Non-Regular Background Mesh based Hex Meshing

Zhihao Zheng *†‡ Liang Dai *‡ Shuming Gao *§

Abstract

Thanks to their superior numerical properties, conforming hex meshes have become a popular choice in the context of simulation. However, due to the global structural constraints, generating hex meshes has been a subject of extensive study, giving rise to numerous fascinating and challenging associated problems. In light of this, this paper introduces an innovative approach based on non-regular background mesh, aimed at automatically generating high-quality adaptive hex meshes for CAD models with arbitrary complexity. The algorithm initially generates a non-regular background mesh for the given solid model through several steps involving shape analysis, quad mesh sweeping and templates-based mesh refinement. Subsequently, to convert this background mesh into a conformal hex mesh, the algorithm identifies the globally optimized core hex mesh via a graph-cut algorithm. To further improve the quality of mesh, the algorithm conducts topology improvement, coupled with geometric optimization guided by quality metrics applied to boundary layers of core mesh. Experimental results showed the effectiveness of the proposed algorithm.

Keywords: hex meshing; grid-based hex meshing; non-regular background mesh; graph-cut optimization.

1 Introduction

One of the core techniques of digital design and simulation is the finite element analysis(FEA). The generation of meshes, discretizing the domain into a finite collection of elements, is the initial step of the procedure of the FEA. As for 3D models, tet meshes and hex meshes are typically used mesh types. The common belief that hex meshes yield better accuracy for a given computational cost has motivated the researchers to develop algorithm generating and processing hex meshes for more than 30 years [1, 2]. Despite the extensive research, there is still no automatic algorithm which can reliably and effectively generate a high-quality hexahedral mesh conforming to the target geometry, due to globally interdependent structural constraints inherent to the hex mesh.

The grid-based methods project the boundary of a carefully constructed lattices onto a target geometry [3]. Due to their robustness, the grid-based methods are the only automatic methods capable of successively hex-meshing any input shape, and are the only automatic methods currently implemented in professional software [4, 5]. However, since the initial mesh connectivity is restricted by the underlying grid, the preservation of features is both geometrically and topologically challenging, when we project the initial mesh to the target geometry while maintaining the inversion-free property of the hex-mesh. Therefore, by relaxing the constraints of the initial mesh type, we proposed a hex meshing algorithm involving construction and editing a target geometry likely mesh, which we call as non-regular background mesh.

Goal. Given an input solid model with feature tags and user specified mesh edge length l_e, our goal is to automatically generate a conformal pure hex mesh whose density is controlled by l_e, while guaranteeing:(1) preservation of feature curves, and (2) a positive scaled Jacobian for each element.

Contributions. To achieve the goal described above, we made the following contributions in this paper:

1. We propose a non-regular background mesh generation method for roughly approximating the target geometry. The relaxation of the constraints on the initial background mesh connectivity frees up more degree of freedom for mesh improvement for subsequent steps.
2. We propose a graph-cut based core mesh extraction method for converting a shape-likely mesh to a conformal hex mesh. By formulating the determination of the final mesh boundary as a graph cut computation, it is more controllable to obtain a final core mesh balancing the geometric fidelity and element quality over boundary layer.

2 Related works

There are many kinds of approaches to hexahedral mesh generation. We restrict our survey to the most important techniques and most relevant techniques for our method, and we refer the interested readers to [1] for an overview of hex mesh generation and processing related techniques.

2.1 Grid-based hex-meshing

Grid-based methods generate hex mesh in an editing

*State Key Lab of CAD & CG, Zhejiang University.

†School of Data Science and Artficial Intelligence, Wenzhou University of Technology.

‡First author and second author contributed equally to this work.

§Corresponding author.

valid mesh style, and can be applied to any complex models. A typical grid-based method first generates a background hex-mesh roughly approximating the model by voxelizing the interior of the model or a domain completely enclosing the model. For the case where the background mesh is slightly larger than the model, some elements of the regular lattices have to be cut from the background mesh. Finally, the boundary vertices of the background mesh are project onto the input boundary of the input model.

The essence of grid-based methods lies in the transformation of the initial regular lattices into a high-quality mesh that closely approximates the input solid model. This transformation is carried out while ensuring that the final hex mesh accurately captures the model's features without introducing any inverted or non-hexahedral elements. In order to achieve this conversion, the academic community has been conducting continuous research [3, 6–9] for over two decades, and the resulting algorithm can generate high-quality hexahedral meshes.

To generate hexahedral mesh preserving features, Gao et al. [10] proposed an Octree-based hexahedral meshing algorithm. The algorithm first converts an adaptive octree into a pure hexahedral mesh by computing its geometric dual and splitting/merging the non-hex cells. Then, the sharp features of the model are then topologically mapped to the boundary of hex mesh. Finally, a locally injective map based deformation is applied to hex mesh to make mesh match the input geometry.

Converting adaptive octree into a pure hexahedral mesh is usually one of the very important steps in grid-based hex-meshing algorithms. To enlarge the space of hexameshable adaptive grids, Pitzalis et al. [11] proposed an algorithm transforming a generic adaptively refined grid into a modified grid that is suitable for conforming hexahedral meshing. By choosing the proper set of unknowns, all compatibility conditions are formulated as linear constraints in an integer programming problem. However, the algorithm assume a regular grid with prescribed binary refinement as input and it seems that there is no guarantee that the input of the algorithm can be extended to non-regular meshes rather than grids.

The grid-based method is a universal meshing algorithm that can ensure high overall quality of mesh. However, the grid-based methods have the following issues: (1) It is difficult to ensure that all geometric constraints are considered, so the generated mesh may lose geometric information; (2) There is still a lack of robust and efficient boundary mesh processing algorithms to ensure the boundary mesh quality and conformality of complex models. However, the grid method has irreplaceable advantages in terms of efficiency, automation and versatility, and is a popular direction in the field of hexahedral meshing.

2.2 Adaptive hex-meshing

Dense meshes can ensure geometric fidelity and computational accuracy, but require expensive computation cost and significant storage. In order to ensure the good geometric approximation and tight error bounds with least number of elements, the adaptive hex meshes are generated by refining mesh in interested local domains.

The general procedure for adaptive refinement technology involves two main steps: initially identifying the regions that require refinement and subsequently applying local refinement to those specific regions.

Two prevalent categories of methods are commonly employed to identify the regions in need of refinement. The first category determines these regions based on geometric features [6, 12–14], while the second category relies on the posterior error [15–17].

Regarding mesh refinement algorithms, two type of widely used approaches are prevalent: the first type refines mesh based on the templates[18–21], and the second type employs dual operations to guide the refinement process[22, 23].

The templates we used in §4.3.2 originate from [18, 19].

