Guaranteed-Quality All-Quadrilateral Mesh Generation with Feature Preservation

Xinghua Liang, Mohamed S. Ebeida, and Yongjie Zhang*

Department of Mechanical Engineering, Carnegie Mellon University, USA
jessicaz@andrew.cmu.edu

Abstract. In this paper, a quadtree-based mesh generation method is described to create guaranteed-quality, geometry-adapted all-quadrilateral meshes with feature preservation for arbitrary planar domains. Given point cloud, our method generates all-quad meshes with these points as vertices and all the angles are within $[45°, 135°]$. For given planar curves, quadtree-based spatial decomposition is governed by the curvature of the boundaries and narrow regions. 2-refinement templates are chosen for local mesh refinement without creating any hanging nodes. A buffer zone is created by removing elements around the boundary. To guarantee the mesh quality, the angles facing the boundary are improved via template implementation, and two buffer layers are inserted in the buffer zone. It is proved that all the elements of the final mesh are quads with angles between $45° \pm \varepsilon$ and $135° \pm \varepsilon$ ($\varepsilon \leq 5°$) with the exception of badly shaped elements that may be required by the specified geometry. Sharp features and narrow regions are detected and preserved. Furthermore, boundary layer meshes are generated by splitting elements of the second buffer layer. We have applied our algorithm to a set of complicated geometries, including the Lake Superior map and the air foil with multiple components.

Keywords: Guaranteed quality, all-quadrilateral mesh, quadtree, sharp feature, narrow region, boundary layer.

1 Introduction

Provably good-quality triangular mesh generation methods were well developed for planar and curved surfaces. However, quadrilateral elements are preferred in finite element analysis due to their superior performance. Although it was proved that any planar n-gon can be meshed by $O(n)$ quads with all the angles bounded between $45° - \varepsilon$ and $135° + \varepsilon$ [5], fewer algorithms exist in the literature for all-quad mesh generation, and most of these algorithms are heuristic. Circle-packing techniques have been developed to construct quads with no angles larger than 120° for polygon interiors, but with no bound on the smallest angle [4]. Later, a balanced quadtree was utilized to generate a quad mesh with bounded minimum angle 18.43°, but the

* Corresponding author.

maximum angle bound is 180° [1]. In this paper, we present an approach to generate guaranteed-quality all-quad meshes for given point cloud or planar curves, in which all the angles of any element are within $[45° \pm \varepsilon, 135° \pm \varepsilon]$, where $\varepsilon \leq 5°$, except badly shaped elements that may be required by the specified geometry.

Our algorithm generates unstructured adaptive all-quad meshes. For given point cloud, we firstly define a size function based on the relative location of these points, and then generate adaptive quadtree using 2-refinement templates. Three cases are considered to produce all-quad meshes conforming to the given point cloud, with all the element angles within $[45°, 135°]$.

For planar curves, six steps are adopted to construct guaranteed-quality all-quad meshes. Firstly we decompose each curve into a set of line segments based on its curvature; Secondly, a strongly balanced quadtree is constructed and hanging nodes are removed using 2-refinement templates. The element size is governed by the curvature of the boundary and narrow regions; Thirdly, elements outside and around the boundary are removed to create the quadtree core mesh and a buffer zone; Next, we design four categories of templates to adjust the boundary edges and therefore improve the angles facing the boundary in the quadtree core mesh; Then the angular bisectors are used to construct the first buffer layer; Finally the points generated during the first layer construction are projected to the boundary and form the second buffer layer. It is proved that all the angles in the constructed mesh are within $[45° \pm \varepsilon, 135° \pm \varepsilon]$ $(\varepsilon \leq 5°)$ for any planar smooth curves. A few bad angles may be required to preserve sharp features such as small angles on the boundary. Boundary layers are generated by splitting the second buffer layer.

We have applied our algorithm to a set of complicated geometries, including the Lake Superior map and the air foil with multiple components. Our robust algorithm efficiently deals with curves in large-scale size, and generates meshes with guaranteed quality while minimizing the number of elements.

The reminder of this paper is organized as follows: Section 2 reviews the related work on quad mesh generation; Section 3 describes the guaranteed-quality meshing of point cloud; Section 4 explains the detailed algorithm for guaranteed-quality meshing of smooth curves; Section 5 talks about sharp feature and boundary layer generation; Section 6 shows two application results; Section 7 presents our conclusion and future work.

2 Previous Work

There are three main categories for direct unstructured all-quad mesh generation [13]: domain decomposition, advancing front and grid-based methods.

Domain Decomposition Methods: The domain is divided into simpler convex or mappable regions, and then template-based, mapping or geometric algorithms are utilized to generate the mesh for each of these regions. Domain decomposition can be achieved by various techniques. Tam and Armstrong

[18] introduced medial axis decomposition. Joe [10] decomposed the domain based on geometric algorithms. Quadros *et al.* [14] introduced an algorithm that couples medial axis decomposition with an advancing front method. In general, these methods produce high quality meshes but they are not robust and may require a great deal of user interaction especially if the domain has non-manifold boundaries.

Advancing Front Methods: This approach starts with the initial placement of nodes on the boundaries of the domain. Quad elements are then formed by projecting each edge on the front towards the interior and a new front is formed using edges on the new boundary. This process is repeated recursively until the domain is completely covered with quads. Zhu [25] is among the first to propose a quadrilateral advancing front algorithm. In his approach, two triangles are created using the traditional advancing front methods then combined to form a single quad. Blacker and Stephenson [6] introduced the paving algorithm in which they place a complete row of quads next to the front toward the interior. White and Kinney [19] enhanced the robustness of the paving algorithm through creating individual quads rather than a complete row. The advancing front methods generate near-boundary elements with high quality. However, the closure algorithms for the elements at the interior are still unstable, especially if the two overlapping elements have large difference in size. In such instance, heuristic decisions are made and these usually generate elements with poor quality in this region. Moreover, the detection and resolution of the closure regions can be very time consuming and sensitive to floating point errors.

Grid-Based Methods: A grid-based method starts with a uniform Cartesian background grid or a quadtree structure generated using the local feature sizes. Quads, conforming to the domain boundaries, are then fitted into that grid. Baehmann *et al.* [2] modified a balanced quadtree to generate a quad mesh for an arbitrary domain. Zhang *et al.* developed an octree-based isocontouring method to generate adaptive quadrilateral and hexahedral meshes [22, 21, 24]. Schneiders *et al.* [16] used an isomorphism technique to conform an adaptive tree structure to the object boundaries. Grid-based algorithms are robust but often generate poor quality elements at the boundary.

Quality Improvement: In finite element analysis, small angles within the mesh usually lead to ill-conditioned linear systems. Further problems are caused due to elements with angles close to 180°. Therefore, a post-processing step is crucial to improve the overall quality of the elements. Smoothing and clean-up methods are the two main categories of mesh improvement. Smoothing methods relocate vertices without changing the connectivity [7]. These methods are simple and easy to implement. However, they are heuristic and sometimes invert or degrade the local elements. To solve this problem, optimization-based smoothing methods were proposed [8]. In this approach, a node is relocated at the optimum location based on the local gradient of the surrounding element quality. Optimization-based methods provide much

better mesh quality but they are not practical due to excessive amount of computations. For this reason such methods are usually combined with a Laplacian smoothing technique [9]. Surface feature preservation represents another challenging problem. Methods based on local curvature and volume preserving geometric flows are presented in [3, 23] to identify and preserve the main surface features. Clean-up methods for quad meshes [17, 11] are utilized to improve the node valence. Pillowing [12] is used to ensure that any two adjacent quads share at most one edge.

