Isotropic Surface Remeshing without Large and Small Angles

Yiqun Wang, Dong-Ming Yan, Xiaohan Liu, Chengcheng Tang, Jianwei Guo, Xiaopeng Zhang, Peter Wonka

Abstract—We introduce a novel algorithm for isotropic surface remeshing which progressively eliminates obtuse triangles and improves small angles. The main novelty of the proposed approach is a simple vertex insertion scheme that facilitates the removal of large angles, and a vertex removal operation that improves the distribution of small angles. In combination with other standard local mesh operators, e.g., connectivity optimization and local tangential smoothing, our algorithm is able to remesh efficiently a low-quality mesh surface. Our approach can be applied directly or used as a post-processing step following other remeshing approaches. Our method has a similar computational efficiency to the fastest approach available, i.e., real-time adaptive remeshing [1]. In comparison with state-of-the-art approaches, our method consistently generates better results based on evaluations using different metrics.

Index Terms—Isotropic remeshing, non-obtuse remeshing, local operation, triangle quality.

1 Introduction

Triangle meshes are omnipresent representations of three-dimensional (3D) data in scientific and engineering applications from geometry modeling to physical simulation, due to its efficiency, simplicity, and flexibility. With the recent progress in acquisition hardware and reconstruction techniques, acquiring point clouds with fine geometric details and creating highly complicated raw meshes with a large amount of points connected by badly shaped triangles (i.e., triangles with angles that are too small or too large) are considerably easy. However, for memory consumption reduction, computational efficiency and accuracy, reducing the number of points and simultaneously improving the mesh quality are often necessary. The goal of recent developments in remeshing is creating high-quality triangle meshes that could be used by practical applications based on raw data [2].

Three desired qualities are involved when discussing a good remeshing algorithm: *fidelity*, *simplicity*, and *element quality* [3]. As the premise of surface representation, the *fidelity*, measured by various distance metrics between the triangle mesh and reference geometry, must be able to faithfully represent a geometric object. Moreover, for a memory-efficient representation and computation, the number of vertices and the complexity of mesh connectivity should be reduced, which is encoded as the requirement of mesh *simplicity*. Finally, the applications related to solving partial differential equations often require triangle meshes with well-shaped triangles, namely, good *element quality*, to construct stable basis functions or enable robust numerical integrators [4]. The most important criterion of the element quality is the minimal/maximal angles. For example, for a wide variety of applications such as geodesic distance computation, acute meshes (i.e., meshes with only acute triangles) are often desirable [5], [6], [7]. However, these goals are often in conflict with each other. For example, reducing the complexity will lower the fidelity, and vice versa. Numerous remeshing algorithms have been proposed to achieve a balance among the three desired qualities of triangle meshes, with the additional concern of the computational speed.

Existing algorithms always consist of two main parts: resampling geometry and rebuilding connectivity. Local mesh operations can be used for these parts. The geometry can be updated by *vertex relocation*, and the connectivity can be updated through *edge flipping*, *edge collapsing*, and *edge splitting*. For example, mesh simplification attempts to reduce the complexity of meshes by iteratively applying edge collapsing operations [12]. This method tends to preserve the fidelity as much as possible, but without considering the angle quality. Another example is the Delaunay mesh construction, which aims at converting input meshes into new meshes that satisfy the so-called Delaunay property without altering the input geometry [13]. A Delaunay mesh can be constructed efficiently using only edge splitting and edge flipping. However, this

- *Y. Wang, D.-M. Yan, X. Liu, J. Guo, X. Zhang are with the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: yiqun.wang@nlpr.ia.ac.cn, yandongming@gmail.com, liuxiaohan2017@ia.ac.cn, jianwei.guo@nlpr.ia.ac.cn, xpzhang@ia.ac.cn*
- *C. Tang is with Stanford University. E-mail: tangchengcheng717@gmail.com*
- *P. Wonka is with KAUST, Thuwal 23955-6900, Saudi Arabia. E-mail: pwonka@gmail.com.*

Fig. 1: Comparisons of the remeshing results of the vase-lion model using different approaches. The top row shows the input mesh (leftmost) and the remeshing results of previous approaches, including RAR [1], MAI [8], SPP [9], IFM [10], NOB [11] (from left to right), using approximately 6.5 k vertices. The bottom row presents the results of our method (leftmost) and those obtained by using the corresponding meshes in the top row as initializations. Our algorithm is able to directly remesh the input surface, and can also be used as a post-process to improve the mesh quality of previous approaches. The light red color indicates a triangle whose maximal angle is greater than the desired upper bound, β_{max} (90° in this example), and the light blue color shows a triangle whose minimal angle is smaller than the lower bound, β_{min} (30° in this example). Table 1 presents the detailed quality statistics.

method does not improve the angle quality. Note that our results can be seen as a special type of Delaunay mesh with only acute triangles. To achieve an even distribution of mesh vertices, numerous methods focus on minimizing surrogate energy functions such as the centroidal Voronoi tessellation (CVT) [14] and optimal Delaunay triangulation (ODT) [15] . The underlying operations involved in CVT and ODT computation can be seen as a combination of vertex relocation and edge flipping. Although they produce good results in practice, these energy functions are not directly related to the mesh quality criteria previously discussed and our method can improve upon these techniques. The local mesh operations can be used directly for dynamic surface tracking [16] and real-time remeshing [17], [1], while guaranteeing the minimal angle bound [8]. However, to the best of our knowledge, no existing method can explicitly control both the minimal/maximal angle bounds at the same time for remeshing.

The main motivation behind our work is designing powerful tools to optimize important mesh quality criteria directly, e.g., the minimal and maximal angle. The newly designed algorithm should have a similar performance to the real-time remeshing approach [1], and allow users to explicitly control the element quality. Additionally, we provide optional control to allow users to gain balance between the fidelity and computation time. Similar to previous works, we use local remeshing operations, e.g., edge collapsing for mesh simplification, edge splitting for subdivision, and edge flipping for valence/angle optimization. The main novelty of our approach is that it explores new combinations of local remeshing operations and proposes new trigger conditions that determine how the local remeshing operations are combined. Moreover, our method can be applied directly to the input mesh, or be used as a post-process of other existing techniques. Figure 1 shows an example of remeshing a complex input geometry. The key contributions of this approach include the following aspects:

- We propose new combinations of local operations for angle improvement.
- Our method produces meshes with superior angle quality compared with current state-of-the-art methods. In addition, we can also improve the general triangulation quality reflected through multiple statistics.
- We provide optional control to trade off computation speed with feature-sensitivity (fidelity). Moreover we provide optional control for users to switch between fast remeshing and feature-

sensitive remeshing (high fidelity).

2 Related work

Various types of remeshing techniques exist depending on specific applications. In this section, we focus our discussion on the works that are most relevant to ours. Specifically, we emphasize the recent approaches for non-obtuse remeshing. A further systematic introduction to remeshing can be found in the survey papers [2], [18] and textbooks [19], [20].

The non-obtuse (or acute) triangulation is first studied on the 2D plane [21], [22], [23], [24], [25]. Unfortunately, none of these approaches can be directly generalized to surfaces in 3D. To the best of our knowledge, Li and Zhang [26] are the first ones to propose a non-obtuse remeshing algorithm on the basis of Laplacian smoothing and mesh simplification. Although they are able to generate non-obtuse meshes for smooth surfaces, they failed to eliminate small angles at the same time.

Recent advances show that mesh optimization and smoothing techniques can drastically improve the mesh quality. These approaches include CVT [27], [14], centroidal Delaunay triangulation [28], ODT [15], blue-noise sampling [29], and discrete optimization (e.g., edge splitting, edge collapsing, and local smoothing). However, not all of them can produce non-obtuse remeshing.

CVT-based approaches have been proven to generate meshes with the highest quality. Considering a sampling domain and the number of sample points, the CVT energy is minimized by using either classic Lloyd iterations [30] or Newton-like methods [14]. The final triangulation is extracted as the dual of the optimized Voronoi diagram. The main differences among different approaches are the methods used to approximate the Voronoi diagram on surfaces. Earlier works are pioneered by Alliez and coauthors [31], [3], [32]. They first parameterize the input mesh on a 2D parameter domain, and apply CVT to this 2D domain. Then, the points and triangulation are lifted back to the original surface to obtain the remesh. Generally, the parameterization-based approaches are able to efficiently generate high-quality meshes. However, they suffer from the issues of robustness and distortion introduced during the parameterization.

Recent works of Yan et al. [33], [34], [11] and Du et al. [35] directly compute CVT on mesh surfaces, where the Voronoi diagram on surfaces is approximated by the restricted Voronoi diagram (RVD) [36]. The RVD-CVT-based approaches can generate high-quality meshes. Additionally, the most recent work [11] is able to produce non-obtuse meshes for smooth surfaces. However, the approach fails for inputs with sharp features or highly varying density functions, as discussed in the paper. Sun et al. [37] propose to reduce the number of obtuse triangles for anisotropic remeshing, where a hexagonal metric is used to replace the Euclidean metric. However, this approach also cannot remove all the obtuse triangles.

Another type of CVT-based remeshing is to use the geodesic distance for computation [38], [39], [40], [41], [42], [43]. Although geodesic-based methods can generate meshes with the same quality as that of RVD-based methods, they involve frequent geodesic path computations, which drastically slow down the remeshing process.

Blue-noise sampling has also been introduced for surface remeshing, including capacity-constrained blue-noise sampling [44], maximal Poisson-disk sampling (MPS) [29], [45], [46], [47], farthest point optimization (FPO) [48], and the simple push-pull (SPP) [9] approaches. The main goal of this category of approaches is to produce point samples with the so-called "blue-noise" property. The mesh quality is then improved under certain constraints while keeping the randomness of the point distribution.

Discrete optimization improves the mesh quality by repeatedly applying local operators on the input geometry, e.g., edge splitting, edge flipping, edge collapsing, and vertex relocation. Our approach falls into this category. Botsch and Kobbelt [17] propose a simple and efficient framework for uniform remeshing, which contains four key components: splitting long edges, collapsing short edges, valence optimization by edge flipping, and tangential smoothing. This framework can generate uniform remeshing in real time, which has been used for multi-resolution shape modeling. Later, Dunyach et al. [1] generalized the framework to real-time adaptive remeshing (RAR) and applied it to mesh deformation applications. Both [17] and [1] are able to remesh input surfaces in real time, but without considering any bounds of the output angles. To address this problem, Hu et al. [8] improved the smallest angle of the mesh by repetitively applying edge collapsing, vertex relocation, and edge splitting while bounding the approximation errors and implicitly preserving features. Although this algorithm could increase the smallest angle to a user-specified threshold (up to 40°), it introduces numerous obtuse triangles at the same time, as shown in Sec. 5. Liu et al. [13] proposed an efficient method to construct Delaunay meshes by inserting fewer additional points than the previous approach [49]. Only simple operations, such as edge splitting and edge flipping, are involved in the computation. Although this algorithm does not alter the geometry of the input mesh, it does not improve the angle quality either. In our work, we simultaneously improve minimal angles and decrease maximal angles to produce meshes without too large and too small angles up to the user specified angle bounds. As a method in the category of discrete optimization, our approach repeatedly applies local operations but with newly designed trigger conditions.

