Pacific Graphics 2012
C. Bregler, P. Sander, and M. Wimmer
(Guest Editors)

Volume 31 (2012), Number 7

Isotropic Surface Remeshing Using Constrained Centroidal Delaunay Mesh

Zhonggui Chen[1,3], Juan Cao[2,3†], and Wenping Wang[3]

[1]School of Information Science and Technology, Xiamen University, China
[2]School of Mathematical Sciences, Xiamen University, China
[3]Department of Computer Science, The University of Hong Kong, Hong Kong

Abstract

We develop a novel isotropic remeshing method based on constrained centroidal Delaunay mesh *(CCDM), a generalization of centroidal patch triangulation from 2D to mesh surface. Our method starts with resampling an input mesh with a vertex distribution according to a user-defined density function. The initial remeshing result is then progressively optimized by alternatively recovering the Delaunay mesh and moving each vertex to the centroid of its 1-ring neighborhood. The key to making such simple iterations work is an efficient optimization framework that combines both local and global optimization methods. Our method is parameterization-free, thus avoiding the metric distortion introduced by parameterization, and generating more well-shaped triangles. Our method guarantees that the topology of surface is preserved without requiring geodesic information. We conduct various experiments to demonstrate the simplicity, efficacy, and robustness of the presented method.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

In computer graphics applications, the triangular mesh is one of the most commonly used shape representations to approximately describe the geometric shape. Many of these meshes acquired by scanning devices or by isosurfacing implicit representations are often highly redundant or/and noisy, and are not suitable for practical applications in rending, editing, animation, compression and simulation, etc. Therefore, it is oftentimes desired to remesh the existing triangular meshes in advance to reduce the complexity and improve the mesh quality.

The requirement of the remeshing approach may vary according to the targeted goal or application. Nonetheless, it is commonly agreed that a good remeshing algorithm is imperative to have the following fundamental properties [AUGA08, FAKG10]: 1) It generates a high-quality mesh that represents the input mesh accurately and preserves the sharp features; 2) It is computationally efficient for handling large meshes; 3) It is general and robust to handle meshes of arbitrary genus; 4) It is simple to practical implement. However, most existing remeshing methods are designed by relaxing some of the mentioned properties, as we will find later in our discussion. Therefore, a new remeshing approach possessing all desirable properties is required.

In this paper, we develop a novel approach to remesh a triangular mesh with arbitrary topology based on a generalized concept of centroidal patch triangulation (CPT) [CH11]. We named our algorithm constrained centroidal Delaunay mesh, or CCDM for short. Compared with existing remeshing algorithms, our algorithm has several desirable advantages. First, it does not rely on any parameterization, thus avoiding the distortions and instabilities caused by the parameterization. Second, we do not need to compute the exact or approximated geodesic distance on the mesh surface, which makes our algorithm very efficient. Specifically, it takes $O(n)$ time for each iteration of our algorithm, where n is the vertex number of the remeshed surface. Third, it is accurate because it always refers to the original mesh during the remeshing, and the topology of the original surface is preserved after remeshing.

† Corresponding author: juancao@xmu.edu.cn

To summarize, our main contributions are as following:

- We introduce the constrained centroidal Delaunay mesh which is a generalization of the centroidal patch triangulation from 2D to surfaces, and is easy to be constructed by a simple iterative algorithm (Section 2 and 3).
- We present an efficient optimization framework combining both local and global optimization methods that results in high-quality and topologically faithful remeshing results (Section 3 and 4).

1.1. Related Work

Since remeshing is a critical step in a number of geometry processing techniques, it has been received a lot of attention in recent years. We restrict our description to unstructured isotropic remeshing methods, and refer the interested readers to [AUGA08] for a comprehensive survey of remeshing technologies. According to whether they rely on a local or global parameterization to decide the correspondence of the new points and the input mesh, the remeshing techniques fall into two categories, i.e., parameterization-based methods and parameterization-free methods.

In the parameterization-based remeshing techniques, the input mesh may globally [AMD02, AdVDI03] or locally [SAG03, SG03, VRS03, FZ08, FAKG10] be mapped onto a 2D domain, the remeshing operation is performed on the 2D parametric domain and then the results are mapped back to the 3D surface. For parameterization-based remeshing methods, a high-quality remeshing result heavily relies on a good surface parameterization, as it is the basis for building a "good" correspondence between the new vertex and the original mesh. On one hand, a good parameterization in terms of minimizing certain distortion measures, such as area and angle distortion, usually requires an input mesh with high-quality. On the other hand, the reason for remeshing is that the input "raw" mesh quality is need to be improved in terms of vertex sampling, regularity and triangle quality. This dilemma between the parameterization and remeshing is faced by all parameterization-based remeshing methods and unfortunately is not well solved by most.

The parameterization-free remeshing methods perform the vertex inserting, removing or relocation operation directly on the surface. Some of the most important paradigms include farthest point sampling method [PC06], advancing front method [SSG03], Delaunay refinement method [DR10, DLS10], and relaxation-based method [Tur92, DGJ02, BK04, VCP08, YLL*09]. The relaxation-based methods have become one of the main tools for generating high-quality remeshing surfaces. In particular, the centroidal Voronoi tessellation (CVT) based methods have been effective for surface remeshing. Valette et al. [VCP08] compute approximated Voronoi cells on surface by clustering mesh vertices. This method is fast but the remeshing result may be poor when the input mesh contains degenerate triangles.

