

# HexDom: Polycube-Based Hexahedral-Dominant Mesh Generation

Yuxuan Yu, Jialei Ginny Liu and Yongjie Jessica Zhang

**Abstract** In this paper, we extend our earlier polycube-based all-hexahedral mesh generation method to hexahedral-dominant mesh generation, and present the Hex-Dom software package. Given the boundary representation of a solid model, Hex-Dom creates a hex-dominant mesh by using a semi-automated polycube-based mesh generation method. The resulting hexahedral dominant mesh includes hexahedra, tetrahedra, and triangular prisms. By adding non-hexahedral elements, we are able to generate better quality hexahedral elements than in all-hexahedral meshes. We explain the underlying algorithms in four modules including segmentation, polycube construction, hex-dominant mesh generation and quality improvement, and use a rockerarm model to explain how to run the software. We also apply our software to a number of other complex models to test their robustness. The software package and all tested models are available in github (<https://github.com/CMU-CBML/HexDom>).

## 1 Introduction

In finite element analysis (FEA), a 3D domain can be discretized into tetrahedral or hexahedral (hex) meshes. For tetrahedral mesh generation, various strategies have been proposed in the literature [54, 15, 19, 33], such as octree-based [38, 56], delaunay triangulation [4], and advancing front methods [8, 23, 37]. Because tetra-

---

Y. Yu

Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA  
e-mail: yuxuany1@andrew.cmu.edu

J. G. Liu

Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA  
e-mail: jialeil@andrew.cmu.edu

Y. J. Zhang (✉)

Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA  
e-mail: jessicaz@andrew.cmu.edu

hedral meshes can be created automatically, it has been widely used in industry. However, to achieve the same precision in FEA, a tetrahedral mesh requires more elements than an all-hex mesh does. As a result, many techniques have been developed to generate all-hex meshes [48, 47, 61, 32, 34] or converting imaging data to all-hex meshes [54, 53, 55]. Also, hex meshes can serve as multiple-material domains [59, 62] or input control meshes for IGA [16, 46, 45, 17, 29, 64, 44, 43]. Some applications of hex mesh generation in new engineering applications can also be found in [52, 50, 18, 60, 58]. Several literatures develop methods for unstructured hex mesh generation, such as grid-based or octree-based [35, 36], medial surface [31, 30], plastering [2, 39], whisker weaving [7], and vector field-based methods [26]. These methods have been used to create hex meshes for certain geometries, but are not robust and reliable for arbitrary geometries. On the other hand, although an all-hex mesh provides a more accurate solution, a high-quality all-hex mesh is more difficult to create automatically.

Compared with all-hex mesh generation, a hex-dominant mesh generation, which combines advantages of both tetrahedral and hex elements, is more automatic and robust for complex solid models. In the literature, several strategies have been proposed to generate hex-dominant meshes. An indirect method was suggested by [49], the domain is first meshed into tetrahedral elements and then merged into a hex-dominant mesh with the packing technology. Several other hex-dominant meshing techniques were also presented in [28, 25, 24]. Such indirect methods create hex-dominant meshes with too many singularities, and the tetrahedral mesh directly influences the quality of the hex-dominant mesh. Similar to unstructured all-hex mesh generation, the direct method is also more preferable for hex-dominant meshes [41, 27]. The polycube-based method [40, 9] is an attractive direct approach to obtain hex-dominant meshes by using degenerated cubes. The polycube-based method was mainly used for all-hex meshing. A smooth harmonic field [42] was used to generate polycubes for arbitrary genus geometry. Boolean operations [21] were introduced to deal with arbitrary genus geometry. In [22], a polycube structure was generated based on the skeletal branches of a geometric model. Using these methods, the structure of the polycube and the mapping distortion greatly influence the quality of the hex mesh. The calculation of the polycube structure with a low mapping distortion remains an open problem for complex geometry. It is important to improve the quality of the mesh for analysis by using methods such as pillowing, smoothing, and optimization [34, 57, 63, 32]. Pillowing is an insert-sheet technique that eliminates situations where two adjacent hex elements share more than one face. Smoothing and optimizing are used to further improve the quality of the mesh by moving the vertices. In our software, we implement all of the above methods to improve the quality of hex elements.

In this paper, we extend our earlier semi-automatic polycube-based all-hex generation to hex-dominant meshing. The software package includes: 1) polycube based geometric decomposition of a surface triangle mesh; 2) generation of the polycube consisting of non-degenerated and degenerated cubes; 3) creation of a parametric domain for different types of degenerated unit cubes including prisms and tetrahedra; and 4) creation of a hex-dominant mesh. We first go through the entire pipeline and

explain the algorithm behind each module of the pipeline. Then, we use a specific example to follow all the steps and run the software. In particular, when user intervention is required, the details of manual work are explained. The paper is organized as follows. In Section 2 we provide an overview of the pipeline. In Section 3 we present the HexDom software package, with a semi-automatic polycube-based hex-dominant mesh generation of a CAD file. Finally, in Section 4 we show various complex models with our software package.

## 2 Pipeline design

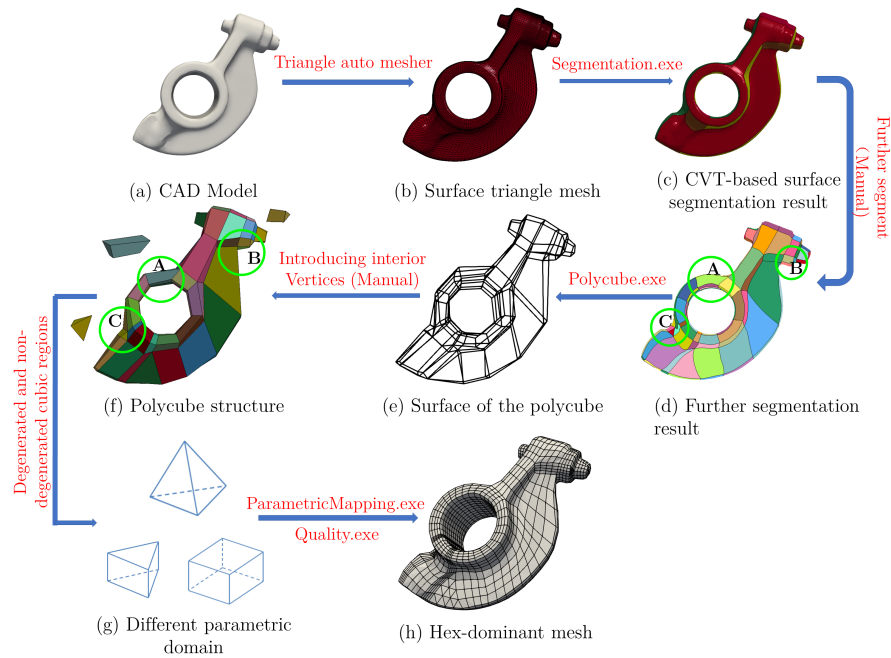


Fig. 1: The HexDom software package. For each process, the black texts describe the object and the red texts show the operation needed to go to the next process. Manual work is involved in further segmentation and introducing interior vertices. Regions A, B and C (green circles) in (d, f) contain a hex, prism and tetrahedral shaped structure, respectively.

