Hybrid meshing using constrained Delaunay triangulation for viscous flow simulations

Feng Wang*,† and Luca di Mare

Department of Mechanical Engineering, Imperial College London, London, UK

SUMMARY

In this paper, we present a generalized prismatic hybrid meshing method for viscous flow simulations. One major difficulty in implementing a robust prismatic hybrid meshing tool is to handle boundary layer mesh collisions, and normally an extra data structure (e.g. quadtree in two-dimensional and octree in three-dimensional) is required. The proposed method overcomes this difficulty via an heuristic approach, and it only relies on constrained delaunay triangulation/tetrahedralization (CDT). No extra data structures are required. Geometrical reasoning is used to approximate the maximum marching distance of each point by walking through the CDT. This is combined with post-processing of marching vectors and distance and prohibition of multilevel differences to form an automatic and robust mechanism to remove boundary layer mesh collisions. Benefiting from the matureness of CDT techniques, the proposed method is robust, efficient and simple to implement. Its capability is demonstrated by generating quality prismatic hybrid meshes for industrial models with complex geometries. The proposed method is believed to be able considerably reduce the effort to implement a robust hybrid prismatic mesh generator for viscous flow simulations. © 2016 The Authors. International Journal for Numerical Methods in Engineering. Published by John Wiley & Sons Ltd.

1. INTRODUCTION

Viscous flow simulations of complex geometries have found its wide applications in industrial CFD. However, creating a quality mesh suitable for these simulations remains a challenge. The approach using multi-block structured meshes needs to decompose the geometry into several blocks, and in each block, a high quality structured mesh can then be created. This method is able to generate optimal viscous meshes for relatively simple geometries. However, for arbitrarily complex geometries, decomposing a geometry into blocks has always been an extremely time-consuming process. A feasible way is to combine the merit of structured and unstructured meshes to generate a prismatic hybrid mesh. This approach creates prismatic boundary layer meshes close to viscous walls to resolve high flow gradients normal to the wall and fills the rest of the domain with an unstructured mesh. This method creates a boundary layer mesh with comparable quality with structured meshes but retains the flexibility of unstructured meshes. There are generally two ways to generate prismatic hybrid meshes [1]: (1) the prismatic mesh is generated after the isotropic mesh; and (2) the prismatic mesh is generated before the isotropic tetrahedral mesh.

The first approach starts from a valid isotropic mesh and grows prismatic elements from the boundary, layer by layer [2, 3]. The tetrahedral mesh is updated correspondingly when the prismatic

*Correspondence to: Feng Wang, Department of Mechanical Engineering, Imperial College London, London, UK.
†E-mail: feng.wang207@ic.ac.uk

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2016 The Authors. International Journal for Numerical Methods in Engineering. Published by John Wiley & Sons Ltd.
meshes are generated. The advantage of this method is that a valid mesh always exists and the isotropic mesh can be used to check mesh intersections. However, this approach might encounter difficulties for internal flow applications. For example, the dimension of the endwall clearance of turbomachinery rotor blades or the seal clearances are several orders of magnitudes smaller than the characteristic dimensions of the models. Growing prismatic elements away from the wall might squeeze the isotropic mesh in these regions and result in a mesh with poor quality.

For the second approach, the basic principle is to gradually ‘inflate’ the boundary discretization at distinct steps and meanwhile generate the prismatic elements. The domain enclosed by the inflated boundary is then filled with an isotropic tetrahedral mesh. This approach can be combined with the advancing front method [4–6], the Delaunay insertion method [7–10] or the combination of the two [11–13]. Compared with the first approach, the second approach is able to provide more freedom in generating the prismatic elements, while optimizing the transition from the prismatic meshes to the isotropic meshes and maximizing their quality. The method described in this paper is based on the second approach.

There are several key issues that have to be addressed in order to make the hybrid meshing efficient and reliable. One issue is to detect collisions of prismatic meshes during the inflation process. The collision detection can be performed after the prismatic meshes are generated [4, 14] or when a new layer of prismatic meshes is being generated [12, 15]. Collision detection in three-dimensional (3D) is a non-trivial process because one has to implement the geometrical predicates carefully in order to avoid the robustness issues caused by finite precision computations. Furthermore, an extra data structure (e.g. quadtree in two-dimensional (2D) or octree in 3D) is required to reduce the computational complexity of search operations required by the algorithm.

Constrained delaunay triangulation/tetrahedralization (CDT) is fundamental to many applications. For example, it can be used as an intermediate mesh by many meshing techniques [16–18] to create quality meshes. It can also be used as a searching data structure. Assume the dimensionality is \( d \), the input to construct a CDT is a planar straight linear graph in 2D and a piece-wise linear complex (PLC) in 3D. The boundary meshes used in engineering applications are a subset of these two geometric types in 2D and 3D, respectively. When \( d = 2 \), CDT always exists and can be constructed without Steiner points. A 2D CDT can be constructed by the divide-and-conquer[19] or by the incremental method [20]. When \( d = 3 \), constructing a CDT is considerably more complicated. It is possible that a PLC might not have a CDT and the simplest example is the Schönhardt polyhedron [21]. Steiner points are required to convert the PLC into a new PLC so that its CDT exists. At the final stage, Steiner points are removed from the CDT. Shewchuk [21] proposed a CDT algorithm with guaranteed success, and this algorithm was further developed by Si [22]. The CDT algorithms are sensitive to numerical errors, and this can be fixed by using exact arithmetic [23] and symbolic perturbations [24, 25].

For Delaunay-based meshing techniques, a CDT has to be constructed in the first place, and it is then refined to achieve the desirable mesh quality and sizes. Because the techniques to construct CDT in both 2D and 3D can be considered to be mature [18, 26], the possibility of only relying on CDT for the collision tests could considerably reduce the effort to implement a robust hybrid meshing tool. Hence, the research question is as follows: *is it possible to use only the CDT to develop a robust hybrid meshing tool?* This paper presents our effort to answer this question, and the contribution of this paper is to present a generalized hybrid meshing method that only relies on CDT. The performance and robustness of the method are demonstrated using academic and industrial geometries with a particular focus on internal flow applications.

