Mesh Denoising via L_0 Minimization

Lei He
Texas A&M University

Scott Schaefer
Texas A&M University

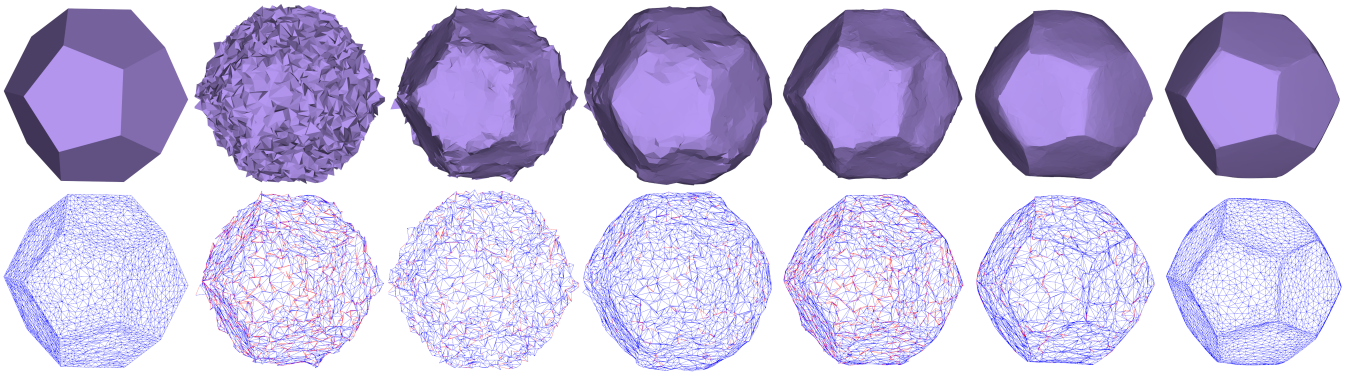

Figure 1: *From left to right: initial surface, surface corrupted by Gaussian noise in random directions with standard deviation $\sigma = 0.4l_e$ (l_e is the mean edge length), bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], bilateral normal filtering [Zheng et al. 2011], our method. The wireframe shows folded triangles as red edges.*

Abstract

We present an algorithm for denoising triangulated models based on L_0 minimization. Our method maximizes the flat regions of the model and gradually removes noise while preserving sharp features. As part of this process, we build a discrete differential operator for arbitrary triangle meshes that is robust with respect to degenerate triangulations. We compare our method versus other anisotropic denoising algorithms and demonstrate that our method is more robust and produces good results even in the presence of high noise.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: mesh denoising, L_0 minimization

Links: DL PDF

1 Introduction

Mesh denoising is an important tool in geometry processing. Surfaces obtained through a scanning process [Levoy et al. 2000] or other reconstruction algorithm are inevitably noisy, even when using high-fidelity scanners. Hence, these surfaces may need to be denoised both for aesthetic reasons and for further geometry processing. However, mesh denoising is inherently challenging as it can be difficult to distinguish features from noise. This problem is especially problematic in the presence of sharp features, which represent high frequency information, and retaining such features can be difficult when high levels of noise are present.

A wide variety of mesh denoising algorithms already exist. While most early work focused on isotropic algorithms that ignore sharp features, recent methods are anisotropic and attempt to preserve sharp features in the data. These methods can be divided into two approaches. The first approach are methods based on prescribed differential information, such as mean curvature. The second approach is to extend the bilateral filter from 2D signal processing to arbitrary 3D meshes in various different fashions.

Contributions
In this paper, we take a different approach to mesh denoising using L_0 minimization. In our context, we use the L_0 norm, which directly measures sparsity, to preserve sharp features and smooth the remainder of the surface. However, the L_0 norm can be difficult to optimize due to its discrete, combinatorial nature. We base our approach on recent work on L_0 minimization for images [Xu et al. 2011]. Doing so requires extending various elements of the minimization from 2D grids of pixels to unstructured triangle meshes representing two-manifolds in $\mathbb{R}^3$. Moreover, our goal is not to create piecewise constant functions as was done for images, but to minimize the curvature of the surface except at sharp features. The benefit of L_0 minimization is that our method handles large amounts of noise and produces higher quality results than current algorithms. In particular, we

- show how to extend L_0 minimization from images to surfaces;
- develop a discrete differential operator to measure planarity of the surface that is robust to poor meshes including those with degenerate triangles;
- integrate a fairing term into the L_0 minimization that improves mesh quality and reduces folded triangles.

2 Related Work

Most early surface smoothing methods are isotropic, which means the filter is independent of surface geometry. Laplacian smooth-

ing [Vollmer et al. 1999] is an example of a simple smoothing algorithm that filters noise efficiently but does not preserve features and shrinks the surface. Taubin [1995] approaches surface smoothing from a signal processing perspective and proposes a non-shrinking, two-step smoothing algorithm. Desbrun et al. [1999] introduces a version of mean curvature flow [Dziuk 1990] for surface fairing using a simplified mass matrix. Kim et al. [2005] combine these two approaches to design a filtering framework for lowpass/highpass filtering with exaggeration and attenuation options. Liu et al. [2002] present a smoothing approach for triangle meshes that preserves volume. Others construct isotropic smoothing methods using global systems of equations [Nealen et al. 2006; Su et al. 2009].

