Generate Good Mesh Respecting to Control Spacing

Xiang-Yang Li * Peng-Jun Wan † Shang-Hua Teng* Alper Ungor*

Summary

This paper presents an efficient algorithm to generate a well-shaped and a well-conformed mesh respecting to a given control spacing $f()$. A control spacing function is used to specify the desired element size. Here, well-shaped mesh means that the circumradius-to-shortest-edge-length ratio of the mesh is bounded from above by a constant; well-conformed means that the element size is within a constant factor of $f()$. Li et. al. *recently proposed a new method named biting to generate high quality mesh by combining the strengths of advancing front and circle packing. In this paper, we show that biting squares instead of circles not only generates high quality mesh but also is very efficient in the sense that the time complexity of the algorithm is $O(n \log n)$ in 2D, where n is the number of mesh vertices generated. And biting square is easier to be extended to 3D than biting circles.*

Introduction

Unstructured mesh generation [1, 7, 5, 10] had been widely used for problems with complex geometry boundaries and with solutions that change rapidly, in order to reduce the problem size. A lot of unstructured mesh generation algorithms had been proposed. Heuristics based methods [7, 11] such as advancing front method, bubble mesh are widely used in engineering. Provable good mesh generation [1, 3, 5, 9, 10] such as Delaunay refinement, sphere packing and quadtree/octree methods are recently proposed by computational geometry community. Recently, the authors had developed a new 2D provably good meshing algorithm called *biting* [5] that combines the strengths of advancing front and the sphere packing method. The algorithm generates a well-shaped and well-conformed mesh respecting to a given smooth control spacing.

It first constructs a *well-spaced* point set by computing a circle packing of the domain and then uses the Delaunay triangulation of this point set as the final mesh. They [5] show that the advancing front method can be used to efficiently construct a quality circle-packing. At a high level, biting method first finds a circle packing of the boundary of the domain and then grows the packing towards the interior of the domain. Each time when a new circle is added to the interior, a larger protection circle is removed (bitten away) from the domain so that no future circles will overlap with this one. The biting method uses advancing front to construct a circle packing instead of the mesh elements themselves. It is showed that biting method does generate a well-spaced point set, whose Delaunay triangulation is well-shaped [5]. The point set generated by biting method can be used for meshless method [4] without constructing the mesh explicitly.

In this paper, we show that biting square instead of circle will also generate a high quality mesh whose size is within a constant factor of the optimal; the time complexity of square-biting method is $O(n \log n)$, where n is the number of the output vertices. And the square-biting method can be extended to three dimensions without worrying much about the boundary protection, which is the main difficulty in the three-dimensional sphere biting method. It is also easier to implement the square-biting method than biting spheres.

Biting to Generate Mesh

Assume that we are given a control spacing function $f()$ to specify the desired elements size. [1] The basic idea of the biting method is to use the advancing front method to construct a circle packing with respect to the spacing $f()$. The input domain boundary is set as the initial advancing front. The biting method selects a vertex of the advancing front, and remove 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 point, 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. This process is called *biting*. The Delaunay triangulation of the biting centers is the final mesh.

*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. Email: xli2, steng, ungor@cs.uiuc.edu
†Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. Email: wan@cs.iit.edu.
[1] Function $f()$ could be the nearest neighbor function defined from the domain geometry or be derived from numerical error analysis.

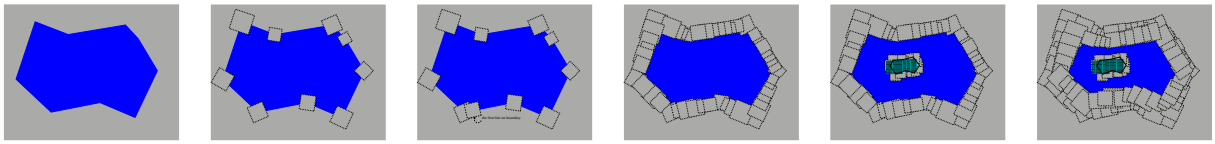


Figure 1: A snapshot of the *biting* scheme: (a) initial PSLG domain; (b) bites only on the vertices of the polygon; (c) first bite of a non-original vertex; (d) biting a layer of the boundary; (e) the first biting in the interior of the domain; (f) biting a layer in the interior of the domain.