2.3 Integer-grid map based hex-meshing

Integer-grid map based hex-meshing algorithms are a promising type of methods, due to the advantages of being formalized hex-meshing as a map optimization problem. The central idea of integer-grid maps is to embed an n-dimensional shape into an n-dimensional voxel grid such that the inverse map deforms the set of covered voexls into a shape-aligned hexahedral mesh. The frame field based methods [24–26] and polcube-based methods[27, 28] are both integer-grid map based algorithms.

An integer-grid map approach usually consists of five steps sequentially generating (i) feature aligned frame field, (ii) seamless map, (iii) integer quantization, (iv) integer-grid map, and (v) hexahedral mesh. There are provably robust algorithms exist for step(iii)[29] and step(v)[30], while steps(ii) and (iv) remain fragile. To improve the robustness of frame field based method, Liu et al. [31] proposed an algorithm to convert a given frame field into a locally meshable one based on a theory of locally meshable frame fields.

3 Approach overview

To achieve the goal described in §1, we discretize the input model to a hex mesh by converting an initial non-regular mesh into the hex mesh conforming to the input geometry. In order to ensure that this pipeline similar to grid-based algorithm can be effective, there are two issues must be addressed in general.

The first issue is how to construct the initial non-regular mesh so that it can be more easily converted to a high-quality hex mesh capturing the features of the input model. This issue is addressed by constructing an adaptive hex mesh roughly approximating the target geometry based on sweeping method. More specifically, the carefully constructed sweep face and carefully chosen sweep direction jointly contribute to a swept mesh that captures as much features of the target geometry as possible. And refining the mesh sub-domains intersecting with uncaptured features improves the mesh density at these sub-domains, and finally ensure the geometric fidelity.

The second issue is how to convert the non-regular mesh to a high-quality mesh capturing all the features of the input model. To address this issue, we extract a core mesh from the initial mesh based on graph-cut algorithm and then optimize the topology and geometry of the core mesh.

The input of the proposed method is a 3D model, and the output is a hex mesh. The following are the main steps of the proposed method:

1. Generation of the non-regular background mesh;
2. Generation of the core mesh;
3. Topological and geometric optimization of the core mesh.

4 Generation of the non-regular background mesh

The first step of our algorithm is to construct a non-regular mesh roughly approximating target geometry. The goal of this step is to generate a swept mesh capturing features of the given model as much as possible with local domains refined.

To achieve this goal, there are two critical issues to be addressed. The first issue is how to ensure the constructed background mesh capturing the features, and the second issue is how to refine the local domains with any topological arrangement.

To address the first issue, we first extract a principal direction based on shape analysis. Then based on this principal direction, we generate an optimized principal face carrying the features information by projection. Finally, we obtain a background mesh capturing features of model by sweeping quad mesh of principal face with geometric constraints. As for the second issue, we make the transition between the domains to be refined and remaining domains by inserting templates into transition layers, which is generated by applying pillowing operations along the interfaces between domains to be refined and remaining domains. The templates inserted to the domains to be refined and transition layers are determined based on the property of the interfaces between the domains to be refined and the remaining domains.

According to the above description, the non-regular background mesh generation algorithm mainly consists of three steps, as shown in Figure 1:

1. Generation of the optimized principal face for sweeping. We first extract a principal direction by shape analysis. Then, we generate an optimized principal face based on the extracted the principal direction.
2. Generation of the initial background mesh by sweeping. We sweep the quad mesh of principal face along the principal direction under geometric constraints to obtain an initial background mesh.
3. Local refinement of initial mesh. We determine intersected domains and refine them locally with corresponding templates.

4.1 Generation of the optimized principal face for sweeping

To generate a sweeping-based background mesh capturing features as much as possible, the principal face should carry feature information as much as possible. The basic idea for generating such a principal face is to generate an initial principal face by projecting the 3D features to 2D plane along a principal direction of model, followed by an optimization of this principal face. Therefore, we obtain the principal face by first extracting the principal direction of model, and then generating and optimizing the principal face.

4.1.1 Extraction of principal direction of model

Since the background mesh is obtained by sweeping the principal face along the principal direction, the main goal of the principal direction extraction is to select the projection direction that can produce a background mesh capturing most geometric information. According to this goal, the determination algorithm of principal direction of model is as follows:

1. Extraction of candidate projection directions. Considering that many CAD models are formed by extrusion, we adopt an intuitive approach, that is, the principal directions of all boundary surfaces are added to the set of candidate projection directions. For the planes, we select their normal vectors as their principal directions. For each of the remain-

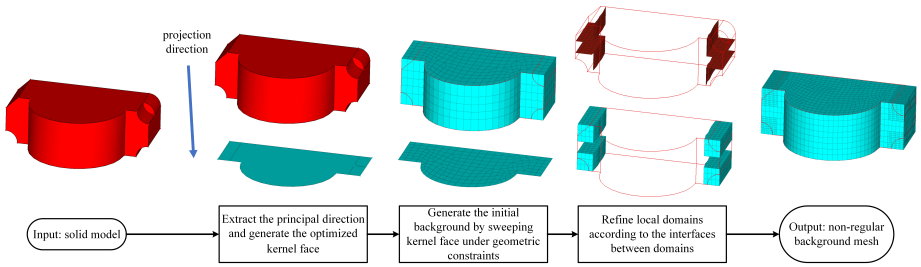


Figure 1: The generation pipeline of non-regular background mesh.

ing surfaces, we take 10×10 sampling points uniformly along the surface, and select the average of the normal vectors on these points as the principal direction of the surface.

2. Principal direction determination based on geometry capture metric. We use a *geometry capture metric* to characterize how well a hexahedral mesh captures the target geometry. For each candidate direction, we generate a corresponding non-regular mesh by the subsequent steps in this section. Then we compute a geometry capture metric for each generated hex mesh. Finally, we select the projection direction as principal direction, if the geometry capture metric of corresponding mesh is highest among all generated mesh.

To approximate the input model well, the background mesh should capture the boundary surfaces as much as possible and wrap the target object as much as possible. Therefore, we use a linear combination of the *boundary surface capture metric* and the *volume capture metric* to represent the geometry capture metric of a hex mesh. The *boundary surface capture metric* and the *volume capture metric* are as follows:

1. *Boundary surface capture metric.* The boundary surface capture metric is positively related to the area of the boundary surface captured by the hex mesh. To reduce the computation cost, we will not actually construct the mesh, and we use a rough estimation method. Considering a swept hex mesh capturing the boundary surfaces parallel or perpendicular to the sweeping direction $\mathbf{d}$, we regard all these surfaces as the surfaces captured by this hex mesh. Specifically, for hex mesh generated based on direction $\mathbf{d}$, its boundary surface capture metric $C^{\mathbf{d}}_{surface}$ is formulated as follows:

$$C^{\mathbf{d}}_{surface} = \frac{\mathcal{A}^{\mathbf{d}}_{captured}}{\mathcal{A}_{boundary}}, \tag{4.1}$$

where $\mathcal{A}^{\mathbf{d}}_{captured}$ indicates the total area of captured the boundary surfaces, and $\mathcal{A}_{boundary}$ indicates the total area of the boundary surface.