Given that isotropic methods do not preserve sharp features in the object, many recent techniques have focused on anisotropic approaches. Several authors explore anisotropic diffusion algorithms for surfaces [Desbrun et al. 2000; Clarenz et al. 2000; Tasdizen et al. 2002; Bajaj and Xu 2003] or images [Tschumperlé 2006] based on PDEs. Hildebrandt et al. [2004] propose a smoothing algorithm using prescribed mean curvature flow to preserve surface features.

Another approach to anisotropic smoothing has been to extend the bilateral filter [Tomasi and Manduchi 1998] from image processing to 3D geometry. Fleishman et al. [2003] propose a bilateral filter inspired approach that filters vertices of the mesh in the normal direction of the surface using local neighborhoods. Jones et al. [2003] present a similar approach as well based on robust statistics. El Ouafdi et al. [2008] present a probabilistic smoothing algorithm that designs a Riemannian distance based diffusion tensor for filtering neighboring vertices.

Several researchers have also explored the idea of filtering face normals instead of directly filtering vertex coordinates. Many of these methods follow a two-step framework: filtering surface normals followed by updating vertex positions [Taubin 2001; Yagou et al. 2002; Yagou et al. 2003; Shen and Barner 2004; Lee and Wang 2005; Sun et al. 2007; Fan et al. 2010; Zheng et al. 2011]. While updating vertices is trivial, filtering normals is important in the quality of the final surface. Yagou et al. [2002] use mean and median filters for estimating face normals and later use alpha-trimming filters [Yagou et al. 2003]. Shen et al. [2004] propose a fuzzy median filter to better estimate face normals. Sun et al. [Sun et al. 2007] improve upon this method by ignoring neighboring normals with large differences when computing face normals and propose a new vertex updating algorithm. These authors then introduce a random walk model to determine the filtering weights [Sun et al. 2008]. The bilateral filter has been used in normal filtering as well [Lee and Wang 2005; Wang 2006]. Most recently Zheng et al. [2011] propose a mesh denoising scheme using a global bilateral normal filter and achieve impressive results.

3 L_0 Minimization for Images

We will briefly review L_0 minimization in the context of images before extending this algorithm to surfaces. The L_0 norm of a vector is the number of non-zero entries, which directly measures sparsity. L_0 minimization has applications in compressed sensing [Donoho 2006]. However, this norm is difficult to optimize directly due to its combinatorial nature. Candes et al. [2006] show that L_1 minimization can also provide a good measure of sparsity. Lipman et al. [2007] also used the "L_1" norm in the context of point denoising.

Recently Xu et al. [2011] provide an algorithm for directly optimizing the L_0 norm in the context of image smoothing to create piecewise constant images. Let c be a vector of pixel colors and ∇c be a vector of gradients of these colors. The authors attempt to minimize $|c - c^*|^2 + |\nabla c|_0$ where $|\nabla c|_0$ is the L_0 norm of ∇c and c^* represents the original image colors to provide a data fidelity term.

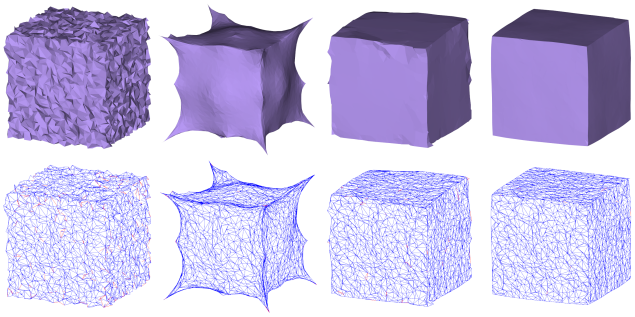

Figure 2: *From left to right: noisy input surface with $\sigma = 0.3l_e$, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.*

To minimize this expression, the authors introduce a set of auxiliary variables δ. The minimization problem then becomes

$$\min_{c,\delta} |c - c^*|^2 + \beta|\nabla c - \delta|^2 + \lambda|\delta|_0$$

where λ controls the level of detail in the final image. The authors optimize this expression with an alternating optimization. First, the authors hold c constant and only minimize for δ.

$$\min_{\delta} \beta|\nabla c - \delta|^2 + \lambda|\delta|_0$$

In this minimization, each entry δ_i will either be 0 or ∇c_i to either minimize the L_0 norm of δ_i or the L_2 difference with ∇c_i. Therefore, if $\sqrt{\frac{\lambda}{\beta}} > \nabla c_i$, δ_i will be set to 0; otherwise $\delta_i = \nabla c_i$. Next, the authors hold δ fixed and optimize for c.

$$\min_{c} |c - c^*|^2 + \beta|\nabla c - \delta|^2$$

This expression is quadratic in c and trivial to minimize. Both of these optimizations alternate until convergence, except the authors multiply β by 2 each iteration to eventually force ∇c to match δ.

4 L_0 Minimization for Surfaces

We will assume that we are given a triangulated manifold, with or without boundary, containing vertices p. For surfaces, we can simply replace c with p and its initial positions p^*. However, we must design a discrete differential operator to replace ∇c that is zero when the surface is flat for arbitrary triangulations irrespective of the rotation or translation of the surface. This constraint implies that we need some form of second order information rather than the first order information provided by ∇c. While there are several candidates for measuring this information on surfaces [Mallet 1989], an obvious choice is the discrete Laplacian operator [Pinkall and Polthier 1993], which is computed as a weighted combination of a vertex and its one-ring where the weights are given by cotangents of angles of the triangles. This operator has found numerous applications in Computer Graphics and has the property that its value, when applied to the vertices, yields a vector normal to the surface whose magnitude is proportional to the mean curvature of the surface at that point [Desbrun et al. 1999].

