Vision, Modeling, and Visualization (2010)

Direct Resampling for Isotropic Surface Remeshing

Simon Fuhrmann, Jens Ackermann, Thomas Kalbe and Michael Goesele

TU Darmstadt

Abstract

We present a feature-sensitive remeshing algorithm for relaxation-based methods. The first stage of the algorithm creates a new mesh from scratch by resampling the reference mesh with an exact vertex budget with either uniform or non-uniform vertex distribution according to a density function. The newly introduced samples on the mesh surface are triangulated directly in 3D by constructing a mutual tessellation. The second stage of the algorithm optimizes the positions of the mesh vertices by building a weighted centroidal Voronoi tessellation to obtain a precise isotropic placement of the samples. We achieve isotropy by employing Lloyd's relaxation method, but other relaxation schemes are applicable. The proposed algorithm handles diverse meshes of arbitrary genus and guarantees that the remeshed model has the same topology as the input mesh. The density function can be defined by the user or derived automatically from the estimated curvature at the mesh vertices. A subset of the mesh edges may be tagged as sharp features to preserve the characteristic appearance of technical models. The new method can be applied to large meshes and produces results faster than previously achievable.

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

1. Introduction

Many computer graphics applications have to deal with surfaces for various reasons. A very popular discrete surface representation is the triangle mesh which is simple to handle and can be rendered efficiently by modern graphics hardware. Modelling a surface in CAD applications as well as digitizing real-world objects typically results in a triangle mesh at some point. These different sources lead to meshes with strongly varying characteristics. For example CAD meshes often have few and skinny triangles to represent a certain surface as exact as possible, whereas meshes from an acquisition pipeline usually contain many small triangles as result of the scanning process. Depending on the target application such as rendering, numerical simulations, transmission, or compression, meshes often need to undergo complete remeshing prior to usage. Important desired properties are, e.g., the complexity of the mesh in terms of number of vertices, the regularity of the connectivity, the quality of the triangle shape, and the sample distribution on the surface.

The goal of this work is to provide a simple but flexible remeshing framework that is capable of handling input meshes with diverse characteristics and allows for automatic generation of high-quality triangle meshes. The output of our remeshing algorithm is a new triangle mesh with uniform or adaptive vertex distribution, isotropic sampling and almost equilateral triangles; thus our method excels in the area of high-quality remeshing. The process is guided by specifying various parameters, for example the number of vertices in the remesh, the degree of adaptiveness to surface curvature, and whether remeshing should preserve surface features such as creases and sharp corners.

We argue that an optimal remeshing approach should be

- *fast and efficient* to be able to handle large meshes,
- *simple but effective* for ease of implementation and to produce high-quality results,
- *general and robust* to handle meshes of arbitrary genus with any amount of boundaries, and
- *accurate* so that the resulting mesh is as close as possible to the input mesh and the vertex sampling follows the prescribed density function.

As shown in the following section, none of the currently available remeshing approaches fulfill all of these criteria.

DOI: 10.2312/PE/VMV/VMV10/009-016

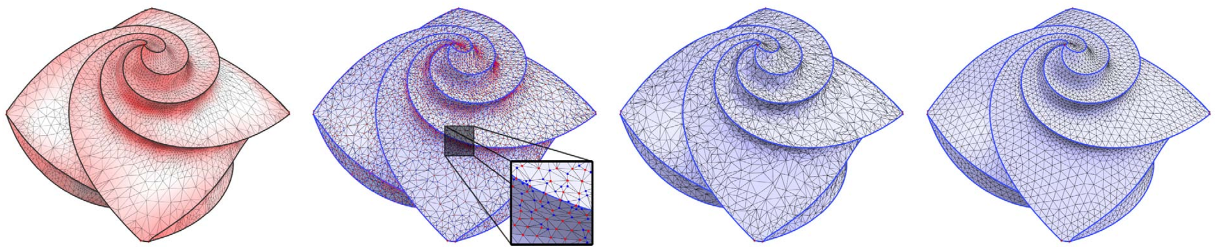

Figure 1: *Stages of the remeshing algorithm: The final vertex distribution is controlled by a density function defined over the original mesh (left). The mesh is resampled and meshed using a mutual tessellation (middle left) that contains both, old vertices (red) and the new vertices (blue). Old vertices are deleted (middle right) and Lloyd relaxation is applied to achieve an isotropic sample distribution (right), which results in well-shaped triangles.*

We therefore introduce a novel remeshing algorithm that is designed to meet all these criteria. In particular,

- it employs a direct resampling strategy and does not rely on global parametrization for meshing,
- it is applicable to large meshes since resampling provides a beneficial initial vertex distribution,
- it is fast because we use efficient algorithms and CPU parallelization in the relaxation framework, and
- it is accurate because it always refers to a geometric reference during remeshing.