Our pipeline uses polycube-based method to create a hex-dominant mesh from an input CAD model. As shown in Fig. 1, we first generate a triangle mesh from the CAD model by using the free software LS-PrePost. Then we use centroidal Voronoi tessellation (CVT) segmentation [13, 14, 12] to create a polycube structure [40].

The polycube structure consists of multiple non-degenerated cubes and degenerated cubes. The non-degenerated cubes will yield hex elements via parametric mapping [6] and octree subdivision [63]. The degenerated cubes will yield degenerated elements such as prisms and tetrahedra in the final mesh. Here, we implement the subdivision algorithm separately for prism-shape regions and tetrahedral-shape regions. The quality of the hex dominant mesh is evaluated to ensure that the resulting mesh can be used in FEA. In case that a poor quality hex element is generated in hex-dominant meshes, the program has various quality improvement functions, including pillowing [57], smoothing, and optimization [32]. Each quality improvement function can be performed independently and one can use these functions to improve the mesh quality. Currently, our software only has a command-line interface (CLI). Users need to provide the required options on the command line to run the software. In Section 3, we will explain in detail the algorithms implemented in the software as well as how to run the software.

### 3 HexDom: Polycube-based hex-dominant mesh generation

Surface segmentation, polycube construction, parametric mapping, and subdivision are used together in the HexDom software package to generate a hex-dominant mesh from the boundary representation of the input CAD model. Given a triangle mesh generated from the CAD model, we first use surface segmentation to divide the mesh into several surface patches that meet the restrictions of the polycube structure, which will be discussed in Section 3.1. The corner vertices, edges, and faces of each surface patch are then extracted from the surface segmentation result to construct a polycube structure. Each component of the polycube structure is topologically equivalent to a cube or a degenerated cube. Finally, we generate the hex-dominant mesh through parametric mapping and subdivision. Quality improvement techniques can be used to further improve the mesh quality.

In this section, we will introduce the main algorithm for each module of the HexDom software package, namely surface segmentation, polycube construction, parametric mapping and subdivision, and quality improvement. We will use a rock-arm model (see Fig. 1) to explain how to run CLI for each module. We will also discuss the user intervention involved in the semi-automatic polycube-based hex-dominant mesh generation.

#### 3.1 Surface segmentation

The surface segmentation in the pipeline framework is implemented based on CVT segmentation [13, 14, 12]. CVT segmentation is used to classify vertices by minimizing an energy function. Each group is called a Voronoi region and has a corresponding center called a generator. The Voronoi region and its corresponding generator are

updated iteratively in the minimization process. In [13], each element of the surface triangle mesh is assigned to one of the six Voronoi regions based on the normal vector of the surface. The initial generators of the Voronoi regions are the three principal normal vectors and their opposite normals vectors ( $\pm X$ ,  $\pm Y$ ,  $\pm Z$ ). Two energy functions and their corresponding distance functions are used together in [13]. The classical energy function and its corresponding distance function provide initial Voronoi regions and generators. Then, the harmonic boundary-enhanced (HBE) energy function and its corresponding distance function are applied to eliminate non-monotone boundaries. The detailed definitions of the energy functions and their corresponding distance functions are described in [13]. The surface segmentation process was also summarized in the **Surface Segmentation Algorithm** in [51].

Once we get the initial segmentation result, we need to further segment each Voronoi region into several patches to satisfy the topological constraints for polycube construction (see Fig. 1(d)). We use two types of patches. The first type of segmented surface patch corresponds to one boundary surface of the non-degenerated cubes and quadrilateral surface of the prism-shape degenerated cubes. The second type of segmented surface patch corresponds to one triangular boundary surface of the degenerated cubes. The choice of types of patches depends on the following three criteria: 1) geometric features such as sharp corners with small angles and prism/tetrahedral-like features; 2) critical regions based on finite element simulation, such as regions with the maximum stress/strain and regions with a high load; and 3) requirements from user applications which enhance the capability of user interaction. For the first type of segmented surface patch, the following three conditions should be satisfied during the further segmentation: 1) two patches with opposite orientations (e.g.,  $+X$  and  $-X$ ) cannot share a boundary; 2) each corner vertex must be shared by more than two patches; and 3) each patch must have four boundaries. For the second type of segmented surface patch, we modified the third conditions to that each patch must have three boundaries.

Note that we define the corner vertex as a vertex locating at the corner of the cubic region or degenerated cubic region in the model. The further segmentation is done manually by using the patch ID reassigning function in LS-PrePost. The detailed operation was shown in [51].

### 3.2 Polycube construction

In this section, we discuss the detailed algorithm of polycube construction based on the segmented triangle mesh. Several automatic polycube construction algorithms have been proposed in the literature [11, 20, 13], but it is challenging to apply these methods to complex CAD models. The polycube structure does not contain degenerated cubes either. Differently, the polycube in this paper consists of cubes and degenerated cubes and is topologically equivalent to the original geometry. To achieve versatility for real industrial applications, we develop a semi-automatic polycube

construction software based on the segmented surface. However, for some complex geometries, the process may be slower due to potentially heavy user intervention.

The most important information we need for a polycube is its corners and the connectivity relationship among them. For the surface of polycube, we can automatically get the corner points and build their connectivity based on the segmentation result by using the algorithm similar to the **Polycube Boundary Surface Construction Algorithm** in [51]. The difference is that we need to adjust the implementation based on different patch types: finding its three corners for a triangular patch and finding its four corners for a quadrilateral patch. It is difficult to obtain inner vertices and their connectivity because we only have a surface input with no information about the interior volume. In fact, this is where users need to intervene. We use LS-PrePost to manually build the interior connectivity. You can find the detailed operation in Appendix A3 in [51]. As the auxiliary information for this user intervention, the **Polycube Boundary Surface Construction Algorithm** will output corners and connectivity of the segmented surface patches into *.k* file. Finally, the generated polycube structure is the combination of non-degenerated cubes and degenerated cubes splitting the volumetric domain of the geometry.

### 3.3 Parametric mapping and subdivision

After the polycube is constructed, we need to build a bijective mapping between the input triangle mesh and the boundary surface of the polycube structure. In our software, we implement the same idea as in [22]: using a non-degenerated unit cube or a degenerated unit cube as the parametric domain for the polycube structure. As a result, we can construct a generalized polycube structure that can better align with the given geometry and generate a high quality hex-dominant mesh.

There are three types of elements in the hex-dominant mesh: hex, prism, and tetrahedral. The hex elements form non-degenerated cubic regions. Prism and tetrahedral elements form degenerated cubic regions. We will use octree subdivision to generate hex elements for non-degenerated cubic regions, while using subdivision to generate prism and tetrahedral elements for degenerated cubic regions. Through the pseudocode in the **Parametric Mapping Algorithm** in [51], we describe how to combine the segmented surface mesh, the polycube structure, and the unit cube to create an all-hex mesh. We use this algorithm to create the hex elements in non-degenerated cubic regions. Each non-degenerated cube in the polycube structure represents one volumetric region of the geometry and has a non-degenerated unit cube as its parametric domain. Region *A* in Fig. 1(d, f) shows an example of a non-degenerated cube and its corresponding volume domain of the geometry marked in the green circle. For degenerated cubes, there are two types of interface, a triangular face and a quadrilateral face. Region *B* in Fig. 1(d, f) shows a prism case, it contains two triangular faces and three quadrilateral faces. For the tetrahedral case shown in Region *C* in Fig. 1(d, f), it contains four triangular faces.

