Laplacian Mesh Optimization

Andrew Nealen
TU Berlin

Takeo Igarashi
The University of Tokyo / PRESTO JST

Olga Sorkine*
TU Berlin

Marc Alexa
TU Berlin

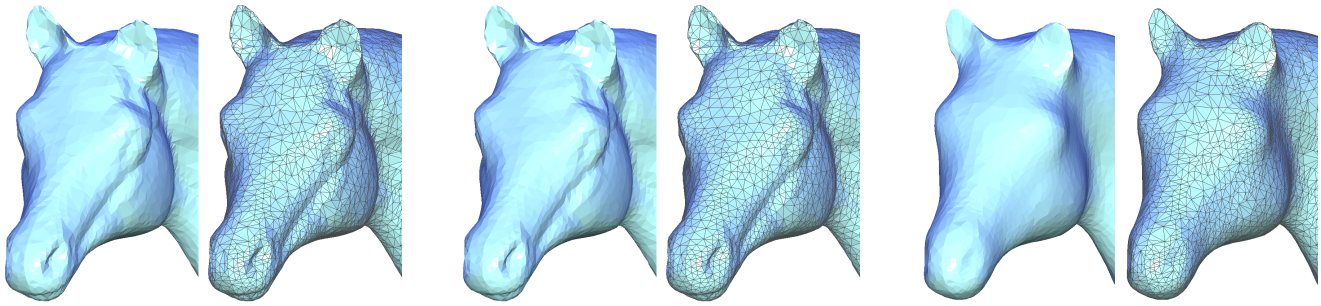


Figure 1: Original HORSE mesh (left) triangle shape optimization (middle) and feature preserving smoothing (right)

Abstract

We introduce a framework for triangle shape optimization and feature preserving smoothing of triangular meshes that is guided by the vertex Laplacians, specifically, the uniformly weighted Laplacian and the discrete mean curvature normal. Vertices are relocated so that they approximate prescribed Laplacians and positions in a weighted least-squares sense; the resulting linear system leads to an efficient, non-iterative solution. We provide different weighting schemes and demonstrate the effectiveness of the framework on a number of detailed and highly irregular meshes; our technique successfully improves the quality of the triangulation while remaining faithful to the original surface geometry, and it is also capable of smoothing the surface while preserving geometric features.

Keywords: Discrete Differential Geometry, Triangle Quality, Remeshing, Smoothing, Fairing, Least Squares

1 Introduction

Polygonal, and specifically triangular meshes are ubiquitous in computer graphics. Whether they have been generated by hand, from real world data, or by other means, obtaining a well-behaved mesh with respect to sampling rate and triangle quality is of great importance for a variety of applications. Our goal is to establish a simple and efficient framework which optimizes triangle shapes, or smoothes an input mesh, solely by means of vertex relocation, while maintaining both sampling rate and connectivity. The underlying assumption is that the input mesh is the *ground truth*, i.e. the sampling rate and connectivity information hold all information on the underlying smooth shape (possibly with implicitly defined or user-tagged sharp features).

Our framework is inspired by recent methods that compute vertex positions as the weighted least-squares solution to positional and Laplacian constraints. In particular, several methods try to preserve all original vertex Laplacians while imposing new positions for a few vertices [Lipman et al. 2004; Sorkine et al. 2004; Yu et al. 2004], which allows detail preserving modeling. Prescribing new vertex Laplacians has several additional applications: a smooth surface has a Laplacian with small magnitude. This observation has lead Sorkine and co-workers to represent the geometry of a mesh by assuming the vertex Laplacians vanish and additionally fixing the positions of a few control vertices [Sorkine and Cohen-Or 2004; Sorkine et al. 2005]. By prescribing vertex Laplacians with modified magnitude or direction, Nealen et al. [2005] achieve a variety of modeling effects, which are used in a sketch-based tool. We explain these techniques in Section 3.

We take these ideas a step further and use positional and Laplacian constraints in *all* vertices. Figure 2 illustrates our basic idea. Constraining all vertices with both positions and vertex Laplacians leads to a simple, flexible, and powerful framework for mesh optimization (Section 4). Inspired by the local triangle shape optimization that Nealen et al. [2005] used to embed the user drawn sketches into the mesh, we show that in our framework *all* triangles can be optimized at the same time (Section 5). Figure 3 shows that fixing only a few vertices (as in [Nealen et al. 2005]) would not suf-

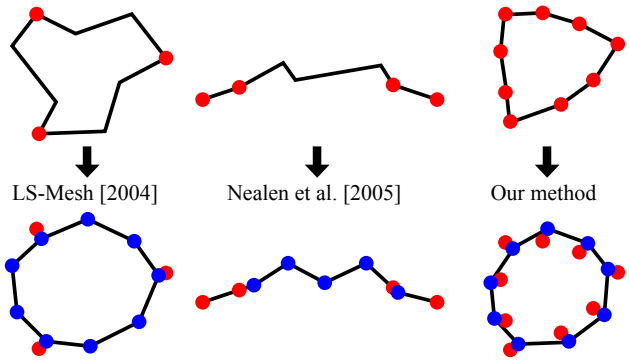


Figure 2: Left: Sorkine and Cohen-Or [2004] reconstruct the surface from a subset of control vertices (red). Middle: Nealen et al. [2005] constrain the boundary vertices (red), and optimize internal triangle shapes. Right: we constrain all vertices and optimize triangle shapes and/or smooth the mesh, with optional feature preservation.

*Supported by the Alexander von Humboldt Foundation.

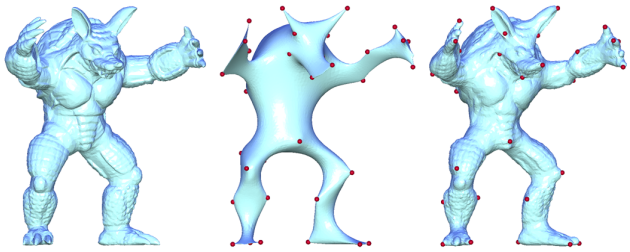


Figure 3: Least squares mesh [Sorkine and Cohen-Or 2004] (middle) and detail preserving triangle shape optimization [Nealen et al. 2005] (right) of the ARMADILLO (left), each with the same set of 40 control vertices (red).

fice. Similarly, the idea of Sorkine and Cohen-Or [2004] to generate fairly smooth meshes by constraining a few positions and assuming the vertex Laplacians vanish turns into a global mesh smoothing technique. This idea is detailed in Section 6. Figure 1 demonstrates both techniques. Furthermore, by adjusting the weights on the different terms of the optimization it is possible to preserve sharp features, if this is desired. Sections 5 and 6 introduce several weighting schemes for the positional as well as the Laplacian constraints and discuss the variety of effects that can be achieved.

2 Related work

Since our framework performs vertex relocation for both triangle shape optimization and smoothing, we briefly list recent work in these fields that is most related to our approach. An extensive overview is beyond the scope of this paper, but can be found on remeshing [Alliez et al. 2005] and geometric signal processing [Taubin 2000]. More recent surveys on mesh smoothing are available in [Hildebrandt and Polthier 2005], [Bobenko and Schröder 2005] and [Chen and Cheng 2005].