To improve the regularity of the vertices (e.g., valence-6 vertices are considerably preferred in the interior of a mesh), multiple other approaches can be used, such as explicit remeshing [50], valence editing [51], and global parameterization [52], [53], [10]. These approaches can generate meshes with highly regular patterns. However, the triangles are always distorted and are difficult to use for adaptive remeshing due to strong constraints enforced by the regularity.

3 Overview

In this work, we present a simple yet effective approach for isotropic remeshing by repeatedly eliminating large angles and improving small angles. These goals are achieved by designing two novel high-level operations, i.e., large angle removal and small angle improvement, which are combinations of the standard local mesh operations, i.e., edge splitting, edge collapsing, edge flipping, and vertex relocation (cf. Fig. 2). The large angle removal operation is achieved by point insertion (edge splitting), whereas the small angle improvement operation is attained by point deletion (edge collapsing). By combining both operations, we are able to keep the number of vertices in the output mesh equal to a user-specified value.

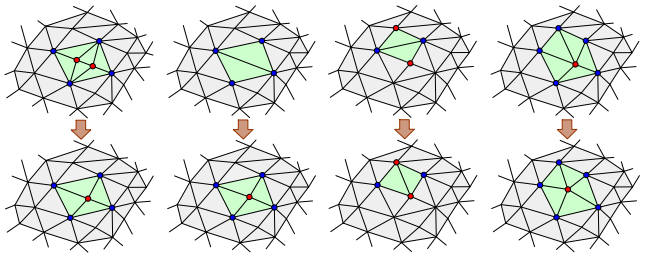


Fig. 2: Illustration of the standard local mesh operations, i.e., edge collapsing, edge splitting, edge flipping and vertex relocation (from left to right).

The proposed framework consists of two main components, i.e., initialization and optimization. First, the output mesh is duplicated from the input mesh, and is initialized by either inserting or removing vertices to reach a user-desired target number of vertices (Sec. 4.1).

Next, we analyze the initial output mesh and label the triangles whose angles are not bounded by the user specified minimal and maximal values, e.g., the default bounds used in our experiments are $[35°, 86°]$. Then, we repeatedly optimize the triangle quality by removing large angles and by improving the small angles (Sec. 4.2). Fig. 3 illustrates the remeshing process of our framework using the botijo model as an example. Moreover, we provide optional extensions that can improve the fidelity of the output mesh in Sec. 4.3. We explain the details of each step in the next section.

4 Our Approach

The inputs to our method include a two-manifold triangle mesh, $\mathcal{M}$; the specified minimal and maximal angle bounds, β_{min} and β_{max}, respectively; and the desired number of target vertices in the output mesh, n_t (optional). We assume that the feature curves of the input mesh to be preserved are pre-specified by the user or provided as input, if any.

4.1 Initialization

The goal of this step is to generate an initial mesh with the user-specified number of target vertices, which matches a sizing function defined on $\mathcal{M}$.

In the context of uniform remeshing, we introduce a constant target edge length, L, computed on the basis of the total surface area, $|\mathcal{M}|$, of the input mesh. In the ideal case where the triangles are identical equilateral triangles, the average triangle area roughly equals to $\frac{|\mathcal{M}|}{2N}$. Accordingly, the reference edge length can be deduced as $L = \frac{2}{\sqrt[4]{3}}\sqrt{\frac{|\mathcal{M}|}{2N}}$ by using the relationship between the area and edge length of equilateral triangles. Thus, the sizing function for uniform remeshing is defined as $\rho(\mathbf{x}) = L$.

For adaptive remeshing, we compute a smoothly varying sizing function, $\rho(\mathbf{x}_i)$, on every vertex by using an approximation error, ϵ and the curvature radius, r_i, on each vertex of the input mesh [1]. We set $\rho(\mathbf{x}_i) = \sqrt{6\,\epsilon\, r_i - 3\epsilon^2}$, computed by the relationship between r_i and the edge length $\rho(\mathbf{x}_i)$, which means that the specified length can approximate the mesh surface within the approximation error. In addition, we introduce two parameters, namely, h_{min} and h_{max}, which are the minimal and maximal edge length bounds of the sizing function $\rho(\mathbf{x}_i)$. In the work of Dunyach et al. [1], these parameters (i.e., ϵ, h_{min}, and h_{max}), which are difficult to adjust, have to be set manually by the user. Here, we introduce a simple method to compute the recommended parameters. Considering the input mesh, we first compute the curvature radius, r_i, on each vertex and the total surface area $|\mathcal{M}|$. Subsequently, we establish the relationship among $|\mathcal{M}|$, r_i, and ϵ in the ideal case where the triangles are identical equilateral triangles, i.e., $S = \sum_i \left\{\frac{\sqrt{3}}{4}\rho(\mathbf{x}_i)^2\right\} = \sum_i \left\{\frac{\sqrt{3}}{4}(6\epsilon r_i - 3\epsilon^2)\right\}$. $\rho(\mathbf{x}_i)$, as is a function of r_i and ϵ, can be seen as a quadratic equation of ϵ, which can be solved efficiently. Then, we can deduce h_{min} and h_{max} from the maximum and minimum edge length $\rho(\mathbf{x}_i)$.

Once the sizing function has been computed, we iteratively perform edge splitting for long edges ($\|\mathbf{x}_a, \mathbf{x}_b\| > \frac{4}{3}\, min(\rho(\mathbf{x}_a), \rho(\mathbf{x}_b))$) or edge collapsing for short edges ($\|\mathbf{x}_a, \mathbf{x}_b\| < \frac{4}{5}\, min(\rho(\mathbf{x}_a), \rho(\mathbf{x}_b))$) in order to generate an initial output mesh with n_t vertices. We also remark that our approach can be used as a post-processing step of numerous previous remeshing approaches, by using their outputs as our

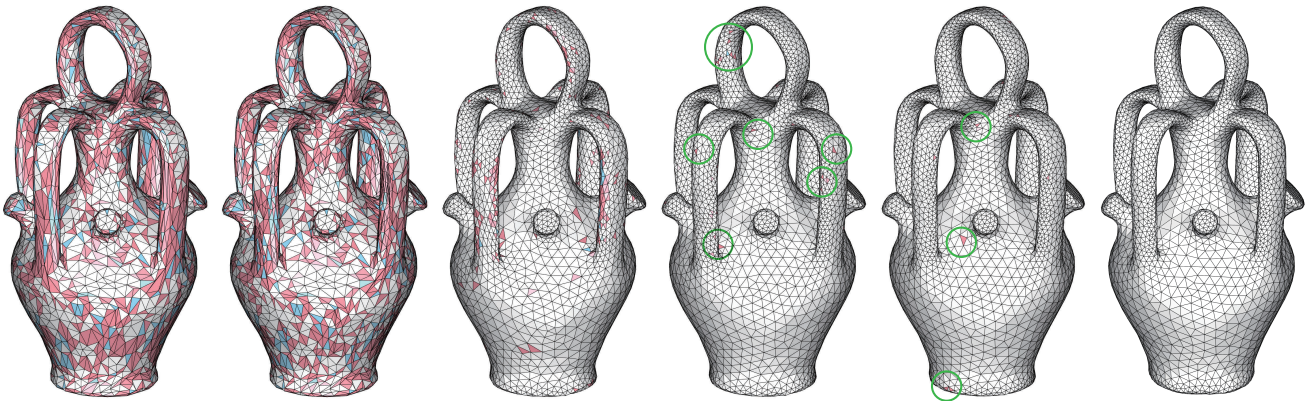


Fig. 3: Remeshing processing of our approach. From left to right are: the input meshes (3353 and 2991 triangles with large and small angles, respectively), initialized after splitting long edges (3368 and 3006, respectively), and results of one iteration (758 and 201, respectively), two iterations (171 and 13, respectively), three iterations (44 and 2, respectively), and 15 iterations (0 and 0, respectively). Both large and small angles are eliminated progressively.

initial results, and keeping the number of vertices unchanged.

4.2 Optimization

The goal of this step is to improve the mesh quality by jointly eliminating large angles and improving small angles. All the triangles with small or large angles outside the desired bounds $[\beta_{min}, \beta_{max}]$ are processed. The default angle bounds are $[35°, 86°]$ if not explicitly specified. Our optimization is based on standard local mesh operators, i.e., edge splitting, edge collapsing, edge flipping, and vertex smoothing. However, we propose several novel trigger conditions and combinations of these operators, which can be efficiently used to achieve our goal. Alg. 1 presents the pseudo code of the proposed optimization procedure.

Algorithm 1: Optimization

input : Initialized mesh $\mathcal{M}_i$, parameter k, β_{min}, β_{max}
output: Output mesh $\mathcal{M}'$ whose angles $\theta \in [\beta_{min}, \beta_{max}]$

```
repeat
    LargeAngleRemoval(k);
    ValenceOptimization();
    VertexSmoothing();
    SmallAngleImprovement(k);
    ValenceOptimization();
    VertexSmoothing();
until ∀θ, θ ∈ [β_min, β_max];
```

Alg. 1 shows four main components in this step: *large angle elimination*, *small angle improvement*, *valance optimization*, and *vertex smoothing*. In each iteration, we first scan all the triangles in the output mesh $\mathcal{M}'$ and store those triangles whose angles are outside the angle bounds in two separate lists L_l and L_s (one for triangles with angles larger than β_{max}, and the other for triangles with angles smaller than β_{min}). Then, we use an additional internal parameter k to control the number of triangles to be processed in each routine.

To eliminate the large angles, we apply vertex insertion for the first k triangles with large angles following the procedure described in Sec. 4.2.1. Then, we further locally apply edge flipping for valence optimization in accordance with Sec. 4.2.2. Finally, we apply local smoothing to the neighborhoods affected by the previous operations. To improve the small angles, we apply the same procedure as the large angle elimination. The major difference is that, the core component, instead of applying vertex insertion, is the vertex removal operation mentioned in Sec. 4.2.4 with edge collapsing. Note that each large angle elimination operation inserts one new vertex, whereas each small angle improvement operation deletes one vertex. By applying both operations at the same time, we can keep the number of the vertices unchanged during the optimization procedure.

4.2.1 *Large angle removal*

Vertex insertion is the essential operation for removing large angles. Starting with the list L_l of triangles with angles larger than β_{max}, we apply k times vertex insertion for the the first k triangles in L_l. We use two different strategies for the smooth and feature cases (for triangles that are incident to one or more feature edges). Here, we make the assumption that all the triangles in L_l are isolated for easy explanation.

Smooth case. If the current triangle to be processed has no incident feature edge, then it is treated as the smooth case. A majority of the triangles in L_l fall into this category. For such case, we first merge the triangle with the neighboring triangle sharing the longest edge to form a quadrilateral Q. Then we split the common edge shared by two triangles on the midpoint. Next, we flip one of the four edges of the quad Q. We select the edge after flipping whose affected angles are optimal, i.e., the root mean square of the summed difference between affected angles minus the optimal value (60°). The position of the newly inserted vertex is further optimized by local tangential smoothing. Figs. 4(a) and 4(b) illustrate this process.