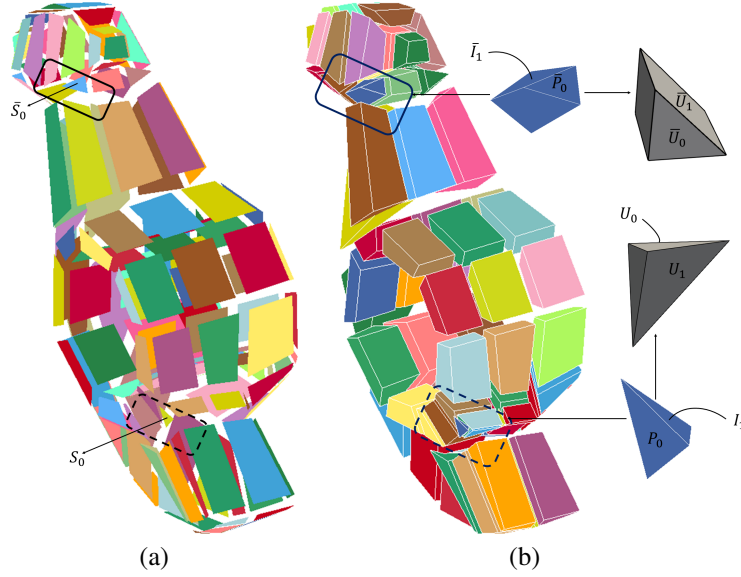


Fig. 2: The polycube construction and the parametric mapping process for prism-shape degenerated cubic regions (see the black boxes) and tetrahedral-shape degenerated cubic regions (see the dashed black boxes). (a) The boundary surface of the polycube generated by **Polycube Boundary Surface Construction Algorithm**; (b)  $\bar{S}_0$ ,  $\bar{P}_0$  and  $\bar{U}_0$  are used for parametric mapping to create boundary vertices of the prism-shape degenerated cubic regions.  $\bar{I}_1$  and  $\bar{U}_1$  are used for linear interpolation to create interior vertices of the prism-shape degenerated cubic regions.  $S_0$ ,  $P_0$  and  $U_0$  are used for parametric mapping to create boundary vertices of the tetrahedral-shape degenerated cubic regions.  $I_1$  and  $U_1$  are used for linear interpolation to create interior vertices of the tetrahedral-shape degenerated cubic regions.

Through the pseudocode in the **Prism Parametric Mapping Algorithm**, we describe how the segmented surface mesh, the polycube structure and the prism-shape degenerated unit cube are combined to generate prism elements. Let  $\{\bar{S}_i\}_{i=1}^N$  be the segmented surface patches coming from the segmentation result (see Fig. 2(a)). Each segmented surface patch corresponds to one boundary surface of the polycube  $\bar{P}_i$  ( $1 \leq i \leq N$ ) (see Fig. 2(b)), where  $N$  is the number of the boundary surfaces. There are also interior surfaces, denoted by  $\bar{I}_j$  ( $1 \leq j \leq M$ ), where  $M$  is the number of the interior surfaces. The union of  $\{\bar{P}_i\}_{i=1}^N$  and  $\{\bar{I}_j\}_{j=1}^M$  is the set of surfaces of the polycube structure. For the parametric domain, let  $\{\bar{U}_k\}_{k=1}^5$  denote the five surface patches of the prism-shape degenerated unit cube (see Fig. 2(b)).

Each prism-shape degenerated cube in the polycube structure represents one volumetric region of the geometry and has a prism-shape degenerated unit cube as its parametric domain. Fig. 2(b) shows an example of prism-shape degenerated cube and its corresponding volume domain of the geometry marked in the black

boxes. Therefore, for each prism-shape degenerated cube in the polycube structure, we can find its boundary surface  $\bar{P}_i$  and map the segmented surface patch  $\bar{S}_i$  to its corresponding parametric surface  $\bar{U}_k$  of the prism-shape degenerated unit cube. To map  $\bar{S}_i$  to  $\bar{U}_k$ , we first map its corresponding boundary edges of  $\bar{S}_i$  to the boundary edges of  $\bar{U}_k$ . Then we get the parameterization of  $\bar{S}_i$  by using the cotangent Laplace operator to compute the harmonic function [63, 5]. Compared to non-degenerated cubic regions algorithm, we introduce three parametric variables in mapping since one face is not axis-aligned. Note that for an interior surface  $\bar{I}_j$  of the polycube structure, we skip the parametric mapping step.

The prism elements can then be obtained from the above surface parameterization combined with subdivision. We generate the prism elements for each prism-shape region in the following process. To obtain vertex coordinates on the segmented patch  $\bar{S}_i$ , we first subdivide the prism-shape degenerated unit cube (see Fig. 3(a)) recursively in order to get their parametric coordinates. The vertex coordinates of triangular faces of the prism-shape degenerated cube are obtained by linear subdivision, while the quadrilateral faces are also obtained by linear subdivision. The physical coordinates can be obtained by using parametric mapping, which has a one-to-one correspondence between the parametric domain  $\bar{U}_k$  and the physical domain  $\bar{S}_i$ . To obtain vertices on the interior surface of the prism region, we skip the parametric mapping step and directly use linear interpolation to calculate the physical coordinates. Finally, vertices inside the cubic region are calculated by linear interpolation. The entire prism elements are built by going through all the prism-shape regions.

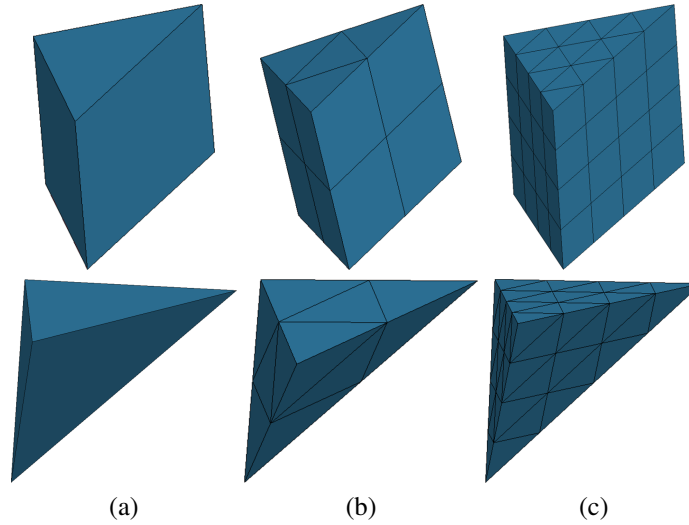


Fig. 3: The subdivision of prism-shape degenerated unit cube (top row) and tetrahedral-shape degenerated unit cube (bottom row). (a) Subdivision level 0; (b) subdivision level 1; and (c) subdivision level 2.



