Commun. Comput. Phys.
doi: 10.4208/cicp.030612.010313a

Vol. **14**, No. 5, pp. 1304-1321
November 2013

Robust and Quality Boundary Constrained Tetrahedral Mesh Generation

Songhe Song[1,2], Min Wan[3,*], Shengxi Wang[4], Desheng Wang[3] and Zhengping Zou[5]

[1] *Department of Mathematics and Systems Science, State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, P.R. China.*
[2] *State Key Laboratory of Aerodynamics, China Aerodynamics and Development Center, Mianyang 621000, P.R. China.*
[3] *Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371.*
[4] *Taiyuan Satellite Launch Center, 030027, P.R. China.*
[5] *National Key Laboratory of Science and Technology on Aero-Engine Aero-Thermodynamics, School of Jet Propulsion, Beijing University of Aeronautics and Astronautics, Beijing 100191, P.R. China.*

Received 3 June 2012; Accepted (in revised version) 1 March 2013

Available online 13 June 2013

Abstract. A novel method for boundary constrained tetrahedral mesh generation is proposed based on Advancing Front Technique (AFT) and conforming Delaunay triangulation. Given a triangulated surface mesh, AFT is firstly applied to mesh several layers of elements adjacent to the boundary. The rest of the domain is then meshed by the conforming Delaunay triangulation. The non-conformal interface between two parts of meshes are adjusted. Mesh refinement and mesh optimization are then preformed to obtain a more reasonable-sized mesh with better quality. Robustness and quality of the proposed method is shown. Convergence proof of each stage as well as the whole algorithm is provided. Various numerical examples are included as well as the quality of the meshes.

AMS subject classifications: 65L50, 65M50, 65N50, 65D18

Key words: Unstructured tetrahedral mesh, Advancing Front Technique, conforming Delaunay triangulation, boundary constrained, mesh refinement, mesh optimization.

*Corresponding author. *Email addresses:* shsong@nudt.edu.cn (S. Song), wanm0003@e.ntu.edu.sg (M. Wan), wsx.nudt@gmail.com (S. Wang), desheng@ntu.edu.sg (D. Wang), zouzhengping@buaa.edu.cn (Z. Zou)

1 Introduction

Automatic tetrahedral mesh generation is always an important yet challenging part in many scientific and engineering computation problems. As the preprocessor of computation or analysis, it plays a significant role in discretizing the domain of interest. Extensive research has been conducted in tetrahedral meshing field and tremendous advances have been made on both theoretical analysis and robust implementation. Most existing tetrahedral meshing techniques can be categorized into three groups: 1. Spatial decomposition methods, i.e octree-based methods [1]; 2. Advancing front triangulation methods [2,3]; 3. Delaunay triangulation based methods [4–8].

The input for most mesh generators is a surface triangulation, which bounds the domain of interest. The desirable generated mesh shall conform with the input triangulation in such a way that the geometric constraints in the triangulation, i.e. edges and faces, are preserved in the generated volumetric mesh. Usually there are two manners to preserve the constraints. The first one is *boundary conforming way*, in which the constraints can be kept as the concatenation of edges/faces. This manner could possibly introduce extra points on the constraints. The second manner is *boundary constrained way*, in which no extra point on the constraints is allowed in the result mesh. It is required that all constraints are preserved in their original and integral forms. Though both manners are acceptable in most computational fields, a mesh constrained to boundary is often preferred due to two reasons. (a) Boundary constrained way respects the input sizing specification and no extra point on constraints modifies the sizing function; (b) it offers more robustness than the conforming way when meshes from two connected regions are merged together. A efficiency and robust boundary constrained mesh generator is also our goal. Next we simply describe how three groups of methods obtain a boundary constrained mesh as well as their advantages and disadvantages.

1. In spatial decomposition methods, a series of recursively variable sized cubes described by an octree is used to approximate the domain of interest. The input for most octree methods is the geometric representation without boundary constraints. The boundary triangulation is generated by intersecting the leaf cubes of octree with the geometric boundary. It is difficult for octree methods to generate boundary constrained meshes.

2. In advancing front methods, mesh is generated element by element along the boundary of existing mesh, i.e. *front*. The triangle on the *front* is taken as base and an apex is selected or created to construct a tetrahedron, the rule of which includes that the new tetrahedron shall not destroy the existing mesh. This construction rule naturally guarantees that the input surface triangulation would not be destroyed. The generated mesh will be constrained to boundary inherently. Besides, advancing front provides good qualities for the elements near the boundary, which are favored by many applications, such as computational fluids. However, the efficiency and the robustness of advancing front is always questionable in the community. It

is time-consuming, and the heuristics more or less required sometimes cause failure especially at the end of the algorithm.

3. Delaunay triangulation has been proved an efficient and robust mesh generator. However, Delaunay-based methods usually provides an initial triangulation of the convex hull of all input data points, which often does not conform the input surface triangulation. The geometric constraints need to be recovered from the initial convex hull. Constrained boundary recovery is referred to this procedure. Many constrained boundary recovery methods have been proposed. Some lack theoretic guarantee [9, 10]. While some [11] do not pay enough attention to the qualities of elements near the boundary, which are one important concern of the community.

The need of an efficient and robust boundary constrained mesh generator with good qualities of elements near the boundary, motivates this study. In this article, a new method is proposed to generate boundary constrained tetrahedral mesh. Advancing front technique and Delaunay triangulation are applied sequentially in different meshing stages. In the proposed method, the problem of boundary constrained issue is circumvented by the advancing front and translated to the conforming boundary recovery which can be well solved by the Delaunay-based methods. Theoretical proofs are also included.

The algorithm presented mainly consists of three stages. In the first stage, the advancing front technique is applied to the input surface triangulation $\mathcal{S}$. This stage is terminated after several layers of elements are generated and $\mathcal{S}$ is not visible to the interior any longer. Through this stage of operations, a new front (surface triangulation) $\mathcal{S}_1$ is obtained as well as several layers of tetrahedral mesh $\mathcal{T}_1$. The operation of advancing front naturally guarantees that $\mathcal{S}$ is preserved constrainedly in $\mathcal{T}_1$. In the second stage, the Delaunay-based method in [12] is applied on $\mathcal{S}_1$. A modified Delaunay refinement (edge/face splits) are used to recover the missing constraints. This stage provides $\mathcal{T}_2$, the mesh of the domain bounded by $\mathcal{S}_1$, and $\mathcal{S}_1$ is recovered in a conforming way. In the third stage, three tasks are to be accomplished, namely coupling meshes from two subdomains, mesh refinement and optimization. Due to the possible edge/face splits operation in the second stage, $\mathcal{T}_1 \cup \mathcal{T}_2$ is not always conformal. Coupling operation aims to turn the existing mesh to a conformal one and is proven not to destroy the constraints on $\mathcal{S}$. Mesh refinement and optimization utilizes some well developed method to obtain a mesh with (a) a mesh size specified by the initial surface triangulation $\mathcal{S}$, (b) a good mesh quality.