Vertex relocation for remeshing. Repositioning vertices is generally treated as a subproblem in the context of *remeshing* [Alliez et al. 2005], motivated by the need to improve triangle shapes with respect to one or more of the extensively studied quality metrics [Pébay and Baker 2003]. There are quite a few algorithms which circumvent the problem of relocating original mesh vertices entirely, by resampling the surface [Turk 1992] using a global parameterization of the mesh [Alliez et al. 2002; Alliez et al. 2003]. Remeshing algorithms, which perform connectivity optimization with vertex relocation, construct a global or a per-vertex, local parameterization, and use this parameterization to lift the vertex to the original surface after relocation in the parameter domain [Vorsatz et al. 2001; Surazhsky and Gotsman 2003; Vorsatz et al. 2003; Surazhsky et al. 2003]. When exact error bounds are not required, the relocated vertex can also be simply projected back onto its original tangent plane [Botsch and Kobbelt 2004], instead of constructing a parameterization. What is common to most of these algorithms (except for the "pure" halftoning approach [Alliez et al. 2002]), is that they are inherently iterative, i.e. they apply the relocation step-by-step, one vertex at a time, until some stoppage criterion is reached. In contrast, all vertex relocations in our system result from the unique solution of one sparse linear system.

Mesh smoothing. The bulk of existing work in mesh smoothing deals with discrete filter design. Taubin's [1995] pioneering work introduces a two-step Laplacian operator to inflate the mesh after smoothing, thereby reducing shrinkage. Desbrun et al. [1999] use implicit integration with Fujiwara [1995] or cotangent weights [Pinkall and Polthier 1993] for scale-dependent, unconditionally stable smoothing, which leaves triangle shapes and sizes mostly unchanged (also known as *mean curvature flow* or MC flow for short). This approach is extended to handle anisotropy and to preserve features by Meyer et al. [2003], and further improved to *prescribed* MC flow by Hildebrandt and Polthier [2005]. Recently, researchers have applied the bilateral filter, known from image processing, to discrete surfaces [Fleishman et al. 2003; Jones et al. 2003]. Our approach to mesh smoothing differs significantly from all of these methods. We rely on the least-squares meshes algorithm [Sorkine and Cohen-Or 2004] to perform inner (triangle shape) and/or outer (surface smoothness) fairing, while applying soft positional constraints to all mesh vertices – where the weights depend on e.g. discrete curvature distribution – to retain specific features. Our optimization technique has similarities with the geometric fairing of Schneider and Kobbelt [2001]; their method is oriented towards freeform surface design and thus prescribes positional constraints only to the boundary vertices, whereas our technique optimizes an existing mesh geometry and/or triangle quality.

3 Basics and notation

The mesh is represented as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, with vertices $\mathbf{V}$ and edges $\mathbf{E}$, where $\mathbf{V} = [\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_n^T]^T$, $\mathbf{v}_i = [v_{ix}, v_{iy}, v_{iz}]^T \in \mathbb{R}^3$ is the original geometry, and $\tilde{\mathbf{V}}'$ denotes the displaced geometry. Furthermore, δ_i is the Laplacian of $\mathbf{v}_i$, the result of applying the discrete Laplace operator to $\mathbf{v}_i$, i.e.

$$\delta_i = \sum_{\{i,j\}\in\mathbf{E}} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = \left[\sum_{\{i,j\}\in\mathbf{E}} w_{ij}\mathbf{v}_j\right] - \mathbf{v}_i, \qquad (1)$$

where $\sum_{\{i,j\}\in E} w_{ij} = 1$, and the choice of weights

$$w_{ij} = \frac{\omega_{ij}}{\sum_{\{i,k\}\in\mathbf{E}} \omega_{ik}} \qquad (2)$$

defines the nature of δ_i. Some popular choices are

$$\omega_{ij} = 1, \qquad (3)$$
$$\omega_{ij} = cot\ \alpha + cot\ \beta, \qquad (4)$$

where (3) are the uniform and (4) the cotangent weights. The angles used in these equations are shown in Fig. 4. In the remainder of this paper, we will refer to the uniform and cotangent Laplacians with normalized weights as δ_u and δ_c respectively. To obtain the Laplacians for the entire mesh, we use the $n \times n$ Laplacian matrix $\mathbf{L}$, with elements

$$\mathbf{L}_{ij} = \begin{cases} -1 & i = j \\ w_{ij} & (i,j) \in \mathbf{E} \\ 0 & \text{otherwise} \end{cases}, \qquad (5)$$

and we denote as $\mathbf{L}_u$ and $\mathbf{L}_c$ the Laplacian matrices with uniform and cotangent weights respectively. With $\mathbf{V}_d = [v_{1d}, v_{2d}, \ldots, v_{nd}]^T, d \in \{x,y,z\}$, the $n \times 1$ vector containing the x,y or z coordinates of the n vertices, we can compute the x, y and z coordinates of the Laplacians $\Delta_d = [\delta_{1d}, \delta_{2d}, \ldots, \delta_{nd}]^T, d \in \{x,y,z\}$ separately as

$$\Delta_d = \mathbf{L}\mathbf{V}_d. \qquad (6)$$

The uniform Laplacian of $\mathbf{v}_i$ points to the centroid of its neighboring vertices, and has the nice property that its weights do not depend on the vertex positions. The cotangent Laplacian is known to be a good approximation of the surface normal, although the weights can become negative and are nonlinear in the vertex positions. When properly scaled by the Voronoi region (see Fig. 4)

$$\overline{\kappa}_i \mathbf{n}_i = \delta_{i,c\overline{\kappa}} = \frac{1}{4A(\mathbf{v}_i)} \sum_{\{i,j\}\in\mathbf{E}} (cot\ \alpha + cot\ \beta)(\mathbf{v}_j - \mathbf{v}_i), \qquad (7)$$

as described by Meyer et al. [2003], we obtain the discrete mean curvature normal $\overline{\kappa}_i \mathbf{n}_i$, which is the unit length surface normal $\mathbf{n}_i$

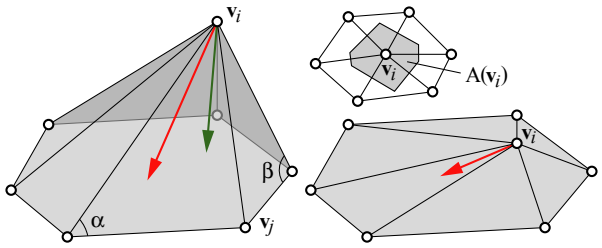


Figure 4: Left: uniform (red) and cotangent (green) Laplacian vectors for a vertex $\mathbf{v}_i$ and its (in this case planar) 1-ring, as well as the angles used in Eqn. 4 for one $\mathbf{v}_j$, Bottom right: the effect of flattening $\mathbf{v}_i$ into the 1-ring plane. While the cotangent Laplacian vanishes, the uniform Laplacian generally does not. Right top: the Voronoi region $A(\mathbf{v}_i)$ around a vertex.

scaled by the discrete mean curvature $\overline{\kappa}_i$. Furthermore, we will use Eqn. 7 to discretize the Laplacian matrix $\mathbf{L}$, denoted as $\mathbf{L}_{c\overline{\kappa}}$, in the context of mesh smoothing (Section 6).

We now describe two interesting applications, which will lead to our optimization in Section 4.