The algorithm can be roughly divided into three stages: preprocessing, resampling, and precise vertex placement (see Figure 1). In the first stage, the mesh curvature is estimated and features are extracted. The second stage creates a new mesh with an exact vertex budget and proper sample distribution. The algorithm proceeds with the third stage that produces a precise isotropic placement of the samples by constructing a weighted centroidal Voronoi tessellation (WCVT).

This paper is organized as follows: Section 2 gives an overview of related work. Section 3 discusses preliminaries and shortly explains the pre-processing step. Section 4 describes how the reference mesh is resampled and robustly meshed to build an initial mesh. The new vertices are then re-distributed to achieve a precise isotropic placement as explained in Section 5. We present results in Section 6 and conclude with a discussion and thoughts about future work in Section 7.

2. Related Work

Remeshing of surfaces is an evolving field with a long history. It is used in many applications, e.g., in mesh editing, animation, simulation, compression, or for the generation of progressive meshes with levels of detail. Creating high-quality meshes strives to achieve two goals: *isotropy*, i.e., almost equilateral triangles and an overall *good vertex distribution*, typically adapted to surface curvature. High-quality meshes can be created by constructing the centroidal Voronoi tessellation (CVT) on the surface of the mesh. The CVT is a Voronoi tessellation whose vertices, or *sites*, are centroids of their corresponding Voronoi cells [DFG99]. In a weighted CVT (WCVT) the sites coincide with the *center of mass* rather than the geometric centroid. Such a tessellation can be constructed by applying Lloyd's relaxation method [Llo82] to the sites. Due to its slow convergence, it requires, however, a good initial placement of the samples.

Alliez et al. [AdVDI03] proposed an isotropic remeshing algorithm that first distributes a given number of vertices over the mesh surface. Processing is, however, performed in a global parameter domain. Such a global parametrization may not exist in general and involves many delicate problems, such as cutting closed and high-genus meshes, numerical instabilities, and the final lifting phase which brings the results back to the 3D domain.

To avoid the global parametrization of the input mesh, Surazhsky et al. [SAG03] proposed an isotropic remeshing algorithm based on their local parametrization approach [SG03]. The method first brings the mesh to the required amount of vertices using edge-collapse and vertex-split operations. Lloyd relaxation is then applied in a framework that parametrizes only a small part of the reference mesh required to relocate a vertex to the centroid of its associated Voronoi cell. One drawback is that the mesh, once at the exact vertex budget, may not necessarily exhibit a sampling density that complies with the density function. Since this makes Lloyd's iteration practically infeasible, Surazhsky et al. propose an area-equalization procedure with much faster convergence to approximate the required sample distribution. However, this approximation still uses local vertex relocations, may get stuck in local optima and does not give any guarantees on the produced sampling.

To alleviate the problems with these two approaches, Fu et al. [FZ09] propose to first apply a Poisson disc sampling on the input mesh and then apply the relaxation framework from [SAG03]. While this approach achieves excellent local distribution after sampling, it is computationally extremely costly even for meshes with only a few thousand vertices. Yan et al. [YLL*09] avoid explicit 2D parametrization. Instead, they repeatedly build the Restricted Voronoi

Diagram for the mesh surface and apply a quasi-Newton optimization method which achieves faster convergence than Lloyd relaxation alone. The approach tolerates input meshes with heavily degenerated triangles which is problematic for parametrization-based methods; they back-project samples to the surface but consequently need to obey sampling theorems to preserve homeomorphism. The proposed method is strictly tailored towards building the CVT and does not support other relaxation schemes.

Similar to Fu et al. [FZ09], we aim at combining the advantages of Alliez et al. [AdVDI03] with the local parametrization framework as proposed by Surazhsky et al. [SAG03] to construct the CVT directly on the mesh. We argue, however, that the precise initial *local* sample distribution provided by the Poisson disc sampling is not necessary. Similar distributions can quickly be achieved using a few Lloyd iterations. Our algorithm is significantly faster and therefore applicable to much larger meshes than the approach by Fu et al. [FZ09].

3. Preliminaries

The input to the remeshing framework is an orientable 2-manifold triangle mesh $\mathcal{M}_o$ of arbitrary genus, any number of boundaries and possibly multiple connected components. Unless normals are provided with the input data, we estimate them using tessellation invariant angle-weighted pseudo-normals [TW98]. We consider the input mesh to approach a C^1-continuous surface everywhere except at boundaries and feature-tagged edges. Feature edges receive special treatment during remeshing and will be preserved in the final mesh $\mathcal{M}_f$. The vertex distribution of $\mathcal{M}_f$ will be either uniform or comply with a density function defined over the reference mesh $\mathcal{M}_o$.

To maintain fidelity to the reference mesh during relaxation, the vertices of $\mathcal{M}_f$ are restricted to the surface of $\mathcal{M}_o$ at all times: A *barycentric coordinate* $b_T = (b_1, b_2, b_3)$, $b_i \geq 0$, $\sum b_i = 1$ with respect to a triangle $T = \triangle_{v_1,v_2,v_3}$ uniquely defines a position $v = \sum b_i v_i$ on T. We use the notation $(b,\, T)_\mathcal{M}$ to refer to any position on mesh $\mathcal{M}$ and call that pair a *barycentric reference* onto $\mathcal{M}$.

