# Multi-Material Adaptive Volume Remesher

Noura Faraj[a], Jean-Marc Thiery[a], Tamy Boubekeur[a]

[a]*LTCI, CNRS, Télécom-ParisTech, Université Paris-Saclay*

**Abstract**

We propose a practical iterative remeshing algorithm for multi-material tetrahedral meshes which is solely based on simple local topological operations, such as edge collapse, flip, split and vertex smoothing. To do so, we exploit an intermediate implicit feature complex which reconstructs piecewise smooth multi-material boundaries made of surface patches, feature edges and corner vertices. Futhermore, we design specific feature-aware local remeshing rules which, combined with a moving least square projection, result in high quality isotropic meshes representing the input mesh at a user defined resolution while preserving important features. Our algorithm uses only topology-aware local operations, which allows to process difficult input meshes such as self-intersecting ones. We evaluate our approach on a collection of examples and experimentally show that it is fast and scales well.

*Keywords:* Multi-material tetrahedral mesh ; volume remeshing ; feature preservation

## 1. Introduction

Multi-material volumetric datasets are widely used to study physical phenomena, model physically-plausible shapes or fabricate/print real objects from digital ones. For instance, a broad range of medical simulations steam from the increasing number of available 3D anatomical images. Those datasets are typically composed of voxel grids acquired using Magnetic Resonance Imaging (MRI) or scanners and accurately labeled by professionals - i.e., each voxel is assigned with a label representing a single material (e.g., organ). The union of these components forms the simulation *domain*, where each material is represented by a single *subdomain*. However, in practice, simulations are often designed to run on a mesh of the input domain, using Finite Elements Methods (FEM). Employing large polyhedra - typically tetrahedra - for constant material regions reduces drastically the computation costs while preserving a good approximations. Indeed, the result of a simulation depends on the domain representation accuracy and the quality on the input mesh since its stability usually depends on the size and shape of its tetrahedra [1]. For instance, tetrahedra with small dihedral angles cause negative volumes under small perturbations, while large angles strongly increase the simulation errors.

As the material is supposed constant within a subdomain, the critical features to preserve during the meshing process reside at the interfaces between labels since they indicate the shape boundaries and the junctions between subdomains. These features can take three forms [2]: (i) the surfaces patches between two labels (*2-junctions*), (ii) the edges between three or more labels (*1-junctions*) and (iii) the corner vertices between four or more labels (*0-junctions*). Generating tetrahedral meshes at the suitable resolution for simulation while accurately capturing such features is a tedious task. Ideally, one could generate high-quality meshes at several resolutions, exploiting each time the previously finer meshes to generate the coarser and trading feature preservation for regularization. Indeed, a regularization process is unavoidable, as smooth boundaries are mandatory for visualization and stability purpose. As the seminal discretization (e.g., from the input image) can fail at meeting these constraints, a remeshing step (i. e. optimization and/or simplification) is often necessary.

*Contributions.* We propose a simple and practical iterative remeshing algorithm for 3D triangulations, which provides high-quality meshes at a chosen resolution while preserving features such as multi-material boundaries. The user can efficiently reach the desired resolution by adjusting the target edge length: in particular, the mesh is processed iteratively until the size constraint is met without starting over from the input mesh, which is essential during fine-tuning remeshing sessions. Our algorithm allows generating uniform as well as adaptive meshes, for which the spatially-varying resolution is driven by a sizing (scalar) field that can be user-defined, e. g. to generate a dense mesh in regions of interest, or based on a distance field. In particular, such a field may be generated from the subdomain boundaries, yelding elements with increasing size when located away from the boundaries, therefore resulting in an isotropic adaptive mesh. This is especially effective to minimize the number of tetrahedra while preserving accurate boundaries.

In order to provide our algorithm with a structured decomposition of the multi-material domain, we use a *feature complex*, similar to the one proposed by Dey et al. [3], but equipped with a *Moving Least Square* (MLS) geometric definition derived from *Hermite Point Set Surfaces* [4]. Doing so, we decorellate the structured geometry of the domain from the mesh and can perform feature-dependent topological operations coupled with a hierarchical MLS smoothing. We can preserve additional features which are either provided by the user, as a set of polylines, or detected on the domain boundary (e.g., surface sharp features). On the contrary to Delaunay-based methods, ours al-

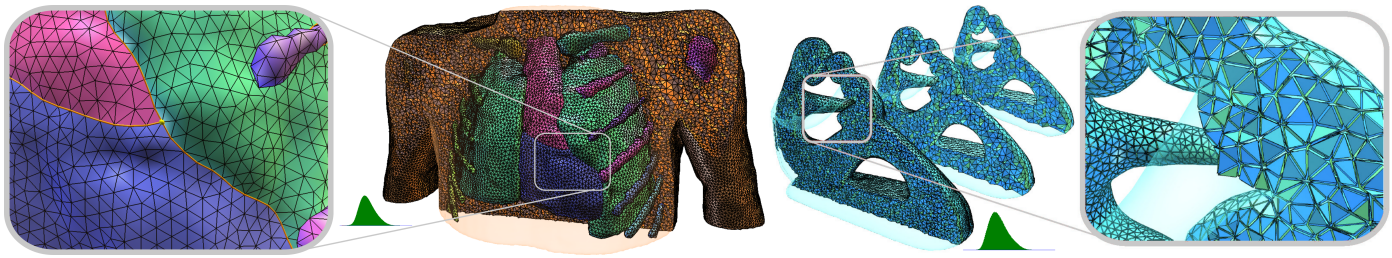

Figure 1: Our method remeshes complex multi-material tetrahedral meshes, with high geometric quality and exact topology preservation.

lows to efficiently generate meshes at any resolution by only changing the target edge length since the MLS representation of the feature complex enables us to mesh the domain geometry directly, even at low meshing resolution.

By using only topology-aware local operations, our ~~technique~~ can process difficult input meshes such as self-intersecting ones, whereas Delaunay-based techniques will necessarily glue intersecting parts. Last, our remeshing method can be used as a complement to any existing meshing method in order to generate a mesh suited to the users needs. Therefore any structured mesh can be processed with our algorithm. For instance, we apply our method on trivial high-resolution tetrahedral mesh generated from segmented voxel grids.

## 2. Background

The generation, simplification and refinement of high-quality tetrahedral meshes are very active research fields. Here, we give a non exhaustive overview of the existing methods and focus on the ones that are able to handle multi-material inputs.

*Meshing.* We can group the meshing methods in three main categories: Delaunay-based, lattice-based and variational.

Delaunay-based methods start by distributing a set of points over the input domain. Then a Delaunay refinement process [5, 6, 7] is used to add *Steiner* points to the triangulation until the input approximation, elements shape and quality criteria are met. This process has first been proposed for domains bounded by a smooth surface [8, 9] and further extended for piecewise smooth boundaries [10]. In order to handle the input domain sharp features, the authors propose to build a *Piecewise Smooth Complex* (PSC), which is composed of surface patches, curves (intersections of surface patches) and points (intersections of curves). Protecting balls of varying size are defined around the edges to preserve the features of this complex during the refinement process [11, 12]. Building upon such a complex, Tournois et al. [13] propose a method coupling refinement and optimization strategies to guide the insertion of Steiner points and directly obtain high-quality meshes. These principles are extended to handle multi-material domains [14] and preserve their 1- and 0-junctions using similar protecting balls [2]. More recently, Dey et al. [3] propose a PSC composed of multi-material junctions, and we use a similar complex to identify the features to preserve. While those methods allow generating high-quality meshes, the implementation of the protecting ball paradigm remains highly non trivial in a multi-material setting.

Lattice-based meshing approaches are inspired by the Marching Cubes algorithm [15] applied to multi-material boundaries [16]. An initial high resolution regular mesh is generated from the volume data and the elements are split according to precomputed boundary configurations until the input domain is well represented. Those methods tend to produce dense meshes and ill-shaped tetrahedra near the boundaries. Furthermore, their resolution depends on the input grid one. The *isosurface stuffing* method [17] uses a similar paradigm to represent single material domains but guarantees theoretical bounds on the tetrahedra' dihedral angles. Inspired by this strategy, Bronson et al. [18] offer similar guarantees for labeled volume data.

Variational approaches [19, 20, 21, 22] insert particles or an initial mesh in the input domain and use a non-linear energy optimization to tailor the feature-aware point distribution. These methods provide high-quality results but are strongly dependent of the initial setting and are computationally expensive.

