Hexahedral Meshing with Varying Element Sizes

Kaoji Xu[1] Xifeng Gao[2] Zhigang Deng[1] and Guoning Chen[1]

[1]Department of Computer Science, University of Houston, USA
[2]Department of Computer Science, New York University, USA

Abstract
Hexahedral (or Hex-) meshes are preferred in a number of scientific and engineering simulations and analyses due to their desired numerical properties. Recent state-of-the-art techniques can generate high quality hex-meshes. However, they typically produce hex-meshes with uniform element sizes and thus may fail to preserve small scale features on the boundary surface. In this work, we present a new framework that enables users to generate hex-meshes with varying element sizes so that small features will be filled with smaller and denser elements, while the transition from smaller elements to larger ones is smooth, compared to the octree-based approach. This is achieved by first detecting regions of interest (ROI) of small scale features. These ROIs are then magnified using the as-rigid-as-possible (ARAP) deformation with either an automatically determined or a user-specified scale factor. A hex-mesh is then generated from the deformed mesh using existing approaches that produce hex-meshes with uniform-sized elements. This initial hex-mesh is then mapped back to the original volume before magnification to adjust the element sizes in those ROIs. We have applied this framework to a variety of man-made and natural models to demonstrate its effectiveness.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Modeling—Mesh Generation

1. Introduction

Hexahedral meshes are preferred by a number of scientific and engineering applications due to their desired numerical properties. For instance, hexahedra offer better alignment with solution features than tetrahedra [YMD11]. However, given any input models, generating hex-meshes with good quality elements and simple structure while conforming to the surface configuration remains an open challenge. Recently, a number of effective techniques have been introduced to help generate high quality hex-meshes from various input objects. However, they typically aim to generate hex-meshes with uniform sizes and may not preserve small features. If small features need to be preserved, a very dense hex-mesh will be resulted.

Recent research in high-order discretizations of partial differential equations also suggests the need for mesh adaptivity to reduce solution error at a reasonable computational cost [YMD11]. This is because using smaller elements in regions of interest (e.g., small scale surface features) helps reduce the discretization error, and thus, improves the simulation accuracy and convergence rate. Using relatively large elements in regions of less interest helps keep the total number of elements small, leading to a faster computation. See Section 4 for an example of the linear elasticity problem. To address this need of mesh adaptivity, a few methods (e.g., octree-based approaches [ZB06, Mar09, ZLX13]) are proposed to preserve small surface features using element subdivision strategies, which often requires a few levels of subdivision in order to preserve small surface features, resulting in hex-meshes with a large number of elements and non-structurally consistent transition at the interface of two levels of the adaptive mesh (i.e., the introduction of a large number of irregular elements). See Figure 1a for an example. These excessive irregular elements result in a complex mesh structure (Figure 1d), complicating subsequent computations that prefer a simpler mesh structure, such as Isogeometric Analysis (IGA) [HT05]. Here, the mesh structure is the *base complex* of the mesh, which is a partitioning of the volume produced by the separation surfaces traced out from the *singularities* of the mesh (i.e., the sets of connected irregular edges). See the bottom row of Figure 1 for some examples. That said, there is a need of a technique that enables the control of hex-element sizes in order to preserve small surface features while maintaining a reasonable element number and a simple mesh structure (i.e., with fewer hexahedral components). All the above reasons motivate our work.

Overview: Our goal is to fill with smaller elements in the volumetric regions adjacent to the small surface features, while the transition between elements with different sizes largely preserves the consistency of mesh structure (i.e., fewer irregular elements are introduced).

Since the aforementioned adaptive strategy via element subdivision as a post-processing will increase the number of elements exponentially (i.e., one hex splits to eight) and introduce numerous irregular elements, we seek for a pre-processing solution. Our approach is inspired by an anisotropic quadrangulation technique

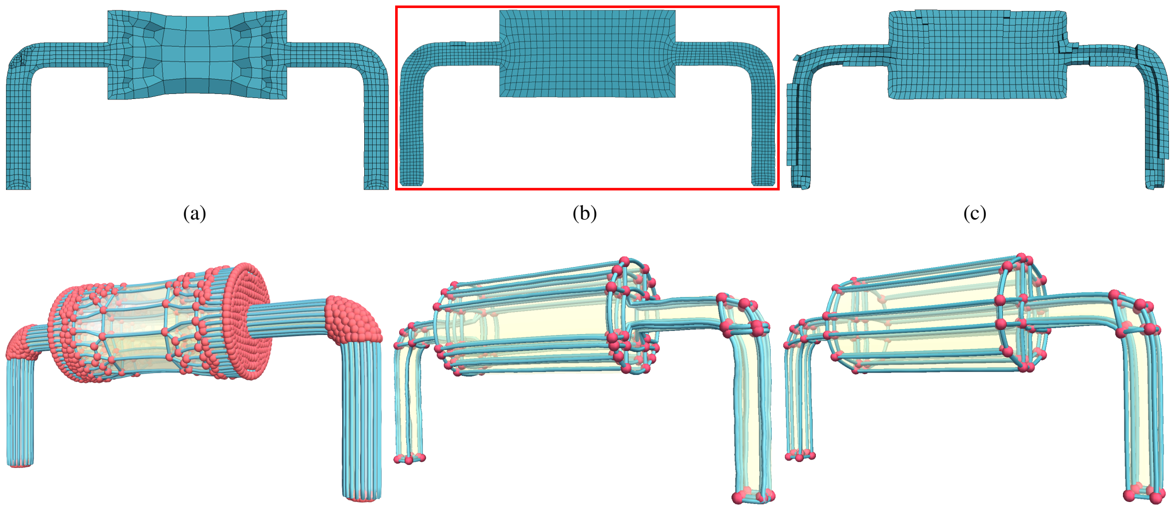


Figure 1: *The hex-meshes of a pipe model generated using an octree-based method [Mes15] (a), our proposed method (b), ℓ_1-polycube approach [HJS*14] (c), respectively. (d-f) show their respective base-complexes. These base-complexes partition their respective hex-meshes into a hexahedral structure, consisting of large hexahedral (or cuboid-like) components. The red dots are the corners of the components and the cyan curves are their edges.*

recently introduced by Panozzo et al. [PPTSH14], in which the surface is deformed based on the surface characteristics to induce a uniform cross field that is later mapped back to the original surface to obtain a general frame field for the generation of an anisotropic quad-mesh. Similarly, if we extract a hex-mesh with uniform size elements from a volume whose sub-regions adjacent to small surface features, referred to as the regions of interest (ROIs), are magnified, the hex-mesh projected back to the original volume will have smaller elements in those ROIs.

To achieve the above goal, we propose a novel and general adaptive hex-meshing framework. First, we detect regions of interest (ROIs) of small scale features (Section 3.1). Second, these ROIs are enlarged with either an automatically determined or a user-specified scale factor to obtain a deformed volume (Section 3.2). Third, a valid all-hex mesh is constructed from this deformed volume, which is mapped back to the volume space enclosed by the original surface (Section 3.3). Finally, to ensure the outcome of a high-quality hex-mesh while preserving the desired element sizes, an effective optimization is performed (Section 3.5). Our framework is simple and intuitive to use. In addition, the user is able to semi-automatically choose ROIs for magnification to address different needs of specific applications, such as the generation of anistropic meshes. Our framework is also flexible so that different techniques can be adapted to accomplish the above individual steps. Figure 2 illustrates the pipeline that is implemented in this work. We have applied our framework to a number of man-made and natural-shape models to demonstrate its effectiveness (Section 4). Figure 1 provides a comparison of our approach with the state-of-the-art ℓ_1-polycube approach and an adaptive octree-based method. From this comparison, we see that our method can generate hex-meshes with element sizes better adapting to the feature size than existing methods that tend to generate hex-meshes with uniform element size, while having a comparable base complex structure that is much simpler than the adaptive octree-based method.

2. Related Work

A wide variety of methods exist to generate a hexahedral mesh from an input triangle mesh [SJ08]. However, since the goal of this work is to produce hex-meshes with elements adaptive to the surface feature sizes while having as simple as possible structures, we only focus on methods that may have the potential to achieve this goal.

Octree-based: Octree-based or grid-based methods typically first fill the interior volume of the input model with a Cartesian grid, then connect the grid to the surface using a variety of methods [Sch96, Mar09]. For instance, Ito et al. [ISS09] introduced an octree-based hex-mesh generation based on a set of refinement templates to perform local refinement. However, these octree-based methods often need excessively fine local element sizes on small scale or concave features and may form low quality elements with irregular connectivities (i.e., introducing a large amount of irregular elements) in those areas [ZB06, ZLX13]. These irregular elements correspond to the singularities of a 3D volumetric parameterization that the hex-grids correspond to. These singularities help define the base complex of the hex-mesh. A large number of singularities may result in a base complex with many hexahedral components, posing challenges for the fitting of basis functions with higher-order smoothness for the subsequent simulations [GDC15, GMD*16]. The recently proposed adaptive hex-meshing techniques [ZZ07, SZ16] based on a similar framework to the octree-based approach still do not address the issue of introducing an excessive number of singularities.