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

Usually, the advancing front is represented as a circular list of already placed points. 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. The following is a formal description of the square biting method:

Algorithm `Biting sqaure to generate mesh respecting to spacing` $f()$.

1. [**Initial Front**] Let the boundary of the domain be the initial front, see Figure 1 (a);
2. [**Vertex Protection**]: Bite all the input vertices by removing their biting squares from the interior of the domain, see Figure 1 (b). The orientation of the square is decided by the two incident edges of the vertex. See Figure 2.
3. [**Edge Protection**]: Bite squares centered on the input boundary: choose a vertex $\boldsymbol{x}$ on the front and remove its biting square, see Figure 1 (c) and (d). The biting square is aligned with the boundary edge. See Figure 2. Repeat until all input boundaries are bitten;
4. [**Interior Biting**]: Choose a vertex $\boldsymbol{x}$ on the front and remove its biting square, see Figure 1 (e) and (f). The biting square is aligned with the axes. Repeat until the advancing front is empty.
5. [**Delaunay Triangulation**]: Construct the Delaunay triangulation of the biting centers.

Notice that, after every biting, the intersection points of current biting with previous biting squares are the candidate biting centers later. Hence for protecting the boundary, the biting centered on a boundary vertex is aligned along the boundary. Thus the introduced new candidate biting is not too close to the boundary compared with its control spacing requirement. See Figure 2 (a).

For the input vertices which are on at least two non-perpendicular boundary edges, it is impossible to align the biting with this two incident boundary edges. Assume input vertex $\boldsymbol{x}$ is on two boundary edges $\boldsymbol{xu}_1$ and $\boldsymbol{xu}_2$ defining the domain. Let $\hat{\boldsymbol{x}}$ be the direction of the line that divides the angle $\angle \boldsymbol{u}_1\boldsymbol{x}\boldsymbol{u}_2$. We use the following criteria to select the orientation of the biting square at $\boldsymbol{x}$: if $135^o \leq \angle \boldsymbol{u}_1\boldsymbol{x}\boldsymbol{u}_2 \leq 225^o$, then one side of the biting square is aligned with $\hat{\boldsymbol{x}}$; if $\angle \boldsymbol{u}_1\boldsymbol{x}\boldsymbol{u}_2 \leq 135^o$, or $\angle \boldsymbol{u}_1\boldsymbol{x}\boldsymbol{u}_2 \geq 225^o$ then the diagonal of the biting square is aligned with $\hat{\boldsymbol{x}}$. See Figure 2 (b) and (c). By simple geometry computation, the angle formed by the boundary edges and intersected side edge of the biting square is maximized.

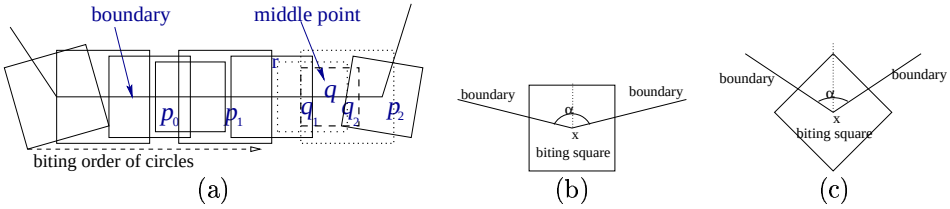


Figure 2: The bitings centered on vertices of input boundary and the orientation of the biting square: (b) $135^o \leq \alpha \leq 225^o$; (c) $\alpha < 135^o$ or $\alpha \geq 225^o$.