*Poorly-shaped elements.* Fig. 2 presents a common classification of tetrahedral degenerencies. In 2D, the quality of the triangle shape is defined as the ratio of the shortest and longest edge because it relates to the minimum angle, but this property is no longer true in 3D. A poorly-shaped tetrahedron can have edges with similar length, e. g. for *slivers*, which are almost flat tetrahedra. A tetrahedron's quality is thus better described by its minimal dihedral angle and its radius-ratio (the ratio between the inscribed and circumscribed spheres' radii). Note that a regular tetrahedron has dihedral angles equal to 70.5 degrees.

*Remeshing and quality improvement.* Since feature preservation and quality constraints are tedious to combine, ill-shaped tetrahedra are likely to be generated during the meshing process. For instance, Delaunay-based refinement processes do not prevent the apparition of slivers - tetrahedra meeting the Delaunay constraints but with poor quality - and induce an unavoidable post-processing step. A first solution, called sliver *exudation* [23], turns the triangulation into a weighted Delaunay triangulation to address this problem. An alternative approach is to

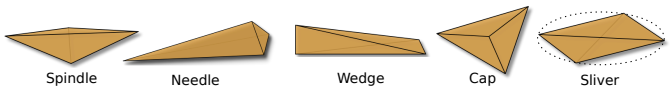


Figure 2: **Ill-shaped tetrahedra:** *needles* and *wedges* (large longest to shortest edge ratio), *caps* (small radius-ratio and three large dihedral angle) and *slivers* (almost flat but with edges of similar length).

*perturb* slivers through vertex relocation and Delaunay connectivity update [24]. In a more general setting, the quality can be improved using optimization based-smoothing and local topological operations such as edge flip or removal [25]. This approach was further improved by Klingner et al. [26] through additional operations such as vertex insertion, multi-face removal and a roll back mechanism.

To cover a large range of resolutions and define more densely meshed regions of interest, both simplification and refinement schemes - generating high-quality meshes - are necessary when remeshing. For visualization purposes, the simplification of tetrahedral meshes is widely used through edge contraction [27, 28], similar to surface simplification techniques [29, 30] or point sampling [31, 32]. Mesh adaptation is widely used to increase simulation accuracy by improving the mesh quality to better capture the studied physical phenomena. Local operations are used in order to modify the mesh and satisfy a given mesh metric field [33, 34, 35]. Unfortunately, only a limited number of methods are generalized to handle multi-material meshes and 1- and 0-junctions. Cutler et al. [36] propose a method to generate high-quality segmented meshes at different resolutions by performing local topological operations to meet the quality and edge length criteria. While this method allows preserving boundary surfaces using a volume based error metric, 1- and 0-junctions are not taken into account. To the best of our knowledge, only ~~the following ones~~ account for 1- and 0-junctions [37, 38]. The authors propose *link conditions* defining if an edge can be collapsed without changing the topology of the features of different degrees.

Similar issues are tackled for the remeshing of triangular *surface* meshes. In particular, following [39, 40], Botsch and Kobbelt [41] propose an iterative remeshing method to generate isotropic high-quality triangular meshes using simple local operations performed in a predefined order.

As discussed in this section, no existing method allows generating high-quality meshes at various resolutions efficiently while preserving multi-material features. Most existing meshing processes allow generating a mesh within several minutes or hours, and in some cases to refine this mesh (Delaunay triangulation), but not to simplify it. Simplification methods can preserve these features but do not meet the quality requirements.

Beyond the contributions listed earlier, our volume remeshing method builds upon several previous ideas. First, the core of our approach is inspired from the *surface* remeshing method of Botsch and Kobbelt [41], using local connectivity modifications only [39, 40] which have proved to be highly efficient in the surface case. Second, we avoid self-intersections robustly by adding imaginary tetrahedra to the triangulation [42]. Third, to unify the local remeshing rules sustaining our algorithm, we perform all operations on an extended complex by linking the outer facets of the triangulation to a dummy vertex.

## 3. Our algorithm

### 3.1. Overview

Given a target edge length $l$, our remeshing method for multi-domain tetrahedral meshes can be summarized as follows:

**Preprocess** detect the boundaries and features to preserve, add imaginary tetrahedra to prevent self-intersections.

1 **split** any edge longer than $e_{\max}$,

2 **collapse** any edge shorter than $e_{\min}$,

3 **flip** edges to minimize the average valence and to optimize locally the dihedral angle distribution,

4 **filter** to relocate vertices, taking features into account,

5 go to **1** unless the target resolution is reached,

**Postprocess** remove slivers and improve mesh quality.

For a target edge length, we use constant values $e_{\max} = 4l/3$ and $e_{\min} = 4l/5$. These thresholds avoid *looping*, and splitting an edge verifying $|e_{max} - l| > |\frac{1}{2}e_{max} - l|$ and collapsing an edge verifying $|e_{min} - l| > |\frac{3}{2}e_{min} - l|$ reduces the deviation from the target length, see [43] for more details. Optionally, we can tailor $l$ in a spatially-varying fashion w.r.t. a sizing field capturing either the distance to the boundary or user-defined regions of interest. In the particular case of an input mesh having already the aimed resolution and processed solely for regularization and quality improvement purposes, the target length is set to a value which is slightly lower than the current average edge length, to introduce perturbations in the optimization [41].

### 3.2. Representation

We note the set of labels associated with either an input multi-material 3D triangulation as $\mathcal{L} = \{l_n\}_{n \in I_\mathcal{L}} \subset \mathbb{Z}$. By convention, null values represent the background, i.e., the parts of the data that do not belong to the represented domain.

#### 3.2.1. Labeled tetrahedral mesh

The mesh is noted $M = \{V, E, T\}$ with $V = \{v_i\}_{i \in I_V} \subset \mathbb{R}^3$ its vertices, $E = \{e_{ij}\}$ its edges connecting adjacent vertices $v_i$ and $v_j$ and $T = \{t_k\}_{k \in I_T}$, its tetrahedra indexed over $V$. We call the triangular faces of the tetrahedra *facets*. We note $T_1(v_i)$ (resp. $F_1(v_i)$) the set of tetrahedra (resp. facets) incident to a vertex $v_i$ and $T_1(e_{ij})$ (resp. $F_1(e_{ij})$) the set of tetrahedra (resp. facets) around an edge $e_{ij}$.

The input mesh is typically composed of $n$ subdomains, with $L(t_k) = l_i$ denoting the label associated with a given tetrahedron $t_k$. We add a special imaginary subdomain to ensure that the remeshing process will not introduce self-intersections of the represented domain [42]. The additional *imaginary* tetrahedra have a null label $L(t_k) = 0$ (i.e., background) and will be processed like any other subdomain. As illustrated in Fig. 3,

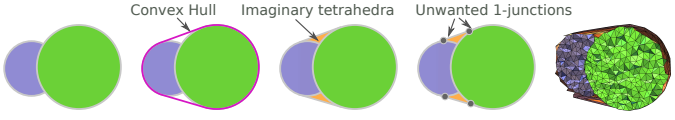


Figure 3: Directly filling the space delimited by the convex hull and the mesh boundaries with imaginary tetrahedra can create unwanted features.

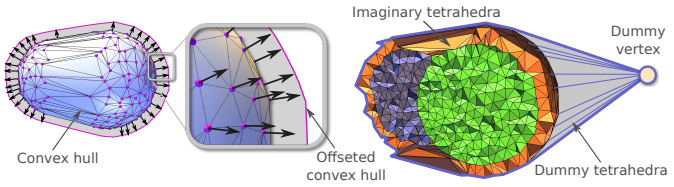


Figure 4: **Pre-processing.** Addition of a layer of imaginary tetrahedra and extension of the complex by connecting the outer facets to the dummy vertex.

naively filling the convex hull of the input vertices generates unwanted features. To tackle this issue, the input mesh is embedded into an inflated convex hull to ensure that the input domain is surrounded by at least one layer of tetrahedra (see Fig. 4). To do so, we duplicate the elements of $V$ laying on the convex hull and displace them in their outer normal direction before triangulating them, using a Restricted Delaunay Triangulation preserving the original mesh outer boundary. The displacement factor is typically set to 4% of the domain's bounding box. To process the outer boundary like any other, i. e. unifying the representation of inter-domain boundaries and 3D surfaces, we connect the outer boundary facets (i.e., of the offseted convex hull) to a dummy vertex creating *dummy* tetrahedra marked with a negative label.