2. *Volume capture metric.* Again, we introduce a rough evaluation method for the final generated mesh volume to reduce the computation cost. For a given direction $\mathbf{d}$, we define the 2D plane perpendicular to $\mathbf{d}$ first intersectiong with target geometry as start plane, while the first leaving plane as end plane. We extract the 2D projection surface by projecting the target geometry along $\mathbf{d}$ to the start plane, and then extrude the extracted surface along $\mathbf{d}$ to end plane to obtain a body. The volume of this body is used to represent the captured volume $\mathcal{V}^{\mathbf{d}}_{sweep}$ roughly. With the captured volume defined, the volume capture metric $C^{\mathbf{d}}_{volume}$ for hex mesh generated based on direction $\mathbf{d}$ is formulated as follows:

$$C^{\mathbf{d}}_{volume} = 1 - \frac{|\mathcal{V}^{\mathbf{d}}_{sweep} - \mathcal{V}_{solid}|}{\mathcal{V}_{solid}}, \tag{4.2}$$

where $\mathcal{V}_{solid}$ represents the volume of the input model.

With the boundary surface capture metric and volume capture metric defined, we formulate the geometry capture metric of a swept mesh based on the direction $\mathbf{d}$ (denoted as $C^{\mathbf{d}}_{geometry}$) as follows:

$$C^{\mathbf{d}}_{geometry} = \lambda \times C^{\mathbf{d}}_{surface} + (1-\lambda) \times C^{\mathbf{d}}_{volume}, \tag{4.3}$$

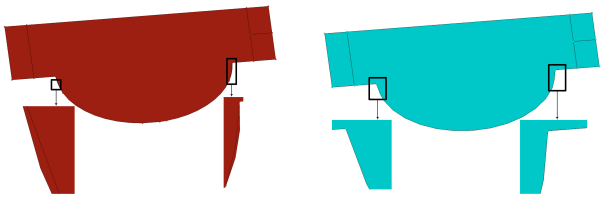
where scalar λ is used to balance the contributions of the two metrics in the geometry capture capability function well. According to our experiments, the boundary surface capture metric deserves higher weight, and it is suitable to use the following formula to determine the λ:

$$\lambda = \begin{cases} 1 & C^{\mathbf{d}}_{surface} \le 0.85 \\ 1.85 - C^{\mathbf{d}}_{surface} & C^{\mathbf{d}}_{surface} > 0.85 \end{cases} \tag{4.4}$$

4.1.2 Generation and optimization of model principal face

In this step, we obtain the principal face of the model by projecting the model along the extracted principal

direction to the start plane. To ensure the principal face carrying the information of boundary features, the feature curves on boundary are projected to the start plane, as shown in Figure 2a. All the projected curves cutting the projected face and form the alignment constraints for the quad mesh generation of principal face. Due to alignment constraints of projected curves,



(a) (b)

Figure 2: Principal faces generated without constraints relaxation(a) and with constraints relaxation(b).

the principal face exhibits an overly complex topology, which will lead to poor quality of the generated quad mesh, or even failure of quad mesh generation. For example, the principal face shown in Figure 2a can not be quad meshed due to the isolated curve. Therefore, some constraints have to be relaxed. More specifically, for each feature node, we first determine whether this node is over-constrained according to the angle between the its adjacent curves as well as the its valence, and then we evenly select and suppress the interior curves adjacent to the feature node. The suppressed curves are limited in the interior curves ensuring that the shape of principal face will not be destroyed in the relaxation process. As shown in Figure 2b, the isolated curve in Figure 2a was recoginized and suppressed. According to our experiments, we consider a feature node to be over-constrained, if the number of its adjacent feature curves is at least 10, or there are two its adjacent feature curves between which the angle is less than 5°.

4.2 Generation of the initial background mesh by sweeping

With carefully selected principal direction and the quad meshed principal face carrying features information in hand, we generate an initial background mesh based on sweeping method to approximate the target geometry.

As the foundation of the swept mesh, the quality of the principal face's quad mesh determines the quality of the initial background mesh. However, there may still be complex topology in principal face even after the relaxation of constraints, and the quad mesh of the principal face generate may contain some low-quality elements. Therefore, a quad mesh optimization is required before sweeping. We optimize the quad mesh rudely by applying the quad collapse operations to the elements with negative Jacobian. For each element to be collapsed, we prefer the pair of diagonally opposing vertices to be merged, if neither of this pair of vertices is constrained to the feature.

Though the principal face carries some features information, except the features distributing along the principal direction. As shown in Figure 3a, the hex mesh is not aligned with the boundary features along the principal direction, which will lead to background mesh with low quality in the subsequent step. Therefore, it is necessary to add extra geometric constraints along the principal direction during the sweeping procedure. Since the misalignment of the mesh is caused by the fixed sweeping step length, we vary the sweeping step length according to the distrbution of the features. More specifically, a sweeping algorithm with geometric constraints is as follows:

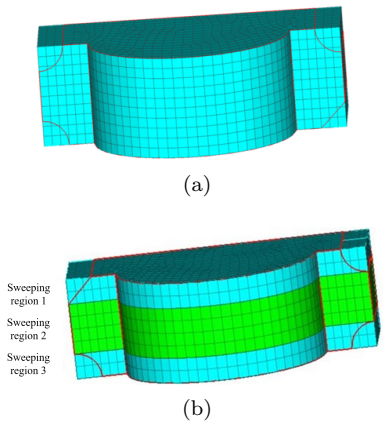
1. Add all boundary feature planar curves whose planes are parallel to the principal face into a set, denoted as S_p.
2. Decompose the sweep path into multiple sweep paths based on S_p. For each element in S_p, we construct a plane that passes through it and is parallel to the principal face. Then we cut the sweep path with the constructed planes into multiple sweep paths.
3. Determine the sweep step for each sweep path based on target mesh edge length l_e.
4. Generate background mesh by sweeping the quad mesh of the principal face along the multiple sweep paths with corresponding step length.

As shown in Figure 3b, with the varied sweep step, the swept hex mesh captures the boundary target geometry better than the previous one.

4.3 Local refinement of initial background mesh

The initial background mesh suffers the same issue as the grid-based mesh, that is, the quality of the elements on boundary layer is not guaranteed. Therefore, we apply the adaptive refinement technique to the boundary layers of initial background mesh, to balance the quality pursuits and corresponding costs of result mesh. The key to refine local domains, which may contain complex topological arrangement, without affecting other domains is applying the pillowing operations along the interfaces between domains, followed by templates insertion. The specific procedure of the local domains refinement is as follows:

1. Determination of the domains to be refined.
2. Classification of interfaces between domains.
3. Refinement of mesh based on classification of inter-



(a)

(b)

Figure 3: The swept mesh generated with one sweeping step(a) and multiple sweeping steps(b). By introducing the geometric constraints along the principal direction, the sweeping path are divided, and the sweeping steps of different sweeping regions varies. With varied sweeping steps, the uncaptured features in (a) are captured by mesh in (b).

faces.