The proposed method combines the advantages of both advancing front method and Delaunay based method while compensates the disadvantages of both. *Efficiency:* since advancing front only meshes a few elements near the boundary, the whole method's efficiency is dominated by the Delaunay kernel, which is proved the fastest among three; *Robustness:* the robustness of AFT has been always questioned by the community, which is not case for our method. The constructive proof for the convergence of the AFT stage is given in this study. Combining with the theoretic conclusion from [11], the completeness and convergence of the whole algorithm is guaranteed as well; *Quality:* the interior mesh

is generated via Delaunay triangulation, which is the optimal triangulation under the nearest neighborhood criterion. This implies the good quality. Besides, advancing front tends to generate elements near the boundary with good qualities. As a matter of fact, the quality can be determined by the criteria of choosing the candidate apex.

Furthermore, those near-the-boundary elements, which are of more importance, sometimes are need to be high-aspect ratio, i.e. anisotropic. This demand becomes more and more commonplace in the computational field. By investigating the measures of two main approaches, i.e. AFT and Delaunay-based methods, taking on the anisotropic issue, we found that less modification is required in AFT method. Control matrix representing different directions as well as the normalized edge lengths recalculation is indispensable in both approaches, while Delaunay-based methods require Delaunay insertion kernel modification. What is more tedious, the reevaluated *Cavity* is not always star-shaped in three dimensions, which means verification and subsequent correction is necessary. On the contrary, except the calculation of normalized edge lengths, the other steps in AFT methods are identical to those in isotropic applications.

The remaining part of the paper is organized as follows. Section 2 gives an overview of our method, three stages of which are presented in Sections 3, 4, and 5. Section 3 gives a brief review of the advancing front technique. Section 4 gives a brief review of the Delaunay triangulation with conforming boundary recovery. Section 5 presents the coupling, mesh refinement and optimization procedures. Section 6 includes some numerical examples and Section 7 concludes the article.

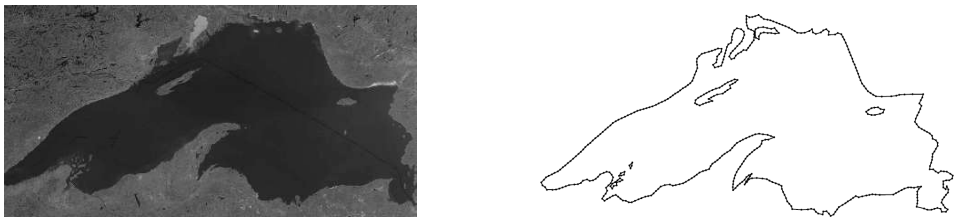
2 An overview of the method

In this section, we briefly describe every stage in the proposed method. Generally speaking, a three dimensional tetrahedral mesh $\mathcal{T}_h$ consists of N tetrahedra $\{K_i\}_{i=1}^N$, the intersection of any two of which is either empty set or a lower dimensional face, i.e. triangle, edge and vertex. By referring to mesh or triangulation, we are mentioning the same thing in this article [9]. A surface triangulation or a triangular mesh $\mathcal{S}_h$ is defined as a complex consisting of triangles, the intersection of any two of which is either empty set or a lower dimensional face, i.e. edge and vertex. A surface triangulation itself should be a surface, i.e. 2-manifold. We restrict this study to watertight surfaces, i.e. 2-manifolds without boundaries.

Let $\mathcal{S}$ be the input surface triangulation bounding the domain Ω. To illustrate all stages in two dimensions, a figure of Lake Superior as well as the PSLG (Planar Straight Line Graph) is shown in Fig. 1(a) and (b). The goal of the algorithm is to obtain a tetrahedral triangulation $\mathcal{T}$ of Ω with constrained boundary recovery, i.e. each element of $\mathcal{S}$ exists in $\mathcal{T}$. The algorithm mainly includes three stages as follows.

1. Advancing front method

   The advancing front technique is first applied. Mesh elements are progressively constructed along the current *front*. One single layer of elements as well as the new *front* is shown in Fig. 2(a) and

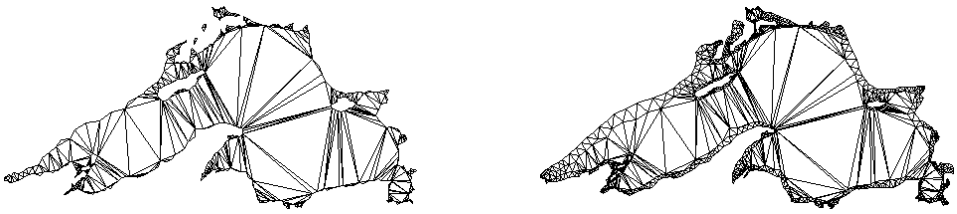


(a) Satellite view (b) PSLG of the boundary

Figure 1: Lake Superior as a two dimensional example.

(a) Elements generated (b) Front after advancing

Figure 2: Illustration of the first advancing front step.



(a) Constrained Delaunay triangulation of the advanced front (b) Combination of CDT and AFT elements

Figure 3: Illustration of constrained Delaunay triangulation afterwards.

(b). This stage terminates when several layers of elements $\mathcal{T}_1$ are generated and $\mathcal{S}$ is not visible to the interior bounded by the new *front*, $\mathcal{S}_1$.

2. Conforming Delaunay triangulation

   A modified Delaunay refinement method in [12] is used to mesh the domain bounded by $\mathcal{S}_1$ and recover the constraints on $\mathcal{S}_1$ in a conforming way. Steiner points are inserted on the constraints to split edges and faces. In this stage, $\mathcal{T}_2$ is obtained, whose boundary is $\mathcal{S}_2$ conforming $\mathcal{S}_1$. This stage is shown in Fig. 3(a).