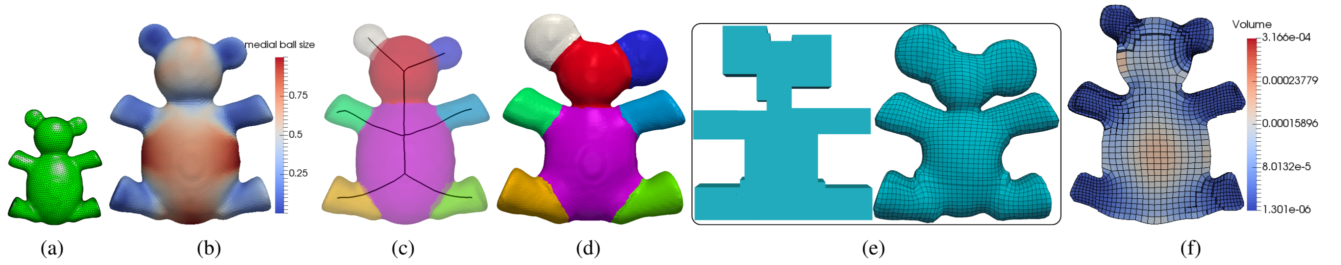


Figure 2: *Pipeline of our framework. Given an input of tetrahedral or triangular mesh (a), we first compute a sizing scalar function(b) to identify the locations with small features. The technique in [TAOZ12] is a good candidate for this task. The feature size of each segment (c) is measured by the medial ball size as visualized in (b). Here blue indicates places with small features, while red corresponds to large segments. The small segments, identified as the ROIs based on their medial ball sizes, are magnified (d). Then, any existing hex-meshing method (e.g., polycube based, frame field based or skeleton-based) can be applied to extract an initial hex-mesh with elements of uniform size (e) in the domain of (d). In this paper we use the ℓ_1-polycube-based method [HJS* 14]. By mapping (e) back to the original volume before deformation, we obtain the output hex-mesh (f) with varying element sizes.*

Polycubes: Polycube map, originally proposed in [THCM04], creates a parametric domain by assembling cubes which are further mapped bijectively onto the object. Polycubes allow the decomposition of an object into a set of larger hexahedral pieces. However, the quality of the resulting hexahedral representation strongly depends on the placement of polycube corners *on* the input triangle mesh to achieve a low-distortion mapping between the polycube representation and the input. This challenging process has received a lot of attention recently [GSZ11, WLL*12]. Automatic methods are usually difficult to control. Livesu et al. [LVS*13] and Huang et al. [HJS*14] introduced Polycuts and ℓ_1−polycubes, respectively, to improve the corner configuration of the conventional polycube to remove unnecessary small cubes. Li et al. [LLWQ13] extended the conventional polycube to a generalized polycube (or GPC), which enables the curved cuboid representation of the elementary subvolumes decomposed via shape analysis. Cherchi et al. [CLS16] simplified the layouts of polycube to get a simpler structure. Recently, Fang et al. [FXBH16] further introduced the closed-Form induced polycubes, which can be applied to more types of models and can generate hex-meshes with layout better aligned with the surface features.

Different from the octree-based method, Kremer et al. [KBLK14] introduced an automatic technique that generate hex-meshes from valid surface quad meshes based on the concept of dual loops. Additional singularities may still be introduced during the construction of a valid hexahedral topology.

Frame field guided approaches: Frame fields, especially cross fields, have been proven useful to assist the placement of quadrilateral elements when quadrangulating a triangular mesh [BZK09], as it provides consistent local frame information everywhere in the domain to guide the orientation of parameterization. Huang et al. [HTZ*11] proposed a first solution to creating a boundary conformal 3D cross field via an expensive optimization. Due to insufficient control on the types of the singularities in the cross field, their approach cannot guarantee to generate an all-hex mesh. Nieser et al. [NRP11] pointed out that only 10 types of singularities can lead to a valid all-hex mesh. Recently, Li et al. [LLX*12] introduced the *Singularity Restricted Field (SRF)* that converts a general 3D cross field to the one that contains only these 10 types of singularities. After regularizing the SRF and fixing degeneracies, high quality hex-meshes can then be generated using the Cube-Cover technique [NRP11]. A similar work by Jiang et al. [JHW*14] also aims to derive a restricted cross field from some initial cross field, which is then used to generate hex-meshes by solving a mixed-integer problem. Despite generating hex-meshes with superior quality, frame field based methods are typically less robust, compared to the polycubes approach, and can easily lead to degeneracy in the parameterization [LLX*12, GMD*16].

On the other hand, a metric-driven frame field generation technique was introduced by Jiang et al. [JFH*15] recently, which may be used to achieve the generation of anisotropy quad-meshes. Similarly, Nieser et al. [NPPZ12] introduced a hexagonal global parameterization guided by the shape-aware 6-RoSy field, which can be used to achieve adaptive triangle meshes. Nonetheless, there is no evidence showing that either method can be directly applied to generate adaptive volumetric meshes.

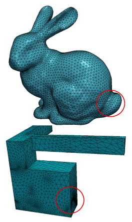
Skeleton and sweeping based approaches: Gao et al. [GMD*16] proposed a structure volume decomposition technique based on a generalized sweeping strategy, which can be used to generate multi-resolution hex-meshes with much simpler structure. In a similar spirit, Livesu et al. [LMPS16] proposed a skeleton-driven adaptive hex-meshing strategy, which is similar to a previous work by Lin et al. [LLD12]. Liu et al. [LZLW15] proposed a T-mesh construction technique using a skeleton-based polycubes. However, all these skeleton-based approaches are highly constrained by the types of models that can be handled.

While the existing polycubes based, frame field based and skeleton based approaches can generate hex-meshes with much simpler structure compared to the octree-based methods, their generated hex-meshes typically possess elements of uniform size. That said, in order to capture small scale surface features, sufficiently dense hex-meshes (i.e., with much more elements) may be needed, which will significantly increase the cost of subsequent computation. Our work effectively addresses this challenge by adding a pre-processing step that magnifies the small surface features before computing an initial hex-mesh with uniform element size. This enables the generation of hex-meshes with denser hex-grids in the volume regions near the small surface features while still having a simple structure.

3. Our Method

Our proposed framework consists of the following major steps (illustrated in Figure 2).

1. We first segment the surface of the input model and compute the size information for each segment. The regions of interest (ROIs) are identified as the segments that have small feature sizes.
2. Given the above segmentation and the identified ROIs with their size information, we magnify them using an as-rigid-as-possible deformation (Figure 2d).
3. Next, we apply any of the existing methods to produce an initial hex-mesh with uniform-sized elements (Figure 2e) from the deformed mesh obtained in the preceding step. In this work, we mostly apply the ℓ_1-polycube method. The obtained hex-mesh is then mapped back to the original volume before deformation to adjust the element sizes in those ROIs (Figure 2f).



Note that users might select the polycube hex-meshing method and perform the magnification on the obtained polycubes instead of the input mesh. However, given the selection of the parameters, the polycubes map deformation may classify the small features with their nearby portion of the surface (e.g., the tail of the bunny is lost in the polycube of the bunny model shown in the inset) or produce polycubes with artifacts (e.g., degenerate elements in Figure 11c) due to the small sizes. To overcome this difficulty, we opt for magnifying the input surface before computing its polycubes map. In fact, this strategy can in turn help handle certain models that the polycubes approach alone may not be able to handle. See the horse model shown in Figure 11 for an example.

In the following, we provide more details on the individual steps.

3.1. Mesh Segmentation

Our goal is to fill with denser hexahedral elements in the regions adjacent to small surface features (e.g., legs of a chair, ears of a bunny, and the tail of a kitten). There are a number of approaches to identify small surface features (please refer to [CGF09] for a comprehensive survey). Considering the need of the size or scale information of each segment, in this work, we opt for a skeleton guided segmentation approach proposed by Tagliasacchi et al. [TAOZ12]. This approach is simple yet robust. In the following, we briefly describe this approach for the sake of readability.