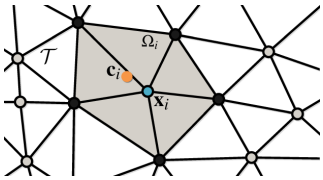


Figure 1: *The 1-ring neighbor patch Ω_i (gray region) of $\mathbf{x}_i$ and its centroid $\mathbf{c}_i$ (orange point).*

Yan et al. [YLL*09] compute the exact restricted Voronoi diagram, and optimize the CVT energy function by using an efficient quasi-Newton method. This method is capable of producing high-quality meshes, but the computation of exact restricted Voronoi diagram in each iteration is computationally expensive. Furthermore, it may fail to build a correct restricted Voronoi diagram for mesh with small features, such as two parallel planes that are close to each other, due to the poor approximation of geodesic distance on surface with the Euclidean distance. Botsch and Kobbelt [BK04] propose a simple but highly efficient method for uniform remeshing. Conceptually, this method is very similar to our approach as they apply the local vertex relocation steps directly on the mesh surface without any reference to global or local parameterization. However, our approach can produce remeshing results with higher qualities due to the global optimization framework.

2. Background

Mesh optimization by means of minimizing some energy is a popular strategy to generate high-quality meshes. There are many energies proposed in the literatures for generating triangular meshes or tetrahedral meshes, e.g., [DGJ02, CX04, TWAD09, CH11] and references therein. Among them, the centroidal patch triangulation (CPT) based mesh smoothing method is a computationally efficient method lastly introduced in [CH11]. In this section, we first briefly recall the concept and corresponding energy function of CPT, and then introduce the so called Delaunay mesh, which is an extension of the Delaunay triangulation to 2D manifold.

2.1. Centroidal Patch Triangulation

Consider a set of points in $\mathbb{R}^N$ denoted by $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^N$. Let $\mathcal{T}$ be a triangulation of $\mathbf{X}$. For any vertex $\mathbf{x}_i \in \mathbf{X}$, the set of all incident faces, denote by Ω_i, is called the 1-ring neighbor patch of $\mathbf{x}_i$ (see Figure 1). The centroid of Ω_i with respect to a density function $\rho(\mathbf{x}) > 0$ is defined as

$$\mathbf{c}_i = \frac{\int_{\Omega_i} \rho(\mathbf{x})\mathbf{x}d\sigma}{\int_{\Omega_i} \rho(\mathbf{x})d\sigma}.$$

A triangulation $\mathcal{T}$ of $\mathbf{X}$ is called a *centroidal patch triangulation* (CPT), if any inner vertex $\mathbf{x}_i \in \mathcal{T}$ coincides with the centroid $\mathbf{c}_i$ of patch Ω_i.

The Delaunay triangulation has been widely used for mesh

generation, due to its many optimal properties, such as maximum minimum angle property. A Delaunay-based objective function for generating CPT, which is proposed in [CH11], is defined as

$$\mathcal{F}(\mathbf{X},\mathcal{T}) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x}-\mathbf{x}_i\|^2 d\sigma, \tag{1}$$

where $\mathcal{T}$ is the Delaunay triangulation of $\mathbf{X}$. The minimization of $\mathcal{F}(\mathbf{X},\mathcal{T})$ is performed through minimizing the energy defined over each patch Ω_i, say,

$$\min_{\mathbf{x}_i} \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x}-\mathbf{x}_i\|^2 d\sigma. \tag{2}$$

Based on the fact that the centroid of Ω_i is the minimizer of Equation (2), a two-step iterative algorithm was designed to generate the centroidal patch triangulation [CH11]. Beginning with a set of points, the following two steps are iterated until some stopping criterion is met:

1. (*connectivity update*) generate the Delaunay triangulation of $\mathbf{X}$;
2. (*vertex relocation*) move each vertex $\mathbf{x}_i$ to the centroid $\mathbf{c}_i$ of the corresponding patch $\Omega_i \subset \mathcal{T}$, ie,

$$\mathbf{x}_i^* = \mathbf{c}_i. \tag{3}$$

Note that, the definition of CPT is similar to the definition of centroidal Voronoi tessellation (CVT) [DFG99], for which each generator coincides with the centroid of the corresponding Voronoi region. Also, the CPT algorithm and CVT algorithm have comparable performance on 2D and 3D mesh generation [CH11]. In this paper, we exploit the potential of CPT algorithm in surface remeshing. A CPT-based surface remeshing method is proposed, which is shown to outperform the existing CVT-based methods in both mesh quality improvement and computational cost.

2.2. Delaunay Mesh

In order to generalize the CPT algorithm to surface remeshing, we first need a generalization of the concept of Delaunay triangulation. There exist several methods to define a Delaunay triangulation of a set of points on a Riemannian manifold, e.g., the *restricted Delaunay triangulation* [ES94] and the *intrinsic Delaunay triangulation* [BS07]. In this paper, we adopt the *Delaunay mesh* [DZM07b] because of its simplicity and fast construction.

A Delaunay mesh is a manifold triangular mesh whose edges form an intrinsic Delaunay triangulation of its vertices [DZM07b]. Intuitively, a mesh is a Delaunay mesh when all its edges are *locally Delaunay*, where an edge is locally Delaunay if the sum of the two angles opposite to it does not exceed π. Deyer et al. [DZM07a] propose an efficient iterative algorithm to generate Delaunay mesh from an arbitrary triangle mesh by edge flipping, and prove that such iterative process terminates in finite steps.

