DIFFERENTIAL FILTERING AND DETEXTURING

A Dissertation

by

LEI HE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Scott Schaefer
Committee Members,	John Keyser
	Jianer Chen
	Ergun Akleman
Head of Department,	Nancy Amato

August 2014

Major Subject: Computer Science

ABSTRACT

Extracting valuable information from 2D or 3D visual data plays an important role in image and geometry processing. Surfaces obtained through a scanning process or other reconstruction algorithms are inevitably noisy due to error in the scanning process and resampling of the data at various processing steps. These surfaces need to be denoised both for aesthetic reasons and for further geometry processing. Similarly, extracting or removing texture patterns from 2D or 3D data is challenging due to the complication of its statistical features. In this dissertation, I describe how to remove surface noise and image texture patterns. In particular, I focus on denoising triangulated models based on L_0 minimization, in which a very important discrete differential operator for arbitrary triangle meshes has been developed. Compared to other anisotropic denoising algorithms, our method is more robust than other anisotropic denoising algorithms, and produces good results even in the presence of high noise. I also introduce how to use bilateral filter appropriately on image texture removal by modifying its range image. While current existing methods either fail to remove the textures completely or over blur main structures, our method delivers best-in-class image detexturing performance.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Scott Schaefer, for his guidance and support throughout my graduate studies. The completion of this dissertation would have been impossible without his help. I have never forgotten that it was Scott who gave me a chance to convince him to take me on as a student when I was in trouble. I have never forgotten that it was Scott who has been patient when I have made mistakes. Scott has always done his best to help his students. It is through Scott that I have learned to conduct high quality research, write technical papers, present in front of audience and even speak English.

I would also like to thank my committee members, John Keyser, Jianer Chen, and Ergun Akleman for their comments and support on this work. I took physically based modeling with John Keyser and image synthesis with Ergun Akleman. I am so grateful to all my committee members for approving my petition of course work so quickly, otherwise I wouldn't be able to graduate on time. Jianer has always replied to my email within 24 hours, even when he was out of the country. I have always had great time with lab friends, Josiah Manson, Jason Smith, Ruoguan Huang, Shu-wei Hsu, Donghui Han and Songgang Xu−−you guys are awesome!

Most of my gratitude goes to my wife, Yuan. I cannot thank her enough for overcoming the hardship and staying abroad with me, handling all the house hold cares, being patient with me playing video games.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

With the increasing use of visual data to represent 2D scenes or 3D models, there is a rising need for processing [31] those data. Image denoising has been studied for decades, but even with the tremendous amount of work on the topic, current techniques are far from perfection. Indeed, data acquisition always comes with some kind of noise, so modeling this noise and removing it efficiently is important. In this dissertation, I will focus on solving two visual data filtering problems: surface denoising and image detexturing. Denoising is a classical and well studied problem, while detexturing is relatively less studied.

1.1 Surface Filtering

Surface processing is a fast-growing area of research in the computer graphics domain and is mostly involved in applying a variety of algorithms, such as model repair, surface smoothing, surface parameterization, remeshing, and surface deformation to geometric models. Triangle meshes are a popular way to represent surfaces for geometry processing because they are flexible and highly efficient. Triangle meshes come from a variety of sources but can be generated through scans of real-world objects. However, 3D scans often contain artifacts and noise. The goal of surface filtering is to extract high quality surfaces from noisy scan data. While dealing with noise, this refinement process is referred to as surface denoising. Surface denoising is an important tool in geometry processing. Surfaces obtained through a scanning process or other reconstruction algorithms are inevitably noisy due to error in scanning process and resampling of the data at various processing steps. These surfaces need to be denoised both for aesthetic reasons and for further geometry processing. Figure 1.1 shows a noisy model obtained by a laser range scanner. However, surface denoising

Figure 1.1: A real world face model. Model courtesy of the Factum Foundation for Digital Technology in Conservation.

is inherently challenging as it can be difficult to distinguish features from noise. This problem is especially problematic in the presence of sharp features that represent high frequency information. Retaining such features can be difficult when high levels of noise are present with the same frequency as sharp features.

Noise on surfaces is usually treated as perturbation. The most commonly used additive model is

$$p^* = p + \eta, \tag{1.1}$$

where the observed vertex p^* is the addition of the original vertex p and the random noise η, which is usually assumed to be Gaussian with zero mean and standard

deviation σ. Surface denoising boils down to recovering a 3D model contaminated by noise η. The larger the noise intensity σ is, the harder it is for denoising algorithms to distinguish noise from data.

In general, surface denoising methods can be classified into two categories: isotropic and anisotropic. The former filters the noisy surface independently of surface geometry, while the latter takes geometry information into account and modifies the diffusion equation in order to preserve sharp features.

An important goal of surface denoising is to preserve these sharp features, making anisotropic methods preferred. However, in the presence of high levels of noise, traditional anisotropic methods have trouble balancing noise removal and feature preservation. As a result, there is still significant room for improving the quality of mesh filtering that we use for fairing models.

1.2 Image Detexturing

Image processing is any form of signal processing for which the input is an image, such as a photograph or video frame. The output of image processing is either an image or a set of characteristics or parameters related to the image. Most image processing methods treat the image as a two-dimensional signal and apply standard signal processing techniques to it. In this field of research there are many problems dealing with image processing, such as image enhancement, restoration, compression, denoising and so on. While there are many interesting problems, this dissertation's secondary objective is to explore image detexturing.

Texture is one of the most fundamental elements in both 3D and 2D art work, an example of this is shown in Figure 1.2. Physcial texture is defined as the tactile quality of the surface of an object, which refers to how it feels if touched. Visual texture is the visual expression of physical texture. Visual texture in general resides

Figure 1.2: Left: a 3D art statue [3]. Right: a 2D painting by Vincent Van Gogh [2].

in the media such as photography, drawlings and paintings. Texture in these media is generally created by repeated shapes and lines either globally or locally, and stored in images.

Creating visual textures could be time-consuming and tedious even for artists, and some natural textures could even be impossible to reproduce by hand, let alone unexperienced people. Therefore, extracting existing textures from various natural scenes or human-created art work would be quite useful. For example, one can transfer the texture pattern from one image to the other to create interesting image mosaics. Also, similar to the goal of image denoising, removing textures will help us acquire and better understand the latent structures underneath the textured scene. Figure 1.3 shows the result of a detexturing method using relative total variation model [62]. Compared to noise, texture is less studied in the the academic literature, but it widely attracts more attention from artists. There are two main approaches to analyze image texture in computer graphics: structured approach and statistical approach. The structured approach treats an image texture as a set of primitive

Figure 1.3: Left: an input image with repeated texture patterns [63]. Right: results of a detexturing method in [62].

texels in some regular pattern, so it works well when analyzing artificial textures. The statistical approach sees an image texture as a quantitative measure of the distribution of intensities (colors) in a region. This approach in general is easier to compute and more widely used, given the fact that most textures in natural scenes or even art work are made of irregular subelements. Standard approaches like edge detection, co-occurrence matrix, laws texture energy measure and power spectrum can be used to describe textures for different purposes.

While the goal is more towards identifying and extracting textures, I do not in this dissertation answer questions like *"what kinds of texture does the image contain?"* or *"how dense is the texture?"*. Instead, with an existing metric for detecting textures, we focus on answering *"how can we develop a better filtering algorithm to remove or extract textures?"*.

Similar to the noise model on surfaces, a image with texture patterns could be

modeled by

$$c^* = c + \eta, \tag{1.2}$$

where c^* is the input textured image, c is the original image without textures and η is the texture. Ideally, we would like to extract the texture-free image c from the input c^* by removing the texture η. It is difficult to remove texture patterns by traditional filtering methods, this dissertation proposes a technique to overcome this difficulty.

1.3 Data Collection and Representation

In technical applications of 3D computer graphics, surfaces are not the only way of representing objects, other representations include wireframe, solids and point clouds. However, this work only considers surfaces to represent 3D objects. As mentioned before, we use triangle meshes, one of the most common polygon mesh representations, as a discrete representation to approximate 3D objects. A triangle mesh, see Figure 1.4, can be defined as $\mathbb{M} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of vertices, $\mathcal{E} = \{e_i j\}$ is the set of edges. Each edge $\{e_i j\} = [v_i, v_j]$ connects a pair of vertices $\{v_i, v_j\}$. Those geometric elements are stored in a data structure to easily obtain the geometry information associated with any element. For example, one can access the local neighborhood of a vertex and use the neighboring vertices to calculate the curvature at this vertex.

Images are widely used to store and visualize 2D data intuitively. They are generally converted into bit information by image sampling. The images this dissertation uses are diverse, they could be natural photos, paintings or even unrealistic scenes. Like traditional image processing techniques, we perform image detexturing operations in a linear color space. In this work, images are represented in an approximate linear color space. While all the data is discrete, we conduct experiments on both synthetic and real world data to verify the effectiveness of the proposed algorithms.

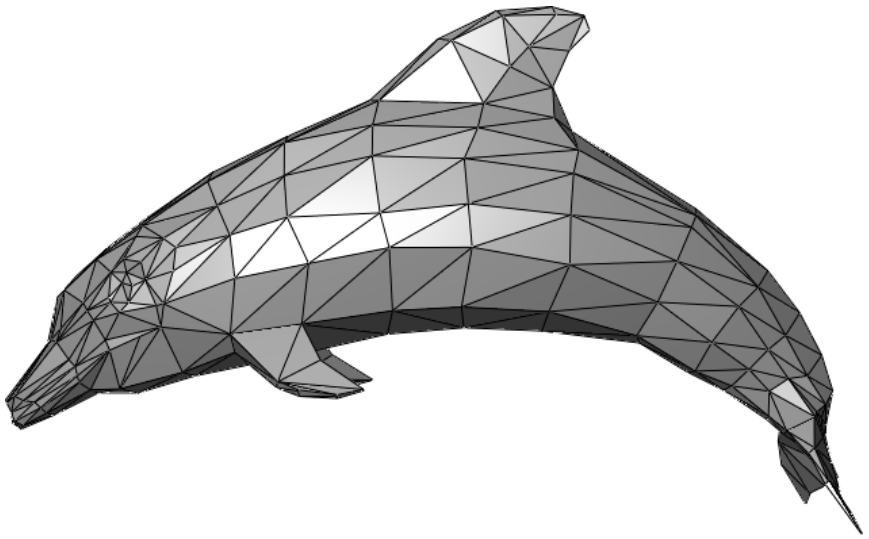


Figure 1.4: A 3D model represented by a triangle mesh

1.4 Dissertation Overview

This dissertation proposes two filtering approaches for extracting smooth surfaces and texture-free images. In particular,

- We extract smooth surfaces by denoising triangluated models with a large amount of noise.
- We acquire texture-free images by detexturing various images with regular and non-regular texture patterns.