Feature case. For input meshes with tagged feature information, we have to exert additional effort in this

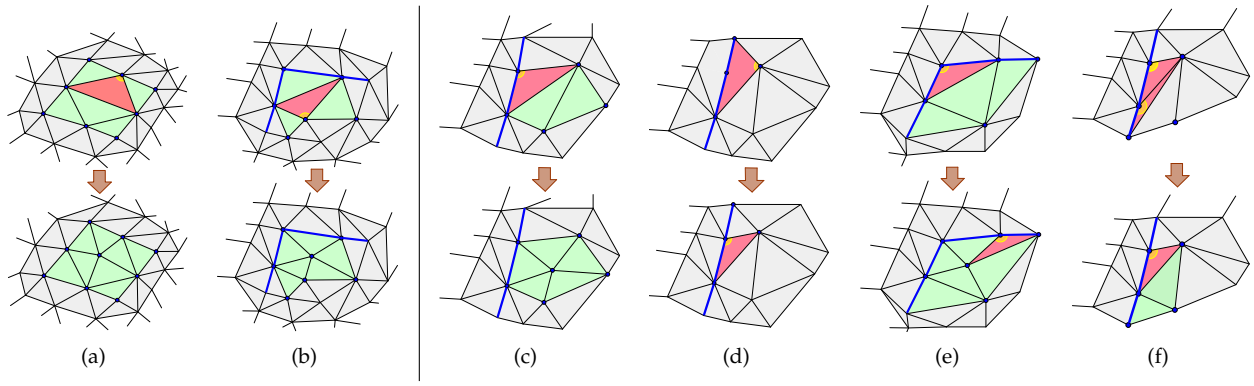


Fig. 4: Vertex insertion strategy for the smooth (a,b) and feature (c∼f) cases. The feature edges are drawn in blue. (a) After merging the triangle with large angles with a neighbor along the longest edge to create a quadrilateral and subsequently a pentagon, a vertex is inserted. This operation is equivalent to edge splitting followed by edge flipping. (b) This procedure also works for the smooth case near the features. (c) The basic case where a single short edge is a feature is resolved by pentagon pointer insertion, topologically equivalent to edge splitting followed by another edge flipping. (d) The case where the long edge is a feature is reduced to case (c) by edge splitting. (e) The case where both the two short edges are features is converted to case (c) by pentagon insertion. (f) A special case that can be resolved directly by edge flipping.

step. Figs. 4(c) to 4(f) present the typical cases when an edge of the triangle to be processed lies on the boundary or is labeled as a feature edge. Specifically, Fig. 4(c) shows that the most common case we encountered is that a single short edge is a feature edge (the edge opposite to the maximal angle is a non-feature edge). This case can be handled by the method for smooth case discussed previously. The other cases could also be easily deduced to this basic case. For example, Fig. 4(d) shows that if the longest edge lies on the boundary or the feature, we apply a local modification by equally splitting the edge at the mid-point, and then proceed as the basic case in Fig. 4(c). Similar procedures are applied for the case when two short edges are features, as shown in Fig. 4(e). We also show another case that can be resolved directly by edge flipping rather than edge splitting in Fig. 4(f). In these situations, several adjacent triangles with large angles on the feature edges could be handled together.

4.2.2 Valence optimization

Once the new vertices are inserted, we further improve the regularity of M' by valence optimization [17], [1], which can be efficiently achieved by applying a serial of edge flipping. The goal of this step is to minimize the square sum of the difference between the valance of each vertex and its corresponding optimal valance. Note that the optimal valences for the inner and boundary vertices are 6 and 4, respectively.

4.2.3 Vertex Smoothing

Next, we apply several iterations ($3 \sim 5$ in our implementation) of tangential Laplacian vertex smoothing to improve the angle quality of M'. We use the following formula for vertex smoothing:

$$c_i = \frac{\sum\limits_{j \in N(v_i)} w_j p_j}{\sum\limits_{j \in N(v_i)} w_j}, \qquad p_i \leftarrow c_i - n_i n_i^T (c_i - p_i), \quad (1)$$

where v_i is the current vertex to be smoothed, n_i denotes the unit normal vector of v_i, $N(v_i)$ represents the one-ring neighboring triangle of v_i, and w_j and p_j indicate the weight and centroid of the j-th neighboring triangle, respectively. This step is necessary as the previous angle optimization steps have modified the geometry, whereas the valence optimization step has changed the connectivity of the output mesh. If the number of affected triangles is considerably less than the total number of triangles, then we only apply smoothing in a local neighborhood of the affected regions, e.g., two or three rings of the affected vertices. Otherwise, we apply global smoothing for the whole output mesh. After each smoothing iteration, we project the vertices in M' to the original input surface M to maintain a high approximation fidelity.

4.2.4 Small angle improvement

After the large angle elimination, we use edge collapsing for small angle improvement (vertex removal). First, we collect all the triangles with small angles less than β_{min} in a list L_s. Then we traverse the first k triangles in L_s. For each visited triangle $t_s \in L_s$, we collapse the edge (if the collapse is legal) opposite to the smallest angle of t_s. Then local smoothing is applied to further improve the angle quality of the affected regions. In practice, if L_s is empty or less than k, then we can temporarily increase the lower bound β_{min} to guarantee that k triangles are collected in L_s. Note that in the large angle removal step, k new vertices are inserted. In this step, k vertices are removed. The number of vertices in the output mesh remains unchanged. Moreover, the internal parameter k is important to the performance of the proposed algorithm. We shall discuss the influence of k in Sec. 5.

4.3 Error-aware extensions

The previously described algorithm is efficient, because only standard local operations are involved

for computation. However, as a consequence of high performance, the fidelity (especially in highly detailed regions) of the output mesh might be sacrificed due to over smoothing. To balance between the fidelity and efficiency, we propose the modified local operations that can reduce approximation errors when applied. Note that these modified operations are optional.

The approximation error is introduced by operations that modify the geometry of the input mesh. As edge splitting does not affect geometric fidelity, we only focus on vertex relocation, edge collapsing, and edge flipping to reduce approximation errors.

Error-aware vertex smoothing. Recall that, in tangential smoothing (Eqn. 1), the update vector u_i of vertex v_i is in the vertex normal plane. This formulation works well in smooth regions but has problems in regions with rich details when the number of vertices is not adequate. Fig. 5 presents a 2D illustration of this issue. Inspired by the feature sensitive term introduced in Lp centroidal Voronoi tessellation (LpCVT) [54], we slightly modify the magnitude of $|u_i|$ to alleviate the error caused by tangential smoothing. We first project the updated vector u_i to every adjacent triangle, and then project back to the vertex normal plane. The smoothing distance $|u_i|$ can be resized by

$$
\begin{aligned}
u_i &\leftarrow \prod_{j \in N(v_i)} \left\{ \left(n_i^u [n_i^u]^t \right) \mathrm{N}_i^v \mathrm{N}_i^{f_j} \right\}^d u_i, \\
\mathrm{N}_i^v &= \left(\mathrm{I}_{3\times 3} - n_i^v [n_i^v]^t \right), \\
\mathrm{N}_i^{f_j} &= \left(\mathrm{I}_{3\times 3} - n_i^{f_j} [n_i^{f_j}]^t \right),
\end{aligned}
$$

where $n_i^{f_j}$ denotes the adjacent triangle normal, n_i^v indicates the vertex normal weighted by the circumjacent facet normal, n_i^u is the unit vector of u_i, and d indicates the number of projections controlling the degree of feature preservation. If $d \to \infty$, then $u_i \to \mathbf{0}$. An infinite number of projections mean that the vector is simultaneously fixed on all faces, so the value must be zero. In highly detailed regions (e.g., the hair of the vase-lion model), the smoothing distance $|u_i|$ becomes smaller. The smoothing distance can even become equal to zero on sharp features, whereas in smooth situations, $|u_i|$ is not considerable. During the projection phase, we construct the vector u_i formed by the projection and steepest points on the reference mesh vertices or edges near the projection point. The place where the original geometry changes drastically will be maintained adaptively in the same way. With this modified smoothing operation, the approximation error could be reduced in some extent, compared with standard tangential smoothing.

Error-aware edge collapsing. For the edge collapsing step, we carefully designed a simple vertex intensity to measure the sharpness of a vertex.

$$
\Phi_i = \min_{j \in N(v_i)} \left\{ \cos \left(\theta \left\langle n_i^{f_j}, n_i^v \right\rangle \right)^m \right\},
$$

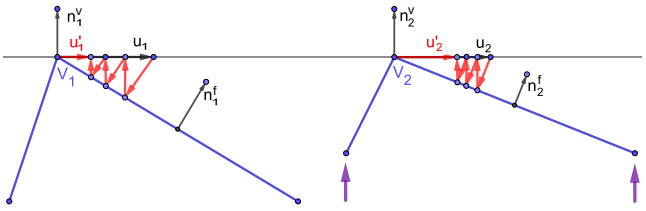


Fig. 5: 2D illustration of the error-aware vertex smoothing scheme. The blue line represents a surface mesh, V_1 and V_2 are the vertices on the mesh, the black horizontal line indicates the normal plane of these vertices, n_1^v and n_2^v denote the vertex normal vectors, respectively, and n_1^f and n_2^f stand for the face normal vectors. V_1 is sharper than V_2. Meanwhile, u_1 and u_2 (black) are updating vectors before applying projections. After multiple projections, the magnitude of updating vector u_1' is significantly reduced compared with that of u_2'.

where $n_i^{f_j}$ is the adjacent triangle normal, n_i^v denotes the vertex normal weighted by the adjacent triangle normal, and $\theta \left\langle n_i^{f_j}, n_i^v \right\rangle$ measures the angle between two faces. The vertex intensity is between 0 and 1. In practice, the parameter m is set to 3 in practice. After defining the feature intensity, we designed an adaptive condition for the collapse, which can be interpreted as follows: v_i can be collapsed to another vertex v_j, if $\Phi_i - \Phi_j > -\Psi \cdot \left({\Phi_i}^2 - 0.5 \right)$.

Error-aware edge flipping. We apply the flipping operation on the basis of the dihedral angle parameter Θ, whenever $\theta \left\langle n_e^{f_0}, n_e^{f_1} \right\rangle < \Theta$. We set $\Theta = 10^\circ$ in practice. Here $n_e^{f_0}$ and $n_e^{f_1}$ are two triangle normals that are incident to the flipping edge e.

5 Experimental Results

In this section, we demonstrate experimental results of the proposed remeshing framework. Then, we analyze the choice of the parameter k and the performance and convergence behavior of the proposed framework. Next, we evaluate the meshing quality with standard metrics, and compare our method with recent representative approaches. Finally, we perform limit tests and discuss the limitations of our algorithm. Our algorithm is implemented in Visual Studio 2015, under the Windows 10 operating system. All the results shown in the paper are conducted on a PC with 4.2 GHz CPU and 8 GB RAM.