The structure of the paper is organized in the following: Section 2 gives the outline of the hybrid meshing method; Section 3 describes the details of computation and post-processing of marching vectors and distance. Section 4 shows how to remove multilevel differences in the boundary layer mesh. Section 5 describes a novel approach to remove boundary layer mesh collisions by reasoning a CDT of the input mesh. Section 6 shows the procedure to update meshes on non-slip walls, and Section 7 briefly the techniques to generate isotropic meshes that fill the domain enclosed by the inflated boundary. Section 8 demonstrates the capability of the proposed method by examples. Conclusions and future work follow in Section 9.

---

4In the following text, the phrases ‘hybrid meshing’ or ‘prismatic meshing’ refers to the second approach and ‘boundary layer meshes’ and ‘prismatic meshes’ are used interchangeably.
2. OVERVIEW

Consider the boundary mesh of a geometry with dimensionality $d$. The hybrid meshing method focuses on $(d-1)$-simplexes in the boundary mesh. When $d = 2$, a $(d-1)$-simplex is an edge; when $d = 3$, it is a triangle. We refer to a boundary entity as a $(d-1)$-simplex in the boundary mesh, namely, an edge in 2D and a triangular face in 3D. The central idea of the hybrid meshing method is the following: boundary entities are gradually marched away from their initial positions in discrete steps, and the boundary layer meshes are created at the same time. The major input parameters to generate a hybrid mesh are

- $T$: Boundary mesh
- $n_l$: Maximum number of layers of the boundary layer mesh
- $d^0$: Initial marching distance
- $s$: Expansion ratio

In 2D, the boundary mesh is a list of edges that represent a manifold polygon. In 3D, it is the surface triangulation of a manifold polyhedron. $n_l$ is the maximum number of layers of prismatic meshes. The actual number of layers of prismatic meshes can be less than $n_l$, depending the geometry and mesh spacing of the boundary mesh. $d^0$ is the initial marching distance. It could be a single value for all the points on $T$, a single value for each point on $T$ or indirectly computed by specifying the aspect ratio of the first layer of prismatic mesh. $s$ is the expansion ratio of the thickness of the prismatic mesh. The nominal thickness of the $i$th layer of the prismatic mesh is computed as

$$d_{n,p}^i = s^{i-1}d^0. \quad (1)$$

The general procedure to create a hybrid mesh is shown in Algorithm 1, and it will be discussed in the following sections in detail. It should be noted that CDTs are constructed twice. In the first pass, the CDT of $T$ is used to detect gaps in the geometries, and for the second pass, the CDT of the inflated $T$ is used to create isotropic $d$-simplex.

**Algorithm 1** Hybrid meshing

1: input: $T$, $n_l$, $d^0$, $S$
2: //the first time to construct CDT
3: detect gaps;
4: initialize the marching front $F$, $F \leftarrow T$;
5: collect a point list $P$, $P \subseteq F$ and switch on the marching status of $P$;
6: // $n$ is the current number of layers that is created.
7: while $n \leq n_l$ & $P \neq \emptyset$ do
8: compute and post-process marching distance and vectors for each point in $P$;
9: march boundary entities and create one layer of boundary layer mesh;
10: update $F$;
11: update $P$;
12: $n = n + 1$;
13: end while
14: //the second time to construct CDT
15: fill isotropic $d$-simplex in the domain enclosed by inflated boundary mesh;

3. MARCHING VECTORS AND DISTANCE

The evaluation of marching vectors and distance is one of the crucial steps in the hybrid meshing. The hybrid meshing starts from a marching front $F$, which is identical to the initial boundary mesh $T$. When the marching front $F$ moves away from $T$, its curvature should be reduced gradually so as to smear concave and convex corners and facilitate the marching process. The key to achieve this lies in the evaluation and post-processing of the marching vectors and distance.
3.1. Computation of Marching Vectors and Distance

A point \( p \in F \) marches along its marching vector by the marching distance. The marching vector is computed on the manifold of \( p \). The manifold of \( p \) is the set of edges/faces on \( F \) with \( p \) as a vertex. In 2D, the marching vector is the bisector of the two edges sharing \( p \). In 3D, the marching vector lies on the bisection plane of the two faces on the manifold forming the wedge with the smallest angle. The location of the marching vector on that plane is evaluated by bisecting the visibility region on that plane [27]. As is shown in Figure 1, the visibility region is represented by a polygonal/polyhedral cone extending outward from the point, and it can be simplified into a visibility cone with the line/circular cross section and half-cone angle, which can also be called the visibility angle \( \beta_{\text{visi}} \). This approach is found to be able to consistently yield a valid normal for the manifold. The marching distance is determined based on the manifold and its characteristic angle \( \beta \). The marching distance \( d_p \) of a local point \( p \) is considered to be a linear function of \( d_{n,p} \):

\[
d_p = (1 + \alpha)d_{n,p}
\]

\( \alpha \) is a function of the characteristic angle \( \beta \). \( \beta \) is the average angle between adjacent edges/faces in the manifold. A manifold is convex if \( \beta \) is larger than 180° and vice versa. In 2D, \( \beta \) is trivial to evaluate, and it is simply the internal angles formed by the two adjacent edges sharing \( p \). In 3D, its evaluation for typical cases is illustrated in Figure 2. \( \alpha \) is negative when the manifold is a convex region and positive when the manifold is a concave region [12]. This means that if \( p \) is on a concave corner, its marching distance is slightly larger than its surrounding points, and if \( p \) is on a convex corner, its marching distance is slightly smaller than its surrounding points. Equation 2 ensures that the curvature of \( F \) is reduced gradually during the marching.

Kallinderis et al [12] proposed a similar equation to Equation 2, but no further details are given on the definitions of \( \beta \), the linear function between \( \alpha \) and \( \beta \) or the determination of a concave or convex region. In this work, \( |\alpha| \) is related to the visibility angle \( \beta_{\text{visi}} \), and its sign is determined by the characteristic angle \( \beta \). \( \alpha \) is equal to \( |\cos(\beta_{\text{visi}})| \). The reason to select a cos function is that its value is between -1 and 1. A local point can only march at most twice of the nominal marching distance \( d_{n,p} \). Furthermore, the cos function varies slowly around 180°; \( \cos(\beta_{\text{visi}}) \) is insensitive to the small

Figure 1. Marching vector and visibility cone. The red arrows represent the computed marching vector, and the yellow line/circular section represents the cross-section of the simplified visibility cone.