3. Coupling, mesh refinement and optimization

   This stage contains three phases. The first is coupling. $\mathcal{T}_1 \cup \mathcal{T}_2$ is not always conformal due to the split operation in stage 2. Coupling operation transforms the whole mesh to a conformal one by split the elements in $\mathcal{T}_1$ correspondingly. The coupling procedure is shown in Fig. 3(b). The

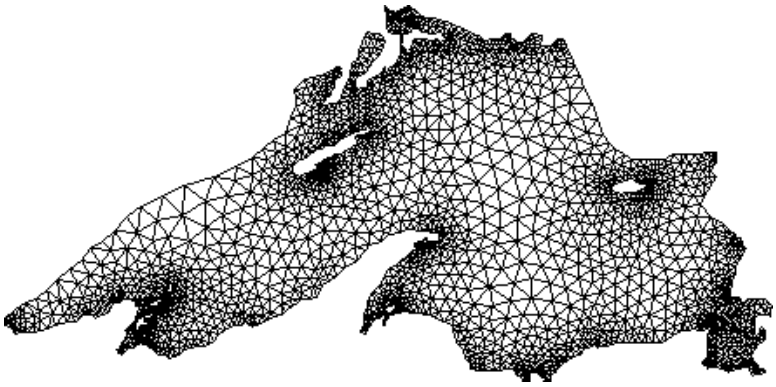


Figure 4: Final mesh after remeshing and optimization.

second is mesh refinement, which generates interior field points according to the sizing function. The third is applying mature mesh optimization technique such as Centroid Voronoi Tessellation to improve the mesh quality further. The final result is shown in Fig. 4.

3 Advancing front triangulation

This section briefly describes the first stage of the proposed method: mesh the region near the input boundary $\mathcal{S}$ and propagate $\mathcal{S}$ to a new surface triangulation $\mathcal{S}_1$ by the advancing front technique. The advancing front triangulation (AFT) have been studied for almost forty years among the mesh community. Since it was first presented in [13], this technique has been improved by many researchers [14–19]. Mature and powerful, this method is widely applied in mesh generating field.

Most advancing front triangulation algorithms can be described as repeatedly performing three steps until certain condition is satisfied: (a) select a triangle from the current *front* as the base of the tetrahedron to be created; (b) build a point candidate list for the potential apex of the tetrahedron; (c) select the best choice from the candidate list to create the tetrahedron and update the *front*.

The manner of building the candidate list and the order to process the triangles on the *front* vary from one implementation to another, while the condition to terminate the algorithm is same, i.e. the *front* is empty or the domain of interest is completely meshed. However, in our method, AFT is required to mesh only a small portion of domain, i.e. the region near the boundary. Imperially two or three layers of elements are enough. Theoretically this stage's aim is to generate a new surface triangulation $\mathcal{S}_1$ such that the initial $\mathcal{S}$ is not visible to the domain bounded by $\mathcal{S}_1$.

To generate such a $\mathcal{S}_1$ could be achieved via two steps. The first step generates a new *front* $\mathcal{S}_a$ such that the faces on $\mathcal{S}$ are not visible to the domain Ω_a bounded by $\mathcal{S}_a$. The second step generates a new *front* $\mathcal{S}_b$ such that the edges on $\mathcal{S}$ are not visible to the domain

Ω_b bounded by $\mathcal{S}_b$. Next we show the existence of $\mathcal{S}_a$ and $\mathcal{S}_b$ as well as the feasibility to find them. First we discuss $\mathcal{S}_a$.

Lemma 3.1. *For the triangle face $\triangle A_iB_iC_i \subset$ front as well as its centroid M_i and unit normal $\overrightarrow{N_i}$ and another triangle $\triangle_j \subset$ front, there exists an ϵ_j such that the tetrahedron $P_jA_iB_iC_i$, where $P_j = M_i + \epsilon_j \overrightarrow{N_i}$, does not intersect $\triangle_j$.*

Proof. Given an arbitrary ϵ and corresponding P_i, suppose R is one vertex of $\triangle_j$.

- If $\overrightarrow{RM} \cdot \overrightarrow{N_i} \leq 0$, it is trivial $R \notin P_iA_iB_iC_i$;
- If $\overrightarrow{RM} \cdot \overrightarrow{N_i} > 0$,
  - either $R \notin P_iA_iB_iC_i$,
  - or $R \in P_iA_iB_iC_i$, then find $\partial R A_iB_iC_i \cap M_iP_i = P_i^{'} = M_i + \epsilon^{'}$, choose $\epsilon^{''} = \frac{1}{2}\epsilon^{'}$ and $P_i^{''} = M_i + \epsilon^{''}$, it could be shown that $R \notin P_i^{''}A_iB_iC_i$.

The above discussion shows that for an vertex R in $\triangle_j$, it is either $R \notin P_iA_iB_iC_i$ or we could update P_i to another $P_i^{''}$ such that $R \notin P_i^{''}A_iB_iC_i$. Similar discussions apply to the other two vertices, which we call S,T. It is also noticed that since $P_i^{''}A_iB_iC_i \subset P_iA_iB_iC_i$, the re-selection of the apex would not re-include the other vertices of $\triangle_j$.

Since we could find a P_j such that $P_jA_iB_iC_i \cap R,S,T = \emptyset$. Due to the fact $\triangle_j$ is a convex set, we have conclusion that $P_jA_iB_iC_i \cap \triangle_j = \emptyset$, namely no intersection. □

Lemma 3.2. *For a triangle face $\triangle A_iB_iC_i \subset$ front as well as its centroid M_i and unit normal $\overrightarrow{N_i}$, there exists an ϵ such that the tetrahedron $A_iB_iC_iP_i$, where $P_i = M_i + \epsilon\overrightarrow{N_i}$, does not intersect with any other elements in the front.*

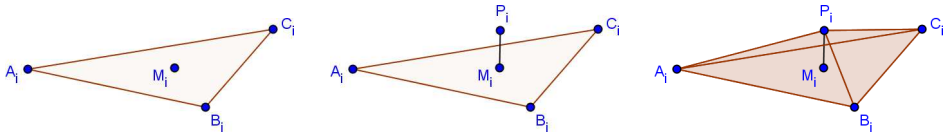
Proof. For each $\triangle_j \subset$ *front* other than $\triangle A_iB_iC_i$, we have shown in Lemma 3.1 that $\exists P_j = M_i + \epsilon_j \cdot \overrightarrow{N_i}$ such that $P_jA_iB_iC_i \cap \triangle_j = \emptyset$.