2. MESH DENOISING VIA L_0 MINIMIZATION

Mesh denoising is inherently challenging as it can be difficult to distinguish features from noise. When the noise intensity is low, current anisotropic methods can remove noise and preserve sharp feature quite well at the same time. However, things become especially problematic in the presence of high levels of noise.

Generally, feature-preserving denoising is achieved by adjusting vertex positions locally or globally while respecting the underlying geometric features. Anisotropic filtering is often needed to preserve features such as sharp edges or corners. 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 share features in the data. These methods are divided into two major approaches. The first approach, anisotropic diffusion flow, is global and based on prescribed differential information such as mean curvature to adjust the direction of diffusion of the high frequency noise. The second approach is to extend the bilateral filter from 2D signal processing to arbitrary 3D meshes by filtering either spatial locations or differential information locally. For example, one can apply bilateral filters on face normals, then use the filtered face normals to guide surface reconstruction. Instead of directly using vertex locations, most advanced mainstream methods adopt differential information such as normals and curvatures to describe the surface's local appearance. That differential information is either explicitly filtered to guide the vertex updating afterwards, or is implicitly incorporated into an energy minimization framework. It is crucial to choose an appropriate differential shape descriptor not only for denoising methods, but also for other geometry processing algorithms that use differential representations. For instance, a shape descriptor,

which contains desirable anisotropic information of features, may not be well defined within regions with sharp features.

In this chapter, 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 [61]. 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.

2.1 Related Work

Most early surface smoothing methods are isotropic, which means the filter is independent of surface geometry. Laplacian smoothing [59] is an example of a simple smoothing algorithm that filters noise efficiently but does not preserve features and shrinks the surface. Taubin [54] approaches surface smoothing from a signal processing perspective and proposes a non-shrinking, two-step smoothing algorithm. Desbrun et al. [13] introduces a version of mean curvature flow [17] for surface fairing using a simplified mass matrix. Kim et al. [29] combine these two approaches to design a filtering framework for lowpass/highpass filtering with exaggeration and attenuation options. Liu et al. [34] present a smoothing approach for triangle meshes that preserves volume. Others construct isotropic smoothing methods using global systems of equations [40, 49].

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 [14, 12, 53, 8] or images [57] based on PDEs. Hildebrandt et al. [24] 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 [56] from image processing to 3D geometry. Fleishman et al. [22] 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. [26] present a similar approach as well based on robust statistics. El Ouafdi et al. [19] 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 [55, 64, 65, 47, 30, 51, 20, 67]. While updating vertices is trivial, filtering normals is important in the quality of the final surface. Yagou et al. [64] use mean and median filters for estimating face normals and later use alpha-trimming filters [65]. Shen et al. [47] propose a fuzzy median filter to better estimate face normals. Sun et al. [51] 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 [52]. The bilateral filter has been used in normal filtering as well [30, 60]. Most recently Zheng et al. [67] propose a mesh denoising scheme using a global bilateral normal filter and achieve impressive results.

2.2 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 [15]. However, this norm is difficult to optimize directly due to its combinatorial nature. Candes et al. [10] show that L_1 minimization can also provide a good measure of sparsity. Lipman et al. [32] also used the "L_1" norm in the context of point denoising.

Recently Xu et al. [61] 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 as a data fidelity term.

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;

Figure 2.1: Left: an input image. Right: the result of L_0 minimization [61].

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. Figure 2.1 shows a smoothing result of this method.

2.3 L_0 Minimization for Surfaces

Assume that we are given a triangulated manifold $\mathbb{M} = (\mathcal{V}, \mathcal{E})$, with or without boundary, containing vertices $p \in \mathcal{V}$. 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. The reason is that using gradients on the surface will generate piecewise constant surfaces, which are not desirable. 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

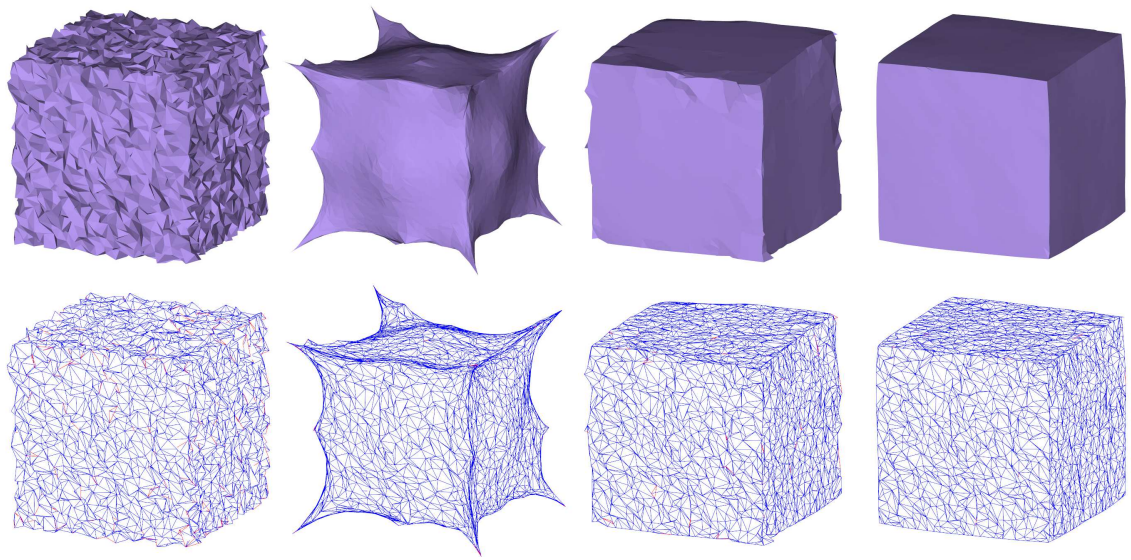


Figure 2.2: From left to right: noisy input surface with $\sigma = 0.3l_e$, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.

candidates for measuring this information on surfaces [36], an obvious choice is the discrete Laplacian operator [43], 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 [13]. Figure 2.2 shows a noisy input surface (left) and the effect of using the discrete Laplacian operator in this L_0 minimization framework (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. Figure 2.3 shows such an example, in which

we modify the planar mesh on the left by lifting up a pair of diagonal vertices and pulling down the other pair by same amount of offset. According to the traditional vertex-based Laplacian operator, the Laplacian of the center vertex in the right mesh is zero in this case. However, the local region around the centering vertex is apparently not flat, so vertex-based discrete Laplacian is not appropriate for measuring sharpness. Mathematically, since the Laplacian of a vertex is calculated based on its one-ring neighborhood, there might be too many degrees of freedoms, we will give more explanations on this in the next section. When observing the sharpness or smoothness of a triangle mesh, we notice that the angle formed by two adjacent triangle faces intuitively measures the sharpness around the targeting edge. Hence, we would like to use the Laplacian of edges on the mesh to measure the sharpness. We will generalize the construction of the vertex-based cotan operator to an operator that acts directly on an edge.

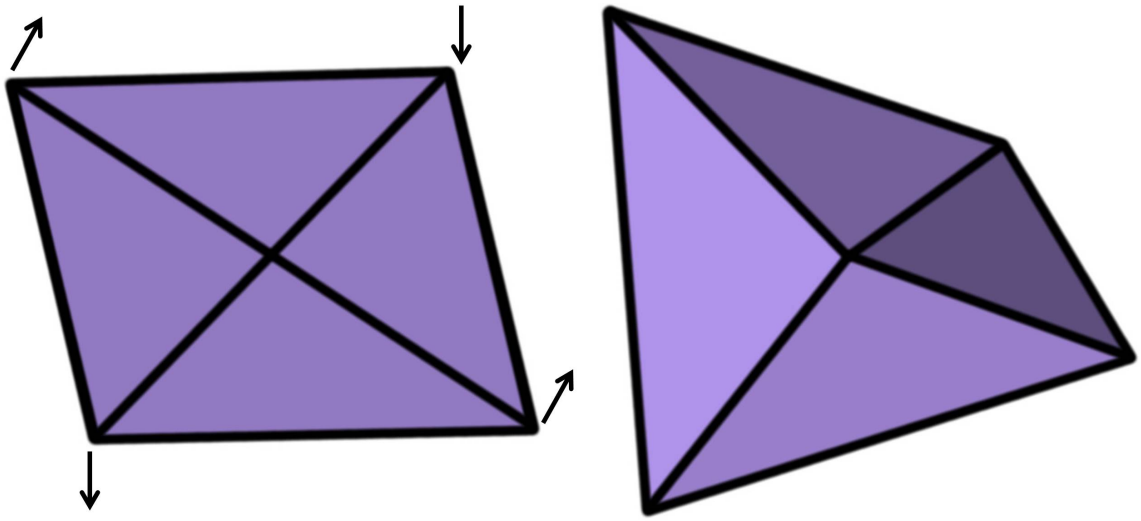


Figure 2.3: Left: a planar mesh formed by four triangles. Right: a modified mesh with zero curvature at the centering vertex.

2.3.1 *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,

$$\begin{aligned} \textstyle\sum_j w_j &= 0 \\ \textstyle\sum_j w_j p_j &= 0 \end{aligned} \tag{2.1}$$

when p_j are planar, which are similar to the properties of generalized barycentric coordinates [23]. 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 2.1 obvious. We will then extend this construction to build a differential edge operator.