As discussed in Sec. 1, our algorithm can either work directly on the original input, or start from the results produced by any of the previous methods. Figs. 1 and 6 show four examples of adaptive remeshing on smooth surfaces. Our algorithm achieves the best angle bounds compared with previous approaches. When applied on the outputs of other algorithms, our algorithm is able to further improve the mesh quality by efficiently eliminating large/small angles. We show two examples of remeshing inputs with sharp features (Fig. 7) or boundaries (the Mask example in supplemental materials). Our algorithm can successfully produce desired results

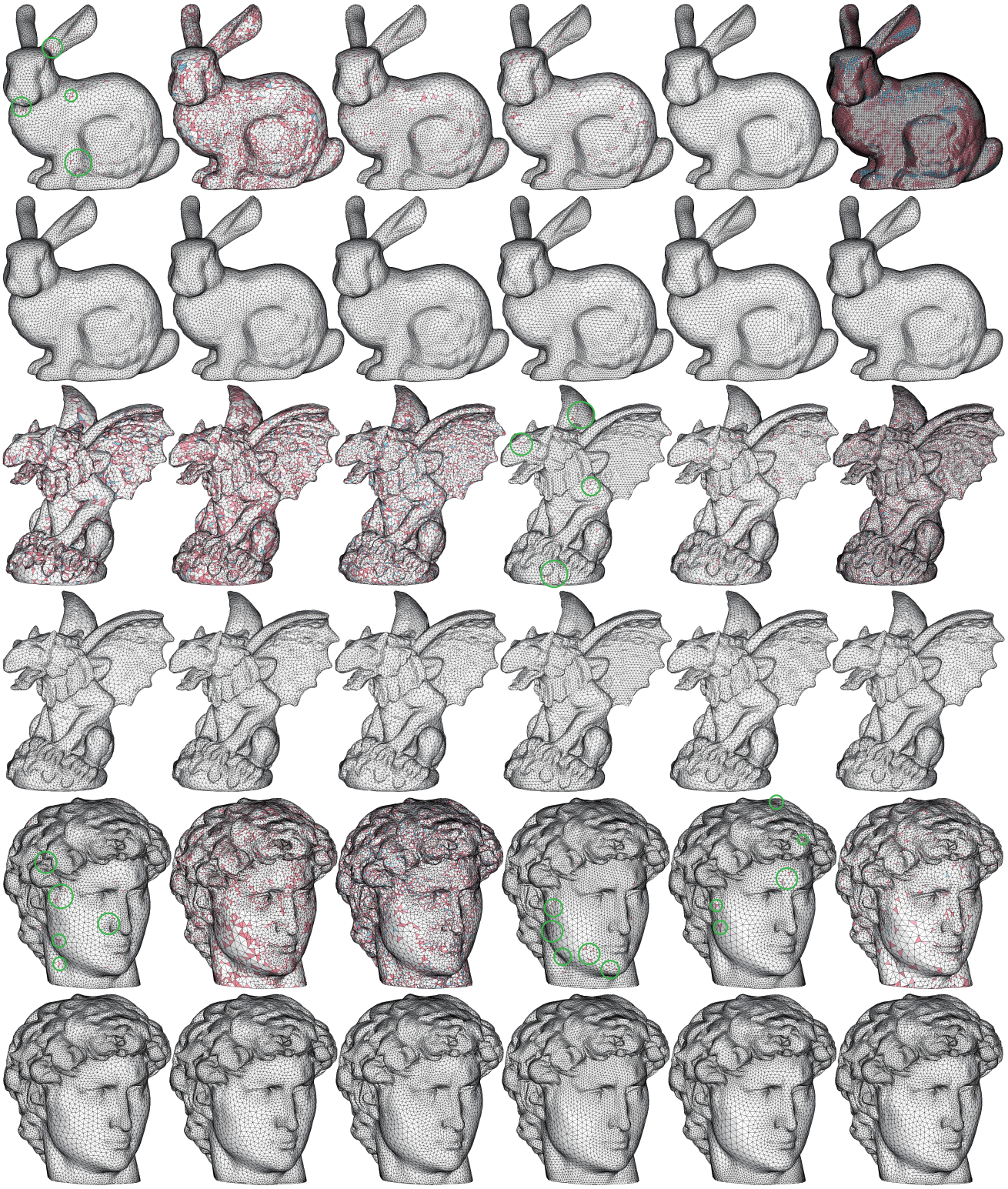


Fig. 6: Comparison of adaptive remeshing using the bunny, the gargoyle and David models. For each test model, the top row presents the input mesh and the remeshing results of previous approaches, including RAR [1], MAI [8], SPP [9], IFM [10], and NOB [11]. The bottom row of each test contains the results of our method (left most) and the results by using the corresponding mesh as initialization.

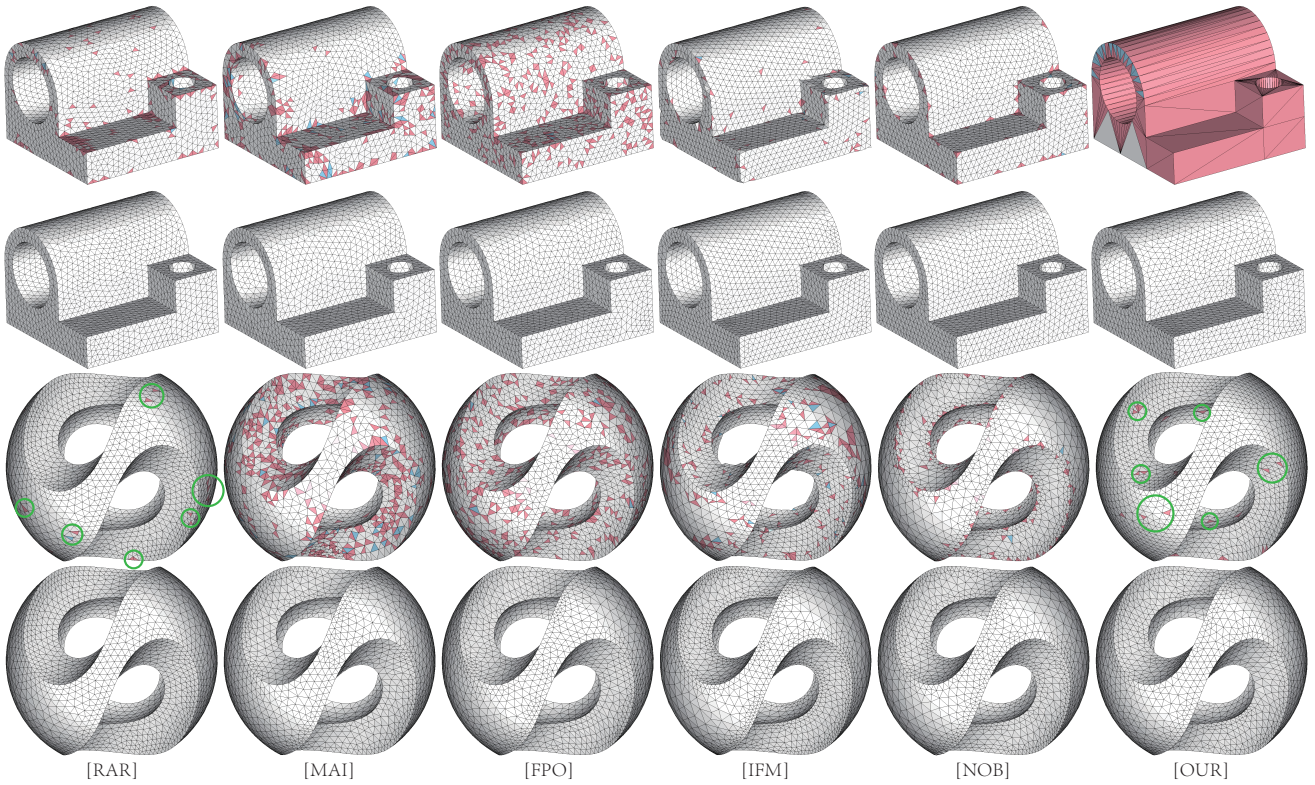


Fig. 7: Comparison of the remeshing results on models with sharp features or boundaries. The joint, sculpt, and mask models are presented from top to bottom. For each test model, the top row is the input mesh and the remeshing results of previous approaches, including RAR [1], MAI [8], FPO [48], IFM [10], and NOB [11]. The bottom row of each test contains the results of our method (left most) and the results by using the corresponding mesh as initialization. Note that the results of SPP [9] are not shown here as it does not support feature preservation.

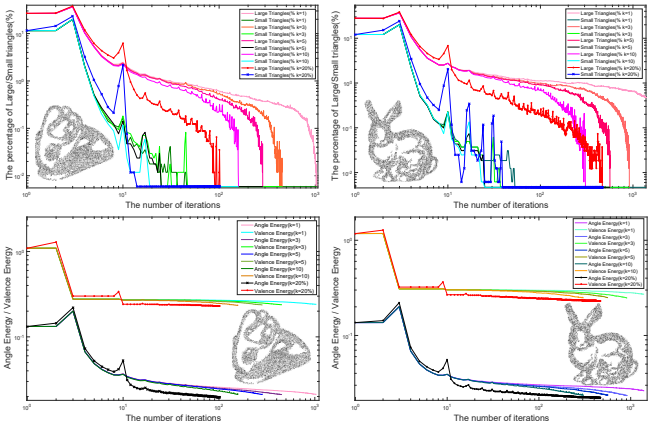


Fig. 8: Comparing the convergence of our algorithm on different k at each iteration. The x-axis is the number of iterations, and the y-axis represents the percentage of triangles with large/small angles. Left: uniform remeshing. Right: adaptive remeshing. Here, the number of iterations refers to each individual step in Alg. 1.

while preserving the features and boundaries, due to our carefully designed feature handling schemes. We further tested our algorithm on a large variety of input meshes with different numbers of vertices, and obtained the desired output meshes robustly.

Parameter selection. The parameter k is important. A small k value can improve the output mesh quality

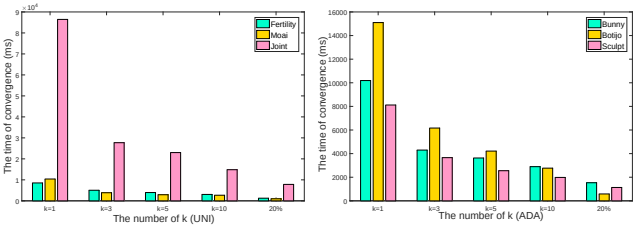


Fig. 9: Comparing the performance of our algorithm on different k at each iteration. The x-axis is the number of vertices in the output mesh $\mathcal{M}'$, and the y-axis indicates the time used for the convergence of each result. Left: uniform remeshing. Right: adaptive remeshing.

but this reduces algorithm performance. Meanwhile, a large k value can improve the performance of the proposed algorithm, but this might affect the distribution of the mesh vertices, which leads to the unsatisfactory fidelity of the output mesh. In our experiments, we set k equals to 20% of the number of triangles with large angle in each iteration by default, and k equals to 5 for error-aware remeshing. We have tested the influence of k for all the models used in this paper. We illustrate the influence of k on convergence and performance using the bunny and fertility models, as shown in Figs. 8 and 9, respectively. The results of other models can be found in supplemental materials.

Performance. We further analyze the performance and the robustness of our algorithm. We apply the

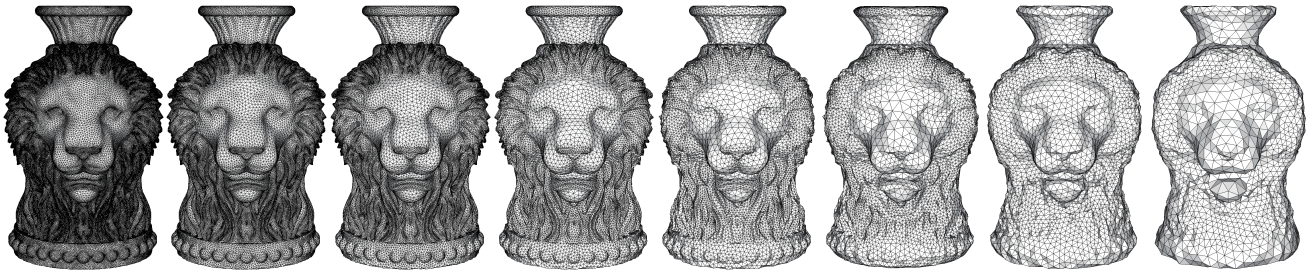


Fig. 10: Remeshing results of the vase-lion model with decreasing resolutions. The numbers of vertices of these multi-resolution are 115, 61, 42, 22, 10, 7, 4 and 2 k. Table 1 in supplementals presents the statistics.

