Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement

Kaimo Hu, Dong-Ming Yan, David Bommes, Pierre Alliez, and Bedrich Benes

Abstract—Surface remeshing is a key component in many geometry processing applications. The typical goal consists in finding a mesh that is (1) geometrically faithful to the original geometry, (2) as coarse as possible to obtain a low-complexity representation and (3) free of bad elements that would hamper the desired application (e.g., the minimum interior angle is above an application-dependent threshold). Our algorithm is designed to address all three optimization goals simultaneously by targeting prescribed bounds on approximation error δ, minimal interior angle θ and maximum mesh complexity N (number of vertices). The approximation error bound δ is a hard constraint, while the other two criteria are modeled as optimization goals to guarantee feasibility. Our optimization framework applies carefully prioritized local operators in order to greedily search for the coarsest mesh with minimal interior angle above θ and approximation error bounded by δ. Fast runtime is enabled by a local approximation error estimation, while implicit feature preservation is obtained by specifically designed vertex relocation operators. Experiments show that for reasonable angle bounds ($\theta \leq 35^\circ$) our approach delivers high-quality meshes with implicitly preserved features (no tagging required) and better balances between geometric fidelity, mesh complexity and element quality than the state-of-the-art.

Index Terms—Surface remeshing, error-bounded, feature preserving, minimal angle improvement, saliency function

1 Introduction

Surface remeshing is a key component in many geometry processing applications such as simulation, deformation, or parametrization [1]. While many remeshing techniques are goal-specified, a common goal is to find a satisfactory balance between the following three criteria:

- The output mesh should be a good approximation of the input, making the *geometric fidelity*, usually measured as the approximation error, a key requirement for most applications.
- The *quality of mesh elements* is crucial for robust geometry processing and numerical stability of simulations, which requires fairly regular meshes in terms of both geometry and connectivity. Particularly, a lower bound on the minimal angle is vital for many simulation applications [2].
- *Mesh complexity*, measured as the number of mesh elements, is important for an efficient representation of complex shapes. Since mesh complexity usually conflicts with geometric fidelity and element quality, a "just enough" resolution for the required element quality and geometric fidelity should be the goal.

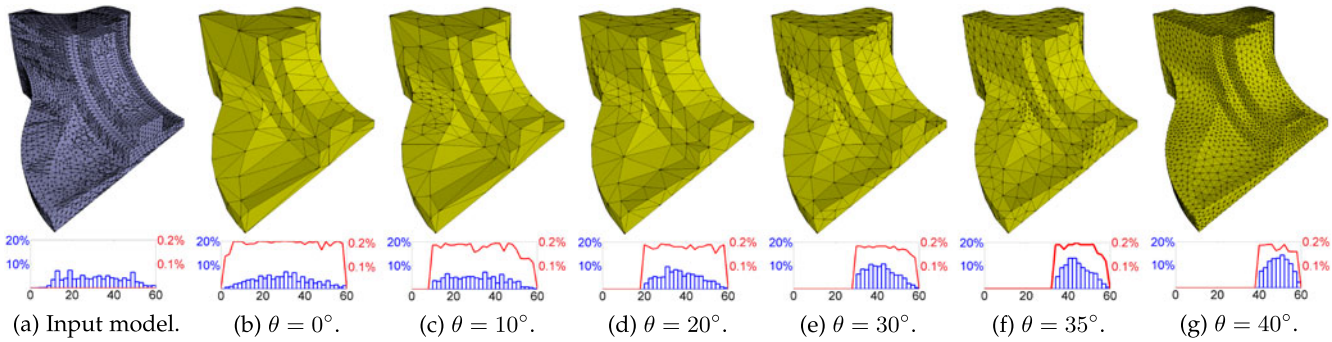
However, to the best of our knowledge, only a few approaches fulfill all of the above criteria. Most existing methods that generate meshes with high element quality often require high mesh complexity or introduce high approximation error [3], [4]. The error-driven methods, while preserving the results in controllable geometric fidelity and low mesh complexity, do not deal with the element quality [5], [6]. In addition, many approaches require the sharp features to be specified or detected in advance [7], [8], [9], which is usually difficult and error-prone.

We propose an approach that controls both the approximation error and element quality simultaneously. Our approach only requires the user to specify the error-bound threshold δ, the desired minimal angle θ and an upper bound on the mesh complexity N, measured as the number of vertices. In an initial phase our algorithm coarsens the mesh as much as possible while respecting the error-bound δ. It then iteratively improves the minimal angle of the mesh until the desired bound for θ or N is met. Since all atomic operations respect the error-bound δ, the result is guaranteed to satisfy geometric fidelity. In contrast, low mesh complexity and high quality of mesh elements are optimization goals, which depending on the model and the desired bounds might or might not be met. However, experiments show that given a reasonable vertex budget N and an angle bound $\theta \leq 35^\circ$ our algorithm is usually able to reach all three goals simultaneously. Moreover, our method preserves geometric features like sharp creases and ridges without explicitly specifying or detecting them.

It is inspired by the remeshing methods that proceed by applying a series of local operators such as edge split, edge collapse, edge flip, and vertex relocate [10], [11]. Contrary to the existing methods that apply these operations sequentially and globally, our core algorithm employs a dynamic priority

- *K. Hu and B. Benes are with Purdue University, 610 Purdue Mall, West Lafayette, IN 47907. E-mail: hukaimo02@gmail.com, bbenes@purdue.edu.*
- *D.-M. Yan is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. E-mail: yandongming@gmail.com.*
- *D. Bommes is with RWTH Aachen University, Schinkelstr 2, Aachen 52062, Germany. E-mail: bommes@aices.rwth-aachen.de.*
- *P. Alliez is with Inria Sophia-Antipolis-Mediterranee, 2004 route des Lucioles BP 93, Sophia-Antipolis Cedex 06902, France. E-mail: pierre.alliez@inria.fr.*

Manuscript received 14 Aug. 2016; revised 7 Nov. 2016; accepted 9 Nov. 2016. Date of publication 24 Nov. 2016; date of current version 27 Oct. 2017. Recommended for acceptance by T. Ju.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2016.2632720



(a) Input model. (b) $\theta = 0°$. (c) $\theta = 10°$. (d) $\theta = 20°$. (e) $\theta = 30°$. (f) $\theta = 35°$. (g) $\theta = 40°$.

Fig. 1. Examples of surface meshes generated with our approach. The input model (a) has 7.2k vertices. From (b) to (g) are the results with different minimal angle threshold θ. The blue histograms show the distribution of minimal angles of triangles and the red curves the corresponding approximation errors of these triangles. The error-bound threshold δ is set to 0.2 percent of the diagonal length of the input's bounding box $(\%bb)$.

queue to maintain all angles which are smaller than the user specified threshold, and then greedily applies local operators whenever an improvement of the mesh is possible. After the initial coarsening our method only improves the mesh in local regions where the element quality is poor, and thus modifies the model as little as necessary with respect to the minimal angle threshold θ and the error-bound constraint δ.

Since we apply local operators directly on the mesh, without using any surface parameterizations or density functions, our algorithm is naturally suitable for high genus, irregular, and multi-component inputs, as well as meshes with very high/low resolutions. In summary, we claim the following contributions:

- A surface remeshing algorithm with minimal angle improvement based on applying local operators, which bounds the "worst element quality" (Section 3).
- A reliable and efficient local error update scheme based on approximated symmetric Hausdorff distance, which bounds the geometric fidelity (Section 4).
- A feature intensity function designed for vertex position relocation, which enables implicit feature preservation and supports geometric fidelity (Section 5).

2 Related Work

The variety of applications leads to a large number of different remeshing techniques. We restrict the discussion to the most relevant aspects of our algorithm, i.e., high-quality remeshing, error-driven remeshing and feature-preserving remeshing. For a more complete discussion we refer the reader to the survey [12].

High Quality Remeshing. Is typically based on sampling and Centroidal Voronoi Tessellation (CVT) optimization [13]. Early approaches apply 2D CVT in a parametric domain [3], [7], [14], [15], [16], [17]. Instead of CVT optimization, Vorsatz et al. [18] utilize a partial system approach in the parametric domain. In general, parametrization-based methods suffer from the additional distortion of the map and the need to stitch parameterized charts for high genus surfaces. Valette et al. [4] perform a discrete version of CVT directly on the input surface. However, the resulting mesh quality can be poor due to the inexact computation. Yan et al. [8], [19], [20] avoid the parameterization by computing the 3D CVT restricted to the surface. Additionally, they proposed blue-noise remeshing techniques using adaptive maximal Poisson-disk sampling [9], [21], farthest point optimization [22], and push-pull operations [23], which improve the element quality as well as introducing blue-noise properties. However, these approaches still suffer from common limitations, e.g., geometric fidelity and the minimal angle cannot be explicitly bounded. Moreover, sharp features must be specified in advance.

Another way to avoid the stitching problem is to operate directly on the surface mesh [24], [25]. An efficient isotropic approach proposed by Botsch and Kobbelt [11] takes an edge length as input and repeatedly splits long edges, collapses short edges, equalizes vertex valences and relocates vertex positions until all edges are approximately of the specified target edge length. To extend this work to an adaptive version, Dunyach et al. [26] replace the constant target edge length with an adaptive sizing field that is sensitive to local curvatures. Since this kind of methods requires neither surface parameterization nor density functions, they are easy to implement, robust for high genus inputs, and efficient for real-time applications. Our method falls into this category. However, we enrich the local operators and apply them in a more selective manner in order to obtain guarantees on the geometric fidelity as well as higher-quality results and implicit feature preservation.

Error-Driven Remeshing. Amounts to generating a mesh that optimizes the tradeoff between geometric fidelity and mesh complexity. Cohen-Steiner et al. [5] propose an error-driven clustering method to coarsen the input mesh. They formulate the approximation problem as a variational geometric partitioning problem, and iteratively optimize a set of planes using Lloyd's iteration [27] to minimize a predefined approximation error.

The mesh simplification techniques are similar to error-driven remeshing in some way. Garland and Heckbert [28] use iterative contractions of vertex pairs to simplify models and maintain surface approximation error based on quadric error metrics. Borouchaki and Frey [29] define a fidelity metric named Hausdorff envelope, and simplify and optimize the reference mesh that stays inside the tolerance volume. While they consider the geometric fidelity and element quality simultaneously, nothing is done to improve the worst element quality. Our method guarantees the worst element quality by improving the minimal angle and keeping the mesh complexity as low as possible, with respect to a given error-bound. Based on the concept of tolerance volume, Mandad et al. [6] propose an isotopic approximation method. Their algorithm generates a surface triangle mesh guaranteed to be within a given tolerance

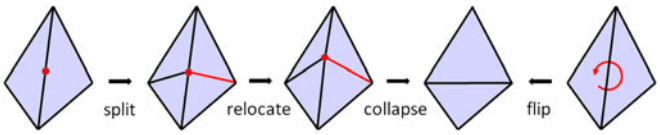


Fig. 2. Common local operators [1].

volume. However, they generate the mesh from scratch, and mainly focuse on mesh complexity rather than element quality. Instead, we strive for good balances between mesh complexity and element quality with a given input.