4.3.1 Determination of the domains to be refined

Since the initial background mesh has captured some boundary features with quality-assured elements for finite element analysis, the mesh local refinement is required by the uncaptured features.

We use an intuitive method based on bounding box to extract the local domains of mesh to be refined. For each uncaptured boundary face, we first calculate its bounding box and label all the elements in the interior of bounding box as elements to be refined, as shown in Figure 4. Then, we cluster the elements to be refined into multiple connected domains by iteratively inserting elements sharing a common face into a set. Finally, we simplify the initial background mesh by removing elements which are neither elements to be refined nor located in the interior of the target geometry. The connected domains after simplified are target domains to be refined.

4.3.2 Classification of interfaces between domains

To obtain a high-quality hex mesh by transitioning the refined domains to the remaining domains in a templates' insertion fashion, it is vital to determine optimum templates for insertion according to the interfaces between domains. It is an often case that some templates are only suitable for layers with specific property.

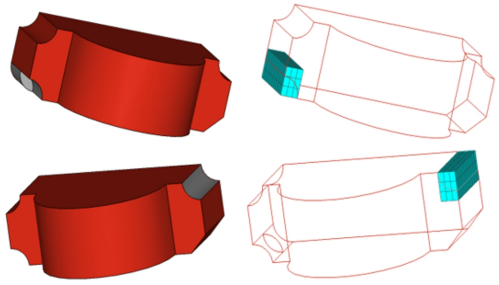


Figure 4: The elements to be refined are determined based on the uncaptured boundary faces.

Therefore, the key for classification of the interfaces between domains is to check whether the interface meets the condition for using preferred template.

Templates. Before describing the classification algorithm of interfaces between domains, we first introduce the templates used in this paper. For elements in the interior of domains to be refined, we use a 2-refinement template(5a) and a 3-refinement template(5b).The former refines an element into $8(2\times2\times2)$ elements, while the latter refines an element into $27(3\times3\times3)$ elements. Due to the smoother mesh edge length variation, the mesh refined by 2-refinement template exhibits a higher quality than the mesh refined by 3-refinement. Therefore, we prefer the 2-refinement template.

To transition the domains refined by 2-refinement template to unrefined domains, we use a composite template (see Figure 6b), which is generated based on the basic template shown in Figure 6a. There is a polyline containing three edges in the basic template, and we label two boundary patches of template adjacent to this polyline as $\mathcal{M}$-patches. To obtain composite template, we perform mirror image transformation to the basic template along the $\mathcal{M}$-patches in arbitrary order. It is obvious that this composite transition template has to be applied to a block consisting of four elements in the pillowed layer sharing an edge. Therefore, before using the composite transition template in pillowed layer, or further 2-refinement template for domains to be refined, it is necessary to ensure that the interface should meet the 2×2 *couplable condition*, that the whole interface can be divided into 2×2 tiny blocks. In composite template, the mirror transformation center polyline contains more mesh edges than other parallel polylines, which means a relatively limited geometric optimization space. Further, the 2×2 block in the interfaces should be as flat as possible to obtain final high-quality mesh.

To transition the domains refined by 3-refinement

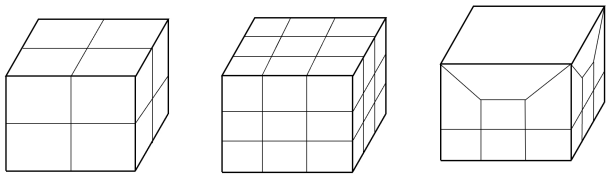


Figure 5: (a) 2-refinement template;(b) 3-refinement template;(c)3-transition template.

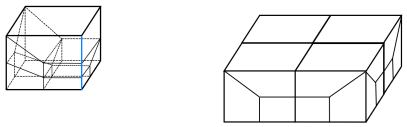


Figure 6: A 2-transition template derived from a basic template. The composite template in (b) is generated by performing mirror transformation to the basic template in (a) along the boundary patches adjacent to the cyan polyline.

to unrefined domains, we use the transition template shown in Figure 5c. Since this transition template is applied to individual transition element and does not require the quads on the interface to be coupled, it can be applied to any interface.

Interface classification. Due to the superior property of 2-refinement template, we try to optimize the interface to meet the 2×2 couplable condition. With this goal, the classification of interfaces between domains by the following procedure:

1. Extract the initial interfaces. We first extract the boundary faces of domains to be refined. And these extracted faces in the interior of the background mesh can be clustered into multiple connected quad sets. Each of these quad sets is an initial interface.
2. Check each domain to be refined whether its adjacent interface meets the 2×2 couplable condition. We only consider the case where the domain to be refined is adjacent to only one interface, since it is trivial to generalize its treatment to domain with multiple adjacent interfaces. We check each domains based on segmentation as follows:
   (a) Extract initial cutting edges. To make each 2×2 tiny block as flat as possible, we first regard the mesh edges on interfaces with dihedral less than θ_{lower} as boundary edges of tiny blocks, and label them as cutting edges.
   (b) Segment the interface into regions based on initial cutting edges. From each end vertices of cutting polylines, we recursively label fresh edge on interface as cutting edge until reaching another end vertices of cutting polyline or boundary of the interface. For each interface, the cutting polylines segment it into multiple regions.
   (c) Check 2×2 couplable condition based on segmentation. For each region, we randomly select a boundary vertex adjacent only one element in this region as start vertex. From the start vertex, we pave the region with 2×2 tiny blocks from its start vertex. If each region can be paved in this way, then this interface meets the 2×2 couplable condition and this interface is divided into classes suitable for 2-refinement template. Otherwise, the interface is divided into classes unsuitable for 2-refinemnt template.

According to our experiments, the $\theta_{lower} = 150°$ behaves well.

4.3.3 Refinement of mesh based on classification of interfaces

After classifying the interfaces, we refine the local domains without affecting remaining domains.

To support the usage of transition elements, we first apply the pillowing operations along the interfaces between domains.

For the domains only adjacent to the interfaces meeting 2×2 couplable condition, we first refine these domains with 2-refinement template, and then refine the pillowed layer connecting these refined domains and their adjacent domains with the composite template.

For the remaining domains to be refined, we first refine these domains by 3-refinement template, and than refine the pillowed layers connecting these refined domains and their adjacent domains with 3-transition template.

For the models tested in this paper, we only refined it once to achieve our expected results. Since the 2×2 couplable condition would not be broken after each refinement, one can refine local domains repeatedly if necessary.