Skeleton-based segmentation: Given an input surface triangle mesh, its skeleton is first extracted using a mean curvature flow method. This method constructs an energy function which contains a term to contract the mesh, a term to attract it to the geometry position of structure and a term to adjust the skeleton to the medial axis position. Iteratively smoothing and updating the weight of each term makes the 3D graph (initially the triangle mesh) converge to a zero-volume degenerate geometry (i.e., 1D skeleton). The obtained skeleton is represented as a 1D graph of curvilinear structure. Via edge collapsing, the surface vertices are associated with the vertices of the skeleton (also called the skeletal points). In the meantime, the distance of each vertex on the 1D skeleton to the input mesh (i.e., the medial ball size) is also obtained during the contraction. With the association of surface vertices and skeletal points, the input surface triangle mesh can be segmented by first computing a shape diameter function for each face of the input mesh, followed by solving a graph cut problem. The shape diameter function for each face is computed as the average distance of its three vertices to their corresponding skeletal points [SSCO08]. The output of the segmentation is a map in which each triangle face of the input mesh is associated to a segmentation label and a medial ball size. This information will be used for the subsequent mesh deformation, since small surface features correspond to small medial ball sizes.

In practice, in order to obtain more desired segmentation and accurate skeleton geometry and medial ball sizes using the above approach, we need to increase the resolution of the input triangle mesh (see Figure 3(b)). However, increasing the resolution of the input mesh will significantly increase the computation cost of the subsequent processes (e.g., ℓ_1-polycube deformation and hex-mesh extraction). To balance the accuracy of the segmentation and the computation cost, we experimentally choose to use meshes with triangle elements ranging from $2K$ to $25K$ for different test models given their characteristics.

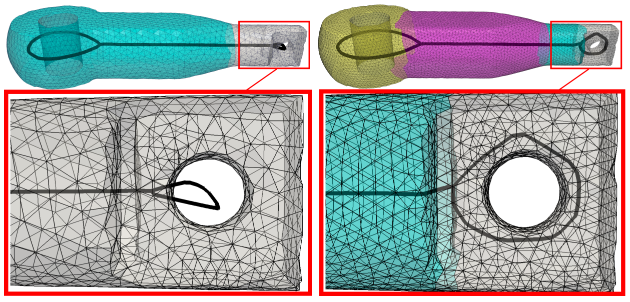

(a) 5k triangles (b) 13k triangles

Figure 3: *Compared to the coarse mesh (a), a denser input triangle mesh will produce more desired segmentation and skeleton (b).*

Identify ROIs: With the segmentation and the size information of each segment, the ROIs can be selected either manually by the users (see Figure 13) or automatically. After the above segmentation, each triangle t_i of the input mesh is associated with a unique medial ball size s_i and a segmentation label j (i.e., it belongs to the j^{th} segment). For each segment, we estimate its feature size $\bar{s}_j$ as the average of the medial ball sizes of all the triangles of this segment. In the meantime, an average feature size of the entire mesh $\bar{s}$ can be computed by averaging all the medial ball sizes. We then choose all the segments that satisfy $\bar{s}_j < \bar{s}$ as the regions of interest (ROIs) automatically.

3.2. Mesh Magnification

After identifying the ROIs with the help of segmentation, the next step is to magnify them. Our goal is to enlarge the selected ROIs in proportion to their feature sizes, while maintaining the non-ROI regions unchanged. This is required in order to fill the hex-elements with the sizes proportional to the feature sizes. Direct magnification of ROIs may produce large distortions and abrupt changes between the magnified regions and the other parts of the mesh. To achieve smooth transition, many surface deformation approaches have been proposed [WLT08, ZZG*12]. [WLT08] achieves the magnification of a user-specified region (i.e., within a magic lens) via the enlarging of the grid cells enclosing this region. To adapt this approach for our problem, we need to generate a dense enough (or even adaptive) grid for the accurate enclosure of ROIs. This dense grid could lead to slow computation. The approach in [ZZG*12] is global and may also deform the other regions that are not ROIs, which is different from our goal.

In this work, we adopt the as-rigid-as-possible (ARAP) deformation approach [ACOL00, SA07, CPSS10]. ARAP deformation has been widely applied in many surface deformation and editing applications. A general pipeline of ARAP deformation is that the user first select a set of vertices for deformation. Then, the user modifies (or moves) one or more of these vertices. Based on the new vertex positions, the ARAP deformation computes the new positions of the entire mesh with regions far away from the edited vertices largely unchanged. However, our goal is to *automatically* magnify the ROIs, including both their surface portions and the enclosed volumes. To utilize ARAP deformation to achieve our goal, we need to develop an approach to estimate the new positions of the vertices within each ROI. In addition, this estimation of the new vertex positions should take into account the information of the feature size of a given ROI. In the following, we describe our solution to this estimation.

Determine bounding boxes and scaling factors for ROIs: First, we compute the bounding box B for each ROI whose principle direction indicates the deformation direction. For each ROI, R, there could have two possible different configurations: 1) R is connected with only one other segment (e.g., the gray and yellow segments shown in Figure 3b); and 2) R is connected with two or more other segments (e.g., the tail of the kitten). In the rest of the paper, we denote the former by $\mathbf{S}_1$ and the latter by $\mathbf{S}_2$, respectively. For $\mathbf{S}_1$, we first extract the boundary of R, which is a simple 1D curve. We then fit a plane P to the vertices of this boundary curve using a least square fitting (Figure 4a). Next, we project all vertices V of R onto P, and compute a bounding rectangle B_S on P as well as its center C based on the projected vertices V_P. B_S is then the base of the bounding box B, whose height is the distance of a vertex in V that is the farthest away from P (Figure 4b). The principal direction of B is then the direction that is perpendicular to the base plane B_S. For $\mathbf{S}_2$, since there are two or more segment boundaries, for simplicity we choose the boundary that connects R with the segment whose medial ball size is the largest among all the neighboring segments of R to fit the plane P. After obtaining P, the bounding box of R can be estimated similarly to $\mathbf{S}_1$. Again, the principal direction of the obtained bounding box B is the direction that is perpendicular to the base plane B_S computed on P.

Second, we determine the scaling factor for the magnification. Since each ROI R has an estimated feature size $\bar{s}_j$, it is natural to compute a scaling factor $sf_j = \bar{s}/\bar{s}_j$ for R. In some situations, while sf_j may not provide a sufficiently large value for the magnification, our framework enables the user to specify the desired scaling factor to compensate that. In addition, different scaling factors may be assigned to different ROIs to address the needs of specific applications. Figures 7 and 13 provide such examples.

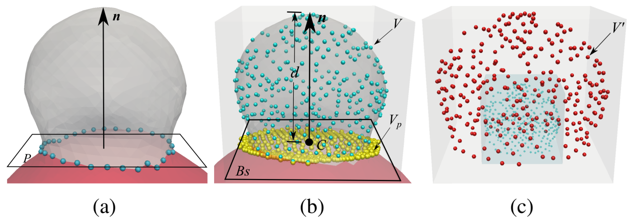


Figure 4: *(a) Fitting a plane P to the boundary vertices (cyan) of an ROI (gray). (b) Projecting the vertices V of the ROI to the plane P to fit a bounding rectangle B_S. (c) Mapping the cyan vertices V of ROI to a magnified cube to obtain their new positions V' (red).*

Magnify ROIs: With the estimated bounding box and its scaling factor of an ROI, R, we now can compute the magnified positions of the vertices in R. Specifically, we first subdivide the bounding box into five tetrahedra [HT89]. Each vertex in R is located in one of these five tetrahedra. We then compute the barycentric coordinates of each vertex with respect to the tetrahedron where it is located. The new position of each vertex is then obtained from the magnified bounding box based on its corresponding tetrahedron and its barycentric coordinates (Figure 4c). The bounding box B is scaled uniformly based on the scaling factor in our experiments. That is, the bounding box is enlarged along the principal direction of B. These new positions are then fed to the ARAP deformation to update the entire mesh. Note that while this simple uniform scaling may result in the new positions of the vertices intersecting with the other parts of the surfaces, it can be effectively resolved by ARAP deformation with proper self-intersection prevention.

To facilitate the subsequent hex-mesh extraction, a tetrahedral mesh is first computed for the input triangle mesh and is then deformed after the ARAP deformation which updates only the surface vertex positions. In particular, we compute the mean value coordinates [JSW05] for each interior vertex with respect to the corresponding surface vertices. Therefore, the update of the surface vertices will result in the update of the positions of the interior vertices. Alternatively, we can apply a volumetric ARAP deformation to update the interior vertices together with the surface vertices [CPSS10]. However, due to the volume preservation in the volumetric ARAP, we find that the transition between ROIs and non-ROIs in the resultant mesh obtained with a surface ARAP is generally smoother than the one obtained from the volumetric ARAP. See Figure 5 for a comparison.