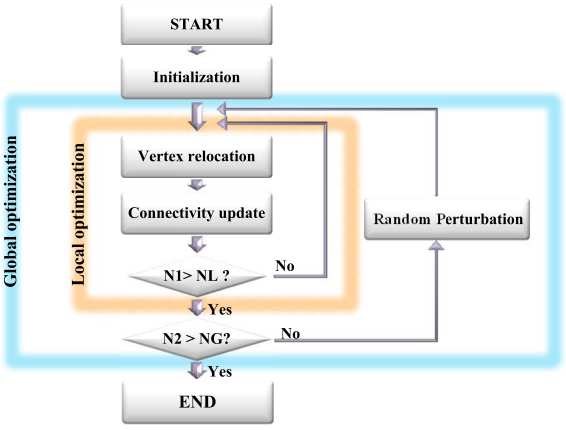


Figure 2: *The pipeline of CCDM-based remeshing framework.*

3. Algorithm of Constrained Centroidal Delaunay Mesh

For a compact surface $S \subset \mathbb{R}^3$ and a set of points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \subset S$, the Delaunay mesh of the point set $\mathbf{X}$ is denoted by $\mathcal{M}$. We define the *constrained centroidal Delaunay meshes* (CCDMs) as the solutions of the following optimization problem:

$$\min_{\mathbf{X},\mathcal{M}} \mathcal{F}(\mathbf{X},\mathcal{M}) \qquad \text{subject to } \mathbf{x}_i \in S, i = 1,\ldots,n,$$

where $\mathcal{F}(\cdot)$ is the function defined in Equation (1), and the points $\mathbf{X}$ are constrained on S.

The algorithm for optimizing a mesh of surface S by minimizing $\mathcal{F}(\mathbf{X},\mathcal{M})$ is given in Algorithm 1. Note that Algorithm 1 differs from CPT iteration only in step 2, where the updated vertices are needed to be projected back onto the surface S. However, the direct application of Algorithm 1 to mesh optimization hardly succeeds in producing high-quality meshes, since Algorithm 1 is a local search method which usually leads to a local minimizer of $\mathcal{F}(\mathbf{X},\mathcal{M})$ corresponding to a mesh with poor quality. Therefore, we propose a complete remeshing framework which uses global optimization method to minimize the CCDM energy function.

3.1. Algorithm Overview

Our remeshing algorithm takes as input an orientable 2-manifold triangular mesh $\mathcal{M}_I$ of arbitrary topology, which may contain boundaries and sharp features. Our output is a high-quality mesh $\mathcal{M}_O$ with a desired number of vertices, and a vertex distribution according to a specified density function. The basic idea is to start with an initial mesh with the same topology as $\mathcal{M}_I$, and then to improve the mesh quality by using CCDM-based optimization. A overview of our algorithm pipeline is given in Figure 2.

The optimization framework consists of an inner loop and an outer loop, which are called as local optimization and global

optimization respectively. These two loops are iterated until the specified maximum iteration numbers *NL* and *NG* are achieved respectively. We will elaborate each component of the CCDM algorithm in the following sections.

3.2. Initial Mesh Construction

Adaptive Sampling: In many practical applications in computer graphics, the distribution of the sites is expected to adapt to some given probability density function $\phi(\mathbf{x})$ defined on $\mathcal{M}_I$. To achieve this goal, a user-specified number of points are initially sampled on $\mathcal{M}_I$ with a distribution complying with $\phi(\mathbf{x})$. However, simply choosing the probability density function $\phi(\mathbf{x})$ as the density function $\rho(\mathbf{x})$ in CCDM energy function can not ensure a consistent distribution of the vertices in the subsequent optimization. Instead, $\sqrt[5]{\phi(\mathbf{x})}$ is empirically demonstrated as a more sophisticated choice for $\rho(\mathbf{x})$. In our experiments, we find that there is no obvious global drift of the vertices during the optimization when $\rho(\mathbf{x}) = \sqrt[5]{\phi(\mathbf{x})}$. Hence, the result mesh is apt to have a consistent vertex distribution adapted to $\phi(\mathbf{x})$. The probability density function $\phi(\mathbf{x})$ can be any positive function given by the user. Generally, we want more sampling points in the regions with high curvature. Thus we consider both the Gaussian curvature K and the mean curvature H of the input surface, and set $\phi(\mathbf{x})$ to $\sqrt{|K|+H^2}$.

Feature Preservation: The features of the input mesh $\mathcal{M}_I$, including sharp edges, boundary edges, and corners may be either specified by the user or extracted automatically. We simply mark the edges with small dihedral angles as the feature edges, but other more sophisticated feature extracting methods can also be applied. Some sampling points are then placed on the feature curves and corners to preserve the features. We use the formula from [AdVDI03] to allocate an appropriate number of points to each feature curve.

Building the Mesh: There are several surface reconstruction methods that can be used to connect the sampling points to obtain a valid triangle mesh. For the purpose of remeshing, the reconstructed mesh should have a topology which is consistent with the reference mesh $\mathcal{M}_I$. Certainly, we can obtain such a mesh by computing the dual triangulation of the geodesic-metric-based Voronoi diagram [LCT11] of $\mathbf{X}$ on $\mathcal{M}_I$. However, this strategy is computationally costly. Therefore, we follow a more efficient method used in [FZ08, FAKG10]. A mutual tessellation [Tur92] is first created by inserting the sampling points to the mesh $\mathcal{M}_I$, and then is cleaned by deleting the original vertices of $\mathcal{M}_I$. The feature edges connecting feature points are preserved during these operations. Neither insertion nor deletion of vertices is allowed to change the topology of the surface, thus the topology is preserved.

3.3. Local Optimization

Algorithm 1 Local optimization of CCDM algorithm

Input: the inpute mesh $\mathcal{M}_I$, and an initial remesh $\mathcal{M}_O$
Output: an optimized remesh $\mathcal{M}_O$

(1) (connectivity update) make $\mathcal{M}_O$ a Delaunay mesh by edge flipping;

(2) (vertex relocation) move each vertex $\mathbf{x}_i$ of $\mathcal{M}_O$ to the centroid $\mathbf{c}_i$ of the corresponding patch $\Omega_i \subset \mathcal{M}_O$, and project the updated vertex back onto $\mathcal{M}_I$;

(3) if the number of iterations exceeds *NL*, then return $\mathcal{M}_O$; otherwise return to step 1.