Feature Preservation. Is crucial in remeshing. However, automatically identifying sharp features on a surface mesh is a difficult problem that depends both on the local shape and on the global context and semantic information of the model. This makes feature detection a strongly ill-posed problem. A wide range of approaches address this problem [30], [31], [32]. However, none of them works reliably for all kinds of meshes. Most remeshing algorithms avoid this problem by assuming that the features have been specified in advance [8], [9], [19], [21], [22], [33]. Some remeshing techniques try to preserve features implicitly [34], [35]. Vorsatz et al. [36] first apply a relaxation in the parameter domain, and then snap the vertices to feature edges and corners. Since they separate remeshing and feature-snapping, the resulting mesh quality near sharp features might be poor. Valette [4] alleviates this issue by embedding the Quadric Error Metric approximation (QEM) criterion inside the CVT optimization. However, the performance of their cluster-based method is highly dependent on the quality of the input, and the sharp features might not be well preserved when users specify a small vertex budget. Jakob et al. [37] propose a general framework for isotropic triangular/quad-dominant remeshing using a unified local smoothing operator, in which the edges naturally align to sharp features. However, little attention is paid on the approximation error and element quality. We address this problem by optimizing the element quality explicitly in combination with implicit feature preservation based on the new defined feature intensity functions.

3 Algorithm Overview

Given a two-manifold triangular mesh M_I, the goal of our algorithm consists in finding an improved surface mesh M_R with approximation error below δ, minimal interior angle above θ and mesh complexity below N. The main idea is to transform the mesh by a series of discrete local operators, e.g., those illustrated below:

In our algorithm, only *edge collapse*, *edge split* and *vertex relocation* operators are employed. Edge flips are implicitly performed as a combination of an edge split followed by an edge collapse (cf. Fig. 2). This convention not only lowers the combinatorial complexity of all operators but is also advantageous for our implicit feature preservation since edge flips tend to destroy sharp creases and would require additional nontrivial checks.

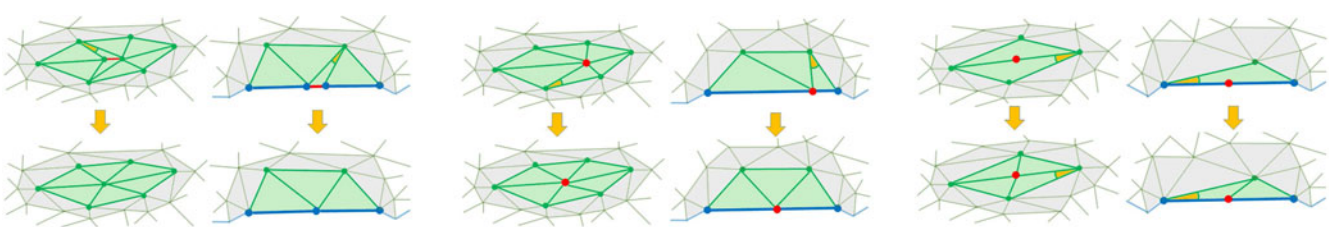
The remeshing algorithm is designed to perform local operators in a conservative manner. More specifically, a local operator is only executed if it respects the approximation error bound δ, does not introduce new interior angles below the current minimal angle θ_{min} and maintains the two-manifoldness of the mesh. This behavior can be interpreted as a conservative greedy algorithm, which in each step identifies the most promising operation that improves the result (either coarsens or improves angles), while never leaving the feasible set of meshes with approximation error below δ. Note that since pure topological edge collapses and edge splits improve the mesh quality very rarely, these operations are always combined with vertex position optimization.

The most crucial design choices of the algorithm are the scheduling of different operators and the efficient modeling and handling of approximation error queries (cf. Section 4). The algorithm passes through three different stages:

The initial phase concentrates on coarsening and runs a mesh simplification, which solely performs edge collapses.

The second phase then tries to lift the minimal interior angle above the user-provided bound θ. For this task we devised three different processes, which are tried subsequently, as illustrated in Fig. 3. Since an edge collapse reduces the complexity, it would be the best way to improve the minimal angle. If no edge collapse is possible, we try to improve the minimal angle by relocating one of the triangle's vertices. If vertex relocation also fails, edge splits are considered as the last option as they increase the mesh complexity. While edge splits do not directly improve the minimal angle, they are crucial to enrich the local mesh connectivity in order to enable improvements in subsequent steps. We apply an approach similar to the longest-side propagation path [38], described in more detail in Section 3.2.

The third stage of our algorithm freezes the mesh connectivity and concentrates on global vertex relocation. This stage is designed to improve the *average* quality of the mesh elements, while not violating the user-specified bounds. The pseudocode of our remeshing algorithm is shown in Algorithm 1.



(a) Edge collapse. (b) Vertex relocation. (c) Edge split.

Fig. 3. Local operators. The local patches before/after applying the operators are shown in the first/second rows, respectively. The inner patch (green) contains the facets that will be directly affected by the local operators, and the outer patch (gray) includes the facets who share vertices with facets of the inner patch. In each sub figure, the left shows the inner case while the right shows the boundary case, in which the boundary edges and vertices are depicted in blue. We depict the current minimal angle θ_{min} in yellow.

Algorithm 1. $Remeshing(M_I, M_R, \delta, \theta, N)$

Input: M_I {Input mesh}
$\delta > 0$ {Error-bound threshold}
$\theta \geq 0$ {Minimal angle threshold}
$N > 0$ {Desired mesh complexity}
Output: M_R {Remeshing result}

$M_R \leftarrow M_I$;
$InitialMeshSimplification(M_I, M_R, \delta)$;
fill Q with angles of M_R smaller than θ; Q is a dynamic priority queue
$\#V \leftarrow$ the number of vertices in M_R;
while $Q \neq \emptyset$ **and** $\#V < N$ **do**
  $\theta_{min} \leftarrow Pop(Q)$;
  $GreedyImproveAngle(\theta_{min}, M_I, M_R, \delta, \theta)$;
  update Q and $\#V$;
end while
$FinalVertexRelocation(M_I, M_R, \delta, \theta)$;

In the following we provide more details on the three phases of initial mesh simplification, greedy angle improvement and vertex relocation.

3.1 Initial Mesh Simplification

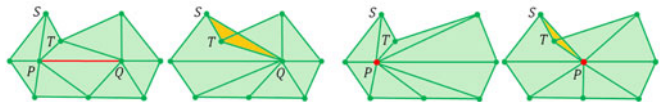
The input mesh is often dense such that the mesh complexity can be significantly reduced without violating the approximation error bound. While such a mesh simplification could potentially also be done on the fly, a specialized pre-process turns out to be more efficient. The goal of this step consists in finding a mesh, which is significantly coarser and offers a good starting point for the second phase of element quality improvement. Simplification is achieved through iteratively collapsing edges and relocating related vertices as long as the approximation error does not exceed δ. Short edges, or those opposite to small angles are likely to lower the mesh quality and are consequently collapsed first. This is achieved through using a priority queue sorted by increasing values of the function $l(h) \cdot (\theta_1(h) + \theta_2(h))/2$, where $l(h)$ denotes the length of halfedge h and $\theta_i(h)$ denote the two angles opposite to h and h's opposite halfedge. The pseudocode of this initial mesh simplification is shown in Algorithm 2.

Algorithm 2. $InitialMeshSimplification(M_I, M_R, \delta)$

fill Q with all halfedges;Q is a dynamic priority queue
while $Q \neq \emptyset$ **do**
  $h \leftarrow Pop(Q)$;
  if $CollapseAndRelocateImproves(h, \delta, M_I, M_R)$ **then**
    $CollapseAndRelocate(h, \delta, M_I, M_R)$;
  end if
  update Q;
end while

3.2 Greedy Improvement of Angles

The second phase is designed to improve the mesh quality by iteratively increasing the smallest angle in the mesh, until the desired angle bound θ is satisfied or the complexity limit N is reached. Our approach repeats the following process: We simulate a potential operation, test whether the resulting mesh improves and only in this case perform the candidate operation. The mesh improvement test additionally contains several important validity checks.



(a) Edge collapse. (b) Vertex relocate.

Fig. 4. Examples of fold-overs in 2D. (a) An edge collapse might flip triangles by pullingän edge SP over a vertex T; (b) A similar problem might occur in a vertex relocation. In 3D, fold-overs occur if the triangle normal before and after applying an operation have opposite orientations.

Mesh Improvement Test. We simulate each potential operation and measure if the following constraints are satisfied:

- *Topology*. For edge collapses topology changes are prevented by checking the *link condition* [39].
- *Geometry*. The operator should not create fold-overs by flipping the orientation facets (cf. Fig. 4).
- *Fidelity*. The approximation error between M_R and M_I should remain below δ (cf. Fig. 6).

In order to improve the minimal angle, we greedily apply operators that successfully pass the mesh improvement test. The operators are tested in the following order: (i) edge collapses, (ii) vertex relocations and (iii) edge splits. The pseudocode of the greedy improvement is shown in Algorithm 3.

Algorithm 3. $GreedyImproveAngle(\theta_{min}, M_I, M_R, \delta, \theta)$

Input: θ_{min} {Angle requiring improvement}
M_I {Input mesh}
$\delta > 0$ {Error-bound threshold}
$\theta \geq 0$ {Minimal angle threshold}
Output: M_R {Locally improved mesh}

$h \leftarrow$ halfedge opposite to θ_{min};
if $CollapseAndRelocateImproves(h, \theta_{min}, \theta, \delta, M_I, M_R)$ **then**
  $CollapseAndRelocate(h, \theta_{min}, \theta, \delta, M_I, M_R)$;
  return;
end if
let v_o, v_s and v_e be h's opposite, start and end vertex;
for $v \in \{v_o, v_s, v_e\}$ **do**
  if $RelocateImproves(v, \theta_{min}, \theta, \delta, M_I, M_R)$ **then**
    $Relocate(v, \theta_{min}, \theta, \delta, M_I, M_R)$;
    return;
  end if
end for
$h_l \leftarrow LongestSidePropagation(h)$;
if $SplitAndRelocateIsValid(h_l, \theta, \delta, M_I, M_R)$ **then**
  $SplitAndRelocate(h_l, \theta, \delta, M_I, M_R)$;
end if

Longest-Side Propagation Path. If neither edge collapse nor vertex relocation is possible, we search the longest edge h_l in the neighborhood [38] and then split it. Starting from an edge e, we iteratively move on to the longest edge of the neighboring two triangles until no further enlargement is possible or until we hit the boundary. Though the edge split operator does not increase θ_{min} directly, it modifies the local connections such that θ_{min} can be improved in later iterations (cf. Fig. 3c). While there might exist other ways for local connectivity modification, experiments show that this strategy works well in our algorithm.