We wish to point out that, a large scaling factor (i.e., $\bar{s}_j$ is much smaller than $\bar{s}$) may cause large distortion at the boundary of the corresponding ROI (see Figure 6c). To avoid this, we recommend that $\bar{s}_j$ after magnification be smaller than $\bar{s}$. Nonetheless, even with

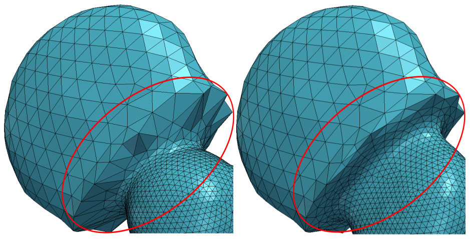


Figure 5: *Volumetric ARAP (left) vs Surface ARAP (right). Red areas show that the transition on the mesh obtained from the surface ARAP is smoother than the one from the volumetric ARAP.*

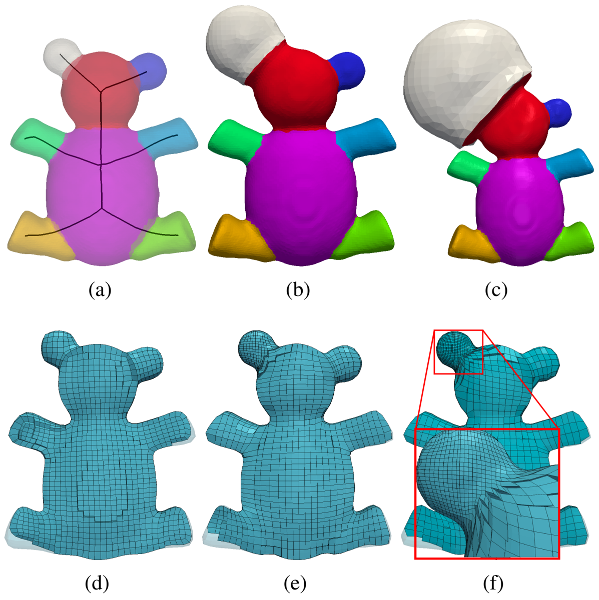


Figure 6: *(a) shows the teddy bear model before deformation. (b) shows the magnification of the left ear of the teddy bear model with a scaling factor of 2.0, while (c) shows the result with a scaling factor 4.0. (d) (e) and(f) are the corresponding hex-meshes of (a) (b) and (c), respectively.*

such a large distortion at the boundary of the ROIs, our pipeline can still generate a valid all-hex mesh with a structure much simpler than the one produced by the octree-based approach. See Figure 6f for an example.

Note that other deformation techniques, such as the frame field driven deformation [PPTSH14], can be potentially applied here to properly magnify the regions containing small scale features. We will provide more discussions on this in the later section.

Anisotropic meshing: With the above deformation framework, anisotropic hex-meshes can be generated. Specifically, instead of performing a deformation with a uniform scaling factor in the three principal directions of the bounding box of an ROI, we can choose different scaling factors for different directions. Figure 7 shows an example, in which the two ears of the bear are magnified with different scaling factors in the three principal directions, i.e., the left ear with $9 \times 4 \times 1$ and the right one with $6 \times 3 \times 2$, respectively. This non-uniform deformation results in hex-elements with different anisotropy orientations in the regions of the two ears. Although this example demonstrates the possibility of generating anisotropy hex-meshes with our framework, in general, the scaling factors of the three principal directions can be automatically computed so that the original bounding box will be deformed into a regular cube after scaling. We plan to explore this further in the future work.

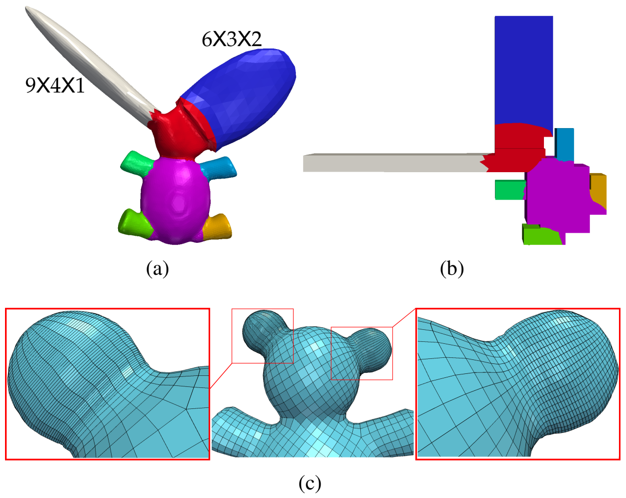


Figure 7: *(a) shows the deformed teddy bear model with two different sets of anisotropy scaling factors for its two ears (i.e., $9 \times 4 \times 1$ for the left ear and $6 \times 3 \times 2$ for the right ear). (b) shows the polycube maps of (a), while (c) shows different anisotropic hex-elements in the two ears.*

3.3. Hex-mesh Extraction

After computing the deformed mesh with the magnified ROIs, we now extract a hex-mesh within its volume. Any efficient hex-meshing techniques can be applied here. However, considering the structure simplicity of the resulting hex-meshes and the robustness of the computation, we opt for the current state-of-the-art polycubes technique, i.e., the ℓ_1−polycube method [HJS*14] for this task.

The ℓ_1-polycube construction introduces a formulation of an ℓ_1-based energy that serves as the core component to deform an arbitrary tetrahedral mesh into a polycube-shaped mesh (See Eq. 1). The input mesh is a tetrahedral mesh $\{T, \bar{X}\}$ where $T = \{t_i\}$ is its set of tetrhedra, and $\bar{X} = \{\bar{x}_i\}$ is its set of node coordinates.

$$\underset{X}{\operatorname{argmin}}\, \alpha E_{\ell_1}(X) + E_{\delta}(X) + \beta E_{\eta}(X) + E_e(X) \qquad (1)$$

where E_{ℓ_1} denotes how far the shape is from being a polycube, E_δ measures the amount of distortion, E_η controls the shape complexity of the resulting polycubes, E_e describes the edge-feature penalties. Iteratively solving the energy gradually transforms the input tet-mesh to a polycube. We use the deformed volume mesh as the input to the ℓ_1-polycube construction. The values of parameters α and β for all our models are provided in Table 1.

After deforming the input mesh into its polycube domain, in which each interior vertex can be represented as a parameterization of vertices on the surface of the polycube tet-mesh [JSW05], uniform hex-grids can be extracted from this polycube domain. By mapping these hex grids to the deformed volume, a hex-mesh with elements of uniform size is generated (Figure 2e). As the connectivity of the volume mesh is preserved during the previous magnification process, it is trivial to map this initial hex-mesh back to the original volume space. The resulting hex-mesh consequently has denser hex-grids in the magnified ROIs (Figure 2f).

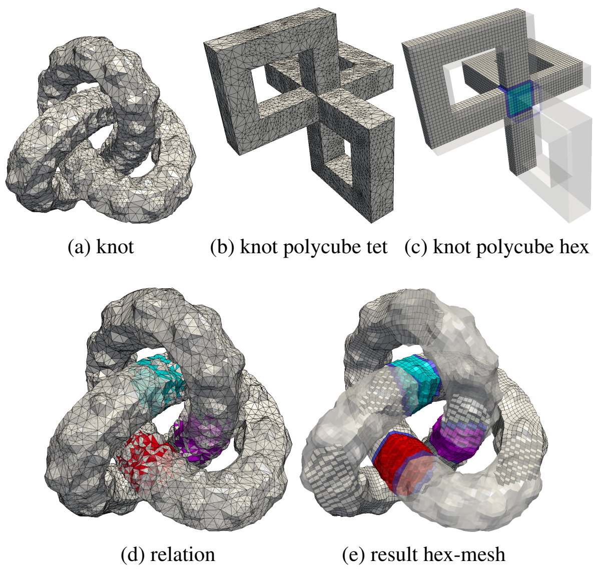


Figure 8: *(a)(b) show the original mesh and its polycubes, (c) shows the hex-mesh generated in the polycube domain that has an overlap region (highlighted by the cyan cube). This overlapping area corresponds to different regions of the origin tetrahedral elements (i.e., cyan, red, and purple regions in (d)). By mapping the hex-elements in the overlapping polycube region back to different regions in the original space, the final hex-mesh is resulted (e).*

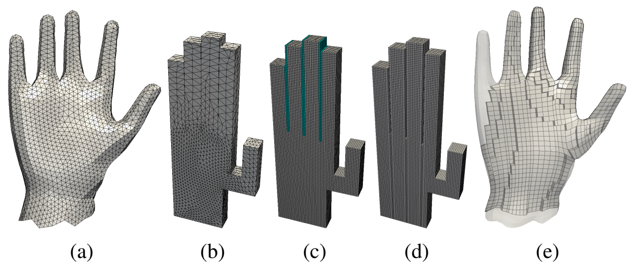


Figure 9: *(a)(b) show the original mesh and its polycubes, (c) shows the hex-mesh generated in the polycube-cube domain. By removing elements in the intersection region (colored in cyan in (c)), we get a clean hex-mesh (d). (e) shows the final hex-mesh in the original space.*