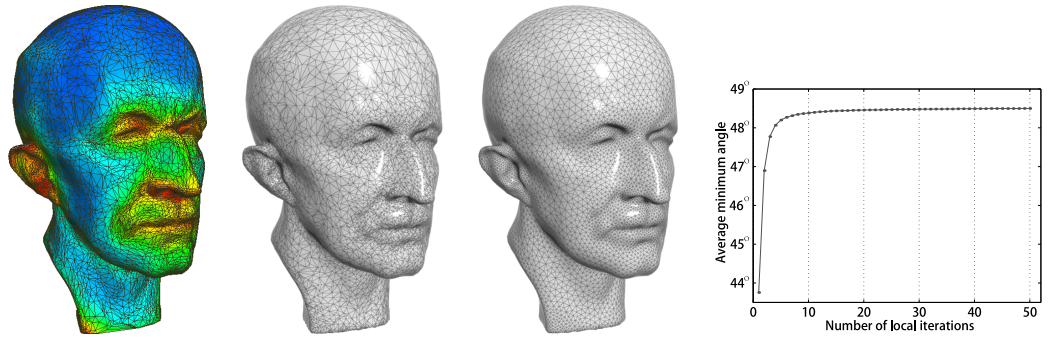
The local optimization of CCDM algorithm has been given in Algorithm 1 (see also Figure 2). The algorithm starts with an initial remesh of the reference mesh surface, and optimizes its connectivity and geometry alternately.

Building Delaunay Mesh: The edge-flipping algorithm for the Delaunay mesh construction proposed by [DZM07a] takes any manifold triangular mesh as input. An edge is locally Delaunay if the sum of the two angles opposite to it does not exceed π. Any edge that is not locally Delaunay, called NLD edge for short, is flipped. Each NLD edge is assigned a priority value, computed as the sum of the opposite angles at the edge minus π. A priority queue is built to maintain the ordering of edge flipping. The feature edges, if any, are not flipped.

Since the sequence of meshes to be made Delaunay in our optimization process evolves gradually, only a few edges of the intermediate meshes are NLD. The edge flip algorithm works quite fast in our experiments. It is known that a planar triangulation can be made Delaunay in $O(n^2)$ edge flips in the worst case. Recently, Cheng and Jin [CJ11] prove that any planar triangulation with a constant angle lower bound can be made Delaunay in $O(n)$ time. This also helps to explain why flip algorithms often run fast in practice.

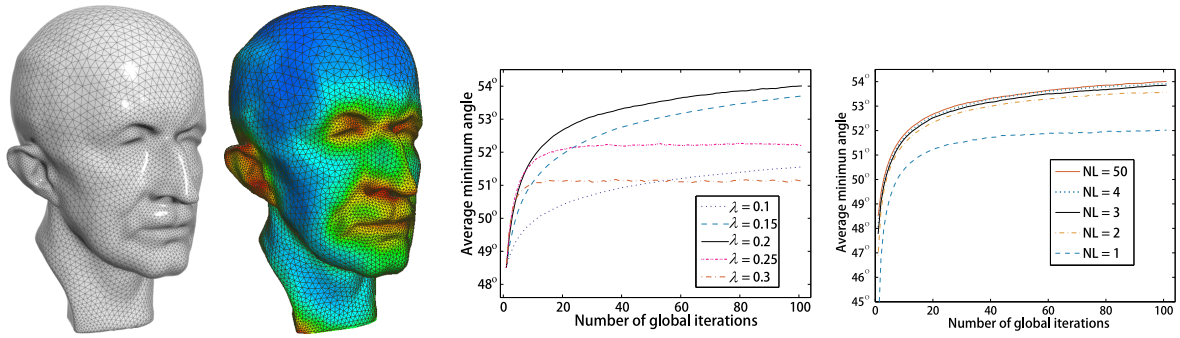
Vertex Relocation: All the vertices of the evolving mesh $\mathcal{M}_O$ are constrained on the reference mesh $\mathcal{M}_I$. The position of the vertex $\mathbf{x}_i \in \mathcal{M}_O$ is coded by barycentric coordinates with respect to a triangle τ in $\mathcal{M}_I$. And τ is called the reference triangle of $\mathbf{x}_i$. A density is assigned to each $\mathbf{x}_i$ according to the density function defined on the reference mesh $\mathcal{M}_I$. Then, the centroid of the 1-ring neighbor patch $\Omega_i \subset \mathcal{M}_O$ is computed by summing up the centroids of the individual triangles, weighted with the triangle mass. This obtains a new position of $\mathbf{x}_i$ which needs to be projected onto $\mathcal{M}_I$.

A brute force method to find the foot point of $\mathbf{x}_i$ on $\mathcal{M}_I$ by traversing all triangles is inefficient. Actually, the foot point of $\mathbf{x}_i$ should be located at some triangle covered by Ω_i for preventing any triangle flips. Any vertex with new position outside Ω_i will be pulled back to its original position. Fur-



(a) (b) (c) (d)

Figure 3: *Local optimization of CCDM method. (a) The original mesh (10k vert.) with visualized density function; (b) initial adaptive sampling and meshing (10k vert.); (c) the remeshing result with 50 iterations of local optimization; (d) average minimum angle graph.*



(a) (b) (c) (d)

Figure 4: *Global optimization of CCDM method, where NL and NG represent the numbers of local iteration and global iteration, respectively. (a) The remeshing result with 100 global iterations; (b) the remeshing result in (a) with visualized density function; (c) optimization with different λ's (NL = 50 and NG=100); (d) optimization with different NL's (λ = 0.2, NG = 100). Due to the stochastic factor in the optimization, we run the global optimization 10 times for each setting, and plot the average graphes.*

thermore, based on the fact that the position deviation of a vertex in each iteration is usually small, only local search in its reference triangle and the triangles incident it is necessary. Thus, the projection operation can be finished in constant time.