5 Generation of the core mesh

Since there are still some features uncaptured as described in §4.2, the non-regular background mesh generated in §4 should be converted into a core mesh by further processing to capture all features of target geometry. For a target geometry, we call a mesh obtained by deleting elements of background mesh as the core mesh of target geometry, if this sub-mesh approximates

the target geometry with high geometric fidelity, as well as a 1:1 corresponding between the sequences of boundary mesh entities and the geometry boundary features of target geometry.

With the definition of core mesh, there are two critical issues that need to be addressed for the conversion from non-regular background mesh to core mesh:

1. In order to achieve a high degree of geometric matching between the core mesh and the boundary of target geometry, how to determine the boundary of core mesh.
2. In order to achieve the efficient capture of the boundary geometry features, how to establish the corresponding relationship between the entities of core mesh and the features of target geometry.

For the first issue, we formulate the determination of boundary of core mesh as a graph cut optimization, based on which we can balance the geometry fidelity and mesh quality over boundary layers. As for the second issue, to establish the corresponding relationship between feature nodes and mesh boundary vertices, as well as the corresponding relationship between feature curves and mesh boundary edges, we utilize the matching degree between corresponding feature entities and mesh entities. The relationship between feature faces and boundary mesh faces is a direct corollary of the previous determined relationship based on mesh boundary segmentation.

According to the description above, the main steps of the core mesh generation algorithm are as follows:

1. Determination the boundary of core mesh based on graph-cut algorithm. We then complete the boundary surface of core mesh via a graph-cut based optimization framework.
2. Establishment of correspondence between core mesh and features. The corresponding relationship between the mesh vertices and feature nodes as well as the relationship between mesh edges and feature curves are established based on searching algorithm. The corresponding relationship between the mesh faces and feature surfaces will be a direct corollary of previous.

5.1 Determination the boundary of the core mesh based on graph-cut algorithm

The goal of extracting core mesh from the non-regular background mesh is to obtain a mesh, whose boundary fits target geometry well. Actually, core mesh extraction can be translated to the determination of a 2D-manifold, which segmenting the non-regular background mesh into the core mesh and the remaining part. In this paper, we reformulate the determination task of this 2D-manifold as a graph-cut optimization problem to find a balance between the geometric fidelity and quality over the boundary layer of the core mesh globally.

However, considering that the carefully constructed principal face ensures the swept background mesh capturing some features of target geometry, we reduce the scale of the graph-cut optimization via a divide-and-conquer approach. According to § 4.3.1, the local domains of non-regular background mesh are determined based on the uncaptured features. It's a direct corollary that the remaining domains of non-regular background mesh belong to the core mesh. Therefore, all we have to do is extract the part belong to the core mesh from each refined domain of non-regular background mesh based on a graph-cut optimization.

Graph-cut optimization formulation. There are two criteria should be considered for the extraction of the 2D-mainifold from each refined domain. On the one hand, the extracted surface should be as close to the uncaptured surface as possible; on the other hand, the quality on the boundary layer of core mesh should be as high as possible, after projecting the extracted surface to the feature surface. We use fidelity metric and smoothness metric describing the two criteria.

Since it is an often case that the two metrics are competitive, we express the trade-off between the fidelity and smoothness as an energy optimization problem, in which each hex in the interior of refined domains must be assigned one of two labels. Two labels are L_{inner} and L_{outer}, representing the hex in the interior of core mesh and in the exterior of core mesh respectively. Searching optimal segmentation is searching for a segmentation S which minimizes the energy $E(S)$:

$$E(S) = \omega \sum_{h \in H} E_f(l_h) + \sum_{p,q \in H, p \bigcap q \in F} E_s(l_p, l_q). \tag{5.5}$$

The former term E_f, fidelity energy, describes the cost of assigning the label l_h to a hex, while the latter term E_s, smoothness energy, describes the cost of assigning the label l_p to hex p and the label l_q to hex q, if p and q share a common face. The constant ω is a user specified scalar to balance the contribution of the two metrics. In our paper, we set $\omega = 1$.

Fidelity term. The cost of assigning a hex h to a given label l_h is measured based on the signed distance function $D_{signed}(h)$ to the uncaptured feature surface:

$$E_f(l_h) = \left(\frac{1 - \frac{D_{signed}(h)}{D_m(l_h)}}{2}\right)^2. \tag{5.6}$$

The signed distance function is defined based on the barycenter of the hex, and the value of the function is negative if the barycenter is in the interior of the target

geometry. If $l_h = L_{outer}$ ($l_h = L_{inner}$, respectively), D_m is the largest (smallest, respectively) value of the signed distance function.

If the label and the barycenter location of hex indicate that they are in same side of uncaptured surface (for example, $l_h = L_{inner}$ while the barycenter of hex h is in the interior of the target geometry), the energy E_f decreases, as the distance between the hex and the boundary surface of target geometry increases; otherwise, the energy E_f increases, as the distance increases.

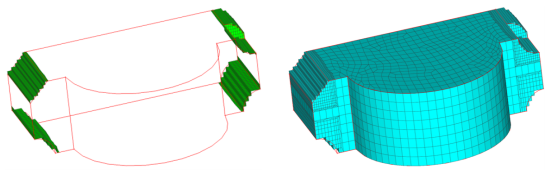
Smoothness term. The smoothness term is designed to maximize the quality of hex in the boundary layer of core mesh. We set the $E_s = 0$ for two hexes sharing a common face to 0, if they have same label. For hexes p, q sharing a common face assigned with different labels, without loss of generality, we assume $l_p = L_{inner}$ and $l_q = L_{outer}$ and we define the smoothness energy between these two hexes:

$$E_s = \begin{cases} (\frac{1-J_p}{2})^2 & J_p \geq 0 \\ \infty & J_p < 0 \end{cases}. \tag{5.7}$$

J_p is the Jacobian of the hex p, after the common face of p, q projected to the boundary face of the target geometry. To avoid the presence of hex with negative in the core mesh, we set the smoothness energy to ∞ if the hex labeled L_{inner} has the negative Jacobian. For the hex p with positive Jacobian, the smoothness gradually decreases to 0 as the Jacobian of p approachs 1.

Graph-cut optimization solver. To compute the best 2D-manifold that minimizes the Equation 5.5, we dualize the mesh of refined domain and adopt the graph cuts algorithms proposed by Boykov et al. [32, 33].

After determining part belong to the core mesh for each refined domain, we assemble them with the unrefined part of background mesh to obtain the core mesh. Figure 7a shows the segmentation surface computed by graph-cut algorithm, and Figure 7b shows the core mesh.



(a) (b)

Figure 7: (a)The segmentation surfaces computed based on graph-cut algorithm.(b)The core mesh obtained based on the segmentation surfaces.

5.2 Establishment of correspondence between core mesh and features

To ensure the features of target geometry captured by core mesh, we have to establish the correspondence between the feature nodes and boundary vertices of mesh, feature curves and boundary edges of mesh, as well as feature faces and boundary faces of mesh. We first establish the correspondence between the feature nodes and mesh vertices, as well as feature curves and mesh edges via searching algorithms respectively. Then, we directly establish the correspondence between feature surfaces and mesh faces according to the established corresponding relationships.