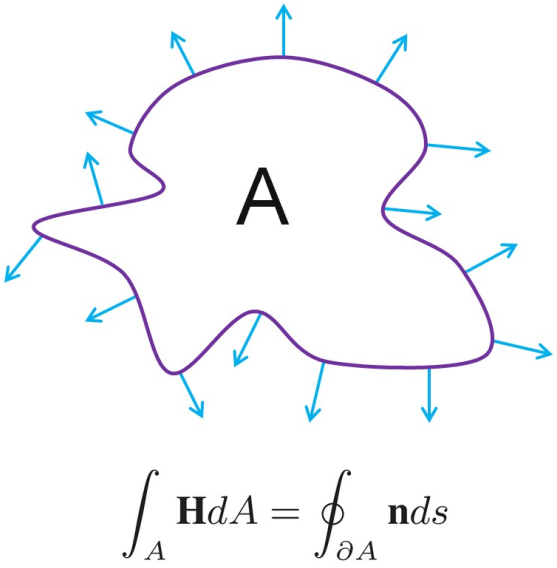


Figure 2.4: Based on divergence theorem, the curvature over a region can be calculated indirectly without directly integrating curvatures.

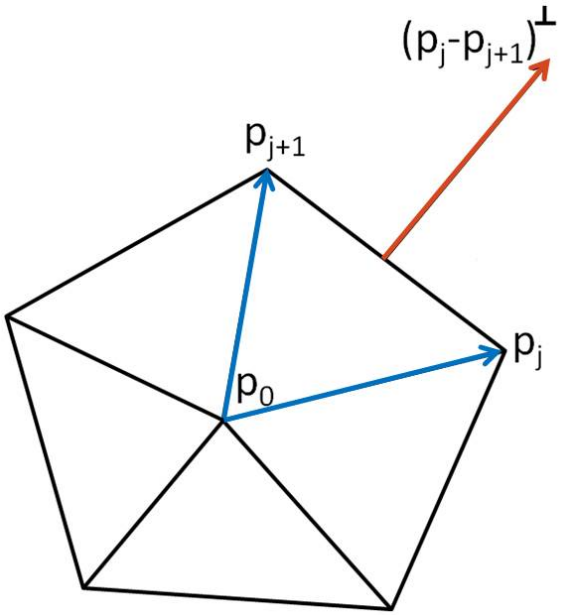


Figure 2.5: The normal vector (red) can be written as a weighted combination of two edge vectors (blue).

One method of building barycentric coordinates is through the use of the divergence theorem [46] illustrated in Figure 2.4, which states that the integral of an outward facing normal over a closed shape is zero. Figure 2.6 (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 (illustrated in figure 2.5) 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

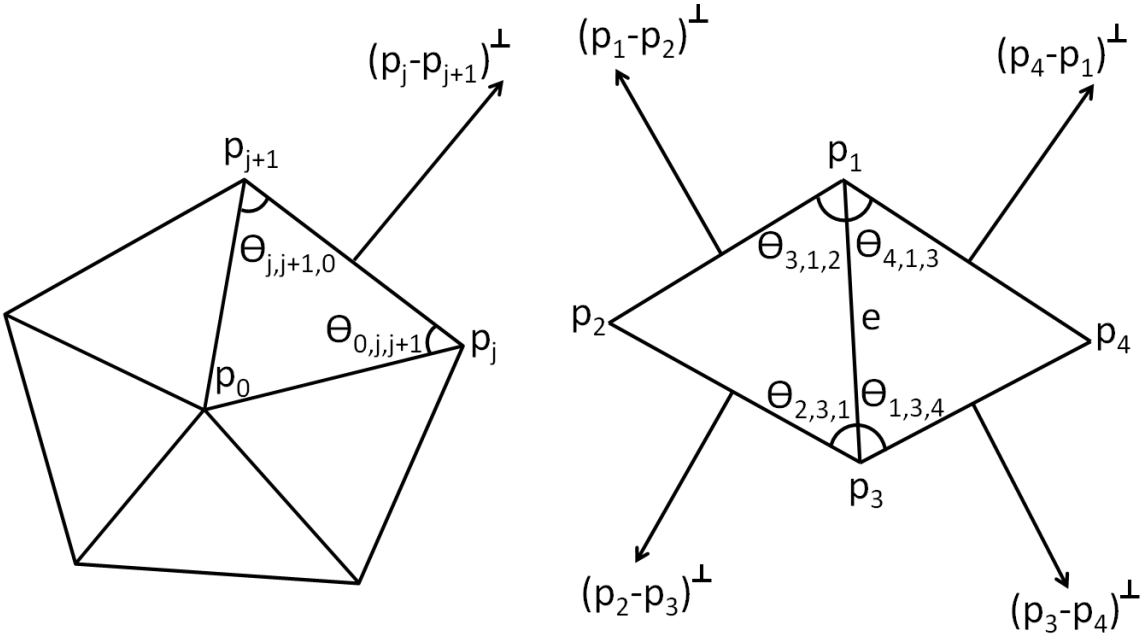


Figure 2.6: Our notation for the one-ring of a vertex and an edge.

from Pinkall et al. [43]. 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 2.1.

We can apply the same construction to build an edge operator. Figure 2.6 (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

$$D(e) = \begin{bmatrix} -\cot(\theta_{2,3,1}) - \cot(\theta_{1,3,4}) \\ \cot(\theta_{2,3,1}) + \cot(\theta_{3,1,2}) \\ -\cot(\theta_{3,1,2}) - \cot(\theta_{4,1,3}) \\ \cot(\theta_{1,3,4}) + \cot(\theta_{4,1,3}) \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}, \tag{2.2}$$

which also satisfies Equation 2.1. 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.

Following the discussion of the vertex-based discrete Laplacian in the last section, here we can review the vertex-based discrete Laplacian again. When there are more than or equal to four vertices in the one-ring neighborhood of a vertex, equation 2.1 will be underdetermined and will have multiple solutions. Some solutions would be undesirable even they can easily satisfy equation 2.1, because the corresponding one-ring configurations are not flat, see figure 2.3. In contrast, the edge-based Laplacian derived from equation 2.1 is always zero, each feasible solution in the family guarantees that the local surface configuration is planar.

Figure 2.2 (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

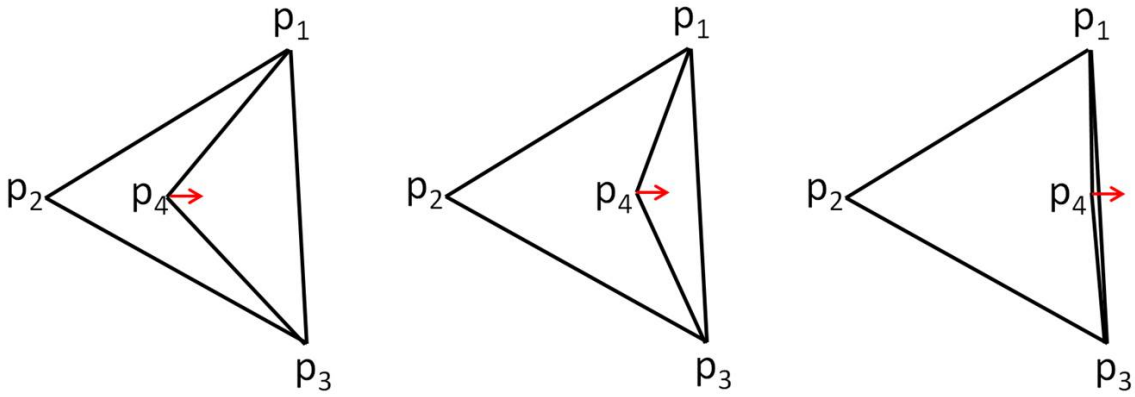


Figure 2.7: An example showing that the folded triangle may not be unfolded due to the potential numerical issue.

since the vertex must pass through a singularity in the weights. Figure 2.7 shows such an example. Kazhdan et al. [28] 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 2.1. If we assume that the p_j are in 2D, then Equation 2.1 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 2.2, 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

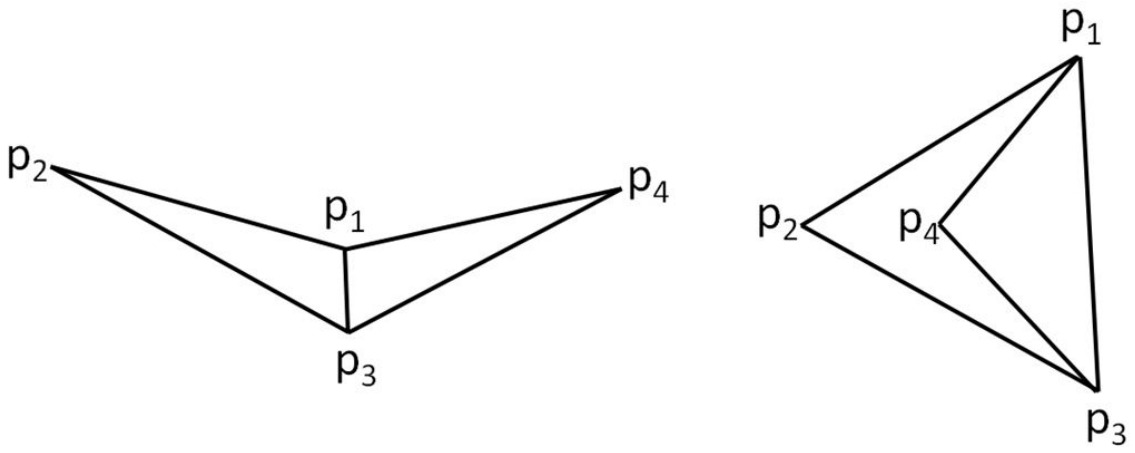


Figure 2.8: Two different per-edge planar configurations.