Figure 2 shows a noisy input surface (left) and the effect of using the discrete Laplacian operator in this L_0 minimization framework

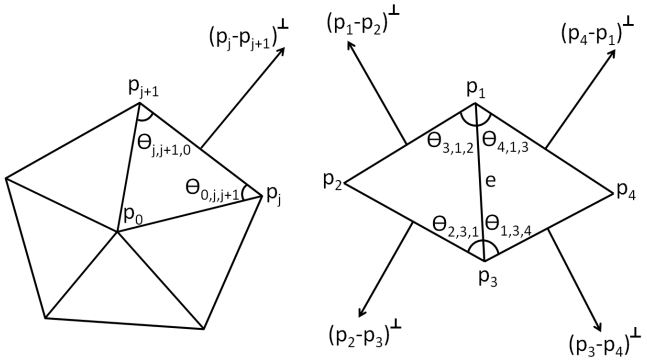

Figure 3: *Our notation for the one-ring of a vertex and an edge.*

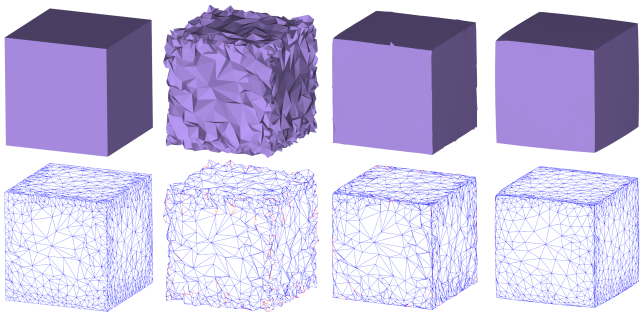

Figure 4: *From left to right: the ground truth, the input mesh with large noise in random directions, our method without regularization, our method with regularization.*

(middle left). While the surface is smoother, the optimization fails to reproduce sharp features and shrinks the surface away from the features. The problem is that the vertex-based Laplacian only constrains the mean curvature vector as opposed to a metric that directly measures sharpness per edge. Hence, we will generalize the construction of the vertex-based cotan operator to an operator that acts directly on an edge.

4.1 Differential Edge Operator

To make the operator independent of translations and rotations, we desire a set of weights w_j that annihilate constant and linear functions; that is,

$$\begin{array}{rcl} \sum_j w_j & = & 0 \\ \sum_j w_j p_j & = & 0 \end{array} \quad (1)$$

when p_j are planar, which are similar to the properties of generalized barycentric coordinates [Floater et al. 2006]. While there have been many different derivations of the cotan Laplacian operator in Computer Graphics, we focus on the barycentric construction that makes the properties in Equation 1 obvious. We will then extend this construction to build a differential edge operator.

One method of building barycentric coordinates is through the use of the divergence theorem [Schaefer et al. 2007], which states that the integral of an outward facing normal over a closed shape is zero. Figure 3 (left) shows a one-ring of a central vertex and outward facing normals $(p_j - p_{j+1})^\perp$ whose lengths are equal to the lengths of the corresponding edge and are planar with the triangle formed between p_j, p_{j+1}, and p_0. Representing $(p_j - p_{j+1})^\perp$ as a weighted combination of the vertices of its triangle yields

$$(p_j - p_{j+1})^\perp = \cot(\theta_{0,j,j+1})(p_{j+1} - p_0) + \cot(\theta_{j,j+1,0})(p_j - p_0).$$

Summing these weights around the central vertex gives the discrete Laplacian from Pinkall et al. [1993]. If the p_j are planar, then

$$\sum_j (p_j - p_{j+1})^\perp = \sum_j w_j p_j = 0$$

and the weights trivially satisfy Equation 1.

We can apply the same construction to build an edge operator. Figure 3 (right) shows an edge, e, and the labels we use to identify the vertices. If we apply the same construction and represent the vectors $(p_i - p_{i+1})^\perp$ as a weighted combination of the vertices of their triangle, we obtain the differential operator $D(e)$ for the edge as

$$D(e) = \begin{bmatrix} -\cot(\theta_{2,3,1}) - \cot(\theta_{1,3,4}) \\ \cot(\theta_{2,3,1}) + \cot(\theta_{3,1,2}) \\ -\cot(\theta_{3,1,2}) - \cot(\theta_{4,1,3}) \\ \cot(\theta_{1,3,4}) + \cot(\theta_{4,1,3}) \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}, \quad (2)$$

which also satisfies Equation 1. Moreover, when the p_j are not planar, the magnitude of the weights applied to the vertices is equal to $2\sin\left(\frac{\gamma}{2}\right)|p_3 - p_1|$ where γ is the dihedral angle between the two polygons. $2\sin\left(\frac{\gamma}{2}\right)$ is a good approximation of γ for angles less than 90°. Hence, the magnitude of the weights applied to the vertices is approximately the dihedral angle times the shared edge length, which provides a measure of the mean curvature.

Figure 2 (middle right) shows the effect of using this cotan edge operator in the L_0 optimization. The result is significantly improved, but there are still shape quality problems. The issue stems from degenerate triangles where the cotan weights approach infinity as an angle approaches zero. In practice, this behavior leads to numerical problems and never allows folded triangles to unfold in planar configurations since the vertex must pass through a singularity in the weights. Kazhdan et al. [2012] have noted this problem before for the cotan vertex operator in the context of mean curvature flow.