Least squares meshes. Sorkine and Cohen-Or [2004] demonstrate how a mesh can be reconstructed from connectivity information alone, using a small subset $\mathbf{C} \subset \mathbf{V}$ of m geometrically constrained vertices (anchors). They solve for x, y and z positions ($\mathbf{V}'_d = [v'_{1d}, v'_{2d}, \ldots, v'_{nd}]^T, d \in \{x,y,z\}$) separately by minimizing the quadratic energy

$$\|\mathbf{L}_u \mathbf{V}'_d\|^2 + \sum_{s \in \mathbf{C}} w_s^2 |v'_{sd} - v_{sd}|^2, \tag{8}$$

where the v_{sd} are the stored anchor positions and the w_s^2 are weighting factors. In practice, with the anchors as the first m vertices (w.l.o.g.), the $(n+m) \times n$ overdetermined linear system $\mathbf{A}\mathbf{V}'_d = \mathbf{b}$

$$\left[\begin{array}{c|c} \mathbf{L}_u \\ \hline \mathbf{I}_{m \times m} & \mathbf{0} \end{array}\right] \mathbf{V}'_d = \left[\begin{array}{c} \mathbf{0} \\ \hline \mathbf{V}_{(1\ldots m)d} \end{array}\right] \tag{9}$$

is solved in the least squares sense using the method of normal equations $\mathbf{V}'_d = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$. Note that the first n rows of $\mathbf{A}\mathbf{V}'_d = \mathbf{b}$ are the Laplacian constraints, while the last m rows are the positional constraints. The reconstructed shape is generally smooth, with the possible exception of small areas around anchor vertices. The minimization procedure moves each vertex to the centroid of its 1-ring, since the uniform Laplacian $\mathbf{L}_u$ is used, resulting in good inner fairness.

Detail preserving triangle shape optimization. Nealen et al. [2005] show how a least squares optimization can improve triangle quality in a small mesh region (including boundary constraints) with negligible vertex drift. Essentially, they modify the linear constraints in Eqn. 9 as

$$\left[\begin{array}{c|c} \mathbf{L}_u \\ \hline \mathbf{I}_{m \times m} & \mathbf{0} \end{array}\right] \mathbf{V}'_d = \left[\begin{array}{c} \Delta_{d,c} \\ \hline \mathbf{V}_{(1\ldots m)d} \end{array}\right], \tag{10}$$

where the uniform Laplacian of each new vertex position is asked to resemble its *undeformed* cotangent Laplacian as closely as possible. The idea here is that while the uniform Laplacian has a tangential component, the cotangent Laplacian does not (see also [Desbrun et al. 1999]), and thus the optimization attempts to remove the tangential components, while preserving the the surface details in the normal direction. As shown in Fig. 3, this method fails to preserve the overall shape when the region is large, or there are insufficient boundary constraints, but works well for local modifications. We have experimented with the anchor selection process of Sorkine and Cohen-Or [2005] for Eqn. 10, an iterative process, which, although fast, still introduces a significant computational overhead. Yet, while this attempt did produce fairly good results after many iterations, it lead us to the *"all vertices are anchors"* approach, which requires no selection process, and will be described and discussed in the next sections.

4 Global vertex relocation framework

We propose a modification of Eqns. 9 and 10, which generalizes both [Sorkine and Cohen-Or 2004] and [Nealen et al. 2005] and gives rise to two interesting applications. Our general $2n \times n$ system $\mathbf{A}\mathbf{V}'_d = \mathbf{b}$ is

$$\left[\begin{array}{c} \mathbf{W}_L\mathbf{L} \\ \hline \mathbf{W}_p \end{array}\right] \mathbf{V}'_d = \left[\begin{array}{c} \mathbf{W}_L\mathbf{f} \\ \hline \mathbf{W}_p\mathbf{V}_d \end{array}\right]. \tag{11}$$

The main modification is that we no longer have a subset of positional constraints, instead, *all* vertices appear both as Laplacian, and positional constraints. As we will describe in more detail further below, we can modify the result of the operation and introduce a good trade-off between triangle quality and geometric error by (non-)uniformly weighting the positional constraints with the diagonal matrix $\mathbf{W}_p$. In some cases, it will also be beneficial to weight the Laplacian matrix $\mathbf{L}$ and the corresponding right-hand side $\mathbf{f}$ with the diagonal matrix $\mathbf{W}_L$. In general, larger weights in $\mathbf{W}_p$ enforce positional constraints and thus preserve the original geometry, which can be useful for high-curvature regions and sharp features. On the other hand, larger weights in $\mathbf{W}_L$ enforce regular triangle shapes and/or surface smoothness. As we describe in the following sections, solving the general system in Eqn. 11 results in

- **detail preserving triangle shape optimization**, when setting $\mathbf{L} = \mathbf{L}_u$ and $\mathbf{f} = \Delta_{d,c}$, or
- **mesh smoothing**, when setting $\mathbf{L} = \mathbf{L}_{c\overline{\kappa}}$ or $\mathbf{L}_c$ (outer fairness) or $\mathbf{L} = \mathbf{L}_u$ (inner and outer fairness) and $\mathbf{f} = \mathbf{0}$.

5 Global triangle shape optimization

Our goal here is to optimize triangle shapes. We measure our success with the *radius ratio* [Pébay and Baker 2003], mapped to $[0,1]$ as

$$t_i = 2\frac{r}{R}, \tag{12}$$

where R and r are the radii of the circumscribed and inscribed circles respectively. This way, $t_i = 1$ indicates a well shaped, $t_i = 0$ – a degenerate triangle. To perform this optimization, we discretize $\mathbf{L}$ in Eqn. 11 using the uniform Laplacian $\mathbf{L}_u$, set $\mathbf{f}$ on the right hand side to $\Delta_{d,c}$, and use $\mathbf{W}_L = \mathbf{I}$, similar to Eqn. 10. A straightforward, yet naive choice, is to use uniform weights $\mathbf{W}_p = \mathbf{W}_{const} = s\mathbf{I}$, where s denotes an arbitrary positive scalar. While this does somewhat improve the overall triangle quality (see Fig. 5(b)), and has the lowest geometric error (Hausdorff distance to (a) [Cignoni et al. 1998]), we can do better in terms of triangle quality.

It can be observed that large geometric error is likely to occur in the vicinity of vertices with high $\overline{\kappa}$. Therefore, a more sensitive choice of weights is a linear curvature-to-weight transfer function (Fig. 5(a), bottom), which maps discrete mean curvature $\overline{\kappa}_i$ to weight w_i for each vertex, resulting in $\mathbf{W}_p = \mathbf{W}_{linear}$. More precisely, given the minimal and maximal discrete mean curvature over the mesh, $\overline{\kappa}_{min}$ and $\overline{\kappa}_{max}$, we linearly map the interval $[\overline{\kappa}_{min}, \overline{\kappa}_{max}]$ to $[0,s]$ (Fig. 5(c)). Since the values for $\overline{\kappa}$ can be very large for a small set of vertices, we truncate these outliers before computing the linear mapping. The threshold value $\overline{\kappa}_\tau$ is computed using a simple box plot test [Tukey 1977], and works well for all models