5.2.1 Establishment of correspondence between mesh vertices and feature nodes

We establish the correspondence between feature node and mesh vertices based on the distance between them and quality after relocation. For each feature node, we select the highest-quality vertex as the node's corresponding vertex from the vertices within a distance less than the target edge length, and relocate the selected vertex. Here, quality of vertex is the lowest Jacobian of adjacent hex to the vertex after relocating this candidate vertex to the feature node.

However, it is necessary to re-establish the corresponding relation, if the valence of feature node is larger than the valence of mesh vertex. Otherwise, different feature curves adjacent this feature node will be corresponded with same mesh polyline.

For a feature node to re-establish the corresponding relationship, we select vertex with highest quality from one-ring vertices of previous selected vertex, to establish the correspondence with the feature node. If valences of all neighbor vertices are still less than valence of feature node, we suppress the adjacent curves in ascending according to the cosine of the dihedral angle of feature curve.

5.2.2 Establishment of correspondence between mesh edges and feature curves

The establishment of correspondence between mesh edges and feature curve, is to determine a sequence of boundary mesh edges corresponding to the feature curve, whose end vertices correspond to the two end feature points of the feature curve. The quality of the correspondence is determined by the closeness between sequence of mesh edges and feature curves.

Determining a sequence of edges corresponding to a feature curve, is equivalent to finding shortest path on boundary of core mesh, whose end vertices correspond to the end feature node of the feature curve. Therefore, the key to find optimum corresponding sequence of

edges is the well definition of the closeness between mesh edges and target geometry curve. We define a combined measure m_{clo} that characterize the closeness between the mesh edges and target feature curve based on their direction fidelity and location fidelity, balanced by the parameter function $\alpha(i,j)$:

$$m_{clo}(i,j) = (1-\alpha(i,j))f_{dir}(i,j)+\alpha(i,j)f_{loc}(i,j). \tag{5.8}$$

To compute the direction fidelity $f_{dir}(i,j)$ and location fidelity $f_{loc}(i,j)$, we first calculate the nearest points of two end vertices, and midpoint of edge (i,j) on feature curve. Then, we use the average of curve tangent on these three nearest points to compute the angular deviation $\theta_{dev}(i,j)$ between the tangent of the mesh edge (i,j) and the tangent of target curve. Similarly, we compute the distance $d(i,j)$ between the curve and mesh edge (i,j) by computing the average distance between the nearest three points on curve and their original points on the mesh edge. According to the angular deviation and distance between curve and mesh edge, we compute the direction fidelity and location deviation as follows:

$$f_{dir}(i,j) = 1 - cos^2\theta_{dev}(i,j), \tag{5.9}$$

$$f_{loc}(i,j) = (\frac{d(i,j)}{d_{max}})^2. \tag{5.10}$$

The d_{max} is the max distance between mesh edge and curve. To reduce the scale of calculate, we limit searching space within the $3l_e$ from the target feature curve. To ensure the shortest path is spatially close to the target feature, we increase the weight of $f_{loc}(i,j)$ as the mesh edge move away from the target feature curve by setting:

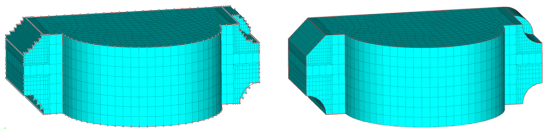
$$\alpha(i,j) = \frac{d(i,j)}{3l_e}. \tag{5.11}$$

Figure 8a shows the mesh edges, whose endpoint is highlighted with balls, determined by our searching algorithm. Each of them corresponds to a feature curve highlighted in red.

5.2.3 Establishment of correspondence between mesh faces and feature surfaces

After segmenting the boundary of core mesh into multiple patches by the sequences of mesh edges corresponding to feature curves, it's intuitive to establish the correspondence between a patch and a feature surface based on the patches boundary edges.

After establish correspondence between feature entities and boundary mesh entities, we project mesh boundary entities to the corresponding feature entities. After projection, we obtain a mesh approximating target geometry well, as shown in Figure 8b.



(a) (b)

Figure 8: (a)eature curves and their corresponding mesh edges. The feature curves are highlighted in red and end vertices of their corresponding mesh edges are highlighted with ball.(b)Mesh approximates the target geometry well after projection according to the corresponding relationship.

6 Topological and geometric optimization of core mesh

After fitting the target geometry with core mesh based on correspondence between core mesh and target geometry, there may be some hexahedra with poor quality over the boundary layer of mesh. Optimization is required for these hexahedra. Due to the geometric constraints, for hexahedra with poor quality adjacent to the feature curves, we have to change geometry and connectivity of mesh in parallel to improve the quality. For remaining hexahedra with poor quality, geometric optimization is enough.In particular, we improve the mesh quality through the following two steps:

1. Topological edition over mesh boundary.
2. Geometric optimization over mesh boundary layer.

For topological optimization, we apply a mesh boundary quality improvement technique [34] based on notion of the fundamental mesh to specific regions, and guide the relocation of mesh vertices based on a mesh quality metric. For geometric optimization, we first apply Laplacian smoothing and then optimize the ill-posed vertices based on the formula $Q(v) = min_{h\in\mathcal{H}_v} SJ(h)$. Here, $\mathcal{H}_v$ refers to the hexahedra adjacent to vertex v, and $SJ(h)$ refers to the scaled Jacobian of hexahedron h. The fundamental idea of optimizing vertices is to select the point with the highest score within a bounding box.

7 Experiments

Our method is tested on a PC with 3.9 GHz AMD Ryzen 7 3800X CPU and 16GB of RAM. The graph-cut optimization is computed by the graph-cut based multiple-label optimization framework (https://vision.cs.uwaterloo.ca/files/gco-v3.0.zip). The remaining part of algorithm in this paper is developed based on HyperMesh, and the programming language is Tcl.

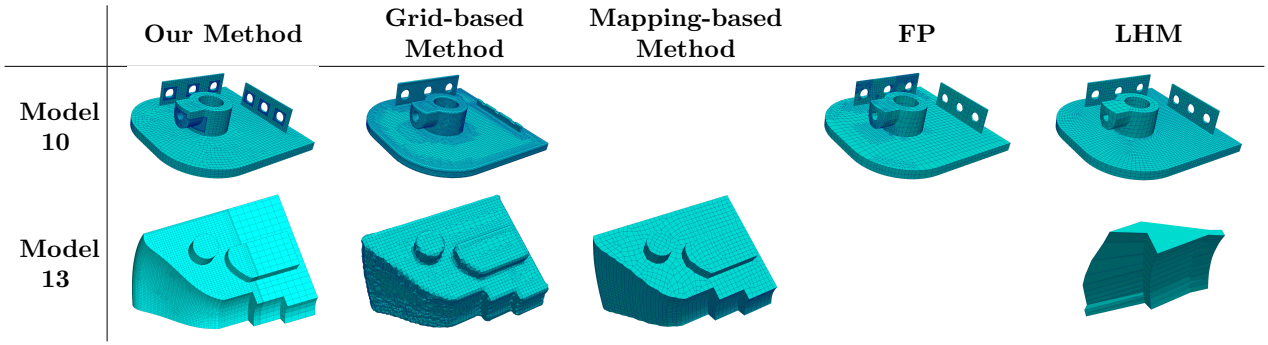


Figure 9: Results of five meshing algorithms for model 10 and model 13.