To improve our edge operator, we return to the properties in Equation 1. If we assume that the p_j are in 2D, then Equation 1 represents three equations with four unknowns that has exactly a one-dimensional null space of solutions spanned by the vector

$$\{-\Delta_{2,3,4}, \Delta_{1,3,4}, -\Delta_{1,2,4}, \Delta_{1,2,3}\}$$

where $\Delta_{j,k,\ell}$ refers to the area of the triangle with vertices p_j, p_k, and p_ℓ. These weights are not scale-independent and require normalization. We use $\Delta_{1,3,4} + \Delta_{1,2,3}$ to normalize the result. Note that this denominator may be zero if both triangles become degenerate. For the cotan weights from Equation 2, the weights are undefined if *either* triangle becomes degenerate. We use this local normalizer for all of our results and never had any issue on any of the surfaces we tried, though we typically saw numerical problems with the cotan weights for meshes with large amounts of noise. However, it is also possible to use a global normalizer such as the surface area of the model to avoid all but global degeneracies. Note that our cotan weights in Equation 2 are a scalar multiple of this null space vector. Therefore, the magnitude of these new weights applied to the vertices is also proportional to mean curvature.

Computing these area weights is trivial in 2D, but requires some thought when p_j are in 3D. Moreover, the weights do not take into account the asymmetry in the vertices due to the edge between p_1 and p_3. For example, if p_1 lies in the triangle defined by p_2, p_3, and p_4, the shape is a valid planar triangulation whose outer edges represent a concave polygon and our operator should return zero. However, if p_2 lies in the triangle defined by p_1, p_3, and p_4, the configuration is that of a folded-back triangle and our operator should be non-zero. Our solution is to compute the areas $\Delta_{2,3,4}$ and $\Delta_{1,2,4}$ using an isometric unfolding of the surface around the shared edge

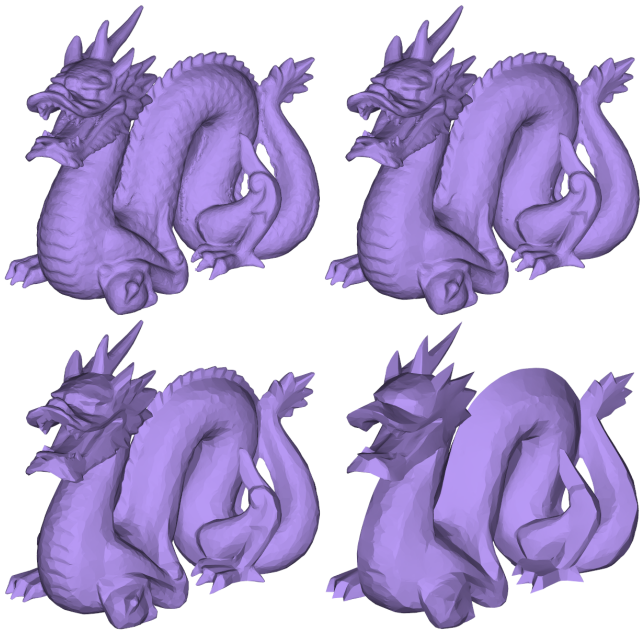

Figure 5: *The effect of the L_0 weight. Top left: an input mesh without any noise, top right: $\lambda = \frac{1}{16}$ default, bottom left: $\lambda =$ default, bottom right: $\lambda = 16$ default.*

(i.e.; $\Delta_{2,3,4} = \frac{1}{2}|p_2 - p_3||p_4 - p_3| \sin(\theta_{1,3,4} + \theta_{2,3,1})$). Expanding this equation yields our area-based edge operator.

$$D(e) = \begin{bmatrix} \frac{\Delta_{1,2,3}((p_4-p_3)\cdot(p_3-p_1))+\Delta_{1,3,4}((p_1-p_3)\cdot(p_3-p_2))}{|p_3-p_1|^2(\Delta_{1,2,3}+\Delta_{1,3,4})} \\ \frac{\Delta_{1,3,4}}{\Delta_{1,2,3}+\Delta_{1,3,4}} \\ \frac{\Delta_{1,2,3}((p_3-p_1)\cdot(p_1-p_4))+\Delta_{1,3,4}((p_2-p_1)\cdot(p_1-p_3))}{|p_3-p_1|^2(\Delta_{1,2,3}+\Delta_{1,3,4})} \\ \frac{\Delta_{1,2,3}}{\Delta_{1,2,3}+\Delta_{1,3,4}} \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

Figure 2 (right) shows the effect of this area-based operator. The result is a much improved surface with far fewer folded triangles.

4.2 Regularization

For surfaces with relatively uniformly shaped triangles, the method in Section 4.1 works well. However, under high amounts of noise with non-uniformly shaped triangles, the optimization still produces flat surfaces but polygons can fold and overshoot sharp edges as shown in Figure 4 (middle right). Our solution is to add a triangle shape regularizer for each edge given by the quadratic

$$(p_1 - p_2 + p_3 - p_4)^2. \tag{3}$$

Figure 4 (right) shows the result of adding such a regularizer. Not only are the triangles better shaped, but the spurious overshoots and fold-backs have been eliminated.