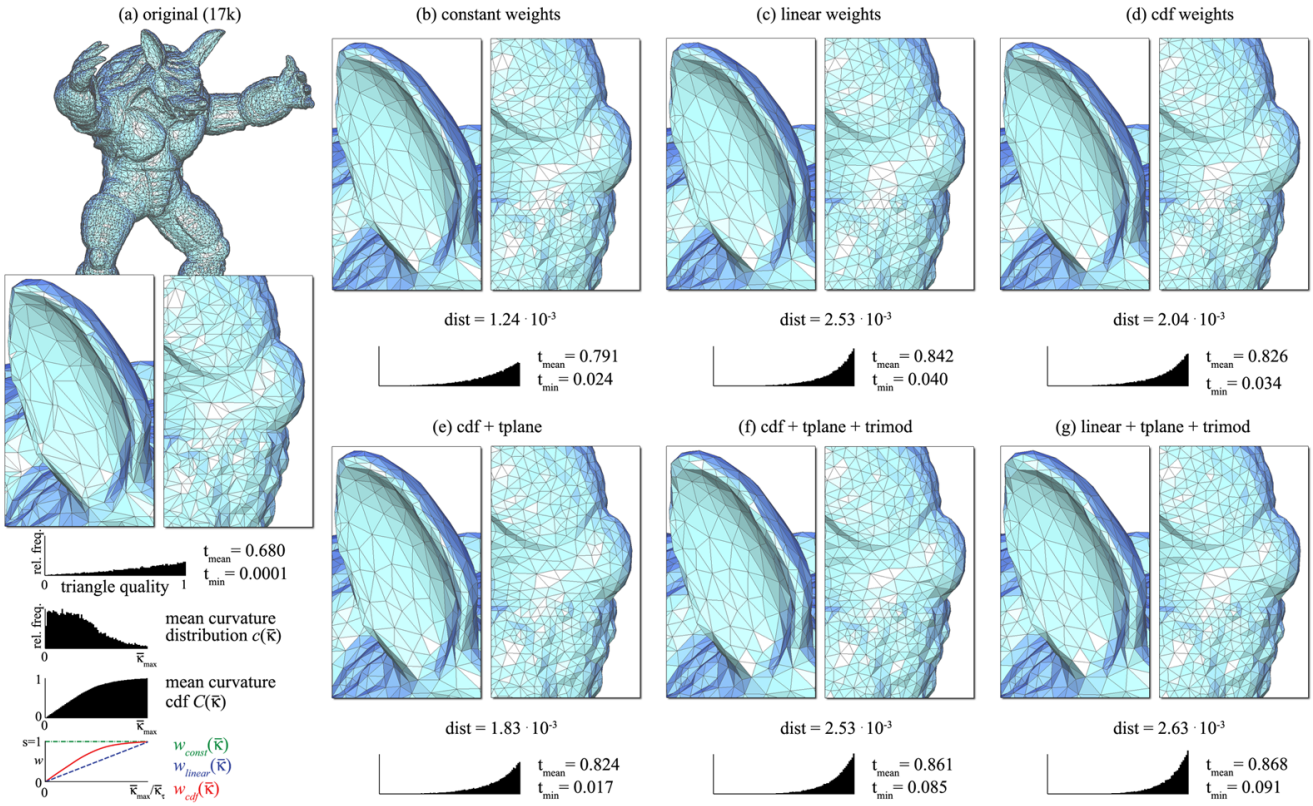


Figure 5: Comparison of triangle optimization weights. The far left column (a) shows (from top to bottom) a 17k vertices ARMADILLO mesh, with the same two close-ups used in (b)-(g), the triangle quality distribution (see Eqn. 12), the distribution function of discrete mean curvature $c(\overline{\kappa})$, the associated cumulative density function (cdf) $C(\overline{\kappa})$, and the three weighting functions, used in (b)-(g) to map from curvature to positional weight for each vertex. Each optimized mesh (b)-(g) shows its triangle quality distribution, its mean (t_{mean}) and minimal (t_{min}) triangle quality, as well as the Hausdorff distance ($dist$) to the original mesh (a) with respect to the bounding box diagonal.

we have encountered. The weights of vertices where $\overline{\kappa} > \overline{\kappa}_\tau$ are clamped to s.

As can be seen in Figs. 5 (b) and (c), while using $\mathbf{W}_{const}$ constrains low curvature vertices excessively, $\mathbf{W}_{linear}$ has a tendency to give them too much freedom, and therefore also has relatively high geometric error. Since we wish to take the relative frequency of mean curvature into account, we have chosen to use the cumulative density function (cdf) of $\overline{\kappa}$ (Fig. 5(a)) to map from $\overline{\kappa}_i$ to weight $w_i \in [0,s]$, resulting in $\mathbf{W}_p = \mathbf{W}_{cdf}$. In detail, given the normalized distribution function of mean curvature $c(\overline{\kappa})$ for a mesh (Fig. 5(a)), we compute the cdf $C(\overline{\kappa})$ as

$$C(\overline{\kappa}) = \int_0^{\overline{\kappa}} c(t)\, dt, \tag{13}$$

shown in Fig. 5(a) (in practice, $C(\overline{\kappa})$ is precomputed as a discrete sum over the vertex curvatures). The rationale behind using $C(\overline{\kappa})$ is that if we have a mesh with a large amount of low curvature vertices (such as the ARMADILLO in Fig. 5), these vertices should be assigned a larger weight than they would have if $\mathbf{W}_{linear}$ was used, thus reflecting the need to retain the detail in these regions and reduce vertex drift. As can be seen from the values in Fig. 5(d), using $\mathbf{W}_{cdf}$ is somewhere between $\mathbf{W}_{const}$ and $\mathbf{W}_{linear}$, and is a good trade-off between triangle quality and geometric error. The cumulative density function also has the advantage of implicitly dealing with outliers without the need to compute $\overline{\kappa}_\tau$.

5.1 Tangent plane constraints

To further reduce the geometric error, planar (2D) constraints can be incorporated in our system. Essentially, we add the following term to the energy in Eqn. 8:

$$\sum_{i=1}^{n} |\mathbf{n}_i \cdot (\mathbf{v}_i' - \mathbf{v}_i)|^2. \tag{14}$$

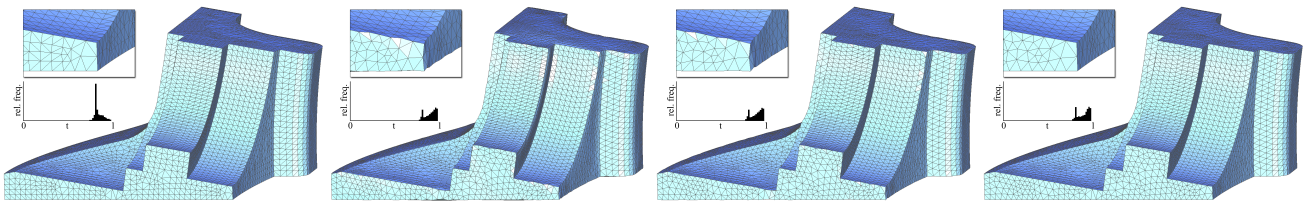
This term penalizes displacement perpendicular to the tangent plane defined by the original vertex position $\mathbf{v}_i$ and the local surface normal $\mathbf{n}_i$ given in Eqn. 7, assuming a first order approximation of the surface around $\mathbf{v}_i$. Since constraints of this form require solving for x, y and z positions simultaneously, we need to couple the $2n \times n$ system (Eqn. 11) for each coordinate x, y and z into a single $6n \times 3n$ system, and add n constraints of the form

$$\mathbf{n}_i \cdot \mathbf{v}_i' = \mathbf{n}_i \cdot \mathbf{v}_i, \tag{15}$$

resulting in a $7n \times 3n$ system. Although this involves significant computational overhead, Fig. 5(e) shows that tangent plane constraints reduce geometric error and have nearly the same mean triangle quality compared to cdf weights alone (Fig. 5(d)).