The vertices on feature curves are only allowed to move along the feature curves , while the corners will be fixed during the optimization. We still apply the position update formula (3) to the feature vertices, despite the incomplete 1-ring neighbor patches of the boundary vertices. This will lead to consistent vertex distributions on the surface and feature curves.

Example: In Figure 3, the Plank model is resampled to 10k vertices and optimized with 50 iterations of local optimization in 1.78 seconds. From Figure 3(d), we can see that the local optimization method converges very soon, and the average minimum angle increases little after the first 5 iterations.

3.4. Global Optimization

The example in Figure 3 shows that the local optimization method tends to get stuck at a poor local minimum very soon. In order to get high-quality remeshing results, we apply the simulated annealing method [KGJV83](SA method for short), a stochastic global optimization algorithm.

The SA method, like other global optimization methods, is generally computationally expensive. It starts from an initial state, and then repeatedly move to a randomly chosen neighbor of the current state. The SA method usually need to test thousands of states before it reaches the global minimum.

To accelerate the global search of SA method, we propose an efficient method for generating candidates of neighboring states, by combining the local optimization method (see Figure 2). The modified SA method only searches in the space of local minimizers. At each iteration, it moves to another local minimizer, only when it reaches a better state.

We use the operation "perturb-optimize" to find the neighbors of the current local minimizers. Operation "perturb" to

each vertex is specified by the following equation:

$$\mathbf{x}_i^* = \mathbf{x}_i + \lambda l_i (u\cos 2\pi v, u\sin 2\pi v)[\mathbf{t}_1, \mathbf{t}_2]^T \qquad (4)$$

where l_i is the average length of the edges adjacent to vertex $\mathbf{x}_i$, u and v are two random numbers in $[0,1]$, and $\mathbf{t}_1, \mathbf{t}_2$ are two mutually orthogonal unit vectors in the tangent plane of $\mathbf{x}_i$. Here λ indicates the magnitude of the perturbation.

As a matter of fact, the effectiveness of SA method on a given problem heavily depends on the choices of the parameters in the algorithm. Unfortunately, there is no general rule for choosing these parameters properly; they depend on the nature of problem. In particular, λ is the only parameter to be set in the SA algorithm in this paper, which is independent on the size of input mesh. If λ is too small, the search will very likely reach the same minimizer, and the computation effort will be wasted. If λ is too large, it would amount to re-starting optimization all over with a random initialization. From plenty of experiments, we find that $\lambda = 0.2$ is a good choice that makes the SA algorithm efficient for all the test models in this paper. As an illustration, we test our global optimization method on Plank model with different λ's and show the results in Figure 4. The local iteration number in these tests is set to 50 ($NL = 50$) to guarantee that a local minimizer is reached in each global iteration, although in the later discussion we will show that NL can be set to a much smaller number. We plot the average minimum angle graphs in Figure 4(c), from which we can observe that $\lambda = 0.2$ works well. The average minimum angle of the mesh is greater than 54^o after 100 global iterations, and the remeshed surface as shown in Figure 4(a) is visually much more regular than the local optimization result in Figure 3(c).

Note that, the local optimization method converges very quickly as shown in Figure 3(d). On the other hand, the local minimizer obtained from each inner loop is just an intermediate result used as the input of the next iteration step of the outer loop. Therefore, a highly accurate local minimizer would not be necessary. To illustrate this point, we again apply our algorithm on the Plank model by running 100 global iterations (i.e., $NG = 100$ and $\lambda = 0.2$) and setting NL to 1, 2, 3, 4 and 50, respectively. As shown in Figure 4(d), the global optimization method can still achieve the comparable performance when $NL \geq 2$. Therefore, we can stop the inner loop in few steps instead of running it to exhaustion to make the global optimization more efficient. To reach a good balance between the quality of the remeshing results and efficiency, we set $NL = 3, NG = 40$, and $\lambda = 0.2$ in all the examples in the rest of this paper.

4. Experimental Results

We implement the presented CCDM-based remeshing framework in C++. All the experiments are conducted on a laptop with a 2.66 GHz Intel processor and 4GB memory.

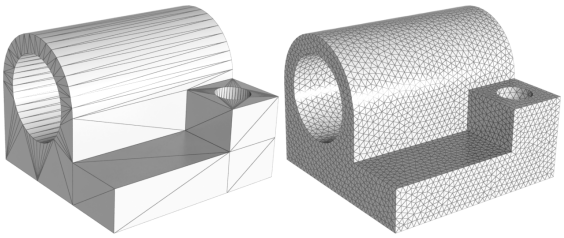


Figure 5: *The Joint model with sharp features (left) remeshed with 6k vertices (right).*

Figure 6: *The noisy Venus model (left) and the remeshed surface with 20k vertices (right).*

We have extensively tested the proposed algorithm on models with diverse characteristics. Our first example shown in Figure 5 is a uniform remeshing of a CAD model with sharp features and nearly degenerate triangles. The sharp features are simply detected according to each edge's dihedral angle, and are preserved in the resulting model by our algorithm. In fact, remeshing models with many nearly degenerate triangles is in general a challenge for parameterization-based remeshing methods [SAG03, FAKG10], since severe distortions will unavoidably be introduced by the parameterization. With the Joint model, which contains many nearly degenerate triangles, we demonstrate the robustness of our method.