4.3 Optimization

Our optimization follows that of Section 3, and we minimize

$$\min_{p,\delta} |p - p^*|^2 + \alpha|R(p)|^2 + \beta|D(p) - \delta|^2 + \lambda|\delta|_0 \tag{4}$$

where p are the vertices of the shape, p^* are their initial positions, $D(p)$ is a vector where the i^{th} entry corresponds to the area-based edge operator applied to the i^{th} edge, and $R(p)$ is a vector whose

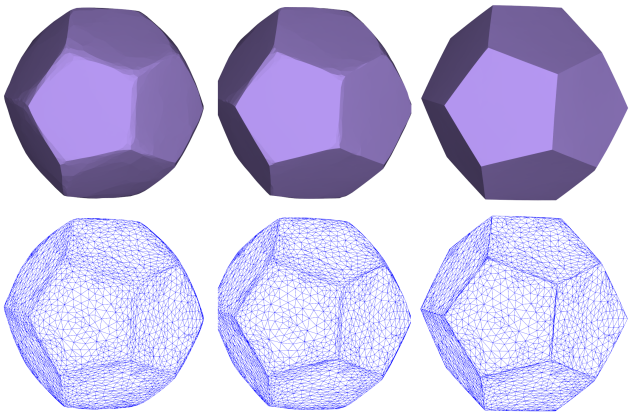

Figure 6: *From left to right shows the results of our method using different speeds μ of 2.0, 1.414 (default) and 1.090. In this example, smaller values of μ produces near ideal results.*

i^{th} entry is the edge regularizer from Equation 3 applied to the i^{th} edge. We again perform an alternating minimization where we hold p fixed to solve for δ,

$$\min_{\delta} \beta|D(p) - \delta|^2 + \lambda|\delta|_0 \tag{5}$$

and then hold δ fixed and solve for p in the same way as Section 3

$$\min_{p} |p - p^*|^2 + \alpha|R(p)|^2 + \beta|D(p) - \delta|^2, \tag{6}$$

which represents a sparse quadratic in p. This entire procedure is summarized in Algorithm 1.

Algorithm 1 Surface smoothing via L_0 minimization

Input: surface with vertices p^*
Initialization: compute λ, $p \leftarrow p^*$, $\beta \leftarrow 10^{-3}$, $\alpha \leftarrow \alpha_0$
repeat
  fix p, solve for δ in (5).
  fix δ, solve for p in (6).
  $\beta \leftarrow \mu\beta$, $\alpha \leftarrow \frac{1}{2}\alpha$
until $\beta \geq 10^3$

In this procedure μ is the speed at which we increase β. We multiply the weight of the regularizer by $\frac{1}{2}$ at each iteration to exponentially decrease the effect of the regularizer during the optimization. This choice gives high weight to the regularizer at the beginning of the optimization when the surface is very noisy and reduces its effect to zero as the optimization continues.

5 Results

Equation 4 contains several parameters. In all of our results, unless otherwise noted to show the effect of a particular parameter, we use default values and do not optimize the parameters for a particular surface. We use $\mu = \sqrt{2}$, $\alpha_0 = 0.1\bar{\gamma}$, and $\lambda = 0.02 l_e^2 \bar{\gamma}$ where l_e is the average edge length of the initial surface and $\bar{\gamma}$ is the average dihedral angle measured in radians from the initial surface, which provides a measure of the initial amount of noise in the surface. Since the L_0 norm does not change with the scale of the surface, we scale λ by l_e^2 to make the result scale independent. In Figures 2 and 4 where we do not use regularization (i.e.; $\alpha_0 = 0$), we increase λ by a factor of four to make up for the lack of smoothing from the regularizer in the first few iterations.