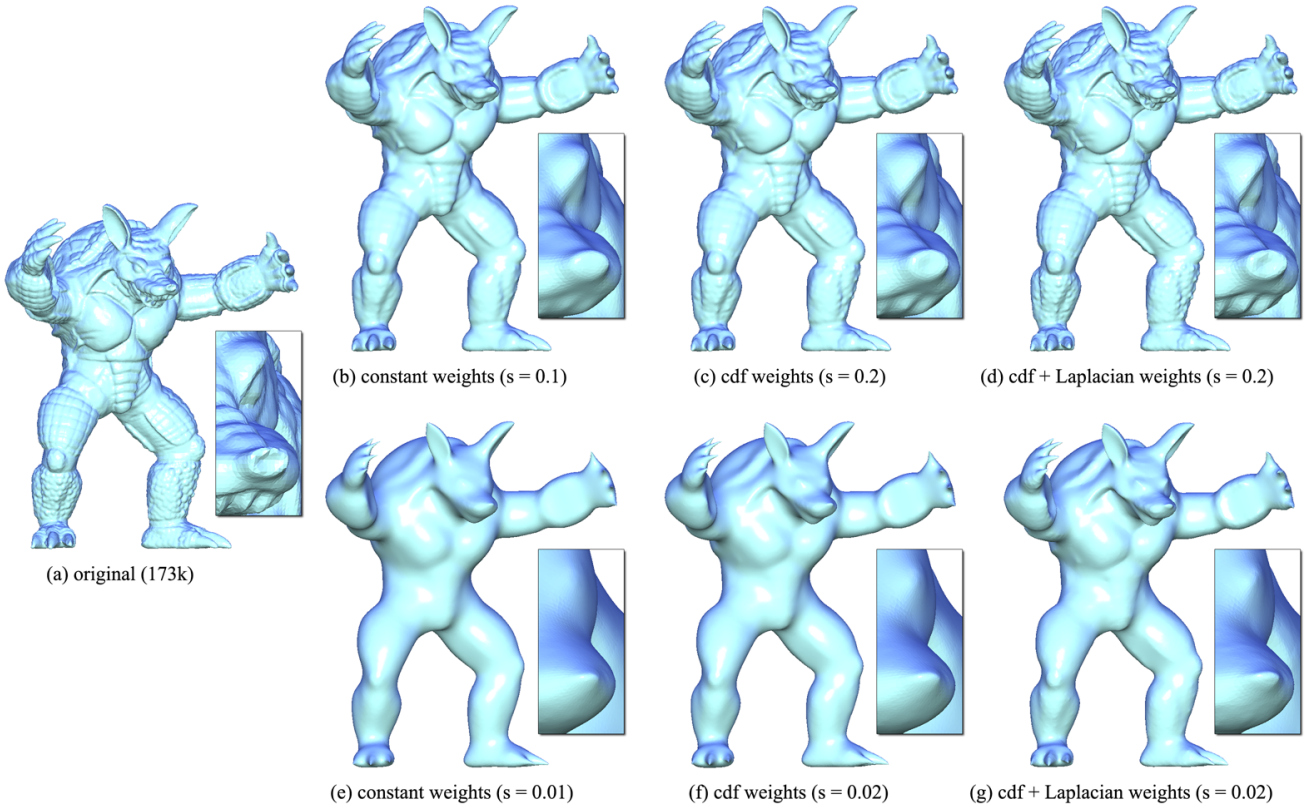
5.2 Triangle quality modulation

The quality of meshes for numerical simulations, such as the Finite Element Method (FEM), is heavily influenced by t_{min}, the minimal triangle quality. To maximize t_{min}, the positional weights $\mathbf{W}_p$ are



(a) original (b) linear weights (c) linear + tplane (d) linear + reduced Laplacian

Figure 6: Triangle optimization on an input mesh (a) with distinct sharp features. Using linear weights (b) breaks the sharp features, and using (2D) tangent plane constraints (c) cannot resolve the problem, since these features would require 1D or zero dimensional (positional) constraints. Instead, we reduce the weight on Laplacian constraints of feature vertices (d).



(a) original (173k) (b) constant weights (s = 0.1) (c) cdf weights (s = 0.2) (d) cdf + Laplacian weights (s = 0.2)

(e) constant weights (s = 0.01) (f) cdf weights (s = 0.02) (g) cdf + Laplacian weights (s = 0.02)

Figure 7: Comparison of smoothing weights. Columns 2 and 3 are generated using the weight functions introduced in Section 5, with different scaling factors s. The right column shows the effect of reducing the weights on the (Laplacian) smoothness constraints of high curvature vertices. All results in this figure were generated with $\mathbf{L} = \mathbf{L}_u$.

modulated with the diagonal matrix $\mathbf{W}_t$, where the entry $w_{t,i}$ for vertex $\mathbf{v}_i$ is set to the minimal triangle quality t of its adjacent triangles. This way, if there is a triangle with small t attached to $\mathbf{v}_i$ it will be allowed to move more than without the modulation. The effect for cdf and linear weighting can be observed in Figs. 5(f) and (g): triangle quality has improved, especially around the ear of the ARMADILLO. The cost for this improved triangle quality is increased geometric error, which can be a worthwhile trade-off if a large t_{min} is required by the application.

5.3 Sharp features

For meshes with distinct sharp features, i.e. edges (1D) and corners (0D), the method described so far produces visible artifacts (Fig. 6(b) and (c)). Clearly, what we want is that vertices on edges only move along the edge (1D constraint), while corners remain fixed in place (0D constraint). This requires solving an $3n \times 3n$ system, as in Section 5.1. We have found a very simple solution which achieves a similar effect, and only requires solving an $n \times n$ system; by reducing the weight on the *Laplacian* constraint of high curvature vertices (and using $\mathbf{W}_p = \mathbf{W}_{linear}$), feature vertices are no longer inclined to move, and their neighboring triangles will be optimized by their non-feature 1-ring vertices. Specifically, we use $\mathbf{W}_L = s\mathbf{I} - \mathbf{W}_{linear}$ in Eqn. 11 (when using weights $\mathbf{W}_{linear} \in [0,s]$) and scale all values to [0,1]. The result of this modification can be seen in Fig. 6(d), where all feature vertices remain in place, while vertices in planar regions are moved towards their respective 1-ring centroids.

6 Mesh Smoothing

Sorkine and Cohen-Or [2004], propose a procedure for smooth hole filling by fitting a thin-plate surface ($\mathbf{L}^2\mathbf{V}'_d = 0$, for infinitely

large anchor weights), while simultaneously improving the triangle quality of the fitted mesh. Since only boundary vertices are constrained, the fitted surface is smooth without detail or features. Our framework (Sec. 4) can be easily adjusted to perform global mesh smoothing, optionally with feature preservation, simply by setting $\mathbf{f} = 0$, and adjusting the positional- ($\mathbf{W}_p$) and Laplacian ($\mathbf{W}_L$) weights. Also, we can perform simultaneous inner and outer fairing with $\mathbf{L}_u$, or outer fairing alone with $\mathbf{L}_c$ or $\mathbf{L}_{c\overline{\kappa}}$. Note that using the non-normalized $\mathbf{L}_{c\overline{\kappa}}$ discretization is similar to the curvature flow approach [Desbrun et al. 1999]: the weights on Laplacian smoothness constraints are scaled by discrete mean curvatures.

To make the magnitude and behavior of our smoothing procedure user-tuneable in a simple and intuitive way, we utilize and adjust three parameters.

- **Positional weights.** The weighting matrices $\mathbf{W}_{const}$, $\mathbf{W}_{linear}$ and $\mathbf{W}_{cdf}$ introduced in Section 5 are applied in the setting of mesh smoothing. These either uniformly smooth the mesh, shown in Fig. 7(b) and (e), or attempt to retain features by placing more (positional) weight on high curvature vertices, as seen in Fig. 7(c) and (f).

- **Scale factor.** While the scale factor s was set to $s = 1$ for all of Section 5, we can now use this scaling to adjust the overall amount of smoothing. Compare the top and bottom rows of Fig. 7, where the scale factors differ by an order of magnitude.

