Automatic and High-quality Surface Mesh Generation for CAD Models

Jianwei Guo[a], Fan Ding[a], Xiaohong Jia[b,c,*], Dong-Ming Yan[a]

[a]*National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*
[b]*KLMM, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*
[c]*School of Mathematical Science, University of Chinese Academy of Sciences, Beijing 100049, China*

Abstract

In this paper, we present a fully automatic framework that tessellates industrial computer-aided design (CAD) models into high-quality triangular meshes. In contrast to previous approaches that are purely parametric or performed directly in 3D space, our method is based on a remeshing algorithm that can achieve accuracy and high-quality simultaneously. Given an input standard CAD model, which is represented by B-rep format, we first rebuild the parametric domain for each surface patch according to an initial triangulation, and then the boundaries of the parametric domain are retriangulated by exploiting the *Constrained Delaunay Triangulation* (CDT). In the second stage, the 2D triangulation is projected back to the 3D space, and a modified global isotropic remeshing process is applied, which further improves the regularity and angle quality of the tessellated meshes. Experiments demonstrate the validity of the proposed approach and its ability to generate high-quality meshes. Moreover, we evaluate our technique and compare it with state-of-the-art CAD model tessellation approaches.

Keywords: CAD Tessellation, High-quality mesh, Remeshing.

1. Introduction

Tessellating *Computer-Aided Design* (CAD) models into discrete representations is important in various downstream applications [1, 2] ranging from engineering to scientific research, such as numerical simulations, rapid prototyping, computational fluid dynamics, and visualization, to name a few. The tessellation process aims to generate a discrete mesh that approximates a CAD model with simple discrete elements, e.g., triangles/quads for a surface mesh and tetrahedra/pyramids/hexahedra for a volumetric mesh. Among all types of tessellations, triangular surface mesh generation attracts the most attention because of its simplicity and flexibility, which is also a prerequisite for many volumetric tetrahedral meshing procedures [3].

Although numerous meshing methods are available, automated generation of high-quality meshes remains a challenge [4, 5]. Even with modern commercial software (e.g., Ansys and Hypermesh) and open-source packages (e.g., NetGen [6] and Gmsh [7]), generating correct and satisfying meshes is still a time consuming process that involves an excessive amount of human effort. As the complexity of the constructed CAD products increases, existing methods cannot always achieve accurate output due to incorrect, degenerate or ambiguous geometric designs in the CAD system, as illustrated in Fig. 1. Even post-processing with mesh repair algorithms (e.g., MeshFix [8] and PolyMender [9]) experience difficulties in dealing with these problems. Moreover, even some CAD models have correct topology and geometry, and also contain many small-scale features (e.g., tiny surfaces or thin blending strips, as shown in Fig. 13) that do not contribute to downstream applications. Preserving such small features creates dense triangulations locally.

From a practical point of view, the ability to generate high-quality meshes is the key to the success of numerical analyses [11]. A good mesh usually achieves balance between quality and computation time, whereas low-quality (extremely thin or badly distorted) elements often hinder the solution convergence and increase analysis errors. Furthermore, surface meshes are used as input to volume mesh generators, thus considerably influencing the generation of quality volumetric meshes and furthering the numerical results.

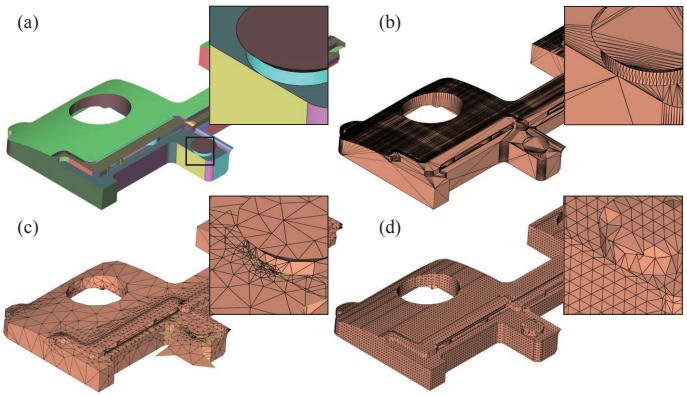


Figure 1: (a) An input CAD model; (b) Tessellation result by using an open-source toolkit called *Open Cascade Technology* (OCCT) [10]; (c) Tessellation result of NetGen [6]; (d) Mesh repair result [9] taking (b) as input.

In this paper, we focus on generating isotropic surface meshes from CAD models that enhance existing approaches in several aspects. The goal is to obtain well-shaped tessellations of CAD models with high accuracy, while considering the simplicity of patch layout and feature preservation. On the one hand, the system robustly handles complex CAD models even with the (near)-degenerate cases (Sec. 4.1) that commonly occur in the conceptual-to-early design of industrial products. We address this issue by first detecting such configurations in an initial triangulation and then removing non-manifold edges or inserting "anchor points" in the parametric domain. Then the use of *constrained Delaunay triangulation* (CDT) can generate correct re-

*Corresponding author
Email address: xhjia@amss.ac.cn (Xiaohong Jia)

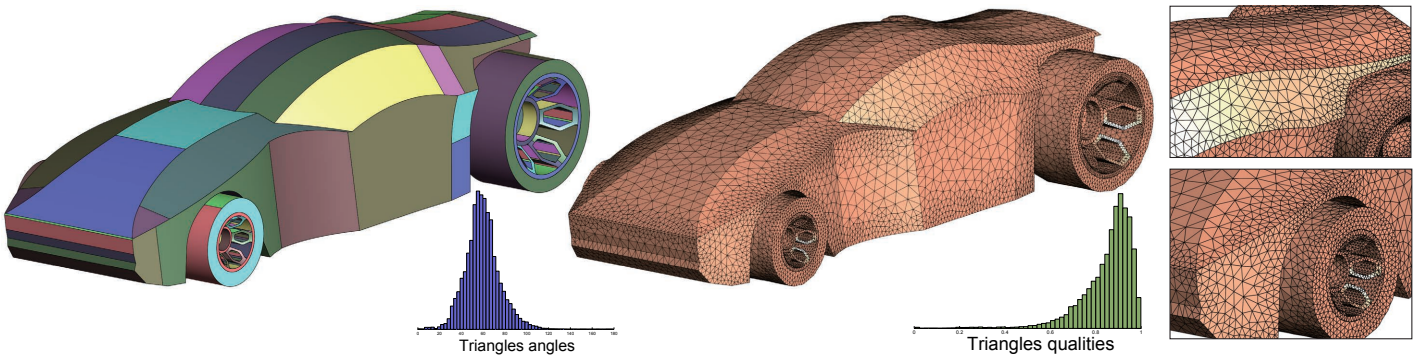


Figure 2: Our algorithm converts an input standard CAD model (881 patches) into a high-quality triangular mesh. At the bottom, we show the histograms of the angle (blue) and triangle quality (green) distributions.

sults. Moreover, we produce high-quality meshes for complex geometric shapes with smooth sizing gradation control. In this step, we first perform patch clustering and constraint simplification to suppress undesired internal features that lead to low-quality elements. Then, we exploit an improved version of a real-time isotropic remeshing technique [12, 13] that applies a series of local operators for mesh optimization. In summary, the main contributions of this work include the following:

- an automatic and robust surface tessellation framework that could generate correct surface meshes from complex industrial CAD models, especially by handling two common cases of degeneracies encountered in many industrial CAD geometries during mesh generation.
- improvement of a state-of-the-art remeshing approach to generate high-quality surface meshes, particularly for angles and regularity.
- introduction of simple yet effective heuristics to merge the small patches and remove the internal feature boundaries, which are basically irrelevant to an overall simulation of the CAD model. Furthermore, this approach allows generating a smooth geometry for improved meshing.

2. Related work

CAD mesh generation. Over the past decades, numerous tessellation methods have been available for meshing analytical 3D surfaces by different geometric primitives, e.g., triangles, quadrilaterals or general *n*-sided polygons. In this section, we restrict the discussion to the most related works, which focus on triangular mesh generation. For more comprehensive discussions, we refer the reader to the survey [14] and textbooks [15, 2, 16].

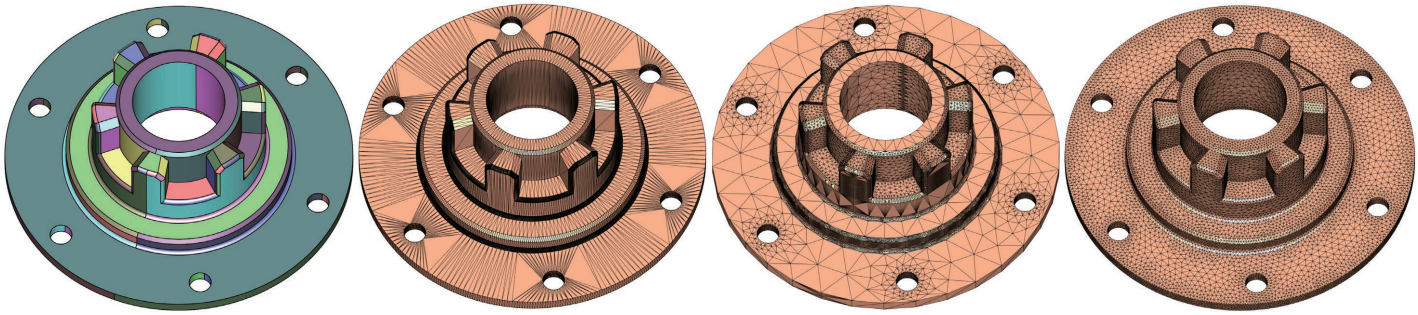
A recent survey by Shimada [17] discusses current trends and issues in meshing and geometric processing for computational engineering analyses. In general, these triangulation algorithms can be categorized into three types:

(1) *Direct approaches*, which work directly on the 3D surface in the physical space. The commonly used approaches include Delaunay triangulation [18, 19, 20], *Advancing Front Technique* (AFT) [21, 22, 23], and octree-based approaches [24, 25]. However, direct methods cannot guarantee the validity of the mesh, e.g., Delaunay-based methods have difficulty in recovering surface boundaries [26], whereas methods using AFT can encounter problems associated with colliding fronts [27].

