the biting-square centered at v is removed.



The remaining domain defined by the new advancing fronts.

Figure 15 : Remove the biting-square centered at a vertex v: We use the solid points to denote the vertices of previous advancing front and use the shaded points to denote the newly introduced vertices in the new advancing fronts.

v_9, v_{20} and v_{21}. By checking the two incident edges of each vertex that is inside the biting-square, we could find all edges that intersect the biting-square. Edges v_5v_6, v_7v_8, v_9v_{10} and $v_{18}v_{19}$ intersect the biting-square $\Pi(v, c_b f(v))$. The intersection points are a, b,c, d, e and f.

Then we show how to construct the new advancing front by just using these intersection points and the corresponding edges.

Procedure: UPDATING THE ADVANCING FRONTS

1. Select an intersected edge going toward the outside of biting-square $\Pi(v, c_b f(v))$, let's say edge E_1. We then have the corresponding intersection point u_1.
2. Search the intersected edges list of the current advancing front to get the edge E_2 that is ordered immediately after the edge that is found in previous step. This can also be done in constant time. We also have the corresponding intersection point u_2 produced by E_2.
3. If there is no intersection point between the two intersection points u_1 and u_2, we then connect u_1 and u_2 along the biting-square to create new edge(s). The new edge(s), together with all edges between edge E_1 and E_2 of current advancing front form a new polygonal advancing front.

   Notice that if u_1 and u_2 are on two different sides of the biting-square, we need create more than one new edges to connect u_1 and u_2 along the biting-square.
4. If there are intersection points between u_1 and u_2, let x be the intersection point on the biting-square that is ordered immediately before u_2. Let E be the edge that produces x. Then we create a new edge by connecting x to u_2 along the side of the biting-square. Then we link the new edge to be the previous edge of edge e, and to be the successive edge of edge E_2.
5. Repeat the above steps 2-4 until intersection point u_1 is checked again.
6. Repeat the above steps 1-5 until all intersection points are checked.

Let's see what the method means by studying the domain illustrated in Figure 15 (a). Assume that we first select the intersected edge v_5v_6 as the edge that is going toward the outside of biting-square, i.e., $E_1 = v_5v_6$. We then have the corresponding intersection point $u_1 = a$. We then find intersected edge v_7v_8 that is ordered immediately after the edge E_1, i.e., $E_2 = v_7v_8$. We also have the corresponding intersection point $u_2 = b$. Notice that there is no intersection point between the two intersection points a and b. We then connect a and b along the biting-square to create a new edge ba. The new edge ba, together with modified edges v_5v_6, v_6v_7 and v_7v_8, forms a new

advancing front. Then we find another outgoing intersected edge v_9v_{10}. Similarly, we will create a new edge dc, and set edge dc as the successive edge of edge $v_{18}d$ and as the previous edge of edge av_{10}. In other words, only three edges are need to be modified to form the new polygon $cv_{10}v_{11}\dots v_{17}v_{18}d$. At last, we will link points f and e and create polygon fev_{19} as a new advancing front. See Figure 15 (b) for the updated advancing fronts.

Thus, if the intersected points are known, we can then update the advancing front in constant time. Notice that all considered edges in this round are all from the same old polygon. Hence the ordering of the edges in new polygon is still monotonically increased. When more than one polygons form the advancing fronts, the same scheme will also correctly construct the new advancing fronts if we assume that there are no holes in the domain. If there are holes in the input domain, we can add some artificial edges to cut the domain into a collection of subdomains without holes. Notice that if there are h holes in the original domain, then we need at least h artificial edges to cut it. The following lemma concludes the above results.

Lemma 5.1 *If the intersected edges is known, we can bite the square centered at interior point v in constant time.*

Notice that, until now, we did not show that there is only a constant number of edges that intersect with the biting-square centered at v. We also did not show that there is only a constant number of vertices that are inside the biting-square $\Pi(v, c_b f(v))$. The remaining of this section is devoted to solve these questions. We first show that only constant number of biting vertices are inside square $\Pi(v, cf(v))$ for any constant c satisfies that $\sqrt{2}\alpha c < 1$.