Figure 6 demonstrates the efficacy of our remeshing method on the second model corrupted with synthetic noise. The result reveals the ability of our method in remeshing even in the presence of extreme amounts of noise. Figure 7 shows an adaptive remeshing of the third model with multiple boundaries. As can be seen in the remeshing result in Figure 7(right), the distribution of vertices on the mesh surface is consistent with that on the boundaries, owing to that we apply the same position update rule for all vertices. Input models with self-intersections would be problematic for CVT-based remeshing method [YLL*09], which approximates the geodesic distance on surface with the Euclidean distance. The fourth model in Figure 8 is a self-intersecting mesh – the Klein bottle, and the remeshing re-

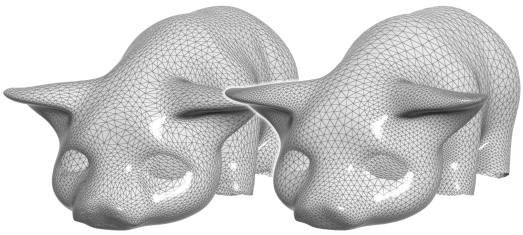


Figure 7: *The adaptive remeshing of the Pig model with multiple boundaries.*

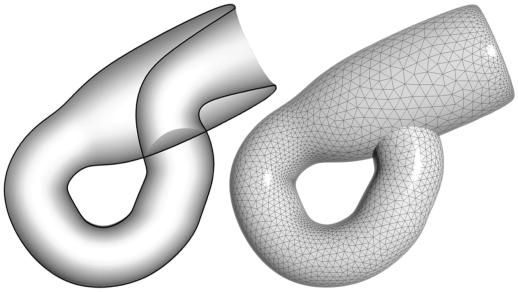


Figure 8: *The Klein bottle model with self-intersection (left), and the remeshing result (right), homeomorphic to the input mesh.*

sult shows that our method works well on input meshes with self-intersections.

Evaluation and Comparison: A statistical analysis of the remeshing results and the runtime of each tested method are given in Table 1. The analyses of the remeshing qualities include the angles and aspect ratio of each triangle, and the approximation error of mesh, which are most commonly used in the remeshing literature. As we can see in Table 1, our remeshing method is capable of generating meshes efficiently with well-shaped triangles ($\angle$Avg$\approx 53^o$).

In Figure 9, the remeshing results from our method, the relaxation-based method [BK04], and the three CVT-type methods (the vertex clustering based method [VCP08], the exact RVD computation based method [YLL*09] and the local parameterization based method [FAKG10]) are compared on the Moai model. The performance statistics of these five algorithms are given in Table 1. All timings of the competing methods are obtained from running the code provided by the authors of the related work on the same machine. And the parameters for each method are also set to the values suggested in the related paper. Specifically, for the method in [BK04], we performed the relaxation iteration 20 times; for the method in [VCP08], we subdivided the input mesh until its vertex number is 50 times the vertex budget; for methods in [YLL*09, FAKG10], we always applied 100 Lloyd or L-BFGS iterations. We can see that our algorithm runs in comparable times to the fast clustering method in [VCP08], and achieves comparable mesh quality to the method in [YLL*09]. The method in [BK04] combines vertex relocation with topological operations in each iteration, which is very similar to our local optimization method. By incorporating global perturbation, our method generates meshes with higher qualities at the cost of relatively long running times as compared to the method in [BK04].

In Figure 10, we show more remeshing results by our method, and the statistics of mesh qualities are given in Table 2.

Limitation: Despite the fast convergence of the two-step iteration of the local optimization process in practice, it is not theoretically guaranteed. We have shown that our method is capable of generating meshes efficiently with well-shape triangles, but a certified minimum angle is not given. The vertex relocation operations in our method improve the quality of the remeshed surface progressively with no regard for reducing the approximation error between the original mesh and the result mesh. Therefore, theoretically there is no guarantee that our method can capture all the small features on meshes. And it sometimes leads to relatively large approximation errors (see Table 1&2). A possibility to overcome this limitation can be to take into consideration the quadric error metrics [GH97] when evaluating the 'optimal' vertex position. We leave this as a future research. Also, further studies on the relation between the user-specified probability density function and the density function used in CCDM energy function are required to ensure the precise density adaptation.

5. Conclusion

We described a relaxation-based algorithm for remeshing an input mesh according to a given density function. This algorithm depends on the concept of the constrained centroidal Delaunay mesh, which can be constructed by alternating Delaunay mesh construction and relocation of the vertices to the centroids of their respective 1-ring neighbor patch. We then embed this simple iterative process in an efficient local/global optimization framework, leading to high-quality meshes with the same topology as the original meshes.

Future Work: In our remeshing framework, we have only considered the vertex relocation rule in Equation (3), but the general algorithm is applicable to other relocation strategies, such as the position update rule from optimal Delaunay triangulation method [CX04]. A systematic comparison between different relocation rules will be useful to understand different mesh configurations and their respective energy functions. Although our algorithm is efficient, it can be further accelerated by parallelizing computation framework, especially the construction of Delaunay mesh. In another perspective, the extension of our framework to anisotropic remeshing problem is also an interesting direction.