(2) *Purely parametric approaches (also called indirect approaches)* [28, 29, 30, 31, 32, 33]. This type of method utilizes bijective mapping between the surface and a parametric domain. The parametric space is triangulated by using any 2D mesh generation procedure, and then the mesh is mapped back onto the original surface. However, this approach introduces size stretching and shape distortion of the elements in degenerated or badly parameterized surfaces when the mapping is not affine.

(3) *Hybrid approaches*. Considering the advantages and disadvantages of the previous approaches, recent algorithms utilize 3D geometric and 2D parametric information to produce high-quality meshes. Starting from an approximated boundary representation (using STL file format), Béchet et al. [34] propose an adaptive surface mesh generating method. The principle used to generate the mesh is based on the Delaunay method, which is associated with refinement and smoothing operations. Wang et al. [35] propose a remeshing-based algorithm, called EQSM, to generate unstructured triangular meshes. An initial triangulation of the points defining the intersection curves of each surface region is generated in the parametric domain. These meshes are then mapped onto 3D space and refined by basic remeshing operations. However, the input of this method requires several restrictions, e.g., the mapping of each patch should be bijective everywhere, and the boundary should be closed. Marchandise et al. [36, 37] present several different parameterization techniques for quality surface remeshing, where the implementation of those mappings, as well as the boundary conditions, have been presented in a comprehensive and unified manner. Aubry et al. [38] present a novel parametric surface meshing technique, which includes semi-structured boundary-layer surface mesh generation and advancing front point creation approaches. Thereafter, [39] extends this approach to obtain a robust anisotropic tessellation technique. However, this approach only handles NURBS-based geometry. Our approach also falls into this category. Compared with the aforementioned methods, our approach can deal with more complex composite parametric surfaces, as well as achieve higher quality.

Model repair. Although various mesh generation methods exist, the tessellated CAD models may still exhibit topological or geometrical defects. Thus, mesh repair, which converts an imperfect CAD model into a clean and manifold closed triangular mesh, is a highly important task. This topic has been extensively studied [40, 41, 42] and a website featuring freely obtainable implementations of several repairing methods is available at `http://www.meshrepair.org`. According to the approach employed, repairing algorithms can be distinguished by local correction or global remeshing. Typical local approaches [43, 44, 45, 46, 8]



(a) Input CAD model (b) Initial triangulation (c) CDT in parametric domain (d) Global remeshing

Figure 3: The pipeline of our framework based on a two-stage remeshing approach.

are performed directly on the input tessellation and only modify the mesh in a small region around individual defects and flaws. By contrast, global methods [47, 48, 49] are typically based on a complete remeshing of the input by using some intermediate data structures (such as volume representation [50, 9, 51]). However, no available approach can address all the defects because the type of defects depends on the upstream and downstream applications in a given scenario. Most repairing algorithms focus on certain defect types and ignore or even introduce other flaws. Furthermore, they do not consider improving the triangle quality of the output mesh, and usually a post-process is required in practical applications.

3. Preliminaries and overview

In this section, we present an overview of the proposed approach. To precisely explain our techniques, we first provide the related terminology used in the paper.

3.1. Definitions

In solid modeling systems, a CAD model is typically represented by a collection of trimmed parametric surface patches $\{\mathcal{P}_i\}_{i=1}^n$. Each patch $\mathcal{P}_i$ is represented by an analytically defined mapping, denoted as $S(u,v) = (x(u,v), y(u,v), z(u,v))$, from a bounded domain of $\mathbb{R}^2$, called *parametric space*, into a 3D coordinate system $\mathbb{R}^3$, called *object space*. Furthermore, each patch $\mathcal{P}_i$ is uniquely identified by its patch ID i and equipped with a surface type C_i, such as plane, cylinder, sphere, and B-spline surface.

To explore the topologic and geometric entities in the CAD model, several third-party CAD kernels are available, e.g., OCCT [10], *Computational Analysis Programming Interface* (CAPRI) [52]. In our implementation, we use OCCT to access the geometry of shapes represented in B-rep format. It also provides access to the topology of the model, including the connectivity information between different patches. However, to operate our mesher, the user is not required to have a deep knowledge of OCCT. The only parameter for OCCT that can be specified by the user is the tessellation *precision*, and this parameter is set as 100 (the default value) in all our experiments.

3.2. Overview

The input to our framework is a standard CAD model, $\mathcal{G} = \{\mathcal{P}_i\}_{i=1}^n$, which consists of n trimmed parametric surface patches that can meet non-smoothly on boundaries. In addition, the user is required to specify the target edge length, l_{target}, as the desired size of the approximation mesh. The output is a 2-manifold watertight triangular mesh $\mathcal{M} = \{t_i\}_{i=1}^m$, consisting of as much as possible of quasi-equilateral triangles t_i. As shown in Fig. 3, our algorithm undergoes four main stages:

1. *Initial triangulation:* Initially, the CAD model $\mathcal{G}$ is converted into a simple triangulation by using OCCT, where the geometric fidelity (measured as approximation error) is guaranteed.
2. *(Near-)degeneracy handling (optional):* We detect the incorrect triangulation caused by poorly designed geometry and repair them by removing non-manifold edges or inserting "anchor points", and then compute CDT in the parametric domain.
3. *Parametric remeshing:* Based on initial triangulation, we rebuild the parametric domain for each patch and reapply CDT to preserve the boundaries for each parametric domain. Then, the 2D triangulation is projected back to 3D space.
4. *Global remeshing:* To improve the mesh quality, a global isotropic remeshing process is applied. The user-specified meshing size is also satisfied in this stage.

Each step involves a precise task while gathering enough information for the next step. Details of each step are given in the following sections.

4. Methodology

4.1. Initial triangulation

The core idea of our algorithm is that we only perform surface remeshing on a discrete model to achieve high efficiency because the edges and facets that depict the discrete surface mesh have linear equations. Therefore, a suitable initial triangulation is required, but it should be generated quickly, and the approximation error should be small.

In our approach, we exploit OCCT to generate the initial triangulation. Although this toolkit is designed to generate triangles for visualization purposes only (leading to non-conforming meshes at face boundaries) and has no gradation in element size, it can control the distance between the initial tessellation and the original analytical surfaces. Thus, geometric fidelity is preserved, i.e., the approximation error is small.

The initial triangulation is not closed yet because each patch $\mathcal{P}_i$ is tessellated into a patch mesh $\mathcal{M}_i$ independently. This condition is favorable because our parametric remeshing stage is still operated in a patch-wise manner. Besides, we collect the original vertices of the input geometry from the B-rep model. These

vertices are then identified in the initial triangulation and marked with a "locked" flag. Thus, if a vertex is marked as "locked", it remains fixed in later steps to preserve the geometric feature unless its flag is changed at a certain stage. We explain this phenomenon and the moment a feature vertex is unlocked in Sec. 4.3.

(Near-)degeneracy handling. As the precision of industrial products increases, a complex CAD model usually contains a large number of surface patches; for example, even the shell of a cellphone can contain hundreds of patches (see Fig. 12). During the construction of the CAD model some operations may generate small artifacts that may not be noticed by the designers. Moreover, the incorrect CAD data arising from translation errors between different CAD products and formats, or even from limitations inside CAD kernels, can cause meshing failures. In this paper, we address two cases of degeneracies encountered in a wide range of complex CAD models in practical engineering applications.

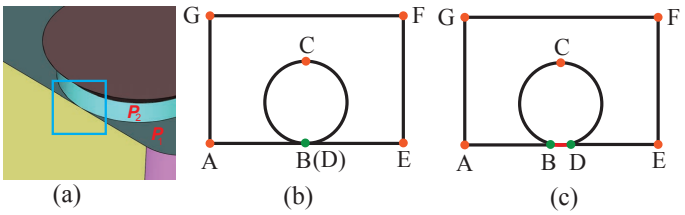


Figure 4: 2D illustration of degeneracy caused by tangency.

(1) *Tangency case:* Non-manifold vertices and edges usually appear due to patch tangency, in which an intersection between the inner and outer boundaries occur. Fig. 4(a) shows such tangency degeneracy, which has barely attracted attention in previous methods. In this case, a planar patch $\mathcal{P}_1$ is tangent to a cylindrical patch $\mathcal{P}_2$. At the position of $\mathcal{P}_1$ where the inner and outer boundaries are tangent, two vertices that coincide or are very close exist, as shown in segment **BD** in Figs. 4(b) and 4(c). As a result, this intersection forms non-manifold triangles because OCCT cannot distinguish the inner and outer regions at this position. To avoid such non-manifold triangles while keeping the shape boundaries, we first detect the planar patch $\mathcal{P}_1$ where the tangency happens by sorting out the original vertices that are linked to multiple (more than two) boundary edges. Then segment **BD** is removed from the constraints when building CDT for this patch, so that only one outer boundary **A** – **B** – **C** – **D** – **E** – **F** – **G** – **A** exists. In other adjacent patches $\mathcal{P}_2$ and $\mathcal{P}_3$, segment **BD** is retained to preserve the boundary. Therefore, they can form a closed manifold mesh in the later stage.