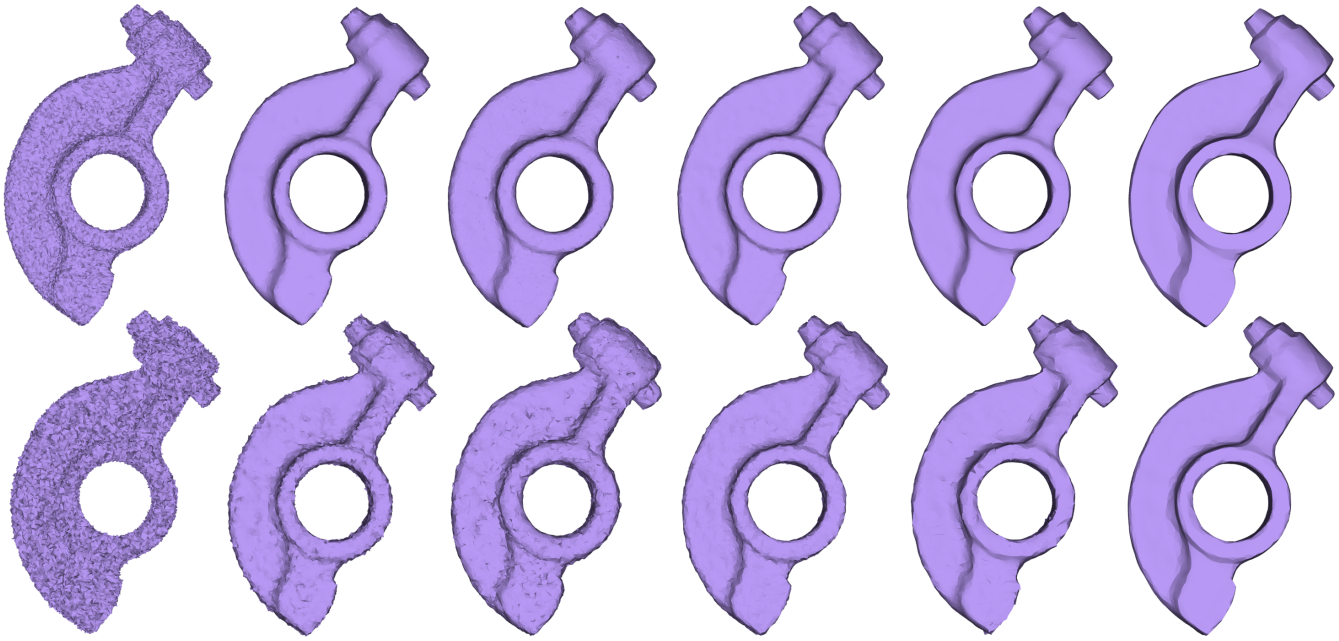

Figure 7: *The performance of several methods with different amount of noise. Each row shows different levels of noise in random directions ($\sigma = 0.3l_e$ top, $\sigma = 0.6l_e$ bottom). From left to right: noisy input, bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], bilateral normal filtering [Zheng et al. 2011], our method.*

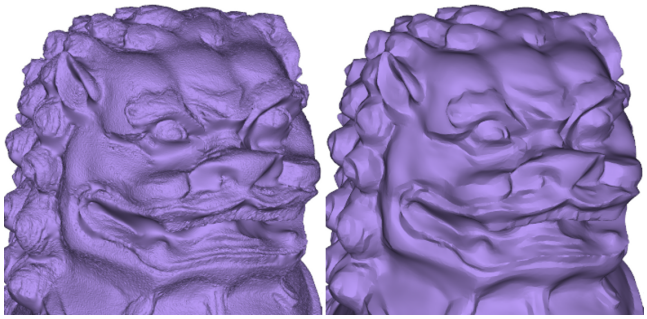

Figure 8: *A model scanned with a laser range scanner. The noisy input surface (left) and our result (right).*

As in Section 3, λ provides a measure of the level of detail to preserve in the input surface. Figure 5 shows the effect of different values of λ on a surface without noise. A small value for λ only removes small scale details in the surface such as the bumps along the dragon's body. Increasing the value of λ gradually removes more details until only large-scale features are preserved.

Figure 6 shows the effect of changing the speed μ at which we increase β for the input from Figure 1. Xu et al. [2011] use $\mu = 2.0$, but we found that the results typically had more rounded corners with this choice. As μ decreases, edges and corners become sharper at the cost of more iterations. In this example, the surface is a platonic solid and using a small value for μ (1.090) can provide an even better result than our default parameter. However, for shapes with curved regions such as in Figure 7, we have found that $\mu = \sqrt{2}$ tends to work best, which is the parameter we used in Figure 1.

We have also compared our method against several popular and recent anisotropic smoothing methods. For each model we create an input surface corrupted by Gaussian noise with a standard deviation σ. In these comparisons, we show both the surface and a wireframe model. We highlight edges with a red color when the dihedral angle is greater than 150° to show folded triangles. Figure 1 illustrates a comparison between bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], and bilateral normal filtering [Zheng et al. 2011]. All of these methods have some parameters that control their results. While we use default parameters for our method, we searched the parameter space of the other methods to find an optimal set of parameters for each model. As Figure 6 illustrates, we can improve upon our result by tuning parameters. However, our default parameters provide a superior solution by themselves.

Figure 7 shows a comparison where we vary the level of noise for the different methods. In low noise situations, all of the methods perform well. As the noise level increases, other methods are unable to remove all of the noise in the resulting surface, whereas our method still produces a high quality result. Figures 8, 9, and 11 also compare these methods on a variety of different surfaces.

Our implementation uses TAUCS [Sivan Toledo and Rotkin 2001] to solve the system of sparse equations in each iteration of the optimization. We used a Intel Core i7 3770K to perform our tests and our times range from about 2 seconds for Figures 1 and 4, which have about 3800 vertices each, to about 3 minutes for the statue in Figure 11, which has 134345 vertices. We have found that 90% of the execution time is taken by solving the system of equations. Since the equations change at each iteration, we cannot simply prefactor the matrix, which leads to longer running times.

Limitations: Our method can fail to produce good results in some cases. Figure 10 shows a CAD shape with an extreme triangulation in that the only vertices that exist lie at sharp features in the model. In this case our method does a good job in some of the cylindrical areas but fails to reproduce the teeth in the gears.