3.1. Pre-Processing

To make the method applicable to technical data sets, e.g. models of mechanical parts which typically contain sharp edges, a set of features may either be specified by the user or procedurally extracted. We use a naive thresholding of the dihedral angle but more robust techniques can be applied [JH02]. We also consider boundary edges (with only one adjacent face) as feature edges and handle this case equally. A feature skeleton is built by chaining all these edges together; such a skeleton may contain both open and closed backbones, where open backbones are terminated by corner vertices and edges of closed backbones form a loop.

A density function can be specified or estimated from the mesh geometry. We use simple formulas from Dyn et al. [DHKL01] to approximate the Gaussian and absolute mean curvature at the mesh vertices, combine and clamp them to remove outliers and apply a contrast exponent γ to control the degree of adaptiveness. Gradation of the density function is influenced by iteratively applying a weighted Laplacian smoothing operator on the density values. See Alliez et al. [AdVDI03] for more details on how density gradation influences the final mesh.

The mesh is then prepared for resampling by applying a simple and fast algorithm by Sander et al. [SNB07] which re-arranges the triangles of the mesh to improve vertex cache efficiency. We will show later why this is a useful property for our resampling procedure. Finally, the input mesh $\mathcal{M}_o$ is taken as reference mesh for the remeshing algorithm and remains unchanged for the rest of the pipeline.

4. Building the Initial Mesh

We aim at distributing a user-defined amount of samples over the mesh surface such that the sample distribution complies with the prescribed density function. The samples are first partitioned between the surface and the feature skeleton. Both parts are then processed separately: The surface is sampled by drawing random barycentric coordinates for each triangle. The feature skeleton is rebuilt from scratch by accurately sampling the backbones according to the density function. The vertex positions of the new skeleton are exact and remain unchanged during remeshing. The new samples from both the smooth parts and the skeleton are finally meshed together using a mutual tessellation [Tur92].

4.1. Sample Partitioning

We first integrate the density function over the surface and the feature edges and obtain two mass quantities D_s and D_f. In the uniform case, this corresponds to the area and length of the surface and the features, respectively. To partition the vertex budget S between the surface and the features, we apply formulas from Alliez et al. [AdVDI03] and obtain the amount of samples S_s and S_f to distribute on the surface and the feature skeleton.

4.2. Triangle Sampling

To sample the surface, we traverse the triangles and deduce the number of samples S_T for each triangle T, $S_T = \frac{S_s}{D_s} \cdot D_T$, where D_T is the mass of T. We round S_T to the nearest integer. This creates a signed quantization error, which is propagated to the next unprocessed face in order.

The task of traversing all triangles for sampling raises the question of a suitable *processing path* over the mesh. One could use a random ordering of the triangles which produces reasonable results with minimal effort, but teleports the local quantization residual to arbitrary locations. Alliez et al. [AdVDI03] generalized the concept of error diffusion to obtain a connected processing path over the mesh triangles. In this method, the local quantization error is diffused to adjacent faces which keeps the error local. In contrast, we decided to take advantage of the fast reordering algorithm from Sander et al. [SNB07] to optimize the triangle ordering for spatial locality in the pre-processing step. This creates an adequate flow over the mesh triangles and delegates the local quantization residual to the next face in processing order. In contrast to the error diffusion approach, we do not need to store the propagated error for multiple unprocessed triangles at the processing boundary.

For efficiency we distinguish between non-uniform and uniform triangle sampling, see Figure 2.

Non-uniform triangle sampling: Given the density values $d_1, d_2, d_3 > 0$ at the corresponding vertices of T, we interpolate the density $d(b) = d_1 b_1 + d_2 b_2 + d_3 b_3$. Without loss of generality, we assume that T lies in the xy-plane. We normalize the density $\tilde{g} := d / \int_T d$ and use $\tilde{g}$ as joint distribution of b_1, b_2. We compute the marginal density g_1 and the cumulative density function (CDF) F_1

$$g_1(b_1) = \int_T \tilde{g}(b_1, b_2) \mathrm{d}b_2 \qquad F_1(x_1) = \int_0^{x_1} g_1(b_1) \mathrm{d}b_1.$$

The CDF is inverted by solving a cubic equation. We then draw a sample from a uniform distribution on $[0,1]$ and transform it into a sample b_1 with distribution g_1. With this sample, the conditional distribution of b_2 is given as $g_2(b_2 \mid b_1) = \frac{\tilde{g}(b_1,b_2)}{g_1(b_1)}$. We compute the CDF

$$F_2(x_2 \mid b_1) = \int_0^{\min(x_2, 1-b_1)} g_2(b_2) \mathrm{d}b_2$$

and solve the resulting quadratic equation to invert it. Again, a uniformly distributed sample is transformed into a sample b_2 with distribution $g_2(\cdot \mid b_1)$, see [HLD04] for details.