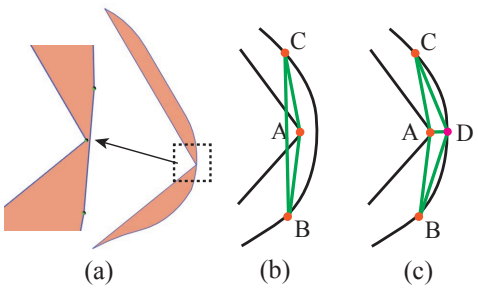


Figure 5: Solution to problem caused by curve proximity.

(2) *Curve proximity case:* Another common case is that one point of the patch is too close to one curve, as shown in Fig. 5(a).

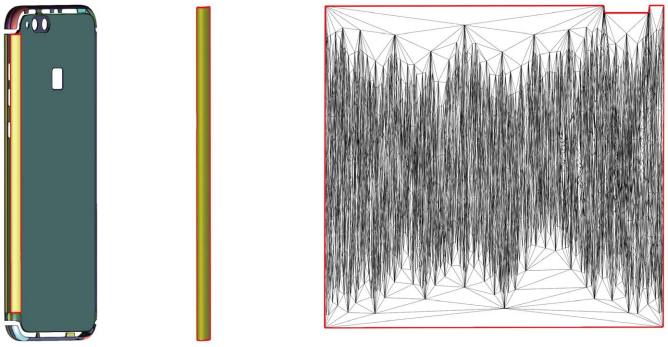


Figure 6: An example of a highly distorted initial parametric space.

Although this case is not a type of degeneracy, it results in intersecting triangles when the boundary sampling rate is sparse (Fig. 5(b)). Thus, it requires dense but unnecessary samples to overcome this problem. In our approach, we calculate the Euclidean distance between a concave point **A** on one border and its nearest border edge to avoid such dense sampling. If the distance is shorter than a threshold (we set it to be $0.04 l_{target}$ where l_{target} is the target edge length), we insert an "anchor point", **D**, on curve $\overset{\frown}{\mathbf{BC}}$, which is the closest point to point **A** (Fig. 5(c)). With the help of this "anchor point", an edge **AD** prohibits the generation of intersecting triangles.

4.2. Parametric remeshing

As each patch $\mathcal{P}_i$ has a 2D parametric representation as designed, therefore a straightforward way to tessellate is to mesh the 2D domain in this parametric space and then map it back to the 3D object space. However, in many cases, such surface mesh generation is easily corrupted. The reason is twofold: (1) the patches in a CAD model are often not topologically connected [53], i.e., small gaps or overlaps between the neighboring patches exist, and the parametrization may be different along one common boundary. These conditions will result in non-conformal triangles; and (2) the parametric space of one patch can be highly/extremely distorted (see Fig. 6). Therefore, a valid mesh in the parametric space may be invalid or at low quality when mapped in a 3D space [54].

To address these problems, we apply a planar parameterization method to rebuild the parametric domain based on the initial triangulation. Thereafter, we recover the boundary edges corresponding to the original curves and perform edge sampling. Finally, the CDT is used to finish the 2D triangulation. Our parametric remeshing is detailed as follows.

Patch reparameterization. We first rebuild a one-to-one mapping from each patch mesh $\mathcal{M}_i$ to a simple 2D domain, using the *Scalable Locally Injective Mappings* (SLIM) parameterization [55]. The result is a pair of parameter coordinates (u, v) for each vertex of the initial triangulation. SLIM is robust and fast by efficiently minimizing a nonlinear, conformal or isometric distortion energy. It is guaranteed to produce optimized maps without any flipped triangles. Furthermore, it corresponds to a locally injective mapping with a free border. Therefore, it is suitable for our method because a complex patch mesh with arbitrarily shaped borders can be parameterized.

Boundary curve recovery and sampling. From the initial triangulation, we recover the 3D boundary edges that correspond to the original curves of the input CAD model. This step aims

Algorithm 1: Boundary curve recovery

Input: a coarse patch mesh $\mathcal{M}_i$;
Output: recovered boundary curves set, $C = \{c_i\}_{i=1}^{n}$;
foreach *halfedge h* **do**
  if *h is a border edge and its vertex is marked as "locked"* **then**
    construct an empty boundary curve $c_i = \varnothing$;
    halfedge $h' \longleftarrow h$;
    do
      add h' into c_i;
      $h' \longleftarrow$ previous halfedge of h' ;
    while *the vertex of h' is not marked as "locked"*;
  add c_i into C;

to preserve the geometry of boundaries in the final mesh. For each patch mesh $\mathcal{M}_i$, we use a tracing scheme for recovery. Here we suppose that the mesh is represented by the half-edge data structure. As shown in Fig. 7, starting from a half-edge h_1 whose vertex is marked as "locked", we recursively visit its previous border half-edge, until another half-edge whose vertex has a flag of "locked" is met. Then these piecewise linear segments form a boundary curve $c_i = \{h_1, ..., h_k\}$. After processing all half-edges, we collect a set of boundary curves, $C = \{c_i\}_{i=1}^{r}$. The pseudo-code of this step is illustrated in Algorithm 1.

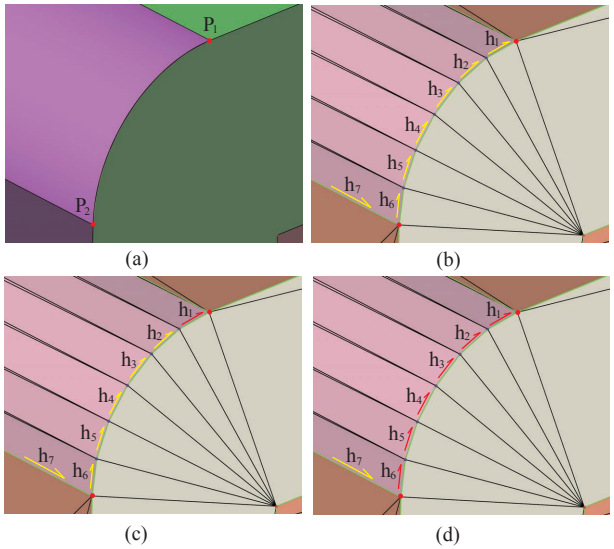


Figure 7: Our tracing scheme for recovering boundary edges. (a) An input parametric curve formed by two points; (b) Half-edge data structure for this curve (we show only one side); (c) Half-edge h_1 whose vertex is marked as "locked" is first identified, shown in red; (d) Other half-edges belonging to this curve are recursively traced until h_7 is met.

Currently, each curve c of the input geometry has been identified in its two incident patch meshes (the neighboring relations between different patches have been stored in advance in the initial triangulation), corresponding to c_i and c_i', respectively. However, one curve can be discretized differently in different patches because OCCT tessellates each patch independently (see Fig. 8). As a result, this discretization causes gaps or overlaps when merging adjacent patch meshes. To achieve consistent discretizations for each shared curve c, we produce the same number of uniform samples on c_i and c_i', respectively. We denote the length

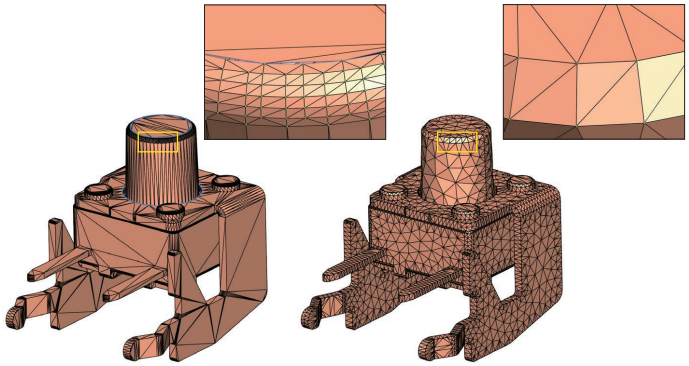


Figure 8: (a) Initial triangulation using OCCT causes gaps or overlaps between neighboring patch meshes. (b) Our boundary curve recovery and sampling could generate consistent vertices along the shared curves.

of curve c as l_c, and the curve sampling interval as $\delta = \alpha l_{target}$ (α is set to $0.6 - 0.8$). Thus, the number of points to be sampled is $N = max(2, \lceil \frac{l_c}{\delta} \rceil)$, where the $max(a, b)$ function returns the greater value of a and b. Finally, we generate N samples uniformly on the piecewise linear segments of c_i and c_i'. The parameter coordinates (u, v) of each sample is interpolated using the values defined on the vertices of c_i and c_i'.