Quality Guarantee of Biting

In this section we show that square-biting method generates a well-shaped mesh, the size of the mesh is within a constant factor of the optimal. To facilitate this, we assume that the spacing function $f()$ is α-Lipschitz, i.e., for any two points $\boldsymbol{x}$, $\boldsymbol{y}$, $|f(\boldsymbol{x} - f(\boldsymbol{y})| \leq \alpha||\boldsymbol{x} - \boldsymbol{y}||$.

The Spacing Conformity. We show that the nearest neighbor value of any point is related to the control spacing function $f()$ by a constant factor. This relation enables us to show that the biting scheme generates a mesh whose size is optimal within a constant factor.

Lemma 0.1 *Assume $\sqrt{2}\alpha c_b < 1$, let $c_e = 2\sqrt{2}c_b/(1-\sqrt{2}\alpha c_b)$. The two biting squares $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$ and $\square(\boldsymbol{y}, c_b f(\boldsymbol{y}))$ do not intersect if $||\boldsymbol{x}-\boldsymbol{y}|| \geq c_e f(\boldsymbol{x})$.*

The proof of the lemma is similar to the proof for circle biting in [5]. It implies that there is at least one other mesh vertex generated in $\square(\boldsymbol{x}, c_e f(\boldsymbol{x}))$ other than $\boldsymbol{x}$. We obtain the following lemma.

Lemma 0.2 *For each vertex $\boldsymbol{x}$ of the mesh generated by the biting method, $nn(\boldsymbol{x})$ satisfies*

$$c_b(1-\sqrt{2}\alpha c_b)f(\boldsymbol{x}) \leq nn(\boldsymbol{x}) \leq \frac{2\sqrt{2}c_b}{1-\sqrt{2}\alpha c_b} f(\boldsymbol{x}). \tag{1}$$

PROOF. First, if there is no other mesh vertex other than $\boldsymbol{x}$ that lies inside biting square $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$, then $c_b f(\boldsymbol{x}) \leq nn(\boldsymbol{x})$. If there is at least one mesh vertex $\boldsymbol{y}$ inside $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$, vertex $\boldsymbol{y}$ must be bitten before $\boldsymbol{x}$. We have $c_b f(\boldsymbol{y}) \leq ||\boldsymbol{y}-\boldsymbol{x}|| \leq \sqrt{2}c_b f(\boldsymbol{x})$. Because $f()$ is α-Lipschitz, $f(\boldsymbol{y}) \geq (1-\sqrt{2}\alpha c_b)f(\boldsymbol{x})$. Hence, $||\boldsymbol{y}-\boldsymbol{x}|| \geq c_b(1-\sqrt{2}\alpha c_b)f(\boldsymbol{x})$.

Recall, for any point $\boldsymbol{y}$ in the exterior of the circle $\square(\boldsymbol{x}, c_e f(\boldsymbol{x}))$, the biting square of $\boldsymbol{y}$ does not overlap with that of $\boldsymbol{x}$. Hence, if $nn(\boldsymbol{x}) > c_e f(\boldsymbol{x})$, then the boundary of $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$ is not covered by any other biting squares. The lemma then follows from property that every point in the domain is covered by at least one biting square. □

Similar to [5], we also can prove that the nearest neighbor function of each non-mesh point in the domain is linearly related to $f()$.

Lemma 0.3 *Assume point $\boldsymbol{y} \in \Omega$ is not a mesh vertex. Then $nn(\boldsymbol{y})$ satisfies*

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

The $nn()$ function deduced from the mesh generated by the biting method is within a constant factor of $f()$ implies the following theorem.

Theorem 0.4 *Size of the mesh generated by the* biting *method is within a constant factor of the optimal possible.*

The Minimal Angle. By showing the centers defining a sphere packing, we can show that the biting scheme generates a well shaped mesh, see [5]. However, the constant bound on the minimal angle so derived may be too small. In this section, we provide a better analysis of the lower bound of the minimal angle.

In our analysis, as in [5], we divide the triangle 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.

The following lemma bounds the minimal angle of the triangles in the first subset.