Uniform triangle sampling: To generate uniform random barycentric coordinates we draw two uniformly distributed random numbers $\tilde{x}_1$, $\tilde{x}_2$ in $[0,1]$. We reorder these values by assigning $x_1 = \min(\tilde{x}_1, \tilde{x}_2)$, $x_2 = \max(\tilde{x}_1, \tilde{x}_2)$, which leads to distributions with expected values $1/3$ and $2/3$ for x_1 and x_2, respectively. The lengths of the three intervals between $0, x_1, x_2, 1$ are then taken as barycentric coordinate $b = (x_1, x_2 - x_1, 1 - x_2)$. This yields samples with a uniform distribution over any triangle.

4.3. Skeleton Sampling

Instead of randomly sampling the feature skeleton and applying Lloyd relaxation in 1D to feature samples, we calculate the exact sample positions analytically. To emulate the WCVT for the 1D case, we place samples such that each sample is associated with the same amount of mass.

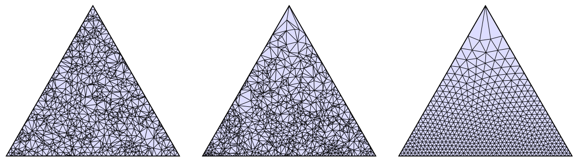
Figure 2: *Triangle sampling: Uniform (left), adaptive with increasing density from top to bottom (middle) and adaptive sampling after Lloyd relaxation (right).*

Once the total mass D_f (integrated density) of the feature skeleton and the amount of samples S_f to distribute on the skeleton is known, we calculate the optimal *sample spacing* R_f^{-1} (or more formally the *inverse sampling rate*) for the whole skeleton, expressed in mass per sample. The optimal mass for the samples of the skeleton are calculated as follows, where B_o is the amount of open backbones, S_f the amount of samples to distribute and C the amount of corner vertices in the skeleton:

$$R_f^{-1} = \frac{D_f}{B_o + S_f - C}$$

To proceed, we derive similar quantities for each backbone: To calculate the optimal amount of samples S_B for a backbone we divide the backbone mass by the sample spacing R_f^{-1}, round to the nearest integer, and obtain a non-fractional amount of samples to distribute on the backbone. This rounding creates a signed quantization error which is delegated to the next backbone in order. The optimal sample spacing per backbone R_{fb}^{-1} is then derived by dividing the backbone mass by the non-fractional sample amount.

We now aim at placing a sample every R_{fb}^{-1} mass on the backbone. We traverse the edges of the backbone in order, cumulate the mass and place a sample when R_{fb}^{-1} mass has been collected. The exact position of a sample within an edge of length l is then calculated by solving the following equation for $x_2 \in [0,l]$, where D_{left} is the remaining mass to collect from a known position x_1 to a yet unknown position x_2 (D_{left} is equal to R_{fb}^{-1} if the last sample was inserted on the same edge, otherwise D_{left} is the remainder from previous edges):

$$\int_{x_1}^{x_2} d(x)\, dx = D_{\text{left}} \quad \text{with} \quad d(x) = (1 - \frac{x}{l}) d_1 + \frac{x}{l} d_2$$

Note that $d(x)$ is the density function with density values d_1 and d_2 at the edge vertices, linearly interpolated over the edge with length l. The new sample is placed on the current edge at position x_2 and the process is repeated for the same edge but starting from x_2, possibly passing edge boundaries until all edges of the backbone are processed.

4.4. Meshing the Samples

The sampling process provides a set of surface and feature samples on the reference mesh surface $\mathcal{M}_o$. We must con-

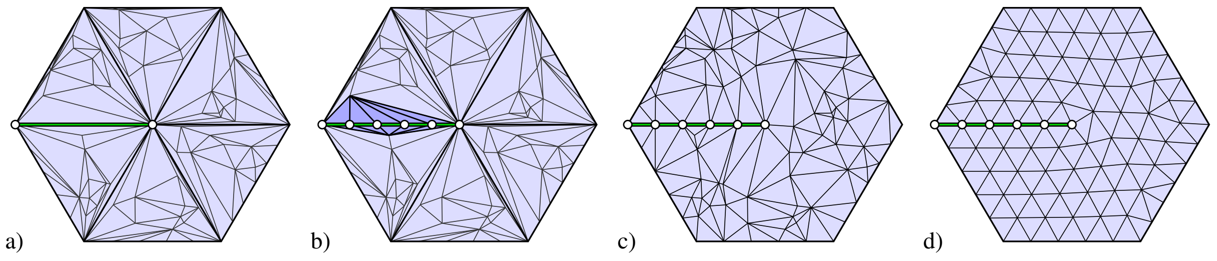

Figure 3: *Mutual tessellation with features: a) A Delaunay triangulation is constructed locally in each triangle of the original mesh. b) Feature edges are then resampled while inserting new triangles. c) Degeneracies are eliminated by flipping edges afterwards. d) Features remain stable during relaxation.*

nect these samples to obtain a valid triangle mesh consisting of the new samples only. Several surface reconstruction algorithms can generate connectivity information for a set of points. These techniques, however, do not incorporate the underlying connectivity information of $\mathcal{M}_o$ and cannot guarantee a topologically equivalent surface. Peyré and Cohen [PC06] exploit geodesic information to build a Voronoi diagram directly on the mesh and connect samples of neighboring Voronoi cells, but this is computationally expensive. We follow a more efficient approach and create a mutual tessellation [Tur92] of both, the original and the new vertices. The original vertices are deleted afterwards.