3.3 Final Vertex Relocation

To achieve a better overall element quality, in the third phase we perform a series of vertex relocations until the

angles can no longer be improved. In contrast to the second phase which only focuses on specific regions where angle improvement is required, the third stage optimizes all vertex locations. We maintain the relocation candidate set in a queue, which is initialized with all vertices. Whenever a vertex is relocated, all its neighbors are added to the queue, since a change in the neighborhood might enable further improvements. The pseudocode of the corresponding function is shown in Algorithm 4.

Algorithm 4. $FinalVertexRelocation(M_I, M_R, \delta, \theta)$

```
fill Q with all vertices of M_R;Q is a FIFO queue
while Q ≠ ∅ do
  v ← Pop(Q);
  if RelocateImproves(v, θ_min, θ, δ, M_I, M_R) ≥ Δθ then
    Relocate(v, θ_min, θ, δ, M_I, M_R);
    add neighbors of v to Q
  end if
end while
```

Optimal vertex positions would be found if no vertex could be relocated to a better position anymore. However, we empirically restrict the angle improvement $\Delta\theta = 0.1°$(cf. Section 6.4), since afterward there are usually no significant changes anymore.

4 Error Metric

4.1 Hausdorff Distance

We use the Hausdorff distance to measure the approximation error between M_R and M_I. Let $d(p, q)$ denote the euclidean distance between two points p and q in 3D space. The distance of a point p to a surface M is defined as the shortest distance between p and any point of M

$$d(p, M) = \min_{q \in M} d(p, q). \tag{1}$$

The one-sided Hausdorff distance from a source surface M to a target surface T is defined as the maximum of all such point to surface distances

$$d_h(M, T) = \max_{p \in M} d(p, T). \tag{2}$$

The one-sided Hausdorff distance is in general not symmetric, i.e., $d_h(M, T) \neq d_h(T, M)$. It is easily possible to construct counter-intuitive cases where $d_h(M, T) = 0$ but $d_h(T, M)$ is arbitrarily large.[1]

The two-sided Hausdorff distance [40] between M and T resolves this issue by symmetrization

$$d_H(M, T) = \max\{d_h(M, T), d_h(T, M)\}. \tag{3}$$

4.2 Approximating d_H with Stratified Sampling

The exact evaluation of the two-sided Hausdorff distance is computationally very expensive [41]. However, by careful surface sampling in combination with local updates of shortest point-to-surface links it is possible to obtain an efficient yet sufficiently accurate approximation, as discussed next.

1. Choose T as a sphere of radius r and M as a hemisphere subset. Then with $r \to \infty$ also $d_h(T, M) \to \infty$.

Assume that M is sampled by a point set $S_M \subset M$. Then the one-sided Hausdorff distance can be approximated by

$$d_h(M, T) \approx \max_{a \in S_M} d(a, T). \tag{4}$$

By additionally sampling T we obtain an approximation of the two-sided Hausdorff distance

$$d_H(M, T) \approx \max\left(\max_{a \in S_M} d(a, T), \max_{b \in S_T} d(b, M)\right), \tag{5}$$

with S_T being a set of point samples on T. Note that our approximation still measures the exact distance from sample points to the complete surface, which provides significantly higher accuracy than a point cloud distance $d_H(S_M, S_T)$. Moreover, it ensures that we strictly underestimate the real distance. Following the triangle inequality, the approximation error of our sampled Hausdorff distance is bounded by $\max\{d_h(M, S_M), d_h(T, S_T)\}$, i.e., the maximum gap between sample points. Consequently, in order to guarantee a good approximation, we target a uniform sampling of the surfaces. However, for piecewise linear triangle meshes the maximal distance often occurs at creases or corners, i.e., at mesh edges or vertices. This two observations motivate our *stratified sampling* approach, which is uniform on faces but additionally adds samples on edges and vertices, as a kind of greedy importance sampling.

Sampling the Facets. Instead of uniformly sampling the complete surface, we empirically found that better results can be obtained by uniformly sampling per triangle. In this way we obtain an automatic adaptivity, since more samples are available in structurally complex and highly curved areas, where also more triangles are necessary to describe such shape. Since our meshes are often highly non-uniform, we add a slight local smoothing of the sampling density by

$$n(f_i) = n_f \cdot \frac{1 + |\mathcal{N}_{f_i}|}{1 + \sum_{f_j \in \mathcal{N}_{f_i}} \frac{\mathcal{A}_j}{\mathcal{A}_i}}, \tag{6}$$

where n_f is the average number of samples per facet specified by the user, $\mathcal{N}_{f_i}$ are the neighbor facets that share vertices with f_i, and $\mathcal{A}_i$ is the area of facet f_i. Choosing a large n_f offers a tighter approximation of the Hausdorff distance, however, resulting in high computational complexity. In our experiments, we found that $n_f = 10$ yields good tradeoffs between efficiency and effectiveness (cf. Fig. 12).

To distribute the samples evenly on a triangle f_i, we first generate $n(f_i)$ samples randomly, and then perform a Lloyd relaxation process onto a bounded Voronoi Diagram (BVD) [42] (Fig. 5b). Usually, five iterations are sufficient to generate quasi-uniformly distributed samples on facets. *Sampling the Edges*. By counting the number of incident Voronoi cells to an edge we first estimate the local sampling density. The resulting number of samples is then evenly distributed along the edge (Fig. 5c).

Sampling the Vertices. The position of a vertex is simply regarded as its own vertex sample.

By setting S_I and S_R as the stratified samples on M_I and M_R, the Hausdorff distance between M_I and M_R is approximated using Eq. (5), as illustrated by Fig. 6.

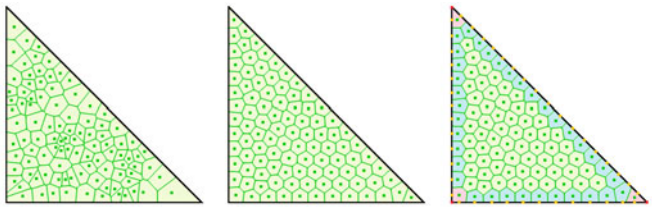


Fig. 5. Stratified sampling process. (a) Initial facet samples; (b) Optimized facet samples with BVD; (c) Edge samples. The facet, edge, and vertex samples are rendered in green, yellow and red, respectively.

4.3 Local Update Scheme

Each operator of our remeshing algorithm only changes a local area $L \subset M_R$ of the target mesh M_R (cf. Fig. 3 green area). Hence, it is possible to rapidly compute the Hausdorff distance for the updated mesh. In general both directions of the Hausdorff distance change and require an update.

Updating $\mathbf{d_h(M_R, M_I)}$. First of all notice that the one-sided Hausdorff distance can be decomposed w.r.t. to a modified local area $L \subset M_R$ into

$$d_h(M_R, M_I) = \max\{d_h(L, M_I), d_h(\bar{L}, M_I)\},$$

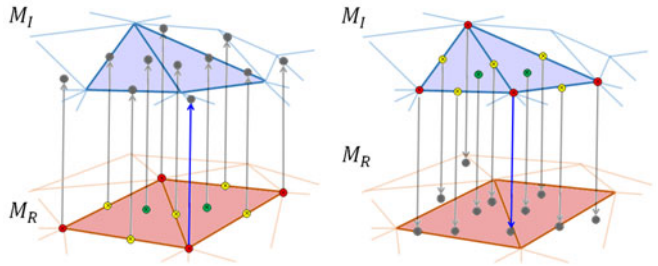
with $M_R = L \cup \bar{L}$ being a partition of M_R. Since $\bar{L}$ is unchanged, checking $d_h(L, M_I) \leq \delta$ is sufficient to verify that the modification does not violate the approximation error bound δ. Thus, we first re-sample the modified local area L with stratified samples S_L, and then efficiently evaluate the approximated Hausdorff distance $d_h(S_L, M_I)$ with a pre-computed axis aligned bounding box tree [43] for the static mesh M_I.

Updating $\mathbf{d_h(M_I, M_R)}$. Checking the opposite direction is more difficult for two reasons. First, since M_R changes, we cannot simply pre-compute a static search tree. Second, decomposing the Hausdorff distance in the second argument is more intricate. In addition to M_R, M_I must also be decomposed correctly. More specifically, we have

$$d_h(M_I, M_R) \approx d_h(S_I, M_R) = \max\{d_h(S_I^L, L), d_h(S_I^{\bar{L}}, \bar{L})\},$$

where $S_I = S_I^L \cup S_I^{\bar{L}}$ is a partitioning of samples on M_I into those closer to L and $\bar{L}$ (the rest of M_R) respectively.

Identifying the correct partitioning of samples S_I would in general require global tests and is thus time consuming. Therefore, in order to enable a rapid update we only approximate this decomposition by tracking the history of samples in S_I. The key idea is to store for each sample $s_j \in S_I$ a link to its closest triangle $t_j \in M_R$ (Fig. 6). After a local modification of M_R these links will typically change in the vicinity of the modified area L. Thus, in order to avoid global checks, we only update the links of samples connecting to a region L^+, which enlarges L by an additional ring of neighboring triangles (cf. Fig. 3 gray area). These new links are efficiently found by constructing and querying an axis aligned bounding box tree of the local area L^+. Note that the resulting approximated Hausdorff distance is a strict overestimator because shorter links to $\bar{L}^+$ might exist that are not investigated by our approximation. A positive side effect of our localized update scheme is that links cannot jump between geodesically far away regions, which could cause topologically inconsistent links.



(a) Stratified samples on M_R and their shortest links to M_I. (b) Stratified samples on M_I and their shortest links to M_R.

Fig. 6. Hausdorff distance approximation with stratified samples. Samples on vertices, edges, and faces are rendered in red, yellow, and green respectively. We use the circles centered with crosses to indicate the samples on one mesh, and the circles centered with dots to indicate their closest points on the other mesh. The approximated one-side Hausdorff distances are shown as the blue links, and the overall approximated Hausdorff distance is simply the length of the longer one.