All of these quality improvement techniques do not guarantee any bounds for the element angles in the final mesh. In this paper, we will present an approach to generate all-quad meshes with guaranteed quality.

3 Guaranteed-Quality Meshing of Point Cloud

Given N points in a planar domain Ω, we aim to find a quad mesh M that includes the given points as vertices and all the elements have the minimum and maximum angle bounds. Let $\mathcal{X}$ denote the set of input points, we start by the spatial decomposition using 2-refinement templates [15] and the strongly balanced quadtree algorithm [20], which means the quadtree level difference between any two neighboring cells is ≤ 1. Compared to 3-refinement, 2-refinement provides a gradual transition and preserves element angles. It is well-known that any quad mesh has an even number of boundary nodes. This fact is utilized in eliminating each pair of the hanging nodes in the quadtree structure using 2-refinement templates. This algorithm considers the vertex sequence $\{V_1, V_2, \ldots, V_{2k}\}$ on the boundaries. Half of the vertices in this list are set to be active and the other half are set to be inactive such that each active vertex is followed by an inactive vertex. Any element that contains only one active node is refined using the transition template in Fig. 1(b) while any element containing two or more active nodes is refined using the refinement template shown in Fig. 1(c).

During the spatial decomposition, an additional requirement is applied so that each cell, c, containing a point $P \in \mathcal{X}$ is surrounded by eight empty cells of the same size. We then subdivide c into 16 identical regions, B_i ($i = 1, 2, \ldots, 16$). One region of these will contain the point P. Here we consider three regions only as shown in Fig. 2(a). The other possibilities can be obtained by symmetry. If $P \in B_1$, we set the grid node $V_1 = P$ as shown in Fig. 2(b). When $P \in B_2 \cup B_3$, the local refinement shown in Fig. 2(c) is carried out first. If $P \in B_2$, we set the grid node $V_2 = P$ and adjust V_3 vertically as shown in Fig. 2(d). Similarly, if $P \in B_3$ as shown in Fig. 2(e), we set $V_3 = P$ and adjust V_2 correspondingly. As proved in Lemma 1, this algorithm guarantees all the angles in the final mesh are within $[45°, 135°]$, and the maximum aspect ratio is 1.5 (*The aspect ratio* of a quad is defined as the longest edge length over the shortest edge length). Fig. 1(d-e) show two testing cases.

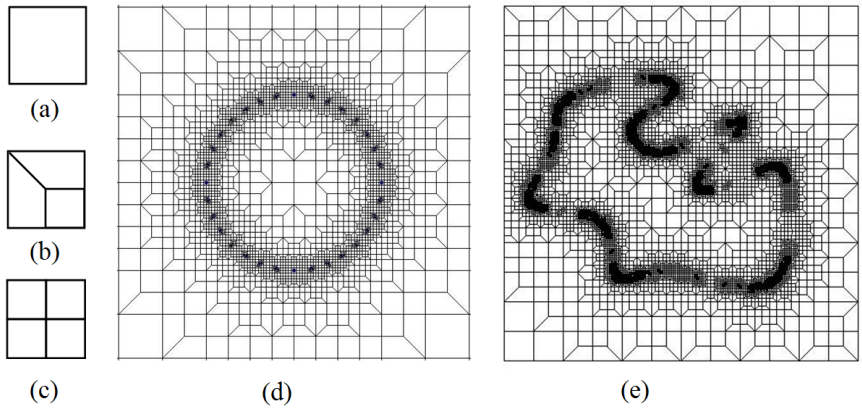


Fig. 1. 2-refinement templates and mesh generation conforming to a set of points. (a) A quadtree cell at Level i; (b) The transition template; (c) The refinement template; (d) A quad mesh of 32 points distributed uniformly along a circle; and (e) A quad mesh of 215 points distributed non-uniformly along a curve.

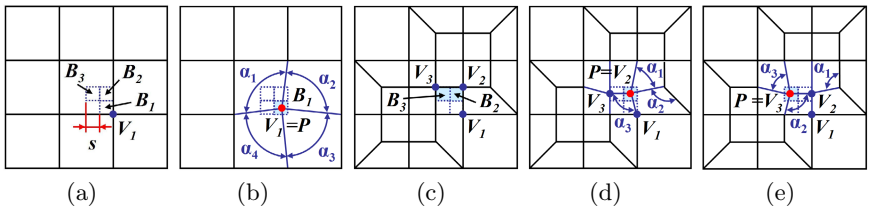


Fig. 2. All-quad mesh generation conforming to a set of points. (a) The cell containing a point is subdivided into 16 regions, supposing $P \in B_1 \cup B_2 \cup B_3$; (b) When $P \in B_1$, $V_1 = P$; (c) Local refinement for $P \in B_2 \cup B_3$; (d) When $P \in B_2$, $V_2 = P$ and V_3 is adjusted; and (e) When $P \in B_3$, $V_3 = P$ and V_2 is adjusted.

Lemma 1. *The method above results in a quad mesh with angles between* $45°$ *and* $135°$ *with the maximum aspect ratio of 1.5.*

Proof. In Fig. 2(b), the maximum displacement that V_1 can move is the diagonal of the square region B_1, $\sqrt{2}s$. The resulting angles α_1, α_2, α_3 and α_4 due to this displacement have a minimum value of $67°$ and a maximum value of $127°$. The maximum aspect ratio of the quad around P is 1.3. When $P \in B_2 \cup B_3$, the local refinement in Fig. 2(c) guarantees that both V_2 and V_3 are surrounded initially by four angles of $90°$. In Fig. 2(d-e), V_2 and V_3 are adjusted. One node is moved to P, and the other node is adjusted to maintain the slope between V_2 and V_3. The worst case here is similar to moving a corner of a square cell of size s a distance of $\sqrt{2}s$. The resulting angles α_1, α_2, and α_3 due to this displacement have a minimum value of $53°$ and a maximum value of $108°$. The maximum aspect ratio of the quads around P is 1.5. Therefore,

our algorithm produces all-quad meshes with angles between 45° and 135° with the maximum aspect ratio of 1.5. ◇

4 Guaranteed-Quality Meshing of Smooth Curves

Given a planar domain Ω and closed smooth curves $\mathcal{C}$ represented by cubic splines, a set of points or polygons, we aim to generate a guaranteed-quality all-quad mesh for the regions enclosed by $\mathcal{C}$. Six steps are applied as shown in Fig. 3, including (1) curve decomposition, (2) adaptive quadtree construction, (3) buffer zone clearance, (4) template implementation, (5) first buffer layer construction, and (6) second buffer layer construction.

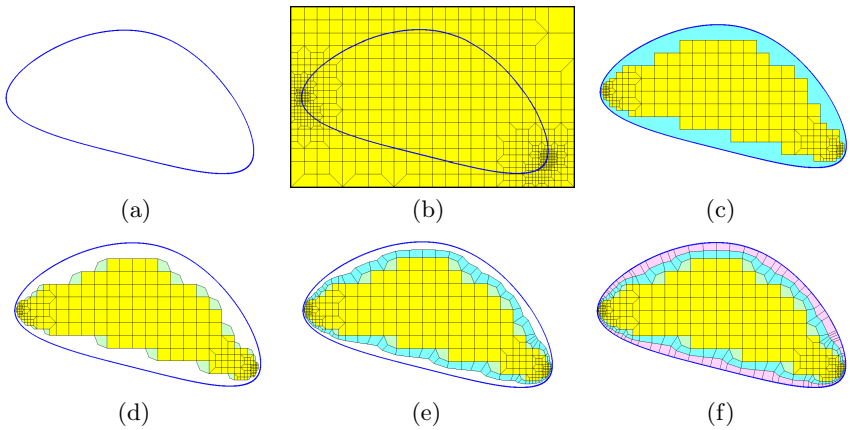


Fig. 3. Flow chart of guaranteed-quality mesh generation. (a) The input curve; (b) Adaptive quadtree construction; (c) Buffer zone clearance; (d) Template implementation; (e) First buffer layer construction; and (f) Second buffer layer construction.