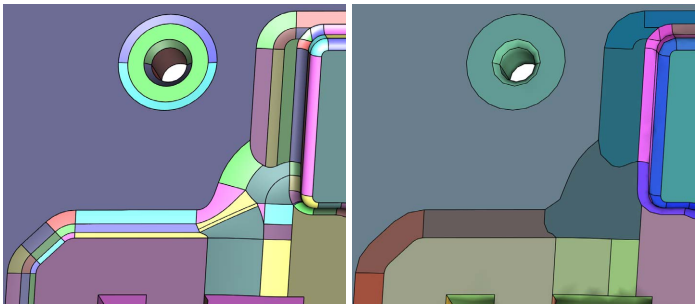
Triangulation. Based on the rebuilt parametric space and recovered boundary curves, we now refine each patch mesh of the coarse triangulation by a parametric remeshing approach. In the 2D space, we organize the sampled points on boundary curves into a bounded region, which is defined by a *Planar Straight Line Graph* (PSLG). The segments of the PSLG are considered as constraints and are fed to our algorithm to build a CDT. Then, an improved mesh is obtained by invoking the Delaunay refinement method with default shape and size criteria[1] (default shape criterion $B = 0.125$, size criterion $s = 0.5$). We do not need to tune the criteria because we will satisfy the user-specified meshing size in the global remeshing stage. Finally, the 2D mesh of each patch is mapped back to the 3D space, and an improved tessellation with correct conformity is obtained, as shown in Fig. 8 (b).

4.3. Global remeshing

Although 2D Delaunay refinement is performed, the quality of parametric remeshing is still unsatisfied. To improve the regularity and angle quality of the final mesh, another high-quality remeshing is required. In our approach, we leverage a real-time isotropic remeshing technique inspired by [12] to refine the mesh. Besides, an industrial CAD product contains many small surfaces and internal "feature" boundaries because it is usually modeled via various Solid Boolean Operations, translations or corrections. Thus, before remeshing, we first remove these small geometric features, in which most unintended discontinuities reside. This global patch-independent remeshing can achieve better quality elements by ignoring the internal CAD patch limitations.

Patch mesh clustering. As illustrated in Fig. 9, a complex CAD model often contains many small surface patches that is not related to the geometry of the model, thus, the arrangement of these patches can not reflect the intention of the designer. The proximity features among these patches lead to low quality elements

[1] http://doc.cgal.org/latest/Mesh_2/



(a) (b)

Figure 9: Illustration of how clustering approach provides us a mesh that is simpler than merely mimicking the CAD topology: (a) patch meshes before grouping, and (b) patch meshes after grouping.

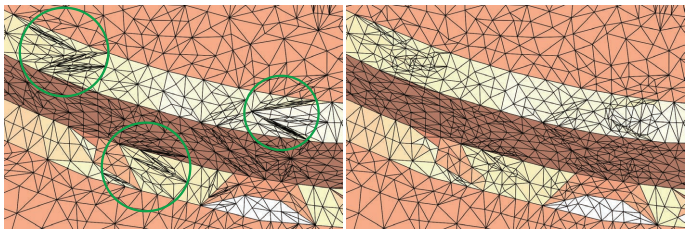


Figure 10: Results before (left) and after (right) edge flipping by angle. Green circles indicate the entangled triangles caused by splitting long edges.

around them because they contribute to constraining the resulting mesh. To suppress these features, we cluster such patches into more desirable subregions using a hierarchical agglomerative clustering algorithm. First, for each patch mesh $\mathcal{M}_i$ consisting of a list of triangles $\{t_j\}$, we compute its weighted centroid $\mathbf{q}_i = \sum |t_j|\mathbf{b}_{t_j} / \sum |t_j|$ and weighted normal $\vec{\boldsymbol{n}}_i = \sum |t_j|\vec{\boldsymbol{n}}_{t_j} / \sum |t_j|$, by averaging the facet barycenter $\mathbf{b}_{t_j}$ and normal $\vec{\boldsymbol{n}}_{t_j}$ of each triangle with area $|t_j|$. Then, we group neighboring patch meshes within a close distance and with similar normals. Specifically, two patch meshes, $\mathcal{M}_i$ and $\mathcal{M}_j$, are grouped together iff:

$$\begin{aligned} \left\|\mathbf{q}_i - \mathbf{q}_j\right\| &< \varepsilon_D, \\ \vec{\boldsymbol{n}}_i \cdot \vec{\boldsymbol{n}}_j &> \varepsilon_A, \end{aligned} \tag{1}$$

where $\cdot$ is the dot product between two normal vectors. We set $\varepsilon_D = 2$ and $\varepsilon_A = 0.8$ by default.

After this step, all patch meshes are assembled by merging common vertices to yield a single mesh. In this merging process, each triangle edge shared by two groups is marked by a "feature" flag.

Constraint simplification. The above clustering approach only works well for small patches. Many undesired internal constraints remain between large patches, which lead to stretched triangles. To remove them, we consider all the "feature" edges along one common boundary between two groups, and compute the dihedral angle between the two incident triangles of each "feature" edge. If all of the dihedral angles are lower than a threshold (default value is 5°), then the "feature" edges on this boundary, as well as their "locked" vertices, are released.

Isotropic remeshing. The general pipeline of our global optimization is similar to that in [12, 13], but we further improve the quality of this algorithm, especially for CAD models. To make the paper self-contained, each local operation is outlined sequentially as follows (⋆ indicates the newly added or improved operations):

- *Edge split.* Given a target edge length l_{target}, we split all edges at their midpoint that are longer than $\frac{4}{3} l_{target}$.
- *Angle optimization*⋆. We flip an edge if the sum of its two opposite angles are larger than 180°. This operation is performed because the initial tessellation forms many edges that are considerably longer than l_{target}. As a result, the above edge split operation generates entangled triangles around these long edges (see Fig. 10), which can lead the algorithm to be stuck in a loop of edge collapsing and splitting. By contrast, our angle optimization could generate better shaped triangles to avoid this problem and could further accelerate the entire remeshing process. This operation is only performed in the first iteration, because the inputs to later iterations are already well-shaped. Therefore, this operation will not conflict with the following valence optimization in later iterations.
- *Edge collapse.* We collapse all edges shorter than $\frac{4}{5} l_{target}$.
- *Valence optimization for inner vertices*⋆. A non-feature edge is flipped if this operation meets two conditions: (1) decreases the squared difference of the valences of the four vertices of the two incident triangles to their optimal value 6 (assuming that no border edges for closed CAD models exist); (2) does not increase the projection distance from the midpoint of this edge to the input CAD model. The former improves valence regularization whereas the latter maintains the approximation error.
- *Valence optimization for feature vertices*⋆. Since the feature edges cannot be flipped, it is difficult to improve the valence of the vertices on feature edges. Thus, we propose valence optimization for feature vertices. We observe that a low-quality triangle with one obtuse angle is usually situated around a feature vertex with valence smaller than 6 (actually, the valence is 4 or 5, because valence 3 is impossible on feature edges). Besides, a feature vertex with a valence larger than 6 results in small angles. To improve the valence, we propagate the valence optimization of feature vertices to inner vertices by edge splitting and collapsing. Fig. 11 illustrates our solution. Vertex **A** is a feature vertex with non-6 valence and its two incident half-edges h_1 and h_2 are on the same side of **A**. We denote θ as the angle between h_1 and h_2, n_t as the number of triangles between h_1 and h_2. The desired number of triangles is computed as $n_t' = \lfloor \frac{\theta}{60^\circ} + 0.5 \rfloor$. Then two cases are considered: (1) If $n_t < n_t'$, as shown in Fig. 11 (a) where $\theta = 180^\circ, n_t = 2, n_t' = 3$, we split the opposite edge **BD** of **A**. The reason for selecting **BD** is that the squared difference between the valence of vertex **C** and the optimal value 6 is smaller than that of **E**. However, splitting introduces another vertex **G** with valence 4, thus, a local edge flip is performed to improve the valence of **G** (see Fig. 11 (b)). (2) If $n_t > n_t'$, as shown in Fig. 11 (c) where $n_t = 4, n_t' = 3$, we collapse the opposite edge **CD** of the smallest angle iteratively until $n_t = n_t'$. But note that the case of $n_t > n_t'$ is rare because above edge collapse and valence optimization for inner vertices can largely prevent this case from happening. Finally, after processing each side of the feature vertices, we can significantly improve the mesh quality near feature edges.
- *Tangential Laplacian smoothing.* To optimize vertex distribution, we relocate the positions by computing the *Opti-*

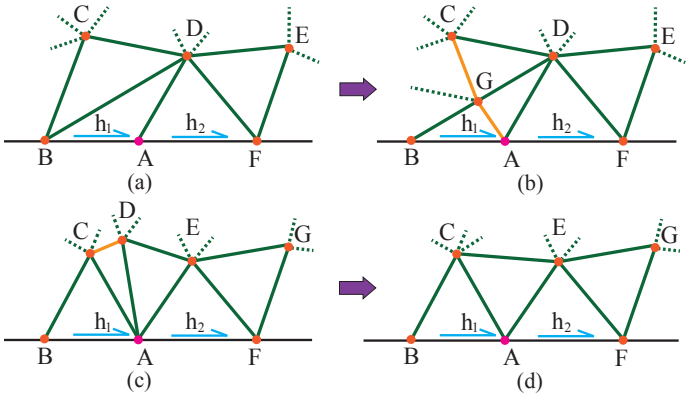


Figure 11: Illustration of the valence optimization for feature vertex.

mal Delaunay Triangulation (ODT). Each vertex is moved to the average $\mathbf{p}_i$ of the barycenter $\mathbf{b}_{t_j}$ of its incident triangles $t_j \in \mathcal{T}$, weighted by the triangle area:

$$\mathbf{p}_i = \frac{\sum_{t_j \in \mathcal{T}} |t_j| \mathbf{b}_{t_j}}{\sum_{t_j \in \mathcal{T}} |t_j|}, \tag{2}$$

Then the new position $\mathbf{p}_i$ can be optionally projected back onto the original surface patch to decrease approximation errors. Since projecting any 3D point back to the underlying surface patch is very slow in OpenCASCADE, to achieve a good balance between the meshing speed and approximation error, we only apply the projections in the first iteration.