Choose $\epsilon = \min_{all\,\triangle_j}\{\epsilon_j\}$ and $P_i = M_i + \epsilon \cdot \overrightarrow{N_i}$. Since $P_iA_iB_iC_i \subset P_jA_iB_iC_i$, $P_iA_iB_iC_i \cap \triangle_j = \emptyset$. Then $\cup\{\triangle_j\} \cap P_iA_iB_iC_i = \emptyset$.

Hence there exists an P_i such that $P_iA_iB_iC_i$ does not intersect with any other elements in the *front*. □

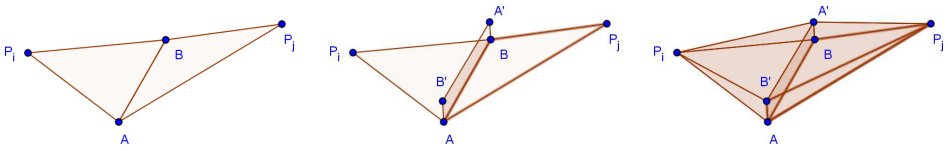
Iteratively, we build such a tetrahedron for each triangle in $\mathcal{S}$ and update the *front*. The initial triangle $\triangle A_iB_iC_i$ in $\mathcal{S}$ is updated to $\triangle A_iB_iP_i$, $\triangle A_iP_iC_i$, and $\triangle P_iB_iC_i$ in $\mathcal{S}_a$. Obviously, the initial triangle $\triangle A_iB_iC_i$ in $\mathcal{S}$ is not visible to the domain Ω_a. The number of triangle faces contained in $\mathcal{S}$ is limited. Hence after limited steps of algorithm, all triangle faces in $\mathcal{S}$ is not visible to Ω_a. One single step of this construction is illustrated in Fig. 5.

Above shows the existence of $\mathcal{S}_a$ as well as the construction way. The existence of $\mathcal{S}_b$ could be shown similarly. Assume the edge AB in $\mathcal{S}$ is of interest. In the $\mathcal{S}_a$, AB is shared



(a) (b) (c)

Figure 5: Illustration of the first part of construction.



(a) (b) (c)

Figure 6: Illustration of the second part of construction.

by triangles $\triangle ABP_i$ and $\triangle ABP_j$ with unit normal $\overrightarrow{N_i}$ and $\overrightarrow{N_j}$. Obviously, P_i and P_j are not in $\mathcal{S}$. The unit normal of AB is defined as the average of $\overrightarrow{N_i}$ and $\overrightarrow{N_j}$: $\overrightarrow{N} = \frac{\overrightarrow{N_i}+\overrightarrow{N_j}}{|\overrightarrow{N_i}+\overrightarrow{N_j}|}$. There exists an ϵ such that $\triangle A'B'P_i$ and $\triangle A'B'P_j$ do not intersect any element on the *front*, where $A' = A+\epsilon\overrightarrow{N}$ and $B' = B+\epsilon\overrightarrow{N}$. The triangles $\triangle ABP_i$ and $\triangle ABP_j$ are substituted by $\triangle A'B'P_i$ and $\triangle A'B'P_j$ in $\mathcal{S}_b$. No edge from $\mathcal{S}$ is visible to Ω_b. This construction is illustrated in Fig. 6.

Through these two kinds of construction, the original $\triangle A_iB_iC_i$ in $\mathcal{S}$ is substituted by $\triangle A_i^{'}B_i^{'}P_i$, $\triangle A_i^{'}P_iC_i^{'}$, and $\triangle P_iB_i^{'}C_i^{'}$ in $\mathcal{S}_b$. The new surface triangulation $\mathcal{S}_b$ consists of $\{A_i^{'},B_i^{'},C_i^{'},\cdots,P_i\}$. The vertices $\{A_i^{'},B_i^{'},C_i^{'},\cdots\}$ on $\mathcal{S}$ do not exist in $\mathcal{S}_b$. $\mathcal{S}$ is not visible to Ω_b. In sum, we can draw the following conclusion.

Theorem 3.1. *Given a watertight surface triangulation $\mathcal{S}$, there exists a new triangulation $\mathcal{S}_1$ such that $\mathcal{S}$ is not visible to the domain bounded by $\mathcal{S}_1$.*

4 Conforming Delaunay triangulation with boundary recovery

In this section, we describe the operation in the second stage of the algorithm: Delaunay-based triangulation on $\mathcal{S}_1$ with conforming boundary recovery. The goal of this stage is to generate $\mathcal{T}_2$ such that each element of $\mathcal{S}_1$ either exists in $\mathcal{T}_2$ or is the union of a set of edges/faces of $\mathcal{T}_2$, and the geometry of the boundary of $\mathcal{T}_2$ conforms with that of $\mathcal{S}_1$.

As mentioned in Section 2, there are several typical methods for conforming boundary recovery, one example of which is the Delaunay refinement method in [7] which is popular in the community. Instead we choose the method in [12] due to its modification on the Delaunay kernel. Such modification eliminates the possible infinite Steiner points insertion in [7]. This method is briefly presented as follows.

Let P be the vertices on $\mathcal{S}_1$. Denote the Delaunay triangulation of P by $\mathcal{T}_d$.

1. First, information from $\mathcal{S}_1$ is collected. All existing constraints in $\mathcal{T}_d$ are marked *Recovered* and all missing ones are marked*Unrecovered*. All *Recovered* constraints are automatically marked *Protected*.
2. Secondly, the Delaunay refinement method that consists of the edges/faces splittings is used. For a missing constraint, the intersection points with $\mathcal{T}_d$ are listed. Intersection points are to be inserted sequentially into $\mathcal{T}_d$ in a modified Delaunay way until this constrain is recovered as a concatenation of several edges/faces, all of which are marked *Protected*.
3. Finally, the elements lying outside the domain are deleted from $\mathcal{T}_d$ and a conforming triangulation $\mathcal{T}_d$ of S is obtained, i.e. the boundary recovery process is completed. For distinguishing convenience, we rename the obtained volumetric mesh $\mathcal{T}_d$ by $\mathcal{T}_2$ and the recovered boundary $\mathcal{S}_2$.