Lemma 0.5 *Let $\triangle \boldsymbol{pqr}$ be a triangle of the first subset. Let l be its shortest edge length. Let R be its circumradius. Then*

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

PROOF. Let $\boldsymbol{c}$ be the its circumcenter. Assume $\boldsymbol{c} \in \square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$. Then $||\boldsymbol{c}-\boldsymbol{x}|| \leq \sqrt{2}c_b f(x)$. Because the mesh is a Delaunay triangulation, $\boldsymbol{x}$ is not in the interior of circumcircle of $\triangle \boldsymbol{pqr}$, i.e., $||\boldsymbol{x}-\boldsymbol{c}|| \geq R$. Thus $f(\boldsymbol{x}) \geq R/(\sqrt{2}c_b)$. Because $f()$ is α-Lipschitz, $f(\boldsymbol{c}) \geq f(\boldsymbol{x}) - \alpha||\boldsymbol{c}-\boldsymbol{x}||$. Also because $||\boldsymbol{c}-\boldsymbol{x}|| \leq \sqrt{2}c_b f(x)$, $f(\boldsymbol{c}) \geq (1-\sqrt{2}\alpha c_b)f(\boldsymbol{x}) \geq (1-\sqrt{2}\alpha c_b)R/(\sqrt{2}c_b)$.

Without loss of generality, assume $||\boldsymbol{p}-\boldsymbol{q}|| = l$, and $\square(\boldsymbol{q}, c_b f(\boldsymbol{p}))$ is bitten before $\square(\boldsymbol{q}, c_b f(\boldsymbol{q}))$, which implies $l \geq c_b f(\boldsymbol{p})$. Because $f()$ is α-Lipschitz, $f(\boldsymbol{c}) \leq f(\boldsymbol{p}) + \alpha R \leq l/c_b + \alpha R$. Using $f(\boldsymbol{c}) \geq (1-\sqrt{2}\alpha c_b)R/(\sqrt{2}c_b)$, we have $(1-\sqrt{2}\alpha c_b)R/(\sqrt{2}c_b) \leq (\sqrt{2}l + \alpha c_b R)/c_b$. Then the lemma follows. □

We now consider elements in the second subset. Let $\triangle \boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3$ be a mesh element from the second subset. Let $\mathcal{C}$ be its circumcircle. Let $\boldsymbol{c}$ be the center of $\mathcal{C}$. Let R be the radius of $\mathcal{C}$. Assume $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are the two closest mesh vertices

on the boundary that separates $\triangle \boldsymbol{xyz}$ and $\boldsymbol{c}$. Note that $\boldsymbol{p}_1$ and/or $\boldsymbol{p}_2$ may be one of the vertices among $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$. Let $\boldsymbol{r}$ be the intersection points of $\square(\boldsymbol{p}_1, c_b f(\boldsymbol{p}_1))$ with $\square(\boldsymbol{p}_2, c_b f(\boldsymbol{p}_2))$ that is inside the domain. Let $h_{\boldsymbol{r}}$ be the distance of point $\boldsymbol{r}$ to boundary segment $\boldsymbol{p}_1\boldsymbol{p}_2$.

Clearly, there is no any mesh vertex between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, and angle $\angle \boldsymbol{p}_1\boldsymbol{x}_i\boldsymbol{p}_2$ is obtuse for $i = 1, 2, 3$. It is easy to show that $\angle \boldsymbol{p}_1\boldsymbol{x}_i\boldsymbol{p}_2 \leq \angle \boldsymbol{p}_1\boldsymbol{r}\boldsymbol{p}_2$. It implies that $\angle \boldsymbol{p}_1\boldsymbol{r}\boldsymbol{p}_2$ is obtuse. Without loss of generality, assume that $\boldsymbol{p}_1$ is bitten before $\boldsymbol{p}_2$. By checking all possible configurations of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, see Figure 3, only case (c) is possible, i.e., $\boldsymbol{p}_2$ is not introduced by biting $\square(\boldsymbol{p}_1, c_b f(\boldsymbol{p}_1))$. (The proof is omitted here due to the space restriction.)