We start constructing the mutual tessellation by inserting samples into the triangles of the original mesh. An incremental Delaunay triangulation [GKS92] is employed using the mesh triangle as *dummy triangle at infinity*. New samples are generated by drawing random barycentric coordinates, as described in Section 4.2, until the desired amount of samples has been inserted. Each new sample is equipped with a barycentric reference $(b,T)_{\mathcal{M}_o}$ that tracks the exact position on the reference surface. To improve numerical stability, we bound new samples away from already existing samples and edges using ε-checks on the barycentric coordinates. If a sample violates the ε-condition, a new sample is drawn. Note that these checks are independent of the triangle size and thus do not affect the sampling density in regions with smaller triangles. This procedure creates a Delaunay triangulation within each triangle, preserves the edges of the original mesh, but introduces degenerated triangles around these edges, see Figure 3a.

After merging the surface samples with the mesh, the feature samples are inserted by splitting the skeleton edges of the original mesh at the positions calculated in Section 4.3. This is possible because no original edges have been modified in the previous step. Splitting edges may create additional degenerated triangles but is numerically stable: All operations are performed on the vertex positions only and not on possibly degenerated quantities such as triangle areas. Finally, after both surface and feature samples have been inserted into the mesh, a global constrained Delaunay triangulation is restored by flipping edges of the mesh if the angles opposite to the edge sum to more than 180°. This eliminates all degenerated triangles, see Figure 3c.

The mutual tessellation is then cleaned by deleting the original vertices. When deleting a vertex, all adjacent triangles are also removed from the mesh, which leaves a hole in the triangulation. This hole is re-triangulated directly in 3D using a procedure similar to Schroeder et al. [SZL92], and we merely check that newly introduced edges are not already present in the mesh. This guarantees that the mesh topology is preserved. It happens that some original vertices cannot be deleted if re-triangulation fails, thus the exact vertex budget is slightly off. Although this is typically not a big problem in practice, we address this issue by randomly deleting newly introduced vertices (but no feature vertices) until the vertex budget is reached.

Overall, the resampling procedure results in a new mesh with a sampling density that globally complies with the prescribed density function. The new vertices are equipped with proper vertex references that point to positions on $\mathcal{M}_o$. This information will be used as initialization for vertex relaxation to optimize the mesh.

5. Improving Vertex Positions

To improve the sampling of the new mesh, we construct a weighted centroidal Voronoi tessellation (WCVT). One way to do this is Lloyd's algorithm [Llo82]. The Lloyd relaxation is a simple iterative method that consists of three basic steps:

- Build the Voronoi diagram of the samples,
- move each sample to the centroid of its Voronoi cell,
- iterate the procedure until convergence.

Lloyd's algorithm is a particularly slow process with bad convergence behaviour, i.e., the main improvement is achieved in the first few Lloyd iterations and performance of convergence quickly decelerates. However, the initial vertex distribution obtained from the sampling process is *globally* correct and *locally* already a good approximation of the density function. This allows for effective, progressive improvement of the mesh with fast convergence.

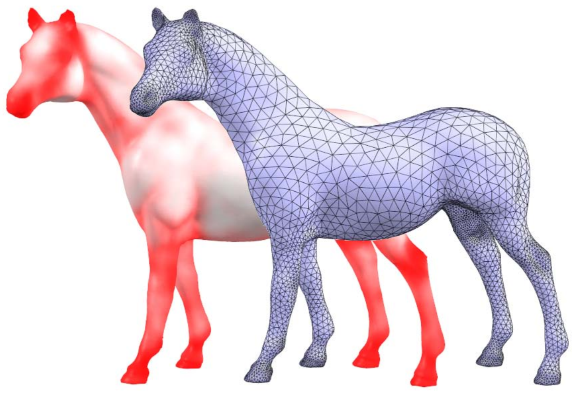

Figure 4: *The horse model with about 50k vertices and the remeshed model with 6k vertices.*

Building the WCVT: To construct the WCVT, we closely follow the approach of Surazhsky et al. [SAG03]: Instead of constructing the Voronoi diagram for the whole mesh at once, a Voronoi cell is created locally for each vertex, which is easily derived from the local Delaunay property. A single vertex v and the adjacent triangles are flattened and the Voronoi cell for v is constructed. A density value is assigned to v and to each vertex of the Voronoi cell polygon. The Voronoi cell is triangulated by connecting the polygon vertices with v, and the cell's centroid is calculated by summing the centroids of the individual triangles, weighted with the triangle mass. This yields a new position v^* in the planar domain and v is relocated to that position.