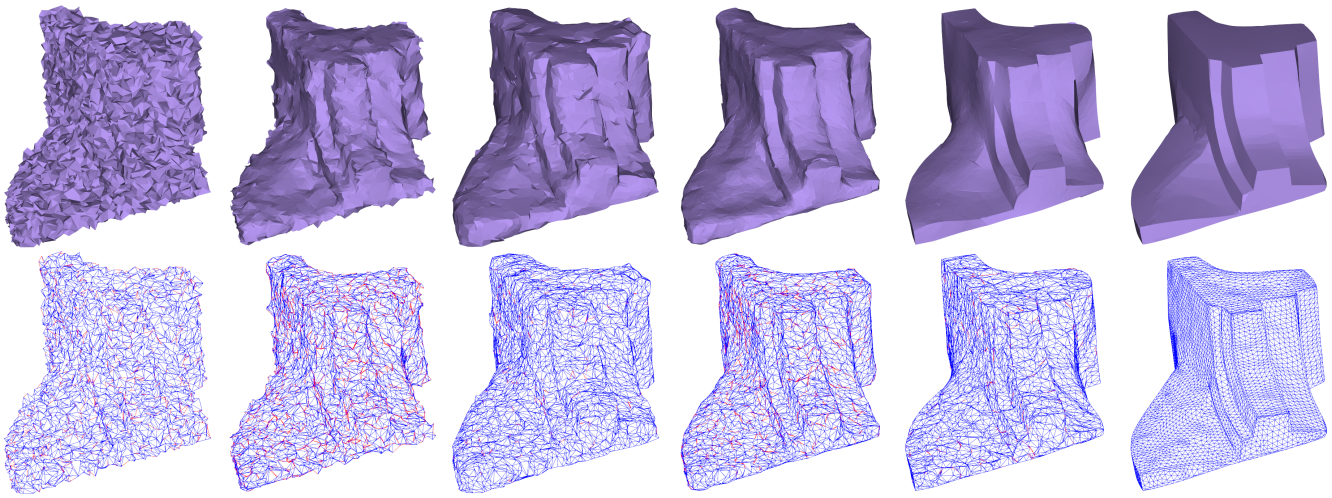

Figure 9: *From left to right: the input mesh with large noise in random directions, bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], bilateral normal filtering [Zheng et al. 2011], our result. We show the wireframe of each surface below.*

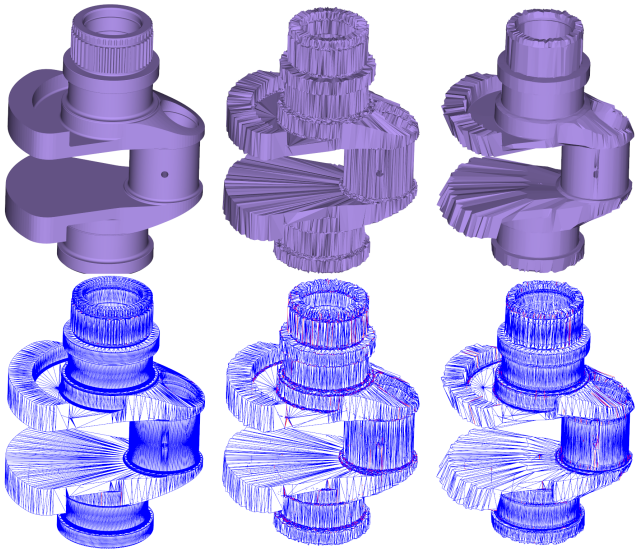

Figure 10: *A failure case. From left to right: the ground truth with an extreme triangulation, the noisy input, our result.*

Acknowledgements

This work was supported by NSF CAREER award IIS 1148976. We would like to thank Jason Smith, the AIM Shape Repository, and the Stanford 3D Scanning Repository for models in this paper.

References

BAJAJ, C. L., AND XU, G. 2003. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans. Graph. 22*, 1, 4–32.

CANDES, E., ROMBERG, J., AND TAO, T. 2006. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory 52*, 2, 489–509.

CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. VIS, 397–405.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. SIGGRAPH, 317–324.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 2000. Anisotropic feature-preserving denoising of height fields and bivariate data. Graphics Interface, 145–152.

DONOHO, D. 2006. Compressed sensing. *IEEE Transactions on Information Theory 52*, 4, 1289–1306.

DZIUK, G. 1990. An algorithm for evolutionary surfaces. *Numerische Mathematik 58*, 1, 603–611.

EL OUAFDI, A. F., ZIOU, D., AND KRIM, H. 2008. A smart stochastic approach for manifolds smoothing. In *Proceedings of the Symposium on Geometry Processing*, 1357–1364.

FAN, H., YU, Y., AND PENG, Q. 2010. Robust feature-preserving mesh denoising based on consistent subneighborhoods. *IEEE Trans. Vis. Comp. Graph. 16*, 2, 312–324.

FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. SIGGRAPH, 950–953.

FLOATER, M., HORMANN, K., AND KOS, G. 2006. A general construction of barycentric coordinates over convex polygons. *Advances in Comp. Math 24*, 311–331.

HILDEBRANDT, K., AND POLTHIER, K. 2004. Anisotropic filtering of non-linear surface features. *Computer Graphis Forum 23*, 3, 391–400.

JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. SIGGRAPH, 943–949.

KAZHDAN, M., SOLOMON, J., AND BEN-CHEN, M. 2012. Can mean-curvature flow be modified to be non-singular? *Computer Graphics Forum 31*, 5, 1745–1754.

KIM, B., AND ROSSIGNAC, J. 2005. Geofilter: Geometric selection of mesh filter parameters. *Computer Graphis Forum 24*, 3, 295–302.