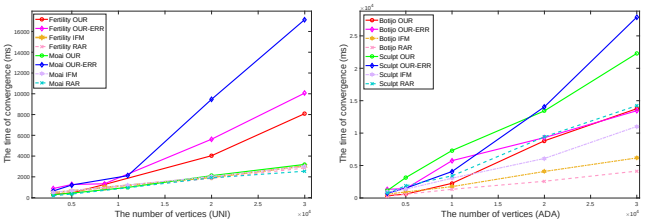


Fig. 11: Comparing the performance of our algorithm with that of RAR [1] and IFM [10]. The x-axis is the number of vertices in the final remeshing, and the y-axis represents the time used for the convergence of each result. *OUR* indicates our method using standard local operations, and *OUR-ERR* refers to our method using modified operations. Left: uniform remeshing. Right: adaptive remeshing.We set 20% of bad triangles as k in this experiment. Moreover, the performance of our method using standard local operations is in the same order of magnitude as RAR.

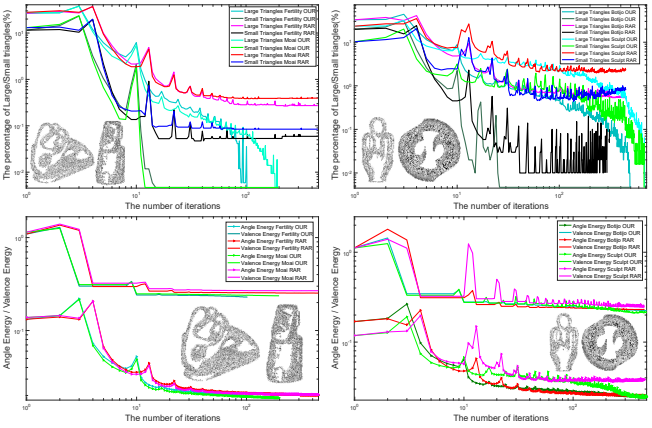


Fig. 12: Comparing the convergence of our algorithm with that of RAR [1]. Top row: the x-axis is the number of iterations, and the y-axis represents the percentage of large/small triangles. Bottom row: the angle/valance energies vs the number of iterations. Left: uniform remeshing. Right: adaptive remeshing.

algorithm on a complex input mesh, i.e., the vase-lion model with 100 k vertices, which cannot be handled satisfactorily by any of the previous methods. We choose different numbers of vertices (Fig. 10) to produce a series of multi-resolution non-obtuse meshes. Fig. 11 shows the timing statistics versus the level of resolution. We also compare the speed with the currently most efficient algorithms, i.e., RAR [1] and instant field meshing (IFM) [10] in Fig. 11, to verify that our algorithm is almost as fast as RAR and IFM. Other recent approaches, i.e., Non-obtuse Remeshing (NOB) [11], Minimal Angle Improvement (MAI) [8], and SPP [9], would spend minutes or even cannot achieve the user-specified angle bounds on this difficult input.

Convergence. To validate the convergence behavior of the algorithm, we plot the number of triangles with large/small angles vs. the number of iterations for both our approach and RAR in Fig. 12. We also measure the angle and valence energies. The angle energy is defined as the average sum of the squared distance between each angle and the optimal angle (i.e., 60°). The valence energy is defined as the average sum of the squared distance of the valance of each vertex and the optimal valence, i.e., 6 for interior vertices and 4 for boundary vertices. Several examples are selected for this experiment, i.e., fertility, moai, botijo and sculpt. We observed the convergence of our algorithm in all tests.

The convergence of our algorithm can be guaranteed algorithmically. In each iteration, we locally process triangles with small/large angles. The operation is accepted only if the small angle is improved or the large angle is reduced. However, we also note that processing angles in a region might affect neighboring triangles(small spikes in Fig. 12). On the one hand, these small fluctuations correspond to edge splitting and edge collapsing in each iteration. Then, the angle quality can be further experimentally optimized in later smoothing iterations. On the other hand, the smoothing operation is related to the ODT [15], [55] energy function, which has been shown to improve the triangle quality considering a fixed connectivity by optimizing vertex positions. In contrast, our combined operations (large angle removal and small angle improvement) can be seen as a connectivity optimization via vertex teleportation, which helps to improve the valence by jumping out of the local minima.

Evaluation and Comparison. Finally, we compare the mesh quality with state-of-the-art approaches, including 1) methods based on blue-noise sampling, e.g., MPS [29], FPO [48], and SPP [9]; 2) CVT, e.g., standard CVT [33] and augmented CVT with obtuse triangle suppression [11]; 3) discrete optimization, e.g., RAR [1] and MAI [8]; and 4) IFM [10].