### 3.3. Elements notations

Here, we define the simplices notations used in the remainder of the document (see Fig. 5). Facets shared by two tetrahedra that belong to different subdomains are *boundary facets*. The vertices (resp. edges) of those facets are *boundary vertices* (resp. *boundary edges*). For each vertex $v_i$, we note $S(v_i) \subset L$ the set of labels incident to $v_i$:

$$S(v_i) = \{L(t_k)_{t_k \in T_1(v_i)}\}.$$

Similarly, we define $S(e_{ij})$ for an edge. *Volume* (resp. *boundary*) vertices verify $|S(v_i)| = 1$ (resp. $|S(v_i)| > 1$), except for the dummy vertex. As shown in Fig. 5, we identify four types of edges:

- *volume edges* connect two volume vertices (green, red),
- *mixed edges* connect a volume and a boundary vertex (gray),
- *boundary* edges connect two boundary vertices and have incident boundary facets (pink and blue),
- all other edges are *critical* edges (yellow).

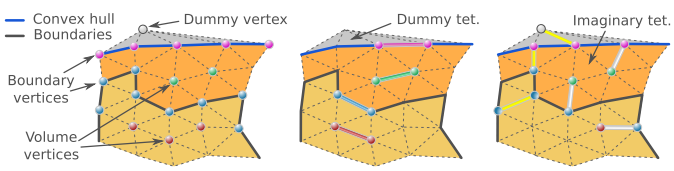


Figure 5: **Classification.** (Left) Vertex types, (Center) Boundary and volume edges and (Right) Mixed and critical edges.

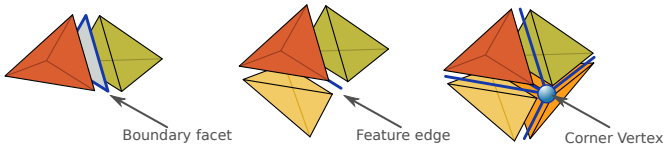


Figure 6: **Feature elements** of various dimension detected using solely incident subdomain indices.

### 3.4. Feature detection

In order to conform to the input boundaries and perform feature-aware operations, we identify feature elements of different dimensions that together form a *feature complex* similar to a point-sampled cell complex [44] or a PSC [3].

*Feature elements.* The *boundary facets*, which are facets between tetrahedra of different labels, are the complex elements of dimension 2. The *feature edges*, which are boundary edges at the intersection of three or more labels ($|S(e_{ij})| > 2$) are the elements of dimension 1. Vertices with three or more incident labels ($|S(v_i)| > 3$) and at least three feature edges in their one-ring are *corner vertices* (see Fig. 6) and are 0-dimensional elements.

The input domain's n-junctions - with $n$ the dimension in the feature complex - are represented in the mesh (see Fig. 7) as follows:

- *2-junctions* are sets of connected boundary facets located at the interface between two subdomains. Each 2-junction forms a triangle surface patch, where its orientation is deduced from one of its incident subdomains and delimited by 1-junctions (if present in the data).
- *1-junctions* are sets of connected feature edges sharing the same set of incident subdomains. Each 1-junction forms a polyline at the intersection of 2-junctions with different subdomains pairs.
- *0-junctions* are corner vertices and are located at the intersection of 1-junctions.

We refer to boundary vertices that lie on 2-junctions but which do not belong to a 1- or 0-junction as *surface vertices*. The vertices that lie on 1-junctions and that are not corner vertices are referred to as *feature vertices*. Additionally, boundary edges linking two surface vertices are called *surface edges*. We do not need to build the feature complex since the elements of the feature complex are already present in the input mesh.

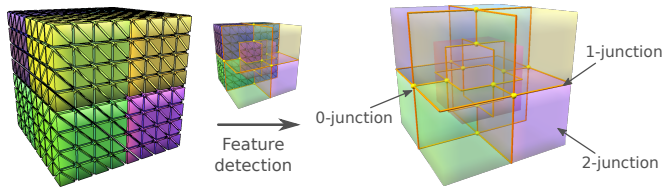


Figure 7: **Feature detection.** The surface patches represent 2-junctions, the polylines 1-junctions and the spheres 0-junctions.

### 3.5. Smooth Modeling

We model piecewise smooth boundaries by fitting 2-junctions with MLS surfaces. These implicit, meshless surfaces are defined through a local operator allowing to project a 3D point on a local surface reconstructed from unorganized point samples. We use a classical Point Set Surface (PSS) definition, proposed by Alexa et al. [45], to represent aliased boundaries and an Hermite Point Set Surfaces (HPSS) definition, proposed by Alexa and Adamson which avoids shrinking of the input model [4], otherwise. Any other definition can be used without any change to our methodology. For instance, in the case of input meshes with limited noise and relevant features, one might prefer a representation handling sharp features [46]. We refer to [47] for an excellent survey on MLS surfaces. We define the MLS surface associated with each 2-junction using a dense, area-based, consistently-oriented point sampling of it.

The MLS representation of the 2-junctions allows us to filter noise and acts as a regularizer at each step of our remeshing algorithm. Storing it also allows us to recover fine geometric details, when navigating from a low-resolution to a high-resolution version of the mesh during the remeshing session.

## 4. Operations

Special care needs to be taken when processing boundaries since the used local operations, described in this section, do not result in the preservation of the feature's topology. Therefore, following [36] and [37, 38], we define feature-aware rules and conditions depending on the feature complex hierarchy. Indeed, our rules assign a priority when processing the mesh elements based on their type which defines a so-called hierarchy.

### 4.1. Classification

The main idea is that only interior vertices should be allowed to be relocated freely in the volume, while n-junction vertices should be moved along their n-junction, in order to preserve the latter (the same idea was used in 2D for the tangential Laplacian [43], resulting in the preservation of the surface features). To fit these constraints, we define three different conditions between the pairs of connected vertices leading to different rules: *similarity*, *inclusion* and *exclusion* (see table 1).

- The *similarity* condition is met for $v_i$ and $v_j$, that are not corner vertices, if $|S(v_i)| = |S(v_j)|$ and $|S(e_{ij})| = |S(v_i)|$ - i.e., for volume vertices, surface vertices linked by a surface edge, feature vertices linked by a surface edge and feature vertices linked by a feature edge.

- The *inclusion* condition if $|S(v_i)| > |S(v_j)|$ and $|S(e_{ij})| = |S(v_j)|$ and $v_j$ is not a corner vertex - i.e., for mixed edges, edges linking a surface vertex and a feature or corner vertex and feature edges linking a feature and a corner vertex.

- Otherwise, the *exclusion* condition is met - i.e., critical edges, edges between feature or corner vertices not belonging to the same 1-junction or linking two 0-junctions.

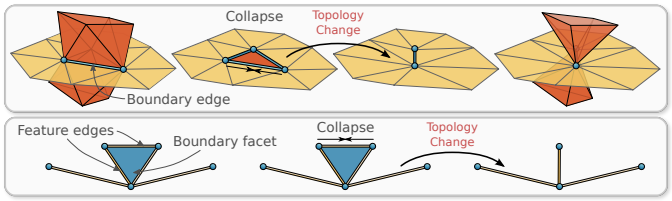


Figure 8: **Topology exceptions.** Additional tests to preserve the topology of the 1- and 2-junctions in small tubular regions.

The pair of vertices meeting the similarity condition affect each other. For the ones meeting the inclusion condition, only the vertex with the highest dimensionality in the feature complex, or the one not belonging to it, will be affected. And finally, for the exclusion condition, none of the vertices will be affected since it would create undesired merging of vertices that belong to different 2- or 1-junctions and fail to preserve small local features of the subdomains. Table 1 summarizes the conditions and the derived rules. In practice, except for corner vertices, a vertex $v_i$ will only be affected by the vertices $v_j$ of its one-ring if $|S(v_i)| \geq |S(v_j)|$ and $|S(e_{ij})| = |S(v_j)|$.

Unfortunately, these conditions do not prevent a change of topology in the same tubular regions as illustrated in Fig. 8. The three edges around the red facet meet the similarity condition but a collapse operation would change the topology of the subdomains, creating a pinched boundary. Similarly, the collapse of any of the three feature edges around the blue facet would change the topology of the 1-junction. Therefore, we perform two additional *topology tests* for boundary and feature edges to detect these configurations and prevent the collapse in these cases. A boundary edge is not collapsed if one of its incident facets is not a boundary one but is composed of three boundary edges. Similarly, a feature edge is not collapsed if one of its incident facets is composed of three feature edges. Now that we have defined the feature-aware rules, we describe our remeshing process in the following.