Vertex relocation: The task of relocating vertex v to v^* boils down to calculating a new barycentric reference $(b^*, T^*)_{\mathcal{M}_o}$ for v^*. To perform the relocation, a patch of $\mathcal{M}_o$ is created that contains a small region required to relocate the vertex, yet large enough to be reused for spatially close relocations. The method in [SG03] first gathers triangles of the reference mesh in a breath-first search to create a roundish patch in 3D. The patch is then flattened using a conformal parametrization with fixed circular boundary using Floater's Mean Value Coordinates [Flo03].

Optimizations: Surazhsky also proposed a caching scheme with fast patch lookup that keeps patches for a small amount of time, which drastically improves the performance of the approach. To make effective use of the caching system, we reorder the vertices of the mesh prior to relaxation. We extend the efficient triangle reordering algorithm from Sander et al. [SNB07] in the following way: Once triangles have been reordered, we traverse the triangles and issue the vertices in order of their first appearance. This creates a processing path over the mesh vertices with spatially low variance. The path allows subsequent relocations to reuse patches that have recently been created and reduces the average cache miss ratio for large meshes that require cache cleanups.

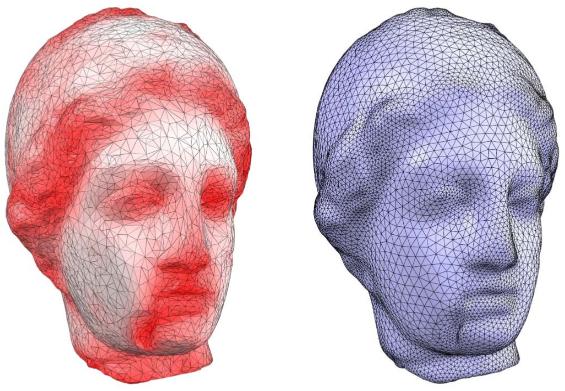

Figure 5: *The Hygieia model with about 8k vertices and visualized density function (left) has been upsampled to 10k vertices (right).*

Another observation is that Lloyd relaxation is easily parallelizable on the CPU by partitioning the vertices into equally large sets for processing in separate threads. Due to thread locking mechanisms, performance is not linear with the amount of threads and we achieved best results with 8 threads. We do not delete patches after time but threshold memory consumption to trigger cleanups, which is a more thread friendly approach. For each cleanup, cached patches are sorted by access time and a fixed amount of patches (20% in our experiments) with earliest access time is deleted from the cache. Experiments show that a cache of 256 MB is sufficient to handle meshes with about a million vertices without deleting any patch from the cache.

6. Results

Figure 4 shows an adaptive remesh of the horse model and a visualization of the mesh curvature used as density function for remeshing. The original model has about 50k vertices and was downsampled to 6k vertices. The algorithm took about 2 seconds for resampling and 5 seconds for 100 Lloyd iterations with 8 threads. Figure 5 shows an example where the Hygieia model was upsampled from 8k to 10k vertices. Figure 9 shows several more models where remeshing to 5k vertices was performed with almost interactive speed.

We also demonstrate our technique on mechanical models. Figure 6 shows a remeshing result of the Fandisk mesh that has been resampled to 4 000 vertices and improved with 100 Lloyd iterations. The final sampling of the skeleton, which only contains open backbones, is a direct result of the resampling procedure and accurately matches with the triangles around the feature creases. Remeshing took 500ms for resampling and 3 seconds for Lloyd relaxation.

A model with typical CAD tessellation is presented in Figure 7. The degenerated triangles are a major issue for parametrization-based methods such as [SG03], which is

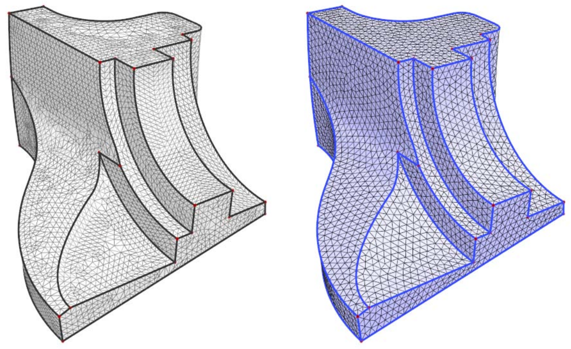

Figure 6: *The Fandisk model with highlighted feature skeleton (left), remeshed with 4k vertices (right).*

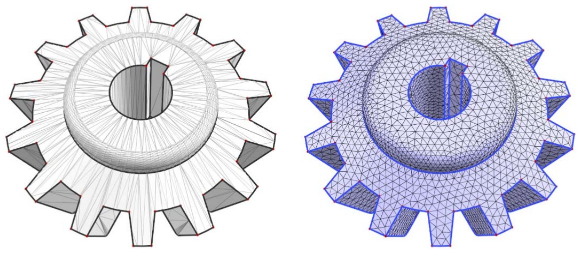

Figure 7: *A typical CAD model with many degenerated triangles and feature skeleton (left) and the remeshed model with 5k vertices (right).*

mainly caused by distortion in the patches and may lead to erroneous vertex relocations. We applied a simple mesh slicing procedure [BK01] with a regular grid to aid patch construction with more well-shaped triangles.