5 Implicit Feature Preservation

A clean representation of geometric features as for instance sharp creases is important in many applications ranging from visualization to simulation. In a polygonal mesh, we distinguish between vertex and edge features. Vertex features include tips, darts, cusps and corners, while edge features are either creases or boundaries. Typical examples are depicted in Fig. 7.

Instead of requiring an explicit tagging of features as most remeshing approaches, e.g., [8], [9], [19], [22], [33], our goal is to implicitly preserve features. This not only releases users from the time-consuming burden of manual tagging but moreover often enables the recovery of features that were lost, e.g., through inappropriate sampling by a 3D laser scanner.

In principle feature preservation could simply be a byproduct of approximation error minimization since incorrectly meshing features usually induces a large approximation error. Nevertheless, there are several reasons why special care is still required. First of all, the minimization of the approximation error is a non-convex problem such that bad initializations might lead to low-quality local minima, not well representing features. Moreover, anticipating feature locations and placing vertices accordingly will speed up the overall method. In our method special feature handling is done whenever vertices are either newly placed or relocated, i.e., during (i) edge collapse, (ii) edge split and (iii) vertex relocation. Since robust and automatic feature detection is a difficult yet unsolved problem, we rely on a softer identification of features by means of a feature intensity function defined at vertices of the mesh.

5.1 Feature Intensity

Feature vertices can be characterized by large *Gaussian curvature* $\mathcal{K}(v)$, which in the discrete setting is identical to the angle defect, i.e.,

Fig. 7. Vertex features: (a) Tips, (b) Darts, (c) Cusps and (d) Corners are feature vertices who are adjacent to zero, one, two and three sharp creases respectively.

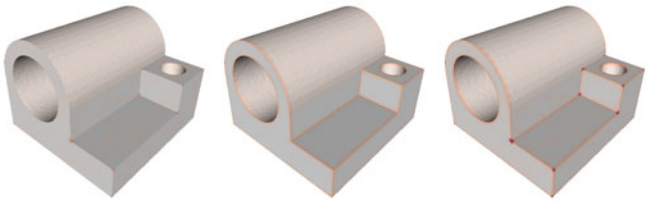


Fig. 8. From left to right Gaussian curvature $|\mathcal{K}|$, feature edge intensity $\mathcal{E}$ and combined feature intensity $\mathcal{F}$, with higher intensities in red.

$$\mathcal{K}(v) = \begin{cases} \pi - \theta_{sum}(v) & \text{v is on a boundary,} \\ 2\pi - \theta_{sum}(v) & \text{otherwise,} \end{cases} \tag{7}$$

where $\theta_{sum}(v)$ is the sum of interior angles adjacent to v.

Feature edges are characterized by large dihedral angles. Accordingly, for a vertex we define the *feature edge intensity* $\mathcal{E}(v)$ to be the maximal unsigned dihedral angle of an edge adjacent to v

$$\mathcal{E}(v) = \max_{e \in \mathcal{N}_e(v)} |\mathcal{D}(e)|, \tag{8}$$

where $\mathcal{N}_e(v)$ are the edges adjacent to v and $\mathcal{D}(e)$ is the dihedral angle at e.

Finally, the *feature intensity* $\mathcal{F}(v)$ is defined as the combination

$$\mathcal{F}(v) = (\tau(|\mathcal{K}(v)|) + 1) \cdot (\tau(\mathcal{E}(v)) + 1) - 1,$$

with the transfer function $\tau(x) = \min\{\pi, 2 \cdot x\}$, which rescales values by a factor of 2 and clamps them at π. Thus, the feature intensity is a value between 0 and $((\pi+1)^2 - 1)$ and corresponds to a *logical or* whenever one of the individual intensities vanishes. Fig. 8 shows an examples of the three fields $|\mathcal{K}|$, $\mathcal{E}$ and $\mathcal{F}$.

5.2 Feature-Sensitive Vertex Relocation

In our remeshing algorithm each local operation is combined with a subsequent relocation of the modified vertex in order to minimize the approximation error. This relocation of a single vertex is done in two stages. First a feature-sensitive initialization, specifically adapted to the local operation, and then a nonlinear minimization of the two-sided Hausdorff distance. A careful initialization of vertex positions is important to avoid poor local minima of the non-convex Hausdorff energy and additionally increases performance.

Edge Collapse Initialization. During an edge collapse of edge e_{ij} the two vertices v_i and v_j merge into a new vertex v_m and a new position for v_m must be specified. Common feature-sensitive approaches use error quadrics [4], [28]. However, since we anyway perform a more accurate approximation error-driven relocation subsequently, a simpler and faster position initialization based on our feature intensity is sufficient. We distinguish two cases. If the edge is incident to a single strong feature we want that v_m snaps onto this feature, meaning that the default behavior is to snap to the vertex v_k with higher feature intensity, i.e., $v_k = \arg\max_{v \in \{v_i, v_j\}} \mathcal{F}(v)$. An unclear situation arises, when both feature intensities are similar, i.e., $\mathcal{F}(v_i) \approx \mathcal{F}(v_j)$. This happens either in regions without features or for edges along a crease, where it is reasonable to initialize v_m to the edge midpoint. We decide for the midpoint initialization based on a parameter ω, whenever

$$|\mathcal{F}(v_i) - \mathcal{F}(v_j)| < \omega \cdot \max(\mathcal{F}(v_i), \mathcal{F}(v_j)),$$

with $\omega = 0.15$ in all our examples.

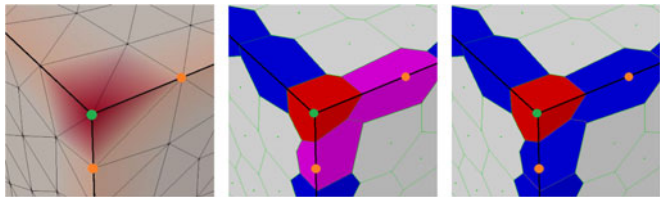


Fig. 9. Visualization of $\mathcal{F}(v)$ in (a). The color of v's Voronoi cell represents the number of important neighbors, with red=0, blue=2, purple=3 and gray=degree(v) is shown in (b) and (c). A wrong classification based on solely feature intensity in (b) is corrected by additionally checking importance of edges in (c).

Edge Split Initialization. Since an edge split does not change the geometry we simply always initialize the new point as the edge midpoint.

Vertex Relocation Initialization. Without topological mesh modifications (edge collapse or split), a vertex relocation is initialized in the following way. The goal is to anticipate a good relocation position for v while preserving feature corners and creases. Therefore, we first classify the vertex v as either being (i) a feature vertex, (ii) a crease vertex or (iii) a smooth vertex. If v is a feature vertex it simply remains at its position. If it is on a crease, we relocate v to the CVT barycenter of its two neighboring crease vertices. And only if it is a smooth vertex we relocate it to the CVT barycenter of all one-ring neighbors. The classification is done by counting how many neighbors v_i are of similar or higher importance as v, i.e., if $\mathcal{F}(v_i) \geq \zeta \cdot \mathcal{F}(v)$ for a tolerance parameter $\zeta \in (0, 1)$. In this way v is classified as (feature vertex/crease vertex/smooth vertex), depending on whether (none/two/all) of its neighbors are of similar feature intensity. Unclear cases where k out of n neighbors are of similar importance are classified towards the closest case, meaning as a crease vertex if k is closer to 2 or as a smooth vertex if k is closer to n. Wrong classifications where two crease vertices are connected by a non-crease edge can be avoided by only counting important neighbors that are connected by an important edge with $|D(e_{v,v_i}| + 1 \geq \zeta \cdot (\mathcal{E}(v) + 1)$, as illustrated in Fig. 9.

The parameter ζ controls the feature classification. For higher values of ζ more vertices are implicitly treated as features and thus prevented from free movement. Fig. 10 shows an example classification for ζ varying between 0.3 and 0.7. In our experiments the default is $\zeta = 0.5$

Nonlinear Hausdorff Distance Minimization. After one of the former initializations is done, we further optimize the position of a vertex v by directly minimizing the approximate two-sided Hausdorff distance of Section 4.2. This optimization is highly nonlinear, since changing the position of v changes the samples S_R of the modified mesh M_R as well as the links from the input mesh samples S_I to M_R. We perform an optimization similar to the Hausdorff distance minimization technique proposed by Winkler et al. [44]. However, we improve the feature-sensitivity by adjusting the weighting scheme with our feature intensity function $\mathcal{F}$, as detailed next.

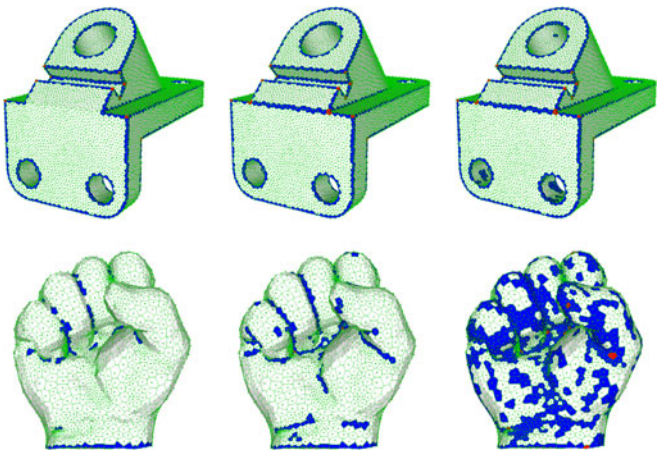


Fig. 10. Influence of the classification tolerance ζ. The Voronoi cells of vertices color code the classification with red=feature vertex, blue=crease vertex and white=smooth vertex. We show the Anchor model with obvious features (top) and the Hand model without obvious features (bottom).