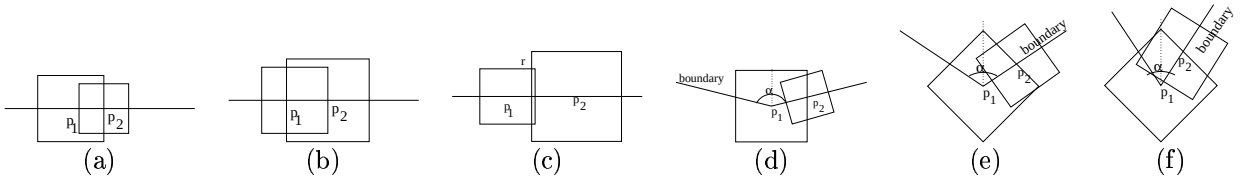


Figure 3: The configurations of adjacent bitings centered on vertices of input boundary: (a): $f(\boldsymbol{p}_2) < f(\boldsymbol{p}_1)$ and $\boldsymbol{p}_2$ is on biting square of $\boldsymbol{p}_1$; (b): $f(\boldsymbol{p}_2) \geq f(\boldsymbol{p}_1)$ and $\boldsymbol{p}_2$ is on biting square of $\boldsymbol{p}_1$; (c): $\boldsymbol{p}_2$ is not on biting square of $\boldsymbol{p}_1$; (d): $\boldsymbol{p}_1$ is a corner and had a large angle α; (e): $\boldsymbol{p}_1$ is a corner and had a small angle α; (f): another case when $\boldsymbol{p}_1$ is a corner and had a small angle α.

For case (c) of Figure 3, we have $h_{\boldsymbol{r}} = c_b f(\boldsymbol{p}_1)$. Because two biting squares are overlapped, and $\angle \boldsymbol{p}_1\boldsymbol{r}\boldsymbol{p}_2$ is obtuse, we have $2c_b f(\boldsymbol{p}_1) \leq ||\boldsymbol{p}_1 - \boldsymbol{p}_2|| \leq \frac{2c_b}{1-\alpha c_b} f(\boldsymbol{p}_1)$. Let $L = ||\boldsymbol{p}_1 - \boldsymbol{p}_2||/2$, $t = \alpha c_b$. Then $h_{\boldsymbol{r}} \geq (1 - \alpha c_b)L$ and $h_{\boldsymbol{r}} < L$. Notice that $R \leq (h_{\boldsymbol{r}}^2 + L^2)/(2h_{\boldsymbol{r}})$. Then $R \leq \frac{t^2-2t+2}{2-2t} L$. Assume that the smallest edge of $\bigtriangledown \boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3$ is incident to point $\boldsymbol{x}_1$. Then $l \geq c_b f(\boldsymbol{x}_1) \geq c_b(f(\boldsymbol{p}_1) - \alpha||\boldsymbol{x}_1 - \boldsymbol{p}_1||)$. From $\angle \boldsymbol{p}_1\boldsymbol{x}_1\boldsymbol{p}_2$ is obtuse, $l \geq c_b f(\boldsymbol{p}_1) - t||\boldsymbol{p}_1 - \boldsymbol{p}_2|| \geq (1 - 3t)L$. Then $R/l \leq \frac{t^2-2t+2}{6t^2-8t+2}$. The following theorem comes from the fact that $\frac{t^2-2t+2}{6t^2-8t+2} < \frac{\sqrt{2}}{1-2\sqrt{2}t}$ when $0 < t < 0.28$.

Theorem 0.6 *The minimal angle θ of any triangle generated by the biting method satisfies, if $t < 0.28$,*

$$\sin(\theta) = \frac{l}{2R} \geq \frac{\sqrt{2}}{4} - t. \tag{4}$$

Numerical Robustness. To construct the Delaunay triangulation of the biting centers, we often have to do the incircle test to see if we need flip some edges. Due to the roundoff error, the incircle test is not always consistent, which in turn will cause the Delaunay kernel to result in a non-valid triangulation. To address this problem, we propose a new method, named ρ-Delaunay, which generates high quality mesh by combining it with our biting method.