Figure 2. Illustration of characteristic angles of manifolds.
variation of the \( \beta_{visi} \). Finally, the non-linear equation between \( \beta_{visi} \) and the local marching distance \( d_p^i \) is shown in Equation 3. The total marching distance of \( p \) is approximately \( \sum_{i=1}^{n} d_p^i \).

\[
d_p^i = \begin{cases} 
(1 + |\cos(\beta_{visi})|)d_{n,p}^i & \text{if } \beta \in [0^\circ, 180^\circ) \\
(1 - |\cos(\beta_{visi})|)d_{n,p}^i & \text{if } \beta \in [180^\circ, 360^\circ) 
\end{cases}
\] (3)

3.2. Post-processing of Marching Vectors and Step

The computed marching vectors and distance need to be post-processed to ensure a smooth variation across the front and facilitate the following marching process. A smooth variation of marching vectors and distance guarantee a smooth variation of orientation and thickness of the final boundary layer mesh. Constraints are put on marching vectors and distance so that no inverted or low quality \( d \)-simplexes are created during the post-processing. The points in the marching front \( F \) are divided into the following four categories, and their marching vectors and distance are post-processed accordingly.

1. **Category 1**: \( p \) is in this category if \( p \in F \) with its manifold \( \Omega \) and if \( \exists (t_i, t_j) \subset \Omega \) such that \( \angle(t_i, t_j) \geq 60^\circ \). The marching vector and distance of \( p \) are fixed. This will improve the mesh quality of \( F \) close to concave and convex corners, such as the red, green and blue dots shown in Figure 2.

2. **Category 2**: \( p \) is in this category if it is not in Category 1 and it is on the border between non-slip walls and non-wall boundaries. Its marching vector and distance are weighted averages of the marching vector and step of the neighbouring points belonging to Category 2.

3. **Category 3**: \( p \) is in this category if \( p \) is not in the first two categories and has at least one neighbouring point in \( \Omega \) whose marching vector is fixed. The marching distance of \( p \) is not smoothed, and only its marching vector is smoothed. The rationale is illustrated in Figure 3. In the figure on the left, the green dot represents a point in satisfying category 1 and its marching vector, and distance is fixed. The black dashed line illustrates the shape of the inflated boundary if both the marching vector and distance of the black dots are smoothed. The black dashed lines represent the shape of the inflated boundary if the marching vector is smoothed. Because the green dot is on a convex corner and its marching distance is slightly smaller the black dots, it is desirable that the marching distance of the black dots is not influenced by the reduction of the marching distance of the green dots, so that the concave corner in the original geometry can be gradually smeared during the inflation. Therefore, the green dot does not contribute to the smoothing of the marching distance of the black dots. For Figure 3 (right), the green dot is on a concave corner, and its marching distance is slightly larger than the black dots, which are in the current category. In order to smear the concave corner, the green dot does not contribute to the smoothing

\[d_p^i\] will also be subject to local smoothing; hence, its value will be updated slightly.
of the marching distance of the black dots, and it only contributes to the smoothing of their marching vectors.

4. Category 4: $p$ is in this category if it is not in the aforementioned three categories. All the neighbouring points of $p$ in its manifold $\Omega$ contribute to the smoothing of its marching vector and distance.

The marching vectors and distance of the points in category 1 are always fixed. This is because their visibility angles are small and any changes to their marching vectors might result in inverted elements. It is also important to note that when a new layer of prismatic elements are generated and the front is updated, the points on the front are re-classified into these four categories. Because the method tries to smear concave and convex corners in the original geometry, a concave or convex point in the initial boundary would become a point in category 4 when it is marching away from the boundary. If the quality of the boundary mesh is low (e.g. sudden change of element sizes), it is possible that a point in category 1 might become a point in category 4 during the inflation. Because the inverted elements are mostly generated by points in category 1, the classification of the points on the front is updated continuously when the front is marching away from the boundary, and this ensures that concave/convex corners are always smeared and no inverted elements are generated during the inflation.

The method to smooth a marching vector or distance of $p$ is the weighted Laplacian smoothing [27]

$$q_i^* = \frac{1}{\sum_j d_{ij}^{-1}} \sum_j \left( d_{ij}^{-1} q_j \right)$$  \hspace{1cm} (4)

where $q_i$ represents the value to smooth, $q_i^*$ is the final value after smoothing, $q_j$ is the value of the neighbouring points of $p$ in $\Omega$, $d_{ij}$ is the Euclidean distance between $p$ and its neighbour points in $\Omega$.

If the smoothed marching vector of $p$ lies outside of the visibility cone, its initial marching vector before smoothing is used instead. This guarantees that no inverted elements are created. Furthermore, the smoothed vector should not deviate from the direction of the previous marching vector for the same point by more than a certain angle, say $10^\circ$. This avoids abrupt changes of mesh orientation along marching directions. If the deviation is larger than $10^\circ$, the final marching vector is taken as the average of the smoothed marching vector and the marching vector one layer below under the condition that no inverted elements are created. If inverted elements are generated, the original marching vector is recovered.

4. UPDATE MARCHING POINT LIST AND ELEMENT CREATION

Assuming the marching distance and vector for $p$ is already computed, $p$ is marched to create a new point, $p^*$. The marching status of $p$ is switched off and removed from $P$, and the newly created point $p^*$ will be appended into $P$ if all of the following criteria are satisfied:

1. Its marching distance is less than a fraction, say 0.60, of the average length of edges sharing $p$ in its manifold.
2. The prismatic element created by marching $p$ has an acceptable quality.
3. The total marching distance of $p$ is less than 30% of its maximum marching distance (MMD).
4. The point is not on a cliff (see Figure 4 and following discussion).