4.1 Curve Decomposition

Given closed smooth curves $\mathcal{C}$ represented by cubic splines, we first decompose $\mathcal{C}$ into a set of piecewise-linear line segments using three criteria: (1) The angle between two neighboring line segments is $\leq 5°$; (2) The approximation error of each line segment is less than a given threshold (i.e., 0.01); and (3) Each curve in $\mathcal{C}$ is represented by line segments in the clockwise direction. As a result, non-uniform points $\mathcal{X}$ are created based on the curve local curvature.

4.2 Adaptive Quadtree Construction

For a set of points $\mathcal{X}$, a size function is first defined. For example, we define the size function as $s_i = min(d_{ij})$, where d_{ij} is the distance between two points i and j ($i, j \in \mathcal{X}$ and $i \neq j$). The basic concept of a spatial representation

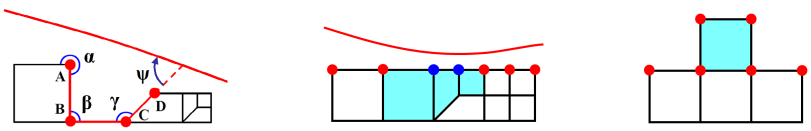
consists of enclosing the points $\mathcal{X}$ in a bounding box, denoted as $\mathcal{B}(\Omega)$, which is corresponding to the root of the spatial decomposition quadtree. This box is either a square with one cell or a rectangle with one row of square cells. Each cell is recursively subdivided based on the size function and the strongly balanced quadtree algorithm. A cell, c, is *crowded* if its size is greater than the size function at any point within c or if the quadtree level difference around c is more than one. The spatial decomposition is achieved by splitting any crowded cell recursively until there is no crowded cell, and 2-refinement templates are used to remove all hanging nodes, see Fig. 3(b).

Narrow region is an important feature of the input curve. A region is defined as *narrow* if it contains only none/one/two quadtree cells in one direction after the domain decomposition. The detected narrow region will be refined until all directions contain more than two quadtree cells, which guarantees the topology will be preserved during the mesh generation.

4.3 Buffer Zone Clearance

After generating the adaptive quadtree, we remove elements near the boundary curves so that there is enough space to construct guaranteed-quality quad elements. Such a process is called *buffer zone clearance*. Here are definitions used in the algorithm description:

Buffer zone: *Any zonal area that serves the purpose of keeping the quad mesh distant from boundaries, e.g., the blue region in Fig. 3(c).*
Boundary edge: *A boundary edge is contained in only one element, e.g., the edges* $\overline{AB}$, $\overline{BC}$ *and* $\overline{CD}$ *in Fig. 4(a).*
Boundary point: *The two vertices of each boundary edge are named boundary points, e.g., the points A, B, C and D in Fig. 4(a).*
Boundary angle: *The angle formed by two neighboring boundary edges facing the nearest boundary, e.g., the angles* α, β, *and* γ *in Fig. 4(a).*
Boundary edge angle: *The angle formed from the boundary edge to the boundary. The angle must be inside the buffer zone, and can not intersect with the quadtree core mesh. The boundary edge angle has a range of* $[-180°, 180°]$, *and the counterclockwise direction is positive. For example, the angle* ψ *in Fig. 4(a) is negative.*



(a) (b) (c)

Fig. 4. Definition and criteria in Buffer Zone Clearance. (a) The red curve is the boundary. A, B, C, and D are boundary points. $\overline{AB}$, $\overline{BC}$, and $\overline{CD}$ are boundary edges. α, β, and γ are boundary angles. ψ is a boundary edge angle; (b) One exception of Criterion 3; and (c) Single element removal (the blue quad).

Three criteria are used in the buffer zone clearance: Criterion 1: all elements outside the regions to be meshed are deleted; Criterion 2: all elements intersecting with curves $\mathcal{C}$ are deleted; and Criterion 3: if the distance of any vertex to the nearest boundary is less than or equal to a pre-defined threshold ε_s, all elements sharing this vertex are deleted. Here we choose $\varepsilon_s = 0.5 * max(s_i)$, where s_i is the size of the ith element sharing this vertex. However, there are two exceptions. For example in Fig. 4(b), the two blue boundary points are very close to the boundary, so they need to be deleted according to Criterion 3. However, we choose to keep them because the boundary edge formed by them are almost parallel to the boundary. In addition, we also need to remove all the single elements. *A single element* is an element whose four vertices are all boundary points, and it has only one edge shared by another element as shown in Fig. 4(c). After the buffer zone clearance, the quadtree core mesh and a buffer zone are created, see Fig. 3(c). The buffer zone will be filled with guaranteed-quality quad elements.

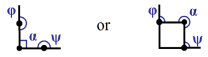
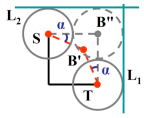
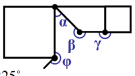
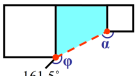
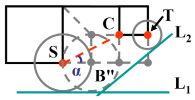
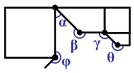
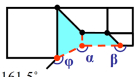
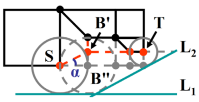
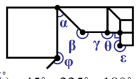
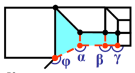
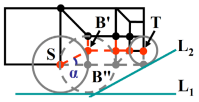
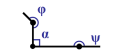
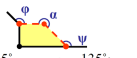
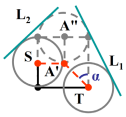
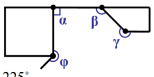
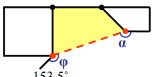
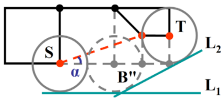
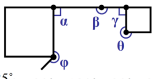
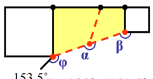
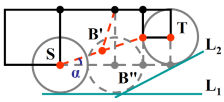
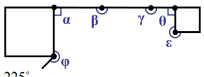
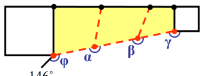
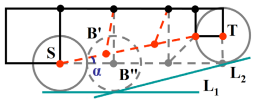
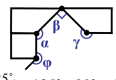
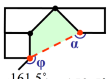
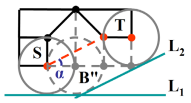
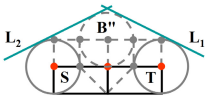
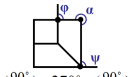
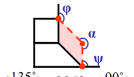
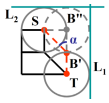
4.4 Template Implementation

If all the boundary edges are parallel to the boundary, it is easy to construct good quality meshes. However, after the buffer zone clearance, the range of the boundary angles is $[45°, 315°]$, and the boundary edge angles are within $[-180°, 180°]$. In order to generate good quality elements around the boundary, we design templates and use them to improve the boundary angles and the boundary edge angles, keeping all the element angles in the quadtree core mesh $\in [45°, 135°]$.