Typically, 4−8 iterations are enough to achieve a good tradeoff between the mesh quality and the performance of the algorithm. Finally, after all iterations, we again perform edge flipping by angles. We found that this operation improves the angle quality, without violating the valence regularization.

4.4. Adaptive mesh generation

In engineering and scientific computation, an adaptive tessellation is preferred because the mesh size is smaller near the small features to achieve high geometrical accuracy and is larger elsewhere to avoid increasing mesh counts unnecessarily. Our algorithm can be modified to generate adaptive meshes by applying a sizing function, $\rho(\mathbf{x})$, that defines a target element size at every point on the model. The sizing function can be specified by the user input, or derived from geometrical features. In the remeshing literature, the preference is local feature size (the distance to the medial axis of the model) or curvature. However, the local feature size is suitable for smooth shapes, whereas most CAD models are inherently non-smooth. Thus, we use the curvature controlled metric to generate adaptive surface mesh.

In our pipeline, we only need to modify the isotropic remeshing stage by replacing the constant target edge length with the adaptive sizing field $\rho(\mathbf{x})$ that is sensitive to local curvatures. To compute the local target edge length (for edge split and collapse) and weighted areas (for tangential smoothing), we apply a warp method for mapping. First, we compute the average value $E(\rho)$ of the sizing field. Then, for any edge $e = (\mathbf{x}_a, \mathbf{x}_b)$, the desired local target edge length is defined as:

$$l'_{target} = \frac{2E(\rho)}{\rho(\mathbf{x}_a) + \rho(\mathbf{x}_b)} l_{target}. \tag{3}$$

For the tangential relaxation, the barycenter computation in Eq. 4 is replaced by:

$$\mathbf{p}_i = \frac{\sum_{t_j \in \mathcal{T}} |t_j| w(\mathbf{b}_{t_j}) \mathbf{b}_{t_j}}{\sum_{t_j \in \mathcal{T}} |t_j| w(\mathbf{b}_{t_j})}, \tag{4}$$

where $w(\mathbf{b}_{t_j}) = \frac{\rho(\mathbf{x}_a)+\rho(\mathbf{x}_b)+\rho(\mathbf{x}_c)}{3E(\rho)}$ is the sizing field at the barycenter $\mathbf{b}_{t_j}$ of triangle $t_j = (\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$.

5. Experimental Results

In this section, we first present a number of experimental results that demonstrate the effectiveness of the proposed algorithm. Then we provide a complete comparison with state-of-the-art approaches in terms of the meshing validity and quality. Our algorithm is implemented in C++ using the CGAL library [56] for CDT in 2D domain. All results presented in this paper are obtained on an Intel i7-3770, 3.40 GHz desktop with 16 GB memory.

5.1. Tessellation results

We have evaluated our method on a wide range of CAD models of different complexities, ranging from CAD components to mechanical assemblies. Figures 2, 3, 12, and 13 illustrate some selected tessellation results, as well as the histograms of the angle and triangle quality distributions. The quality of a triangle t is measured by $Q(t) = \frac{6}{\sqrt{3}} \frac{S_t}{p_t h_t}$, where S_t is the area of t, p_t is the half-perimeter of t and h_t is the longest edge length of t [57]. More numerical statistics of the meshing quality are presented in Table 1. Here Q_{avg} is the average triangle quality in the triangulation; θ_{min} and θ_{max} are the minimal and maximal angles, and $\bar{\theta}_{min}$ is the average of the minimal angles of all triangles; $\theta_{<30^\circ}\%$ is the percentages of triangles with angles smaller than 30°; $\theta_{>90^\circ}\%$ and $\theta_{>120^\circ}\%$ are the percentage of triangles with angles larger than 90° and 120°, respectively; $\xi = \frac{1}{n}\sum_{i=1}^{n} |\delta_i - 6|$ is the measure of mesh regularity, where δ_i represents the valence of the ith vertex. The value of ξ is positive, and a smaller value indicates a more regular triangulation. d_{RMS} is the root mean square distance, and d_H is the Hausdorff distance between the output mesh and input CAD model, measured with the Metro tool [58]. Here the input CAD model is approximated with a dense triangular mesh.

From these results, we observe that our approach is robust to tessellate CAD models and to generate high-quality triangular meshes. Furthermore, we can remove the internal feature boundaries to achieve better quality, as shown in Fig. 13. Notice that the output meshes may still have a few low-quality triangles. The reason is that a complex CAD model usually contains various features that affect surface mesh quality, i.e., short curves, curve proximities and small input angles. An ill-shaped and jagged boundary will result in poor quality mesh elements [59]. This result is unavoidable because the features formed by input boundaries cannot be fully suppressed by our clustering approach. Cleaning such features on the geometry level is another topic about CAD defeaturing [60], which is beyond the scope of this paper. However, histograms of the angle and triangle quality distributions demonstrate that we can generate high-quality meshes generally.

Performance evaluation. Our algorithm can achieve greater efficiency than approaches that directly work on smooth surfaces, because most of our geometry computations are performed on the discrete model. Specifically, the processing time mainly comprises three components: initial triangulation, parametric remeshing, and global remeshing. Thanks to OCCT, the first step

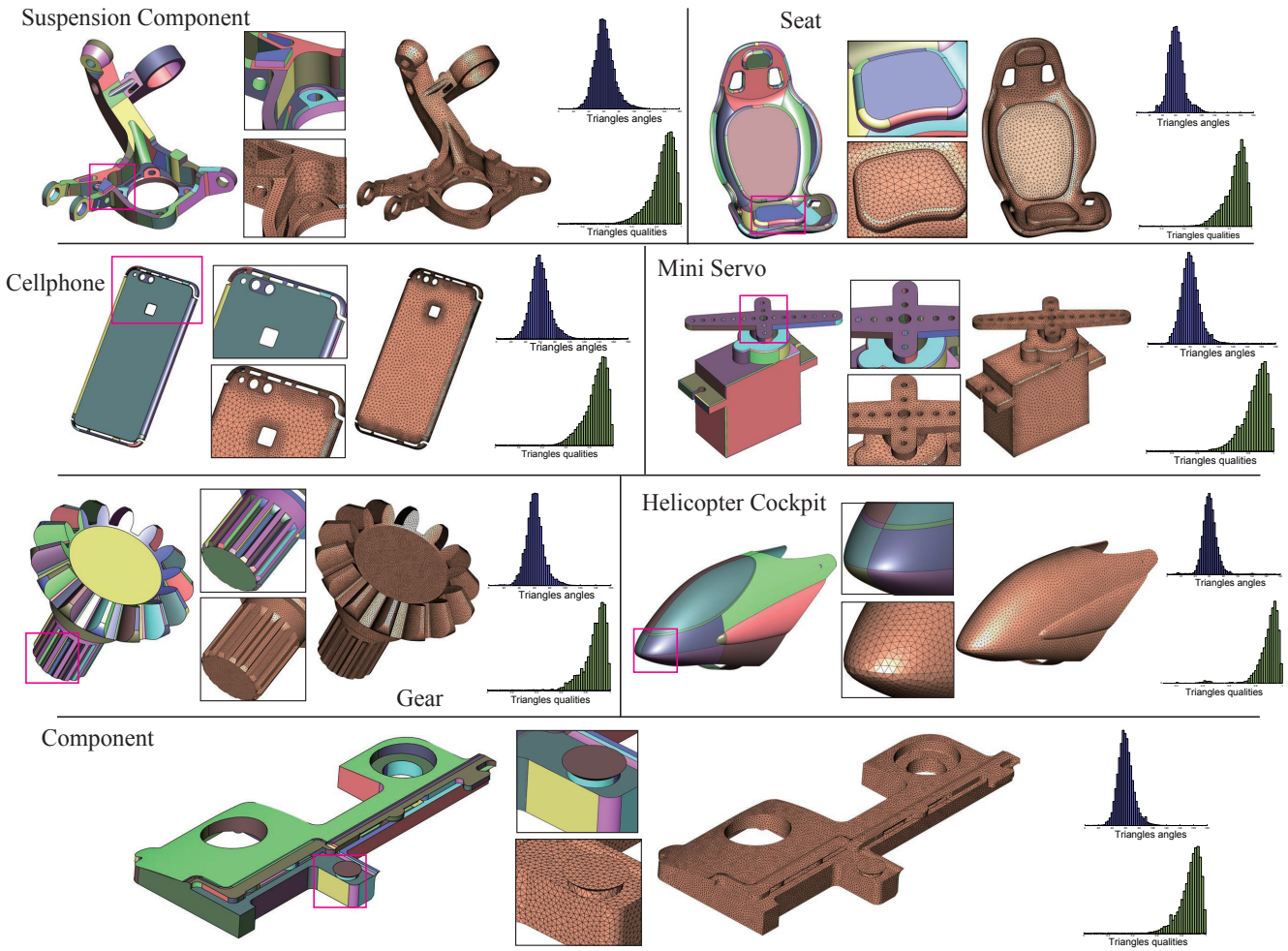


Figure 12: A gallery of examples generated by our framework. For each model, we show the input CAD geometry, tessellation result, zoom-in views, and histograms of the angle (blue) and triangle quality (green) distributions.

Table 1: Our tessellation qualities. $|\mathcal{P}|$ is the number of surface patches in the input; $|\mathcal{M}|$ is the number of triangles in the output. The other measurements are explained in Sec. 5. We use $\star$ after the model name to indicate that the timing also includes the time cost of degeneracy repair.