Table 1: Comparison between the results of three meshing algorithms. We compare the resulting meshes in five aspects: meshability(**Meshability**), min scaled Jacobian($\mathbf{SJ_{min}}$), average scaled Jacobian($\mathbf{SJ_{ave}}$), conformality(**Conformality**), and number of hexahedral(#**H**). The notation ✓(✖, repsectively) indicates that the corresponding method can (cannot, respectively) generate a hex mesh of model. The notation ☑ indicates that the corresponding method generate a hex mesh of model with defects, for example, a hex mesh with negative scaled Jacobian cell or an incomplete mesh that cannot fully represent the model. The notation ☒ indicates that the failure of the method may be caused by inappropriate inputs.

Model	Method	Meshability	$\mathbf{SJ_{min}}$	$\mathbf{SJ_{ave}}$	Conformality	#H
model 10	Our	✓	0.08	0.81	High	40155
	Grid	☑	0.11	0.75	Low	76719
	Mapping	✖	-	-	-	-
	FP	✓	0.09	0.80	High	14225
	LHM	☑	−0.995	0.96	High	9286
model 13	Our	✓	0.06	0.94	High	60516
	Grid	✓	0.17	0.81	Low	25360
	Mapping	✓	0.4	0.92	High	13382
	FP	✖	-	-	-	-
	LHM	☑	−1	−0.62	Low	22

7.1 Result comparison

To demonstrate the effectiveness of our method, we conduct comparative experiments with several algorithms. We compare our method with two mature algorithms implemented in commercial software and two state-of-the-art methods. Specifically, we used our method, the grid-based hex meshing algorithm implemented in Bolt, the mapping-based hex meshing algorithm implemented in HyperMesh, grid-based method [9] and frame-field based method [31] to generate hex meshes for 13 CAD models respectively. We compare the resulting meshes of different methods in terms of meshability, (min and average) scaled Jacobian, conformality and number of cells. Meshability refers to whether a hexahedral mesh of the model can be generated via the corresponding method.

We compared our method with the grid-based hex meshing algorithm implemented in Bolt, the mapping-based hex meshing algorithm implemented in HyperMesh, grid-based method [9] (FP, for short) and frame-field based method [31](LHM, for short). When generating hex mesh by grid-based method with the commercial software Bolt, we set the mesh edge length and refinement levels used by Bolt to be the same as those used by our algorithm for each model. And we apply the built-in Laplacian smoothing operation and dual operations in Bolt to improve the quality on the boundary layer. When generating hex mesh by mapping-based method with the commercial software HyperMesh, we set the mesh edge length used by HyperMesh to be the same as the one used by our algorithm for each model. Due to the complexity of the model, we have to manually decompose the model before hex meshing. For the other two algorithms, we used the default parameters. The results of five meshing algorithms for model 10 and model 13 are shown in Figure 9, and comparison between these results is given in Table 1. (Please refer to the supplementary materials for the complete comparison data and hexahedral meshes obtained by five algorithms for the 13 models.)

Mesh quality. The meshes obtained by our method is generally better than other methods in term

of the $\mathbf{SJ_{min}}$. This is due to the fact that our algorithm starts from a swept mesh with local regions refined rather than an adaptive octree. However, locally refined mesh patterns may restrict the mesh's applicability in certain scenarios. Frame-field based method [31] behaves best in terms of $\mathbf{SJ_{ave}}$, if it can produce an conformal mesh for the input. The hex elements with negative **SJ** in the mesh generated by the frame field based method[31] are located near the singular lines in our experiments. In term of the conformality, our method behaves similarly as the mapping-based method as well as FP [9], and behaves better than the grid-based method, since the final mesh is converted from an initial background mesh capturing some features and boundary layer has been improved by topological and geometric optimization.

Robustness. According to the statistics in this table, we can know that our method can stably generate hexahedral meshes for these 13 models. Considering the simplicity of the models, for model $1-3$, we guess that there may be some issues with the input models, leading to failures in both FP[9] and LHM[31]. Besides, our method is more effective on certain types of models, and our method may fail for models for whom it is very difficult to generate a high-qualtiy quad-meshable principal face under a constant projection direction.

Efficiency performance. Based on our algorithm, most models take more than 30 minutes to generate a hexahedral mesh. The most time-consuming step is the mesh refinement. However, for the LHM method and FP method, the vast majority of models take no more than 10 minutes if the model can be successfully hex-meshed.

7.2 Limitations

There are still some problems with our method.

1. Our method may produce hex mesh with low quality for model with spline surfaces (for example, model 13 as shown in Figure 9) and swept model with multiple sweeping axis (for example, model 10 shown in Figure 9). This is due to the high probability for these models that there are some features that cannot be easily captured by the background mesh generated based on sweeping, and in order to fit these features, large distortion will occur.
2. Our method is not good at meshing the model with dense features. Excessive feature constraints will result in poor quality of the generated background mesh, which cannot be improved even with subsequent optimization.
3. The use of limited templates may become the bottleneck of this method to obtain high-quality mesh.

8 Conclusions and future work

In this paper, we proposed a novel algorithm for automatically hex meshing the CAD models. Compared with the previous grid-based hex meshing algorithms, this algorithm has the following characteristics:

1. The non-regular background hex mesh generated by projecting model features into 2D plane and sweeping the projected 2D shape is adopted. Because it is able to capture part of boundary features, it can help improve the quality of mesh boundary.
2. By conducting the graph-cut optimization, a globally optimized core mesh is obtained. And by establishing the correspondence between feature and mesh via searching algorithm based on matching degree, the generated hex mesh can effectively capture all features of the model.
3. By applying pillowing operation, and inserting 2 types of transition templates to the interfaces between domains, local domains with any topological arrangement is able to be refined without affecting other domains.

In the future, we will improve our algorithm in the following aspects:

1. **Generalizing the non-regular background mesh generation algorithm.** As discussed in § 7.2, heuristic rules in the current algorithm may result in low-quality meshes for specific models. For instance, we plan to generalize the algorithm by projecting features onto a general surface rather than a plane, allowing us to obtain a more universally applicable principal face.
2. **Integrating topology optimization and correspondence establishing.** The integration of them aims to combine topology optimization and the establishment of correspondence between target geometry and core mesh into a single step. By editing the mesh's topology before projecting it onto the target geometry, there will be no need to suppress feature curves through variable valence of mesh vertices.
3. **Designing additional templates for refinement and transition.** Introducing more templates will enable us to achieve uniform mesh density variation in a more flexible manner.