Uniform Grid: If the quadtree core mesh is uniform, there are only three possible boundary angles after the buffer zone clearance: 90°, 180°, and 270°. Only 90° and 270° will possibly introduce bad elements when we fill the buffer zone. To guarantee the element quality, we use Templates 1(a-b) to modify these quad elements. In Tab. 1, the left column shows the original configurations, and the middle column shows the modified templates. For each template, the boundary angles are denoted in the figure and the boundary angle sequence is provided. For example, in Template 1(a), the boundary angle sequence is $\binom{180°}{270°}$-90°-$\binom{180°}{270°}$, which indicates $\varphi = \binom{180°}{270°}$, $\alpha = 90°$, $\psi = \binom{180°}{270°}$, where "$\binom{180°}{270°}$" means "$[180°, 270°]$". By using Templates 1(a-b), the possible boundary angles in the uniform mesh become 112.5°, 180°, 202.5°, and 225°. It is obvious that the boundary angle range is improved from [90°, 270°] to [112.5°, 225°].

The right column of Tab. 1 shows the boundary edge angle for each template. For example in Templates 1(a-b), the grey points and dash lines indicate the edges deleted during the buffer zone clearance. We draw two solid-line circles at the starting point S and the ending point T, and draw a dash-line circle at one of the deleted points, denoted as B''. The solid-line circle means

Table 1. Template implementation of Categories 1-4.

Template	Original template	Modification	Boundary edge angles
1(a-b) Uniform: Adaptive:	 or $(^{180^\circ}_{270^\circ})$-90°-$(^{180^\circ}_{270^\circ})$ or $(^{90^\circ}_{180^\circ})$-270°-$(^{90^\circ}_{180^\circ})$ $(^{180^\circ}_{315^\circ})$-90°-$(^{180^\circ}_{315^\circ})$ or $(^{90^\circ}_{225^\circ})$-270°-$(^{90^\circ}_{225^\circ})$	$(^{112.5^\circ}_{202.5^\circ})$ - 225° - $(^{112.5^\circ}_{202.5^\circ})$ $(^{112.5^\circ}_{247.5^\circ})$ - 225° - $(^{112.5^\circ}_{247.5^\circ})$	
2(a)	 $(^{225^\circ}_{315^\circ})$ - 45° - 225° - 180°	 $(^{161.5^\circ}_{251.5^\circ})$ - 153.5°	
2(b)	 $(^{225^\circ}_{315^\circ})$ - 45° - 225° - 135° - 225°	 $(^{161.5^\circ}_{251.5^\circ})$ - 153.5° - 180°	
2(c)	 $(^{225^\circ}_{315^\circ})$ - 45° - 225° - 180° - 90°	 $(^{161.5^\circ}_{251.5^\circ})$ - 153.5° - 180° - 180°	
3(a)	 $(^{225^\circ}_{315^\circ})$ - 90° - $(^{225^\circ}_{315^\circ})$	 $(^{135^\circ}_{225^\circ})$ - 225° - $(^{135^\circ}_{225^\circ})$	
3(b)	 $(^{225^\circ}_{315^\circ})$ - 90° - 135° - 225°	 $(^{153.5^\circ}_{243.5^\circ})$ - 161.5°	
3(c)	 $(^{225^\circ}_{315^\circ})$ - 90° - 180° - 90° - 270°	 $(^{153.5^\circ}_{243.5^\circ})$ - 180° - 161.5°	
3(d)	 $(^{225^\circ}_{315^\circ})$-90°-180°-180°-90°-270°	 $(^{146^\circ}_{236^\circ})$ - 180° - 180° - 169°	
4(a)	 $(^{225^\circ}_{270^\circ})$ - 135° - 90° - 225°	 $(^{161.5^\circ}_{206.5^\circ})$ - 153.5°	
4(b)	225° - 90° - 225°	180°	
4(c)	 $(^{90^\circ}_{180^\circ})$ - 270° - $(^{90^\circ}_{180^\circ})$	 $(^{135^\circ}_{225^\circ})$ - 225° - $(^{90^\circ}_{180^\circ})$	

Note: The boundary angle sequence below each template starts from left to right. In 1(a), $(^{180^\circ}_{270^\circ})$ -90°-$(^{180^\circ}_{270^\circ})$ indicates $\varphi = (^{180^\circ}_{270^\circ})$, α=90°, $\psi = (^{180^\circ}_{270^\circ})$. "$\varphi = (^{180^\circ}_{270^\circ})$" means "$\varphi \in [180^\circ, 270^\circ]$".

the boundary line must be on or outside the circle since the center is an existing point, while the dash-line circle means the boundary line must be inside the circle since the center is a deleted point. The radius of each circle is half size of the maximum element surrounding its center. By balancing the three circles, we can determine two boundary lines L_1 and L_2, which give the range of possible boundary edge angles for each template. In Templates 1(a-b), the slope of L_1 is $-90°$, and the slope of L_2 is $0°$. Since the boundary edge $\overline{SB'}$ has a slope of $22.5°$ and the boundary edge $\overline{B'T}$ has a slope of $67.5°$, the boundary edge angle becomes $[-67.5°, 67.5°]$. As a final result, we obtain that the range of the boundary edge angles in Templates 1(a-b) is $[-67.5°, 67.5°]$.

Adaptive Grid: For the adaptive mesh, the range of boundary angles after the buffer zone clearance is [45°, 315°]. Here Templates 1(a-b) are also used, but the boundary angles φ and ψ have larger ranges. In addition, three other categories of templates are designed to improve the boundary angles as listed in Tab. 1: the boundary angle $\alpha = 45°$ in Category 2, $\alpha = 90°$ in Category 3, and $\alpha = \{135°, 225°, 270°, 315°\}$ in Category 4. All the other possible configurations can be derived from these basic templates. Meanwhile, templates beginning with $\alpha = 180°$ are not listed in Tab. 1, because they are either categorized in other templates or good enough and therefore need not to be improved.

Here we use Template 2(a) as an example to explain how to derive these templates based on the 2-refinement algorithm (Fig. 1(a-c)). Fig. 5(a) shows a row of three uniform cells, where the black points indicate the boundary nodes. We then use 2-refinement templates to refine these cells, and Fig. 5(b) is one possible result. Here we always keep the left element and the corresponding boundary points in Fig. 5(b). The red points are boundary points which are very close to the boundary. These points and their neighboring cells (grey cells) are removed during the buffer zone clearance, which results in Fig. 5(c). The cells now have new boundary points and boundary edges. However, the boundary edges $\overline{AB}$, $\overline{BC}$ and $\overline{CD}$ form a sharply concave geometry with the minimum boundary angle of 45°. To guarantee the mesh quality, we must eliminate the concaveness by modifying the boundary edges as shown in Fig. 5(d), and finally we obtain Template 2(a) in Tab. 1. Changing the number of cells in Fig. 5(a) or/and applying 2-refinement templates to these cells differently, we can obtain the rest templates in Tab. 1 in similar ways. It is noticed that the same template can be derived from different combinations, and some combinations are impossible and never appear in the mesh. Moreover, the number of cells in Fig. 5(a) can be restricted to be between 1 and 4. If the number of cells is greater than 4, they can always be split so that each part contains no more than 4 cells.