As is known, the classical Delaunay incremental point insertion algorithm is mainly three steps: determine the *Base*, search the *Cavity* and connect the *Ball*. The modified Delaunay insertion way, i.e. the modified Delaunay kernel, has changed the manner of searching the *Cavity*. The searching *Cavity* procedure would be blocked by the possible *Protected* constraints, not only dominated by the empty sphere criteria. This modification could prevent the insertion from destroying the already recovered constraints and eliminate infinite points insertions. Also this modified Delaunay kernel leads the proof of the convergence of this method. Interested readers please see the details and proof in [12]. We only list the conclusion as follows.

Lemma 4.1. *The boundary recovery algorithm presented in Section* 4 *for recovering the missing edges/faces of $\mathcal{S}_1$ is convergent, i.e. the missing constraints can be recovered as a concatenation of edges/faces of $\mathcal{T}_2$ through a finite number of intersection points insertion in the modified Delaunay insertion procedure* [12].

5 Coupling, mesh refinement and optimization

In this section, all three phases involved in the last stage are presented sequentially: coupling operation, mesh refinement and optimization.

5.1 Coupling

Obviously, the mere union of results from the first and second stages, $\mathcal{T}_1 \cup \mathcal{T}_2$, may be non-conformal, which is not favorable in most applications. The coupling operation aims to turn $\mathcal{T}_1 \cup \mathcal{T}_2$ into a conformal mesh without destroying the integrity of $\mathcal{S}$.

The main idea is to split the elements corresponding to the disagreement between $\mathcal{S}_1$ and $\mathcal{S}_2$. If one constraint X in $\mathcal{S}_1$ is detected to be recovered as union of several edges/faces in $\mathcal{S}_2$, the element adjacent to X in $\mathcal{T}_1$ is split correspondingly. After all such possible split operations, $\mathcal{T}_1$ is turned to $\mathcal{T}_1'$. $\mathcal{T}_1' \cup \mathcal{T}_2$ is conformal and constrained to $\mathcal{S}$. The algorithm and proof of convergence is provided below. In the following statement, we denote a tetrahedron with base F and apex v by vF.

There are two operations in the coupling algorithm: detection and split. Detecting operation is performed on triangle faces. For a face F in $\mathcal{S}_1$, if it does not exist in $\mathcal{S}_2$, we can detect the faces $\{F_i\}_{i=1}^M$ in $\mathcal{S}_2$, concatenation of which is F. Once such mapping is established, the inverse map from $\mathcal{S}_2$ to $\mathcal{S}_1$ can be determined as well. From the split operation in Section 4, we can have the following claim.

Claim 5.1. For a triangle face F_2 in $\mathcal{S}_2$, there is one and only one face F_1 in $\mathcal{S}_1$ such that $F_2 \subset F_1$.

If a triangle face F on $\mathcal{S}_1$ is missing in $\mathcal{S}_2$ and is detected that $F = \bigcup_{i=1}^M F_i$, $F_i \in \mathcal{S}_2$, $i = 1,\cdots,M$, the tetrahedron vF in $\mathcal{T}_1$ is split to $\{vF_i\}_{i=1}^M$. As for the tetrahedron vF, we have the following claim.

Claim 5.2. let F be one triangle face on $\mathcal{S}_1$, there is one and only one tetrahedron vF in $\mathcal{T}_1$.

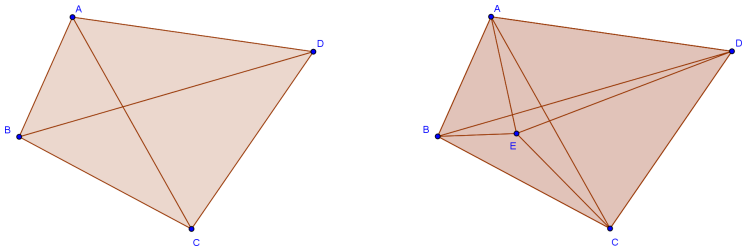
This claim can be verified by exploiting the fact that $\mathcal{S}_1$ is the boundary of $\mathcal{T}_1$.

The whole algorithm is in Table 1, which applies to both edge/face splitting. An example of face splitting is shown in Fig. 7.

Table 1: The coupling algorithm.

Inputs	
1.	The surface $\mathcal{S}_1$ and its faces $\{F_{1i}\}_{i=1}^M$
2.	The recovered surface $\mathcal{S}_2$ and its faces $\{F_{2i}\}_{i=1}^M$
3.	$\mathcal{T}_1$ from stage 1
4.	$\mathcal{T}_2$ from stage 2
Algorithm	
1	Initialize $flag[M]$ to 0
2	For $i = 1:N$
3	find F_{1j} such that $F_{2i} \subset F_{1j}$ (see Claim 5.1)
4	find v_j such that $v_j F_{1j} \in \mathcal{T}_1$ (see Claim 5.2)
5	IF $v_j F_{2i} \in \mathcal{T}_1$
6	$flag[j] = 1$
7	continue
8	END IF
9	Add $v_j F_{2i}$ to $\mathcal{T}_1$
10	END For
11	For $j = 1:M$
12	IF $flag[j] == 0$
13	Remove $v_j F_{1j}$ from $\mathcal{T}_1$
14	END For
Outputs	
1.	A new mesh $\mathcal{T}_1'$

(a) F_1 in $\mathcal{S}_1$ (left) is detected to be recovered as $\{F_{2i}\}_{i=1}^3$ in $\mathcal{S}_2$ (right)



(b) Correspondingly vF_1 in $\mathcal{T}_1$ (left) is split to $\{vF_{2i}\}_{i=1}^3$ in $\mathcal{T}_1'$ (right)

Figure 7: Procedure of coupling meshes.

From above description, it is easy to find that one single split operation only destroys the triangle face F on $\mathcal{S}_2$. Combined with the fact that $\mathcal{S}_2$ is not visible to $\mathcal{S}$, namely $\mathcal{S}_2 \cap \mathcal{S} = \emptyset$, it implies that every split operation would not destroy $\mathcal{S}$. Hence we can draw the conclusion as follows.