3.4. Untangling self-intersection in polycube

After the preceding deformation, the polycubes of the resultant mesh may have intersections; that is, the two neighboring polycubes that correspond to different regions in the original volume may overlap. See Figure 8c for an example, in which the polycubes of the bumpy knot model overlap in the cyan region. To address this, we classify those regions that are involved in the overlapping into several clusters in the original space, as shown in Figure 8d. This is achieved by first identifying the vertices in the overlapping region in the polycube domain, and then grouping them based on their geodesic distances [CWW13] on the original volume using the K-means clustering [Ste57, Mac67]. In our current experiments, K is selected based on the heuristics of the surface configuration (e.g., 3 for the bumpy knot). Then, different parameterizations are computed in the overlapping polycube, each of which corresponds to a cluster. Each parameterization will then induce a hex-mesh that will be mapped back to its corresponding region in the original space (Figure 8). Figure 9 shows another example of the hand model, in which the resultant polycubes corresponding to different fingers overlap. Using the clustering strategy, we successfully construct a valid hex-mesh from this overlapping polycubes. Nonetheless, we wish to point out that fully addressing the issue of the hex-mesh extraction from volume with overlapping is challenging and beyond the scope of this work, which we plan to investigate further in the future work.

3.5. Hex-mesh Optimization

Similar to other polycube-based hex-mesh generation techniques [GSZ11, LVS*13, HJS*14], we insert a padding layer [She07] to each of the hex-meshes shown in our result section (except the robot hex-mesh). To improve the geometric quality of these hex-meshes, we first improve the regularity of the distribution of their elements using the parametric-domain based optimization [GDC15], and then untangle the flipped hexahedral elements through the edge-cone based optimization [LSVT15], followed by the quality improvement scheme proposed in [BDK*03].

4. Results and Discussion

We have applied our adaptive hex-meshing framework to a number of man-made and natural shape objects. Figure 10 shows a set of resulting hex-meshes with element sizes adaptive to the surface feature scales. Table 1 provides the statistics of the generated hex-meshes and the timing information of our framework. All timing information is obtained in a workstation with Intel(R) Xeon(R) CPU E5-1620 v2 @3.70GHz and 48GB Memory @1866MHz.

Comparison with the octree-based results: Figure 1 compares the hex-mesh generated using our method with the one obtained by an octree-based method [Mes15] for a pipe model. To preserve the small scale features, the base complex of the hex-mesh obtained by the octree-based method has 2656 components, while the base complex of the mesh produced by our method possesses only 86 components. Figure 1d and Figure 1e show the visual comparison of their base complexes [GDC15].

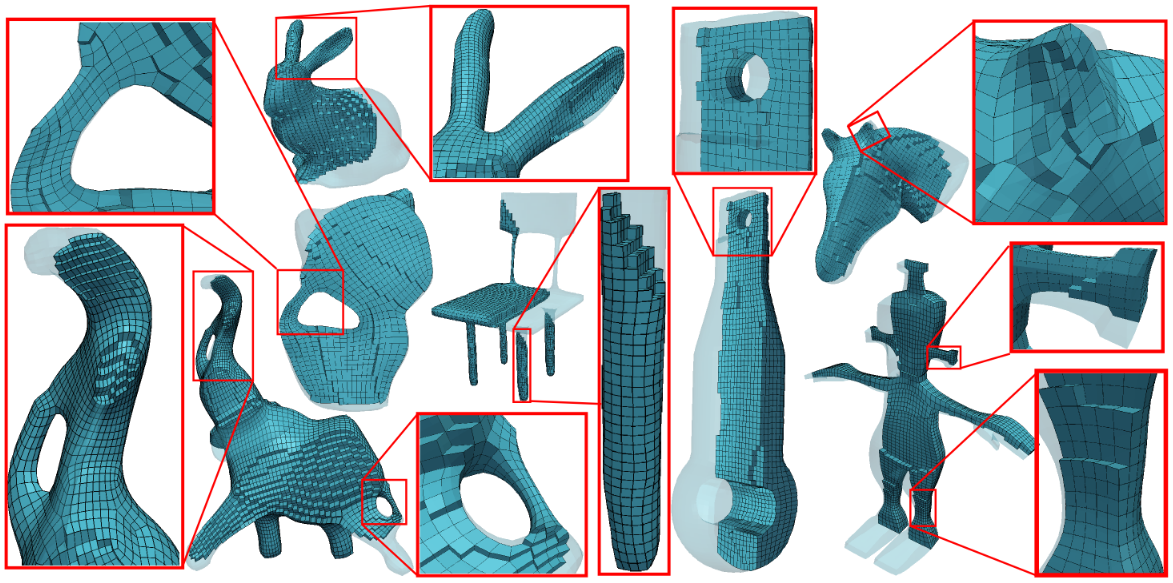
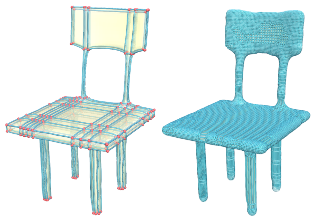


Figure 10: *Some other models in our experiments*



The inset shows another comparison of the base complexes of the hex-meshes generated by our method (left) and the octree-based method (right), respectively, for a chair model. While our mesh has only 266 components, the octree-based mesh consists of 35223 components. More comparisons with the octree-based method can be found in the supplemental document.

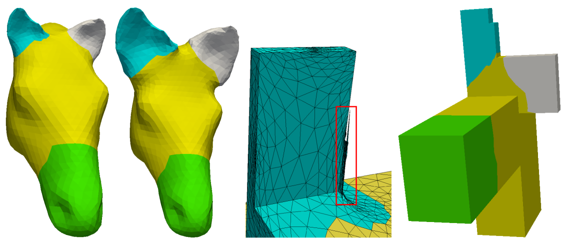
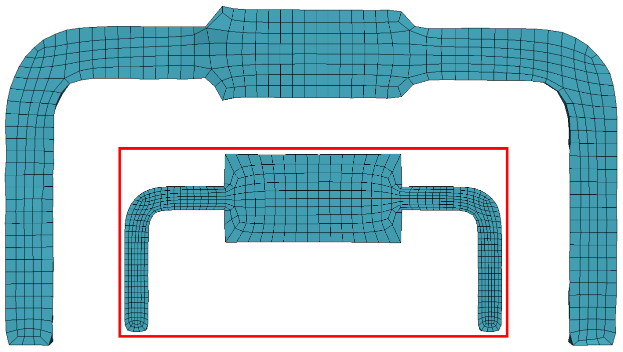


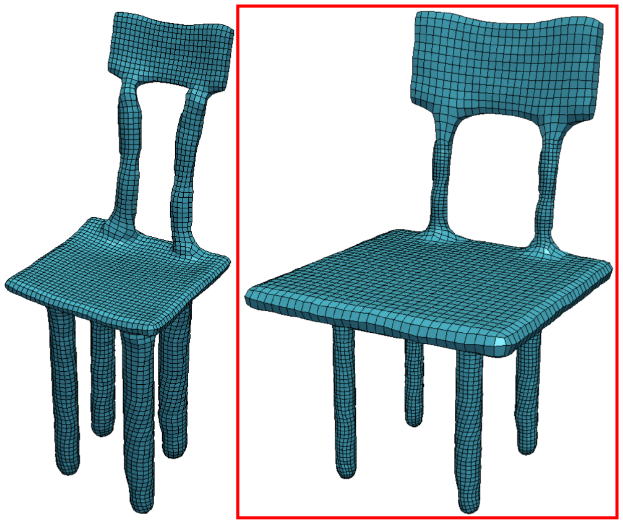
Figure 11: *Magnification may help the construction of ℓ_1-polycube. (a) shows the origin mesh. (b) shows the magnification on the two ears. (c) shows the failure to turn the left ear of (a) into a valid polycube, while the magnification of the two ears (b) helps construct a valid polycube (d).*

Comparison with the ℓ_1-polycubes results: Figure 1b and Figure 1c compare our approach with the original ℓ_1-polycube method [HJS*14]. With a similar number of elements, our approach adaptively fills the volume regions adjacent to the small surface features with smaller (and denser) elements. Also, the complexity of the base complex of the hex-mesh obtained from our approach is comparable to the mesh with the original ℓ_1-polycube (e.g., 48 components in our mesh versus 32 in the one obtained using the original ℓ_1-polycube for the rod model). In fact, the result of ℓ_1-polycube is a tradeoff between the number of singularities (corners) and the smoothness of cubes. For some models with small regions (e.g., ears and tails of those animal models), the ℓ_1-polycube method is likely to smooth out these regions to be planes in the polycube; otherwise, there will be many more singularities in order to preserve these regions. In addition, with the magnification step as a pre-processing before the ℓ_1-polycube construction, some challenging situation for ℓ_1-polycube may be mitigated. Figure 11 provides such an example. The original horse model has thin surface feature at the tips of its ears. Directly using the original mesh to construct the ℓ_1-polycube will lead to an invalid polycube structure (see the highlighted region in Figure 11c). After magnifying the ears of the horse, the ℓ_1-polycube can be constructed correctly (Figure 11b, 11d).