Type	Model	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{M}\rvert$	Q_{avg}	θ_{min}	$\bar{\theta}_{min}$	θ_{max}	$\theta_{<30^\circ}\%$	$\theta_{>90^\circ}\%$	$\theta_{>120^\circ}\%$	ξ	d_{RMS}	$d_H(\times 10^{-2})$	$Time(s)$
Uniform	Gear (Fig. 12)	229	77K	0.87	2.42	48.81	157.25	0.43	1.68	0.03	0.28	0.017	0.074	8.5
	Helicopter Cockpit (Fig. 12)	72	18K	0.88	7.05	50.23	153.63	1.45	0.68	0.21	0.27	0.006	0.236	5.2
	Component$\star$ (Fig. 12)	183	42K	0.87	2.55	49.39	168.08	0.29	1.54	0.04	0.34	0.005	0.649	5.9
Adaptive	Car (Fig. 2)	881	93K	0.81	0.15	45.32	178.67	1.28	3.05	0.07	0.41	0.074	0.575	15.7
	Cover (Fig. 3)	130	48K	0.86	0.78	48.22	130.82	0.32	1.85	0.01	0.31	0.037	0.146	5.6
	Suspension Component (Fig. 3)	249	37K	0.85	3.08	47.60	156.53	0.53	2.75	0.02	0.30	0.102	0.512	6.8
	Seat$\star$ (Fig. 12)	257	23K	0.84	0.01	46.92	178.88	0.44	2.76	0.04	0.33	0.029	0.217	10.2
	Mini Servo (Fig. 12)	168	35K	0.86	0.88	48.13	176.04	0.27	2.18	0.03	0.32	0.036	0.232	4.5
	Cellphone$\star$ (Fig. 12)	985	58K	0.83	0.63	45.42	177.81	1.58	3.91	1.04	0.43	0.031	0.177	20.1
	Heart (Fig. 13)	93	26K	0.87	0.01	49.05	177.42	1.62	1.19	0.05	0.32	0.006	0.227	8.2

Table 2: Comparison of graded tetrahedral meshing qualities. The best result of each measurement is marked in **bold** font. $|\mathcal{P}|$ is the number of input patches; $|\mathcal{M}|$ is the number of surface triangles; $|\tau|$ is the number of generated tetrahedra.

Model	Method	$\lvert\mathcal{M}\rvert$	$\lvert\tau\rvert$	$Time(s)$	Triangle Mesh						Tetrahedral Mesh						
					Q_{avg}	$\bar{\theta}_{min}$	$\theta_{<30^\circ}\%$	ξ	d_{RMS}	$d_H(\times 10^{-2})$	ϑ_{min}	ϑ_{max}	$\bar{\vartheta}_{min}$	κ_{min}	$\bar{\kappa}$	$\vartheta_{<10^\circ}\%$	$\vartheta_{<20^\circ}\%$
Receiver (Fig. 14), $\lvert\mathcal{P}\rvert = 112$	NetGen	26K	146K	16.5	0.824	45.52	3.36	0.35	0.084	0.339	**2.17**	173.27	69.58	0.005	0.75	0.11	1.38
	Gmsh	20K	106K	34.8	**0.829**	**46.82**	2.15	0.61	0.017	**0.045**	0.67	177.91	69.65	0	0.73	0.14	1.51
	Ours	26K	142K	**7.6**	0.827	46.28	**1.01**	**0.31**	**0.013**	0.161	1.98	**172.64**	**70.04**	**0.018**	**0.77**	**0.09**	**1.24**
Airplane bracket (Fig. 15), $\lvert\mathcal{P}\rvert = 228$	NetGen	55K	136K	18.3	0.799	44.14	3.1	**0.32**	0.165	0.796	0.07	179.28	**69.68**	0	0.72	0.35	1.71
	Gmsh	63K	×	89.5	0.631	38.14	16.28	0.76	0.128	0.473	×	×	×	×	×	×	×
	Ours	49K	125K	**9.8**	**0.817**	**45.78**	**2.61**	0.34	**0.031**	**0.136**	**0.27**	**178.89**	69.54	0	**0.76**	**0.29**	**1.41**

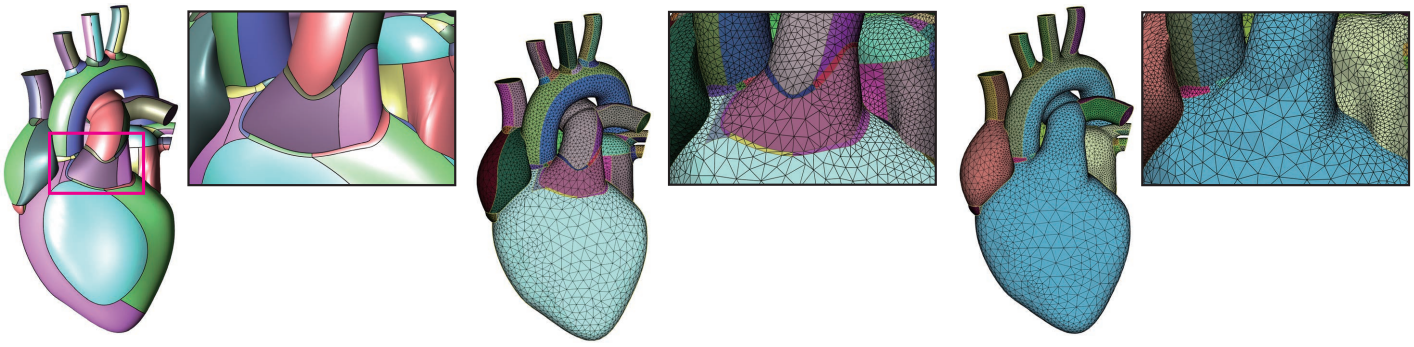


Figure 13: Tessellation results without (middle) and with (right) the operations of patch mesh clustering and constraint simplification.

can be performed very efficiently. Parametric remeshing is the most time consuming step in our framework, because if we process each surface patch one-by-one, the total processing time is linear with the number of input surface patches. Fortunately, we found that our parametric remeshing works on each patch mesh independently. Thus this step can be parallelized. We use the OpenMP library for threaded parametric remeshing.

In the last step, although we have to carefully deal with the sharp features that are common in CAD models, our algorithm is still considerably faster by utilizing real-time remeshing algorithms [12, 13]. Table 1 lists the timings for tessellating each model. In addition, performance can be further improved by disabling the back-to-surface projection without considerable loss of quality.

5.2. Comparisons

Comparison with open-source methods. We now compare our approach with two popular open-source software, NetGen [6] and Gmsh [7], which are both readily available and widely used in the meshing and simulation community. For fair comparison, we tune the parameters of each algorithm to generate surface meshes with a similar number of vertices and triangles. In addition, the sizing function used in all methods is defined based on the curvature. First, we compare the robustness of these three methods. As shown in Fig. 1, NetGen cannot handle the degeneracy caused by tangency. While Gmsh is able to deal with such a case, it easily generates intersecting triangles near very narrow sharp features, as illustrated in Fig. 14. By contrast, our algorithm is more robust in obtaining correct tessellation results by carefully handling tangency degeneracy and performing boundary curve recovery before global remeshing.

Next, we conduct a comparison in terms of tessellation quality. Besides using surface mesh quality metrics, we also evaluate these three methods by generating tetrahedral meshes. Then we compute two criteria for evaluating the quality of a tetrahedral mesh: dihedral angle ϑ, and radius ratio κ. The radius ratio, κ, is defined as $\kappa = \frac{3r_{in}}{r_{cir}}$, where r_{in} and r_{cir} are the inscribed and circumscribed sphere radius of a tetrahedron, respectively. In our approach, we use TetGen [61] to generate tetrahedral meshes from each surface mesh. Figures 14 and 15 present the comparison for tetrahedral meshing. The quality is listed in Table 2, where NetGen runs by default in parallel mode. Comparison results verify that our tessellation technique yields higher quality surface and tetrahedral meshes. Furthermore, NetGen and Gmsh often lead to high-density triangles (marked with green circles) around tiny features, which are undesirable in practical simulation.

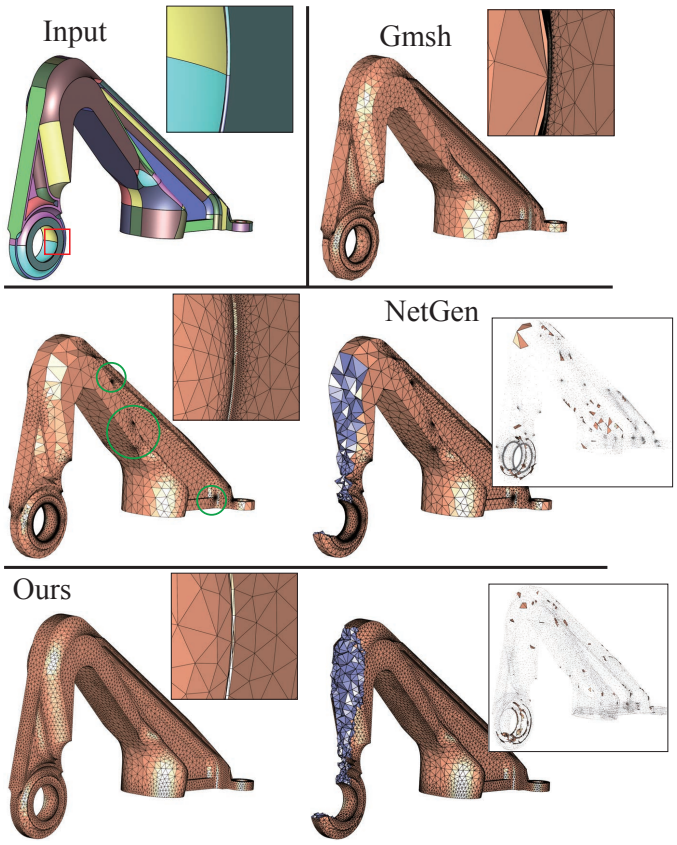


Figure 14: Comparison with NetGen [6] and Gmsh [7] on the "Airplane bracket" dataset. From the output of Gmsh, we failed to generate a tetrahedral mesh. For NetGen and our method, we show the surface mesh, cutaway view of tetrahedral meshing, and livers with dihedral angles smaller than 10°.