We wish to relocate v in order to minimize the two-sided Hausdorff distance between the local area $L \subset M_R$ consisting of all one-ring triangles of v and the subregion of M_I with links into L, referred to as $M_{I\to L}$. Assume that the corresponding subsets of samples for our Hausdorff distance approximation are $S_L \subset S_R$ and $S_{I\to L} \subset S_I$. Then according to Motzkin and Walsh's theorem [45] there exist weights w_i and $\hat{w}_i$ such that minimizing Eq. (5) w.r.t. the center vertex v is equivalent to minimizing

$$\sum_{a_i \in S_L} w_i |a_i(v) - \hat{a}_i|^2 + \sum_{b_i \in S_{I\to L}} \hat{w}_i |\hat{b}_i(v) - b_i|^2, \tag{9}$$

where $\hat{a}_i \in M_{I\to L}$ and $\hat{b}_i(v) \in L$ are the closest points to a_i and b_i respectively. Freezing the closest point pairs and assuming a linear barycentric relation

$$a_i(v) - \hat{a} = \alpha_i v + \beta_i d_i + \gamma_i e_i - \hat{a} = \alpha_i v - p_i,$$

with constant $p_i = \hat{a} - \beta_i d_i - \gamma_i e_i$ and similarly expressed $\hat{b}_i(v) - b_i = \hat{\alpha}_i v - \hat{p}_i$, the optimal position v^* can be computed analytically as

$$v^* = \frac{\sum_{a_i} w_i \alpha_i p_i + \sum_{b_i} \hat{w}_i \hat{\alpha}_i \hat{p}_i}{\sum_{a_i} w_i \alpha_i^2 + \sum_{b_i} \hat{w}_i \hat{\alpha}_i^2}. \tag{10}$$

To find optimal weights w_i and $\hat{w}_i$, Winkler et al. [44] use Lawson's algorithm [46] with iterative updates of the form

$$w_i^{(k+1)} = w_i^{(k)} \cdot d(a_i^{(k)}, \hat{a}_i^{(k)}), \tag{11}$$

where $d(a_i^{(k)}, \hat{a}_i^{(k)})$ is the Euclidean distance of the closest-point pair $(a_i, \hat{a}_i)$ after the kth iteration, and initialization $w_i^{(0)} = 1$. The idea behind this scheme is that samples with larger distances get a higher weight in the next iteration. Based on our feature intensity function and a sample density estimation, the weight update can be improved to

$$w_i^{(k+1)} = w_i^{(k)} \cdot d(a_i^{(k)}, \hat{a}_i^{(k)}) \cdot \mathcal{V}(a_i^{(k)}) \cdot \mathcal{F}(a_i^{(k)}), \tag{12}$$

where $\mathcal{V}(a_i)$ is the Voronoi cell area of sample a_i, and $\mathcal{F}(a_i)$ is the linearly interpolated feature intensity value at a_i. Fig. 11 illustrates the additional weighting factors. Advantages are a better feature preservation, improved robustness w.r.t. non-uniform sampling and faster convergence.

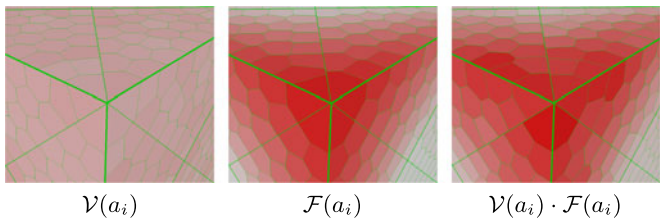


Fig. 11. Sample weights of a cube corner, with the integration of $\mathcal{V}(a_i)$ and $\mathcal{F}(a_i)$. Higher intensity of red highlights larger weight values.

Each time a local operator is applied, we iteratively optimize the new position of the effected vertex by minimizing Eq. (9) using the new weighting defined in Eq. (12). The optimization procedure is similar to the Expectation-maximization (EM) algorithm: in each iteration, we first calculate the optimal position v^* of vertex v using Eq. (10), and then move v to $v + \lambda(v^* - v), \lambda \in (0, 1]$ with default $\lambda = 0.9$ and update the closest point pairs. In practice, a few iterations usually suffice to get very close to the optimum.

6 Experiments and Discussions

We implemented our approach in C++ and tested on a 64 bit Windows 8.1 operating system. The CGAL library [47] provided most of the basic data structures and operations. Timings for all the examples were conducted on a single 3.40 GHz Intel(R) Core(TM) i7-2600 K CPU with a 16 GB RAM. We provide next a complete evaluation of our algorithm and comparisons with state-of-the-art approaches.

6.1 Evaluation of the Local Error Update Scheme

For efficient local error update we use the axis aligned bounding box tree of CGAL. Since a global error update is too compute-intensive we only verify how the inner patch sizes and sampling densities affect the effectiveness and efficiency of the local error update scheme (Fig. 12). We find that even when the inner patch size is set to the one-ring (cf. Fig. 3), our local error update approach catches more than 99.9 percent of the globally nearest points. The accuracy increases little with higher sampling density and larger inner patch size. Consequently, we set the inner patch sizes as one-ring facets and the average sampling number per facet as ten in all experiments.

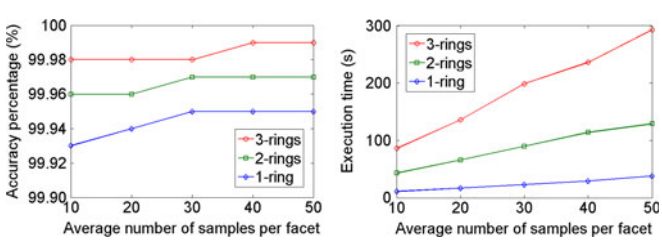


Fig. 12. The nearest point searching accuracy and execution time of our local error scheme, with respect to different sampling densities and inner patch sizes. In this experiment, $\delta = 0.2$ percent of the bounding box's diagonal length($\%bb$) and $\theta = 30^\circ$. The above data are the averages of 10 consecutive executions with the Rockerarm model (Table 2) as input.

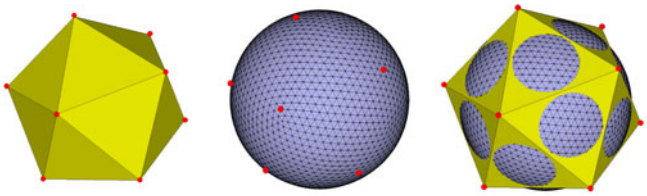


Fig. 13. Demonstration of interpolation and approximation. We use an icosahedron (a) to interpolate and approximate a sphere ((b) and (c)). In interpolation, the vertices of M_R are guaranteed to be on the surface of M_R, while in approximation, the error is minimized, regardless of whether the vertices of M_R are on the surface of M_I.

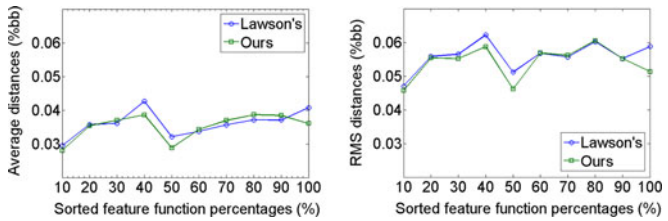


Fig. 14. Comparison of geometric fidelity between Lawson's weighing scheme [44] and ours(Eq. (12)). We sort the vertices in M_I according to their saliency function values in ascending order, and compute their average distance and RMS distance in each bin. In this experiment, $\delta = 0.2(\%bb)$, and $\theta = 30^\circ$. The above data are the averages of 10 consecutive executions with the Hand model (Fig. 10) as input.

6.2 Evaluation of Vertex Position Optimization

Solely applying the vertex position initialization makes most vertices of M_R stay near or on the surface of M_I. Though this quasi-interpolation preserves sharp features, the optimal geometric fidelity is usually achieved when M_R is an approximation of M_I. We visually demonstrate the difference between interpolation and approximation in Fig. 13. In the interpolation case, the Hausdorff distance (Hdist) [48] between the sphere and the icosahedron is $6.48(\%bb)$, and the root mean square (RMS) distance is $4.84(\%bb)$; while in the approximation case, the Hdist and RMS distances between them are $5.00(\%bb)$ and $1.09(\%bb)$, respectively.

However, the approximation might destroy features when minimizing the local sample pair distances [44]. Fig. 14 compares the average distance and RMS distance based on Lawson's weighting scheme (Eq. (11)) and our improved weighting scheme (Eq. (12)). Generally, our weighting scheme reduces the average distance and RMS distance about 2.3 and 3.1 percent respectively. However, it reduces the approximation error of vertices on sharp features about 11.8 and 12.8 percent, respectively. Therefore, applying the improved weighting scheme does not only reduce the approximation error, but also better preserves sharp features.

We extensively tested how the iteration count and relocation ratio (λ) affect the approximation error and execution time in the vertex position optimization procedure (in Fig. 15). We found that setting the iteration count to two and the relocate ratio to 0.9 yields the best balance between effectiveness and efficiency. This configuration is used in all our later experiments.

6.3 Evaluation of Initial Mesh Simplification

In general, applying Algorithm 2 further reduces 20 percent vertices on average, however, the execution time is 2-3 times slower. Fig. 16 shows the remeshing results of the Egea model with and without applying initial mesh simplification, and Table 2 further compares the differences (OUR versus OUR*). Usually, if users care more about the execution time than the mesh complexity, Algorithm 2 can be disabled. However, we enable Algorithm 2 by default for better balancing between mesh complexity, element quality and approximation error.

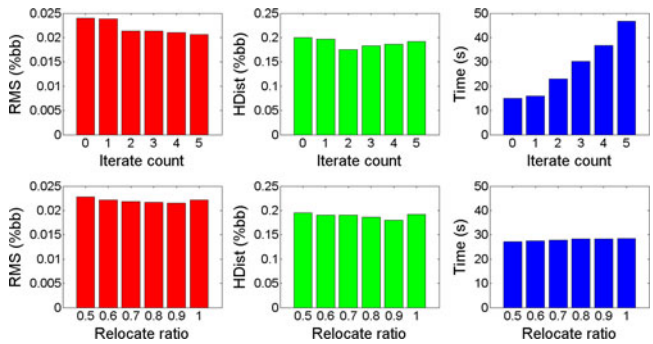


Fig. 15. Effectiveness of the iteration count and relocate ratio. For separability, RMS and Hdist bars are plotted in different scales. We set $\delta = 0.2(\%bb)$, and $\theta = 30^\circ$. The above data are the averages of 10 consecutive executions with the Homer model (Table 2) as input.

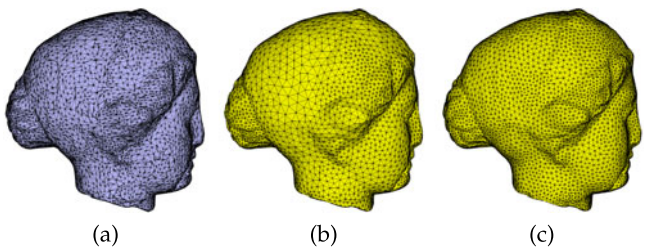


Fig. 16. Egea models with/without executing Algorithm 2. we set $\theta = 40^\circ$ and $\delta = 0.20(\%bb)$. The input (a) has 8.3k vertices. The result with Algorithm 2 enabled has 4.2k vertices (b), and spends 330 seconds; the result with Algorithm 2 disabled has 7.6k vertices (c), and spends 133 seconds.