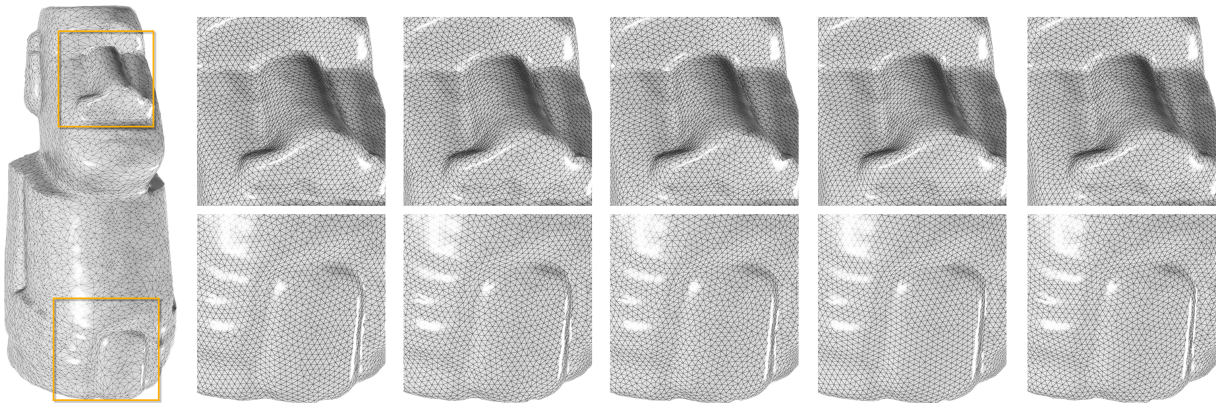


Figure 9: *Comparison of remeshing results of the Moai model between different methods. From left to right: original model, the results from our method, [BK04], [VCP08], [YLL*09], and [FAKG10]. The statistics for the resulting meshes are summarized in Table 1.*

Model (#vert)	Method	Time (sec)	∠ (deg) Avg	∠ (deg) Min	$< 30^o$ (%)	Aspect Ratio Avg	Aspect Ratio Min	Error (10^{-3})
Joint (221 – 6,000)	Ours	3.64	53.41	29.89	2.71×10^{-2}	0.923	0.609	1.34
	[YLL*09]	66.12	53.51	24.95	6.38×10^{-2}	0.920	0.566	1.56
Venus (134,359 – 20,000)	Ours	17.61	53.39	24.41	4.16×10^{-3}	0.924	0.552	5.39
	[VCP08]	10.52	49.09	22.84	4.17×10^{-3}	0.861	0.409	4.05
Pig (8,000 – 10,000)	Ours	7.17	52.95	28.65	5.57×10^{-2}	0.919	0.559	3.87
	[FAKG10]	73.35	52.23	20.24	9.73×10^{-2}	0.903	0.387	3.09
	[VCP08]	5.55	45.79	13.76	0.15	0.871	0.254	2.34
Klein Bottle (25,800 – 5000)	Ours	3.97	53.19	32.65	0	0.921	0.642	4.60
Moai (10,002 – 30,000)	Ours	21.35	54.21	38.22	0	0.934	0.645	3.72
	[BK04]	4.12	51.93	33.16	0	0.907	0.632	3.21
	[VCP08]	15.99	50.98	31.51	0	0.886	0.538	2.94
	[YLL*09]	263.16	54.80	38.79	0	0.939	0.636	2.80
	[FAKG10]	157.09	53.62	38.02	0	0.922	0.638	3.21

Table 1: *Comparison of running times and meshing qualities. The vertex numbers before and after remeshing are listed below the model names. ∠Avg is the average of the minimum angle in each triangle. ∠Min is the smallest angle in the mesh. $< 30^o$ is the percentage of angles smaller than 30^o. The measurement of the aspect ratio of a triangle is from [FB99]. The error is the Hausdorff distance between the original model and the remeshed model with respect to the bounding box diagonal.*

Acknowledgments

The work of Wenping Wang was partially supported by the National Basic Research Program of China (2011CB302400), the Research Grant Council of Hong Kong (718209 and 718010), and the State Key Program of NSFC project (60933008). Zhonggui Chen and Juan Cao were supported by NSFC (61100105 and 61100107) and Natural Science Foundation of Fujian Province of China (2011J05007 and 2012J01291). The surface models are courtesy of Aim@Shape.

References

[AdVDI03] ALLIEZ P., DE VERDIRE E. C., DEVILLERS O., ISENBURG M.: Isotropic surface remeshing. In *Proc. Shape Modeling Int* (2003), pp. 49–58. 2, 4

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive Geometry Remeshing. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference 21(3)* (2002), 347–354. 2

[AUGA08] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring*. Springer Berlin Heidelberg, 2008, pp. 53–82. 1, 2

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Symposium on Geometry Processing* (2004), pp. 189–196. 2, 7, 8

[BS07] BOBENKO A., SPRINGBORN B.: A discrete Laplace-Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38* (2007), 740–756. 3

[CH11] CHEN L., HOLST M.: Efficient mesh optimization schemes based on optimal delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering 200*, 9–12 (2011), 967–984. 1, 2, 3

[CJ11] CHENG S.-W., JIN J.: Edge flips and deforming surface meshes. In *Proceedings of the 27th annual ACM symposium on Computational geometry* (2011), pp. 331–340. 4

[CX04] CHEN L., XU J.: Optimal Delaunay triangulations. *Journal of Computational Mathematics 22*, 2 (2004), 299–308. 2, 7

[DFG99] DU Q., FABER V., GUNZBURGER M.: Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review 41* (1999), 637–676. 3