Table 1 presents a statistical analysis and comparison of the remeshing results. The Avg∠ and Min∠ are the average of the minimum angle in each triangle and the smallest angle in the triangulation, respectively. The error is the Hausdorff distance w.r.t. the bounding box diagonal as calculated by Metro [CRS98]. All comparisons are taken from the results of related work and have not been re-evaluated for normalization. Note that we achieve a low error while keeping an exact vertex budget.

For all timings we used an AMD Opteron multicore system with 2.7GHz per CPU. We limited the memory consumption for the patch cache to 1GB but this limit was never reached, not even for the Beethoven model (Figure 8), which used 630MB RAM for 84k patches.

7. Conclusion and Future Work

We presented an efficient remeshing framework for relaxation-based methods. In particular, we used Lloyd's algorithm to build a WCVT on the mesh to obtain an isotropic sample distribution. The key to performance is the initial

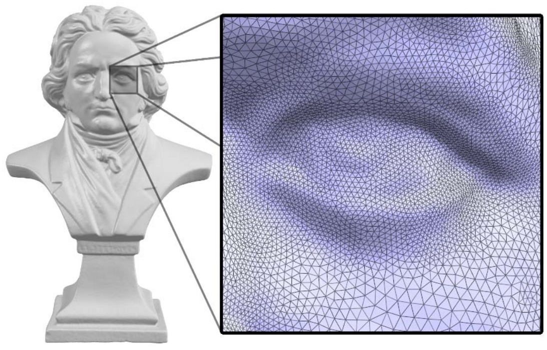

Figure 8: *The Beethoven model remeshed from 1.5M vertices to 500k vertices. Resampling and Lloyd relaxation took less than 5 minutes with 8 threads.*

Model	Vertices	Time	∠ (deg)		Error
		(sec)	Avg	Min	(10^{-3})
Hygieia (original)	8 268	–	34.7	0.25	–
Hygieia (our)	10 000	9 / 3.5	52.9	30.1	3.5
Hygieia [SG03]	8 750	17	52.4	25.9	2.7
Hygieia [FZ09]	6 529	113	51.9	35.4	n/a
Horse (original)	48 485	–	37.1	1.27	–
Horse (our)	6 000	16 / 7	51.9	29.8	4.9
Horse [SG03]	5 695	28	50.1	9.1	6.1
Horse [FZ09]	3 017	103	51.9	35.7	n/a
Fandisk (original)	6 475	–	43.5	17.0	–
Fandisk (our)	4 000	3 / 1.4	53.3	20.6	1.7
Fandisk [SG03]	5 135	17	49.1	16.8	0.4
Beethoven (orig.)	1.5M	–	34.2	0.01	–
Beethoven (our)	500k	676 / 280	52.7	28.0	1.4

Table 1: *Analysis and comparison of the remeshing results. We always applied 100 Lloyd iterations. The timings are for the full pipeline with 1 and 8 threads, respectively. Note that total times are not normalized across publications. [SG03] used P4 with 2.4GHz; [FZ09] used P4 with 2.8GHz, we used AMD Opteron with 2.7GHz.*

sampling procedure which is itself efficient and simple, and provides the means for fast convergence of Lloyd's relaxation method. The mesh topology is preserved because both, the edge flip algorithm and hole re-triangulation do not introduce edges that are already present in the mesh. In theory, nothing prevents the algorithm from creating surface self-intersections, but we never observed this in practice. Our results compare favorably to state-of-the-art techniques in both processing time and mesh quality and fulfill the desired criteria listed in Section 1.

In the future, we would like to investigate different data structures for patch caching that may be more suitable for parallelization and will probably reduce the drop in performance for a larger amount of threads. Separate locks for different regions of the mesh are also possible using vertex clustering algorithms. We would also like to improve the parametrization strategy to handle complicated regions