### 4.2. Split edges

In the first step of our iterative procedure, all the edges with a length superior to $e_{\max}$ are split, i.e., a vertex $v_k$ is added at their mid-point, dividing every tetrahedron around the edge into two. Note that the two new tetrahedra are assigned with the same label as the one they are subdividing. The set of labels of the added vertex $v_k$ is the set of labels of the tetrahedra around the current edge, i.e., $S(v_k) = S(e_{ij})$. As only insertions are performed at this step, no feature preservation rule is required.

### 4.3. Collapse edges

Now that the long edges have been split, we remove short edges in order to get a uniform edge length close to the target one. To do so, we collect all the edges to collapse with a length smaller than $e_{\min}$, and proceed to perform the possible collapses. At this point, our thorough classification becomes instrumental, as not all edges can be collapsed without modifying the topology of the subdomains, inverting tetrahedra or even resulting in an invalid data structure [29, 48].

| Condition | Type of $v_i$ | Type of $v_j$ | Type of $e_{ij}$ | $\lvert S(v_i)\rvert$ | $\lvert S(v_j)\rvert$ | $\lvert S(e_{ij})\rvert$ | # $f_i$ | # $f_j$ | Update $v_i$ | Update $v_j$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Similarity | volume | volume | volume | 1 | 1 | 1 | n/a | n/a | yes | yes |
| Similarity | surface | surface | surface | 2 | 2 | 2 | - | - | yes | yes |
| Similarity | feature | feature | feature | $n_i > 2$ | $= n_i$ | $= n_i$ | $< 3$ | $< 3$ | yes | yes |
| Inclusion | volume | boundary | mixed | 1 | $> 1$ | 1 | - | - | yes | no |
| Inclusion | surface | feature or corner | surface | 2 | $> 2$ | 2 | - | - | yes | no |
| Inclusion | feature | corner | feature | $n_i > 2$ | $n_j > n_i$ | $n_i$ | $< 3$ | - | yes | no |
| Exclusion | boundary | boundary | critical | $n_i$ | $n_j$ | $\neq min(n_i, n_j)$ | - | - | no | no |
| Exclusion | feature | feature or corner | surface | - | - | - | $> 2$ | - | no | no |
| Exclusion | corner | corner | - | - | - | - | - | $> 2$ | no | no |

Table 1: **Conditions and rules ensuring the preservation of the subdomains' topology.** All operations depend on the type of the edge and its vertices. We can easily detect if the vertices will be affected during the current operation depending on their number of incident subdomains and of incident feature edges noted $\#f_i$, i.e., on their dimension in the feature complex.

*Feature preserving collapse.* We start by evaluating which condition the current edge meets in order to define which type of collapse to perform, according to table 1:

- *similarity*: mid-point collapse (both vertices are affected),
- *inclusion*: toward the vertex with the highest number of subdomains, i.e., with the lowest dimension in the feature complex,
- *exclusion*: no collapse, since a contraction would change the topology of the subdomains.

The tetrahedra incident to the affected vertex, or vertices, are set to be updated, except the ones incident to the current edge, which are set to be removed. In order to ensure the validity of the operation, we perform the following tests:

1. all the updated tetrahedra have a positive volume,
2. all the new edges are shorter than $e_{max}$,
3. no incident non boundary facet has three boundary edges,
4. no incident boundary facet has three feature edges.

The third (resp. fourth) topology test is performed for boundary (resp. feature) edges. The operation is performed if these four constraints are met. Some of the new edges, incident to the remaining vertex, may be shorter than $e_{\text{min}}$. Therefore, we evaluate their length and set the ones that do not respect the length criteria to be collapsed. This process is repeated until all edges are either smaller than $e_{\text{min}}$ or impossible to collapse.

Following Botsch and Kobbelt [41], once the overall edge length is close enough to the target one, the local connectivity is changed in order to minimize the average valence and optimize locally the dihedral angle distribution, using an edge flip operation described in the following.

### 4.4. Flip edges

Flipping an edge in a tetrahedral mesh induces far more changes than for a triangular mesh. Indeed, the operation changes the number of tetrahedra adjacent to the edge, except when there are exactly two or four adjacent tetrahedra. It removes an edge and replaces it with facets. The flip operation, also called *edge removal*, has been first proposed in [49] and further studied in [25, 26], as a mesh improvement strategy. For each edge $e_{ij}$, we explore the space of possible flip operations and perform the one that offers the best quality for $T_1(e_{ij})$. Specifically, it should maximize the worst dihedral angle for volume edges and average the boundary vertices valences, i. e. minimize the average boundary vertices valence's deviation from 6 for surface vertices and 4 for feature vertices. In the mean time, the operation generating inverted tetrahedra or edges that already exist are discarded. Recalling that boundary edges have exactly two incident boundary facets, we preserve the 2-junctions by flipping the edges towards one of the two boundary vertices of these facets, enforcing a boundary edge in the new configuration. Last, feature edges are not flipped since processing them would fail to preserve 1-junctions.

### 4.5. Filtering

Once the connectivity has been updated to locally optimize the dihedral angle distribution, we relocate the vertices to improve their distribution (see Fig. 9). Each vertex is relocated by averaging the subset of its one-ring vertices that meet the similarity or inclusion condition (according to table 1) and, for boundary vertices, that verify the topology tests. To prevent the inversion of tetrahedra, first we smooth the feature vertices, then the surface ones and finally the volume vertices.

*Feature vertices.* The feature vertices lie at the intersection of three or more 2-junctions. For each adjacent 2-junction, we perform a tangential Laplacian smoothing using the positions

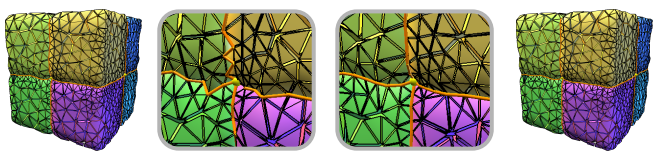

Figure 9: **Feature preserving smoothing.** Using the feature complex, a feature preserving hierarchical smoothing is performed.

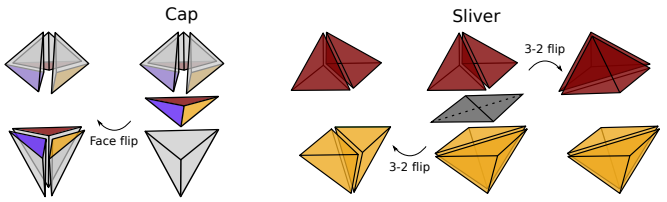


Figure 10: **Sliver removal.** Optimization step to remove the remaining few poorly-shaped tetrahedra: (Left) face flip, for cap tetrahedra, or (Right) edge removal for other kinds of degeneracy.

of its neighbors on the related 1-junction and the vertex normal corresponding to the current 2-junction, then project the result on its MLS surface. The smoothed position is then obtained by averaging the projected points.

*Surface vertices.* We perform a tangential Laplacian smoothing by considering (i) the surface, updated feature and corner vertices of its one ring, that verify the *similarity* or *inclusion* condition and the topology tests, and (ii) the corresponding vertex normals, which are computed using its adjacent boundary facets. The smoothed position is then projected onto the MLS surface modeling the related 2-junction.

*Volume vertices.* Finally, the vertices that do not belong to the feature complex are smoothed by performing a Laplacian smoothing using all its adjacent vertices, since they all verify the *similarity* and *inclusion* conditions.

### 4.6. Quality improvement

After a few iterations (typically 5 to 10), the target resolution is usually reached and the overall mesh quality is improved. However, we observed that performing a few cycles of smooth and flip operations drastically improves the final quality.

A final optimization process might be necessary to remove slivers or poorly-shaped tetrahedra. For each tetrahedron not meeting the quality criteria, we identify its type of degeneracy (see Fig. 2) and we perform the operation that improves the local quality while respecting the previously prescribed rules.

First, *needles* or *wedges*, which have a large longest to shortest edge ratio in addition to a small radius-ratio, are removed by collapsing their first collapsable shorter edge improving the local quality. Note that, since our method equalizes the edge length, these types of elements are unlikely to appear. Then the *caps*, which have a small radius-ratio and three large dihedral angles, are removed by flipping the face opposite to the three largest dihedral angles. Finally, for *slivers*, which have two large and two small dihedral angles, we perform the flip of one of the two edges, shared by the pairs of facets forming the larger dihedral angles, that best improves the local quality (see Fig. 10). Usually, a small number of tetrahedra do not meet the quality criteria, and only a few iterations are required.