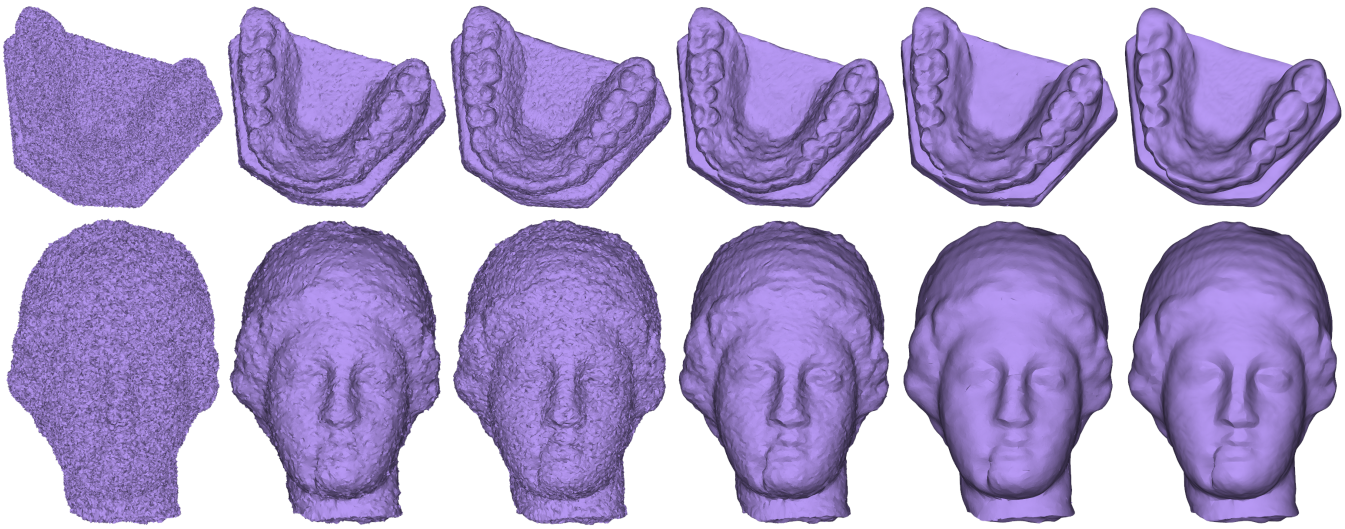

Figure 11: *Left to right: the input mesh with large noise in random directions, bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], bilateral normal filtering [Zheng et al. 2011], our result.*

LEE, K.-W., AND WANG, W.-P. 2005. Feature-preserving mesh denoising via bilateral normal filtering. In *Proceedings of Computer Aided Design and Computer Graphics*, 275–280.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital michelangelo project: 3d scanning of large statues. SIGGRAPH, 131–144.

LIPMAN, Y., COHEN-OR, D., LEVIN, D., AND TAL-EZER, H. 2007. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph. 26*, 3, 22:1–22:5.

LIU, X., BAO, H., SHUM, H.-Y., AND PENG, Q. 2002. A novel volume constrained smoothing method for meshes. *Graphical Models 64*, 169–182.

MALLET, J.-L. 1989. Discrete smooth interpolation. *ACM Trans. Graph. 8*, 2, 121–144.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2006. Laplacian mesh optimization. GRAPHITE, 381–389.

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

SCHAEFER, S., JU, T., AND WARREN, J. 2007. A unified, integral construction for coordinates over closed curves. *Computer Aided Geometric Design 24*, 8-9, 481–493.

SHEN, Y., AND BARNER, K. E. 2004. Fuzzy vector median-based surface smoothing. *IEEE Trans. Vis. Comp. Graph. 10*, 3, 252–265.

SIVAN TOLEDO, D. C., AND ROTKIN, V. 2001. Taucs: A library of sparse linear solvers.

SU, Z., WANG, H., AND CAO, J. 2009. Mesh denoising based on differential coordinates. Shape Modeling International, 1–6.

SUN, X., ROSIN, P. L., MARTIN, R. R., AND LANGBEIN, F. C. 2007. Fast and effective feature-preserving mesh denoising. *IEEE Trans. Vis. Comp. Graph.*, 925–938.

SUN, X., ROSIN, P. L., MARTIN, R. R., AND LANGBEIN, F. C. 2008. Random walks for feature-preserving mesh denoising. *Computer Aided Geometric Design 25*, 7, 437–456.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2002. Geometric surface smoothing via anisotropic diffusion of normals. VIS, 125–132.

TAUBIN, G. 1995. A signal processing approach to fair surface design. SIGGRAPH, 351–358.

TAUBIN, G. 2001. Linear anisotropic mesh filtering. *IBM Research Report RC22213(W0110-051)*.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision*, 839–846.

TSCHUMPERLÉ, D. 2006. Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's. *Int. J. Comput. Vision 68*, 1, 65–82.

VOLLMER, J., MENCL, R., AND MLLER, H. 1999. Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum 18*, 3, 131–138.

WANG, C. C. L. 2006. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. *IEEE Trans. Vis. Comp. Graph. 12*, 4, 629–639.

XU, L., LU, C., XU, Y., AND JIA, J. 2011. Image smoothing via l0 gradient minimization. *ACM Trans. Graph. 30*, 6, 174:1–174:12.

YAGOU, H., OHTAKE, Y., AND BELYAEV, A. 2002. Mesh smoothing via mean and median filtering applied to face normals. GMP, 124–131.

YAGOU, H., OHTAKE, Y., AND BELYAEV, A. G. 2003. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. *Computer Graphics International Conference*, 28–33.

ZHENG, Y., FU, H., AU, O. K.-C., AND TAI, C.-L. 2011. Bilateral normal filtering for mesh denoising. *IEEE Trans. Vis. Comp. Graph. 17*, 10, 1521–1530.