Finally, to demonstrate the effectivity of our improved remeshing approach, we compare it with original real-time adaptive remeshing (RAR) method [13]. Fig. 16 shows the remeshing results, in which we use the same initial triangulation and 5 iterations for our method and RAR. We achieve a higher quality mesh, especially near the feature boundaries, due to our operations of angle optimization and valence optimization for feature vertices. Besides, the values of ξ of our method and RAR are 0.35 and 0.48, respectively.

Comparison with commercial meshers. Many commer-

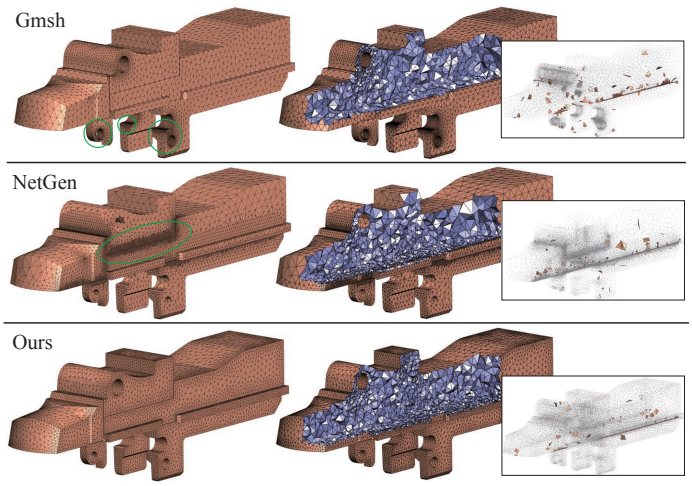


Figure 15: Comparison on "Receiver". Each row shows the surface mesh, cut-away view of tetrahedral meshing, and slivers with dihedral angles smaller than 10°.

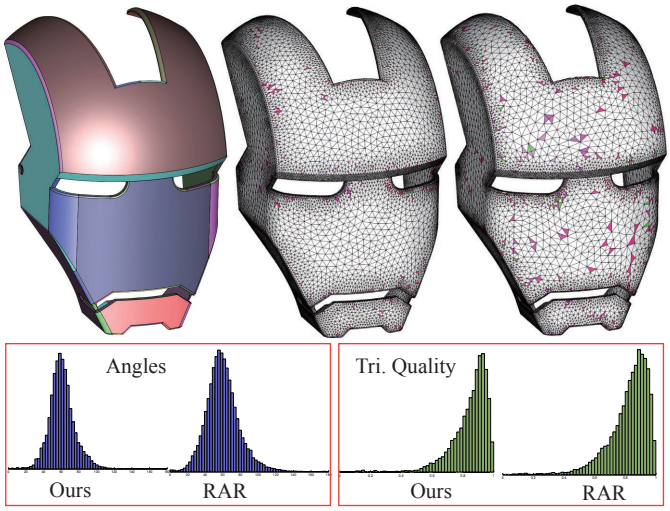


Figure 16: Remeshing comparison with RAR [13] on "Iron man mask". The top row (from left to right) is the input CAD, our remeshing result, and RAR's result. The red color indicates obtuse triangles and the green color shows triangles with $\theta_{min} < 30°$. The bottom row shows histograms of the angle (left) and triangle quality (right) distributions.

cial software, such as Altair HyperMesh[2], Pro/ENGINEER[3], MeshGems[4], are available in the market. These software packages are fully optimized and provide various state-of-the-art surface meshing algorithms to generate high-quality meshes. However, they fail when handling complex models with degeneracies, as shown in Fig. 17. In this example, for the tangency case (labeled by (1) in the figure), MeshGems and HyperMesh generate degenerate triangles. Although Pro/ENGINEER can correctly handle this tangency case, it fails to tessellate the model where the curve proximity is met, as labeled by (2) in the figure. Compared with these packages, our tessellation framework is much simpler, and it outperforms them by successfully tessellating this complex model. Therefore, our algorithm can be used as a drop-in replacement for the task involving CAD tessellation.

[2]http://www.altairhyperworks.com.au/product/HyperMesh
[3]https://www.ptc.com/en/products/cad/pro-engineer
[4]http://www.meshgems.com/

5.3. Limitations

Although the tessellation results of our algorithm satisfy the requirements for most simulation applications, we have not tackled the exact approximation errors. We hope to address this issue by integrating another parameter (approximation tolerance) into the remeshing stage in the future. Next, the SLIM parameterization method we used only grants local injectivity, thus a global overlap could still happen depending on the shape of the patch. Although we did not meet this case in our experiments, it remains problematic. We would like to explore other more advanced parametrization techniques. In addition, since our framework does not involve any geometry repair mechanism, we cannot handle geometry errors existing in original CAD models, e.g., self-intersection. Thus, designers are required to modify the geometry manually or automatically by using other repairing algorithms.

6. Conclusion and Future Work

This paper has proposed a novel method for automated mesh generation for CAD models. Our algorithm relies on two classical remeshing approaches to improve the regularity and angle quality of the mesh. Robustness has been emphasized through the entire meshing process, and by taking care of the degeneracies, unintended construction artifacts, and shape boundary preservation. The experiments on a variety of complex models and the comparison with the state-of-the-art tessellation packages, demonstrate that our approach is simple yet efficient in generating high-quality surface meshes.

In our future work, we would like to extend our approach to a mesh generator that can control anisotropy and directionality via a metric tensor field. We are also interested in exploring more accurate sizing functions rather than the simple curvature to define adaptive mesh with smoothed gradient of element scales. Lastly, although we considered filtering small features, we indeed did not distinguish between important and unimportant features. One good research topic is to recognize such important features (e.g., tight radius fillets, turbulators, and flow diversion devices) and preserve them properly during mesh generation.

Acknowledgements

This work is partially funded by the National Natural Science Foundation of China (61802406, 61872354, 61772523 and 61620106003), the Beijing Natural Science Foundation (4184102), the Open Funding Project of State Key Laboratory of Virtual Reality Technology and Systems of Beihang University (Grant No. BUAA-VR-17KF-06), and the Open Projects Program of National Laboratory of Pattern Recognition (NLPR) (Grant No. 201700020).

[1] B. M. Klingner, B. E. Feldman, N. Chentanez, J. F. O'Brien, Fluid animation with dynamic meshes, ACM Trans. on Graphics (Proc. SIGGRAPH) 25 (3) (2006) 820–825.
[2] P. J. Frey, P.-L. George, Mesh generation: application to finite elements, ISTE, 2007.
[3] P.-L. George, H. Borouchaki, Delaunay triangulation and meshing.
[4] T. J. Baker, Mesh generation: Art or science?, Progress in Aerospace Sciences 41 (1) (2005) 29–63.
[5] Y. Ito, Challenges in unstructured mesh generation for practical and efficient computational fluid dynamics simulations, Computers & Fluids 85 (2013) 47–52.

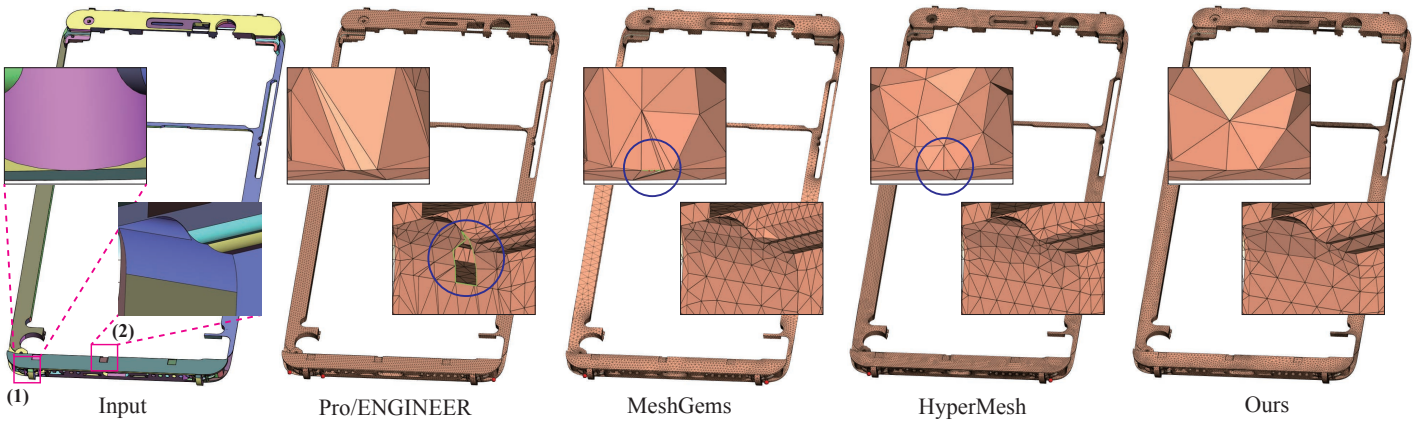


Figure 17: Comparison with some commercial meshers. The green lines represent border edges, while red points represent the vertices of degenerate triangles.