Lemma 5.1. *After the coupling operation described in this subsection, the obtained mesh is still constrained to* $\mathcal{S}$.

5.2 Mesh refinement

Though being a conformal and valid mesh after the coupling operation, the mesh $\mathcal{T}_1' \cup \mathcal{T}_2$ often does not meet the mesh size requirement given or imposed by the initial surface triangulation $\mathcal{S}$. To obtain an appropriate mesh, mesh refinement is then performed. By saying refinement, we refer to both mesh refinement and coarsening.

In mesh generation field, a sizing function is usually used to describe the intended mesh density. Sometimes it is equivalent to the control space. This could be given as input or interpolated by mesh generator itself, which is the case of our study. The already obtained $\mathcal{T}_1' \cup \mathcal{T}_2$ serves as the background mesh, in which the sizing function is interpolated.

Various methods could be applied to the following refinement procedure. One example is Ruppert's inserting circumcenters into bad elements [20]. In our method, the normalized edge lengths are calculated [21] and is used as a criterion for the refinement

or coarsening of the given edge. Usually, the value $C_s=\sqrt{2}$ is assigned to the splitting parameter, and the value $C_c=1/\sqrt{2}$ the coarsening parameter. If the edge length is larger than C_s then a splitting operation is performed, while contraction is applied if the edge length is less than C_c. This approach has been used in the surface remeshing topic [22], which also inspires this use in volumetric mesh refinement. The splitting and contracting operations are briefly described as follows.

`Splitting:` If the edge length is larger than C_s, the midpoint of the edge length is inserted in a modified Delaunay way in Section 4.

`Contracting:` If the edge length is less than C_c and neither of the edge ends is on the constraints, both of the edge ends are removed and the midpoint is inserted in a modified Delaunay way. If the edge length is less than C_c and one of the edge ends is on the constraints, only the other edge end is removed. This would guarantee the constraints are not destroyed.

5.3 Mesh optimization

To obtain a mesh with better quality, Centroid Voronoi tessellation (CVT) is this optimization phase. Been studied for the last decade, CVT has been proven useful in diverse applications, one important of which is mesh optimization. The Lloyd's method described in [23] is used in our study only with some modifications and some extra measures.

The iterations between constructing Voronoi diagrams and calculating centroids are performed only on those vertices inside the domain Ω bounded by $\mathcal{S}_2$. The centroids, i.e. Voronoi generators in the next iteration, are checked whether bounded by $\mathcal{S}_2$ and inserted in the modified Delaunay way of Section 4. This would guarantee the new mesh of Ω is still constrained by $\mathcal{S}_2$.

The density function $\rho(x)$ used in CVT optimization is suggested as in [23]:

$$\rho(x)=C/h(x)^5, \tag{5.1}$$

where $h(x)$ is the sizing function, C is a constant usually set to 1. The sizing function is computed on a regular or adaptive grid. The voxels intersecting with the input surface mesh is determined by the intersected triangle size, e.g. shortest edge length. The sizing function for all other non-intersecting voxels is interpolated. This interpolation scheme contains two steps, with the sizing function of all non-intersecting voxels initialized to the average of the whole domain. The first step is smoothing on all voxels via any classic approach, e.g. Gaussian filter or Laplacian smoothing. The second step is to recover the sizing function of voxels intersecting the surface to their values before smoothing. These two steps are performed iteratively until some criteria are satisfied such as the difference of sizing function is smaller than a certain tolerance.

Another alternative optimization way is to construct standard Delaunay triangulation in each iteration, which may destroy the integrity of $\mathcal{S}_2$. Hence further conforming

boundary recovery on $\mathcal{S}_2$ and coupling operation is required. In experiment, we found that this alternative could effectively reduce the number of bad tetrahedra near the $\mathcal{S}_2$, only requiring extra boundary recovery and coupling operation.

Combining the Lemmas 3.1, 4.1 and 5.1, we can draw the conclusion as follows:

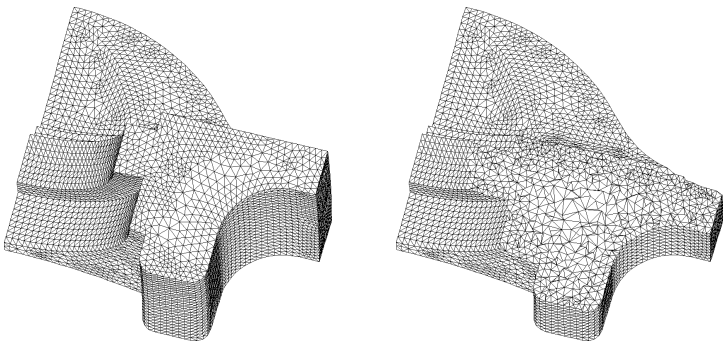
Theorem 5.3. *Let $\mathcal{S}$ be an arbitrary surface triangulation, the algorithm we described in Section* 3, 4, *and* 5 *could effectively generate tetrahedral volumetric mesh constrained to $\mathcal{S}$*

6 Numerical examples

In this section, various examples are presented to illustrate the effectiveness, efficiency and robustness of the proposed method. In our experiment, we choose one or two layers of $\mathcal{T}_1$, since Delaunay triangulation meshing is more efficient than AFT meshing. we prefer efficiency to quality in this study. All experiments were conducted on a PC with Intel Pentium 4 CPU of 3.2GHz and 4GB memory. Four examples are presented. The four geometric models are: a fan disk, an airplane, a femur and a sculpture named "bimba". The examples include pictures of surface triangulations, cutting views of volumetric triangulations. They are presented in Figs. 8 to 13. Also, three tables are given to present the mesh data and statistics on the element quality.