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 2.2 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. Figure 2.8 shows two different planar configurations, 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 (i.e.; $\Delta_{2,3,4} = \frac{1}{2}|p_2 - p_3||p_4 - p_3|\sin(\theta_{1,3,4} + \theta_{2,3,1})$

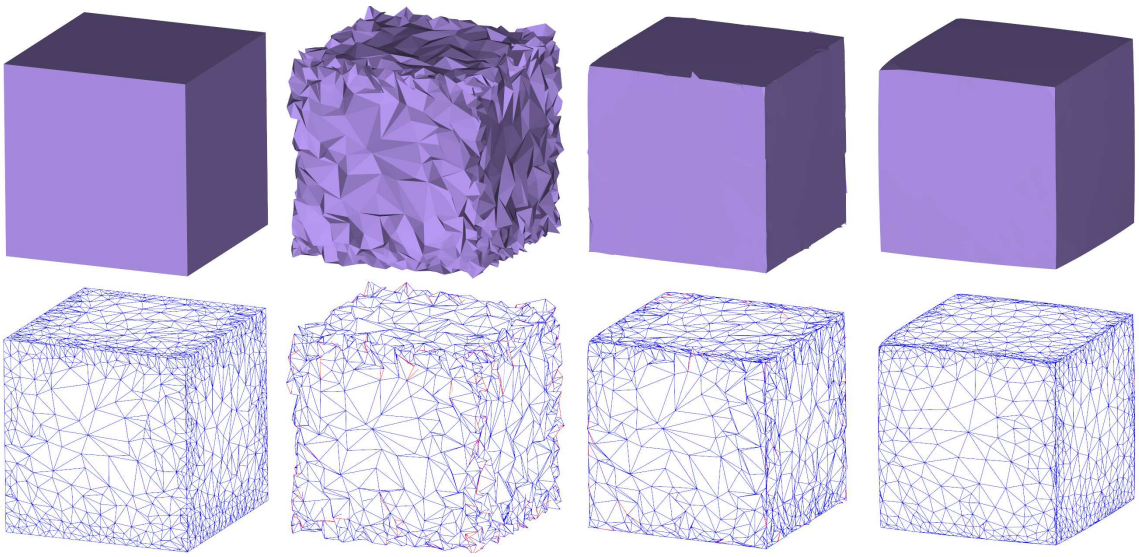


Figure 2.9: From left to right: the ground truth, the input mesh with large noise in random directions, our method without regularization, our method with regularization.

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

2.3.2 Regularization

For surfaces with relatively uniformly shaped triangles, the method in Section 2.3.1 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 2.9 (middle right). Our solution is to add a triangle shape regularizer for each edge given by the quadratic

$$(p_1 - p_2 + p_3 - p_4)^2. \tag{2.3}$$

Figure 2.9 (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.

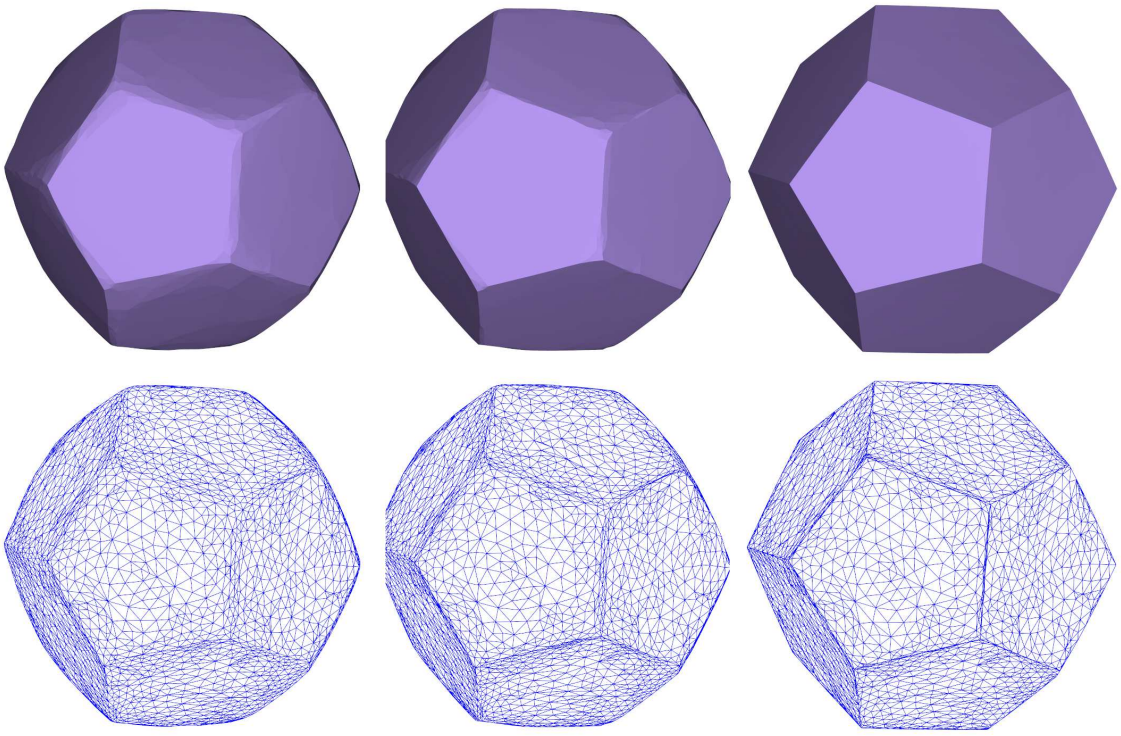


Figure 2.10: From left to right shows the results of our method using different speeds μ of 2.0, 1.414 (default) and 1.090. In this example, smaller values of μ produces near ideal results.

2.3.3 Optimization

Our optimization follows that of Section 2.2, and we minimize

$$\min_{p,\delta} |p - p^*|^2 + \alpha |R(p)|^2 + \beta |D(p) - \delta|^2 + \lambda |\delta|_0 \tag{2.4}$$

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 i^{th} entry is the edge regularizer from Equation 2.3 applied to the i^{th} edge. We again perform an alternating minimization where we hold p fixed to solve for δ,

$$\min_{\delta} \beta |D(p) - \delta|^2 + \lambda |\delta|_0 \tag{2.5}$$

and then hold δ fixed and solve for p in the same way as Section 2.2. When p is fixed, $D(p)$ is a constant, Equation 2.5 can be solved locally for each vertex.

$$\min_{p} |p - p^*|^2 + \alpha |R(p)|^2 + \beta |D(p) - \delta|^2, \tag{2.6}$$

which represents a sparse quadratic in p. Minimizing this energy function only requires solving a linear system of equations. This entire procedure is summarized in Algorithm 1.

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.

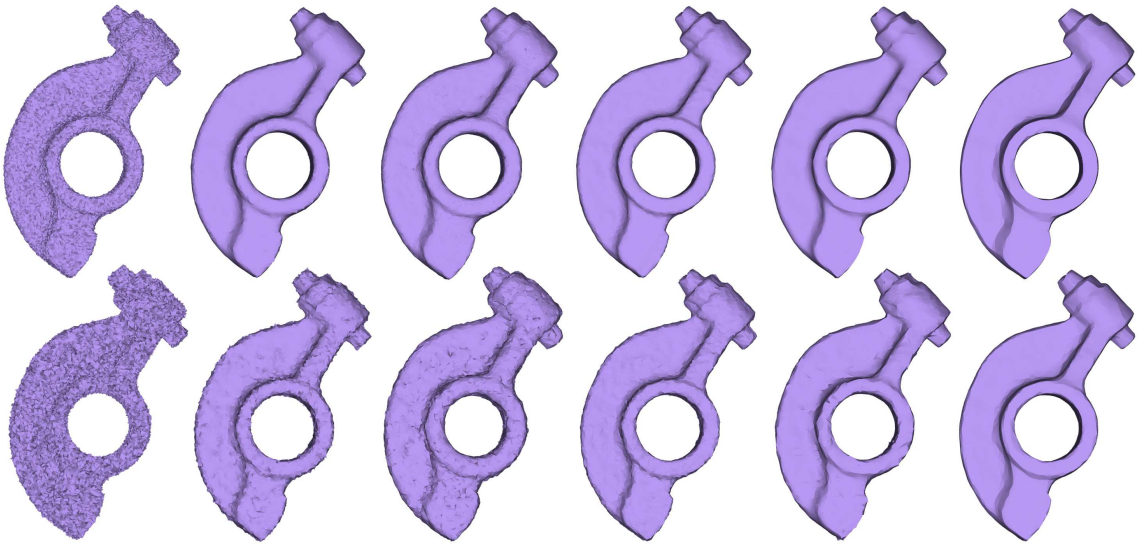


Figure 2.11: The performance of several methods with different amount of noise. Each row shows different levels of noise in random directions ($\sigma = 0.3l_e$ top, $\sigma = 0.6l_e$ bottom). From left to right: noisy input, bilateral filtering [22], prescribed mean curvature flow [24], mean filtering [64], bilateral normal filtering [67], our method. Model courtesy of the AIM Shape Repository [1].

2.4 Results

Equation 2.4 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 $\lambda = 0.02l_e^2\bar{\gamma}$ 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 2.2 and 2.9 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.

As in Section 2.2, λ provides a measure of the level of detail to preserve in the

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 Eq. 2.5.
  fix δ, solve for p in Eq. 2.6.
  $\beta \leftarrow \mu\beta$, $\alpha \leftarrow \frac{1}{2}\alpha$
until $\beta \geq 10^3$

input surface. Figure 2.12 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 2.10 shows the effect of changing the speed μ at which we increase β for the input from Figure 2.13. Xu et al. [61] 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 2.11, we have found that $\mu = \sqrt{2}$ tends to work best, which is the parameter we used in Figure 2.13.