Acknowledgements

We thank all anonymous reviewers for their valuable comments.

References

[1] Pietroni N., Campen M., Sheffer A., Cherchi G., Bommes D., Gao X., Scateni R., Ledoux F., Remacle J., Livesu M. "Hex-mesh generation and processing: a survey." *ACM transactions on graphics*, vol. 42, no. 2, 1–44, 2022

[2] Schneider T., Hu Y., Gao X., Dumas J., Zorin D., Panozzo D. "A large-scale comparison of tetrahedral and hexahedral elements for solving elliptic PDEs with the finite element method." *ACM Transactions on Graphics (TOG)*, vol. 41, no. 3, 1–14, 2022

[3] Schneiders R. "A grid-based algorithm for the generation of hexahedral element meshes." *Engineering with computers*, vol. 12, 168–177, 1996

[4] "CoreForm." Website, 2023. `https://coreform.com/products/coreform-cubit/government/`

[5] "Cubit." Website, 2023. `https://cubit.sandia.gov/`

[6] Zhang H., Zhao G., Ma X. "Adaptive generation of hexahedral element mesh using an improved grid-based method." *Computer-Aided Design*, vol. 39, no. 10, 914–928, 2007

[7] Owen S.J., Shepherd J.F. "Embedding features in a cartesian grid." *proceedings of the 18th International Meshing Roundtable*, pp. 117–138. Springer, 2009

[8] Qian J., Zhang Y. "Automatic unstructured all-hexahedral mesh generation from B-Reps for non-manifold CAD assemblies." *Engineering with Computers*, vol. 28, 345–359, 2012

[9] Gao X., Shen H., Panozzo D. "Feature preserving octree-based hexahedral meshing." *Computer graphics forum*, vol. 38, pp. 135–149. Wiley Online Library, 2019

[10] Gao X., Shen H., Panozzo D. "Feature Preserving Octree-Based Hexahedral Meshing." *Computer Graphics Forum*, vol. 38, no. 5, 135–149, 2019. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13795`

[11] Pitzalis L., Livesu M., Cherchi G., Gobbetti E., Scateni R. "Generalized Adaptive Refinement for Grid-Based Hexahedral Meshing." *ACM Trans. Graph.*, vol. 40, no. 6, dec 2021. URL `https://doi.org/10.1145/3478513.3480508`

[12] Cunha A., Canann S., Saigal S. "Automatic boundary sizing for 2D and 3D meshes." *ASME APPLIED MECHANICS DIVISION-PUBLICATIONS-AMD*, vol. 220, 65–72, 1997

[13] Owen S.J., Saigal S. "Neighborhood-based element sizing control for finite element surface meshing." *6th international meshing roundtable Proceedings*, pp. 143–154. 1997

[14] Tchon K.F., Khachan M., Guibault F., Camarero R. "Three-dimensional anisotropic geometric metrics based on local domain curvature and thickness." *Computer-Aided Design*, vol. 37, no. 2, 173–187, 2005

[15] Babuvška I., Rheinboldt W.C. "Error estimates for adaptive finite element computations." *SIAM Journal on Numerical Analysis*, vol. 15, no. 4, 736–754, 1978

[16] Zienkiewicz O., Zhu J., Gong N. "Effective and practical h–p-version adaptive analysis procedures for the finite element method." *International Journal for Numerical Methods in Engineering*, vol. 28, no. 4, 879–891, 1989

[17] Verfurth R. "A combined conjugate gradient-multi-grid algorithm for the numerical solution of the Stokes problem." *IMA Journal of Numerical Analysis*, vol. 4, no. 4, 441–455, 1984

[18] Schneiders R. "Octree-based generation of hexahedral element meshes." *Proceedings of the 5-th International Meshing Roundtable, 1996*, pp. 205–215, 1996

[19] Ebeida M.S., Patney A., Owens J.D., Mestreau E. "Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates." *International Journal for Numerical Methods in Engineering*, vol. 88, no. 10, 974–985, 2011

[20] Edgel J. *An adaptive grid-based all hexahedral meshing algorithm based on 2-refinement*. Brigham Young University, 2010

[21] Owen S.J., Shih R.M., Ernst C.D. "A template-based approach for parallel hexahedral two-refinement." *Computer-Aided Design*, vol. 85, 34–52, 2017

[22] Borden M.J., Benzley S.E., Mitchell S.A., White D.R., Meyers R.J. "The Cleave and Fill Tool: An All-Hexahedral Refinement Algorithm for Swept Meshes." *IMR*, pp. 69–76. 2000

[23] Tchon K.F., Dompierre J., Scamarero R. "Automated refinement of conformal quadrilateral and hexahedral meshes." *International Journal for Numerical Methods in Engineering*, vol. 59, no. 12, 1539–1562, 2004

[24] Nieser M., Reitebuch U., Polthier K. "Cubecover–parameterization of 3d volumes." *Computer graphics forum*, vol. 30, pp. 1397–1406. Wiley Online Library, 2011

[25] Huang J., Tong Y., Wei H., Bao H. "Boundary aligned smooth 3D cross-frame field." *ACM transactions on graphics (TOG)*, vol. 30, no. 6, 143, 2011

[26] Li Y., Liu Y., Xu W., Wang W., Guo B. "All-hex meshing using singularity-restricted field." *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, 1–11, 2012

[27] Gregson J., Sheffer A., Zhang E. "All-hex mesh generation via volumetric polycube deformation." *Computer graphics forum*, vol. 30, pp. 1407–1416. Wiley Online Library, 2011

[28] Mandad M., Chen R., Bommes D., Campen M. "Intrinsic mixed-integer polycubes for hexahedral meshing." *Computer aided geometric design*, vol. 94, 102078, 2022

[29] Brückler H., Bommes D., Campen M. "Volume parametrization quantization for hexahedral meshing." *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, 1–19, 2022

[30] Lyon M., Bommes D., Kobbelt L. "HexEx: Robust hexahedral mesh extraction." *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, 1–11, 2016

[31] LIU H., BOMMES D. "Locally Meshable Frame Fields." *ACM Trans. Graph*, vol. 42, no. 4, 2023

[32] Boykov Y., Veksler O., Zabih R. "Fast approximate energy minimization via graph cuts." *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 11, 1222–1239, 2001

[33] Boykov Y., Kolmogorov V. "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 9, 1124–1137, 2004

[34] Ledoux F., Le Goff N., Owen S.J., Staten M.L., Weill J.C. "A constraint-based system to ensure the preservation of sharp geometric features in hexahedral meshes." *Proceedings of the 21st international meshing roundtable*, pp. 315–332. Springer, 2013