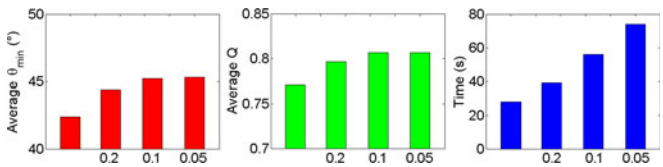


Fig. 17. The effectiveness of $\Delta\theta$ in Algorithm 4. In each sub figure, the first bar indicates the value when no final vertex relocation is applied. In this experiment, $\delta = 0.2(\%bb)$ and $\theta = 30^\circ$. The above data are the averages of 10 consecutive executions with the Helmet model (Table 2) as input.

6.4 Evaluation of Final Vertex Relocation

To measure how the overall element quality is improved by applying Algorithm 4, we introduce two new measurements: the first is the average minimal angle of all triangles in M_R and the second is the average value of triangle qualities defined as $Q_t = 2\sqrt{3}S_t/(p_t h_t)$ [49], where S_t is the area of triangle t, p_t the in-radius of t and h_t the length of the longest edge in t. We investigated the statistical element quality and the execution time with varying $\Delta\theta$ in Algorithm 4 (Fig. 17), and found when $\Delta\theta < 0.1^\circ$, the quality improvement is not significant. In our experiments, we set the default $\Delta\theta$ as 0.1 degree in Algorithm 4.

TABLE 1
Influence of θ and σ

θ	#V	Q_{min}	θ_{max}	**RMS(%bb)**	$V_{567}(\%)$	Time
0	*0.15 k*	0.040	174.4	0.043	76.5	1:42*
10	*0.20 k*	0.188	155.1	0.038	71.3	2:18
20	*0.26 k*	0.340	135.5	0.036	78.6	2:41
30	*0.40 k*	0.482	117.7	0.031	81.7	3:01
35	*0.73 k*	0.552	109.3	0.029	85.5	3:44
40	*2.8 k*	0.640	98.8	0.022	98.5	6:14

** This indicates the execution time of Algorithm 2.*
V_{567} is the percent of vertices with valences 5, 6 and 7. We set $\delta = 0.2(\%bb)$, and the Fandisk (Fig. 1a) is the input.

6.5 Influence of the Minimal Angle Threshold and the Mesh Complexity

Our algorithm produces results with either desired minimal angle bound or desired mesh complexity, depending on the user specified parameters. In order to get the desired minimal angle, N in Algorithm 1 should be set very large; otherwise, θ should be set very large (e.g., 60 degree). Usually, the larger the value θ or N is specified, the better element quality is achieved. We tested the remeshing results of the Fandisk model with θ varying from 0 to 40 degree and N varying from 0.15 k to 2.8 k, and depict the results in Fig. 1. The complete attributes are listed in Table 1. Note that in Fig. 1g, one vertex has been relocated a little away from the crease after refinement, such that the minimal angle is improved. This happens when optimizing a mesh with sharp features up to a high minimal angle threshold.

We compare our results with the state-of-the-art methods, and find only in a small portion of results presented in [19] and [9], that the minimal angles exceeds 35 degree (with maximum 38 degree). For most other methods, the minimal angles vary in the range $[25°, 35°]$. Contrary to the previous work, our method is able to generate results with minimal angles higher than 35 degree in all test cases. The complete comparison with the state-of-the-art methods is shown in Section 6.8.

6.6 Influence of the Approximation Error Threshold

We demonstrate the influence of δ in Fig. 18. The results indicate that δ does not influence the mesh complexity and the execution time significantly with a fixed θ value. However, two interesting phenomena are observed: 1) when $\theta \leq 35°$, the larger the θ value is, the lower the mesh complexity we achieve; however, when θ is set to 40 degree, both mesh complexity and execution time increase dramatically. This is because when θ is small, the edge collapse operator is preferentially applied, which increases the minimal angles while reduces the mesh complexity. However, when θ is large, more edge split operators are applied to modify local connections. 2) Within a fixed θ, when δ increases, the mesh complexity decreases slightly and smoothly, since the higher δ is set, the more edge collapses are possible.

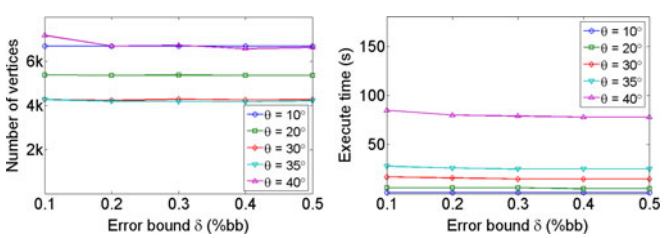


Fig. 18. Influence of the error-bound threshold δ. The above data are the averages of 10 consecutive executions with the Elephant model (Table 2) as input.

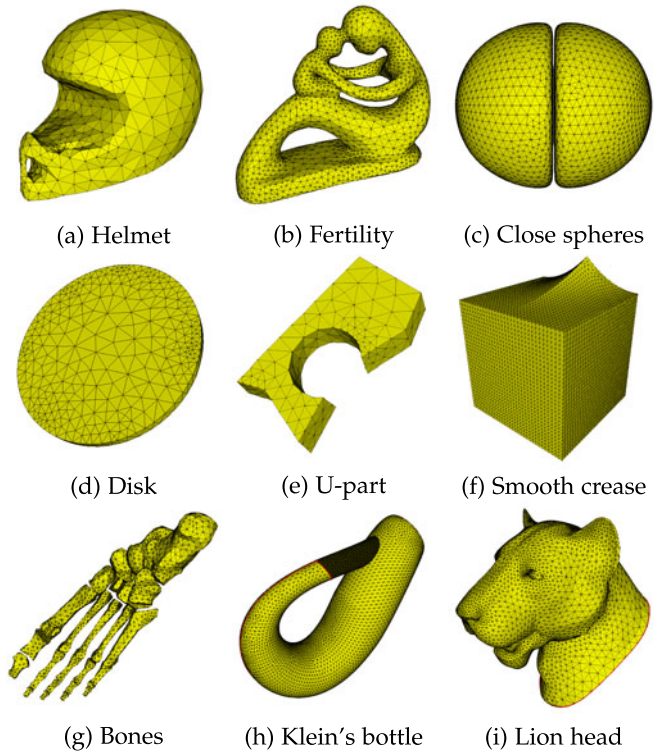


Fig. 19. Selected results. In this experiment, δ is set to $0.2\%(bb)$ and θ is set to 35 degree. In (h) and (i), the boundaries are rendered in red. The dark part of (h) means the triangle normals are inside.

We compare our results with those provided by Yan et al. [9], [19], [22]. The best approximation error is $0.10(\%bb)$, which is generated by [8] with the Homer model as input. However, it can not be explicitly controlled. For other methods, the approximation error is between $0.3 - 0.5(\%bb)$, and still cannot be strictly bounded. In contrast, our algorithm is able to explicitly control the approximation error, and achieve a value below $0.07(\%bb)$ with the same input. More complete comparisons are in Section 6.8.

6.7 Robustness

Since in our local error update scheme, the closest point pairs of stratified samples are reliably initialized and locally updated, our method is robust to models with complex topology, holes, and intersections of surfaces. For example, the Close spheres model (Fig. 19c) is composed of two rounded half spheres that are positioned very close to each other, and the Klein bottle model (Fig. 19h) exhibits non-orientable surfaces that are self-intersecting. Our method generates correct results for both of them.

By integrating the feature intensity function, our method successfully handles models with and without clean sharp features. The Helmet (Fig. 19a), Fertility (Fig. 19b) and Close spheres models are smooth, whereas the U-part (Fig. 19e) and the Smooth crease models (Fig. 19f) have sharp features. In addition, since the feature intensity also captures boundaries, our method is capable of remeshing models with boundaries (Fig. 19i) and smooth features (Fig. 19f).