---

**Prism Parametric Mapping Algorithm**


---

**Input:** Segmented triangle mesh  $\{\bar{S}_i\}_{i=1}^N$ , polycube structure

**Output:** Prism elements in prism-shape degenerated cubic regions

1: Find boundary surfaces  $\{\bar{P}_i\}_{i=1}^N$  and interior surfaces  $\{\bar{I}_j\}_{j=1}^M$  in the polycube structure

**Surface parameterization step:**

2: **for** each prism-shape degenerated cube in the polycube structure **do**

3:     Create a prism-shape degenerated cube region  $\{\bar{U}_k\}_{k=1}^5$  as the parametric domain

4:     **for** each surface in the prism-shape degenerated cube **do**

5:         **if** it is a boundary surface  $\bar{P}_i$  **then**

6:             **if** the surface is not axis-aligned **then**

7:                 Get the surface parameterization  $f : \bar{S}_i \rightarrow \bar{U}_k \subset \mathbb{R}^3$

8:             **else**

9:                 Get the surface parameterization  $f : \bar{S}_i \rightarrow \bar{U}_k \subset \mathbb{R}^2$

10:             **end if**

11:         **end if**

12:     **end for**

13: **end for**

**Parametric mapping and subdivision step:**

14: **for** each prism-shape degenerated cube in the polycube structure **do**

15:     Subdivide the prism-shape degenerated unit cube recursively to get parametric coordinates

16:      $v_{para}$  **for** each surface in the prism-shape degenerated cube **do**

17:         **if** it is a boundary surface  $\bar{P}_i$  **then**

18:             Obtain physical coordinates using  $f^{-1}(v_{para})$

19:         **else if** it is an interior surface  $\bar{I}_j$  **then**

20:             Obtain physical coordinates using linear interpolation

21:         **end if**

22:     **end for**

23:     Obtain interior vertices in the prism-shape degenerated cubic region using linear interpolation

24: **end for**

---

We perform the similar procedure for the tetrahedra-shape degenerated cubes in the polycube structure. Through the pseudocode in the **Tetrahedral Parametric Mapping Algorithm**, we describe how the segmented surface mesh, the polycube structure and the tetrahedral-shape degenerated unit cube are combined to generate tetrahedral elements. Fig. 2(b) shows an example of tetrahedral-shape degenerated cube and its corresponding volume domain of the geometry marked in the dashed black boxes. The difference is that we use  $\{U_k\}_{k=1}^4$  to denote those four surface patches of the tetrahedra-shape degenerated unit cube for the parametric domain. We also introduce three parametric variables in mapping when one of the surfaces is not axis aligned. Then, the tetrahedral elements can be obtained from this surface parameterization combined with linear subdivision. We generate tetrahedral elements for each tetrahedral-shape region in the following process. To obtain vertex coordinates on the segmented patch  $S_i$ , we first subdivide the tetrahedral-shape degenerated unit cube (see Fig. 3(bottow row)) recursively in order to get their parametric coordinates by applying linear subdivison. The physical coordinates can be obtained by using the parametric mapping between the parametric domain  $U_k$  and

the physical domain  $S_i$ .  $I_1$  and  $U_1$  are combined for linear interpolation to obtain vertices on the interior surface of the tetrahedra-shape degenerated cubic region. Finally, vertices inside the tetrahedra-shape degenerated cube region are calculated by linear interpolation. The entire tetrahedral elements are built by going through all the tetrahedral regions.

---

### Tetrahedral Parametric Mapping Algorithm

---

**Input:** Segmented triangle mesh  $\{S_i\}_{i=1}^N$ , polycube structure

**Output:** Tetrahedral elements in tetrahedral-shape degenerated cubic regions

1: Find boundary surfaces  $\{P_i\}_{i=1}^N$  and interior surfaces  $\{I_j\}_{j=1}^M$  in the polycube structure

**Surface parameterization step:**

2: **for** each tetrahedral-shape degenerated cube in the polycube structure **do**

3:     Create a tetrahedral-shape degenerated cube region  $\{U_k\}_{k=1}^4$  as the parametric domain

4:     **for** each surface in the tetrahedral-shape degenerated cube **do**

5:         **if** it is a boundary surface  $P_i$  **then**

6:             **if** the surface is not axis-aligned **then**

7:                 Get the surface parameterization  $f : S_i \rightarrow U_k \subset \mathbb{R}^3$

8:             **else**

9:                 Get the surface parameterization  $f : S_i \rightarrow U_k \subset \mathbb{R}^2$

10:             **end if**

11:         **end if**

12:     **end for**

13: **end for**

**Parametric mapping and subdivision step:**

14: **for** each tetrahedral-shape degenerated cube in the polycube structure **do**

15:     Subdivide the tetrahedral-shape degenerated unit cube recursively to get parametric coordinates  $v_{para}$

16:     **for** each surface in the tetrahedral-shape degenerated cube **do**

17:         **if** it is a boundary surface  $P_i$  **then**

18:             Obtain physical coordinates using  $f^{-1}(v_{para})$

19:         **else if** it is an interior surface  $I_j$  **then**

20:             Obtain physical coordinates using linear interpolation

21:         **end if**

22:     **end for**

23:     Obtain interior vertices in the tetrahedral-shape degenerated cubic region using linear interpolation

24: **end for**

---

Based on the **Prism Parametric Mapping Algorithm**, we implemented and organized the code into a CLI program (PrismGen.exe) that can generate prism elements by combining parametric mapping with subdivision. Here, we run the following command to generate the prism elements for the rockerarm model:

```
1 PrismGen.exe -i rockerarm_indexpatch_read.k -p
2 rockerarm_polycube_structure.k -o rockerarm_prism.vtk -s 2
```

There are four options used in the command:

- **-i:** Surface triangle mesh of the input geometry with segmentation information (rockerarm\_indexpatch\_read.k);

- **-o**: Prism mesh (rockerarm\_prism.vtk);
- **-p**: Polycube structure (rockerarm\_polycube\_structure.k); and
- **-s**: Subdivision level.

We use **-i** to input the segmentation file generated in Section 3.1 and use **-p** to input the polycube structure created in Section 3.2. Option **-s** is used to set the level of recursive subdivision. There is no subdivision if we set **-s** to be 0. In the rockerarm model, we set **-s** to be 2 to create a level-2 prism elements in the final mesh. The output prism elements are stored in the *vtk* format (see Fig. 4(a)) and they can be visualized in Paraview [1].

Based on **Tetrahedral Parametric Mapping Algorithm**, we implemented and organized the code into a CLI program (TetGen.exe) that can generate tetrahedral elements by combining parametric mapping with linear subdivision. Here, we run the following command to generate tetrahedral elements for the rockerarm model:

```
1 TetGen.exe -i rockerarm_indexPatch_read.k -p
2 rockerarm_polycube_structure.k -o rockerarm_tet.vtk -s 2
```

There are five options used in the command:

- **-i**: Surface triangle mesh of the input geometry with segmentation information (rockerarm\_indexPatch\_read.k);
- **-o**: Tet mesh (rockerarm\_tet.vtk);
- **-p**: Polycube structure (rockerarm\_polycube\_structure.k); and
- **-s**: Subdivision level.