Let ρ be a positive constant less than 1. A triangulation is ρ-Delaunay [6] if for every simplex, the sphere $B(\boldsymbol{c}, \rho R)$ is empty, where $\boldsymbol{c}$ is simplex's circumcenter, R is simplex's circumradius. Hence the traditional Delaunay triangulation is 1-Delaunay under our definition. To make a mesh ρ-Delaunay, we just check every edge of the mesh: if it does not satisfy the ρ-local-Delaunay property, we just flip the edge. Then the next theorem shows that the ρ-Delaunay triangulation of the point set generated by our biting method generates a well-shaped mesh.

Theorem 0.7 [**Well-shaped ρ-Delaunay**] *The circumradius-to-shortest-edge-length ratio is at most $\frac{\sqrt{2}}{(1-\sqrt{2}t)\rho - \sqrt{2}t}$.*

Proof. Let $\boldsymbol{c}$ be the circumcenter of $\triangle \boldsymbol{pqr}$. Assume $\boldsymbol{c} \in \square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$. Noticed that $||\boldsymbol{x} - \boldsymbol{c}|| \geq \rho R$ because of the ρ-Delaunay triangulation property. As showed in Lemma 0.5, we have $f(\boldsymbol{c}) \geq (1 - \sqrt{2}\alpha c_b)\rho R/(\sqrt{2}c_b)$, and $f(\boldsymbol{c}) \leq l/c_b + \alpha R$. Then the theorems follows, if $\rho \geq \frac{\sqrt{2}t}{1-\sqrt{2}t}$ and $t < \sqrt{2}/4$. □

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 output vertices. We first show that there are only a constant number of vertices in any biting square $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$ by using the fact that packing circles do not intersect [5]. Here a packing circle at point $\boldsymbol{x}$ as $B(\boldsymbol{x}, c_p f(\boldsymbol{x}))$, where $c_p = c_b/(2 + \alpha c_b)$.

Lemma 0.8 *There are at most constant biting vertices in any biting square $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$.*

Proof. Let $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\cdots$, $\boldsymbol{y}_k$ be the t biting vertices in $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$, then $||\boldsymbol{y}_i - \boldsymbol{x}|| \geq c_b f(\boldsymbol{y}_i)$ for $1 \leq i \leq t$. Note that $\boldsymbol{y}_i$ is inside $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$, then $f(\boldsymbol{y}_i) \geq f(\boldsymbol{x}) - \alpha||\boldsymbol{y}_i - \boldsymbol{x}|| \geq (1 - \sqrt{2}\alpha c_b) f(\boldsymbol{x})$, and $f(\boldsymbol{y}_i) \leq (1 + \sqrt{2}\alpha c_b) f(\boldsymbol{x})$.

Packing circle $\square(\boldsymbol{y}_i, c_p f(\boldsymbol{y}_i))$ is contained in $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}) + c_p f(\boldsymbol{y}_i))$ implies that all packing circles are contained in $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}) + (1+\sqrt{2}\alpha c_b) c_p f(\boldsymbol{x}))$. All packing circles do not overlap implies

$$\sum_{i=1}^{k} \pi (c_p f(\boldsymbol{y}_i))^2 \leq 4(c_b f(\boldsymbol{x}) + (1+\sqrt{2}\alpha c_b) c_p f(\boldsymbol{x}))^2.$$

Then $k \leq \frac{4}{\pi}(\frac{3+(1+\sqrt{2})\alpha c_b}{1-\sqrt{2}\alpha c_b})^2$. □

We analyze the complexity of the algorithm by considering the complexity of every stage of the biting. For biting input vertices, the spacing function definition makes sure that there are no intersections among all biting squares centered on input vertices. We can bite the squares centered at input vertices at constant time: just two incident edges of that vertex can intersect with its biting square. Same arguments hold for the biting of vertices at the input boundaries. To bite squares centered at vertices in the interior of the input domain, the following observation is important: the orientation of the biting squares does not affect the theoretical bound of the generated mesh quality. Hence, to simplify the analysis, we always align the biting squares along the coordinates.