From the right column of Tab. 1, we can obtain the range of the boundary edge angles for each template. We also take Template 2(a) as an example. L_1 is horizontal and its slope is $0°$. Since the dash-line circle at the center B'' is

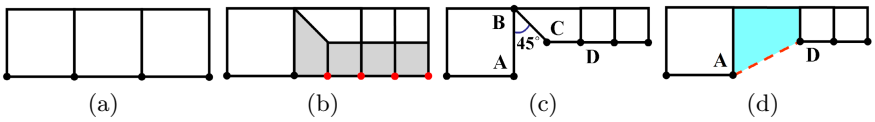


Fig. 5. An example of deriving a template. (a) Given uniform cells; (b) Cells after applying 2-refinement templates; (c) Cells after the buffer zone clearance; and (d) Cells after the template implementation.

double size of the solid-line circle at the center T, and L_2 is tangent to both circles, we can obtain the slope of L_2 is 39.5°. The boundary edge angles of the two horizontal boundary edges are within $[-39.5°, 0°]$. However, the boundary edge $\overline{SC}$ has a slope of 26.5°, so the boundary edge angle becomes $[-13°, 26.5°]$. As a final result, the range of the boundary edge angles in Template 2(a) is $[-39.5°, 26.5°]$.

Lemma 2. *After the template implementation, all the element angles in the quadtree core mesh are within* $[45°, 135°]$*, all the boundary angles are within* $[112.5°, 251.5°]$*, and all the boundary edge angles are within* $[-90°, 90°]$.

Proof. It is obvious that all the element angles in the quadtree core mesh are within $[45°, 135°]$. From the modified templates in Tab. 1, we can see that the boundary angles are within $[112.5°, 251.5°]$, noticing that the possible boundary angle 90° in Template 4(c) will be further improved by Template 1(a)/3(a). We check the boundary edge angles of each template in Tab. 1 and list them in Tab. 2. From the table, it is obvious that all the boundary edge angles are in the range of $[-67.5°, 90°]$. Due to the symmetry of these templates, we release the range of the boundary edge angle to $[-90°, 90°]$. ⋄

Discussion: During the template implementation, the boundary angles and the boundary edge angles are improved by filling with new quads, changing the shape of quads or changing the connectivity of quads. All these improvements will help to guarantee the element quality during the following first and second buffer layer construction. Moreover, we classify all the templates into three priorities as shown in Tab. 2: high, medium and low. Templates in lower priority will not be implemented until all the higher priority templates are implemented.

4.5 First Buffer Layer Construction

For each boundary point, we use angular bisectors to calculate a corresponding point inside the buffer zone called *the first buffer point*. Then, each pair of neighboring boundary points and their first buffer points construct a quad. All these new quads form the first buffer layer as presented in Fig. 3(d).

Here we develop an algorithm to calculate the first buffer points using angular bisectors. As shown in Fig. 6(a), the boundary point B is shared by two boundary edges $\overline{AB}$ and $\overline{BC}$ with the boundary edge angles of φ and ψ.

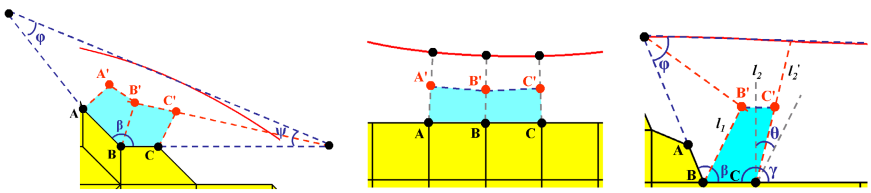
Table 2. Boundary edge angles of each template in Tab. 1.

Template	Slope of L_1	Slope of L_2	α	Boundary edge angle	Priority
1(a)	$-90°$	$0°$	$22.5°$	$[-67.5°, 67.5°]$	Low
1(b)	$-90°$	$0°$	$22.5°$	$[-67.5°, 67.5°]$	Medium
2(a)	$0°$	$39.5°$	$26.5°$	$[-39.5°, 26.5°]$	High
2(b)	$0°$	$28.1°$	$26.5°$	$[-28.1°, 26.5°]$	High
2(c)	$0°$	$28.1°$	$26.5°$	$[-28.1°, 26.5°]$	High
3(a)	$-63.4°$	$45°$	$45°$	$[-63.4°, 90°]$	Low
3(b)	$0°$	$26.5°$	$18.5°$	$[-26.5°, 18.5°]$	High
3(c)	$0°$	$26.5°$	$18.5°$	$[-26.5°, 18.5°]$	High
3(d)	$0°$	$14°$	$11.3°$	$[-14°, 11.3°]$	High
4(a)	$0°$	$26.5°$	$26.5°$	$[-26.5°, 26.5°]$	High
4(b)	$-26.5°$	$26.5°$	-	$[-26.5°, 26.5°]$	High
4(c)	$-90°$	$0°$	$45°$	$[-45°, 90°]$	Medium

Suppose $\psi \geq \varphi$, we draw the angular bisectors of the larger angle ψ and the boundary angle β. Finally, the two angular bisectors intersect at $B^{'}$ which is the corresponding first buffer point of B. By using all the boundary points and their first buffer points, a set of quad elements are generated to form the first buffer layer.

There are two special cases for the first buffer point calculation. If both boundary edges of a boundary point are approximately parallel to the boundary, or the boundary edge angles are less than 10°, we create the first buffer point as the middle of each boundary point and its projection on the boundary. For example in Fig. 6(b), since the two boundary edges $\overline{AB}$ and $\overline{BC}$ are almost parallel to the boundary, we project B to the boundary, and take the middle point of B and its projection as the corresponding first buffer point $B^{'}$. Similarly, $A^{'}$ and $C^{'}$ are calculated. These boundary points A, B, C and the first buffer points $A^{'}$, $B^{'}$, $C^{'}$ construct two quads for the first buffer layer.

As shown in Fig. 6(c), the second special case is introduced by Templates 1(a-b) in Tab. 1. The boundary angles at boundary points B and C are $\alpha = 112.5°$ and $\beta = 180°$, and the angular bisectors are l_1 and l_2. $B^{'}$ is the first buffer point of B. It can be observed that $B^{'}$ is close to l_2, which may



(a) (b) (c)

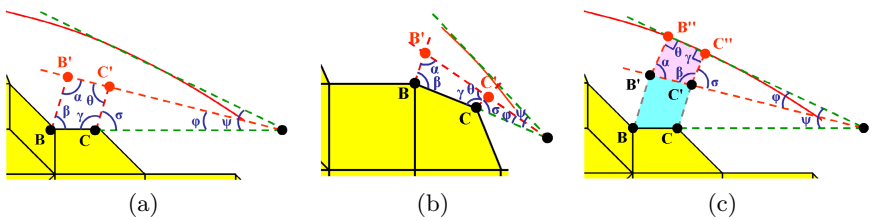
Fig. 6. First buffer layer construction. (a) The general case; (b) The parallel case; and (c) The special case.

introduce a quad with bad angles or even self-intersection. Suppose l_1 and l_2 form an angle $\theta = (\beta - \alpha)/2 = 33.75°$, l_2 is rotated by $\theta/2$ so that it is away from $B^{'}$. This new angular bisector $l_2^{'}$ will be used to calculate $C^{'}$.

With these three cases in Fig. 6, we can get all the corresponding first buffer points and generate the first buffer layer. After the construction of the first buffer layer, the original boundary points have now become inner points, and new boundary points are on the first buffer layer. The following Lemma 3 proves the angle bounds of the first buffer layer.