We use **-i** to input the segmentation file generated in Section 3.1 and use **-p** to input the polycube structure created in Section 3.2. Option **-s** is used to set the level of recursive subdivision. There is no subdivision if we set **-s** to be 0. In the rockerarm model, we set **-s** to be 2 to create a level-2 tetrahedral mesh. The output tetrahedral elements are stored in the *vtk* format (see Fig. 4(b)) and they can be visualized in Paraview.

### 3.4 Quality improvement

We integrate three quality improvement techniques in the software package, namely pillowing, smoothing and optimization. Users can improve mesh quality through the command line options. We first use pillowing to insert one layer of elements around the boundary [63] of the hex elements. By using the pillowing technique, we ensure that each hex element has at most one face on the boundary, which can help improve the mesh quality around the boundary. After pillowing, smoothing and optimization [63] are used to further improve the quality of hex elements. For smoothing, different relocation methods are applied to three types of vertices: vertices on sharp edges of the boundary, vertices on the boundary surface, and interior vertices. For each sharp-edge vertex, we first detect its two neighboring vertices on the curve, and then calculate their middle point. For each vertex on the boundary surface,

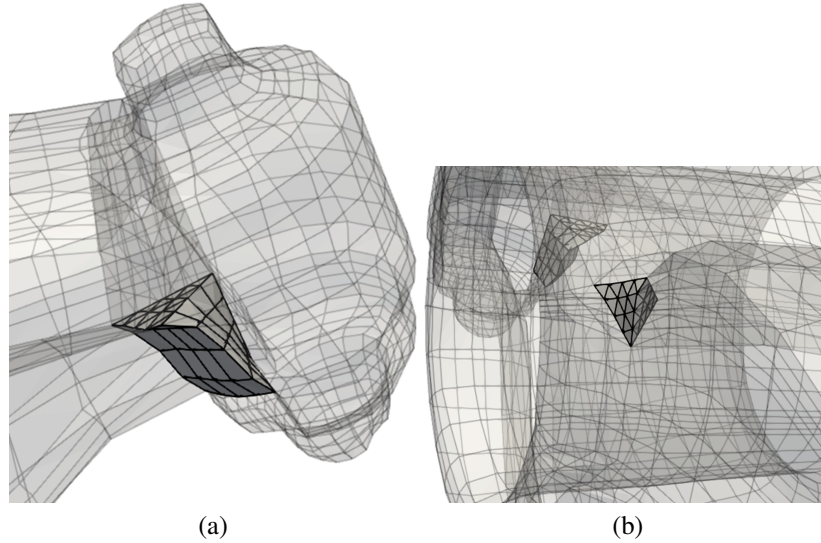


Fig. 4: Prism elements and tetrahedral elements of the rockerarm model (some elements are removed to show the interior of Fig. 1(h)). (a) Prism elements in a prism-shape region; and (b) tetrahedral elements in a tetrahedral-shape region.

we calculate the area center of its neighboring boundary quadrilaterals (quads). For each interior vertex, we calculate the weighted volume center of its neighboring hex elements as the new position. We relocate the vertex iteratively. Each time the vertex moves only a small step towards the new position and this movement is done only if the new location results in an improved local Jacobian. If there are still poor quality hex elements after smoothing, we run the optimization whose objective function is the Jacobian. Each vertex is then moved toward an optimal position that maximizes the worst Jacobian. We presented the **Quality Improvement Algorithm** in [51] for quality improvement.

## 4 HexDom Software and Applications

The algorithms discussed in Sections 3 were implemented in C++. The Eigen library [10] is used for matrix and vector operations. We used a compiler-independent building system (CMake) and a version-control system (Git) to support software development. We have compiled the source code into the following software package:

- **HexDom software package:**
  - **Segmentation module (Segmentation.exe);**
  - **Polycube construction module (Polycube.exe);**

- **Hex-dominant mesh generation module (HexGen.exe, PrismGen.exe, TetGen); and**
- **Quality improvement module (Quality.exe).**

The software is open-source and can be found in the following Github link (<https://github.com/CMU-CBML/HexDom>).

We have applied the software package to several models and generated hex-dominant meshes with good quality. For each model, we show the segmentation result, the corresponding polycube structure, and the hex-dominant mesh. These models include: rockerarm (Fig. 1); two types of mount, hepta and a base with four holes (Fig. 5); fertility, ant, bust, igea, and bunny (Fig. 6). Table 1 shows the statistics of all tested models. We use the scaled Jacobian to evaluate the quality of hex elements. The aspect ratio is used as the mesh quality metric for prism and tet elements which is the ratio between the longest and shortest edges of an element. The aspect ratio is computed with the LS-PrePost, which is a pre and post-processor for LS-DYNA [3]. From Table 1, we can observe that the generated hex-dominant meshes have good quality (minimal Jacobian of hex elements  $> 0.1$ ). Figs. 5-6(a) show the segmentation results of the testing models. Then, we generate polycubes (Figs. 5-6(b)) based on the surface segmentation. Finally, we generate hex-dominant meshes (Figs. 5-6(c)).

Table 1: Statistics of all the tested models.

Model	Input triangle mesh (vertices, elements)	Number of elements			Jacobian (worst) Hex	Aspect ratio (min, max)	
		Hex	Prism	Tet		Prism	Tet
rockerarm (Fig. 1)	(11,705, 23,410)	3,840	704	128	0.20	(1.32, 4.93)	(1.62, 2.98)
mount1 (Fig. 5)	(929, 1,868)	4,224	640	128	0.20	(1.91, 19.57)	(1.67, 3.69)
mount2 (Fig. 5)	(1,042, 2,096)	6,720	1,024	128	0.28	(2.63, 9.54)	(2.30, 4.03)
hepta (Fig. 5)	(692, 1,380)	3,776	1,280	128	0.50	(1.51, 4.48)	(1.63, 3.14)
base (Fig. 5)	(5,342, 10,700)	3,712	384	128	0.34	(1.28, 2.33)	(1.70, 8.31)
fertility (Fig. 6)	(6,644, 13,300)	2,752	320	128	0.20	(2.96, 11.40)	(1.69, 2.69)
ant (Fig. 6)	(7,309, 14,614)	4,480	1,536	128	0.21	(1.16, 6.52)	(1.60, 2.79)
bust (Fig. 6)	(12,683, 25,362)	118,272	20,480	8,192	0.10	(1.35, 48.66)	(1.56, 4.96)
igea (Fig. 6)	(4,532, 9,060)	6,016	3,584	1,024	0.21	(1.61, 12.25)	(1.58, 4.69)
bunny (Fig. 6)	(14,131, 28,258)	2,752	1,472	128	0.20	(1.44, 10.08)	(1.98, 4.18)

## 5 Conclusion and future work

In this paper, we present a new HexDom software package to generate hex-dominant meshes. The main goal of HexDom is to extend the polycube-based method to hex-dominant mesh generation. The compiled software package makes our pipeline accessible to industrial and academic communities for real-world engineering applications. It consists of six executable files, namely segmentation module (Segmentation.exe), polycube construction module (Polycube.exe), hex-dominant mesh generation module (HexGen.exe, PrismGen.exe, TetGen.exe) and quality improvement module (Quality.exe). These executable files can be easily run in the Command Prompt platform. The rockerarm model was used to explain how to run these programs in detail. We also tested our software package using several other models.