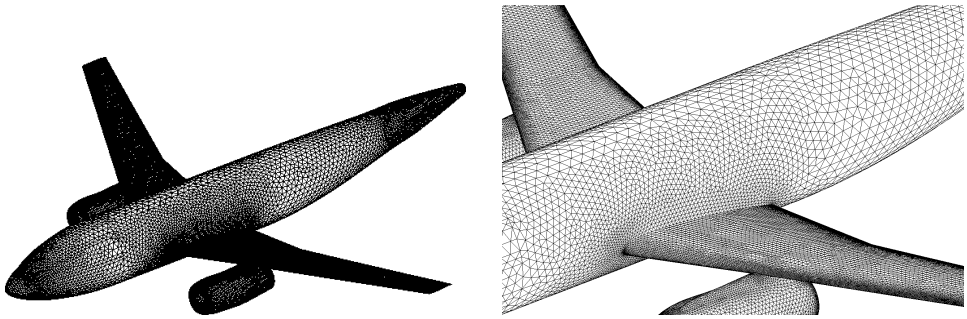
In Table 2, the geometric data for each example are provided which include the total number of vertices in the triangulation, the vertices of the surface triangulation (i.e. boundary vertices), total number of tetrahedra and the triangles of the surface triangulation. In Table 3, some statistics of elements quality (after optimization through the construction of CVT) are provided. They include the average quality, minimum quality, ratio of good elements, ratio of bad elements. Here, the quality of each element [20] ranges from 0 to 1.0, with 1.0 being optimal. And we say a tetrahedron is a good element if its quality is above 0.4, and a bad element if less than 0.1.

At last, mesh quality histograms are drawn in Fig. 14. As is shown, the average of



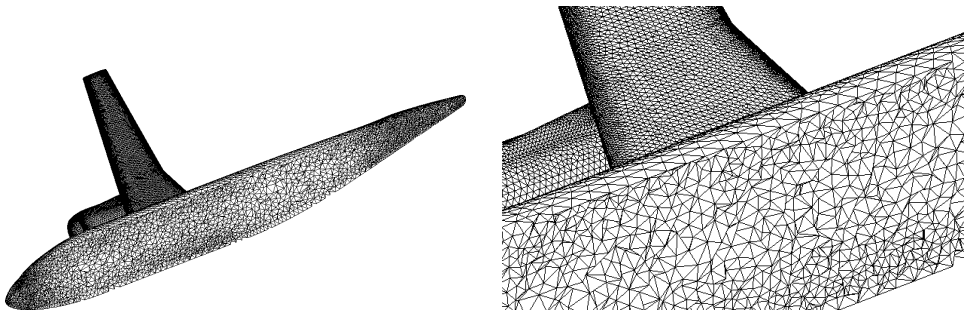
(a) Surface mesh (b) Interior volumetric mesh

Figure 8: Example I: Fandisk.



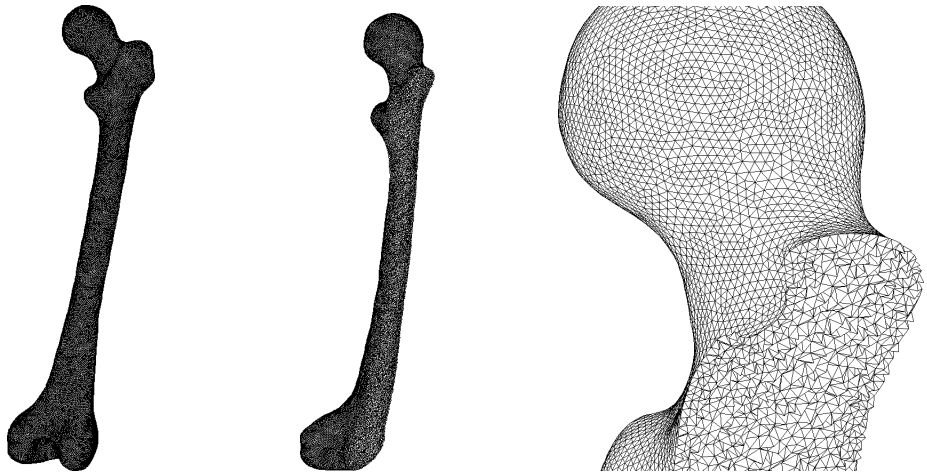
(a) Surface mesh (b) Zoomed view

Figure 9: Example II: Airplane.



(a) Interior volumetric mesh (b) Zoomed view

Figure 10: Example II: Airplane.

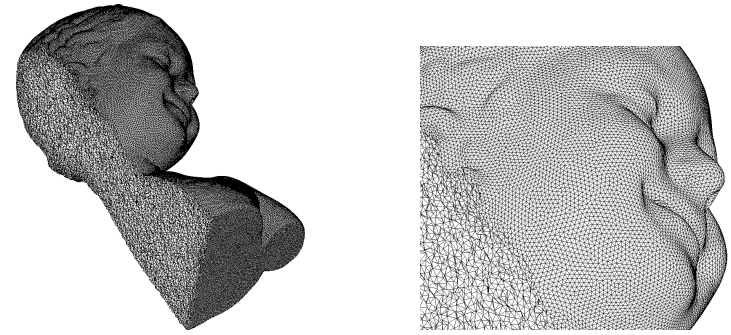


(a) Surface mesh (b) Interior volumetric mesh (c) Zoomed view

Figure 11: Example III: Femur.

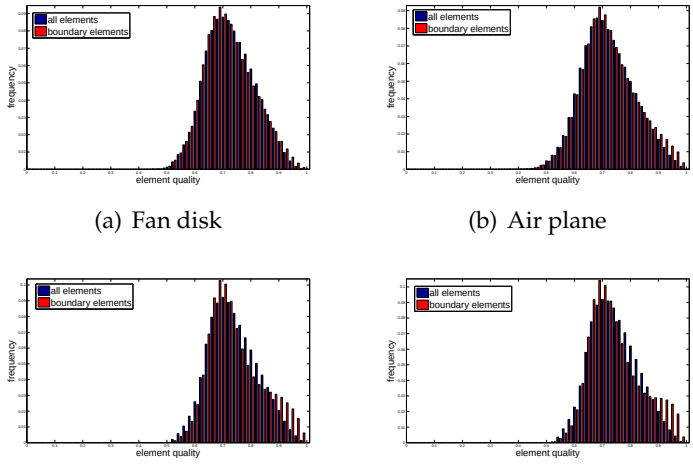
(a) Surface mesh (b) Surface mesh

Figure 12: Example IV: Bimba.



(a) Interior volumetric mesh (b) Zoomed view

Figure 13: Example IV: Bimba.



(a) Fan disk (b) Air plane

(c) Femur (d) Bimba

Figure 14: Statistics on the element quality: all elements and boundary elements.

Table 2: Mesh statistics of the examples.