Lemma 5.2 *There are at most constant biting vertices in any square $\Pi(x, cf(x))$, where $\sqrt{2}\alpha c < 1$.*

PROOF. Let y_1, y_2, $\cdots$, y_k be the k biting vertices in square $\Pi(x, cf(x))$. The fact that y_i is inside $\Pi(x, cf(x))$ implies that

$$f(y_i) \geq f(x) - \alpha||y_i - x|| \geq (1 - \sqrt{2}\alpha c) f(x),$$

and

$$f(y_i) \leq (1 + \sqrt{2}\alpha c) f(x).$$

Then packing-sphere $B(y_i, c_p f(y_i))$ is contained inside the square $\Pi(x, cf(x) + c_p f(y_i))$. In other words, all packing-spheres are contained in a square $\Pi(x, cf(x) + (1 + \sqrt{2}\alpha c)c_p f(x))$. Recall that all packing-spheres do not overlap. Then an area argument implies that

$$\sum_{i=1}^{k} \pi(c_p f(y_i))^2 \leq 4(cf(x) + (1 + \sqrt{2}\alpha c)c_p f(x))^2.$$

It implies that $k \leq \frac{4}{\pi}(\frac{1+\sqrt{2}\alpha c + c/c_p}{1-\sqrt{2}\alpha c})^2$. Notice that c_p is a constant that depends on α and c_b. Then the lemma follows. □

We then show that only constant number of edges of current advancing front can intersect with a biting-square centered at a vertex v. Notice that after the boundary segments are bitten, all edges of the advancing front are the edges of previous biting-squares. Let's consider an intersected edge e, which is the side edge of biting-square centered at vertex u. Therefore the edge e intersects with current biting-square $\Pi(v, c_b f(v))$ implies that biting-squares $\Pi(v, c_b f(v))$ and $\Pi(u, c_b f(u))$ intersect. Thus, we have $||u - v|| \leq \frac{2\sqrt{2}c_b}{1-\sqrt{2}\alpha c_b} f(v)$. In other words, all vertices whose incident edges will possibly intersect with biting-square $\Pi(v, c_b f(v))$ is inside sphere $B(v, \frac{2\sqrt{2}c_b}{1-\sqrt{2}\alpha c_b} f(v))$. Then a simple application of above lemma 5.2 yields that there is only constant number of vertices whose biting-squares intersect with $\Pi(v, c_b f(v))$. Notice that, here we need $\sqrt{2}\alpha \frac{2\sqrt{2}c_b}{1-\sqrt{2}\alpha c_b} < 1$. In other words, the biting constant c_b satisfies that $\alpha c_b < \frac{1}{4+\sqrt{2}}$.

It remains to show that there is only constant number of vertices that are inside the biting-square centered at v. Notice that the vertices of advancing front are the corner vertices of previous biting-squares or the intersection points by previous biting-squares. In the first case, let's assume that $\Pi(u, c_b f(u))$ has one corner points inside $\Pi(v, c_b f(v))$. Then $\Pi(u, c_b f(u))$ intersects with $\Pi(v, c_b f(v))$. In the second case, let's assume that point x is inside $\Pi(v, c_b f(v))$, and it is the intersection point of previous biting-squares centered at u_1 and u_2. Then these two previous biting-squares intersect $\Pi(v, c_b f(v))$, Otherwise vertex x could not be inside $\Pi(v, c_b f(v))$. In both cases, we know that all inside vertices are from the biting-squares that intersect $\Pi(v, c_b f(v))$. As discussed in previous paragraph, we know that there is only a constant number of bitten vertices whose biting-squares intersect with $\Pi(v, c_b f(v))$. Thereofore, these biting-squares will only generate constant number of intersection points. Thus there is only constant number of vertices of current advancing front that are inside $\Pi(v, c_b f(v))$.

Therefore, for every biting, we only need to spend $O(\log n)$ time to update the new advancing front and the priority search tree, where n is the total number of the vertices in previous advancing front. Then we have the following theorem.

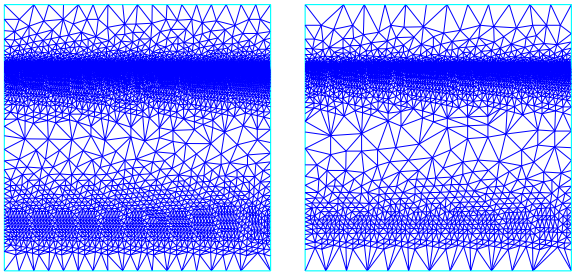
Theorem 5.3 *The biting vertices can be computed in $O(N \log N)$ time, where N is the total number of vertices generated.*

6 Experimental Result

In this section, we give some experimental results to show that the biting method generates well-shaped and well-conformed mesh in two dimensions. The input domain is a 9 by 9 square.