### 4.7. Adaptive sizing field

Our algorithm supports spatially-varying edge length targets. We present in this section results based on sizing fields

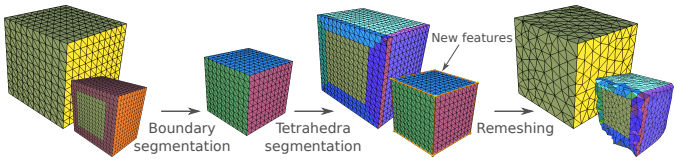


Figure 11: Closed surface features are detected on the outer boundary of the represented domain, and are added to the feature complex by segmenting the imaginary tetrahedra.

that we derive automatically from the input geometry. In particular, in order to get graded meshes of reduced size, we propose to compute the sizing field based on a distance field from the boundaries. This distance field is computed by first discretizing the space inside the triangulation's offseted bounding box, resulting in a voxel grid of a user-defined resolution. In a second step, each voxel is assigned the smallest distance from its center to the set of boundary facets of the input model. These distances are efficiently computed using an acceleration structure detailed in section 5. Finally, the grid values are normalized. Given a target length for the boundary edges $l_B$ and one for the volume edges $l_V$, the target length $l_{ij}$ for the current edge $e_{ij}$ is given by:

$$l_{ij} = (l_V - l_B)D(m_{ij}) + l_B$$

with $m_{ij}$ the edge mid-point and $D : \mathbb{R}^3 \rightarrow \mathbb{R}^+$ the normalized distance field from the boundaries. Note that any user-defined sizing field can be used to tailor the edge length and produce dense meshes in regions of interests.

### 4.8. Additional feature preservation

The preserved 1- and 0- features stem from the multi-material junctions. Nevertheless, the input domain's sharp features are not taken into account. In the following, we explain how to preserve additional features by either segmenting the imaginary tetrahedra, hence creating multi-material junctions, or by explicitly tagging features edges and corner vertices to be preserved during the remeshing process.

*Closed features.* We propose to detect closed features on the outer boundary of the represented domain, and to add them to the feature complex by segmenting the imaginary tetrahedra. To do so, the outer boundary facets - with one incident imaginary tetrahedron - are clustered in surface patches with similar normals delimited by closed feature lines using the Variational Shape Approximation (VSA) method [50].

Detected feature lines represent sharp creases of the input domain. In order to preserve them during the remeshing process, we add them to the feature complex by propagating the facet's segmentation to the imaginary tetrahedra. The imaginary tetrahedra are therefore labeled implicitly, creating 1-junctions where the detected close features lie (see Fig. 11 for an illustraton). In concave regions, facets belonging to two different regions might be connected by critical edges (see Fig. 12). In that case, performing a straightforward flood-filling, propagating the facet segmentation to the imaginary tetrahedra, induces unconnected components, conflict regions and sparse 1- and 0-junctions created on the boundaries (see Fig. 12). To overcome this problem, we split imaginary critical edges as a pre-process.

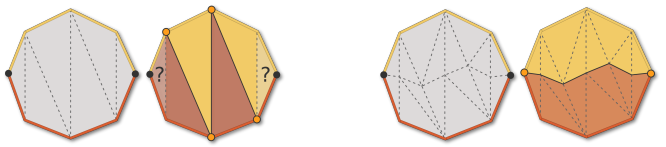

Figure 12: To overcome conflicts that can occur in concave regions, the imaginary critical edges are split as a pre-process.

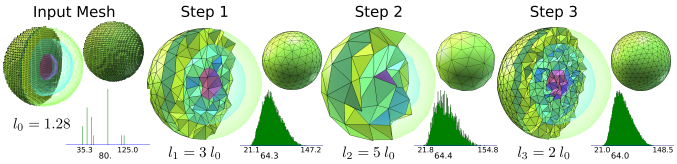


Figure 13: **Parameter space exploration.** High-quality meshes generated for each target length, in 5 iterations plus 2-3 flip and smooth cycles, with dihedral angles between [21.1, 147.2], [21.8, 154.8] and [21.0, 148.5] for step 1, 2 and 3 respectively.

*Marked features.* The described feature detection and preservation method is limited to closed junctions. However, not all relevant feature lines (e.g., surface curvature-based ones) are closed. Hence, we propose to preserve user-prescribed polylines by adding (i) each polyline as a distinct 1-junction to the feature complex and (ii) their end points and intersections as corner vertices. The features are explicitly tagged and not implicitly defined by the multi-material junctions. Hence when splitting a feature edge, the two new edges and the inserted vertex are tagged as features. Consequently, our conditions definitions (see Table 1) still hold. Table 2 summarizes the specific rules for this case: if the user wants to preserve the exact input feature vertices, we set the feature polylines being smoothed by the remeshing process as corner vertices.

| Condition | # $f_i$ | # $f_j$ | Is $e_{ij}$ feat. | Update $v_i$ | Update $v_j$ |
|---|---|---|---|---|---|
| Similarity | 2 | 2 | yes | yes | yes |
| Inclusion | 0 | >= 1 | no | yes | no |
| Inclusion | 2 | 1 or > 2 | yes | yes | no |
| Exclusion | > 0 | > 0 | no | no | no |
| Exclusion | 1 or > 2 | 1 or > 2 | - | no | no |

Table 2: Feature-aware rules for tagged features.

## 5. Results and comparisons

Performances were measured on an Intel Core2 Duo (single thread) at 2.4 GHz with 8GB of main memory. We used the Computational Geometry Algorithms Library (CGAL) 3D triangulation code as an underlying mesh structure and its Axis-Aligned Bounding Box (AABB) tree implementation to efficiently compute the adaptive sizing field. This tree is built using the input mesh's boundary facets and is used as an acceleration structure to compute the distance field. Since the conditions are evaluated using local adjacency information, we do not need to build the feature complex explicitly in contrast to [3]. We only store the list of labels incident to each vertex and construct the MLS representation of the 2-junctions. Note that we tagged the elements of the user-provided features. Our implementation is robust to poorly-shaped tetrahedra with zero or negative volume. Furthermore, any type of triangulation can be remeshed by our method, allowing us to process a wide range of input. We demonstrate the validity of our approach on both synthetic and acquired segmented voxel grids. We apply our method on naive meshes generated from a segmented 3D voxel grid defined through a five-cells decomposition of all the voxels contained in the bounding box of the domain while unifying internal and external boundaries. The resulting high-resolution mesh reproduces the grid topology with aliased boundaries. As explained in the previous section, our rules depend of the element's dimension in the feature complex. For instance, we start by flipping the boundary edges that improve the average valence of the boundary vertices and then flip the volume edge that locally maximizes the minimal dihedral angle. The MLS representation of the surface patches allows smoothing noisy boundaries and features. Furthermore, it allows refining as well as simplifying the input mesh while preserving the boundaries' shape, since the representation is independent from the current mesh state. To evaluate the quality of the resulting meshes, we report the dihedral angle distributions using green histograms and the distributions of radius-ratio (multiplied by 3 for normalization) using orange ones. Fig. 13 illustrates an interaction session where the user navigates between different resolutions. High-quality meshes are generated, within seconds, for three different target lengths, in 5 iterations plus 2-3 flip and smooth cycles. Even though no sliver removal step was performed, the resulting meshes all have dihedral angles above 21 degrees.

Our approach is targeted but not limited to multi-material input domains, and we evaluate our approach on single material input meshes as well (see Fig. 1, 14, 15 and 16). We can observe the good angle distribution and assess visually the overall mesh quality. We compare our method with state-of-the-art single-material meshing techniques. In Fig. 15, the meshes of the first row are generated from the same Sphere surface mesh using the same size and quality parameters. The first mesh is generated by a Delaunay Refinement process and the second by using DelPSC [11]. The meshes of the second row are generated using the Refinement mesh as an input. We performed our remeshing process, using 5 iterations and 3 flip-smooth cycles, in 20 seconds and performed an ODT [51, 52] optimization with the same time limit. We can see that our method does not produce slivers since the smallest dihedral angle is 23.2 degrees, whereas other methods need an additional optimization

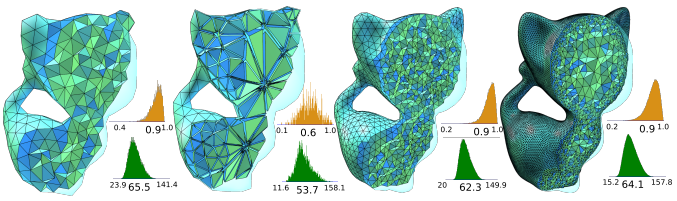

Figure 14: **Kitten.** High-quality isotropic and adaptive remeshing.