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 2.13 illustrates a comparison between bilateral filtering [22], prescribed mean curvature flow [24],

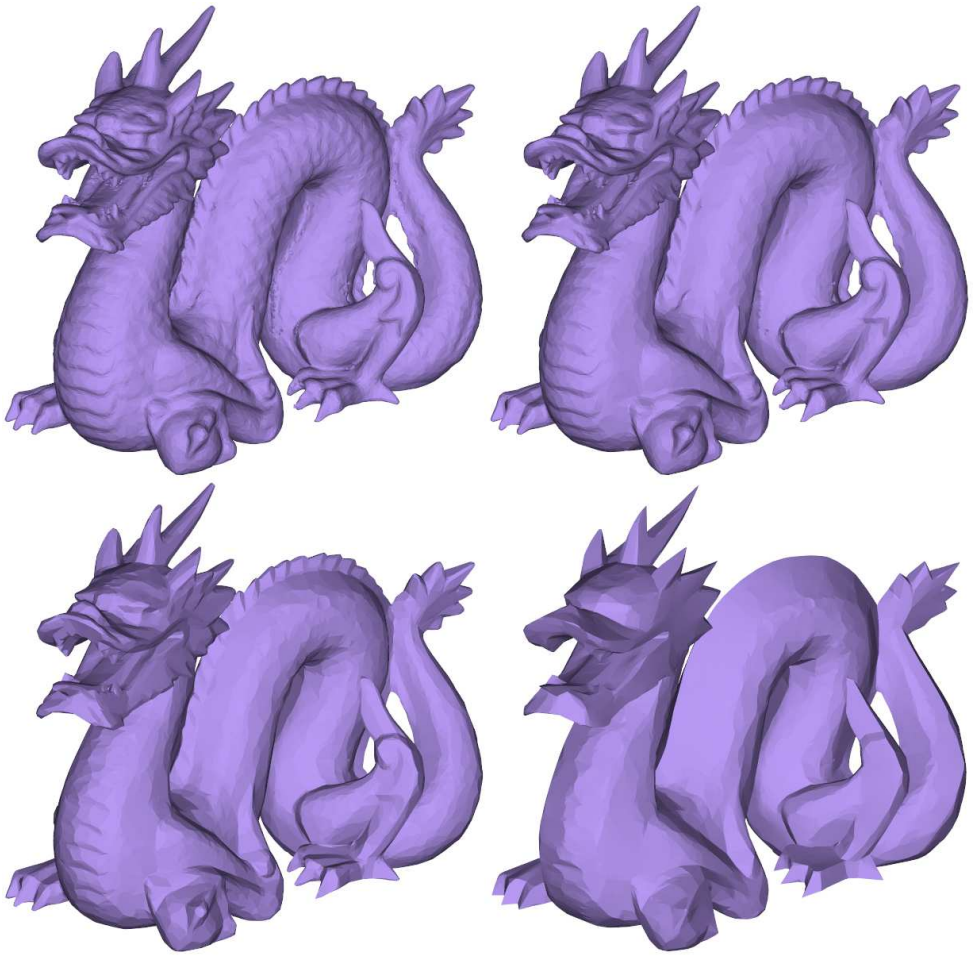


Figure 2.12: The effect of the L_0 weight. Top left: an input mesh without any noise, top right: $\lambda = \frac{1}{16}$ default, bottom left: $\lambda =$ default, bottom right: $\lambda = 16$ default. Model courtesy of the Stanford 3D Scanning Repository [6].

mean filtering [64], and bilateral normal filtering [67]. 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 2.10 illustrates, we can improve upon our result by tuning parameters. However, our default parameters provide a superior solution by themselves.

Figure 2.11 shows a comparison where we vary the level of noise for the different

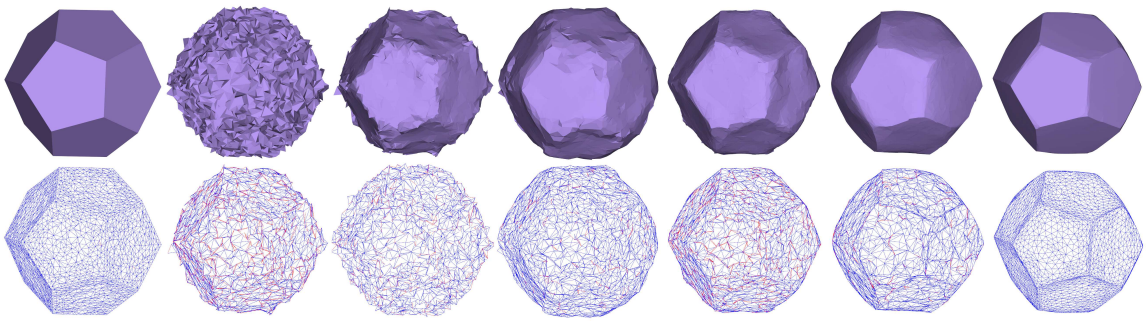


Figure 2.13: From left to right: initial surface, surface corrupted by Gaussian noise in random directions with standard deviation $\sigma = 0.4l_e$ (l_e is the mean edge length), bilateral filtering [22], prescribed mean curvature flow [24], mean filtering [64], bilateral normal filtering [67], our method. The wireframe shows folded triangles as red edges.

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 2.16, 2.14, and 2.15 also compare these methods on a variety of different surfaces.

We tested our algorithm on different kinds of models. Among these examples provided in this dissertation, three of them are real world models such as the models in figure 2.16, 2.17, 2.18, and others are with synthetic noise added by ourselves for verifying our method.

Our implementation uses TAUCS [48] 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 less than half second for Figures 2.13 and 2.9, which have about 3800 vertices each, to about 8 minutes for the model in Figure 2.19, which has over seven million vertices. Table 2.1 provides the running time of most examples provided in this dissertation. The provided running time measures the entire process, including loading mesh data, filtering process by optimization and saving mesh data. We have found that 90% of the execution time is taken by solving

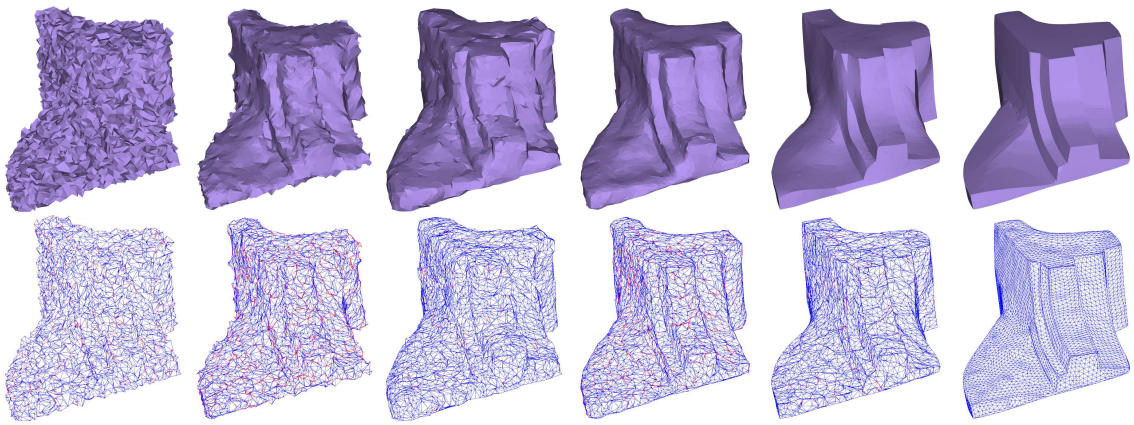


Figure 2.14: From left to right: the input mesh with large noise in random directions, bilateral filtering [22], prescribed mean curvature flow [24], mean filtering [64], bilateral normal filtering [67], our result. We show the wireframe of each surface below. Model courtesy of the AIM Shape Repository [1].

the system of equations. Since the equations change at each iteration, we cannot simply pre-factor the matrix, which leads to longer running times.

From table 2.1, we can see that the running time is basically proportional to the vertex number except for the *face* model. The reason is that the optimization time is not only governed by the vertex number, but is also controlled by the intermediate surface shape. As the optimization goes, the surface becomes smoother and the mean curvatures of more edges would be quite small. More likely, those low-curvature edges would be judged by the local optimization to be smooth edges. In this case, there will be more and more zero entries in the B vector if we simply write the linear system as $AX = B$. As a result, more and more vertices would start moving as the optimization goes and this will slow down the optimization. In Table 2.1, *deadseascrolls* and *anubis* are two such models, which are quite flat and with zero curvature for most edges. The increasing number of non-zero entries will slow down the performance of solving linear systems in TAUCS [48].

2.5 Applications of Differential Operators

In this section, we briefly introduce several extended applications of edge-based differential operator. Our metric can uniquely measures the planarity of the local shape, it can easily be incorporated into a feature-preserving minimizer based application, which could be but not limited to surface diffusion and 3D mesh planarization.

2.5.1 Surface Diffusion

We demonstrate the power of our differential operator on feature-preservation by fitting it into a surface diffusion method, specifically mean curvature flow. Mean curvature flow is one of the fundamental flows that has been used to evolve the surface geometry. Mathematically, it can be conducted by either minimizing the gradient of surface or minimizing surface area. Kazhdan et al. [28] recently proposed a modified mean curvature flow method to avoid the numerical instability by fixing the Laplacian matrix in the linear system for each iteration. We take our new area-based differential operator into their framework to calculate the a new stiffness matrix to replace the Laplacian matrix. As a result, our system evolves the surface with an undefined flow, we call it *feature preserving flow* in this dissertation. Our

Table 2.1: Performance of our algorithm

model name	vertex number	running time (s)
dedecahedron	3643	0.33
fandisk	6475	0.58
lion	17578	1.52
dragon	50000	9.32
teeth	116604	11.96
igea	134345	12.37
face	3511328	494.01
deadseascrolls	4359035	141.57
anubis	7192875	451.41