- **Laplacian weights.** To further increase feature preservation, practically any function which reduces the weight $\mathbf{W}_L$ on Laplacian smoothness constraints of feature vertices can be applied. In this work we have experimented with Gaussian falloffs, and variants of the function used in Section 5.3 to automatically deduce $\mathbf{W}_L$ from the mean curvature cdf (see Figs. 8 and 7(d) and (g)), but many other choices are conceivable, e.g. user-tagged feature vertices.

The results of using these three parameters are shown in Fig. 7, where the full ARMADILLO model is smoothed. Note how varying the weights and the scale factor results in certain feature preservation, which can be seen in the closeup of the snout and teeth. For a detailed analysis of using $\mathbf{W}_{const}$ with varying s, see [Sorkine and Nealen 2006]. A typical example of feature preserving smoothing

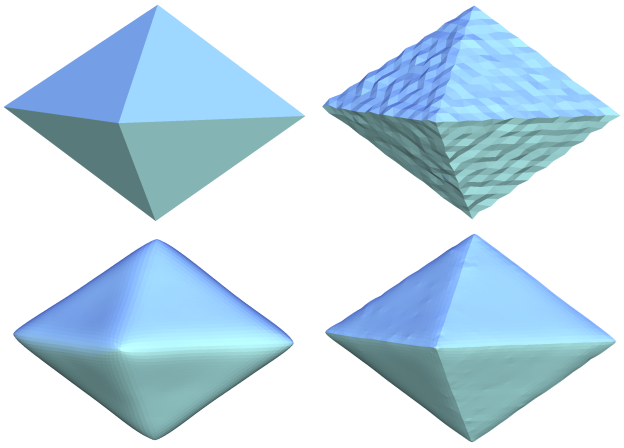
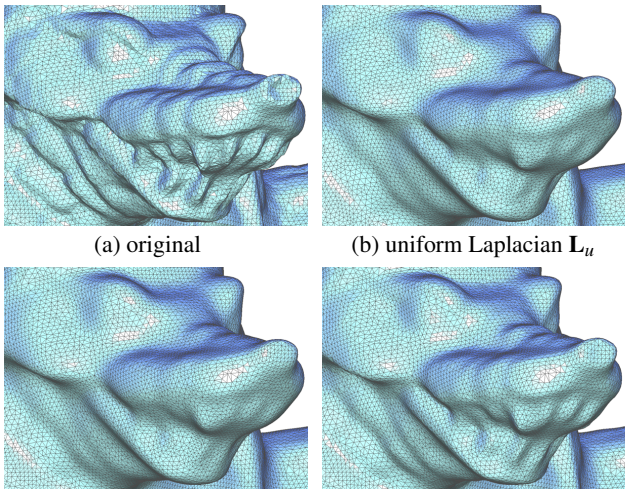


Figure 8: Smoothing a pyramid. Top row: original DOUBLEPYRAMID and the noisy version. Bottom row: Using $\mathbf{W}_p = \mathbf{W}_{linear}$ alone smoothes out sharp features, while additionally reducing the weights on Laplacian constraints of feature vertices recovers most of the original shape.



(a) original (b) uniform Laplacian $\mathbf{L}_u$

(c) cotangent Laplacian $\mathbf{L}_c$ (d) weighted Laplacian $\mathbf{W}_L\mathbf{L}_c$

Figure 9: Comparison of umbrella (b) and cotangent (c) discretization of the $\mathbf{L}$ matrix. In (d), Laplacian constraints $\mathbf{L}_c\mathbf{V}'_d = 0$ are relaxed on feature vertices.

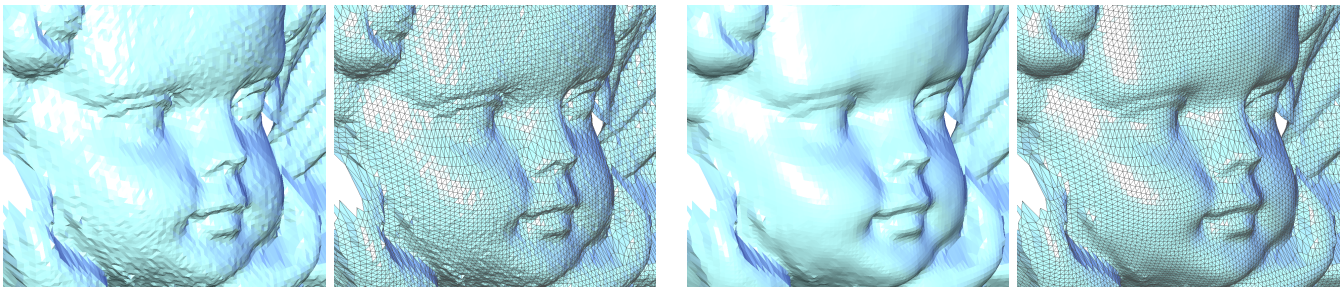
is shown in Fig. 8. The original DOUBLEPYRAMID model in the upper left is contaminated with Gaussian noise, shown in the upper right. In the lower left the result of using $\mathbf{W}_{linear}$ is shown, where the edges are visible, albeit heavily smoothed. If we additionally use Laplacian weights $\mathbf{W}_L$, thereby relaxing Laplacian constraints as described above, we can successfully recover most of the original model.

The results shown in Figs. 7 and 8 use $\mathbf{L} = \mathbf{L}_u$, resulting in a coupling of inner and outer fairness. While this works well for regularly sampled meshes, it has been pointed out by Desbrun et al. [1999] that this no longer holds for irregularly sampled meshes, and unwanted object deformation can occur. This effect is somewhat reduced in our framework, thanks to positional constraints, yet we can also decouple inner and outer fairness simply by using L_c or $L_{c\overline{\kappa}}$. As in other work on curvature flow, this moves each vertex along its normal, while leaving the tangential component unchanged. The effect is clearly visible, when comparing Fig. 9 (b) and (c); while the mesh in (b) has nicely shaped triangles, the triangle shapes in (c) mostly reflect the original triangulation in a smoothed version. Note that $\mathbf{L}_c$ can also be coupled with the feature preserving Laplacian weighting matrix $\mathbf{W}_L$ (Fig. 9(d)).

7 Discussion and future work

Our framework can optimize triangle shapes and smooth meshes with results similar to existing methods, but has the advantage that the solution is well-defined and can be computed using optimized sparse linear solvers [Toledo 2003]. Timing our algorithm amounts to measuring the time for the factorization of the system matrix, since this part makes up approximately 70-80% of the entire runtime. On an Intel Centrino Duo with 2.0 GHz we can factorize the $n \times n$ system matrix ($3n \times 3n$ for tangent plane constraints) for meshes with 173k, 43k and 17k vertices in 27s(43s), 3s(25s) and 1s(6s) respectively.

In addition, setting specific least-squares weights allows optimizing the results with respect to the particular model and the application at hand. The parameters we expose allow either trading off geometric error for triangle quality, or determine the intensity of the smoothing operation, including the amount of detail preservation. While the user may decide which weights and parameters to chose, we suggest the following combinations.



(a) original (b) smoothed

Figure 10: Smoothing the noisy ANGEL dataset, using $\mathbf{W}_{cdf}$ and $\mathbf{W}_L\mathbf{L}_c$.

- **Smooth models.** Models, for which the mean curvature distribution is fairly smooth, such as the ARMADILLO model used throughout this paper, generally benefit from using $\mathbf{W}_{cdf}$ for triangle shape optimization, since details in mid-range curvature regions are retained, the overall geometric error is low, and it produces a good distribution of triangle shapes. For smoothing, $\mathbf{W}_{cdf}$ nicely preserves detail in high curvature regions to a certain degree, which can be improved by additionally reducing weights on specific Laplacian smoothness constraints with $\mathbf{W}_L$.