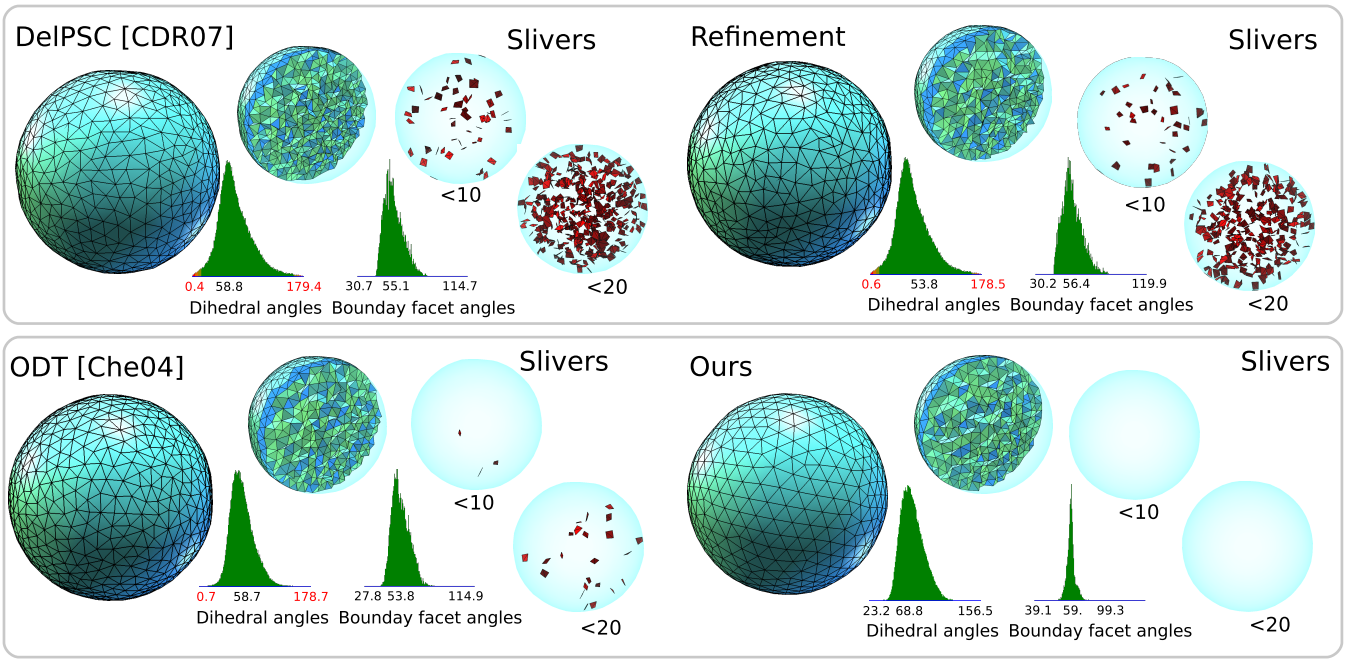


Figure 15: **Comparison.** First row: Mesh generated from the same Sphere surface mesh using the same size and quality parameters. The first mesh is generated by using DelPSC [11] and the second by a Delaunay Refinement process. Second row: Meshes generated using the Refinement mesh as an input.

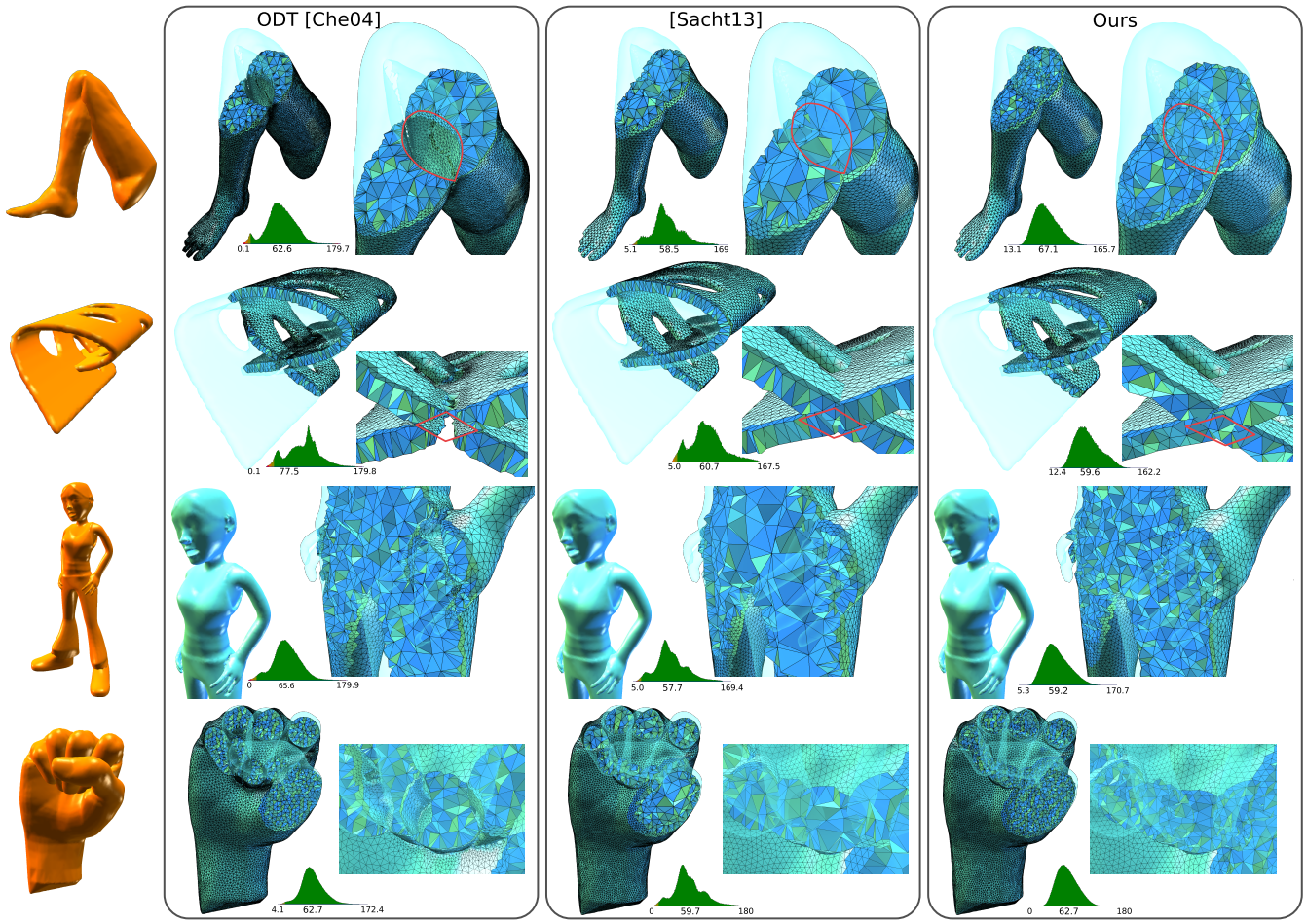


Figure 16: **Self intersecting mesh processing.** Note that Delaunay-based techniques, such as ODT can not process self-intersecting surfaces. The green histograms show the dihedral angle distributions.

| Model | Input. #tet | $s_B - s_V$ | Output #tet | In complex #tet. | Timing |
|---|---|---|---|---|---|
| Spheres | 3 999 | 1 | 20 062 | 6 165 | 8s |
| | | 1-5 | 23 588 | 2 000 | 10s |
| | | 0.6-5 | 44 784 | 10 669 | 31s |
| 4 labels sphere | 750 000 | 2 | 85 853 | 30 834 | 1m24s |
| | | 6 | 3 192 | 1 153 | 6s |
| | | 4-5 | 5 470 | 2 074 | 20s |
| Hepatic Vessel | 4 757 904 | 1.1-8 | 679 580 | 187 616 | 20m |
| | | 2.5-8 | 61 170 | 16 546 | 3m20s |
| Hand | 4 158 720 | 2.4 - 8 | 3 717 750 | 88 167 | 8m38s |
| | | 1.1 - 8 | 617 920 | 219 739 | 7m30s |
| Kitten | 371 183 | 7.6 | 9 439 | 3 393 | 57s |
| | | 7.6-8 | 6 291 | 856 | 2s |
| | | 3.6 | 299 669 | 31 322 | 53s |
| | | 1.1-13 | 784 717 | 244 424 | 14m08s |

Table 3: **Performances.** The results are obtained by setting two scalars values $s_B$ and $s_V$ for adaptive meshes, or a single scalar $s_B = s_V$ for isotropic meshes, defining the target edge lengths as $l_B = s_B * l_0$ for the boundaries and $l_V = s_v * l_B$ for the volume, $l_0$ denoting the average input boundary edge length. Output #tet is the number of tetrahedra in the triangulation and in complex and #tet displays the number of tetrahedra that belong to the input domain, i. e. that are not imaginary. Note that further increased performance is to be expected, if one desires to ignore imaginary tetrahedra in the mesh.

step. Note that on a similar example, NODT [13] performs better than ODT but also produces slivers. Our method prevents the apparition of degenerated tetrahedra with a large longest to shortest edge ratio, such as needles and wedges. Since the edge lengths are equalized, most of the slivers and spindles are removed on account of the angle-based flip and the remaining few are taken out as a final process, along with the cap tetrahedra (see Fig. 2). Note that this final step is often unnecessary.