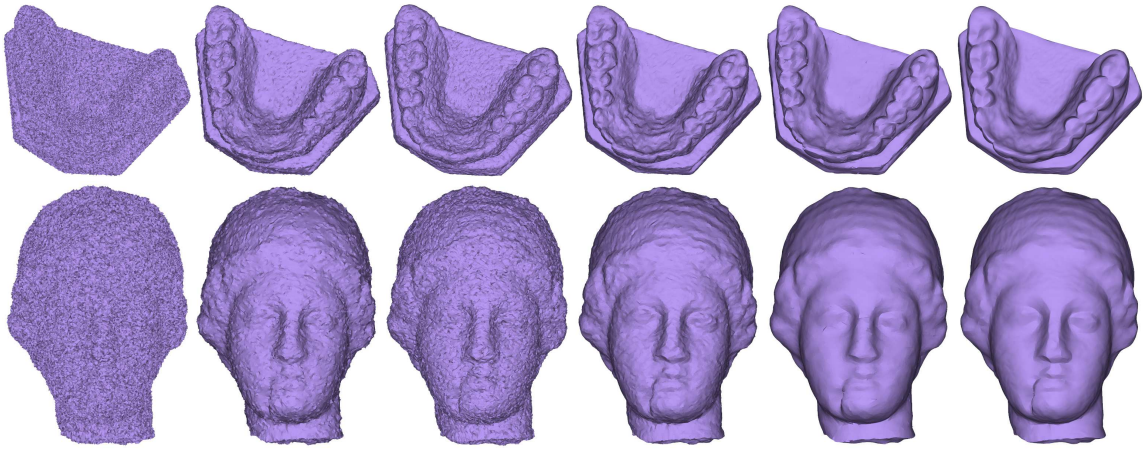


Figure 2.15: From left to right: the input mesh with large noise in random directions, bilateral filtering [22], prescribed mean curvature flow [24], mean filtering [64], bilateral normal filtering [67], our result. Model courtesy of the AIM Shape Repository [1].

feature preserving flow is able to smooth the surface and also preserve the prominent features, see figure 2.20.

2.5.2 Quadrilateral Mesh Planarization

Constraining 3D meshes to restricted classes is important in architectural and industrial design. Contemporary modeling methods for polygon meshes generally start with a free-form surface. Once the final shape is finalized, it needs to be converted into a mesh with planar faces for future use [44]. Quadrilateral meshes with planar faces are particularly suitable for the design of free-form glass structures [35]. Roi et al. [44] recently introduces an interactive optimization framework for generating planar quadrilateral meshes by enforcing a very intuitive nonlinear metric to measure planarity of a quad face. To measure the planarity of a quad face, we instead apply the area-based differential operator on both of its diagonals, where each diagonal splits the quad into two triangles. The sum of the planarity energy of each quad

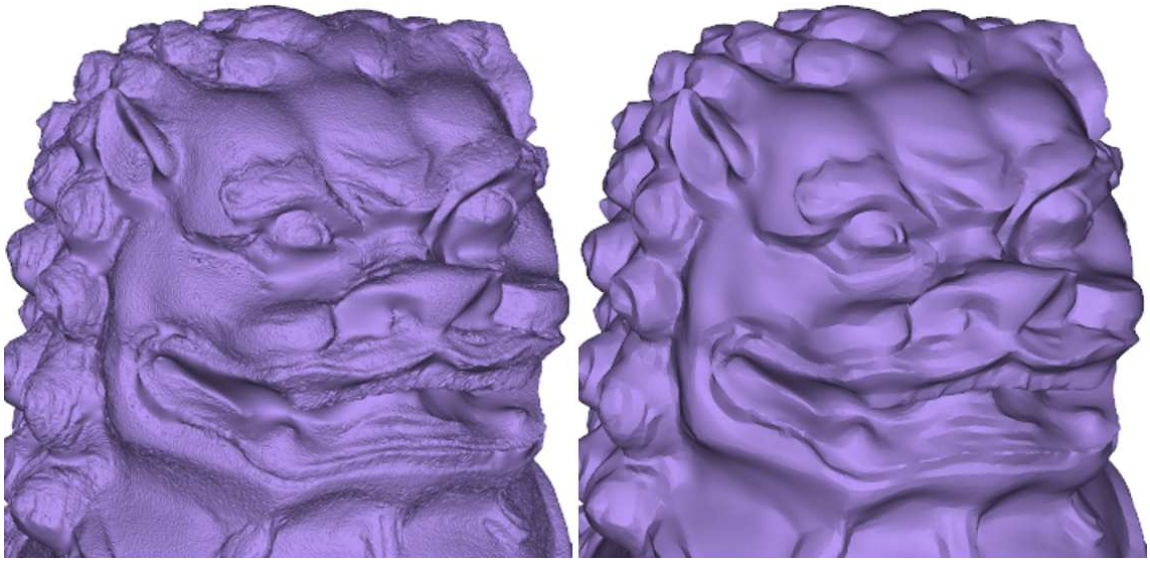


Figure 2.16: A model scanned with a laser range scanner. The noisy input surface (left) and our result (right). Model courtesy of the AIM Shape Repository [1].

forms the planarity and data fidelity terms of the objective function

$$\min_S \sum_{p \in S} |p - p^*|^2 + \sum_{q \in S} \alpha \cdot |D(e_q)|^2,$$

where p represents the vertices, q represents the quads, e_q is the diagonal of q and α controls the planarization strength. Figure 2.21 provides shows an example by using this method, the resulting mesh has almost *zero* curvatures on each face, this demonstrates that our metric can measure the quad planarity very well. Note that we neither intended to choose an optimal α nor tried to introduce more appropriate constraints to preserve the mesh quality. Also, our planarity metric is linear in terms of the vertex positions, so we only need to solve a quadratic minimization problem at each iteration. The resulting quads under our minimization framework, like some existing methods, are not completely planar though.

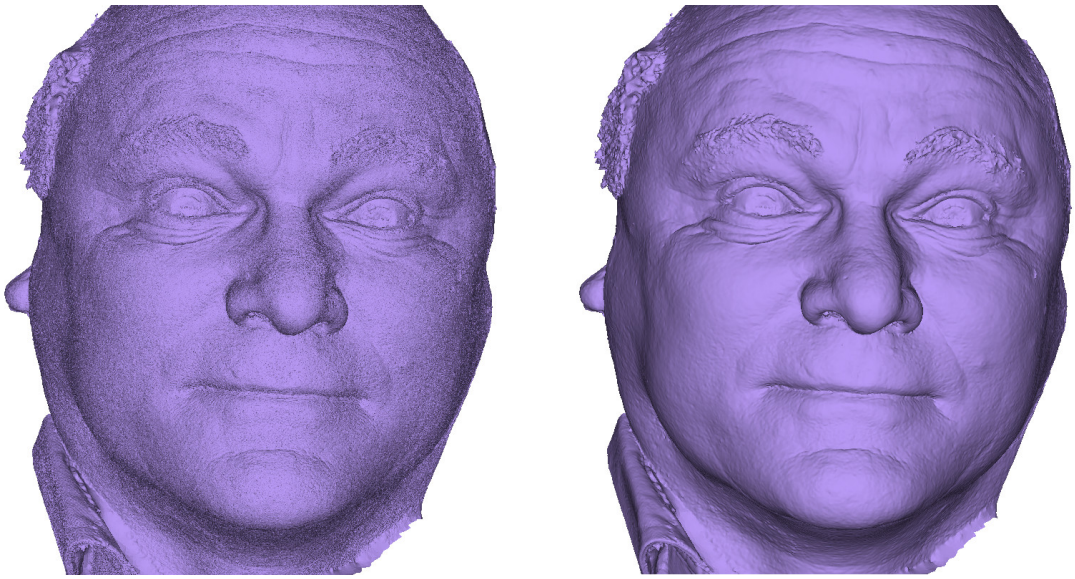


Figure 2.17: Left: a noisy face model with 3511328 vertices. Right: our smoothing result. Model courtesy of the Factum Foundation for Digital Technology in Conservation.

2.6 Discussion

2.6.1 Surface Quality Evaluation

While our algorithm is able to produce high quality results, it is quite difficult to evaluate it quantitatively. One popular quantitative evaluation method is to measure the L^2 vertex-based error between the denoised surface and the ground-truth model [67]. However, this evaluation metric cannot truly tell us how good the surface quality is. It only takes vertex positions into account and neglects any other differential properties such as normal and curvature, which can be better used to analyze the local shape. In addition, our goal is not to reconstruct the vertex positions of the original surface, but to reconstruct the entire shape of the unknown ground truth underneath the noisy model. Besides considering the shape similarity, we also desire the mesh to have high quality, which can be measured by some common

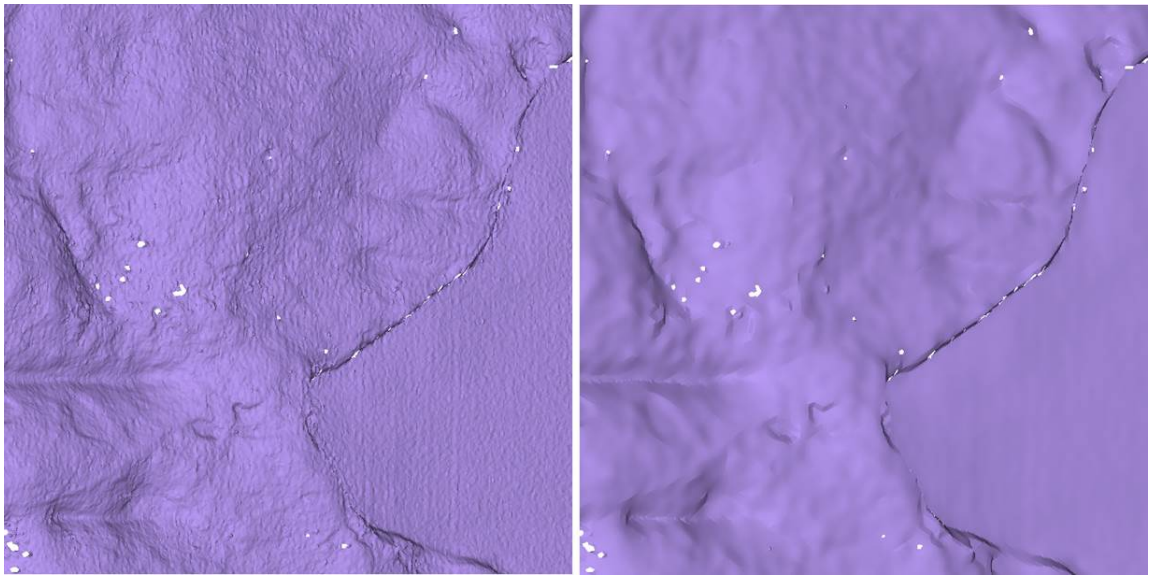


Figure 2.18: Left: a part of a noisy model with 4359035 vertices. Right: our smoothing result. Model courtesy of the Factum Foundation for Digital Technology in Conservation.

metrics such as smoothness. We are currently lacking of a well-rounded metric that is able to measure the shape similarity and quality by combining them together, thus we have not evaluated our method quantitatively in this dissertation. We leave this as an area for future work.