TABLE 2
Comparison with the State-of-the-Art Methods

Input	Methods	$\#V$	Q_{min}	$\theta_{min}(^\circ)$	$\theta_{max}(^\circ)$	Hdist (%bb)	RMS (%bb)	$\theta < 30^\circ(\%)$	$V_{567}(\%)$	Time
	[RAR]	**2.1 k**	0.556	27.9	107.2	*0.20*/0.94	0.120	0.02	**100**	**< 0 : 01**
	[MMGS]	5.8 k	0.056	3.4	172.5	*0.47*	0.103	1.73	93.4	0 : 01
	[CVT]	*5.8 k*	0.588	28.3	104.6	0.21	0.030	0.02	99.9	0 : 48
Rockerarm	[MPS]	*5.8 k*	0.516	32.0	113.6	0.48	0.033	**0**	**100**	0 : 05
(3.4 k)	[OUR*]	2.8 k	0.559	*35.0*	108.5	*0.20*/0.20	0.025	**0**	89.0	0 : 29
	[OUR*]	*3.0 k*	0.612	38.6	102.2	*0.20*/0.17	0.024	**0**	95.4	0 : 38
	[OUR*]	3.8 k	**0.646**	*40.0*	**98.1**	*0.20*/**0.14**	**0.020**	**0**	98.5	0 : 51
	[OUR]	3.6 k	0.639	*40.0*	99.0	*0.20*/0.20	0.025	**0**	97.9	2 : 58
	[RAR]	**2.6 k**	0.569	29.2	106.5	*0.20*/0.55	0.070	0.02	99.9	**< 0 : 01**
	[MMGS]	7.2 k	0.210	13.1	152.2	*0.43*	0.028	1.07	95.8	0 : 01
	[CVT]	*7.2 k*	0.568	25.3	102.3	0.10	0.021	0.02	99.9	1 : 14
Homer	[MPS]	*7.2 k*	0.513	32.0	115.0	0.31	0.023	**0**	**100**	0 : 05
(6.0 k)	[OUR*]	4.8 k	0.553	*35.0*	109.2	*0.20*/0.09	0.010	**0**	91.8	0 : 46
	[OUR*]	*5.0 k*	0.600	37.8	103.6	*0.20*/0.09	0.010	**0**	95.2	0 : 59
	[OUR*]	6.9 k	**0.643**	*40.0*	**98.5**	*0.20*/**0.07**	**0.009**	**0**	98.7	1 : 29
	[OUR]	4.3 k	0.635	*40.0*	99.5	*0.20*/0.17	0.018	**0**	97.8	4 : 48
	[RAR]	**1.6 k**	0.607	30.0	98.4	*0.20*/2.61	0.570	0.03	99.8	**< 0 : 01**
	[MMGS]	9.0 k	0.270	13.5	143.3	*0.41*	0.080	1.11	93.6	0 : 01
	[CVT]	*9.0 k*	0.543	31.7	110.3	**0.12**	**0.018**	**0**	99.9	1 : 23
Triceratops	[MPS]	*9.0 k*	0.506	32.0	114.8	0.46	0.062	**0**	**100**	0 : 29
(2.8 k)	[OUR*]	2.1 k	0.552	*35.0*	109.3	*0.20*/0.16	0.028	**0**	87.0	0 : 28
	[OUR*]	*3.0 k*	0.605	38.4	103.0	*0.20*/0.18	0.024	**0**	93.8	0 : 41
	[OUR*]	4.8 k	0.634	*40.0*	99.6	*0.20*/0.19	0.036	**0**	97.7	1 : 39
	[OUR]	3.5 k	**0.644**	*40.0*	**98.4**	*0.20*/0.19	0.040	**0**	97.1	2 : 36
	[RAR]	2.9 k	0.480	24.9	115.5	*0.20*/5.0	0.112	0.12	**100**	**< 0 : 01**
	[MMGS]	11 k	0.187	10.9	155.1	*0.29*	0.034	1.24	93.5	0 : 02
	[CVT]	*11 k*	0.560	26.8	107.7	**0.11**	0.018	0.01	99.8	1 : 43
Elephant	[MPS]	*11 k*	0.505	32.0	114.9	0.38	0.061	**0**	**100**	0 : 31
(6.9 k)	[OUR*]	4.4 k	0.553	*35.0*	109.2	*0.20*/**0.11**	0.014	**0**	90.4	0 : 49
	[OUR*]	*5.0 k*	0.617	39.1	101.5	*0.20*/0.13	0.013	**0**	96.6	1 : 05
	[OUR*]	6.8 k	**0.638**	*40.0*	**99.1**	*0.20*/**0.11**	**0.010**	**0**	98.7	1 : 45
	[OUR]	**2.7 k**	0.633	*40.0*	99.7	*0.20*/0.14	0.021	**0**	98.4	4 : 25
	[RAR]	4.1 k	0.545	27.5	109.4	*0.20*/0.69	0.072	0.02	**100**	**< 0 : 01**
	[MMGS]	12 k	0.260	13.1	142.4	*0.41*	0.041	0.59	93.8	0 : 03
	[CVD]	*12 k*	0.150	9.6	160.1	0.34	0.028	0.71	96	0 : 05
Bunny	[CVT]	*12 k*	0.603	30.6	103.1	0.20	0.018	**0**	99.9	3 : 57
(34 k)	[MPS]	*12 k*	0.510	32.0	114.6	0.37	0.035	**0**	**100**	0 : 24
	[OUR*]	37 k	0.550	*35.0*	109.6	*0.20*/**0.07**	**0.003**	**0**	95.8	3 : 23
	[OUR]	*1.5 k*	0.637	39.8	99.2	*0.20*/0.19	0.033	**0**	97.2	10 : 41
	[OUR]	1.7 k	**0.646**	*40.0*	**98.2**	*0.20*/0.19	0.031	**0**	97.7	12 : 28

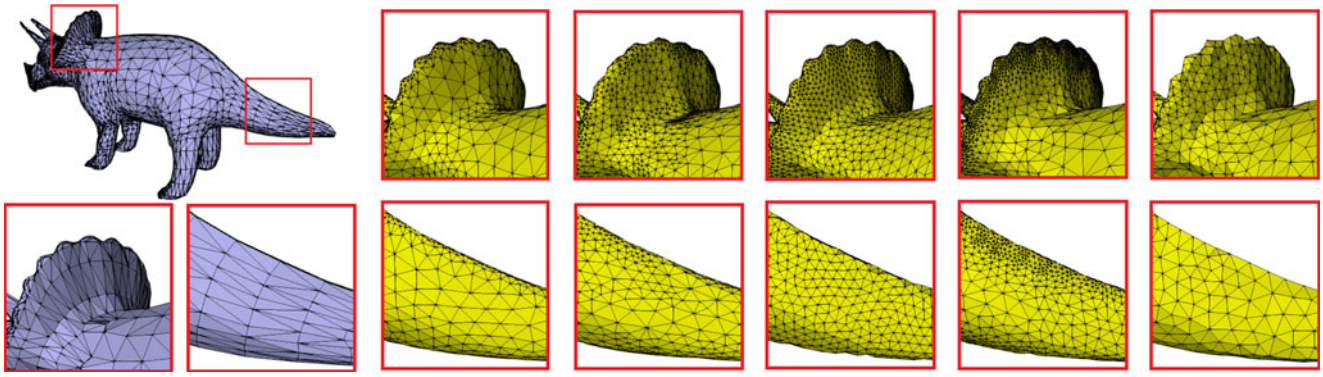
For all the method, the input parameters are highlighted with italic fonts, and the best results are highlighted in bold. In column Hdist(%bb), the values before "/" are the input, and the values after "/" are the real HDist measured by Metro [48].

Our method requires no surface parameterization, making it naturally suitable for high-genus models (Figs. 19a and 19b) as well as models with multiple components (Fig. 19g). Note that if reasonable, adaptivity is created automatically by the approximation error parameter δ without requiring an a priori estimation of a density function. The Helmet, Fertility, U-part and Lion head models in Fig. 19 clearly illustrate this advantage.

Our method is suitable to process models with very high/low resolutions and/or badly shaped triangles. To tackle dense models, typically the initial mesh simplification strongly reduces the mesh complexity. For very coarse models, users can optionally increase the sampling density for better error control. We present two typical examples: the input Fertility model has 13 k vertices, whereas our remeshing result (Fig. 19b) has only 2.9 k vertices, thanks to the effectiveness of Algorithm 2. The input U-part model possesses only 86 vertices. By sampling 50 points in each facet averagely, we get the result with 347 vertices, and the geometric fidelity is well-controlled.

6.8 Comparisons

We compare our approach to the state-of-the-art techniques in terms of efficiency, geometric fidelity, element quality and mesh complexity. For simplicity, only the most efficient methods (RAR [26] and MMGS, an improvement of YAMS [24]) and the methods that produce the best results in Yan and Wonka's conduction [9] (CVT [8] (100 iterations in our experiments), CVD [4] and MPS [9]) are compared with identical inputs. Among all the compared methods, CVT and CVD require the number of vertices to be specified, and the Hdist is required in RAR and MMGS. To make the results comparable, we set the Hdist of RAR to the same value as that of our method, and carefully adjusted the



(a) Input Triceratops model (b) RAR (c) MMGS (d) CVT (e) MPS (f) OUR

Fig. 20. A close-up comparison of results with state-of-the-art approaches. In our method, δ is set to $0.20(\%bb)$ and θ is set to 40 degree.

Hdist parameter for MMGS, such that it generates results with the same complexity as CVT, MPS and CVD. In our method, OUR* means Algorithm 2 is disabled while OUR means it is enabled. A detailed comparison is listed in Table 2, and Fig. 20 shows a close-up comparison. More visual comparisons are provided in the supplemental materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2016.2632720.

From all compared methods, RAR performs most efficiently and introduces the lowest mesh complexity. However, the geometric fidelity cannot be guaranteed. MMGS is also efficient, yet it introduces much higher Hdist distances. Our method is almost at the same level of efficiency as CVT when $\theta = 35°$, but is substantially slower when $\theta = 40°$.

According to [9], CVT performs best in keeping high geometric fidelity, but cannot explicitly bound the approximation error. By setting δ comparable to CVT's best results, our method consistently produces results with strictly bounded Hdist, and produces lower Hdist and RMS distances in most cases. We also find that our method consistently generates results with higher minimal angle and triangle quality than MPS, due to the fact that our method provides a better improvement of "worst element quality" measured by the minimal angle. However, since our method does not optimize the global connectivity of the input, our results have lower regularity (measured as V_{567}) than MPS.

From Table 2, we see that by setting the desired resolution lower than MMGS, CVT, MPS and CVD, we still get results with higher geometric fidelity and better bounds of minimal angle. For the very dense Bunny model, when Algorithm 2 is enabled, our method even reduces the complexity of the Bunny model to 5 percent of the input without violating the error-bound constraint. When it is disabled, the resolutions are still lower than the inputs in most cases, since the edge collapse operator has high priority in Algorithm 1.

To the best of our knowledge, RAR, MMGS, CVT and MPS require sharp features to be specified or detected in advance, which may be time-consuming or error-prone. Though CVD is able to preserve features implicitly, it leads to results with lower geometric fidelity and element quality than our method.

Since our method does not optimize the element quality globally, the average element quality is not superior to remeshing based on farthest point optimization [22] (FPO). However, we consistently produce results with better geometric fidelity and larger minimal angle than FPO.

TABLE 3
Cases Where Too High θ is Specified

Input	#V	θ_{min}	Hdist(%bb)	Fail Types
Rockerarm	4.2k	41.22	*0.20*/0.19	Degenerated edges
Homer	6.1k	41.13	*0.20*/0.14	Infinite loops
Triceratops	4.8k	40.02	*0.20*/0.19	Degenerated edges
Elephant	4.8k	41.23	*0.20*/0.13	Infinite loops
Bunny	2.4k	41.42	*0.20*/0.19	Infinite loops

We set $\theta = 60°$, run Algorithm 1 until it fails, and record the best results it achieved. Algorithm 2 is enabled here.

6.9 Limitations

Although practically the minimal angles can be improved to values above 35 degree in all our test cases, we do not have any theoretical guarantee for the convergence with a specified θ. To challenge our algorithm, we set θ to its theoretical upper bound and show the best results that our algorithm achieved in Table 3. We found the approximation error can still be bounded. However, the algorithm runs into infinite loops or generates degenerated edges while refining.

Another limitation is that we can only tackle two-manifold meshes, for the reason that our local operators crucially rely on the topology information of local regions. Finally, our method cannot remesh noisy models robustly, since the feature intensity function interprets noise as some kind of features and thus tries to preserve it.

7 Conclusion

We introduced a novel surface remeshing algorithm based on minimal angle improvement. In this framework, the minimal angle of the input model is sequentially improved by applying local operators. Furthermore, an efficient and reliable local error update scheme is designed and embedded for explicitly bounding the approximation error, and two novel feature intensity functions are defined and utilized for vertex relocation, in order to preserve features implicitly. Compared to the state-of-the-art, our method consistently generates results with higher element quality, lower mesh complexity as well as satisfied error-bounds. The resulting

meshes are well suited for robust numerical simulations, since they offer bounded approximation error and large minimal angles.