Assume that we want to bite vertex $\boldsymbol{x}$, and had computed all edges in current advancing fronts which intersect with the edges of $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$. Assume that the advancing front is compromised of p polygons $\mathbf{P}_1$, $\mathbf{P}_2$, $\cdots$, $\mathbf{P}_p$. Assume that the edges of each polygon are numbered in monotonically increasing clockwise order. Let $e_{i,j}$, $1 \leq j \leq m_i$ be the edges of polygon $\mathbf{P}_i$ which intersect the edges of biting square, and $v_{i,j}$, $1 \leq j \leq m_i$ be the corresponding intersection points. Notice that $\sum m_i$ is bounded by a constant by Lemma 0.8. Then in constant time, we can sort the intersected edges $\{e_{i,j} | 1 \leq j \leq m_i\}$ for each $\mathbf{P}_i$ in increasing order. We sort the intersection points $v_{i,j}$, $1 \leq j \leq m_i$, $1 \leq i \leq p$ by the clockwise order in the four edges of $\square(x, c_b f(x))$. Let $\widetilde{v_{i,j}}$ be sort result. Then we update the advancing front as following (proof is omitted here due to space restriction).

1. Select an edge going toward the outside of $\square(\boldsymbol{x}, c_b f(\boldsymbol{x}))$, let's say edge $e_{1,1}$. Then $v_{1,1}$ is the intersection points.
2. Search the intersected edges list of polygon $\mathbf{P}_1$ to get edge $e_{1,2}$ which is ordered immediately after edge $e_{1,1}$. Note edge $e_{1,2}$ produces an intersection point $v_{1,2}$.
3. If there is no intersection points between $v_{1,1}$ and $v_{1,2}$, i.e., $\widetilde{v_{1,1}}$ and $\widetilde{v_{1,2}}$ are adjacent, then we connect $v_{1,1}$ and $v_{1,2}$ along the biting square to form a polygon together with all edges in $\mathbf{P}_1$ between edge $e_{1,1}$ and $e_{1,2}$. Notice here we assume that the polygon is stored as linked list. From the address of edge $e_{1,1}$ and $e_{1,2}$, we can get the link of all edges of $\mathbf{P}_1$ starting at $e_{1,1}$ and ending at $e_{1,2}$ in constant time.
4. If there are intersections between $v_{1,1}$ and $v_{1,2}$, let v be the intersection points on the biting square which is ordered immediately before $v_{1,2}$. Let e be the edge that produces v. Because we assume that there is no hole inside a polygon, hence edge e must be from polygon $\mathbf{P}_1$. [2] Then we link the edge $e_{1,2}$ to edge e.
5. Repeat the above steps 2-4 until intersection point $v_{1,1}$ is checked again.
6. Repeat the above steps until all intersection points are checked.

Thus, if the intersected points are known, then we can update the advancing front in constant time. Note that because all considered edges in this round are all from same old polygon, hence the ordering of the edges in new polygon is still monotonically increased. The following lemma concludes the above results.

Lemma 0.9 *If the intersected edges are known, we can bite the square at interior point $\boldsymbol{x}$ in constant time.*

Then we show how to find the edges of previous advancing front which intersect the current biting square. Notice that we can find all intersected edges if we find all previous biting vertices inside the current biting square. Hence, the question becomes how to report all vertices in a coordinates-aligned square.

Notice that there is no any order requirement for interior bitings. We can select special vertex in current front for new biting. If we select the vertex with the largest y value, then we can 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 [8], 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 $O(n)$. More importantly, the priority search tree can be updated in $O(\log n)$ time per deletion of a vertex and insertion of a vertex. Notice that there are only a constant intersections to be reported, hence only a constant number of deletions and insertions are necessary fro updating the current advancing front. Then we have the following lemma.

[2] 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.