Criterion 1 ensures a smooth transition from the boundary layer mesh to the isotropic tetrahedral mesh. Criterion 2 avoids generating prismatic meshes with poor quality. The quality is assessed by the largest angle between the edges generated by $p$ and $p^*$ and the normal of the base surface. The default value is $80^\circ$. If the largest angle is larger than the specified value, a bad element or inverted element (if larger than $90^\circ$) is generated. The marching status of $p$ is switched off, and its marched point $p^*$ is not appended to $P$. 

© 2016 The Authors. International Journal for Numerical Methods in Engineering. Published by John Wiley & Sons Ltd.

Int. J. Numer. Meth. Engng 2016; 108:1667–1685

DOI: 10.1002/nme
Criterion 3 controls the height of the prismatic meshes so as to avoid boundary layer collisions in regions with narrow gaps. If $p_{-1}$ is the position of $p$ in the initial boundary, the height of the prismatic mesh for $p$ is taken as the distance between $p_{-1}$ and the projection of $p^*$ onto the marching vector of $p_{-1}$.

Switching off the marching status of certain points does not cause difficulties in 2D. However, in 3D, highly stretched quadrilateral faces of the prisms might be exposed to the isotropic tetrahedral generator. Transition elements, like tetrahedron and pyramid, are created to hide these highly stretched faces from the tetrahedral mesh generator [13, 28]. Criterion 4 is used to control the quality of the transition elements. This is shown in Figure 5. Because the boundary layer mesh is created layer by layer, if the difference in number of layers between neighbouring points is larger than one, low quality tetrahedra or pyramid might be created. We propose an effective approach to remove such multilevel difference. We define a point $p \in F$ as a cliff point if it has an incomplete manifold $\Omega$. The manifold $\Omega$ of $p$ is complete if starting from a face in $\Omega$, one can always go back to the same face in clockwise or counter-clockwise directions; otherwise, $\Omega$ is incomplete. The marching status of a cliff point is always turned off, and the multilevel difference is then automatically removed. This is illustrated in Figure 4. However a special situation of incomplete manifolds is the points on the border of non-slip and non-wall boundaries (described in Section 6). These points always have an incomplete manifold when $n > 1$, and their marching statuses are only switched off if their marching violates other criteria.

After the marching status of each point in the front is determined, in 2D, only triangles are generated. In 3D prism, tetrahedra and pyramid elements are created if there are 3, 2 or 1 points on the base face in the front that are marched. This is illustrated in Figure 5. To update the marching front $F$, for each base face $t \in F$ if the marching status of vertices of $t$ is all switched off, $t$ is removed from $F$. If the marching status of certain vertices of $t$ is switched off, $t$ is used to create transition elements, as is shown in Figure 5 for 3D cases. If all these vertices are marched, then the edge/face $t^*$ created by their marched points is added into $F$. 

Figure 4. Illustration of cliff points. Cliff points are highlighted in red dots.

Figure 5. Element formation in 3D. The red dots represent the marching points, and the green dots represent the points on the cliff.
5. BOUNDARY LAYER MESH COLLISION REMOVAL

Let \( \mathbf{l}_p \) be the marching vector of a point \( p \in T \), the MMD of \( p \) can be computed as

\[
\text{MMD} = (\hat{\mathbf{p}} - \mathbf{p}) \cdot \mathbf{l}_p
\]

(5)

where \( \hat{\mathbf{p}} \) is the intersection between \( T \) and the ray emanating from \( p \) along the direction of \( \mathbf{l}_p \). Given the initial boundary mesh \( T \), we do not attempt to compute neither the exact position of \( \hat{\mathbf{p}} \) nor the value of MMD. An approximated value of MMD should be sufficient for detecting boundary layer mesh collisions. An approximated MMD can be obtained by traversing a CDT of \( T \). The viscous boundary layer is only a thin layer attached to the body and normally several orders of magnitudes smaller than the characteristic dimension of the body. In regions where the true MMD is much larger than the thickness of the boundary layer mesh, an approximate estimation of the true MMD is sufficient. For regions with small gaps and cavities, it will be shown in the following that traversing a CDT would yield accurate estimation of the size of small gaps and cavities.

For a point \( p \in T \), we walk through the CDT along the direction of its marching vector. The walk will finally hit a boundary entity \( t \in T \). The distance between \( p \) and \( t \) is taken as the approximated MMD of \( p \). This is described in Algorithm 2, and Figure 6 illustrates the basic idea. The distance between \( p \) and \( t \) is computed as follows: if the projection of \( p \), \( p_{prj} \), to \( t \) is inside \( t \), its MMD is the distance between \( p \) and \( p_{prj} \), if not, its MMD is equal to the shortest distance between \( p \) and the vertices of \( t \).

The method to walk through the CDT is similar to the jump-and-walk method, which is an efficient point location technique used extensively in Delaunay triangulations [29–32]. If the dimensionality of the mesh to walk through is \( d \), Mucke et al [33] showed that its computational

\begin{algorithm}
\textbf{Algorithm 2 Jump and Walk}
\begin{algorithmic}
\Require \text{CDT}, \text{enquired point } p_0, \text{a marching vector, a point } p_\infty \text{ far enough from } p_0 \text{ along the marching vector;}
\\lVert t \rVert \text{ is an } (d - 1)\text{-simplex in the CDT, } \sigma \text{ is a } d\text{-simplex in the CDT}
\\lVert t_0 \rVert \text{ is an } (d - 1)\text{-simplex with } p_0 \text{ as one of its vertex, } \sigma_0 \text{ is a } d\text{-simplex in the CDT with } t_0 \text{ as one of its face.}
\Repeat
\hspace{1em} walk to the } i \text{th neighbour } \sigma_{neig} \text{ of } \sigma \text{ through its } i \text{th } (d - 1)\text{-simplex } t_{neig} \text{ if an objective function is optimized;}
\hspace{1em} t \leftarrow t_{neig};
\hspace{1em} \sigma \leftarrow \sigma_{neig};
\Until \quad t \subseteq T
\end{algorithmic}
\end{algorithm}

Figure 6. Approximation of MMD. The red tetrahedra is the ones that are visited by the walking.
complexity is $O(n^{1/(d+1)})$. In our case, $n$ is the number of points on $T$. Starting from a point in the mesh, $d$-simplexes are walked through neighbour by neighbour. In Algorithm 2, $t$ is a $(d - 1)$-simplex in a CDT, and $\sigma$ is a $d$-simplex in a CDT. The destination point $p_\infty$ is a point far enough from $p_0$ along the marching vector, and it is strictly outside of the CDT. The algorithm starts from a $d$-simplex $\sigma_0$ with $p_0$ as one of its vertices and a $(d - 1)$-simplex $t_0$ with $p_\infty$ strictly lying above it\(^1\). The criteria to decide which neighbours to walk through depend on whether $p_\infty$ lies below the edges/faces to walk through and the choice of an objective function.