2.6.2 Limitations

Our method can fail to produce good results in some cases. Figure 2.23 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 as the noise level exceeds the feature size.

Our edge-based smoothness metric is specific to triangulated surfaces. Therefore, it cannot be applied to any other non-triangle meshes such as quadrilateral

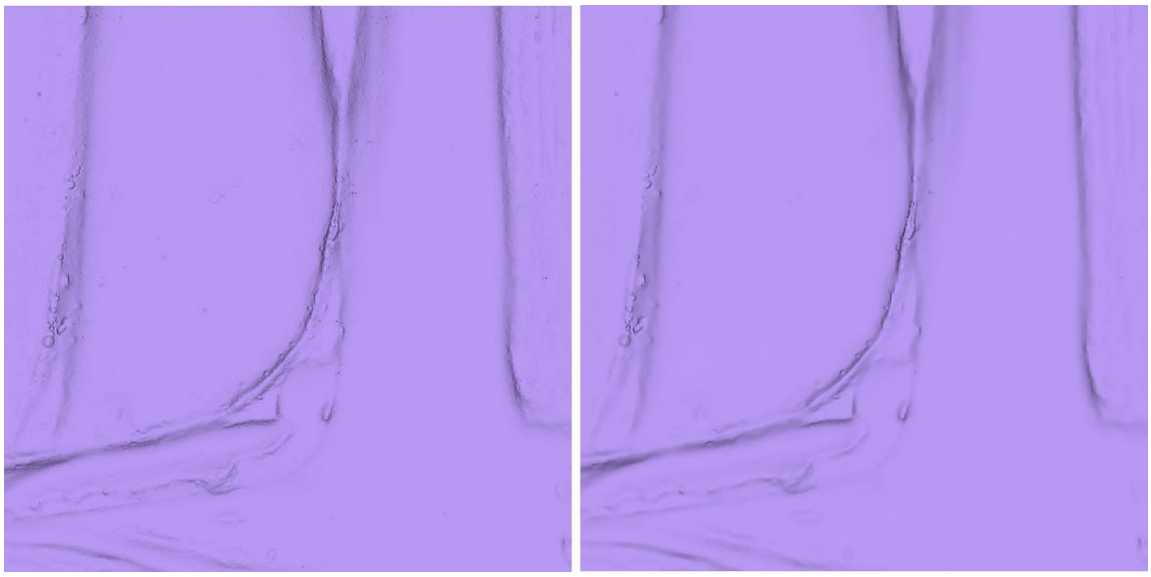


Figure 2.19: Left: a part of a noisy model with 7192875 vertices. Right: our smoothing result. Model courtesy of the Factum Foundation for Digital Technology in Conservation.

meshes. Even for triangle meshes, our algorithm may not be appropriate for some non-manifold models. For example, there is no clear way for edge-based Laplacian operator to measure the sharpness of the non-manifold edge in Figure 2.22. However, it is not clear how to measure smoothness at all along this edge since the surface is not manifold.

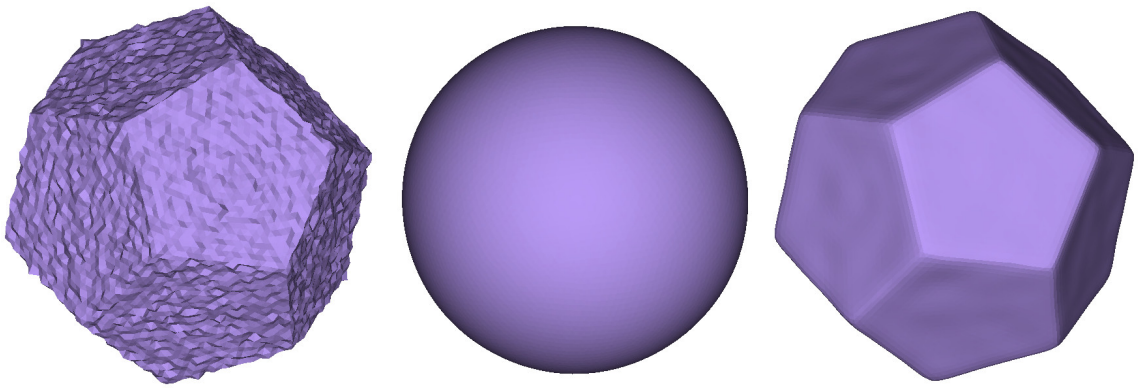


Figure 2.20: From left to right: input mesh, the result under mean curvature flow, the result under our feature preserving flow.

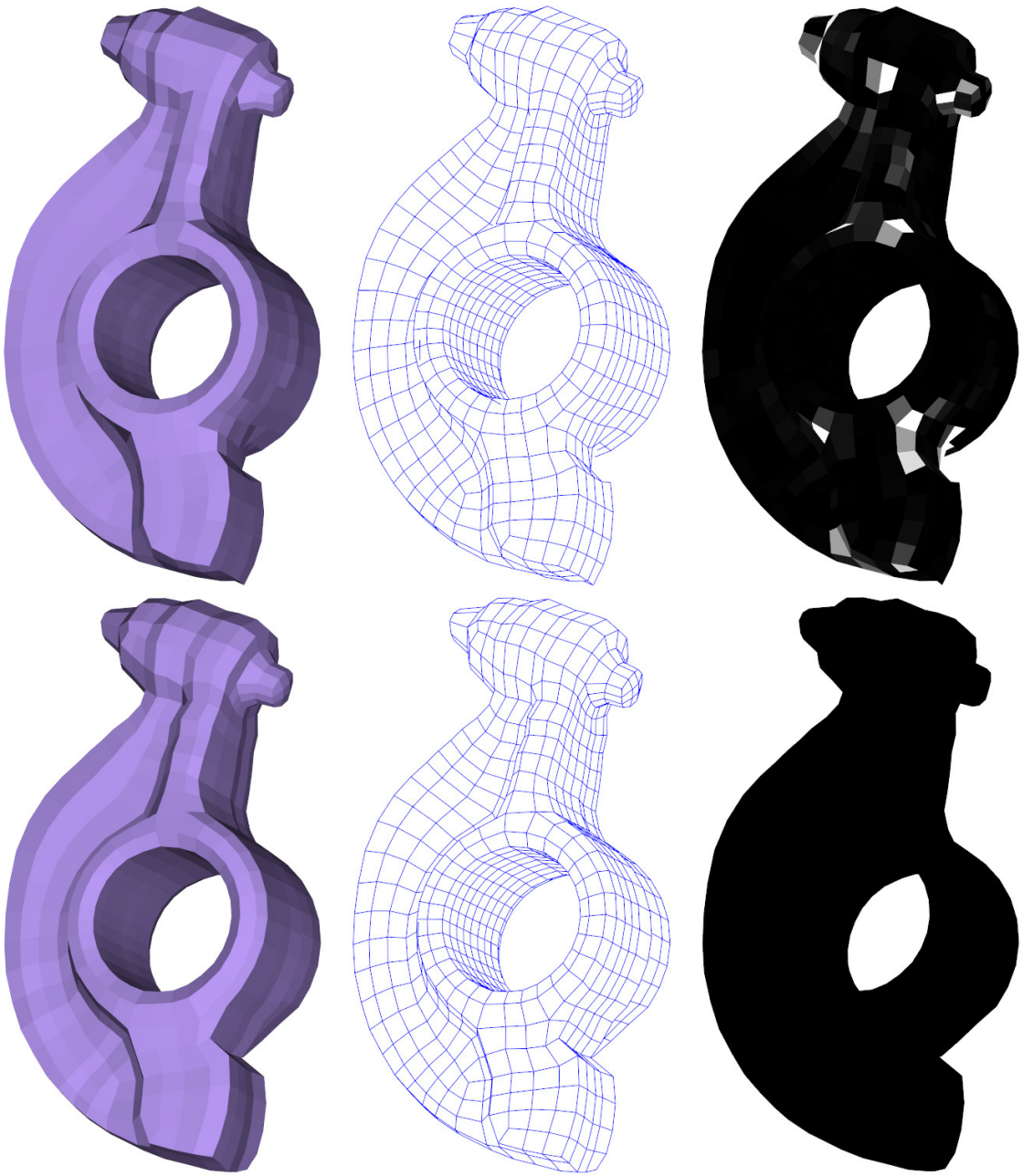


Figure 2.21: Top: input mesh. Bottom: the resulting planar quad mesh with our planarity metric. The coloring model on the right most denotes the planarity error: the darker, the less error.

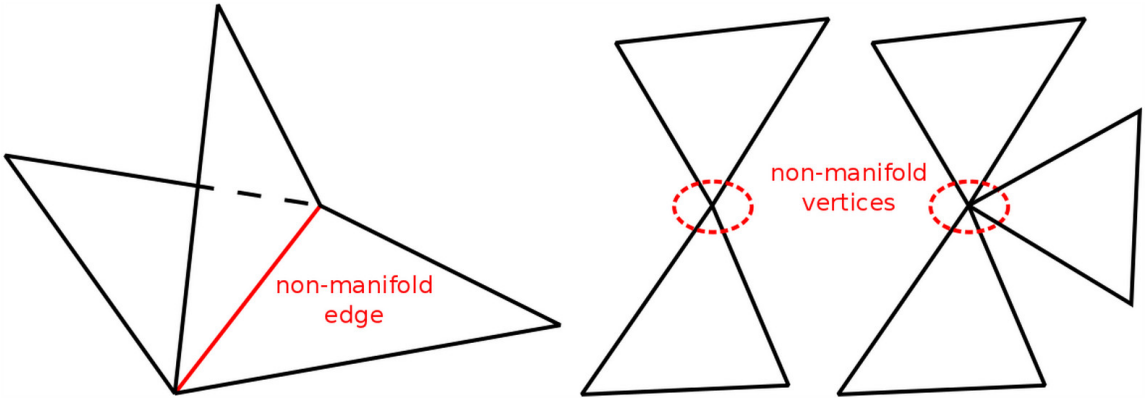


Figure 2.22: Examples of non-manifold vertices and edges.