Table 1 lists the statistics. In each column, two

Model	Method	$\|\mathbf{X}\|$	$\|t\|$	Q_{min}	Q_{avg}	θ_{min}	$\bar{\theta}_{min}$	θ_{max}	$\theta_{<\beta_{max}}\%$	$\theta_{>\beta_{max}}\%$	$V_6\%$	$d_{RMS}(\times 10^{-3})$	$d_H(\times 10^{-2})$
Vaselion (Standard Local Operations) $\theta_{<\beta_{min}}=35°$ $\theta_{<\beta_{max}}=86°$	Input/OUR	50k/6.5k	100k/13k	0.01/**0.73**	0.57/0.91	0.38/35.17	29.53/52.13	178.26/**85.81**	65.05/**0.00**	66.46/**0.00**	28.41/73.21	– /2.05	– /1.73
	MPS/OUR	6.4k/6.4k	12.8k/12.8k	0.51/0.71	0.82/0.89	32.01/35.02	46.34/50.80	114.59/86.00	3.42/**0.00**	18.41/**0.00**	51.13/74.87	1.63/2.40	1.33/1.75
	RAR/OUR	6.8k/6.8k	13.7k/13.7k	0.40/0.71	0.89/0.90	17.02/35.03	50.57/51.06	124.60/85.99	1.05/**0.00**	3.25/**0.00**	73.80/76.55	2.30/2.60	1.61/1.75
	Mai/OUR	6.4k/6.4k	12.8k/12.8k	0.42/0.72	0.77/0.89	23.38/35.00	42.69/50.77	124.85/85.99	16.87/**0.00**	36.86/**0.00**	29.78/76.03	**0.71**/2.45	**0.27**/1.65
	CVT/OUR	6.4k/6.4k	12.8k/12.8k	0.41/0.71	0.85/0.90	18.05/35.08	47.98/51.33	127.04/86.00	4.74/**0.00**	12.27/**0.00**	57.64/76.18	2.15/2.60	1.10/1.70
	NOB/OUR	6.4k/6.4k	12.7k/12.7k	0.00/0.71	0.86/0.90	0.16/35.02	48.30/51.18	141.63/86.00	3.76/**0.00**	10.01/**0.00**	57.30/75.73	2.20/2.54	1.05/1.73
	SPP/OUR	6.5k/6.5k	13k/13k	0.29/0.72	0.81/0.91	10.71/**35.72**	44.53/51.74	126.61/85.99	12.74/**0.00**	20.68/**0.00**	51.19/77.07	2.01/2.87	1.48/3.18
	FPO/OUR	6.5k/6.5k	13k/13k	0.51/0.72	0.86/0.90	29.87/35.60	50.25/51.40	114.39/86.00	0.63/**0.00**	14.01/**0.00**	54.45/76.40	1.51/2.40	1.51/1.70
	IFM/OUR	6.5k/6.5k	13k/13k	0.41/0.72	0.93/**0.94**	20.23/35.04	54.41/**54.47**	123.77/86.00	0.22/**0.00**	1.25/**0.00**	84.07/**85.78**	2.25/2.61	1.40/1.84
Vaselion (Modified Local Operations) $\theta_{<\beta_{min}}=30°$ $\theta_{<\beta_{max}}=90°$	Input/OUR	50k/6.5k	100k/13k	0.01/0.64	0.57/0.87	0.38/30.11	29.53/49.08	178.26/90.00	52.20/**0.00**	59.11/**0.00**	28.41/66.03	– /1.54	– /1.26
	MPS/OUR	6.4k/6.4k	12.8k/12.8k	0.51/0.64	0.82/0.87	**32.01**/30.58	46.34/48.83	114.59/90.00	**0.00**/**0.00**	11.64/**0.00**	51.13/61.24	1.62/1.41	1.33/1.28
	RAR/OUR	6.8k/6.8k	13.7k/13.7k	0.40/**0.65**	0.89/0.88	17.02/30.02	50.57/49.67	124.60/**89.99**	0.26/**0.00**	1.49/**0.00**	73.80/66.77	2.30/1.77	1.61/1.23
	Mai/OUR	6.4k/6.4k	12.8k/12.8k	0.42/0.64	0.77/0.86	23.38/30.00	42.69/47.79	124.85/90.00	0.03/**0.00**	26.87/**0.00**	29.78/63.36	**0.71**/1.41	**0.27**/1.14
	CVT/OUR	6.4k/6.4k	12.8k/12.8k	0.41/0.64	0.85/0.87	18.05/30.00	47.98/48.83	127.04/90.00	1.12/**0.00**	7.18/**0.00**	57.64/63.44	2.15/1.88	1.10/0.99
	NOB/OUR	6.4k/6.4k	12.7k/12.7k	0.00/0.64	0.86/0.87	0.16/30.10	48.30/48.85	141.63/**89.99**	0.84/**0.00**	5.30/**0.00**	57.30/62.84	2.20/2.00	1.05/1.05
	SPP/OUR	6.5k/6.5k	13k/13k	0.16/0.64	0.81/0.87	7.64/30.03	44.50/48.59	157.68/**89.99**	3.92/**0.00**	14.11/**0.00**	51.19/61.57	2.01/1.86	1.48/1.46
	FPO/OUR	6.5k/6.5k	13k/13k	0.51/**0.65**	0.86/0.87	29.87/30.15	50.25/49.18	114.39/**89.99**	0.01/**0.00**	7.24/**0.00**	54.45/61.50	1.51/1.76	1.51/1.14
	IFM/OUR	6.5k/6.5k	13k/13k	0.19/**0.65**	**0.91**/0.90	11.33/30.11	**52.22**/51.77	155.23/**89.99**	0.39/**0.00**	1.76/**0.00**	**73.65**/72.68	1.34/1.34	1.16/0.99
Botijo	Input/OUR	3k/5k	6k/10k	0.03/0.72	0.59/0.90	1.57/35.11	30.72/51.67	176.54/85.98	64.44/**0.00**	63.96/**0.00**	37.83/72.48	– /0.51	– /0.35
	MPS/OUR	5k/5k	10k/10k	0.55/0.71	0.82/0.88	32.00/35.00	46.14/50.06	109.65/86.00	3.73/**0.00**	18.97/**0.00**	52.53/63.62	0.87/0.59	0.59/0.47
	RAR/OUR	5k/5k	10k/10k	0.57/0.71	0.89/0.90	30.10/35.05	50.76/51.22	106.72/85.98	0.24/**0.00**	2.03/**0.00**	76.33/76.93	1.18/1.06	1.11/1.10
	Mai/OUR	5k/5k	10k/10k	0.47/0.72	0.77/0.90	26.89/35.50	43.00/51.02	119.59/86.00	14.61/**0.00**	39.10/**0.00**	30.75/69.97	**0.35**/0.54	**0.13**/0.44
	CVT/OUR	5k/5k	10k/10k	0.62/0.72	0.92/0.92	31.43/35.76	52.98/52.89	99.14/85.97	0.06/**0.00**	0.81/**0.00**	81.30/81.39	0.72/0.68	0.34/0.34
	NOB/OUR	5k/5k	10k/10k	0.70/**0.74**	**0.94**/**0.94**	31.13/**36.41**	54.17/**54.15**	88.37/**84.94**	0.01/**0.00**	0.02/**0.00**	81.33/81.40	0.48/0.48	0.32/0.32
	SPP/OUR	5k/5k	10k/10k	0.70/0.71	0.89/0.90	33.02/35.57	50.50/51.22	89.47/85.97	0.04/**0.00**	1.24/**0.00**	67.36/75.34	0.68/0.64	0.57/0.57
	FPO/OUR	5k/5k	10k/10k	0.52/0.71	0.85/0.89	31.00/35.05	50.29/50.58	113.17/86.00	0.40/**0.00**	15.02/**0.00**	57.54/66.54	0.68/0.60	0.47/0.46
	IFM/OUR	5k/5k	10k/10k	0.36/0.72	0.93/0.93	15.92/35.39	54.02/54.01	132.86/86.00	0.92/**0.00**	2.20/**0.00**	86.63/**86.76**	0.51/0.51	0.41/0.40
Bunny	Input/OUR	35k/8k	70k/16k	0.00/0.71	0.71/0.91	0.00/35.11	36.75/51.85	180.00/86.00	31.44/**0.00**	53.82/**0.00**	75.27/76.17	– /0.38	– /0.33
	MPS/OUR	8k/8k	16k/16k	0.41/0.71	0.80/0.90	21.84/35.08	44.98/51.70	126.67/85.99	7.54/**0.00**	23.34/**0.00**	47.48/76.58	0.48/0.36	0.42/0.38
	RAR/OUR	8k/8k	16k/16k	0.56/0.71	0.90/0.90	28.96/35.02	51.46/51.58	107.15/86.00	0.23/**0.00**	1.28/**0.00**	75.54/75.60	0.35/0.35	0.43/0.43
	Mai/OUR	8k/8k	16k/16k	0.43/0.72	0.77/0.91	22.41/35.15	43.03/51.92	123.91/85.99	14.05/**0.00**	39.32/**0.00**	29.70/76.06	**0.34**/0.35	**0.12**/0.35
	CVT/OUR	8k/8k	16k/16k	0.64/0.72	0.93/0.93	34.80/36.00	54.23/54.02	98.78/85.98	0.01/**0.00**	0.28/**0.00**	**86.83**/85.78	0.53/0.53	0.37/0.35
	NOB/OUR	8k/8k	16k/16k	0.72/**0.74**	**0.94**/**0.94**	36.36/36.36	**54.45**/54.42	89.45/**84.46**	**0.00**/**0.00**	0.02/**0.00**	86.80/**86.83**	0.58/0.58	0.38/0.38
	SPP/OUR	8k/8k	16k/16k	0.70/0.72	0.89/0.90	**37.03**/35.04	51.07/51.42	89.51/86.00	**0.00**/**0.00**	3.43/**0.00**	74.76/82.26	0.42/0.37	0.56/0.51
	FPO/OUR	8k/8k	16k/16k	0.49/0.71	0.84/0.90	30.01/35.06	48.60/51.70	116.33/85.99	2.38/**0.00**	16.98/**0.00**	52.34/77.15	0.49/0.36	0.45/0.42
	IFM/OUR	8k/8k	16k/16k	0.39/0.71	0.92/0.91	20.84/35.01	52.78/52.61	129.51/85.99	0.77/**0.00**	2.84/**0.00**	78.97/80.34	0.42/0.40	0.47/0.45
Gargoyle	Input/OUR	30k/10k	60k/20k	0.22/**0.71**	0.77/0.90	7.88/35.00	42.03/51.57	143.85/**86.00**	20.95/**0.00**	29.39/**0.00**	41.33/71.22	– /0.65	– /0.56
	MPS/OUR	10k/10k	20k/20k	0.50/0.70	0.82/0.88	32.00/35.00	46.33/49.48	115.00/**86.00**	3.63/**0.00**	19.09/**0.00**	51.73/63.06	0.78/0.71	0.68/0.65
	RAR/OUR	10k/10k	20k/20k	0.00/**0.71**	0.69/0.89	0.00/**35.02**	37.68/50.62	180.00/**86.00**	37.25/**0.00**	42.02/**0.00**	50.69/76.27	0.62/0.71	0.57/0.54
	Mai/OUR	10k/10k	20k/20k	0.46/0.70	0.77/0.88	25.49/**35.02**	42.98/49.45	119.91/**86.00**	14.83/**0.00**	37.86/**0.00**	29.96/61.07	**0.33**/0.65	**0.13**/0.58
	CVT/OUR	10k/10k	20k/20k	0.44/**0.71**	0.87/0.89	21.91/35.01	49.42/50.44	123.36/**86.00**	2.27/**0.00**	5.84/**0.00**	62.59/66.86	0.84/0.70	0.38/0.45
	NOB/OUR	10k/10k	20k/20k	0.44/**0.71**	0.88/0.89	21.78/35.01	49.77/50.43	117.02/**86.00**	2.11/**0.00**	5.11/**0.00**	64.73/68.24	0.83/0.75	0.38/0.47
	SPP/OUR	10k/10k	20k/20k	0.24/0.70	0.77/0.88	9.22/35.00	41.92/49.57	148.57/**86.00**	23.03/**0.00**	29.03/**0.00**	44.65/61.93	0.98/0.77	0.86/0.85
	FPO/OUR	10k/10k	20k/20k	0.48/**0.71**	0.85/0.89	30.51/35.01	49.38/50.54	117.54/**86.00**	1.80/**0.00**	15.89/**0.00**	52.41/70.59	0.75/0.66	0.62/0.62
	IFM/OUR	10k/10k	20k/20k	0.41/**0.71**	**0.93**/**0.93**	20.00/35.01	**54.21**/53.88	122.84/**86.00**	0.15/**0.00**	1.12/**0.00**	80.72/**80.88**	0.72/0.71	0.56/0.56
David	Input/OUR	12k/15k	120k/60k	0.00/0.70	0.88/0.89	0.04/35.00	50.06/50.26	178.18/86.00	1.46/**0.00**	6.62/**0.00**	64.18/66.81	– /0.37	– /0.39
	MPS/OUR	15k/15k	59k/59k	0.41/**0.71**	0.81/0.90	23.41/35.00	45.39/51.19	126.68/86.00	5.45/**0.00**	22.10/**0.00**	49.29/67.87	0.65/0.56	0.54/0.54
	RAR/OUR	15k/15k	60k/60k	0.63/**0.71**	0.91/0.91	31.03/35.00	52.07/52.05	99.85/86.00	0.08/**0.00**	0.53/**0.00**	74.05/74.09	0.57/0.56	0.95/0.94
	Mai/OUR	15k/15k	60k/60k	0.53/0.70	0.82/0.89	28.95/35.00	46.73/50.26	111.83/86.00	0.07/**0.00**	23.75/**0.00**	41.68/65.30	**0.18**/0.47	**0.08**/0.50
	CVT/OUR	15k/15k	60k/60k	0.56/**0.71**	0.92/0.92	33.10/35.04	53.08/52.87	107.82/86.00	0.01/**0.00**	0.92/**0.00**	80.06/79.81	0.63/0.61	0.59/0.59
	NOB/OUR	15k/15k	60k/60k	0.69/**0.71**	0.92/0.92	36.04/35.00	53.20/53.13	89.97/86.00	**0.00**/**0.00**	0.62/**0.00**	79.92/79.94	0.54/0.54	0.30/0.30
	SPP/OUR	15k/15k	60k/60k	0.16/**0.71**	0.74/0.88	5.92/35.00	39.62/50.13	150.30/**85.00**	32.05/**0.00**	35.66/**0.00**	41.10/67.05	0.88/0.63	1.06/0.85
	FPO/OUR	15k/15k	10k/10k	0.50/0.70	0.85/0.89	27.90/35.00	50.25/50.72	115.32/86.00	0.60/**0.00**	14.44/**0.00**	56.62/68.15	0.61/0.57	0.68/0.67
	IFM/OUR	15k/15k	60k/60k	0.47/0.70	**0.94**/**0.94**	26.51/**35.06**	**54.80**/54.64	119.47/86.00	0.05/**0.00**	0.65/**0.00**	**85.49**/**85.49**	0.55/0.55	0.59/0.59
Joint	Input/OUR	0.2k/3.4k	0.4k/6.8k	0.01/0.71	0.22/0.88	0.48/35.09	9.35/49.87	173.21/85.99	98.65/**0.00**	91.03/**0.00**	27.15/75.21	– /0.41	– /0.25
	MPS/OUR	3k/3k	6k/6k	0.59/0.70	0.82/0.89	31.98/35.08	46.08/50.80	104.96/85.98	2.79/**0.00**	21.34/**0.00**	54.23/71.46	0.55/0.48	0.37/0.29
	RAR/OUR	3.4k/3.4k	6.8k/6.8k	0.07/0.70	0.86/0.88	4.50/35.14	48.28/49.70	171.00/85.98	1.42/**0.00**	5.63/**0.00**	78.36/78.58	**0.40**/0.40	0.31/0.26
	Mai/OUR	3k/3k	6k/6k	0.00/0.71	0.87/0.92	0.11/35.28	49.78/52.70	179.77/85.99	4.69/**0.00**	8.21/**0.00**	75.11/76.84	0.46/0.48	0.22/0.25
	CVT/OUR	3.4k/3.4k	6.8k/6.8k	0.58/0.72	0.91/0.92	32.90/36.00	52.99/53.00	105.67/85.96	0.03/**0.00**	3.29/**0.00**	82.03/83.29	0.60/0.48	0.26/0.25
	NOB/OUR	3.4k/3.4k	6.8k/6.8k	0.57/0.71	0.93/0.93	26.99/35.51	53.42/53.85	101.05/85.97	0.62/**0.00**	3.26/**0.00**	81.53/83.19	0.99/0.45	0.42/**0.21**
	FPO/OUR	3.4k/3.4k	6.8k/6.8k	0.54/0.72	0.85/0.91	32.39/35.50	49.90/52.55	110.77/85.94	0.28/**0.00**	15.71/**0.00**	57.58/80.28	0.74/0.47	0.39/0.31
	IFM/OUR	3.4k/3.4k	6.8k/6.8k	0.53/**0.73**	**0.96**/0.95	31.12/**36.19**	**56.11**/55.91	110.99/**85.65**	0.07/**0.00**	0.31/**0.00**	90.44/**90.70**	0.58/0.47	1.07/0.36
Sculpt	Input/OUR	3.7k/3k	7.3k/6k	0.68/0.70	0.92/0.90	36.43/35.06	53.90/51.10	94.48/85.98	**0.00**/**0.00**	0.91/**0.00**	88.99/74.42	– /0.33	– /0.12
	MPS/OUR	3k/3k	6k/6k	0.49/0.71	0.82/0.90	29.23/35.20	46.28/50.92	115.85/85.99	3.21/**0.00**	19.66/**0.00**	54.65/81.18	0.42/0.36	0.22/0.14
	RAR/OUR	3k/3k	6k/6k	0.56/0.70	**0.92**/**0.92**	27.73/35.06	53.05/53.09	102.54/**85.93**	0.12/**0.00**	0.43/**0.00**	85.95/**86.13**	**0.27**/**0.27**	**0.09**/**0.09**
	Mai/OUR	3k/3k	6k/6k	0.50/0.70	0.83/0.90	30.00/35.00	47.17/**51.41**	115.24/86.00	6.89/**0.00**	20.44/**0.00**	56.20/74.07	0.39/0.34	0.10/0.13
	CVT/OUR	3k/3k	6k/6k	0.56/**0.72**	0.86/0.90	31.63/**35.47**	48.98/51.17	108.40/**85.93**	0.63/**0.00**	11.19/**0.00**	67.68/82.21	0.70/0.35	0.24/0.20
	NOB/OUR	3k/3k	6k/6k	0.57/0.71	0.87/0.89	29.48/35.02	49.52/50.56	103.49/85.99	0.13/**0.00**	10.09/**0.00**	67.78/72.21	0.90/0.43	0.31/0.18
	FPO/OUR	3k/3k	6k/6k	0.51/0.71	0.85/0.89	30.00/35.00	49.76/50.40	114.49/85.99	0.55/**0.00**	15.89/**0.00**	57.38/73.48	0.34/0.34	0.16/0.13
	IFM/OUR	3k/3k	6k/6k	0.32/0.71	0.87/0.89	16.64/35.07	48.87/50.21	137.34/85.96	6.96/**0.00**	10.80/**0.00**	70.56/76.72	0.49/0.44	0.80/0.24