Lemma 3. *All the four angles of any element in the first buffer layer are in the range of* $[45° \pm \varepsilon, 135° \pm \varepsilon]$*, where* $\varepsilon \leq 5°$.

Proof. To simplify the proof, we first assume that the boundary curve around each boundary edge is a straight line. In Fig. 7(a), $B^{'}$ and $C^{'}$ are the first buffer points corresponding to the boundary points B and C, and the boundary edge angle of $\overline{BC}$ is ψ. Suppose $B^{'}$ and $C^{'}$ are on the angular bisector of ψ. There are two possible cases we need to consider in order to prove this lemma.



(a) (b) (c)

Fig. 7. Guaranteed-quality proof. (a) The first buffer layer (Case 1); (b) The first buffer layer (Case 2); and (c) The second buffer layer.

The first case is shown in Fig. 7(a). The boundary angle of B is $\leq 180°$ and the boundary angle of C is $\geq 180°$. From Lemma 2, we can obtain $\psi \in [-90°, 90°]$. Here we choose $\psi \in [0°, 90°]$ due to the symmetry, therefore $\varphi = \psi/2 \in [0°, 45°]$. After the template implementation, all the boundary angles are in the range of $[112.5°, 251.5°]$. Therefore, the boundary angle of B is in the range of $[112.5°, 180°]$ and the boundary angle of C is in the range of $[180°, 251.5°]$. Since $\overline{BB'}$ and $\overline{CC'}$ are angular bisectors, $\beta \in [56.25°, 90°]$ and $\gamma \in [90°, 125.75°]$. Likewise, we can get $\sigma = 180° - \gamma \in [54.25°, 90°]$, $\theta = \varphi + \sigma \in [54.25°, 135°]$, $\alpha = 180° - \beta - \varphi \in [45°, 125.75°]$. Therefore, all the angles of the quad $BCC^{'}B^{'}$ are within $[45°, 135°]$.

As shown in Fig. 7(b), the second case is that both of the boundary angles are greater than 180°. This case is introduced by Templates 1(a-b). From Tabs. 1-2, we can obtain the two boundary angles are 202.5° at B and 225° at C, and the boundary edge angle $\psi \in [-67.5°, 67.5°]$. Here we choose $\psi \in [0°, 67.5°]$ due to the symmetry. Therefore, we have $\beta = 202.5°/2 = 101.2°$, $\gamma = 225°/2 = 112.5°$, $\sigma = 180° - \gamma = 67.5°$, $\varphi = \psi/2 \in [0°, 33.8°]$,

$\theta = \sigma + \varphi \in [67.5°, 101.3°]$, and $\alpha = 180° - \beta - \varphi \in [45°, 78.8°]$. It is obvious that all the four angles of the quad $BCC^{'}B^{'}$ are within [45°, 112.5°].

Following the similar ways, we can easily prove that all the angles in the two special cases (Fig. 6(b-c)) are also within $[45°, 135°]$. In summary, all the four angles of any element in the first buffer layer are within $[45°, 135°]$. Considering that the boundary curve has a small perturbation $\varepsilon \leq 5°$, we relax the angle range to $[45° \pm \varepsilon, 135° \pm \varepsilon]$. ◇

4.6 Second Buffer Layer Construction

The second buffer layer construction is simple. As shown in Fig. 7(c), we project the first buffer points to the boundary and obtain the corresponding second buffer points. Then we construct a quad using these first and second buffer points, and all these new quads form the second buffer layer, see Fig. 3(f). We use Lemma 4 to prove its angle bounds.

Lemma 4. *All the four angles of any element in the second buffer layer are in the range of* $[45° \pm \varepsilon, 135° \pm \varepsilon]$*, where* $\varepsilon \leq 5°$*.*

Proof. Similar to Lemma 3, we assume that the boundary curve is a straight line around each boundary edge. In Fig. 7(c), $B^{'}$ and $C^{'}$ are the first buffer points of the boundary points B and C. $B^{'}$ and $C^{'}$ are on the angular bisector of ψ, which is the boundary edge angle of $\overline{BC}$. From Lemma 2, we have $\psi \in [-90°, 90°]$. Here we choose $\psi \in [0°, 90°]$ due to the symmetry, therefore $\varphi = \psi/2 \in [0°, 45°]$. It is obvious that $\gamma = \theta = 90°$. We can derive that $\alpha = \sigma = 90° - \varphi \in [45°, 90°]$, and $\beta = 180° - \sigma \in [90°, 135°]$. Therefore, all the four angles of the quad $B^{'}C^{'}C^{''}B^{''}$ are within $[45°, 135°]$. Considering that the boundary curve may have a small perturbation $\epsilon \leq 5°$, the angular range is relaxed to $[45° \pm \varepsilon, 135° \pm \varepsilon]$. ◇

Remark: After applying the designed six steps in Fig. 3, the element angles in the quadtree core mesh, the first buffer layer and the second buffer layer are all in the range of $[45° \pm \varepsilon, 135° \pm \varepsilon]$ ($\varepsilon \leq 5°$).

5 Sharp Feature and Boundary Layer

To preserve sharp features, we keep the quadtree core mesh nearby each sharp feature uniform during the adaptive quadtree construction. An additional template in Fig. 8(a) is implemented, and the boundary angle range is improved from $[90°, 270°]$ to $[116.6°, 206.6°]$. Suppose P is the corner and α is the sharp angle as shown in Fig. 8(b-d), we develop different meshing algorithms for three various cases: $\alpha \in (0°, 135°]$, $(135°, 270°]$, and $(270°, 360°)$.

When $\alpha \in (0°, 135°]$ as shown in Fig. 8(b), we first draw two lines $l_1^{'}$ and $l_2^{'}$, where $l_1^{'} \perp l_1$ and $l_2^{'} \perp l_2$. In the green area enclosed by $l_1^{'}$ and $l_2^{'}$, we find the closest point M after constructing the first buffer layer, then project it to the boundary and obtain two corresponding points L and N. The four

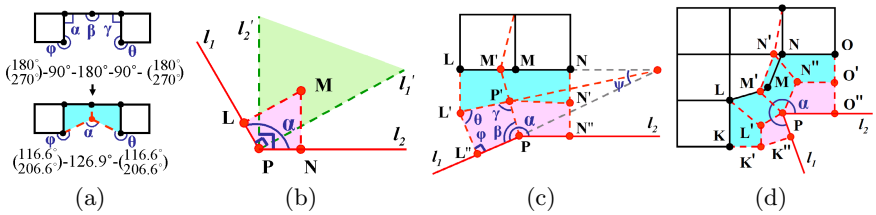


Fig. 8. A new template and three algorithms for sharp features. (a) An additional template; (b) $\alpha \in (0°, 135°]$; (c) $\alpha \in (135°, 270°)$; and (d) $\alpha \in [270°, 360°)$.

points then form a quad $PNML$. Since L and N are the projections of M, if $\alpha \geq 45°$, obviously all the four angles will be within $[45°, 135°]$; otherwise if $\alpha < 45°$, we just keep α, and the range of all the four angles is $[\alpha, 180° - \alpha]$.