Since only local topological operators are used, our method allows processing self-intersecting volumes meshes. In Fig. 16, we show (i) on the left results of a Delaunay-based meshing method of a self-intersecting input domain creating holes in the output mesh, (ii) in the middle the tetrahedral meshes generated by the method proposed by Sacht et al. [53] and (iii) on the right our remeshing resulting using the later meshes as an input. Note that the output of Sacht et al. [53] (Fig. 16) have been optimized in *a different pose*: they smooth the input surface until intersections are resolved, triangulate the resulting smoothed surface and only then put the triangulation in a geometric state that is compatible with the input geometry of the surface. On the contrary, we optimize the mesh in its input pose directly.

Last, we present additional synthetic results (see Fig. 17) and apply our method on segmented medical images (see Fig. 1 and 17). Note that the extremely low dihedral angles (bottom left mesh in Fig. 17) correspond to isolated tetrahedra or small features w.r.t. the target edge length in the input. Our method optimizes the geometry under the hard constraint of exact feature preservation. Table 3 shows the performances for the remeshing of the presented various input data sets. Note that the performances strongly depend on the input and the edge aimed length. Timings are obtained using the mesh resulting from the previous remeshing session as an input, in order to emphasize the dependance to the *current state* instead of the *input state*. We made sure to present sessions where the resolution was changed smoothly and others where the resolution was changed arbitrarily, to balance illustration and fairness of comparison with existing previous work.

## 6. Limitations & Future Work

Our feature-aware operations strictly preserve the input topology, and this property may cause problems for noisy datasets. Indeed, poorly-shaped tetrahedra are generated when the target edge length is significantly larger than the size of the feature to preserve. To tackle this issue, part of these feature-preserving conditions could be relaxed or the target edge length could be changed during a post-processing stage. Additionally, when starting from a segmented voxel grid, a pre-processing step can be performed to remove isolated voxels and small features as well as merge corner vertices being too close [3]. More generally, groups of poorly-shaped tetrahedra could be removed using a feature preserving version of the vertex insertion strategy proposed in a single material setting [26]. These processes can be combined with a roll back mechanism in order to cancel the operations that did not improve the quality. Since we use local topological operations only, our remeshing method can be applied on a portion of the mesh using only local neighborhood information. Therefore, based on the streaming algorithm for compressing tetrahedral volume meshes proposed [28], an extension of our method to model data that do not fit in main memory seems foreseeable.

## 7. Conclusion

We have proposed a new efficient multi-domain adaptive remesher for tetrahedral meshes. Our approach is based on local operations which simultaneously refine or simplifying the tetrahedra while improving the quality through local topology changes and point relocations. Additionally, our framework allows to preserve features detected on the outer boundary of the domain as well as user-defined features. By decorrelating the piecewise smooth boundary model from the mesh resolution using an MLS approach, our method provides high-quality meshes at different resolutions using well known simple local topological operators and is general enough to be applied to any structured mesh. As a result, our high-quality adaptive remesher can compete with state-of-the-art methods [3] but is free from the computation of a Delaunay triangulation/Voronoi diagram of multi-material domains, requires only to build a PSC instead, has lower memory cost, can process self-intersecting meshes, and is significantly easier to implement.


[1] Shewchuk JR. What is a good linear element? - interpolation, conditioning, and quality measures. In: 11th International Meshing Roundtable. 2002, p. 115–26.

[2] Boltcheva D, Yvinec M, Boissonnat JD. Feature preserving Delaunay mesh generation from 3d multi-material images. In: Symposium on Geometry Processing. 2009, p. 1455–64.

[3] Dey TK, Janoos F, Levine JA. Meshing interfaces of multi-label data with Delaunay refinement. Engineering with Computers 2012;28:71–82.

[4] Alexa M, Adamson A. Interpolatory point set surfaces-convexity and hermite data. ACM Transactions on Graphics (TOG) 2009;28:20:1–20:10.

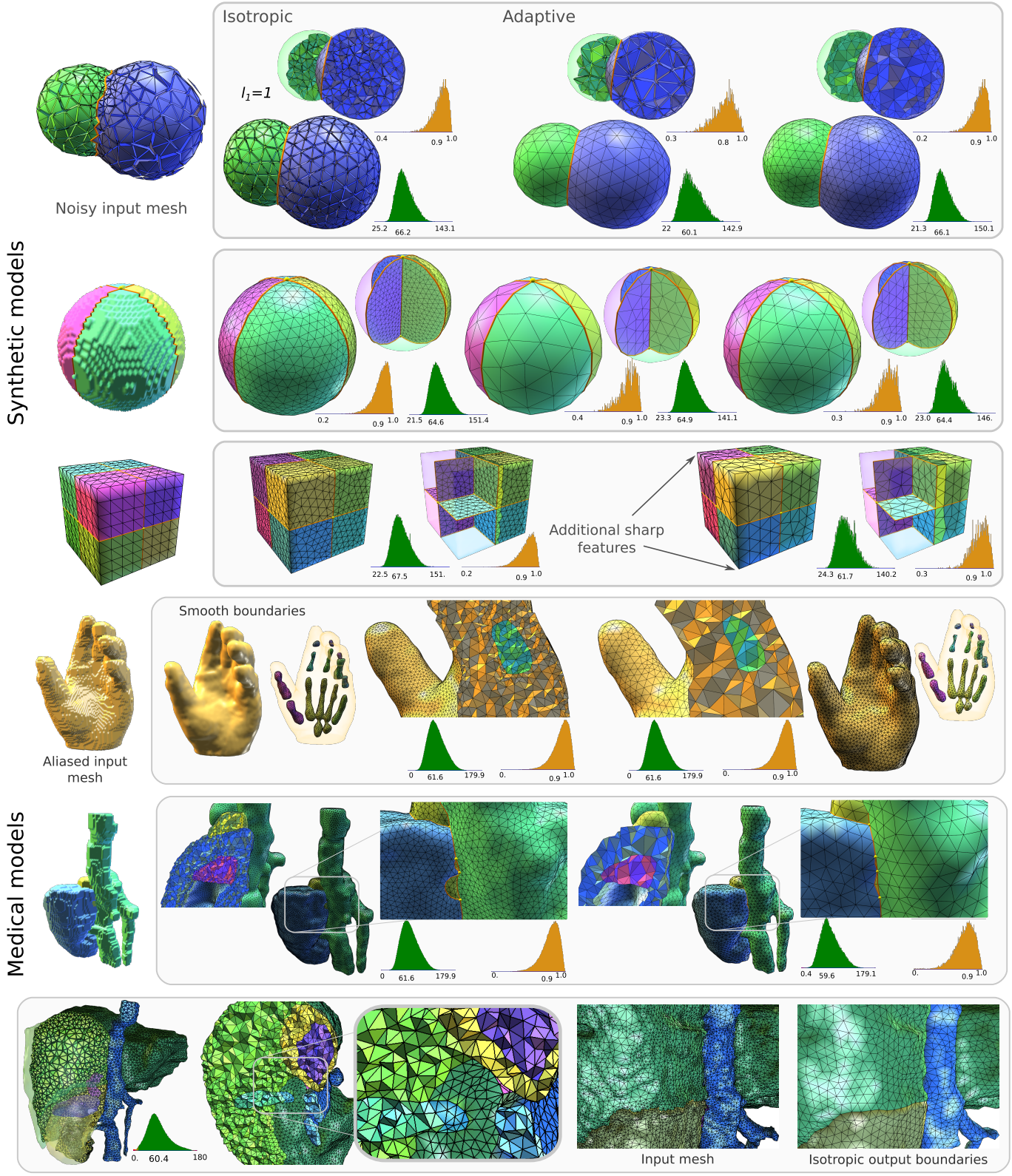


Figure 17: **Remeshing synthetic and medical data at different resolutions.** The last row of the synthetic data illustrates the addition of closed features to the feature complex which are detected on the mesh outer boundary. The green histograms show the dihedral angle distributions and the orange ones display the radius-ratio distributions.