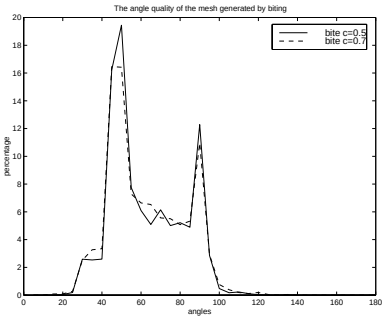
The spacing function used is same as that by George and Borouchaki (1998). In other words, if point has coordinates (x,y), then its control spacing value is defined as following.

$$f(x,y) = \begin{cases} 1 - 0.95\frac{y}{2} & \text{if } y \leq 2 \\ 0.05 \times 20^{\frac{y-2}{2.5}} & \text{if } 2 < y \leq 4.5 \\ 0.2^{\frac{y-4.5}{2.5}} & \text{if } 4.5 < y \leq 7 \\ 0.2 + 0.8(\frac{y-7}{4})^4 & \text{if } 7 < y \leq 9 \end{cases}$$

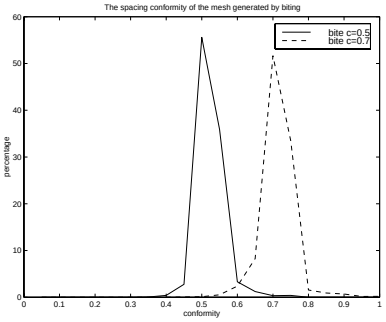


(a) biting by $c_b = 0.5$ (b) biting by $c_b = 0.7$

Figure 16 : Meshes generated by the biting method.



Angle distribution of the meshes.



Vertex conformity distribution of the meshes

Figure 17 : The qualities of the meshes generated by the biting method.

In Figure 16, we show two meshes generated by our biting method: Figure (a) is generated by setting biting constant $c_b = 0.5$; Figure (b) is from $c_b = 0.7$. There are 6728 mesh vertices, and the minimal angle is about 13^o for mesh showed by Figure 16 (a); there are 3435 mesh vertices, and the minimal angle is about 7^o for mesh showed by Figure 16 (b). The quality of the mesh generated by the biting method is illustrated in the Figure 17. Although the theoretic bound for the biting constant is about 0.33, we found that by setting $c_b = 0.5$ (even $c_b = 0.7$), it also generates well-shaped and well-conformed mesh. We also found that the angle distributions of two biting instances are almost same. The conformity of the mesh vertices is almost same as the biting constant, which matches our guess about the conformity.

7 Conclusion

In this paper, we present a variation of the biting method, which combines the merits of the advancing front and the sphere packing methods. It is as simple and as practical as the advancing front methods. It is efficient in time complexity and is simple and straightforward to be extended to three dimensions. The biting scheme is theoretically efficient than the classic advancing front method because it explicitly maintains the set of candidates for new mesh vertices, and it does not have to handle the case when fronts meet each other or one front meets itself. The new scheme resolves this difficulty that occurs at the end of the standard advancing front method. The size of the generated mesh is within a constant factor of the optimal.

Note that the biting square method can be extended to generate three-dimensional meshes: by replacing the biting-square as the biting-cube. Unlike the biting sphere method that has to use complicated method to protect the boundary, the boundary faces and edges are protected easily by biting cube in three dimensions, because the biting-cube naturally pushes the intersection points away from the boundary. The complete prove of the quality of the mesh is omitted here. Recall that for the generated tetrahedron element whose circumcenter is inside the domain, we have the similar theoretical bound on the radius-edge ratio as two-dimensional counterpart. However when the circumcenter is not inside the domain, the proof is much more complicated. Notice that, we can always apply the boundary protection method as Delaunay refinement Shewchuk (1998) to improve the radius-edge ratio of this kind of tetrahedra.