[6] J. Schöberl, Netgen an advancing front 2d/3d-mesh generator based on abstract rules, Computing and visualization in science 1 (1) (1997) 41–52.

[7] C. Geuzaine, J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331,.

[8] M. Attene, A lightweight approach to repairing digitized polygon meshes, The Visual Computer 26 (11) (2010) 1393–1406.

[9] T. Ju, Robust repair of polygonal models, ACM Trans. on Graphics (Proc. SIGGRAPH) 23 (3) (2004) 888–895.

[10] Open Cascade Technology, http://www.opencascade.com.

[11] Q. Du, D. Wang, L. Zhu, On mesh geometry and stiffness matrix conditioning for general finite element spaces, SIAM Journal on Numerical Analysis 47 (2) (2009) 1421–1444.

[12] M. Botsch, L. Kobbelt, A Remeshing Approach to Multiresolution Modeling, in: Eurographics Symposium on Geometry Processing, 2004, pp. 189–196.

[13] M. Dunyach, D. Vanderhaeghe, L. Barthe, M. Botsch, Adaptive remeshing for real-time mesh deformation., in: Eurographics (Short Papers), 2013, pp. 29–32.

[14] S. J. Owen, A survey of unstructured mesh generation technology, in: International Meshing Roundtable, 1998, pp. 239–267.

[15] M. Bern, P. Plassmann, Mesh Generation, Elsevier Science, 2000.

[16] D. S. Lo, Finite element mesh generation, CRC Press, 2014.

[17] K. Shimada, Current issues and trends in meshing and geometric processing for computational engineering analyses, ASME Journal of Computing and Information Science in Engineering 11 (2) (2011) 021008.

[18] J. C. Caendish, D. A. Field, W. H. Frey, An apporach to automatic three-dimensional finite element mesh generation, International journal for numerical methods in engineering 21 (2) (1985) 329–347.

[19] B. Lévy, Y. Liu, L_p centroidal Voronoi tesselation and its applications, ACM Trans. on Graphics (Proc. SIGGRAPH) 29 (4) (2010) 119:1–11.

[20] B. Lévy, N. Bonneel, Variational anisotropic surface meshing with Voronoi parallel linear enumeration, in: Proceedings of the 21st International Meshing Roundtable, 2012, pp. 349–366.

[21] K. Nakahashi, D. Sharov, Direct surface triangulation using the advancing front method, in: 12th Computational Fluid Dynamics Conference, 1995, p. 1686.

[22] T. Lan, S. Lo, Finite element mesh generation over analytical curved surfaces, Computers & Structures 59 (2) (1996) 301–309.

[23] J. R. Tristano, S. J. Owen, S. A. Canann, Advancing front surface mesh generation in parametric space using a riemannian surface definition., in: International Meshing Roundtable, 1998, pp. 429–445.

[24] M. A. Yerry, M. S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, International Journal for Numerical Methods in Engineering 20 (11) (1984) 1965–1990.

[25] A. A. Shostko, R. Löhner, W. C. Sandberg, Surface triangulation over intersecting geometries, International journal for numerical methods in engineering 44 (9) (1999) 1359–1376.

[26] Y. Liu, S. Lo, Z.-Q. Guan, H.-W. Zhang, Boundary recovery for 3d delaunay triangulation, Finite Elements in Analysis and Design 84 (2014) 32–43.

[27] J. R. Shewchuk, Delaunay refinement mesh generation, Tech. rep., DTIC Document (1997).

[28] X. Sheng, B. E. Hirsch, Triangulation of trimmed surfaces in parametric space, Computer-Aided Design 24 (8) (1992) 437–444.

[29] J.-C. Cuillière, An adaptive method for the automatic triangulation of 3d parametric surfaces, Computer-Aided Design 30 (2) (1998) 139–149.

[30] H. Borouchaki, P. Laug, P.-L. George, Parametric surface meshing using a combined advancing-front generalized delaunay approach, International Journal for Numerical Methods in Engineering 49 (1-2) (2000) 233–259.

[31] Y. Zheng, N. P. Weatherill, O. Hassan, Topology abstraction of surface models for three-dimensional grid generation, Engineering with Computers 17 (1) (2001) 28–38.

[32] R. J. Cripps, S. Parwana, A robust efficient tracing scheme for triangulating trimmed parametric surfaces, Computer-Aided Design 43 (1) (2011) 12–20.

[33] P. Laug, Some aspects of parametric surface meshing, Finite Elements in Analysis and Design 46 (1) (2010) 216–226.

[34] E. Béchet, J.-C. Cuilliere, F. Trochu, Generation of a finite element mesh from stereolithography (stl) files, Computer-Aided Design 34 (1) (2002) 1–17.

[35] D. Wang, O. Hassan, K. Morgan, N. Weatherill, Eqsm: An efficient high quality surface grid generation method based on remeshing, Computer Methods in Applied Mechanics and Engineering 195 (41) (2006) 5621–5633.

[36] E. Marchandise, J.-F. Remacle, C. Geuzaine, Quality surface meshing using discrete parametrizations, in: Proceedings of the 20th International Meshing Roundtable, Springer, 2011, pp. 21–39.

[37] E. Marchandise, J. Remacle, C. Geuzaine, Optimal parametrizations for surface remeshing, Engineering with Computers 30 (3) (2014) 383–402.

[38] R. Aubry, B. K. Karamete, E. L. Mestreau, S. Dey, A three-dimensional parametric mesher with surface boundary-layer capability, Journal of Computational Physics 270 (2014) 161–181.

[39] R. Aubry, S. Dey, E. L. Mestreau, B. K. Karamete, D. Gayman, A robust conforming nurbs tessellation for industrial applications based on a mesh generation approach, Computer-Aided Design 63 (2015) 26–38.

[40] T. Ju, Fixing geometric errors on polygonal models: a survey, Journal of Computer Science and Technology 24 (1) (2009) 19–29.

[41] M. Campen, M. Attene, L. Kobbelt, A practical guide to polygon mesh repairing., in: Eurographics (Tutorials), 2012.

[42] M. Attene, M. Campen, L. Kobbelt, Polygon mesh repairing: An application perspective, ACM Computing Surveys (CSUR) 45 (2) (2013) 15.

[43] G. Turk, M. Levoy, Zippered polygon meshes from range images, in: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94, ACM, 1994, pp. 311–318.

[44] P. Borodin, M. Novotni, R. Klein, Progressive gap closing for meshrepairing, in: Advances in Modelling, Animation and Rendering, Springer, 2002, pp. 201–213.

[45] P. Liepa, Filling holes in meshes, in: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, 2003, pp. 200–205.

[46] M. Campen, L. Kobbelt, Exact and robust (self-) intersections for polygonal meshes, in: Computer Graphics Forum, Vol. 29, Wiley Online Library, 2010, pp. 397–406.

[47] C. Andújar, P. Brunet, D. Ayala, Topology-reducing surface simplification

using a discrete solid representation, ACM Trans. on Graphics 21 (2) (2002) 88–105.
[48] S. Bischoff, D. Pavic, L. Kobbelt, Automatic restoration of polygon models, ACM Trans. on Graphics 24 (4) (2005) 1332–1352.
[49] Q.-Y. Zhou, T. Ju, S.-M. Hu, Topology repair of solid models using skeletons, IEEE Trans. on Vis. and Comp. Graphics 13 (4).
[50] F. S. Nooruddin, G. Turk, Simplification and repair of polygonal models using volumetric techniques, IEEE Trans. on Vis. and Comp. Graphics 9 (2) (2003) 191–205.
[51] S. Bischoff, L. Kobbelt, Structure preserving CAD model repair, in: Computer Graphics Forum (Proc. EUROGRAPHICS), Vol. 24, Wiley Online Library, 2005, pp. 527–536.
[52] R. Haimes, Capri: Computational analysis programming interface. a solid modeling based infra-structure for engineering analysis and design. revision 2.0, Massachusetts Inst. of Technology, Cambridge, MA.
[53] F. Dassi, A. Mola, H. Si, Curvature-adapted remeshing of cad surfaces, Engineering with Computers.
[54] A. Loseille, Unstructured mesh generation and adaptation, Handbook of Numerical Analysis 18 (2017) 263–302.
[55] M. Rabinovich, R. Poranne, D. Panozzo, O. Sorkine-Hornung, Scalable locally injective mappings, ACM Trans. on Graphics 36 (2) (2017) 16.
[56] CGAL, Computational Geometry Algorithms Library, http://www.cgal.org.
[57] P. Frey, H. Borouchaki, Surface mesh evaluation, in: 6th Intl. Meshing Roundtable, 1997, pp. 363–374.
[58] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, Computer Graphics Forum 17 (2) (1998) 167–174.
[59] J. R. Shewchuk, Mesh generation for domains with small angles, in: Proceedings of the sixteenth annual symposium on Computational geometry, ACM, 2000, pp. 1–10.
[60] K. Inoue, T. Itoh, A. Yamada, T. Furuhata, K. Shimada, Face clustering of a large-scale cad model for surface mesh generation, Computer-Aided Design 33 (3) (2001) 251–261.
[61] H. Si, Tetgen, a Delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Softw. 41 (2) (2015) 11:1–11:36.