Examples	fandisk	airplane	femur	bimba
Boundary vertices	16853	71477	28213	74764
Boundary triangles	33702	142958	56422	149524
Total vertices	91614	209700	148197	497273
Total tetrahedra	476463	974270	780343	2680102

Table 3: Elements quality statistics of the examples.

Examples	fandisk	airplane	femur	bimba
Average quality	0.72	0.71	0.73	0.75
Minimal quality	0.23	0.12	0.43	0.04
Percentage of good elements	0.99	0.99	0.99	0.99
Percentage of bad elements	0	0	0	7.46×10^{-7}

Table 4: Near-the-boundary elements quality statistics of the examples.

Examples	fandisk	airplane	femur	bimba
Element number	110805	427979	167728	423940
Average quality	0.71	0.72	0.74	0.76
Minimal quality	0.23	0.12	0.43	0.04
Percentage of good elements	0.99	0.99	0.99	0.99
Percentage of bad elements	0	0	0	4.72×10^{-6}

all mesh elements' quality is more than 0.7, which is also indicated by Tables 3 and 4. It is also observed from this series of histograms that the boundary elements contain more regular tetrahedra as the mesh size increases. This is due to the advancing front method's manner, i.e. trying to construct as regular elements as possible. On the contrary, the boundary elements could not be well optimized by CVT iteration due to the constraints of the boundary. The red bars also indicate a relatively small portion of elements whose qualities are around 0.8. Combining the merits from AFT and limitation from CVT, the boundary elements' qualities are a little higher than the whole mesh quality. This phenomena is due to the mesh optimization method, which only applies to the inner elements. A more complete mesh optimization method is assumed to be improve the results and will be our future study.

7 Conclusion

In this article, a novel method for automatic tetrahedral mesh generation is proposed based on advancing front technique and conforming Delaunay triangulation with boundary recovery. The aim is to generate a boundary constrained tetrahedral mesh. The merits of both kinds of approaches are exploited. The advancing front method generates the

first several layers of elements adjacent to the input triangulation, which guarantees the boundary conforming as well as the good element quality. The conforming Delaunay triangulation meshes the remaining region of interest, which eliminates the heuristics and instability in the advancing front. Steiner points are generated according to the sizing function and inserted. Two mature and effective optimization methods are applied to improve the mesh quality. The validity and the convergence of the method is proved and various numerical examples proves the effectiveness, robustness, and the efficiency.

Acknowledgments

This research is sponsored by Singapore MOE ARC 29/07 T207B2202, MOE RG 59/08 M52110092, NRF 2007IDM-IDM 002-010, Natural Science Foundation of China 10971226 and 91130013, 973 Program of China 2009CB723800 and the foundation of State Key Laboratory of Aerodynamics. The authors thank the referees for their valuable suggestions on the improvement of this paper.

References

[1] Shephard M, Georges M. Automatic three-dimensional mesh generation by the finite octree technique. International Journal for Numerical Methods in Engineering, 2005; 32(4):709–749.
[2] Rassineux A. Generation and optimization of tetrahedral meshes by advancing front technique. International Journal for Numerical Methods in Engineering, 1998; 41(4):651–674.
[3] Löhner R, Parikh P. Generation of three-dimensional unstructured grids by the advancing-front method. International Journal for Numerical Methods in Fluids, 2005; 8(10):1135–1149.
[4] George P, Hecht F, Saltel E. Automatic mesh generator with specified boundary. Computer Methods in Applied Mechanics and Engineering, 1991; 92(3):269–288.
[5] Weatherill N. The integrity of geometrical boundaries in the two-dimensional Delaunay triangulation. Communications in Applied Numerical Methods, 2005; 6(2):101–109.
[6] Frey P, Borouchaki H, George P. 3D Delaunay mesh generation coupled with an advancing-front approach. Computer Methods in Applied Mechanics and Engineering, 1998; 157(1-2):115–131.
[7] Shewchuk J, Miller G, David R. Delaunay Refinement Mesh Generation, 1997.
[8] Karamete B, Beall M, Shephard M. Triangulation of arbitrary polyhedra to support automatic mesh generators. International Journal for Numerical Methods in Engineering, 2000; 49(1-2):167–191.
[9] George P, Borouchaki H. Delaunay Triangulation and Meshing: Application to Finite Elements. Kogan Page, 1998.
[10] Weatherill N, Hassan O. Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. International Journal for Numerical Methods in Engineering, 1994; 37(12):2005–2039.
[11] Du Q, Wang D. Constrained boundary recovery for three dimensional Delaunay triangulations. International Journal for Numerical Methods in Engineering, 2004; 61(9):1471–1500.

[12] Du Q, Wang D. Boundary recovery for three dimensional conforming Delaunay triangulation. Computer Methods in Applied Mechanics and Engineering, 2004; 193(23-26):2547–2563.
[13] George J. Computer Implementation of the Finite Element Method, 1971.
[14] Lo S. A new mesh generation scheme for arbitrary planar domains. International Journal for Numerical Methods in Engineering, 2005; 21(8):1403–1426.
[15] Löhner R. Three-dimensional grid generation by the advancing-front method. Numerical Methods in Laminar and Turbulent Flow, 1987.
[16] Mavriplis D. An advancing front Delaunay triangulation algorithm designed for robustness. Journal of Computational Physics, 1995; 117(1):90–101.
[17] Peraire J, Vahdati M, Morgan K, Zienkiewicz O. Adaptive remeshing for compressible flow computations. Journal of Computational Physics, 1987; 72(2):449–466.
[18] Shostko A, Löhner R. Three-dimensional parallel unstructured grid generation. International Journal for Numerical Methods in Engineering, 2005; 38(6):905–925.
[19] Ito Y, Shih A, Soni B. Reliable isotropic tetrahedral mesh generation based on an advancing front method. 13th International Meshing Roundtable, Citeseer, 2004; 95–105.
[20] Ruppert J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. Journal of Algorithms, 1995; 18(3):548–585.
[21] Frey P. About surface remeshing. Proceedings of the 9th International Meshing Roundtable, Citeseer, 2000; 123–136.
[22] Wang D, Hassan O, Morgan K, Weatherill N. EQSM: An efficient high quality surface grid generation method based on remeshing. Computer Methods in Applied Mechanics and Engineering, 2006; 195(41-43):5621–5633.
[23] Du Q, Wang D. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. Int. J. Numer. Meth. Eng., 2002; 56:1355–1373.