However, there are still some limitations (Section 6.9), which motivate future work in the following aspects: 1) providing a theoretical convergence guarantee with a specified θ; 2) extending the current implementation such that this framework can be applied to triangle soups or even point clouds, for error-bounded and feature preserving mesh reconstruction; and 3) exploring more robust feature intensity functions for measuring sharp features on noisy models.

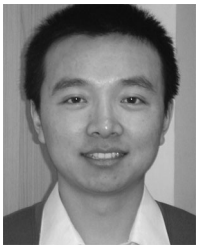
Acknowledgments

We wish to thank Mario Botsch for providing the source code and results of their method, and Pascal Frey for providing the MMGS software. This work was partially supported by the European Research Council (ERC Starting Grant Robust Geometry Processing #257474), the National Science Foundation of China (61373071, 61372168, 61620106003) and the German Research Foundation (DFG, grant GSC 111, Aachen Institute for Advanced Study in Computational Engineering Science).

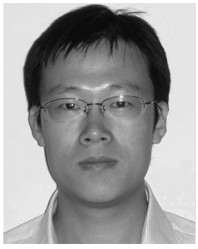
References

[1] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon Mesh Processing*. Natick, MA, USA: AK Peters, 2010.
[2] J. R. Shewchuk, "What is a good linear element? interpolation, conditioning, and quality measures," in *Proc. 11th Int. Meshing Roundtable*, 2002, pp. 115–126.
[3] P. Alliez, É. C. de Verdière, O. Devillers, and M. Isenburg, "Centroidal Voronoi diagrams for isotropic surface remeshing," *Graph. Models*, vol. 67, no. 3, pp. 204–231, 2005.
[4] S. Valette, J.-M. Chassery, and R. Prost, "Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 2, pp. 369–381, Mar./Apr. 2008.
[5] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 905–914, 2004.
[6] M. Mandad, D. Cohen-Steiner, and P. Alliez, "Isotopic approximation within a tolerance volume," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 64:1–64:12, 2015.
[7] P. Alliez, M. Meyer, and M. Desbrun, "Interactive geometry remeshing," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 347–354, 2002.
[8] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted Voronoi diagram," *Comput. Graph. Forum*, vol. 28, no. 5, pp. 1445–1454, 2009.
[9] D.-M. Yan and P. Wonka, "Gap processing for adaptive maximal Poisson-disk sampling," *ACM Trans. Graph.*, vol. 32, no. 5, pp. 148:1–148:15, Oct. 2013.
[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Proc. 20th Annu. Conf. Comput. Graph. Interactive Techn.*, 1993, pp. 19–26.
[11] M. Botsch and L. Kobbelt, "A remeshing approach to multiresolution modeling," in *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Process.*, 2004, pp. 185–192.
[12] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, "Recent advances in remeshing of surfaces," in *Shape Analysis and Structuring, Mathematics and Visualization*. Berlin, Germany: Springer, 2008, pp. 53–82.
[13] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, pp. 637–676, 1999.
[14] P. Alliez, É. C. de Verdière, O. Devillers, and M. Isenburg, "Isotropic surface remeshing," in *Proc. Shape Model. Int.*, 2003, pp. 49–58.
[15] V. Surazhsky, P. Alliez, and C. Gotsman, "Isotropic remeshing of surfaces: A local parameterization approach," in *Proc. 12th Int. Meshing Roundtable*, 2003, pp. 215–224.
[16] V. Surazhsky and C. Gotsman, "Explicit surface remeshing," in *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Process.*, 2003, vol. 43, pp. 20–30.
[17] S. Fuhrmann, J. Ackermann, T. Kalbe, and M. Goesele, "Direct resampling for isotropic surface remeshing," in *Proc. Int. Workshop Vis. Model. Vis.*, 2010, pp. 9–16.
[18] J. Vorsatz, C. Rössl, and H.-P. Seidel, "Dynamic remeshing and applications," *J. Comput. Inf. Sci. Eng.*, vol. 3, no. 4, pp. 338–344, 2003.
[19] D.-M. Yan, G. Bao, X. Zhang, and P. Wonka, "Low-resolution remeshing using the localized restricted Voronoi diagram," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 10, pp. 1418–1427, Oct. 2014.
[20] D.-M. Yan and P. Wonka, "Non-obtuse remeshing with centroidal Voronoi tessellation," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 9, pp. 2136–2144, Sep. 2016.
[21] J. Guo, D.-M. Yan, X. Jia, and X. Zhang, "Efficient maximal Poisson-disk sampling and remeshing on surfaces," *Comput. Graph.*, vol. 46, pp. 72–79, 2015.
[22] D.-M. Yan, J. Guo, X. Jia, X. Zhang, and P. Wonka, "Blue-noise remeshing with farthest point optimization," *Comput. Graph. Forum*, vol. 33, no. 5, pp. 167–176, 2014.
[23] A. G. M. Ahmed, J. Guo, D.-M. Yan, X. Zhang, and O. Deussen, "A simple push-pull algorithm for blue-noise sampling," *IEEE Trans. Vis. Comput. Graph.*, 2016.
[24] P. Frey and P. George, *Mesh Generation, Application to Finite Elements*. Paris, France: Hermès Sci. Publications, 2000, Art. no. 814.
[25] R. Narain, A. Samii, and J. F. O'Brien, "Adaptive anisotropic remeshing for cloth simulation," *ACM Trans. Graph.*, vol. 31, no. 6, 2012, Art. no. 152.
[26] M. Dunyach, D. Vanderhaeghe, L. Barthe, and M. Botsch, "Adaptive remeshing for real-time mesh deformation," in *Proc. Eurographics Short Papers*, 2013, pp. 29–32.
[27] S. A. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[28] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proc. 24th Annu. Conf. Comput. Graph. Interactive Techn.*, 1997, pp. 209–216.
[29] H. Borouchaki and P. Frey, "Simplification of surface mesh using Hausdorff envelope," *Comput. Methods Appl. Mech. Eng.*, vol. 194, no. 48, pp. 4864–4884, 2005.
[30] Y. Sun, D. L. Page, J. K. Paik, A. Koschan, and M. A. Abidi, "Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing," in *Proc. Int. Conf. Image Process.*, 2002, pp. 825–828.
[31] X. Jiao and M. T. Heath, "Feature detection for surface meshes," in *Proc. 8th Int. Conf. Numerical Grid Generation Comput. Field Simul.*, 2002, pp. 705–714.
[32] J. Chen, et al., "Bilateral blue noise sampling," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 216:1–216:11, 2013.
[33] Z. Zhong, et al., "Particle-based anisotropic surface meshing," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 99.
[34] L. Kobbelt and M. Botsch, "Feature sensitive mesh processing," in *Proc. 19th Spring Conf. Comput. Graph.*, 2003, pp. 17–22.
[35] B. Lévy and Y. Liu, "L_p centroidal Voronoi tessellation and its applications," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 119:1–119:11, 2010.
[36] J. Vorsatz, C. Rössl, L. Kobbelt, and H.-P. Seidel, "Feature sensitive remeshing," *Comput. Graph. Forum*, vol. 20, no. 3, pp. 393–401, 2001.
[37] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung, "Instant field-aligned meshes," *ACM Trans. Graph.*, vol. 34, no. 6, Nov. 2015, Art. no. 189.
[38] M. C. Rivara, "New mathematical tools and techniques for the refinement and/or improvement of unstructured triangulations," in *Proc. 5th Int. Meshing Roundtable*, 1996, pp. 77–86.
[39] H. Edelsbrunner, *Geometry and Topology for Mesh Generation*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[40] J. Henrikson, "Completeness and total boundedness of the Hausdorff metric," *MIT Undergraduate J. Math.*, vol. 1, pp. 69–80, 1999.
[41] M. Tang, M. Lee, and Y. J. Kim, "Interactive Hausdorff distance computation for general polygonal models," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 74.
[42] J. Tournois, P. Alliez, and O. Devillers, "2D centroidal Voronoi tessellations with constraints," *Numerical Math.: Theory Methods Appl.*, vol. 3, no. 2, pp. 212–222, 2010.
[43] P. Alliez, S. Tayeb, and C. Wormser, "3D fast intersection and distance computation," in *CGAL User and Reference Manual*. CGAL Editorial Board, 2016.

[44] T. Winkler, K. Hormann, and C. Gotsman, "Mesh massage," *Visual Comput.*, vol. 24, no. 7–9, pp. 775–785, 2008.
[45] T. S. Motzkin and J. L. Walsh, "Polynomials of best approximation on a real finite point set," *Trans. Amer. Math. Soc.*, vol. 92, no. 2, pp. 231–245, 1959.
[46] C. Lawson, *Contributions to the Theory of Linear Least Maximum Approximation*. Los Angeles, CA, USA: UCLA-Math., 1961.
[47] CGAL, computational geometry algorithms library. [Online]. Available: http://www.cgal.org
[48] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Comput. Graph. Forum*, vol. 17, no. 2, pp. 167–174, 1998.
[49] P. J. Frey and H. Borouchaki, "Surface mesh evaluation," in *Proc. 6th Int. Meshing Roundtable*, 1997, pp. 363–374.



Kaimo Hu received the bachelor's and PhD degrees from Tsinghua University, in 2006 and 2012, respectively. He is a post doctoral research assistant in the Department of Computer Graphics Technology, Purdue University. His research interests include computer graphics, shape analysis, geometric processing, and procedural modeling.



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

David Bommes is an assistant professor in the Computer Science Department, RWTH Aachen University. In May 2016, he received the EUROGRAPHICS Young Researcher Award. He is leading the Mesh Generation and Optimization Group, which scientifically contributes to various areas of geometry processing, in particular direction fields, parametrization, quad/hex mesh generation, and nonlinear/mixed-integer optimization.

Pierre Alliez received the PhD degree in computer science from Telecom ParisTech, in 2000. He is a senior researcher and team leader with Inria Sophia Antipolis. He published on topics commonly referred to as geometry processing. He received the Starting Grant from the European Research Council on Robust Digital Geometry Processing, in 2011.

Bedrich Benes received the MS and PhD degrees from Czech Technical University. He is a professor in the Computer Graphics Technology Department, Purdue University. His research is primarily in the area of procedural modeling, real-time rendering, and 3D computer graphics in general. To date he has published more than 100 peer reviewed publications.