- **CAD models.** CAD models, or models with mostly flat surfaces and some sharp features, are weighted with $\mathbf{W}_{linear}$ in Figs. 6 and 8. Using $\mathbf{W}_{cdf}$ has little effect in these cases, since the associated cdf has a high weight value for the numerous low (or zero) curvature vertices. This fixes these vertices in place, where they should actually be allowed to move freely in their tangent plane. The situation is similar for the smoothing application; here, we wish to de-noise the planar regions, thus necessitating a low weight on the positional constraints of these vertices, while holding feature curves and points in place, again making $\mathbf{W}_{linear}$ the better choice.

We ran the algorithm on some real world noisy data, and a few more meshes with bad triangle shapes. For the ANGEL dataset, acquired using 3D photography [Bouguet and Perona 1998], using $\mathbf{W}_{cdf}$ and $\mathbf{W}_L\mathbf{L}_c$ succeeds in removing the noise while retaining the prominent features around the nose and eyes (Fig. 10). The improvement in triangle quality on the TWEETY model in Fig. 11 is clearly visible, while a significant change in (flat) shading is not. Figures 11 and 12 illustrate the components of our algorithm: the original mesh (a) can be optimized with respect to triangle shapes (b) and adaptively smoothed with curvature dependent positional weights $\mathbf{W}_{cdf}$ (c) and reduced Laplacian weights $\mathbf{W}_L$ (d). Note that while in Figs. 11(c),(d) both inner and outer fairness is improved, Figs. 12(c),(d) leave inner fairness untouched, since $\mathbf{L}_{c\overline{\kappa}}$ and $\mathbf{L}_c$ decouple the fairness criteria.

We have not explicitly treated the issue of volume preservation, but the simple fact that the original mesh geometry is a substantial part of the result, and can be granularly controlled with $\mathbf{W}_p$, reduces shrinkage significantly.

We would like to stress the fact that all these optimizations based on adjusting the weight matrices $\mathbf{W}_p$ and $\mathbf{W}_L$ have no adverse effect on timing. It is conceivable that better feature detection could steer the weighting, leading in turn to better feature preservation, again at no additional cost (in the optimization computation). Consequently, we are interested in improved automated feature detection in our ongoing work. Other potential avenues for future work are volume preservation by analyzing (and adjusting) the spectral properties of the system matrix, and preservation of higher order discrete surface properties.

8 Acknowledgements

This work was carried out during a visit to the User Interface Research Group of The University of Tokyo (Autumn 2005), supported by a JSPS (Japan Society for the Promotion of Science) research grant.

References

ALLIEZ, P., MEYER, M., AND DESBRUN, M. 2002. Interactive geometry remeshing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 347–354.

ALLIEZ, P., ÉRIC COLIN DE VERDIÈRE, DEVILLERS, O., AND ISENBURG, M. 2003. Isotropic surface remeshing. In *SMI '03: Proceedings of the Shape Modeling International 2003*.

ALLIEZ, P., UCELLI, G., GOTSMAN, C., AND ATTENE, M., 2005. Recent advances in remeshing of surfaces. Part of the state-of-the-art report of the AIM@SHAPE EU network.

BOBENKO, A. I., AND SCHRÖDER, P. 2005. Discrete willmore flow. In *Eurographics Symposium on Geometry Processing*, 101–110.

BOTSCH, M., AND KOBBELT, L. 2004. A remeshing approach to multiresolution modeling. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 185–192.

BOUGUET, J.-Y., AND PERONA, P. 1998. 3D photography on your desk. In *ICCV'98 proceedings*, 43–50.

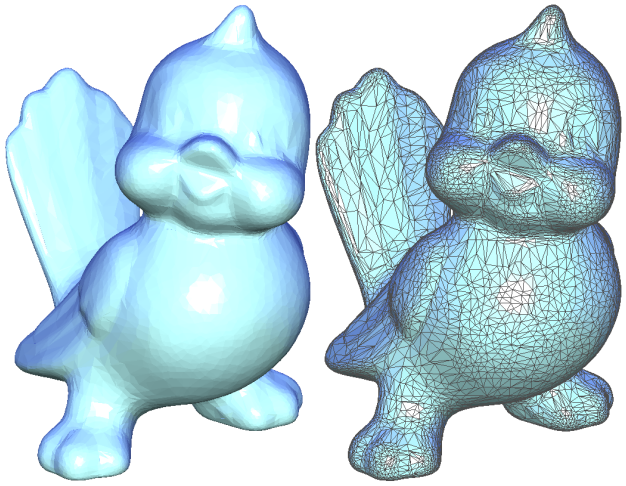
CHEN, C.-Y., AND CHENG, K.-Y. 2005. A sharpness dependent filter for mesh smoothing. *Comput. Aided Geom. Des. 22*, 5, 376–391.

CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. 1998. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2, 167–174.

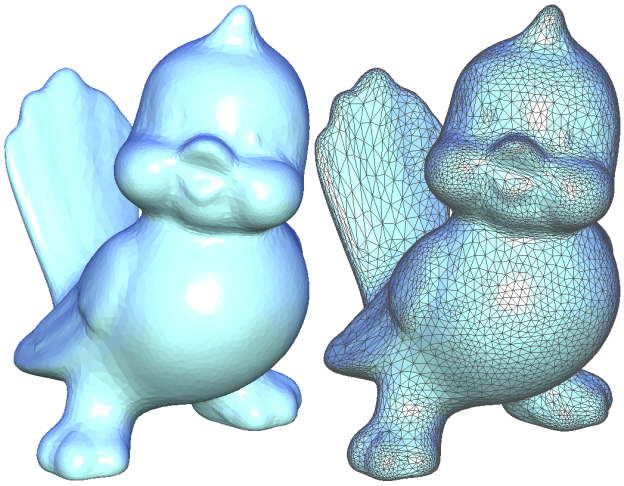
DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 317–324.

FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3, 950–953.

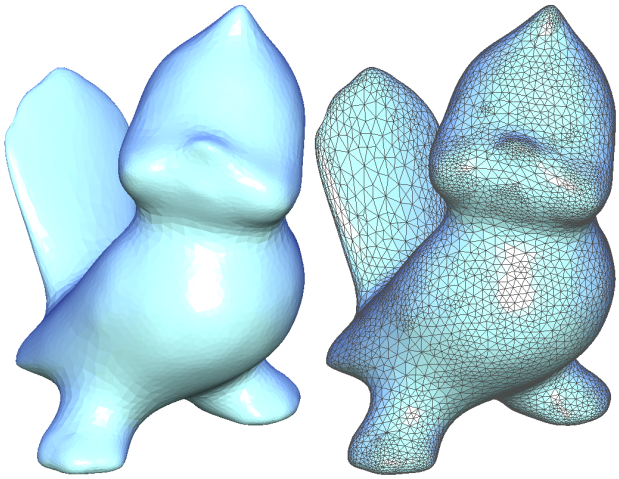
FUJIWARA, K. 1995. Eigenvalues of Laplacians on a closed Riemannian manifold and its nets. In *Proc. AMS.*, 2585 – 2594.