Results with other hex-meshing methods: As described earlier, the initial hex-mesh with uniform element size can be generated with any existing methods from the deformed mesh. Figure 12 provide the results of initial hex-meshes generated with the closed-form polycube and polycut, respectively. Their corresponding final hex-meshes with adaptive element sizes are shown in the highlight rectangle. These results demonstrate the flexibility of our framework.

User interaction on ROIs: As mentioned earlier, in addition for our system to automatically determine a scaling factor for an ROI, we also offer the users the ability to manually deform the ROI or specify a desired scaling factor. Figure 13 shows an example where different scale factors are assigned to the two ears and four legs of the teddy bear model. The resultant hex-mesh is shown in Figure


(a) hex-meshes generated by [LVS*13]


(b) hex-meshes generated by [FXBH16]

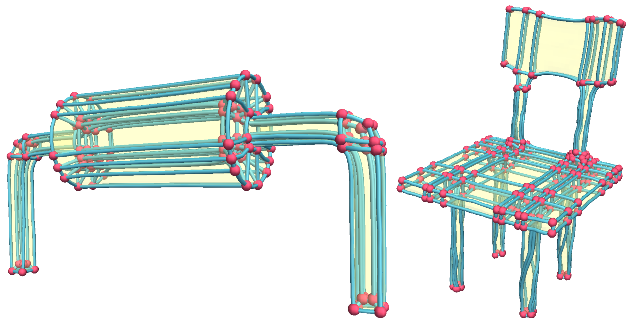

(c) base complex of (a) (d) base complex of (b)

Figure 12: *Results with other hex-meshing methods. We first generate the initial hex-mesh with uniform element size in the magnified space. Final hex-mesh with adaptive element sizes is shown in the highlight rectangle. (c) and (d) are base complexes of (a) and (b), respectively.*

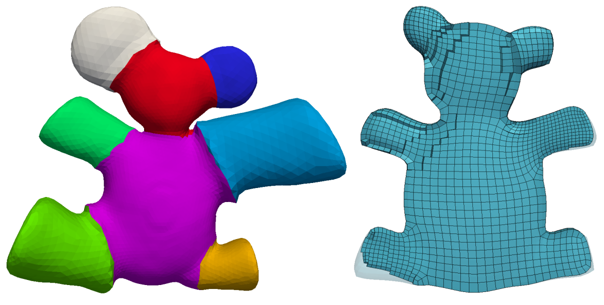

(a) (b)

Figure 13: *(a) the magnification result of the teddy bear with different scale factors (i.e., 1.5 and 2.0 for two ears, 1.2, 1.5, 1.8, 2.2 for legs, respectively). (b) the obtained hex-mesh.*

13b, in which the hex-elements have varying sizes in those ROIs given their different scale factors. Figure 14 shows an example where a user-selected ROI is manually deformed.

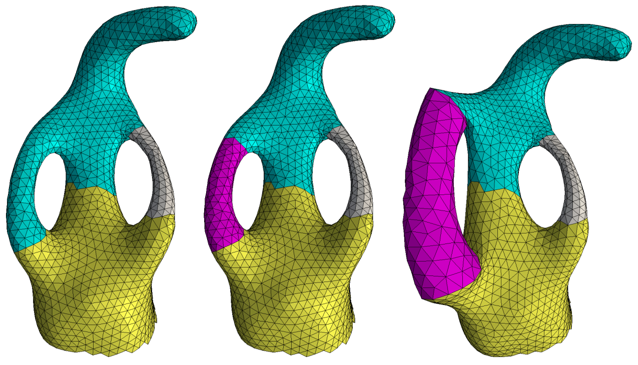

(a) (b) (c)

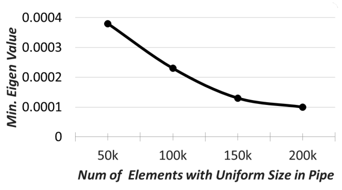
Figure 14: *With user interaction, we can easily modify segment that may not reflect the desired segmentation due to the coarse triangulation in (a) and magnify it individually ((b) $\rightarrow$ (c)).*

Meshing quality in the linear elasticity problem Linear elasticity models an elastically deformable body under infinitesimal displacement or traction. According to Bathe [Bat95], isotropic linear elastic energy is a quadratic energy of the displacement field v, taking the following form:

$$E = \int_{\Omega} \mu\varepsilon : \varepsilon + \frac{\lambda}{2}\mathbf{tr}^2(\varepsilon)dx,$$

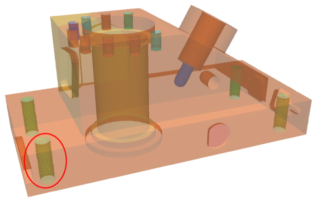
where $\varepsilon = (\nabla v + \nabla v^T)/2 - \mathbf{I}$ is the infinitesimal strain tensor and μ, λ are *Lamé*'s coefficients. We discretize the above continuous energy using the FEM method with trilinear shape function and 10 point (degree 19) Gauss quadrature (10^3 points for each hex element) to approximate the per-hex integral with high accuracy. The induced Euler-Lagrangian equation is an elliptic PDE. Consequently, we have a linear system $Ax = b$ in the discrete case whose left hand side is the stiffness matrix. The eigenvalues of this stiffness matrix indicate the quality of the PDE system. Specifically, the minimal non-trivial eigenvalue (i.e., > 0) indicates the accuracy of the solving of this PDE system [BPM*95, She02]. Note that this minimal non-zero eigenvalue indicates the L_2 error of the discrete approximation of the continuous energy system, which is the most common error measurement used by many PDE problems [She02].

We use two hex-meshes of the pipe model to perform the above eigen analysis, also known as the Modal Analysis [GHL*14]. One is the traditional uniform-size hex-mesh with 9,045 elements, and the other is a hex-mesh generated using our method with 9,044 elements with denser hex-grids near small features. The minimal eigenvalue obtained on the former mesh is 0.0025, while it is 0.0006 for the later. To assess which value is more accurate, a ground truth minimal eigenvalue is needed, which is not known in our case. In the meantime, it has been established that the simulations performed on the meshes with a finer resolution have a higher accuracy than those performed on the coarser meshes [She02]. This is also supported by the plot in the inset, showing that the minimal eigenvalue obtained on the meshes with increasing resolutions for the pipe model decreases accordingly and converges toward 0.0001. This indicates that the minimal eigenvalue obtained from our mesh is closer to the ground truth.



Other magnification approach: The idea of our method is to deform the original mesh before the hex-mesh extraction. Therefore, the sizes of the hex-elements can be implicitly adjusted via the inverse deformation. That said, any other appropriate mesh deformation technique can be combined with our method. Figure 15 shows an example that the input mesh is first deformed based on a surface frame field [PPTSH14], and then the deformed (volume) mesh is used to extract a hex-mesh before mapping it back to the original volume. Nonetheless, this frame field driven surface deformation may result in self-intersecting surfaces, which cannot guarantee to form a valid polycube structure.

Limitations: There are a number of limitations of our current approach. First, our method has difficulties in handling certain mesh regions with inner surface close to outer surface for some CAD models (e.g., the inner structure of the model highlighted in the inset). The resultant hex-meshes in those area may exhibit large distortion. Furthermore, the skeletonization-based segmentation technique may not be able to handle models that do not have tubular shape, such as cars, airplanes and mechanical components. Though, this may be addressed by applying a more general mesh segmentation technique. Second, although a solution to the hex-mesh extraction from the self-intersecting volume caused by the deformation is described, there is no guarantee that it can solve all possible cases. We intend to explore a more generic solution to this issue in a future work. Third, as mentioned earlier, in order to obtain more desired segmentation results for complex models (e.g., with high genus and many small features close to each other), denser triangle meshes will be required, which will significantly increase the computation time. Take the elephant model as an example (Figure 10). We need 10K or more triangles at the nose and teeth of the model in order to successfully separate them. Combing with the other portion of the model, the total number of triangles needed could be more than 25K, resulting in an even denser tetrahedral mesh as the input for the ℓ_1-polycube construction. A more efficient hex-meshing technique, like the recently introduced fast volumetric polycube construction method [FBL16], can be applied to address this issue. Fourth, our current magnification is limited by a relatively small scaling factor due to the simple bounding box guided ARAP deformation. Using a larger scaling factor may not only cause distortion, as already seen in Figure 6c, but also lead to the change of the characteristics of the feature (Figure 17), which may in turn affect the quality of the extracted hex-mesh when mapping back to the original space. Fifth, the deformed mesh may result in a different polycube and singularity structure from the one obtained from the original mesh, as demonstrated by the example shown in Figure 16. To fully understand how our magnification process affects the polycube and singularity structure requires a thorough sensitivity experiment on different configurations of ROIs and the magnification process, which is beyond the scope of this work. Nonetheless, we believe this small discrepancy in the polycube structure is acceptable when compared to the excessive number of singularities introduced by the octree-based method. Though, a future study is needed to better control the additional singularities caused by the deformation. Nonetheless, we wish to point out that many of the above limitations are the inherent issues of the selected techniques for the individual steps of the proposed framework, which should not be considered as the limitations of our general framework. In the future, improved methods can be applied for these steps without modifying our general framework.