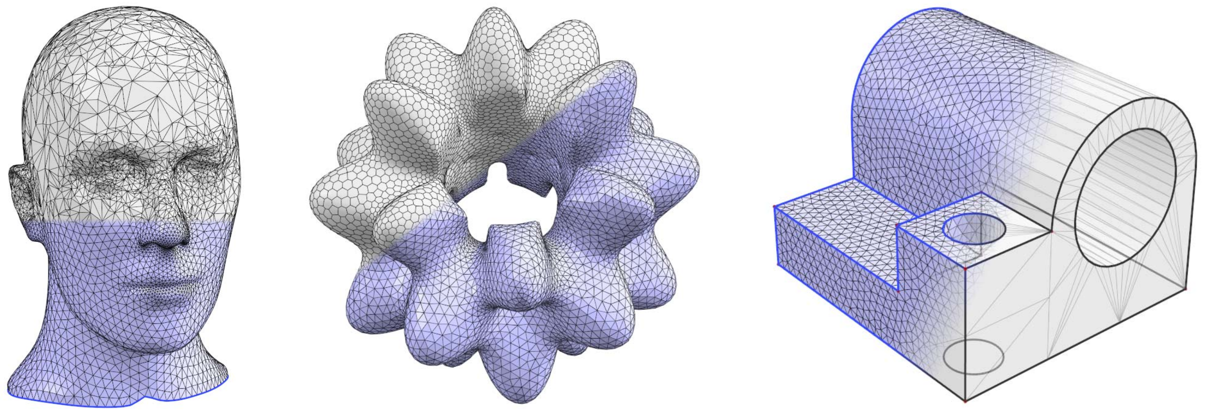

Figure 9: *More remeshing results: The Mannequin model (left) with resampled subdivision surface* (top) *and the remeshed model* (bottom)*, the remeshed Bumpy Torus model (middle) with Voronoi regions* (top) *and triangulation* (bottom)*, and the Joint model (right). All meshes are remeshed with 5k vertices at almost interactive speed.*

of the mesh in a consistent way. For example non-manifold connectivity is problematic and the system fails to create patches in these regions. Meshes with heavily degenerated triangles prevent the parametrization strategy from creating roundish patches, which causes large distortions and harms the accuracy of vertex relocations.

References

[AdVDI03] ALLIEZ P., DE VERDIÈRE É. C., DEVILLERS O., ISENBURG M.: Isotropic surface remeshing. In *Proc. SMI* (2003), pp. 49–58. 2, 3, 4

[BK01] BOTSCH M., KOBBELT L.: A robust procedure to eliminate degenerate faces from triangle meshes. In *Proc. of VMV* (2001). 7

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *CGF 17*, 2 (1998), 167–174. 7

[DFG99] DU Q., FABER V., GUNZBURGER M.: Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review 41*, 4 (1999), 637–676. 2

[DHKL01] DYN N., HORMANN K., KIM S.-J., LEVIN D.: Optimizing 3D triangulations using discrete curvature analysis. *Mathematical Methods for Curves and Surfaces* (2001), 135–146. 3

[Flo03] FLOATER M. S.: Mean value coordinates. *CAGD 20* (2003). 6

[FZ09] FU Y., ZHOU B.: Direct sampling on surfaces for high quality remeshing. *CAGD 26*, 6 (2009), 711–723. 2, 3, 7

[GKS92] GUIBAS L. J., KNUTH D. E., SHARIR M.: Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica 7*, 1-6 (June 1992), 381–413. 5

[HLD04] HÖRMANN W., LEYDOLD J., DERFLINGER G.: *Automatic Nonuniform Random Variate Generation*. Statistics and Computing. 2004. 4

[JH02] JIAO X., HEATH M. T.: Feature detection for surface meshes. In *Proc. 8th Int. Conf. on Numerical Grid Generation in Computational Field Simulations* (2002), pp. 705–714. 3

[Llo82] LLOYD S. P.: Least squares quantization in PCM. *IEEE Trans. on Information Theory 28*, 2 (1982), 129–137. 2, 5

[PC06] PEYRÉ G., COHEN L. D.: Geodesic remeshing using front propagation. *IJCV 69*, 1 (2006), 145–156. 5

[SAG03] SURAZHSKY V., ALLIEZ P., GOTSMAN C.: Isotropic remeshing of surfaces: a local parameterization approach. In *Proc. 12th Int. Meshing Roundtable* (2003), pp. 215–224. 2, 3, 6

[SG03] SURAZHSKY V., GOTSMAN C.: Explicit surface remeshing. In *SGP* (2003), pp. 20–30. 2, 6, 7

[SNB07] SANDER P. V., NEHAB D., BARCZAK J.: Fast triangle reordering for vertex locality and reduced overdraw. *ACM TOG 26*, 3 (2007), 89. 3, 4, 6

[SZL92] SCHROEDER W. J., ZARGE J. A., LORENSEN W. E.: Decimation of triangle meshes. In *Proc. SIGGRAPH* (1992), pp. 65–70. 5

[Tur92] TURK G.: Re-tiling polygonal surfaces. In *Proc. SIGGRAPH* (1992), no. 2, pp. 55–64. 3, 5

[TW98] THÜRMER G., WÜTHRICH C. A.: Computing vertex normals from polygonal facets. *JGT 3*, 1 (1998), 43–46. 3

[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 *Proc. of SGP* (2009), pp. 1445–1454. 2