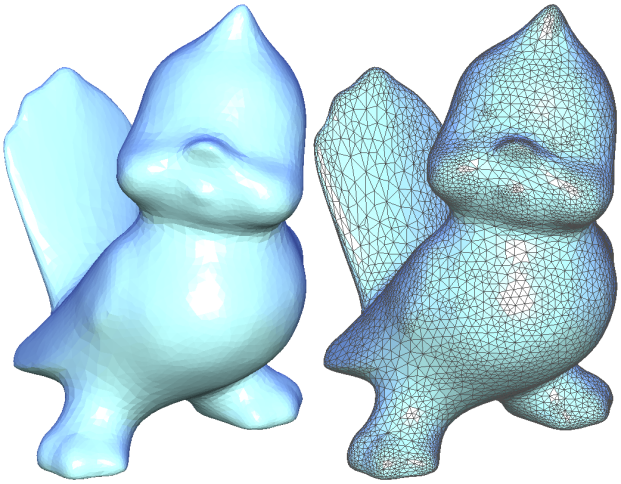
(a) original



(b) triangle optimization with $\mathbf{W}_{cdf}$



(c) smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{L}_u$
(inner and outer fairness)



(d) feature preserving smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{W}_L\mathbf{L}_u$
(inner and outer fairness)

Figure 11: TWEETY results.

HILDEBRANDT, K., AND POLTHIER, K. 2005. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum (Eurographics Proceedings) 23*, 3, 101–110.

JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. 22*, 3, 943–949.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., AND LEVIN, D. 2004. Differential coordinates for interactive mesh editing. In *International Conference on Shape Modeling and Applications*, 181–190.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and Mathematics III*, pages 35–57.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3, 1142–1147.

PÉBAY, P. P., AND BAKER, T. J. 2003. Analysis of triangle quality measures. *Math. Comput. 72*, 244, 1817–1839.

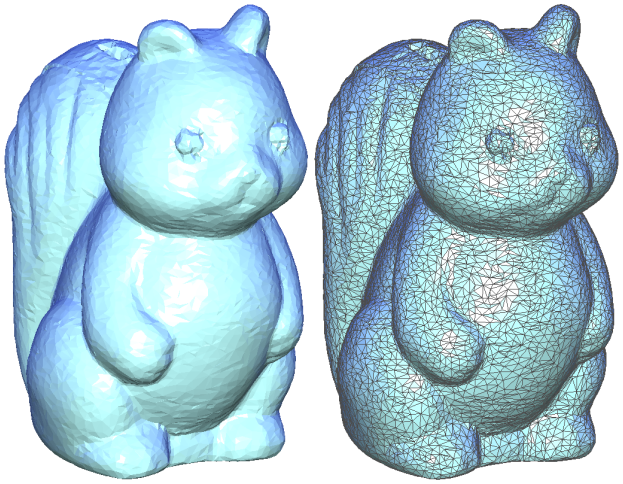
PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1, 15–36.

SCHNEIDER, R., AND KOBBELT, L. 2001. Geometric fairing of irregular meshes for freeform surface design. *Computer Aided Geometric Design 18*, 4, 359–379.

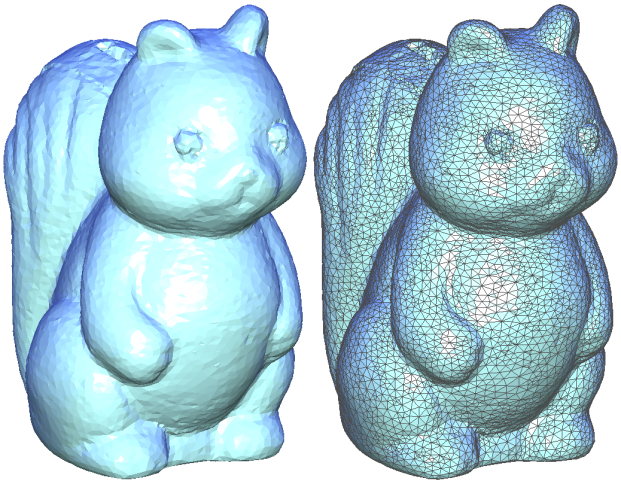
SORKINE, O., AND COHEN-OR, D. 2004. Least-squares meshes. In *Proceedings of Shape Modeling International*, 191–199.

SORKINE, O., AND NEALEN, A. 2006. A note on Laplacian mesh smoothing. *Submitted for publication*.

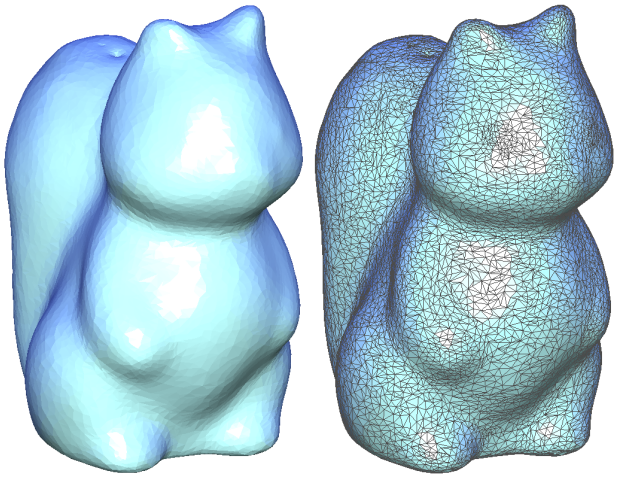
SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, 179–188.



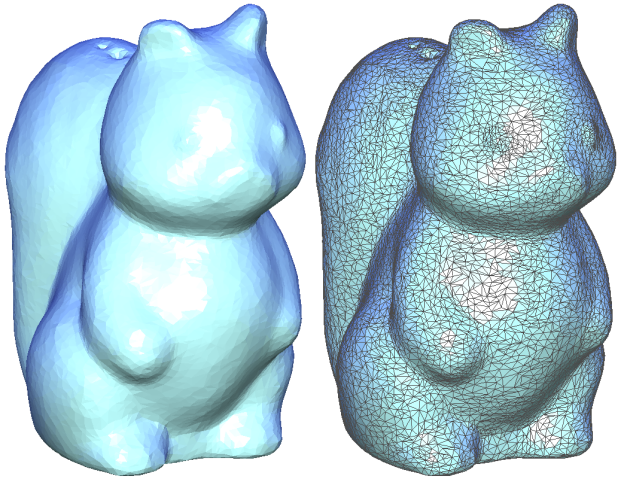
(a) original



(b) triangle optimization with $\mathbf{W}_{cdf}$



(c) smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{L}_{c\overline{\kappa}}$
(outer fairness only, inner fairness untouched)



(d) feature preserving smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{W}_L\mathbf{L}_c$
(outer fairness only, inner fairness untouched)

Figure 12: SQUIRREL results.

SORKINE, O., COHEN-OR, D., IRONY, D., AND TOLEDO, S. 2005. Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics 11*, 2, 171–180.

SURAZHSKY, V., AND GOTSMAN, C. 2003. Explicit surface remeshing. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 20–30.

SURAZHSKY, V., ALLIEZ, P., AND GOTSMAN, C. 2003. Isotropic remeshing of surfaces: a local parameterization approach. In *proceedings of 12th International Meshing Roundtable*, 215–224.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 351–358.

TAUBIN, G., 2000. Geometric signal processing on polygonal meshes. EUROGRAPHICS state-of-the-art report.

TOLEDO, S. 2003. TAUCS*: A Library of Sparse Linear Solvers*. Tel Aviv University.

TUKEY, J. W. 1977. *Exploratory Data Analysis*. Addison-Wesley.

TURK, G. 1992. Re-tiling polygonal surfaces. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 55–64.

VORSATZ, J., RÖSSL, C., KOBBELT, L., AND SEIDEL, H.-P. 2001. Feature sensitive remeshing. *Computer Graphics Forum 20*, 3, 393–401.

VORSATZ, J., RÖSSL, C., AND SEIDEL, H.-P. 2003. Dynamic remeshing and applications. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, 167–175.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with Poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3, 644–651.