When $\alpha \in (135°, 270°]$ as shown in Fig. 8(c), after the template implementation, we use the angular bisector of α to modify the quadtree core mesh close to the sharp feature. First we find that the angular bisector intersects the boundary edge $\overline{LM}$ at $M^{'}$, so we move M to $M^{'}$. Then we create the first buffer points $L^{'}$ and $N^{'}$. Suppose the boundary edge angle of $\overline{LM^{'}}$ is ψ, the first buffer point $P^{'}$ of $M^{'}$ is calculated as the intersection of the angular bisector of ψ with $\overline{PM^{'}}$. The constructed yellow quads form the first buffer layer, its quality is guaranteed by Lemma 3. Finally, we obtain the second buffer layer by projecting $L^{'}$ and $N^{'}$ to the boundary. To prove the second buffer layer is also in good quality, we take the quad $PP^{'}L^{'}L^{''}$ as an example and check all its four angles. It is obvious that $\psi \in [0°, 90°]$ and $\varphi = 90°$. If $\alpha \in [180°, 270°]$, it is straightforward to obtain $\beta = \alpha/2 \in [90°, 135°]$, $\theta = 90° - \psi/2 \in [45°, 90°]$, and $\gamma = 180° - \beta + \psi/2 \in [45°, 135°]$. If $\alpha \in (135°, 180°)$, ψ can be restricted to be within $[0°, 67.5° - (180° - \alpha)]$ from Templates 1(a-b), then we have $\beta = \alpha/2 \in (67.5°, 90°)$, $\theta = 90° - \psi/2 \in [45°, 90°]$, and $\gamma = 180° - \beta + \psi/2 \in [90°, 123.25°]$. Now all the four angles of the quad $PP^{'}L^{'}L^{''}$ are within $[45°, 135°]$. Similarly, we can prove that the quad $P^{'}PN^{''}N^{'}$ is of good quality.

When $\alpha \in (270°, 360°)$ as shown in Fig. 8(d), we also use the angular bisector of α to modify the local quadtree core mesh. $M^{'}$ is the intersection point of the angular bisector of α with $\overline{LM}$. Then we move M to $M^{'}$, and move N to $N^{'}$ so that $M^{'}N^{'}//MN$. The angular bisector of $\angle M^{'}PO^{''}$ intersects the angular bisector of $\angle ON^{'}M^{'}$ at point $N^{''}$; likewise, we create $L^{'}$ in the similar way. Later, we use points $K^{'}$, $L^{'}$, P, $N^{''}$ and $O^{'}$ to construct the first buffer layer, and use $K^{''}$, $O^{''}$ to obtain the second buffer layer. It is noticed that no second buffer layer is created at the sharp feature. This method may not always guarantee good angles, therefore in some cases we have to rotate the bisectors $\overline{PM^{'}}$, $\overline{PN^{''}}$ and $\overline{PL^{'}}$ so that all the angles near the sharp feature are of good quality.

In order to apply a mesh to Computer Fluid Dynamics (CFD) simulations, we must generate one or more boundary layers. In our algorithm, the boundary layer construction is conveniently obtained by splitting the elements of

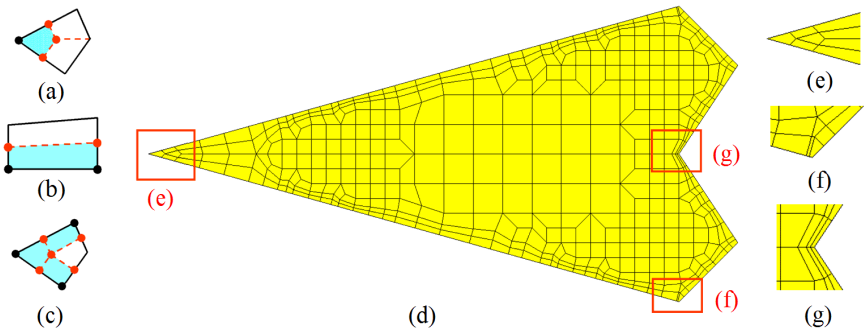


Fig. 9. Templates for boundary layer generation and the delta wing with a small sharp angle of 31°. (a-c) Templates with one, two and three boundary points; (d) Final mesh with all angles $\in [31°, 135°]$; (e-g) Zoom-in pictures of (d).

the second buffer layer. Only three templates are needed as shown in Fig. 9(a-c). The black points are boundary points, the red points are new points, and the blue quads are elements constructed for the boundary layer. It is obvious that Fig. 9(b-c) preserve all the angles. Fig. 9(a) splits an angle using the angular bisector, resulting in a change of the angle range to $[22.5°, 157.5°]$. This only happens at the sharp feature with an angle $> 270°$. Fig. 9(d-g) show the delta wing with 6 sharp angles and a boundary layer.

Remark: Our algorithm guarantees all the angles in the final mesh are within $[45° \pm \varepsilon,\ 135° \pm \varepsilon]$ ($\varepsilon \leq 5°$), with the exception of bad elements that may be required by sharp features with an angle $< 45°$ or $> 270°$.

6 Results

We have applied our algorithm to two complicated models: the Lake Superior map and the air foil with multiple components. Our results were computed on a PC equipped with an Intel Q6600 CPU and 4GB DDR-II Memories. As shown in Fig. 10, the Lake Superior map consists of seven closed smooth curves, which has narrow regions. The constructed mesh conforms to the input curves accurately and all the elements are quads with angles within $[43°, 135°]$. The mesh adaptivity is controlled by a size function based on the boundary curvature and narrow region. It took 82 seconds to generate the mesh. The final mesh has 32789 nodes and 30321 quads, the quadtree has 10 levels, and the maximum aspect ratio is 10.

In Fig. 11, the air foil consists of three components, all of them contain sharp angles. It took 15 seconds to generate the mesh. Before generating a boundary layer, the mesh has 24514 nodes and 22929 quads, the quadtree has 14 levels, the angle range in the final mesh is $[45°, 135°]$, and the maximum aspect ratio is 4.7. After generating one boundary layer, the angle range becomes $[27°, 153°]$ due to the template in Fig. 9(a). This result shows that

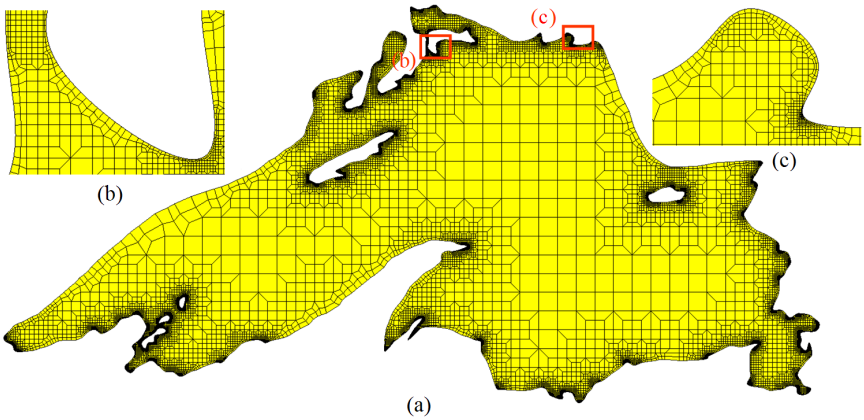


Fig. 10. The Lake Superior map. (a) Final guarantee-quality all-quad mesh with all angles $\in [43°, 135°]$; and (b-c) Zoom-in pictures of (a).