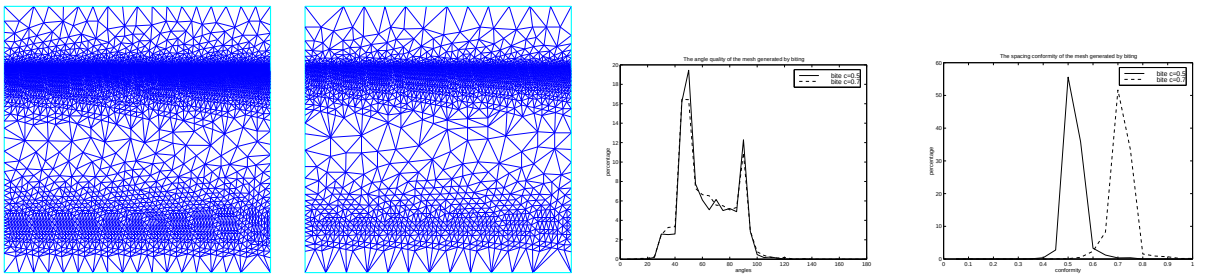
Lemma 0.10 *The intersected edges can be computed in $O(\log n)$ time, where n is the total number vertices in previous advancing front; after biting the current square, the new advancing front can be updated in $O(\log n)$ time.*

Theorem 0.11 *The biting vertices can be computed in $O(N \log N)$ time, where N is the total number of vertices generated.*

Experimental Result

In this section we give some experimental results to show that the biting method generates well shaped and well-conformed mesh. The input domain is a 9 by 9 square. The spacing function used is same as that in [2]. 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 $c_b = 0.7$ (c) angle distribution (d) conformity distribution

Figure 4: Meshes generated by the biting method and the quality of the meshes.

In Figure 4, 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 points, and the minimal angle is about 13^o for mesh showed by Figure 4 (a); there are 3435 mesh points, and the minimal angle is about 7^o for mesh showed by Figure 4 (b). The quality of the mesh generated by the biting method is illustrated in the Figure 4. 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 intuition about the conformity.

Note that the biting square method can be extended to generate 3D mesh: by replacing the biting square as the biting cube. Similarly, the boundary face and edges are naturally protected because of the biting cubes push the intersection points away from the boundary, if we align the bitings centered on boundary faces with the corresponding face. But the time complexity is higher than that of 2D.

References

1. T. D. Blacker. Paving: a new approach to automated quadrilateral mesh generation. *Int. Jour. for Numerical Methods in Eng*, 32:811–847, 1991.
2. Paul-Louis George and Houman Borouchaki. *Delaunay Triangulations and Meshing.* HERMES, 1998.
3. X.-Y. Li. Funtional delaunay refinement. Submitted for publication.
4. X.-Y. Li, S. H. Teng, and A. Ungor. Point placement for meshless methods using sphere packing and advancing front methods. ICES00:International conference on Computational Engineering Science.
5. X. Y. Li, S. H. Teng, and A. Üngör. Biting: advancing front meets sphere packing. *Int. Jour. for Numerical Methods in Eng*, 1999.
6. X. Y. Li, S. H. Teng, and A. Üngör. Biting ellipse to generate anisotropic mesh. In *8th International Meshing Roundtable*, 1999.
7. R. Lohrer. Progress in grid generation via the advancing front technique. *Engineering with Computers*, 12:186–210, 1996.
8. Edward M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14:257–270, 1985.
9. G. L. Miller, D. Talmor, S.-H. Teng, N. Walkington, and H. Wang. Control volume meshes using sphere packing: generation, refinement, and coarsening. In *5th International Meshing Roundtable*, pages 47–61. Sandia National Laboratories, 1996.
10. J. R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *14th Annual ACM Symposium on Computational Geometry*, pages 86–95, 1998.
11. K. Shimada. *Physically-based Mesh Generation:Automated Triangulation of surfaces and Volumes via Bubble Packing.* PhD thesis, MIT, Cambridge, 1993.