TABLE 1: Comparison of remeshing quality with previous techniques. The best result of each measurement is marked in **bold** font. $|\mathbf{X}|$ is the number of vertices; $|t|$ represents the number of triangles. Sec 5 explains the other measurements.

groups of values are separated by "/". The former means the quality before applying our method, and the latter indicates the quality after applying our method. In the last example in the table, the values correspond to the example of multi-resolution remeshing shown in Fig. 10. We use the standard evaluation criteria utilized in previous works [11], [8], [9]. In addition, θ_{min} and θ_{max} are the minimal and maximal angles in a mesh, respectively, and $\bar{\theta}_{min}$ denotes the average minimal angle of all triangles. In isotropic remeshing, the optimal angle of θ_{min} and $\bar{\theta}_{min}$ is 60° for a planar surface. However, achieving this value for curved surfaces is impossible, which is also affected by the existence of singular vertices. Q_{min} and $\bar{Q}_{ave}$ measure the regularity of triangles. An equilateral triangle has $Q(t) = 1$ and a degenerate triangle has $Q(t) = 0$ [56]. A larger value of $Q(t)$ indicates a better shape of triangle t. The global regularity of the remesh is dominated by the percentage of valence 6 vertices. We see that CVT-based approaches have improved regularity. We also state the percentages of the triangles with the minimal angle smaller than β_{min} and the maximal angle larger than β_{max}. Our method is shown to be the only one that can achieve both no small and large angles in practice.

The most related work with ours is RAR [1], which improves the mesh quality by further equaling the edge lengths. RAR has the fastest speed because only the splitting of long edges and collapsing of short edges are involved in the computation. However, RAR lacks explicit control of the angle quality, as listed in Table 1, as a certain amount of small and obtuse angles exist in their results. Another closely related work is MAI [8], which aims to improve the minimal angle quality; however, it introduces numerous obtuse triangles at the same time. The efficiency of MAI can be low when the target number of vertices is high, because it processes one triangle at one iteration. The blue-noise sampling based approaches have low mesh quality due to their random nature. Although the most recent work SPP is able to produce non-obtuse meshes, it has no explicit control of the minimal and maximal angle bounds as our method. The same problem exists for CVT and NOB. The performances of blue-noise remeshing and CVT-related approaches are relatively slow because the global optimization is involved. Although IFM has high performance and regularity, it cannot produce acute meshes as our method. We exclude the results of MAI for uniform remesh because it is designed solely for automatically sizing control with bounded error.

Discussion and Limitation. Our method runs robustly on a large variety of input meshes. However, we have some issues when the required angle bounds are too tight, or when remeshing surfaces with very thin and long features uses a small target number of vertices.

We conducted two limit tests to explore the limitations of our remeshing framework. In the first example, we select a smooth model without tiny features (i.e., the Venus body) to test the extreme angle bounds that our algorithm can achieve, as shown in Fig. 13. We set the number of vertices to 2k, and keep increasing the lower bound and decreasing the upper bound until the algorithm no longer converges. Starting from the range $[35°, 86°]$, the algorithm stops working when it reaches the bounds $[46°, 80°]$ and $[40°, 76°]$. This example demonstrates that our approach can produce meshes with very high triangle quality. The result also confirms the common observation that the better the valence, the higher the triangle quality, and the larger the approximation error (Hausdorff distance).

In the second example, we select a model with very thin features (the ear of the elk model) to test the robustness of the algorithm. Starting from 2 k vertices, we gradually decrease the number of vertices, as shown in Fig 14. The input angle bounds of this example are $[30°, 90°]$. The algorithm stops working when the vertex number reaches 800. This test also shows that the approximation error increases when the vertex budget decreases. The statistics of these tests are provided in the supplemental materials.

These limitations can also be observed in the lion-vase examples as shown in Figs. 1 and 10. This input mesh contains numerous tiny sharp features around the hair region. Although our algorithm could improve the angle quality over existing methods, it cannot further reduce the Hausdorff distance near feature regions using the default angle bounds $[35°, 86°]$. If we slightly loosen the angle bounds a little bit to $[30°, 90°]$ and apply our error-aware local operators, then we can further reduce the approximation errors (except when comparing with MAI) as well as improve the angle quality.

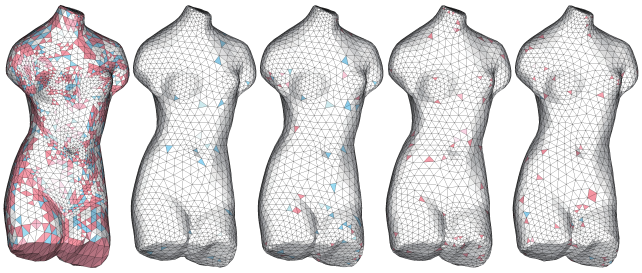


Fig. 13: Limit test on the Venus body model by increasing the lower angle bound and decreasing the upper bound using a fixed number of vertices. From left to right are: the input, our results with bounds $[40°, 77°]$, $[40°, 76°]$, $[45°, 80°]$, $[46°, 80°]$. The red and blue colors indicate the triangles with angles large than $77°$ and smaller than $45°$, respectively. The quality statistics is given in Table 1 in the supplemental material.

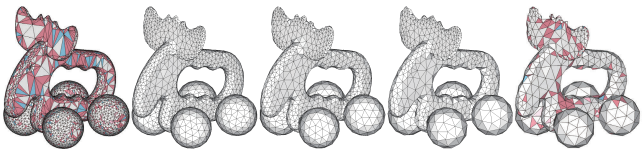


Fig. 14: Limit test on the Elk model by decreasing the vertex budget. From left to right: input, result of 2k, 1.6k, 1.2k, and 0.8k vertices. The algorithm does not converge when using 0.8k vertices. The input angle bounds in this example are $[30°, 90°]$. The quality statistics are given in Table 1 in the supplemental material.

6 Conclusion and Future Works

We have presented a novel isotropic remeshing framework that simultaneously removes large angles and improves small angles. The key components are two carefully designed criteria for point insertion and deletion. Our approach can either be used directly on input raw meshes, or be used as a post-processing step for any previous remeshing algorithm. The proposed method is shown to run robustly for a large variety of input meshes, even for those models with complicated geometry details that cannot be handled faithfully with state-of-the-art approaches.

Although our approach can achieve the best angle quality compared with existing approaches, challenges are present that deserve further investigation. For example, how to select the smallest number of points for a given approximation tolerance? What is the relationship between the angle and valence

energies? In the future, we also plan to improve the performance of our framework to achieve real-time remeshing (e.g., by GPU acceleration [57]) with massive models, extend our method for anisotropic remeshing [58], [59], and apply our approach to applications in animation and simulation.

ACKNOWLEDGMENTS

We thank anonymous reviewer for their valuable comments and suggestions. This work is partially funded by the National Natural Science Foundation of China (61772523, 61620106003, 61331018), the Beijing Natural Science Foundation (4184102), and the KAUST Visual Computing Center. Y. Wang and D.-M. Yan are joint first author with equal contribution. D.-M. Yan is the corresponding author.

REFERENCES

[1] M. Dunyach, D. Vanderhaeghe, L. Barthe, and M. Botsch, "Adaptive remeshing for real-time mesh deformation," in *Eurographics Short Papers Proceedings*, 2013, pp. 29–32.

[2] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, "Recent advances in remeshing of surfaces," in *Shape Analysis and Structuring*, 2008, pp. 53–82.

[3] P. Alliez, É. C. d. Verdiére, O. Devillers, and M. Isenburg, "Isotropic surface remeshing," in *Shape Modeling International - SMI*, 2003, pp. 49–58.

[4] J. R. Shewchuk, "What is a good linear element? interpolation, conditioning, and quality measures," in *11th Intl. Meshing Roundtable*, 2002, pp. 115–126.

[5] R. Kimmel and J. Sethian, "Minimal discrete curves and surfaces," in *Proc. of National Academy of Sci.*, vol. 95, 1998, p. 8431C8435.

[6] X. Ying, X. Wang, and Y. He, "Saddle vertex graph (SVG): a novel solution to the discrete geodesic problem," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 32, no. 6, pp. 170:1–170:12, 2013.

[7] K. Crane, F. de Goes, M. Desbrun, and P. Schröder, "Digital geometry processing with discrete exterior calculus," in *ACM SIGGRAPH 2013 Courses*, 2013, pp. 7:1–7:126.

[8] K. Hu, D.-M. Yan, D. Bommes, P. Alliez, and B. Benes, "Error-bounded and feature preserving surface remeshing with minimal angle improvement," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 23, no. 12, pp. 2560–2573, 2017.

[9] A. Ahmed, J. Guo, D.-M. Yan, J.-Y. Franceschi, X. Zhang, and O. Deussen, "A simple push-pull algorithm for blue-noise sampling," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 23, no. 12, pp. 2496–2508, 2017.

[10] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung, "Instant field-aligned meshes," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 189:1–189:15, 2015.

[11] D.-M. Yan and P. Wonka, "Non-obtuse Remeshing with Centroidal Voronoi Tessellation," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 22, no. 9, pp. 2136–2144, 2016.

[12] P. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms," in *SIGGRAPH 97 Course Notes: Multiresolution Surface Modeling*, 1997.