[5] Chew LP. Guaranteed-quality mesh generation for curved surfaces. In: Symp. on Computational geometry. 1993, p. 274–80.

[6] Ruppert J. A delaunay refinement algorithm for quality 2-dimensional mesh generation. J Algorithms 1995;18:548–85.

[7] Shewchuk JR. Tetrahedral mesh generation by Delaunay refinement. In: Fourteenth annual symposium on Computational geometry. 1998, p. 86–95.

[8] Boissonnat JD, Oudot S. Provably good sampling and meshing of surfaces. Graphical Models 2005;67:405 –51.

[9] Cheng SW, Dey TK. Quality meshing with weighted Delaunay refinement. SIAM Journal on Computing 2003;33:69–93.

[10] Cheng SW, Dey TK, Levine A. A practical Delaunay meshing algorithm for a large class of domains. In: Proceedings of the 16th International Meshing Roundtable. 2007, p. 477–94.

[11] Cheng Sw, Dey TK, Ramos EA. Delaunay refinement for piecewise smooth complexes. In: 18th Annual ACM-SIAM Symposium Discrete Algorithms. 2007, p. 1096–105.

[12] Dey T, Levine J. Delaunay meshing of piecewise smooth complexes without expensive predicates. Algorithms 2009;2:1327–49.

[13] Tournois J, Wormser C, Alliez P, Desbrun M. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. ACM Transactions on Graphics (TOG) 2009;28(3):Art–No.

[14] Pons JP, Ségonne F, Boissonnat JD, Rineau L, Yvinec M, Keriven R. High-quality consistent meshing of multi-label datasets. In: 20th international conference on Information processing in medical imaging. 2007, p. 198–210.

[15] Lorensen WE, Cline HE. Marching cubes: A high resolution 3d surface construction algorithm. ACM Siggraph Computer Graphics 1987;21(4).

[16] Wu Z, Sullivan JM. Multiple material marching cubes algorithm. International Journal for Numerical Methods in Engineering 2003;58:189–207.

[17] Labelle F, Shewchuk JR. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. ACM Transactions on Graphics (TOG) 2007;26.

[18] Bronson J, Levine J, Whitaker R. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In: Jiao X, Weill JC, editors. 21st International Meshing Roundtable. 2013, p. 191–209.

[19] Crossno P, Angel E. Isosurface extraction using particle systems. In: IEEE Visualization. 1997, p. 495–8.

[20] Meyer M, Whitaker R, Kirby R, Ledergerber C, Pfister H. Particle-based sampling and meshing of surfaces in multimaterial volumes. Visualization and Computer Graphics, IEEE Transactions on 2008;14:1539 –46.

[21] Dardenne J, Valette S, Siauve N, Burais N, Prost R. Variational tetrahedral mesh generation from discrete volume data. The Visual Computer 2009;25:401–10.

[22] Goksel O, Salcudean S. Image-based variational meshing. Medical Imaging, IEEE Transactions on 2011;30:11 –21.

[23] Cheng SW, Dey TK, Edelsbrunner H, Facello MA, Teng SH. Silver exudation. J ACM 2000;47:883–904.

[24] Tournois J, Srinivasan R, Alliez P. Perturbing Slivers in 3D Delaunay Meshes. In: 18th International Meshing Roundtable. 2009, p. 157–73.

[25] Freitag LA, Ollivier-gooch C. Tetrahedral mesh improvement using swapping and smoothing. International Journal for Numerical Mthods In Engineering 1997;40:3979–4002.

[26] Klingner BM, Shewchuk JR. Agressive tetrahedral mesh improvement. In: Proceedings of the 16th International Meshing Roundtable. 2007, p. 3–23.

[27] Garland M, Zhou Y. Quadric-based simplification in any dimension. ACM Transactions on Graphics (TOG) 2005;24:209–39.

[28] Isenburg M, Lindstrom P, Gumhold S, Shewchuk J. Streaming compression of tetrahedral volume meshes. In: Proceedings of Graphics Interface 2006. Canadian Information Processing Society; 2006, p. 115–21.

[29] Hoppe H. Progressive meshes. In: 23rd annual conference on Computer graphics and interactive techniques. 1996, p. 99–108.

[30] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: 24th annual conference on Computer graphics and interactive techniques. 1997, p. 209–16.

[31] Uesu D, Bavoil L, Fleishman S, Shepherd J, Silva CT. Simplification of unstructured tetrahedral meshes by point sampling. 2005.

[32] Farias R, Mitchell J, Silva C, Wylie B. Time-critical rendering of irregular grids. In: Computer Graphics and Image Processing. 2000, p. 243 –50.

[33] Alauzet F, Li X, Seol ES, Shephard MS. Parallel anisotropic 3d mesh adaptation by mesh modification. Engineering with Computers 2006;21(3):247–58.

[34] Loseille A, Löhner R. Robust boundary layer mesh generation. In: Proceedings of the 21st International Meshing Roundtable. Springer; 2013, p. 493–511.

[35] Dapogny C, Dobrzynski C, Frey P. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. Journal of Computational Physics 2014;262:358–78.

[36] Cutler B, Dorsey J, McMillan L. Simplification and improvement of tetrahedral models for simulation. In: Proc. Symposium on Geometry Processing. 2004, p. 93–102.

[37] Thomas DM, Natarajan V, Bonneau GP. Link Conditions for Simplifying Meshes with Embedded Structures. Transactions on Visualization and Computer Graphics 2010;:1007–19.

[38] Vivodtzev F, Bonneau GP, Hahmann S, Hagen H. Substructure topology preserving simplification of tetrahedral meshes. In: Topological Methods in Data Analysis and Visualization. 2011, p. 55–66.

[39] Kobbelt L, Bareuther T, peter Seidel H. Multiresolution shape deformations for meshes with dynamic vertex connectivity. 2000.

[40] Vorsatz J, Rössl C, peter Seidel H. Dynamic remeshing and applications. In: Department, Stony Brook University. Her. 2003, p. 167–75.

[41] Botsch M, Kobbelt L. A remeshing approach to multiresolution modeling. In: Symposium on Geometry Processing. 2004, p. 185–92.

[42] Kraus M, Ertl T. Simplification of nonconvex tetrahedral meshes. In: Hierarchical and Geometrical Methods in Scientific Visualization. 2002, p. 185–96.

[43] Botsch M. High quality surface generation and efficient multiresolution editing based on triangle meshes. Ph.D. thesis; 2005.

[44] Adamson A, Alexa M. Point-sampled cell complexes. In: ACM Transactions on Graphics (TOG); vol. 25. 2006, p. 671–80.

[45] Adamson A, Alexa M. Approximating bounded, non-orientable surfaces from points. In: Shape Modeling International. 2004, p. 243–52.

[46] Oztireli C, Guennebaud G, Gross M. Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression. Computer Graphics Forum 2009;28:493–501.

[47] Cheng ZQ, Wang YZ, Li B, Xu K, Dang G, Jin SY. A survey of methods for moving least squares surfaces. In: Point-Based Graphics. 2008, p. 9–23.

[48] Dey T, Edelsbrunner H, Guha S, Nekhayev D. Topology preserving edge contraction. Publ Inst Math(Beograd)(NS) 1999;66:23–45.

[49] de l'Isle EB, George. PL. Optimization of tetrahedral meshes. In: Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, IMA Volumes in Mathematics and its Applications. 1995, p. 97–128.

[50] Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. ACM Transactions on Graphics (TOG) 2004;23:905–14.

[51] Chen L. Mesh smoothing schemes based on optimal Delaunay triangulations. In: 13th International Meshing Roundtable. 2004, p. 109–20.

[52] Alliez P, Cohen-Steiner D, Yvinec M, Desbrun M. Variational tetrahedral meshing. ACM Transactions on Graphics (TOG) 2005;:617–25.

[53] Sacht L, Jacobson A, Panozzo D, Schüller C, Sorkine-Hornung O. Consistent volumetric discretizations inside self-intersecting surfaces. Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing) 2013;32(5):147–56.