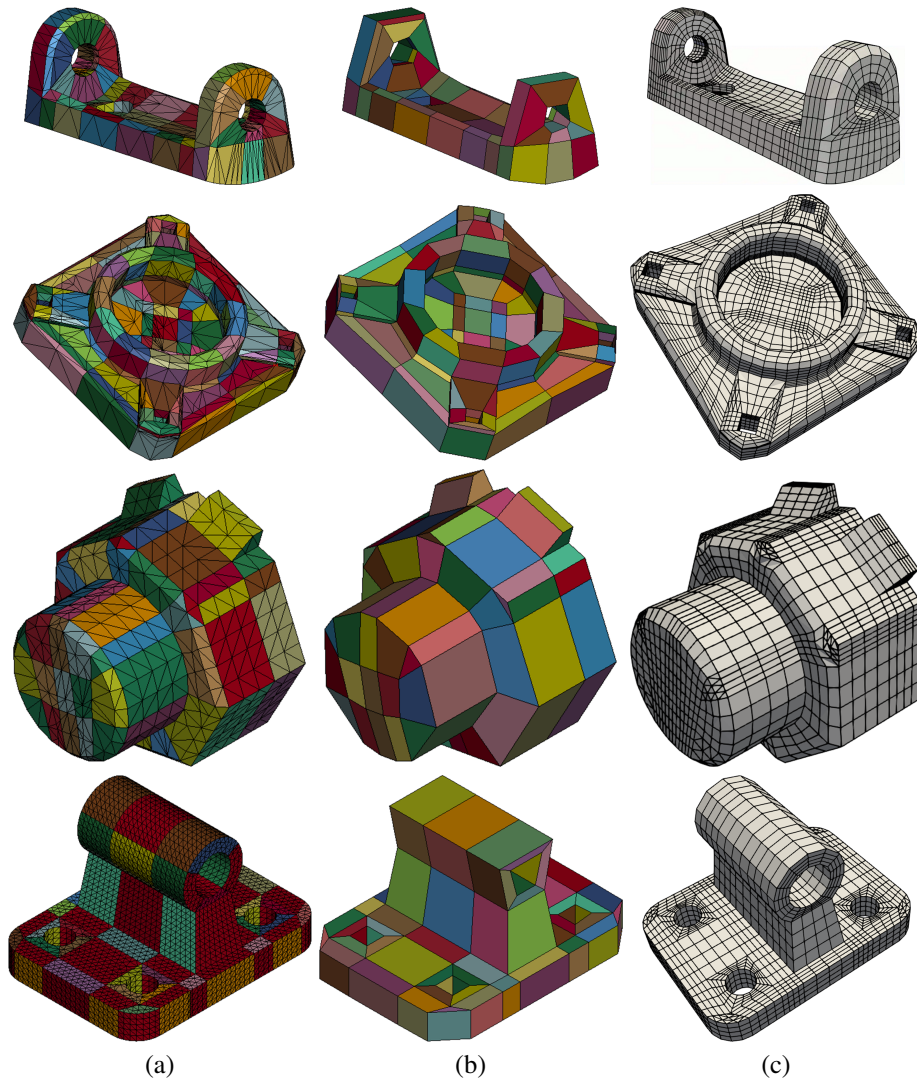


Fig. 5: Results of two types of mount, hepta and a base with four holes. (a) Surface triangle meshes and segmentation results; (b) polycube structures; and (c) hex-dominant meshes.

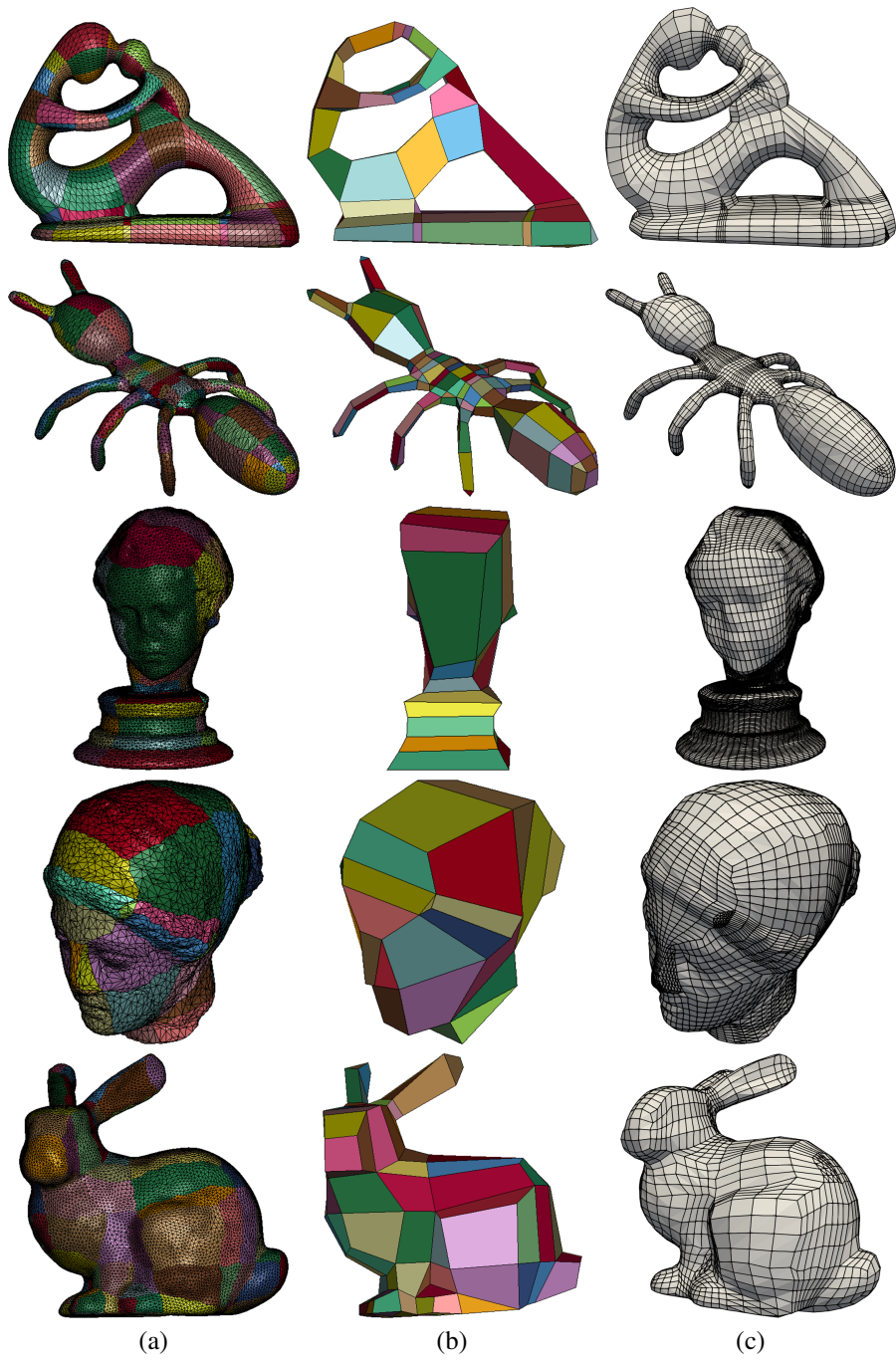


Fig. 6: Results of fertility, ant, bust, igea, and bunny models. (a) Surface triangle meshes and segmentation results; (b) polycube structures; and (c) hex-dominant meshes.