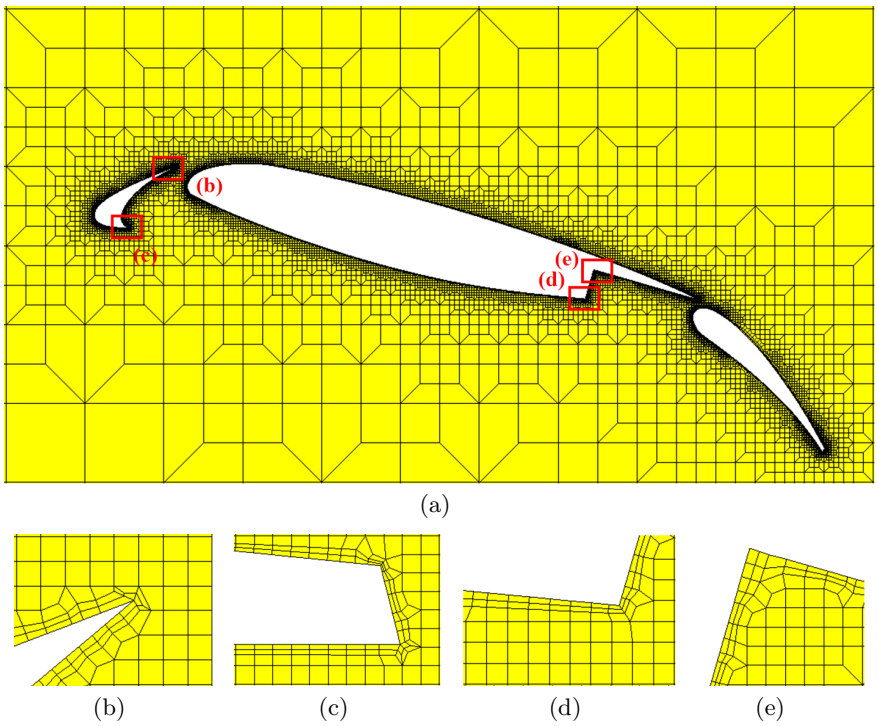


Fig. 11. The air foil with multiple components. (a) All-quad mesh with all angles $\in [45°, 135°]$ before boundary layer generation, and $[27°, 153°]$ after generating one boundary layer; (b-e) Zoom-in pictures of (a).

our algorithm can deal with large-scale inputs while minimizing the number of elements, which are important for aerodynamics simulations.

7 Conclusion and Future Work

In this paper, we present a quadtree-based meshing method, which creates guaranteed-quality all-quad meshes with feature preservation for arbitrary planar domains. It is proved that all the elements of the final mesh are quads with angles between $45° \pm \varepsilon$ and $135° \pm \varepsilon$ ($\varepsilon \leq 5°$), except badly shaped elements required by the specified geometry. Our algorithm can conveniently generate boundary layers on the final mesh for the CFD simulation. We have applied our algorithm to a set of complicated geometries, including the Lake Superior map and the air foil with multiple components.

In the current algorithm, the meshes generated inside and outside the curves do not conform to each other. In the future, we will improve our algorithm so that it can be used to generated interior and exterior meshes at the same time. We are also planning to extend our algorithm to dynamic re-meshing. All the ideas in the paper can be extended to 3D hexahedral meshes. As part of our future work, we would like to explore quality proof for octree-based hexahedral mesh generation.

Acknowledgement

This research was supported in part by ONR grant N00014-08-1-0653, which is gratefully acknowledged.

References

1. Atalay, F.B., Ramaswami, S.: Quadrilateral meshes with bounded minimum angle. In: 17th Int. Meshing Roundtable, pp. 73–91 (2008)
2. Baehmann, P.L., Wittchen, S.L., Shephard, M.S., Grice, K.R., Yerry, M.A.: Robust geometrically based, automatic two-dimensional mesh generation. Int. J. Numer. Meth. Eng. 24, 1043–1078 (1987)
3. Baker, T.J.: Identification and preservation of surface features. In: 13th Int. Meshing Roundtable, pp. 299–309 (2004)
4. Bern, M., Eppstein, D.: Quadrilateral meshing by circle packing. Int. J. Comp. Geom. & Appl. 10(4), 347–360 (2000)
5. Bishop, C.J.: Quadrilateral meshes with no small angles (Manuscript) (1991)
6. Blacker, T.D., Stephenson, M.B.: Paving: A new approach to automated quadrilateral mesh generation. Int. J. Numer. Meth. Eng. 32, 811–847 (1991)
7. Canann, S.A., Tristano, J.R., Staten, M.L.: An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes . In: 7th Int. Meshing Roundtable, pp. 211–224 (1998)
8. Freitag, L., Jones, M., Plassmann, P.: An efficient parallel algorithm for mesh smoothing. In: 4th Int. Meshing Roundtable, pp. 47–58 (1995)

9. Freitag, L.A.: On combining Laplacian and optimization-based mesh smoothing techniques. Trends in Unstructured Mesh Generation, ASME 220, 37–43 (1997)
10. Joe, B.: Quadrilateral mesh generation in polygonal regions. Comput. Aid. Des. 27(3), 209–222 (1991)
11. Kinney, P.: CleanUp: Improving quadrilateral finite element meshes. In: 6th Int. Meshing Roundtable, pp. 437–447 (1997)
12. Mitchell, S.A., Tautges, T.J.: Pillowing doublets: Refining a mesh to ensure that faces share at most one edge. In: 4th Int. Meshing Roundtable, pp. 231–240 (1995)
13. Owen, S.: A survey of unstructured mesh generation technology. In: 7th Int. Meshing Roundtable, pp. 26–28 (1998)
14. Quadros, W.R., Ramaswami, K., Prinz, F.B., Gurumoorthy, B.: LayTracks: A new approach to automated geometry adaptive quadrilateral mesh generaton using medial axis transform. Int. J. Numer. Meth. Eng. 61, 209–237 (2004)
15. Schneiders, R.: Refining quadrilateral and hexahedral element Meshes. In: 5th Int. Meshing Roundtable, pp. 383–398 (1996)
16. Schneiders, R., Schindler, R., Weiler, F.: Octree-based generation of hexahedral element meshes. In: 5th Int. Meshing Roundtable, pp. 205–216 (1996)
17. Staten, M.L., Canann, S.A.: Post refinement element shape improvement for quadrilateral meshes. Trends in Unstructured Mesh Generation, ASME 220, 9–16 (1997)
18. Tam, T., Armstrong, C.G.: 2D finite element mesh generation by medial axis subdivision. Adv. Eng. Software 13, 313–324 (1991)
19. White, D.R., Kinney, P.: Redesign of the paving algorithm: Robustness enhancements through element by element meshing. In: 6th Int. Meshing Roundtable, pp. 323–335 (1997)
20. Yerry, M.A., Shephard, M.S.: A modified quadtree approach to finite element mesh generation. IEEE Computer Graphics Appl. 3(1), 39–46 (1983)
21. Zhang, Y., Bajaj, C.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric Data. Comput. Meth. Appl. Mech. Eng. 195, 942–960 (2006)
22. Zhang, Y., Bajaj, C., Sohn, B.-S.: 3D finite element meshing from imaging data. Comput. Meth. Appl. Mech. Eng. 194, 5083–5106 (2005)
23. Zhang, Y., Bajaj, C., Xu, G.: Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. Commun. Numer. Meth. Eng. 25, 1–18 (2009)
24. Zhang, Y., Hughes, T., Bajaj, C.: An automatic 3D mesh generation method for domains with multiple materials. Comput. Meth. Appl. Mech. Eng. (in press, 2009)
25. Zhu, J.Z., Zienkiewicz, O.C., Hinton, E., Wu, J.: A new approach to the development of automatic quadrilateral mesh generation . Int. J. Numer. Meth. Eng. 32, 849–866 (1991)