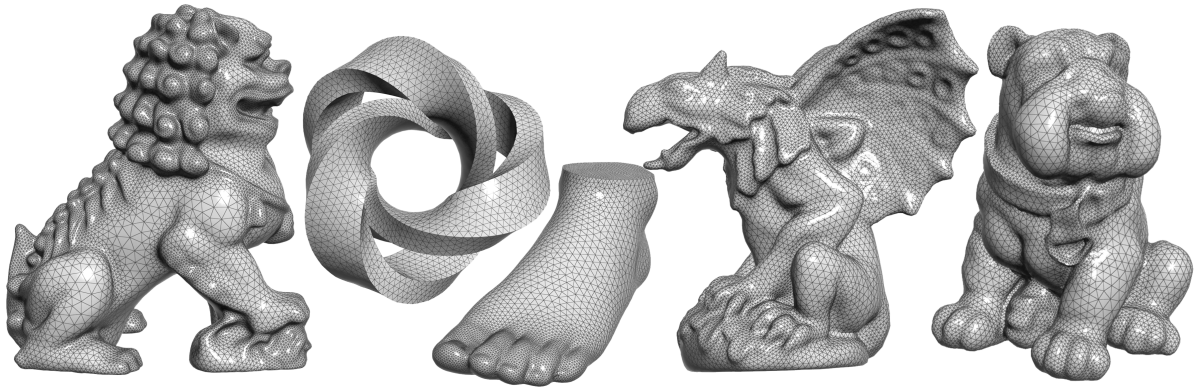


Figure 10: *More remeshing results. The input meshes are shown in the supplementary material.*

Model	#Vert (before – after)	Time (sec)	∠ (deg) Avg	∠ (deg) Min	$< 30^o$ (%)	Aspect Ratio Avg	Aspect Ratio Min	Error (10^{-3})
Lion	131,056 – 30,000	23.56	53.45	32.79	0	0.923	0.639	3.87
Twist	800 – 5,000	3.14	53.75	35.52	0	0.929	0.644	3.56
Foot	10,016 – 6,000	3.95	53.82	24.24	2.01×10^{-2}	0.928	0.561	2.00
Gargoyle	39,318 – 18,000	15.62	53.34	28.14	9.26×10^{-4}	0.924	0.621	4.51
Dog	158,297 – 10,000	8.25	52.78	26.65	1.33×10^{-3}	0.917	0.574	4.67

Table 2: *The statistics of the remeshing results in Figure 10.*

[DGJ02] DU Q., GUNZBURGER M. D., JU L.: Constrained centroidal Voronoi tessellations for surfaces. *SIAM J. Sci. Comput. 24*, 5 (2002), 1488–1506. 2

[DLS10] DEY T. K., LEVINE J. A., SLATTON A.: Localized delaunay refinement for sampling and meshing. *Computer Graphics Forum 29*, 5 (2010), 1723–1732. 2

[DR10] DEY T. K., RAY T.: Polygonal surface remeshing with Delaunay refinement. *Engineering with Computers 26*, 3 (2010), 289–301. 2

[DZM07a] DYER R., ZHANG H., MÖLLER T.: Delaunay mesh construction. In *Proceedings of the Symposium on Geometry Processing* (2007), pp. 273–282. 3, 4

[DZM07b] DYER R., ZHANG H., MÖLLER T.: Voronoi-Delaunay duality and Delaunay meshes. In *Proceedings of the Symposium on Solid and Physical Modeling* (2007), pp. 415–420. 3

[ES94] EDELSBRUNNER H., SHAH N. R.: Triangulating topological spaces. In *Proceedings of the Symposium on Computational Geometry* (1994), pp. 285–292. 3

[FAKG10] FUHRMANN S., ACKERMANN J., KALBE T., GOESELE M.: Direct resampling for isotropic surface remeshing. In *VMV 2010: Vision, Modeling & Visualization* (2010), pp. 9–16. 1, 2, 4, 6, 7, 8

[FB99] FREY P. J., BOROUCHAKI H.: Surface mesh quality evaluation. *International Journal for Numerical Methods in Engineering 45*, 1 (1999), 101–118. 8

[FZ08] FU Y., ZHOU B.: Direct sampling on surfaces for high quality remeshing. In *Proceedings of the Symposium on Solid and Physical Modeling* (2008), pp. 115–124. 2, 4

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 209–216. 7

[KGJV83] KIRKPATRICK S., GELATT C. D., JR., VECCHI M. P.: Optimization by simulated annealing. *Science 220* (1983), 671–680. 5

[LCT11] LIU Y.-J., CHEN Z., TANG K.: Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces. *IEEE Trans. Pattern Anal. Mach. Intell. 33* (2011), 1502–1517. 4

[PC06] PEYRÉ G., COHEN L. D.: Geodesic remeshing using front propagation. *Int. J. Comput. Vision 69* (2006), 145–156. 2

[SAG03] SURAZHSKY V., ALLIEZ P., GOTSMAN C.: Isotropic remeshing of surfaces: a local parameterization approach. In *Proceedings of 12th International Meshing Roundtable* (2003), pp. 215–224. 2, 6

[SG03] SURAZHSKY V., GOTSMAN C.: Explicit surface remeshing. In *Proceedings of Eurographics Symposium on Geometry Processing* (2003), pp. 17–28. 2

[SSG03] SIFRI O., SHEFFER A., GOTSMAN C.: Geodesic-based surface remeshing. In *In Proc. 12th Intnl. Meshing Roundtable* (2003), pp. 189–199. 2

[Tur92] TURK G.: Re-tiling polygonal surfaces. *ACM SIGGRAPH Computer Graphics 26*, 2 (1992), 55–64. 2, 4

[TWAD09] TOURNOIS J., WORMSER C., ALLIEZ P., DESBRUN M.: Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28*, 3 (2009), 75:1–75:9. 2

[VCP08] VALETTE S., CHASSERY J. M., PROST R.: Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics 14* (2008), 369–381. 2, 7, 8

[VRS03] VORSATZ J., RÖSSL C., SEIDEL H.-P.: Dynamic remeshing and applications. In *Proceedings of the Symposium on Solid Modeling and Applications* (2003), pp. 167–175. 2

[YLL*09] YAN D.-M., LÉVY B., LIU Y., SUN F., WANG W.: Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. In *Proceedings of the Symposium on Geometry Processing* (2009), pp. 1445–1454. 2, 6, 7, 8