Conventionally, the objective function is the distance between the point opposite to the $(d - 1)$-simplex to walk through in the neighbouring element and $p_\infty$. The $(d - 1)$-simplex is chosen to walk through if it minimizes such distance, and $p_\infty$ lies below this $(d - 1)$-simplex. If a $(d - 1)$-simplex is on the boundary, because there is no points opposite to this $(d - 1)$-simplex, this $(d - 1)$-simplex is not considered. The algorithm will turn to $(d - 1)$-simplexes where $p_\infty$ lies below and the distance to $p_\inf$ is minimized and finally hit the boundary if there are no $(d - 1)$-simplexes to walk through. This is illustrated in Figure 7 (right).

We have found through experiments that MMD can be approximated more accurately if a hybrid objective function is used. If both $(d - 1)$-simplexes are not on the boundary, the objective function that evaluates the distance is used. Otherwise, the objective function that evaluates the volume formed by the $(d - 1)$-simplexes and $p_\infty$ is used. This function is optimized if a smaller negative value is obtained \(^2\). Figure 7 shows the comparison between the objective function minimizing the distance and the hybrid objective function. It can be seen clearly that the objective function only by distance always tries to visit a $d$-simplex in the mesh while the one with a hybrid objective function quickly hits a boundary entity and yields a more accurate MMD approximation. It is also possible to use the volume as the objective function throughout the jump-and-walk, but computing volume is much more expensive than computing distance; therefore, the hybrid objective function is preferred.

It should be noted that the jump-and-walk approach only approximates the MMD and does not necessarily yield the exact value. Based on our observation, the accuracy is generally inversely proportional to the number of visited $d$-simplexes. Fortunately, for narrow regions, only a few $d$-simplexes are visited; therefore, an accurate value can be obtained. In the case of visiting many $d$-simplexes, which indicates the size of a gap or the cavity is large, it is found to be still sufficiently accurate to avoid boundary layer mesh collisions. Figure 6 shows the tetrahedron that is visited by the jump-and-wall for MMD approximation. Despite that more than 50 tetrahedra have been visited, compared with the exact value, the relative error is still within 2%.

However, we have found that boundary layer mesh collisions might still not be completely removed by solely relying on MMD approximations. There are situations that MMD might fail to predict possible collisions. As is illustrated in Figure 8 (top), because the MMDs are large values for points close to a concave corner with roughly 90°, collisions are still theoretically possible. Interestingly, if their marching vectors and distance are postprocessed in the way described in Section 3,

---

\[^{1}\] This $(d - 1)$-simplex can always be found, because $p_\infty$ is strictly outside of the CDT.

\[^{2}\] This is because $(d - 1)$-simplexes of a $d$-simplex are oriented anticlockwise. A valid $(d - 1)$-simplex to walk through always yields a negative volume.
collisions are found to be effectively removed. The reasons are as follows: (1) the marching vectors are smoothed, the possible locations where collisions could take place are pushed further away; and (2) the marching distance of a concave point is slightly larger than its surrounding points, its marching status will be switched off at an earlier stage than its surrounding points. This effect will propagate to surrounding points to prune prisms around the corner, because multilevel difference is not allowed. The other situation is shown in Figure 8 (bottom). Two spacious domains are connected by a narrow duct. Within the duct, collisions can be predicted satisfactorily. Close to the regions where the narrow duct meets the two spacious domains, even if MMD is accurately predicated, collisions are still possible. Fortunately, this can be automatically fixed by the prohibition of multilevel difference. As long as the MMD of its neighbours is predicated accurately enough, the termination of the marching of its neighbours will propagate in all directions and switch off the marching status of its surrounding points recursively. Based on our extensive tests, we have found that the combination of MMD, post-processing of marching vectors and distance and prohibition of multilevel difference is effective in avoiding boundary layer collisions.

6. BOUNDARY MESHES ON NON-SLIP WALLS

The previous sections assume that all boundaries are non-slip walls. In engineering applications, there are also boundary types such as freestream, periodic, symmetry and so on. We refer to these boundaries as non-wall boundaries. Non-wall boundaries are adjacent to non-slip walls or to each other, and no boundary layer meshes are created on them. On the borders between non-wall boundaries and non-slip walls, points are still marched to create boundary layer meshes on non-slip walls. The marching vectors of these points need to be carefully controlled so that the boundary layer meshes lie on these non-wall boundaries. In most cases, non-wall boundaries are planar surfaces; the marching vectors of points on the border of non-wall and non-slip boundaries are simply projected on these surfaces. If non-wall boundaries are curved surfaces, a parametric definition of these curved surfaces is required to guide the points to march on the surfaces.

In particular, periodic boundary is one of the most commonplace boundary types in engineering applications, such as turbomachinery. Some solvers require meshes on periodic boundaries to match exactly. This can be achieved by pairing points on periodic boundaries and synchronizing their marching directions and marching vectors.
After boundary layer meshes are created, boundary meshes on non-wall boundaries need to be modified, because only their borders with non-slip walls are moved and their interior points are still at their original locations. In our current implementation, mesh optimization including mesh smoothing and topology cleanup is operated directly on the surface meshes. If meshes on periodic boundaries are required to match exactly, only one of them is optimized, and its counterpart is recreated by shifting it by one periodicity.

7. GENERATION OF ISOTROPIC ELEMENTS

The CDT of the domain enclosed by the inflated boundary is refined by Delaunay refinement to yield meshes with good quality and desired mesh sizes. The Delaunay refinement is described in our previous work [34]. The algorithm is based on the ones developed by Shewchuk [35] in 2D and Si [16] in 3D. The major difference is that the boundary mesh of the CDT is never modified in the refinement. This ensures that the isotropic mesh and the boundary layer mesh have a point-to-point matching at their interface. The isotropic mesh is optimized by combined mesh swapping and mesh smoothing [36]. The final mesh quality is further enhanced by the open-source mesh improvement tool-kit Mesquite [37].