[13] Y. Liu, C. Xu, D. Fan, and Y. He, "Efficient construction and simplification of Delaunay meshes," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 174:1–174:13, 2015.

[14] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On centroidal Voronoi tessellation - energy smoothness and fast computation," *ACM Trans. on Graphics*, vol. 28, no. 4, pp. 101:1–101:11, 2009.

[15] L. Chen, "Mesh smoothing schemes based on optimal delaunay triangulations," in *International Meshing Roundtable*, 2004, pp. 109–120.

[16] T. Brochu and R. Bridson, "Robust topological operations for dynamic explicit surfaces," *SIAM J. Sci. Comput.*, vol. 31, no. 4, pp. 2472–2493, June 2009.

[17] M. Botsch and L. Kobbelt, "A remeshing approach to multiresolution modeling," in *Proc. of Symp. of Geometry Processing*, 2004, pp. 189–196.

[18] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. T. Silva, M. Tarini, and D. Zorin, "Quad-mesh generation and processing: A survey," *Computer Graphics Forum*, vol. 32, no. 6, pp. 51–76, 2013.

[19] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon Mesh Processing*. AK Peters, 2010.

[20] S.-W. Cheng, T. K. Dey, and J. R. Shewchuk, *Delaunay Mesh Generation*. CRC Press, 2012.

[21] B. S. Baker, E. Grosse, and C. S. Raffery, "Nonobtuse triangulation of polygons," *Disc. & Comp. Geom.*, vol. 3, pp. 147–168, 1988.

[22] M. Bern, S. Mitchell, and J. Ruppert, "Linear-size nonobtuse triangulation of polygons," in *ACM Symp. on Comp. Geom.*, 1994, pp. 221–230.

[23] H. Erten and A. Üngör, "Computing acute and non-obtuse triangulations," in *CCCG 2007*, 2007, pp. 205–208.

[24] H. Erten and A. Üngör, "Computing triangulations without small and large angles," in *Sixth International Symposium on Voronoi Diagrams, ISVD 2009*, 2009, pp. 192–201.

[25] S. Saraf, "Acute and nonobtuse triangulations of polyhedral surfaces," *European Journal of Combinatorics*, vol. 30, no. 4, pp. 833 – 840, 2009.

[26] J. Li and H. Zhang, "Nonobtuse remeshing and decimation," in *Proc. of Eurographics Symposium on Geometry Processing*, 2006, p. Proc. of Symp. of Geometry Processing.

[27] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms," *SIAM Review*, vol. 41, pp. 637–676, 1999.

[28] Z. Chen, J. Cao, and W. Wang:, "Isotropic surface remeshing using constrained centroidal delaunay mesh," *Computer Graphics Forum*, vol. 31, no. 7-1, pp. 2077–2085, 2012.

[29] D.-M. Yan and P. Wonka, "Gap processing for adaptive maximal Poisson-disk sampling," *ACM Trans. on Graphics*, vol. 32, no. 5, pp. 148:1–148:15, 2013.

[30] S. A. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[31] P. Alliez, M. Meyer, and M. Desbrun, "Interactive Geometry Remeshing," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 21(3), pp. 347–354, 2002.

[32] V. Surazhsky, P. Alliez, and C. Gotsman, "Isotropic remeshing of surfaces: a local parameterization approach," in *12th Intl. Meshing Roundtable*, 2003, pp. 204–231.

[33] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted Voronoi diagram," *Computer Graphics Forum (Proc. SGP)*, vol. 28, no. 5, pp. 1445–1454, 2009.

[34] D.-M. Yan, G. Bao, X. Zhang, and P. Wonka, "Low-resolution remeshing using the localized restricted Voronoi diagram," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 20, no. 10, pp. 418–1427, 2014.

[35] X. Du, X. Liu, D.-M. Yan, C. Jiang, J. Ye, and H. Zhang, "Field-aligned isotropic surface remeshing," *Computer Graphics Forum*, 2018, accepted.

[36] H. Edelsbrunner and N. R. Shah, "Triangulating topological spaces," *IJCGA*, vol. 7, no. 4, pp. 365–378, 1997.

[37] F. Sun, Y.-K. Choi, W. Wang, D.-M. Yan, Y. Liu, and B. Lévy, "Obtuse triangle suppression in anisotropic meshes," *Comp. Aided Geom. Design*, vol. 28, no. 9, pp. 537–548, 2011.

[38] O. Sifri, A. Sheffer, and C. Gotsman, "Geodesic-based surface remeshing," in *Proceedings of the 12th International Meshing Roundtable, IMR 2003*, 2003, pp. 189–199.

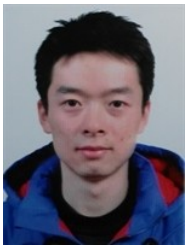
[39] G. Peyré and L. D. Cohen, "Geodesic remeshing using front propagation," *Int. J. Comput. Vision*, vol. 69, no. 1, pp. 145–156, 2006.

[40] Y. Fu and B. Zhou, "Direct sampling on surfaces for high quality remeshing," in *ACM symposium on Solid and physical modeling*, 2008, pp. 115–124.

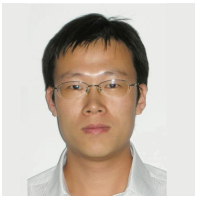
[41] Y.-J. Liu, Z. Chen, and K. Tang, "Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 8, pp. 1502–1517, 2011.

[42] X. Wang, X. Ying, Y.-J. Liu, S.-Q. Xin, W. Wang, X. Gu, W. Mueller-Wittig, and Y. He, "Intrinsic computation of cen-

troidal Voronoi tessellation (CVT) on meshes," *Computer-Aided Design*, vol. 58, no. 0, pp. 51–61, 2015.
[43] Y.-J. Liu, C.-X. Xu, R. Yi, D. Fan, and Y. He, "Manifold differential evolution (mde): A global optimization method for geodesic centroidal voronoi tessellations on meshes," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 35, no. 6, pp. 243:1–11, 2016.
[44] Z. Chen, Z. Yuan, Y.-K. Choi, L. Liu, and W. Wang, "Variational blue noise sampling," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 18, no. 10, pp. 1784–1796, 2012.
[45] J. Guo, D.-M. Yan, X. Jia, and X. Zhang, "Efficient maximal Poisson-disk sampling and remeshing on surfaces," *Computers & Graphics*, vol. 46, no. 6-8, pp. 72–79, 2015.
[46] M. S. Ebeida, A. A. Rushdi, M. A. Awad, A. H. Mahmoud, D. Yan, S. A. English, J. D. Owens, C. L. Bajaj, and S. A. Mitchell, "Disk density tuning of a maximal random packing," *Computer Graphics Forum (Proc. SGP)*, vol. 35, no. 5, pp. 259–269, 2016.
[47] A. Abdelkader, A. H. Mahmoud, A. A. Rushdi, S. A. Mitchell, J. D. Owens, and M. S. Ebeida, "A constrained resampling strategy for mesh improvement," *Computer Graphics Forum (Proc. SGP)*, vol. 36, no. 5, pp. 189–201, 2017.
[48] D.-M. Yan, J. Guo, X. Jia, X. Zhang, and P. Wonka, "Blue-noise remeshing with farthest point optimization," *Computer Graphics Forum (Proc. SGP)*, vol. 33, no. 5, pp. 167–176, 2014.
[49] R. Dyer, H. Zhang, and T. Möller, "Delaunay mesh construction," in *Eurographics Symposium on Geometry Processing*, 2007, pp. 273–282.
[50] V. Surazhsky and C. Gotsman, "Explicit surface remeshing," in *Symposium on Geometry Processing*, 2003, pp. 20–30.
[51] Y. Li, E. Zhang, Y. Kobayashi, and P. Wonka, "Editing operations for irregular vertices in triangle meshes," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 29, no. 6, pp. 153:1–153:11, 2010.
[52] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," in *Proc. ACM SIGGRAPH*, 2002, pp. 355–361.
[53] M. Nieser, J. Palacios, K. Polthier, and E. Zhang, "Hexagonal global parameterization of arbitrary surfaces," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 18, no. 6, pp. 865–878, 2011.
[54] B. Lévy and Y. Liu, "L p centroidal voronoi tessellation and its applications," in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4. ACM, 2010, p. 119.
[55] Z. Gao, Z. Yu, and M. J. Holst, "Feature-preserving surface mesh smoothing via suboptimal delaunay triangulation," *Graphical Models*, vol. 75, no. 1, pp. 23–38, 2013.
[56] P. Frey and H. Borouchaki, "Surface mesh evaluation," in *6th Intl. Meshing Roundtable*, 1997, pp. 363–374.
[57] Y.-S. Leung, X. Wang, Y. He, Y.-J. Liu, and C. C. L. Wang, "A unified framework for isotropic meshing based on narrow-band euclidean distance transformation," *Computational Visual Media*, vol. 1, no. 3, pp. 239–251, 2015.
[58] X. Wang, T. H. Le, X. Ying, Q. Sun, and Y. He, "User controllable anisotropic shape distribution on 3D meshes," *Computational Visual Media*, vol. 2, no. 4, pp. 305–319, 2016.
[59] Y. Cai, X. Guo, Y. Liu, W. Wang, W. Mao, and Z. Zhong, "Surface approximation via asymptotic optimal geometric partition," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 23, no. 12, pp. 2613–2626, 2017.



Yiqun Wang received his Bachelor's degree from Chongqing University, China. He is working toward the PhD degree in Institute of Automation, Chinese Academy of Sciences. His research interests include computer graphics, geometric processing and pattern recognition.



Dong-Ming Yan received his PhD degree from Hong Kong University in 2010, and his Bachelor's and Master's degrees from Tsinghua University in 2002 and 2005, respectively. He is currently an associate professor at the National Laboratory of Pattern Recognition of the Institute of Automation, Chinese Academy of Sciences. His research interests include computer graphics and geometric processing.

Xiaohan Liu received his BE degree in software engineering from Nanjing University of Aeronautics and Astronautics in 2017. He is currently working toward his PhD degree in Institute of Automation, Chinese Academy of Sciences. His research interests include computer graphics and virtual reality.

Chengcheng Tang received his PhD and MS degrees from King Abdullah University of Science and Technology (KAUST) in 2015 and 2011, respectively, and his bachelor's degree from Jilin University in 2009. He is currently a postdoctoral scholar in the Computer Science Department at Stanford University. His research interests include computer graphics, applied geometry, computational design, and machine learning.

Jianwei Guo is an assistant researcher in National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences(CASIA). He received his Ph.D. degree in computer science from CASIA in 2016, and bachelor degree from Shandong University in 2011. His research interests include computer graphics, geometry processing and 3D shape analysis.

Xiaopeng Zhang received the PhD degree in computer science from Institute of Software, Chinese Academic of Sciences in 1999. He is a professor in National Laboratory of Pattern Recognition at Institute of Automation, Chinese Academic of Sciences. He received the National Scientific and Technological Progress Prize (second class) in 2004. His main research interests include computer graphics and image processing.

Peter Wonka received his PhD in computer science and his MS in urban planning from the Technical University of Vienna, Vienna, Austria, in 2001 and 2002, respectively. He is currently a professor in the Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. His research interests include computer graphics, visualization, computer vision, remote sensing, image processing, and machine learning.