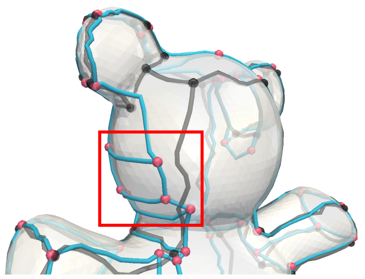


Figure 16: *The singularities (in dark gray) of a uniform-sized hex-mesh and the singularities (in cyan) of a mesh with varying element sizes of the bear. The latter was generated with a scale value of 4.0 on the left ear of the bear. The red rectangle highlights a place where the two singularity structures are different.*

5. Conclusion

In this work, we present a new framework to enable the generation of hex-meshes with elements of varying sizes. This is achieved by

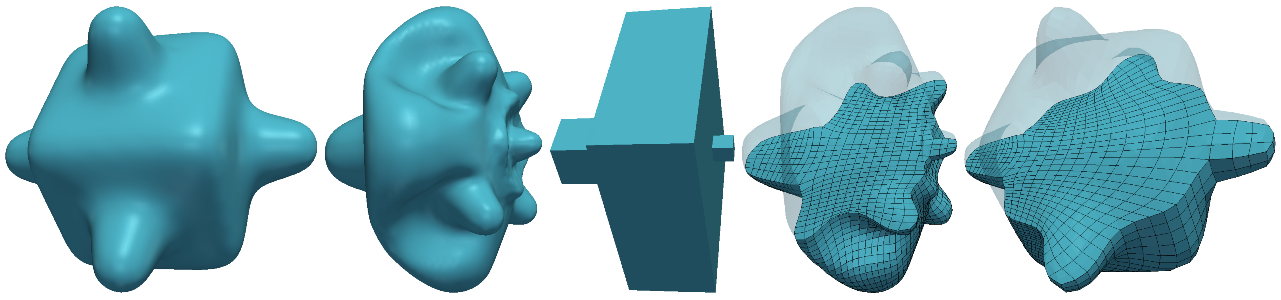


Figure 15: *This example shows that an input model (a) is deformed based on a user-customized frame field (b). The deformed surface is then used to construct the ℓ_1-polycube (c) and to extract a hex-mesh filling the deformed volume (d), which is mapped back to the original space to obtain the output hex-mesh (e).*

Table 1: *Statistics of the resultant hex-meshes. * denote the meshes obtained using the original ℓ_1-polycube [HJS*14]. ^{o} indicate meshes generated by the octree-base method [Mes15]. #Tet and #Tri show the numbers of tetrahedra and triangles in the input meshes, respectively. #Seg is number of the segments of each model. #Scal is the maximum scaling factor for the magnification. #Sin is the number of singularities. #Com is the number of components in the base complex. MSJ/ASJ represents the minimum and average scaled Jacobians, respectively. H Dis denotes the Hausdorff distance from the boundary of the hex-mesh to the input surface (‱ of the diameter of the bounding box of each model). S Time, P time and H time show the timing for segmentation, ℓ_1-polycube construction, and hex-mesh extraction. More complete statistics is provided in the supplemental document.*

Model	#Tet	#Tri	#Seg	#Scal	#Sin	#Com	#Hex	MSJ / ASJ	H Dis	S Time	P Time	H Time
bumpycube	–	39936	–	–	60	121	16937	0.266/0.905	2.65	–	18m27s	16s
bear	98548	14626	8	2.2	188	467	7697	0.443/0.918	3.13	1.3s	209m31s	55s
bear*	10912	4588	–	–	140	104	10700	0.470/0.935	2.43	–	1m53s	10s
bunny	24169	7098	3	2.0	60	77	14571	0.275/0.900	2.92	0.5s	6m27s	19s
chair	14163	7490	8	2.0	160	266	9659	0.274/0.918	1.81	0.6s	2m43s	50s
chair^o	14163	7490	–	–	12005	35223	41664	0.076/0.864	1.08	–	–	1.2s
elephant	119321	24806	9	1.8	526	5827	23002	0.242/0.887	1.51	1.9s	193m10s	110s
horse	41831	7506	4	2.0	78	124	7523	0.259/0.876	6.14	0.5s	14m19s	21s
kitty	4521	1946	2	2.0	80	129	7124	0.291/0.887	2.56	0.2s	36s	6s
pipe	16285	4442	3	2.0	80	86	9045	0.419/0.933	1.59	0.2s	2m37s	12s
pipe^o	16285	4442	–	–	1847	2656	5571	0.092/0.790	3.92	–	–	1.1s
pipe*	16285	4442	–	–	80	69	7168	0.258/0.925	1.45	–	2m32s	14s
robot	27606	9032	10	1.9	196	598	8013	0.254/0.941	1.80	0.6s	6m30s	25s
rod	70178	13818	4	1.5	48	66	11448	0.063/0.932	1.14	1.2s	53m53s	48s

first detecting the regions of interest (ROIs) of the surface that contain small scale features, followed by an automatic magnification process of the ROIs using the as-rigid-as-possible (ARAP) deformation. A hex-mesh with uniform element size is then extracted from the deformed volume mesh using any existing hex-meshing technique, before being mapped back to the original space. Our framework is simple and intuitive to use, and has been applied to a variety of 3D models to demonstrate its effectiveness. The hex-meshes generated with our approach have simpler structure, when compared to those obtained by the octree-based method, while better adapting the element size to the small scale features, when compared to the results with uniform element size.

There are a number of limitations of the current approach that we plan to address as the future work. In addition, the current framework can be further extended to magnified specific local (volume) regions based on other sizing information, such as an error sizing function from the finite element analysis [POB11]. We plan to explore these directions in the future.

Acknowledgements

We would like to thank Xianzhong Fang, Jin Huang, Marco Livesu, and Yang Liu for helping generate hex-meshes with their respective methods for comparison. This work was partially supported by NSF IIS-1553329.

References

[ACOL00] Alexa M., Cohen-Or D., Levin D.: As-rigid-as-possible shape interpolation. In *Siggraph 2000 Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 157–164. 5

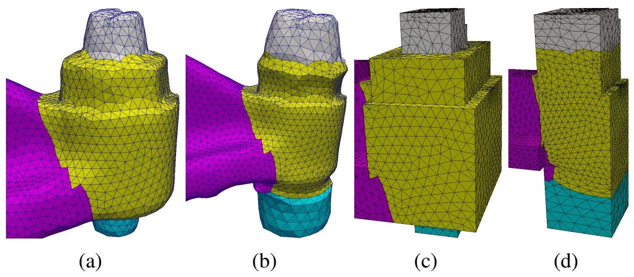


Figure 17: *This example shows that the magnification may alter the surface characteristics. Before magnification (a), the surface exhibits multiple cylindrical structure, while only one cylindrical structure is retained after magnification. This is reflected in their subsequent polycubes shown in (c) and (d).*

[Bat95] BATHE K. J.: *Finite Element Procedures in Engineering Analysis*. Prentice Hall, 1995. 9

[BDK*03] BREWER M., DIACHIN L. F., KNUPP P., LEURENT T., MELANDER D.: The mesquite mesh quality improvement toolkit. In *Proceedings of International Meshing Roundtable* (2003). 7

[BPM*95] BENZLEY S. E., PERRY E., MERKLEY K., CLARK B., SJAARDEMA G.: A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *In Proceedings, 4th International Meshing Roundtable* (1995), pp. 179–191. 10

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (July 2009), 77:1–77:10. 3

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009) 28*, 3 (Aug 2009). 4

[CLS16] CHERCHI G., LIVESU M., SCATENI R.: Polycube simplification for coarse layouts of surfaces and volumes. *Computer Graphics Forum* (2016). 3