Our software has limitations which we will address in our future work. First, the hex-dominant mesh generation module is semi-automatic and needs user intervention to create polycube structure. Second, the degenerated cubic regions and non-degenerated cubic regions need to be handled separately. We will improve the underneath algorithm and make polycube construction more automatic. In addition, we will also develop spline basis functions for tetrahedral and prism elements to support isogeometric analysis for hybrid meshes.

## Acknowledgment

Y. Yu, J. Liu and Y. Zhang were supported in part by Honda funds. We also acknowledge the open source scientific library Eigen and its developers.

## References

1. Ahrens, J., Geveci, B., Law, C.: Paraview: An end-user tool for large data visualization. *The Visualization Handbook* **717** (2005)
2. Blacker, T.D., Stephenson, M.B.: Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* **32**(4), 811–847 (1991)
3. Corporation, L.S.T.: Ls-dyna keyword user’s manual (2007)
4. Delaunay, B.N.: Sur la sphere vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh I Estestvennykh Nauk* **7**(793-800), 1–2 (1934)
5. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution analysis of arbitrary meshes. In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 173–182 (1995)
6. Floater, M.S.: Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design* **14**(3), 231–250 (1997)
7. Folwell, N., Mitchell, S.: Reliable whisker weaving via curve contraction. *Engineering with Computers* **15**(3), 292–302 (1999)
8. Frey, P.J., Borouchaki, H., George, P.L.: Delaunay tetrahedralization using an advancing-front approach. In: *5th International Meshing Roundtable*, pp. 31–48. Citeseer (1996)
9. Gregson, J., Sheffer, A., Zhang, E.: All-Hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum* **30**(5), 1407–1416 (2011)
10. Guennebaud, G., Jacob, B.: Eigen v3 (2010). <http://eigen.tuxfamily.org>
11. He, Y., Wang, H., Fu, C., Qin, H.: A divide-and-conquer approach for automatic polycube map construction. *Computers & Graphics* **33**(3), 369–380 (2009)
12. Hu, K., Zhang, Y., Liao, T.: Surface segmentation for polycube construction based on generalized centroidal Voronoi tessellation. *Computer Methods in Applied Mechanics and Engineering* **316**, 280–296 (2017)
13. Hu, K., Zhang, Y.J.: Centroidal Voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering* **305**, 405–421 (2016)
14. Hu, K., Zhang, Y.J., Xu, G.: CVT-based 3D image segmentation and quality improvement of tetrahedral/hexahedral meshes using anisotropic Giaquinta-Hildebrandt operator. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* **6**(3), 331–342 (2018)
15. Khan, D., Plopski, A., Fujimoto, Y., Kanbara, M., Jabeen, G., Zhang, Y., Zhang, X., Kato, H.: Surface remeshing: A systematic literature review of methods and research directions. *IEEE Transactions on Visualization and Computer Graphics* (2020). DOI 10.1109/TVCG.2020.3016645



16. Lai, Y., Liu, L., Zhang, Y.J., Chen, J., Fang, E., Lua, J.: Rhino 3D to Abaqus: A T-spline based isogeometric analysis software framework. In: *Advances in Computational Fluid-Structure Interaction and Flow Simulation*, pp. 271–281 (2016)
17. Lai, Y., Zhang, Y.J., Liu, L., Wei, X., Fang, E., Lua, J.: Integrating CAD with Abaqus: A practical isogeometric analysis software platform for industrial applications. *Computers and Mathematics with Applications* **74**(7), 1648–1660 (2017)
18. Li, A., Chai, X., Yang, G., Zhang, Y.J.: An isogeometric analysis computational platform for material transport simulation in complex neurite networks. *Molecular & Cellular Biomechanics* **16**(2), 123 (2019)
19. Liang, X., Zhang, Y.: An octree-based dual contouring method for triangular and tetrahedral mesh generation with guaranteed angle range. *Engineering with Computers* **30**(2), 211–222 (2014)
20. Lin, J., Jin, X., Fan, Z., Wang, C.: Automatic polycube-maps. In: *Advances in Geometric Modeling and Processing, Lecture Notes in Computer Science*, vol. 4975, pp. 3–16. Springer Berlin / Heidelberg (2008)
21. Liu, L., Zhang, Y., Hughes, T.J., Scott, M.A., Sederberg, T.W.: Volumetric T-spline construction using Boolean operations. *Engineering with Computers* **30**(4), 425–439 (2014)
22. Liu, L., Zhang, Y., Liu, Y., Wang, W.: Feature-preserving T-mesh construction using skeleton-based polycubes. *Computer Aided Design* **58**, 162–172 (2015)
23. Lohner, R., Parikh, P.: Three-dimensional grid generation by the advancing front method. *International Journal for Numerical Methods in Fluids* **8**, 1135–1149 (1988)
24. Meshkat, S., Talmor, D.: Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. *International Journal for Numerical Methods in Engineering* **49**(1-2), 17–30 (2000)
25. Meyers, R.J., Tautges, T.J., Tuchinsky, P.M.: The "hex-tet" hex-dominant meshing algorithm as implemented in cubit. In: *International Meshing Roundtable*, pp. 151–158. Citeseer (1998)
26. Nieser, M., Reitebuch, U., Polthier, K.: Cubecover - parameterization of 3D volumes. *Computer Graphics Forum* **30**(5), 1397–1406 (2011)
27. Owen, S.J.: A Survey of Unstructured Mesh Generation Technology. In: *International Meshing Roundtable*, Dearborn, MI, vol. 194, pp. 4135–4195 (1998)
28. Owen, S.J., Saigal, S.: H-morph: An indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering* **49**(1-2), 289–312 (2000)
29. Pan, Q., Xu, G., Zhang, Y.: A unified method for hybrid subdivision surface design using geometric partial differential equations. A Special Issue of Solid and Physical Modeling 2013 in *Computer Aided Design* **46**, 110–119 (2014)
30. Price, M.A., Armstrong, C.G.: Hexahedral mesh generation by medial surface subdivision: Part II. Solids with flat and concave edges. *International Journal for Numerical Methods in Engineering* **40**(1), 111–136 (1997)
31. Price, M.A., Armstrong, C.G., Sabin, M.A.: Hexahedral mesh generation by medial surface subdivision: Part I. Solids with convex edges. *International Journal for Numerical Methods in Engineering* **38**(19), 3335–3359 (1995)
32. Qian, J., Zhang, Y.: Automatic unstructured all-hexahedral mesh generation from B-Reps for non-manifold CAD assemblies. *Engineering with Computers* **28**(4), 345–359 (2012)
33. Qian, J., Zhang, Y., O'Connor, D.T., Greene, M.S., Liu, W.K.: Intersection-free tetrahedral meshing from volumetric images. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* **1**(2), 100–110 (2013)
34. Qian, J., Zhang, Y., Wang, W., Lewis, A.C., Qidwai, M.A.S., Geltmacher, A.B.: Quality improvement of non-manifold hexahedral meshes for critical feature determination of microstructure materials. *International Journal for Numerical Methods in Engineering* **82**(11), 1406–1423 (2010)
35. Schneiders, R.: A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers* **12**(3-4), 168–177 (1996)
36. Schneiders, R.: An algorithm for the generation of hexahedral element meshes based on an octree technique. 6th *International Meshing Roundtable* pp. 195–196 (1997)