8. RESULTS

The proposed method has been implemented in C++, and its capability is demonstrated with several examples from 2D academic test cases to 3D industrial cases with complex geometries for internal flow applications. All the timing statistics are obtained in a workstation with Intel Xeon E5-1630@3.70GHz with 16G RAM.

8.1. Academic Test Cases

Figure 9 shows the meshes of three wedges with smallest angles of $5^\circ$, $11^\circ$ and $22^\circ$, respectively. Despite the simplicity of the geometries, it is a challenging case, because boundary layer mesh collision might occur around the regions with the smallest angle.

The thickness of the first layer is set to a fixed value for all the points, and the expansion ratio normal to the wall is by default set to 1.2. These three simple geometries are to demonstrate the capability of the method in meshing geometries with small input angles in the geometry. For the top pictures in Figure 9, the smallest input angle is $5^\circ$; no boundary layer mesh is created near the regions where the smallest input angle is. However, the mechanism to stop prismatic meshes from being generated for point a and b is different. For point a, its MMD is sufficiently large, but it would yield an extremely skewed prismatic elements; therefore, its marching status is switched off. For point b, its marching status is switched off because the thickness of the prismatic element is larger than a fraction, which is a user-defined value, of its MMD. As the smallest input angle increases gradually to $11^\circ$ and $22^\circ$, boundary layer meshes are gradually recovered around that region.

![Figure 9. 2D hybrid meshes for three right triangles with smallest angles of 5°, 11° and 22°, respectively.](image-url)
8.2. Industrial Test Cases – 2D

The flow passages in gas turbines are used as the industrial test cases to demonstrate the capability of the method. The selected geometries are characterized by large cavities connected by seal clearances and secondary flow passages (e.g. cooling holes) combined with the main gas path. The dimension of the seal clearances and the cooling holes is several magnitude smaller than the characteristic dimension of the geometry; therefore, these are rather challenging cases.

Figure 10 shows the hybrid mesh for an idealized cavity that is below the combustion chamber of a gas turbine engine. The geometry is characterized by three cavities connected by two labyrinth seals. This test case demonstrates the capability of the removal of the boundary layer collision around the regions close to the seal clearance. The marching distance of the first layer of prismatic mesh is fixed throughout the initial front, and the expansion ratio in the normal direction is 1.2. The geometry is fairly complex, and the number of layers of the prismatic mesh would change considerably from spacious regions to narrow regions. We can see in spacious regions, excellent boundary layer meshes are created and the transition of mesh sizes from boundary layer meshes to triangles is optimal. Close to narrow regions, the growth of boundary layer meshes is trimmed automatically to avoid collisions. Besides, one can also observe a smooth variation of the orientation and the height of the boundary layer meshes.

Figure 11 shows the hybrid mesh of the shroud cavity below a compressor stator blade. The geometry is represented by two cavities connected by a seal. What is different from the previous case shown in Figure 10 is that the marching distance of the first layer is indirectly computed by fixing the aspect ratio of the first layer of the prismatic mesh. Close to the seal clearances, the thickness of the first layer will be much smaller than the thickness of the first layer around the regions far way from the seal. This would result in a better resolution of the boundary layer round the fins of the seal. We can observe that the thickness of the boundary layer mesh varies smoothly on the walls, and the transition between prismatic meshes to triangles are optimal. Trimming of the prismatic mesh is considerably reduced compared with the case shown in Figure 10. Therefore, for complex geometries specifying the aspect ratio of the first layer is recommended.

8.3. Industrial Test Cases – 3D

Figure 12 shows the hybrid meshes for a simplified nozzle guide vane (NGV) with cooling holes, as is shown in the top left figure in Figure 12. The NGV is the stator blade immediately behind the combustion chamber. The temperature entering NGV is higher than the melting temperature of
Figure 11. Hybrid mesh for Cavity-2.

Figure 12. Top-left: Overview of the geometry. Top-right: Overview of the boundary layer mesh of the NGV. Bottom: Close-up view of the hybrid mesh.

The NGV; therefore, cooling airs are injected into the passage around the blade to form a film to insulate the blade from the hot air in the passage. The effectiveness of the film cooling is crucial to the reliability of gas turbine engines. The geometries used here are idealized. The dimension of the cooling holes is exaggerated, and their relative position on the blade does not necessarily represent the design intent. The geometry consists of the NGV passage, 88 cooling holes, and a spanwise channel inside the NGV, and it is discretized by 59,709 points and 119,794 triangles.

Figure 12 (top right) shows the boundary layer mesh on the walls, and figures on the bottom show a close-up view of the hybrid mesh. The thickness of the boundary layer mesh varies smoothly in the geometry, and trimming is invoked around the cooling slots to remove potential collisions.

The test case shown in Figure 11 represents a stator cavity without non-axi-symmetric features. For the secondary air system of turbomachinery, the cavities might be connected by non-axi-symmetric orifices. The geometry shown in Figure 11 is revolved by $10^\circ$ around the engine axis, and an non-axi-symmetric orifice is added to demonstrate the capability of the method in meshing...
3D cavities. This is demonstrated in Figure 13. The figure on the top-left shows the geometry and the surface mesh of the final hybrid mesh. A sectional view of the mesh is shown in the figure on the top-right. The figures on the bottom show the close-up view of the hybrid mesh. The thickness of the boundary layer meshes changes smoothly in the domain, and trimming is invoked close to the regions around the seal. The test case demonstrates that the removal of boundary layer meshes works equally well both in 2D and 3D.

8.4. CFD Applications

The developed meshing tool has been routinely used in the group for CFD analysis. The CFD solver is an in-house fully implicit, cell-centred, unstructured, finite volume solver. A list of popular turbulence models are available, such as Spalart–Allmaras, Wilcox $k-\omega$, Menter $k-\omega$ SST, $k-e$ and an explicit algebraic Reynolds stress model. Inviscid fluxes can be constructed by Roe Riemann solver, AUSM and AUSM UP+. Gradient reconstruction is performed with the weighted least-square approach, and second order accuracy in space discretization is achieved by the biased Monotonic Upstream-centered Scheme for Conservation Laws (MUSCL) scheme and the van Albada limiter. The computations shown in this paper are performed with the Wilcox $k-\omega$ and Roe Riemann solver.