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRODER P.: A simple geometric model for elastic deformations. *ACM Transactions on Graphics (TOG) 29*, 3 (July 2010). 5

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph. 32* (2013). 7

[FBL16] FU X., BAI C., LIU Y.: Efficient volumetric polycube-map construction. *Computer Graphics Forum (Pacific Graphics) 35*, 7 (2016). 10

[FXBH16] FANG X., XU W., BAO H., HUANG J.: All-hex meshing using closed-form induced polycube. *Acm Transactions on Graphics (Proceedings of SIGGRAPH 2016) 35*, 4 (2016). 3, 9

[GDC15] GAO X., DENG Z., CHEN G.: Hexahedral mesh re-parameterization from aligned base-complex. *Acm Transactions on Graphics (Proceedings of SIGGRAPH 2015) 35*, 4 (2015). 2, 7

[GHL*14] GAO X., HUANG J., LI S., DENG Z., CHEN G.: An evaluation of the quality of hexahedral meshes via modal analysis. In *1st Workshop on Structured Meshing: Theory, Applications, and Evaluation* (2014). 10

[GMD*16] GAO X., MARTIN T., DENG S., COHEN E., DENG Z., CHEN G.: Structured volume decomposition via generalized sweeping. *IEEE ransactions on Visualization and Computer Graphics 22*, 7 (2016), 1899–1911. 2, 3

[GSZ11] GREGSON J., SHEFFER A., ZHANG E. G.: All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum 30*, 5 (2011), 1407–1416. 3, 7

[HJS*14] HUANG J., JIANG T. F., SHI Z. Y., TONG Y. Y., BAO H. J., DESBRUN M.: l1-based construction of polycube maps from complex shapes. *Acm Transactions on Graphics 33*, 3 (2014). 2, 3, 6, 7, 8, 11

[HT89] HACON D., TOMEI C.: Tetrahedral decompositions of hexahedral meshes. *Eur. J. Comb. 10*, 5 (1989), 435–443. 5

[HT05] HUGHES T.J. COTTRELL J.A. B. Y.: Isogeometric analysis: Cad, finite elements, nurbs, exact geometry, and mesh refinement. *Computer Methods in Applied Mechanics and Engineering 194* (2005), 4135–4195. 1

[HTZ*11] HUANG J., TONG Y., ZHOU K., BAO H., DESBRUN M.: Boundary aligned smooth 3d cross-frame field. *Acm Transactions on Graphics 30*, 6 (2011), 143:1–143:8. 3

[ISS09] ITO Y., SHIH A. M., SONI B. K.: Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *Int. J. Numer. Meth. Engng*, 77 (2009), 1809–1833. 2

[JFH*15] JIANG T., FANG X., HUANG J., BAO H., TONG Y., DESBRUN M.: Frame field generation through metric customization. *Acm Transactions on Graphics (Proceedings of SIGGRAPH 2015) 34*, 4 (2015), 40:1–40:11. 3

[JHW*14] JIANG T., HUANG J., WANG Y., TONG Y., BAO H.: Frame field singularity correction for automatic hexahedralization. *IEEE Trans Vis Comput Graphics 20*, 8 (2014), 1189–1199. 3

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. *Acm Transactions on Graphics (Proceedings of SIGGRAPH 2005) 24*, 3 (July 2005), 561–566. 5, 7

[KBLK14] KREMER M., BOMMES D., LIM I., KOBBELT L.: Advanced automatic hexahedral mesh generation from surface quad meshes. In *Proceedings of the 22nd International Meshing Roundtable* (2014), Springer, pp. 147–164. 3

[LLD12] LIN H., LIAO H., DENG C.: Filling triangular mesh model with all-hex mesh by volume subdivision fitting, 2012. 3

[LLWQ13] LI B., LI X., WANG K., QIN H.: Surface mesh to volumetric spline conversion with generalized poly-cubes. *IEEE TVCG 19*, 9 (2013), 1539–1551. 3

[LLX*12] LI Y. F., LIU Y., XU W. W., WANG W. P., GUO B. N.: All-hex meshing using singularity-restricted field. *Acm Transactions on Graphics (Proceedings of SIGGRAPH 2012) 31*, 6 (2012). 3

[LMPS16] LIVESU M., MUNTONI A., PUPPO E., SCATENI R.: Skeleton-driven adaptive hexahedral meshing of tubular shapes. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 237–246. 3

[LSVT15] LIVESU M., SHEFFER A., VINING N., TARINI M.: Practical hex-mesh optimization via edge-cone rectification. *Transactions on Graphics (Proc. SIGGRAPH 2015) 34*, 4 (2015). 7

[LVS*13] LIVESU M., VINING N., SHEFFER A., GREGSON J., SCATENI R.: Polycut: Monotone graph-cuts for polycube base-complex construction. *Acm Transactions on Graphics 32*, 6 (2013). 3, 7, 9

[LZLW15] LIU L., ZHANG Y., LIU Y., WANG W.: Feature-preserving t-mesh construction using skeleton-based polycubes. *Computer-Aided Design 58* (2015), 162–172. 3

[Mac67] MACQUEEN J. B.: Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* (1967), University of California Press, pp. 281–297. 7

[Mar09] MARÉCHAL L.: Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *proceedings of the 18th International Meshing Roundtable*. Springer, 2009, pp. 65–84. 1, 2

[Mes15] MESHGEMS: Volume meshing: Meshgems-hexa, 2015. 2, 7, 11

[NPPZ12] NIESER M., PALACIOS J., POLTHIER K., ZHANG E.: Hexagonal global parameterization of arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics 18*, 6 (June 2012), 865–878. 3

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: Cubecover - parameterization of 3d volumes. *Computer Graphics Forum 30*, 5 (2011), 1397–1406. 3

[POB11] PAUDEL G., OWEN S. J., BENZLEY S. E.: *Hexahedral Mesh Refinement Using an Error Sizing Function*. Springer, 2011. 11

[PPTSH14] PANOZZO D., PUPPO E., TARINI M., SORKINE-HORNUNG O.: Frame fields: Anisotropic and non-orthogonal cross fields. *Acm Transactions on Graphics (Proceedings of SIGGRAPH 2014) 33*, 4 (2014). 2, 6, 10

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), pp. 109–116. 5

[Sch96] SCHNEIDERS R.: A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with computers 12*, 3-4 (1996), 168–177. 2

[She02] SHEWCHUK J. R.: What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures. In *In Proc. of the 11th International Meshing Roundtable* (2002). 10

[She07] SHEPHERD J.: *Topologic and geometric constraintbased hexahedral mesh generation*. PhD thesis, University of Utah, 2007. 7

[SJ08] SHEPHERD J. F., JOHNSON C. R.: Hexahedral mesh generation constraints. *Eng. with Comput. 24*, 3 (June 2008), 195–213. 2

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24*, 4 (2008), 249–259. 4

[Ste57] STEINHAUS H.: Sur la division des corps matÃľriels en parties. *Bull. Acad. Polon. Sci. 4* (1957), 801–804. 7

[SZ16] SUN L., ZHAO G.: Adaptive hexahedral mesh generation and quality optimization for solid models with thin features using a grid-based method. *Engineering with Computers 32*, 1 (2016), 61–84. 2

[TAOZ12] TAGLIASACCHI A., ALHASHIM I., OLSON M., ZHANG H.: Mean curvature skeletons. *Computer Graphics Forum 31*, 5 (2012), 1735–1744. 3, 4

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. *Acm Transactions on Graphics 23*, 3 (2004), 853–860. 3

[WLL*12] WANG K., LI X., LI B., XU H., QIN H.: Restricted trivariate polycube splines for volumetric data modeling. *IEEE Trans. Vis. Comput. Graphics 18*, 5 (2012), 703–716. 3

[WLT08] WANG Y. S., LEE T. Y., TAI C. L.: Focus plus context visualization with distortion minimization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1731–1738. 5

[YMD11] YANO M., MODISETTE J. M., DARMOFAL D. L.: The importance of mesh adaptation for higher-order discretizations of aerodynamic flows. In *20th AIAA CFD Conference, AIAA-2011* (2011), vol. 3852. 1

[ZB06] ZHANG Y. J., BAJAJ C.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering 195*, 9-12 (2006), 942–960. 1, 2

[ZLX13] ZHANG Y. J., LIANG X., XU G.: A robust 2-refinement algorithm in octree or rhombic dodecahedral tree based all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering 256* (2013), 88–100. 1, 2

[ZZ07] ZHANG H., ZHAO G.: Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method. *Finite Elements in Analysis and Design 43*, 9 (2007), 691–704. 2

[ZZG*12] ZHAO X., ZENG W., GU X. F. D., KAUFMAN A. E., XU W., MUELLER K.: Conformal magnifier: A focus plus context technique with local shape preservation. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 1928–1941. 5