37. Seveno, E., et al.: Towards an adaptive advancing front method. In: 6th International Meshing Roundtable, pp. 349–362 (1997)
38. Shephard, M.S., Georges, M.K.: Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical methods in engineering* **32**(4), 709–749 (1991)
39. Staten, M., Kerr, R., Owen, S., Blacker, T.: Unconstrained paving and plastering: Progress update. *Proceedings of 15th International Meshing Roundtable* pp. 469–486 (2006)
40. Tarini, M., Hormann, K., Cignoni, P., Montani, C.: Polycube-maps. *ACM Transactions on Graphics* **23**(3), 853–860 (2004)
41. Teng, S.H., Wong, C.W.: Unstructured mesh generation: Theory, practice, and perspectives. *International Journal of Computational Geometry & Applications* **10**(3), 227–266 (2000)
42. Wang, W., Zhang, Y., Liu, L., Hughes, T.J.R.: Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology. *Computer Aided Design* **45**(2), 351–360 (2013)
43. Wang, W., Zhang, Y., Scott, M.A., Hughes, T.J.R.: Converting an unstructured quadrilateral mesh to a standard T-spline surface. *Computational Mechanics* **48**(4), 477–498 (2011)
44. Wang, W., Zhang, Y., Xu, G., Hughes, T.J.R.: Converting an unstructured quadrilateral/hexahedral mesh to a rational T-spline. *Computational Mechanics* **50**(1), 65–84 (2012)
45. Wei, X., Zhang, Y., Hughes, T.J.R.: Truncated hierarchical tricubic  $C^0$  spline construction on unstructured hexahedral meshes for isogeometric analysis applications. *Computers and Mathematics with Applications* **74**(9), 2203–2220 (2017)
46. Wei, X., Zhang, Y.J., Toshniwal, D., Speleers, H., Li, X., Manni, C., Evans, J.A., Hughes, T.J.: Blended B-spline construction on unstructured quadrilateral and hexahedral meshes with optimal convergence rates in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* **341**, 609–639 (2018)
47. Xie, J., Xu, J., Dong, Z., Xu, G., Deng, C., Mourrain, B., Zhang, Y.J.: Interpolatory Catmull-Clark volumetric subdivision over unstructured hexahedral meshes for modeling and simulation applications. *Computer Aided Geometric Design* **80**, 101867 (2020)
48. Xu, G., Ling, R., Zhang, Y.J., Xiao, Z., Ji, Z., Rabczuk, T.: Singularity structure simplification of hexahedral meshes via weighted ranking. *Computer-Aided Design* **130**, 102946 (2021)
49. Yamakawa, S., Shimada, K.: Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. *International Journal for Numerical Methods in Engineering* **57**(15), 2099–2129 (2003)
50. Yu, Y., Liu, H., Qian, K., Yang, H., McGehee, M., Gu, J., Luo, D., Yao, L., Zhang, Y.J.: Material characterization and precise finite element analysis of fiber reinforced thermoplastic composites for 4D printing. *Computer-Aided Design* **122**, 102817 (2020)
51. Yu, Y., Wei, X., Li, A., Liu, J., He, J., Zhang, Y.J.: HexGen and Hex2Spline: Polycube-based hexahedral mesh generation and spline modeling for isogeometric analysis applications in LS-DYNA. Springer INdAM Series: Proceedings of INdAM Workshop "Geometric Challenges in Isogeometric Analysis." (2021)
52. Yu, Y., Zhang, Y.J., Takizawa, K., Tezduyar, T.E., Sasaki, T.: Anatomically realistic lumen motion representation in patient-specific space–time isogeometric flow analysis of coronary arteries with time-dependent medical-image data. *Computational Mechanics* **65**(2), 395–404 (2020)
53. Zhang, Y.: Challenges and advances in image-based geometric modeling and mesh generation. In: *Image-Based Geometric Modeling and Mesh Generation*, pp. 1–10. Springer (2013)
54. Zhang, Y.: *Geometric Modeling and Mesh Generation from Scanned Images*. Chapman and Hall/CRC (2016)
55. Zhang, Y., Bajaj, C.L.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering* **195**(9-12), 942–960 (2006)
56. Zhang, Y., Bajaj, C.L., Sohn, B.S.: 3D Finite Element Meshing from Imaging Data. *Computer Methods in Applied Mechanics and Engineering* **194**(48-49), 5083–5106 (2005)
57. Zhang, Y., Bajaj, C.L., Xu, G.: Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Communications in Numerical Methods in Engineering* **25**(1), 1–18 (2009)

58. Zhang, Y., Bazilevs, Y., Goswami, S., Bajaj, C.L., Hughes, T.J.R.: Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering* **196**(29-30), 2943–2959 (2007)
59. Zhang, Y., Hughes, T.J.R., Bajaj, C.L.: An automatic 3D mesh generation method for domains with multiple materials. *Computer Methods in Applied Mechanics and Engineering* **199**(5-8), 405–415 (2010)
60. Zhang, Y., Liang, X., Ma, J., Jing, Y., Gonzales, M.J., Villongco, C., Krishnamurthy, A., Frank, L.R., Nigam, V., Stark, P., Others: An atlas-based geometry pipeline for cardiac Hermite model construction and diffusion tensor reorientation. *Medical Image Analysis* **16**(6), 1130–1141 (2012)
61. Zhang, Y., Liang, X., Xu, G.: A robust 2-refinement algorithm in octree and rhombic dodecahedral tree based all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering* **256**, 562–576 (2013)
62. Zhang, Y., Qian, J.: Resolving topology ambiguity for multiple-material domains. *Computer Methods in Applied Mechanics and Engineering* **247**, 166–178 (2012)
63. Zhang, Y., Wang, W., Hughes, T.J.R.: Solid T-spline construction from boundary representations for genus-zero geometry. *Computer Methods in Applied Mechanics and Engineering* **249**, 185–197 (2012)
64. Zhang, Y., Wang, W., Hughes, T.J.R.: Conformal solid T-spline construction from boundary T-spline representations. *Computational Mechanics* **51**(6), 1051–1059 (2013)