Figure 14 shows the CFD solution of the mesh shown in Figure 11. Because the compressor is shrouded, there are axial gaps between the metal that is stationary and the one that is rotating. The functionality of the cavity is to prevent the air from going back to previous blade rows as much as possible. The air enters the slot on the right and leaves the cavity from the slot on the left because of the pressure difference between these two slots. Figure 14 (left) and Figure 14 (right) show the contour plots of the static pressure and turbulent kinetic energy. The streamlines are plotted to demonstrate the flow pattern in the cavity.
Figure 15. Top-left: Sectional view of the hybrid mesh. Bottom-left: Close-up view of the hybrid mesh around the lip and the nose. Top-right: Static pressure at the meridional plane. Bottom-right: Static pressure at the exit of the intake.

Figure 16. Top-left: CDT of Cavity-1. Bottom-left: Close-up view of the CDT of Cavity-1 around the seals. Top-right: CDT of Cavity-2. Bottom-left: Close-up view of the CDT around the seal clearance.

Figure 15 shows the analysis of an aspirated intake of an aero-engine [38]. The figure on the top-left shows a section view of the mesh. Close-up views of the mesh around the lip and the nose are shown in the bottom-left. The figures on the right column show the static pressure distribution on the meridional plane and close to the intake exit when the intake is operating at cross-wind conditions. The blue region shows the flow separation on the lower lip and the separation propagating through at the intake exit, the position of which is the inlet of the fan blade.

8.5. Mesh Statistics

The CDT of the 2D and 3D geometries for the industrial cases is shown in Figures 16 and 17, respectively. The figures on the top row of Figure 16 shows the CDT of Cavity-1 and Cavity-2. The figures below them show a close-up view of the CDTs. It can be observed that for regions with narrow gaps, a point on the boundary only needs to walk through a couple of triangles to hit the boundary along its marching vector and its MMD can be computed rather accurately. For the 3D geometries, same conclusions can also be drawn. Table I summarizes the statistics in approximating the MMD. It shows the number of $d$-simplexes in the CDT, the minimum, maximum and mean number of $d$-simplexes that are visited in the jump-and-walk. The standard deviation is also provided to show the amount of variation of visited $d$-simplexes. It is interesting to see that the standard deviation is close to or even larger than the mean value. This indicates that the number of $d$-simplexes...
has a low central tendency towards the mean value. Finally, the wall-clock time to approximate the MMD is shown. The computational cost increases with the number of $d$-simplex in the CDT. Furthermore, by comparing the data between 3D cavity and NGV, it can been seen that the number of $d$-simplexes in 3D cavity is roughly 4 times larger than that of NGV, but the time cost is 7 times larger. This indicates that the time cost also depends on the topology of the geometry.

The number of elements in the meshes and the total running time is summarized in Tables II and III, respectively. The time cost in optimizing the mesh is not included, because the final mesh can be optimized by a third-party software (e.g., Mesquite). The number of elements in the boundary layer meshes is generally twice the number of isotropic elements. This is typical for internal flow computations. The exception is the case of intake. This is due to that the existence of far-field boundaries and the walls that are far away from each other. In terms of the wall-clock time, for 2D cases, the time cost for generating the boundary layer meshes is lower than creating isotropic elements; however, for 3D cases, the time cost in generating the boundary layer meshes is increased considerably and is more than twice the time cost in generating the isotropic tetrahedra. Same trend can be observed for the time cost of approximating MMD. Its time cost in 3D is around twice...
Table III. Overall timing statistics.

| Model       | MMD [s] | B. layer [s] | Isotropic mesh [s] |
|-------------|---------|--------------|--------------------|
| Cavity-1    | 0.24    | 0.94         | 1.35               |
| Cavity-2    | 0.12    | 0.13         | 1.48               |
| NGV         | 7.80    | 13.90        | 4.34               |
| 3D cavity   | 52.10   | 70.70        | 25.20              |
| Intake      | 4.20    | 13.13        | 13.20              |

Figure 18. Histogram of minimum dihedral angles of the isotropic tetrahedra for 3D cavity, Intake and NGV, respectively.

the time cost in generating isotropic tetrahedra. In the case of intake, because only a subset of the boundaries are walls, the time cost in approximating MMD and generating boundary layer meshes are lower. The quality of the isotropic elements in 3D is shown in Figure 18. The mesh quality is assessed by the minimum dihedral angle of a tetrahedron. The minimum dihedral angle for the 3D cavity and intake is larger than 15º. In the case of NGV, because the existence of the small input angle in the geometry, the minimum dihedral angle is downgraded to around 10º.

9. CONCLUDING REMARKS AND FUTURE WORK

Viscous flow computations are widely used in industries. Because of the geometrical complexity of industrial models, a prismatic meshing tool is desired. However, implementing a robust prismatic meshing tool is not trivial. One difficulty is to implement an extra data structure (e.g. quadtree in 2D and octree in 3D) to handle boundary layer mesh collisions. This paper aims to overcome this difficulty and has presented a generalized hybrid meshing method that solely relies on CDTs and does not require extra data structures. Because constructing a CDT is a pre-requisite of many Delaunay-based tetrahedral meshing method, the proposed method is a simple extension of a Delaunay-based tetrahedral mesh generator, and it greatly simplifies the implementation of a hybrid mesh generator. The capability of the proposed method is demonstrated on complex industrial cases with a particular focus on internal flow applications.

Because the focus of this paper is on removal of boundary layer mesh collisions, other challenging issues in hybrid meshing are not covered in detail. One major limitation is the handling singular points, ridges and corners. For internal flow applications, these geometrical features are avoided as much as possible for mechanical reasons, but for external flow computations, these geometrical features are present, and the current method has to be extended to handle these features in order to generate good quality boundary layer meshes.
ACKNOWLEDGEMENTS
The authors gratefully acknowledge Rolls-Royce plc for funding this work and granting permission for its publication. This work was performed under TSB for SILOET WP 3.1, and the first author was funded by the EPSRC Dorothy Hodgkin Postgraduate Award. The authors are also grateful to anonymous reviewers for their advice that has improved this manuscript considerably.