Furthermore, the biting method can also be used to conduct the refinement and coarsening of a mesh. The key observation is that the biting scheme generates a sequence of biting-squares such that the center of a later square is not contained inside any previous biting-squares. Assume that we are given a mesh M and a new spacing function $f(x)$. For each mesh vertex v, the biting-square centered at v is defined as $\Pi(v, c_b f(v))$. The orientation of the biting-square is defined as in the biting scheme. Notice that for refinement, the set of biting-squares defined on mesh vertices of M may not cover the entire domain. Then we bite the remaining pieces of domain by the biting method and the Delaunay triangulation of all mesh vertices and new intro-

duced bitten vertices is the final mesh. For coarsening a mesh M, the biting-squares defined on original mesh vertices may not be ordered such that the center of each later biting-square is not contained inside previous biting-squares. However, we can remove the biting-squares whose centers are contained inside other squares using a similar approach to MIS method, or we can apply the topological sorting method to extract a sequence of biting-squares from M. Then the Delaunay triangulation is constructed as the final mesh.

References

Babuška, I.; Aziz, A. K. (1976): On the angle condition in the finite element method. *SIAM J. Numer. Anal.*, vol. 13(2), pp. 214–226.

Bern, M.; Eppstein, D.; Gilbert, J. R. (1990): Provably good mesh generation. . In *the 31st Annual Symposium on Foundations of Computer Science, IEEE*, pages 231–241, 1990.

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

Chew, L. P. (1997): Guaranteed-quality delaunay meshing in 3d (short version). . In *13th ACM Sym. on Comp. Geometry*, pages 391–393, 1997.

George, P.-L.; Borouchaki, H. (1998): *Delaunay Triangulations and Meshing*. HERMES.

Li, X. Y. (2000): *Sliver-free Three Dimensional Delaunay Mesh Generation*. PhD thesis, University of Illinois at Urbana-Champaign, 2000.

Li, X. Y.; Teng, S. H. (1998): Dynamic load balancing for parallel adaptive mesh refinement. . In *5th International Symposium on Solving Irregularly Structured Problems in Parallel*, pages 144–155, Berkeley, 1998.

Li, X. Y.; Teng, S. H.; Üngör, A. (1999): Biting ellipse to generate anisotropic mesh. . In *8th International Meshing Roundtable*, 1999.

Li, X. Y.; Teng, S. H.; Üngör, A. (2000): Biting: advancing front meets sphere packing. *Int. Jour. for Numerical Methods in Eng*, vol. 49, no. 1-2, pp. 61–81.

Lohrer, R. (1996): Progress in grid generation via the advancing front technique. *Engineering with Computers*, vol. 12, pp. 186–210.

Lohrer, R.; Parikh, P. (1988): Three dimensional grid generation by the advancing-front method. *Int. J. Numer. Meth. Fluids*, vol. 8, pp. 1135–1149.

McCreight, E. M. (1985): Priority search trees. *SIAM Journal on Computing*, vol. 14, pp. 257–270.

Miller, G. L.; Talmor, D.; Teng, S. H. (1998): Optimal coarsening of unstructured meshes. *Journal of Algorithms*. invited and accepted to a special issue for SODA 97.

Miller, G. L.; Talmor, D.; Teng, S. H.; Walkington, N. (1995): A delaunay based numerical method for three dimensions: generation, formulation, and partition. . In *Proc.* 27^{th} *Annu. ACM Sympos. Theory Comput.*, pages 683–692, 1995.

Miller, G. L.; Talmor, D.; Teng, S. H.; Walkington, N. (1998): On the radius–edge condition in the control volume method. *SIAM J. on Numerical Analysis*. accepted and to appear.

Mitchell, S. A.; Vavasis, S. A. (1992): Quality mesh generation in three dimensions. . In *ACM Symposium on Computational Geometry*, pages 212–221, 1992.

Ruppert, J. (1992): A new and simple algorithm for quality 2-dimensional mesh generation. . In *Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 83–92, 1992.

Shewchuk, J. R. (1998): Tetrahedral mesh generation by delaunay refinement. . In *14th Annual ACM Symposium on Computational Geometry*, pages 86–95, 1998.

Shimada, K.; Gossard, D. C. (1995): Bubble mesh: automated triangular meshing of non-manifold geometry by sphere-packing. . In *third Symp. on Solid Modeling and Appl.*, pages 409–419, 1995.

Strang, G.; Fix, G. J. (1973): *An Analysis of the Finite Element Method*. Prentice-Hall.