REFERENCES
1. Aubry R, Lohner R. Generation of viscous grids at ridges and corners. *International Journal for Numerical Methods in Engineering* 2009; 77:1247–1289.
2. Ito Y, Nakahashi K. Improvements in the reliability and quality of unstructured hybrid mesh generation. *International Journal for Numerical Methods in Engineering* 2004; 45:79–108.
3. Lohner R, juan C. Generation of non-isotropic unstructured grids via directional enrichment. *Internal Journal for Numerical Methods in Engineering* 2000; 49:219–232.
4. Lohner R. Matching semi-structured and unstructured grids for Navier Stokes calculations, 1993. AIAA-93-3348-CP.
5. Pirzadeh S. Unstructured viscous grid generation by the advancing-layers method. *AIAA Journal* 1993; 32(8):1735–1737.
6. Hassan O, Morgan K, Probert EJ, Peraire J. Unstructured tetrahedral mesh generation for three-dimensional viscous flows. *International Journal for Numerical Methods in Engineering* 1996; 39(4):549–567.
7. Merriam ML. An efficient advancing front algorithm for Delaunay triangulation, *AIAA-91-0792*, 1991.
8. Marcum DL, Weatherill NP. Unstructured grid generation using iterative point insertion and local connection. *AIAA Journal* 1994; 33(9):94–1926.
9. Marcum DL. Generation of unstructured grids for viscous flow applications. *AIAA* 1995:95–0212.
10. Marcum DL. Efficient generation of high-quality unstructured surface and volume grids. *Engineering with Computers* 2001; 17:211–233.
11. Sharov D, Nakahashi K. Hybrid prismatic/tetrahedral grid generation for viscous flow applications. *AIAA Journal* 1998; 36(2):157–162.
12. Kallinderis Y, Khawaja A, McMorris H. Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries. *AIAA journal* 1996; 34(2):291–298.
13. Khawaja A, Kallinderis Y. Hybrid grid generation for turbomachinery and aerospace applications. *International Journal for Numerical Methods in Engineering* 2000; 49:145–166.
14. Garimella RV, Shephard MS. Boundary layer mesh generation for viscous flow simulation. *International Journal for Numerical Methods in Engineering* 2000; 49:193–218.
15. Ito Y, Shihi AM, Bharat K, Soni ea. Multiple marching direction approach to generate high-quality hybrid meshes. *AIAA journal* 2007; 45(1):162–167.
16. Si H. Constrained Delaunay tetrahedra mesh generation and refinement. *Finite Elements in Analysis and Design* 2010; 46:33–46.
17. Shewchuk JR. Tetrahedra mesh generation by Delaunay refinement. *Proceedings of the 14th Annual Symposium on Computational Geometry*, Minneapolis, USA, 1998; 86–95.
18. Shewchuk J. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. *Applied Computational Geometry Towards Geometric Engineering* 1996; 1148:203–222.
19. Chew LP. Constrained Delaunay triangulation. *Algorithmica* 1989; 4:97–108.
20. Anglada MV. An improved algorithm for constructing restricted Delaunay triangulations. *Computers & Graphics* 1997; 21:215–223.
21. Shewchuk JR. Constrained Delaunay tetrahedralizations and provably good boundary recovery. *Proceedings of 11th International Meshing Roundtable*, Ithaca, USA, 2002; 193–204.
22. Si H, Kalus G. 3D boundary recovery by constrained Delaunay tetrahedralization. *International Journal for Numerical Methods in Engineering* 2011; 85(11):1341–1364.
23. Shewchuk JR. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry* 1997; 18:305–365.
24. Edelsbrunner H, Mucke E. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics* 1990; 9:66–104.
25. Shewchuk JR. Sweep algorithms for constructing higher-dimensional constrained Delaunay triangulations. *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, Hong Kong, China, 2000; 350–359.
26. Teten SH. A delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software* 2015; 41(2):1–36.
27. Kallinderis Y, Ward S. Prismatic grid generation for three-dimensional complex geometries. *AIAA Journal* 1993; 31(10):1850–1856.
28. Connell SD, Braaten E. Semistructured mesh generation for three-dimensional Navier Stokes calculations. *AIAA Journal* 1995; 33(6):1017–1024.
29. Green PJ, Sibson R. Computing Dirichlet tessellations in the plane. *Computer Journal* 1978; 21:168–173.
30. Bowyer A. Computing Dirichlet tessellations. *The Computer Journal* 1981; 24:162–166.

© 2016 The Authors. *International Journal for Numerical Methods in Engineering*. Published by John Wiley & Sons Ltd. DOI: 10.1002/nme
31. Si H. Three dimensional boundary conforming Delaunay mesh generation. Ph.D. Thesis, TU Berlin, 2008.
32. Shewchuk JR. Delaunay refinement algorithms for triangular mesh generation. Technical Report, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 2001.
33. Mucke E, Saias I, Zhu B. Fast randomize point location without preprocessing in two- and three-dimensional Delaunay triangulations. Proceedings of the 12th Annual Symposium on Computational geometry, Philadelphia, USA, 1996; 274–283.
34. Wang F. Whole aero-engine meshing and CFD simulation. Ph.D. Thesis, Imperial College London, 2013.
35. Available from: http://www.cs.cmu.edu/~quake/triangle.html [Accessed on April 2012].
36. Freitag LA, Ollivier-Gooch C. Tetrahedral mesh improvement using swapping and smoothing. Internal Journal for Numerical Methods in Engineering 1997; 40:3979–4002.
37. Brewer M, Folwell N, Knupp P, Leurent T, Melander D. The mesquite mesh quality improvement toolkit. Proceedings of the 12th International Meshing Roundtable, Santa Fe, USA, 2003; 239–250.
38. Carnevale M, Wang F, Green JS, di Mare L. Lip stall suppression in powered intakes. Journal of Propulsion and Power 2016; 32(1):161–170.