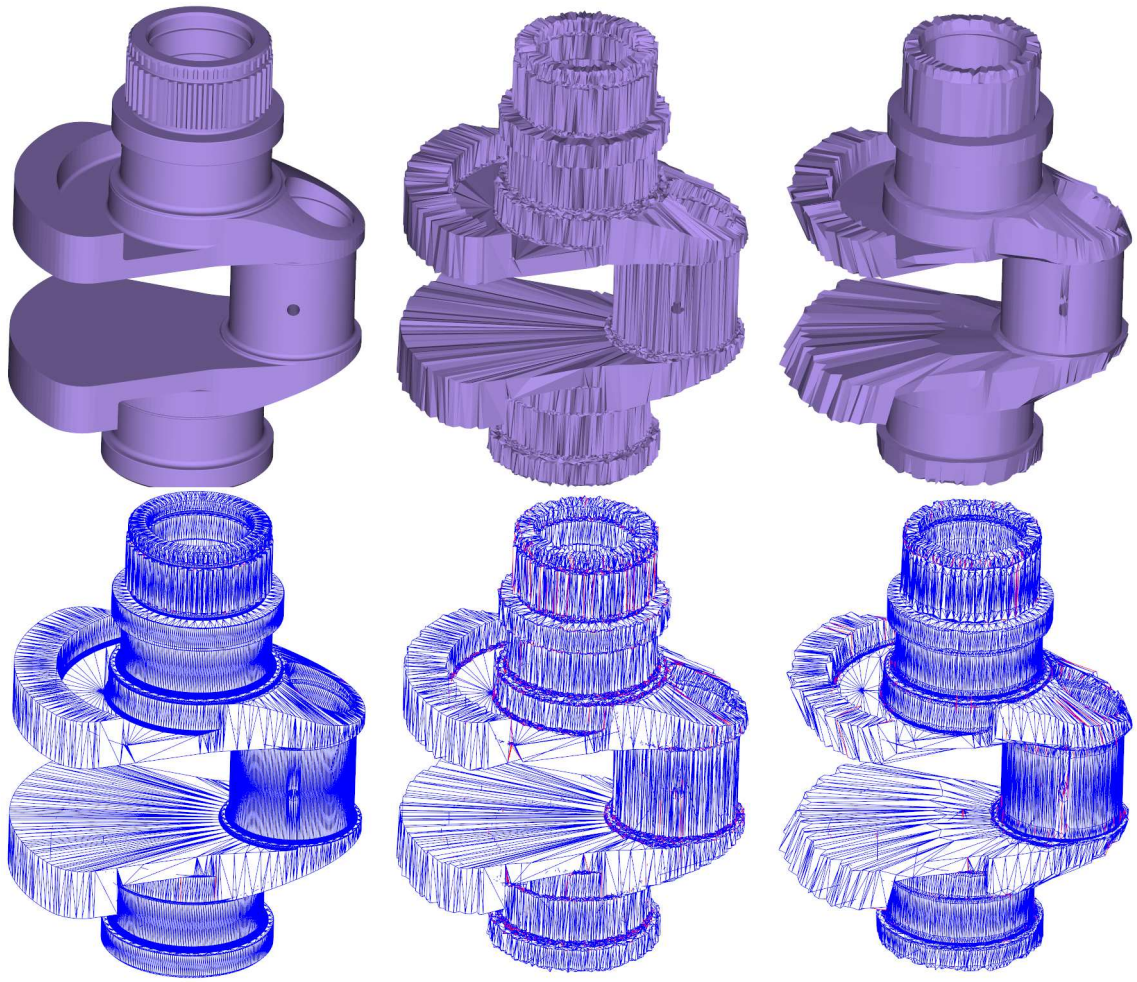


Figure 2.23: A failure case. From left to right: the ground truth with an extreme triangulation, the noisy input, our result.

3. IMAGE DETEXTURING

Textures exist in natural scenes as well as man-made art pieces captured in an image. Textures appears in various forms that include but are not limited to sand, wood grain, fur, canvas, glass and leather. Examples like drawings on brick walls, patterns on wool knits, and image mosaics using different materials are all textured materials. Our work focuses on images of these objects. While some images may just have pure textures in them such as Figure 3.1, others like examples in Figure 3.2 consist of structure and texture layers.

Decomposing an image into meaningful components is an important and challenging inverse problem in image processing [7]. Perhaps the most studied problem of this form is image denoising. In this problem, the target image is assumed to have been corrupted by noise, and the goal is to separate the structure of the image from the noise. The problem we address in this dissertation is a related but different problem. The goal is to decompose a textured image into structural parts and texture parts. While human perception is fully capable of understanding those textured images, the definition of texture is even more vague than noise. For example, a "structure" at a fine scale could be considered as "texture" in a larger scale.

In this work, we present a simple and effective approach for texture removal by modifying the traditional *bilateral filter* in a novel way. Since the range kernel in the bilateral filter is designed to preserve sharp edges, an ideal range image needs to solely have the latent structures embodied. However, the original image may contain too many strong textures and be incapable of capturing the structural information precisely. Hence, we propose to obtain an approximate range image for better estimating the range kernel that performs like an edge-stopping function. The modified

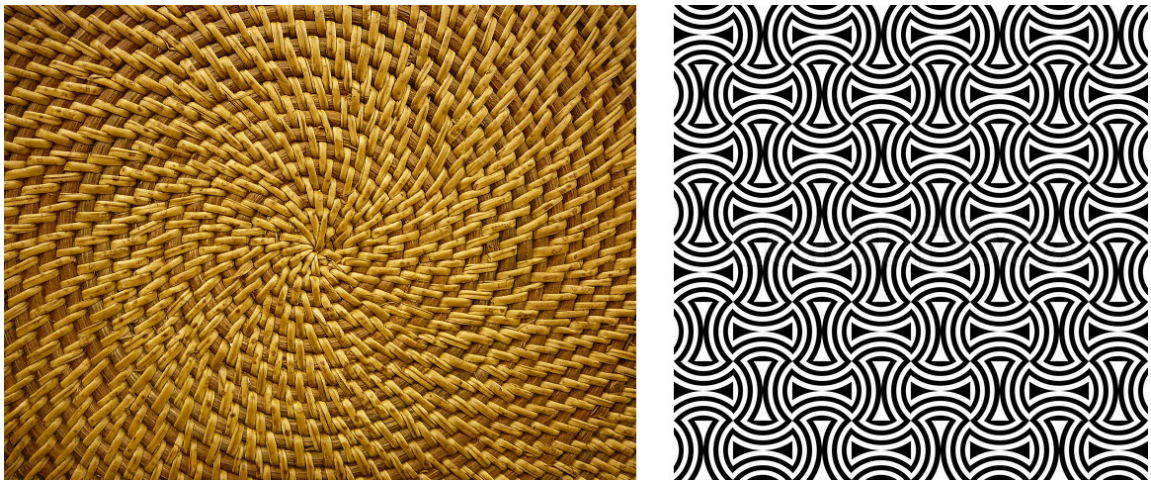


Figure 3.1: Examples of pure-texture images [38].

filter not only makes the texture and major structure highly distinguishable, but also preserves both sharp and shallow edges. Note that our filter is also a *joint (cross) bilateral filter* and shares the same motivation with filtering techniques introduced by [42, 18], in which the flash image has been used to compute range kernel instead of the input no-flash image.

3.1 Background

A popular image structure-texture decomposition model, originally proposed in [45], builds up an optimization framework by regularizing total variation. Based on this model, several image decomposition approaches have been developed [37, 58, 66, 7]. By weighting textures and structures indiscriminatingly during optimization, this framework cannot distinguish textures from latent structures very well. The resulting images are generally over-blurred. Most recently, a modified regularizer related to total variation has been proposed by [62] to achieve high quality results. However, this method still tends to eliminate weak yet important structural information.

Many feature preserving image filtering methods can also be used for removing

Figure 3.2: Examples of *structure+texture* images [9, 5].

textures, even though these algorithms are not initially designed to solve the detexturing problem. Local filters such as the bilateral filter [56, 16, 42, 18, 11] and histogram based filters [27] are more suitable for smoothing out small-scale details while preserving sharp edges. Similar to the structure-texture decomposition model, global methods generally build up an energy minimization problem by replacing the regularization term with other constraints, such as weighted least squares [21] and L_0 gradient minimization [61]. Subr et al. [50] develop a model for capturing *oscillations*, which can be used to distinguish high-contrast details like textures, but this method tends to suppress details.

Our image detexturing process is highly related to two image processing techniques: *bilateral filter* and *total variation (TV) model*.

3.1.1 Bilateral Filter

Bilateral filter, first introduced by [56], is a non-linear and local edge-preserving and noise-removing filter for images. Sylvain et al. [41] provide a detailed description of its theory and applications.

Following the concept of Gaussian filtering, the bilateral filter is also defined as

a weighted average of the intensity of neighboring pixels. The difference is that the weights depend not only on the spatial distance of pixels, but also on the range differences (e.g. intensity differences). The bilateral filter is defined as

$$I_p^{bf} = \frac{1}{W_p} \sum_q G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(|I_p - I_q|) I_q, \tag{3.1}$$

where the normalization factor W_p ensure the weights sum to 1.0:

$$W_p = \sum_q G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(|I_p - I_q|). \tag{3.2}$$

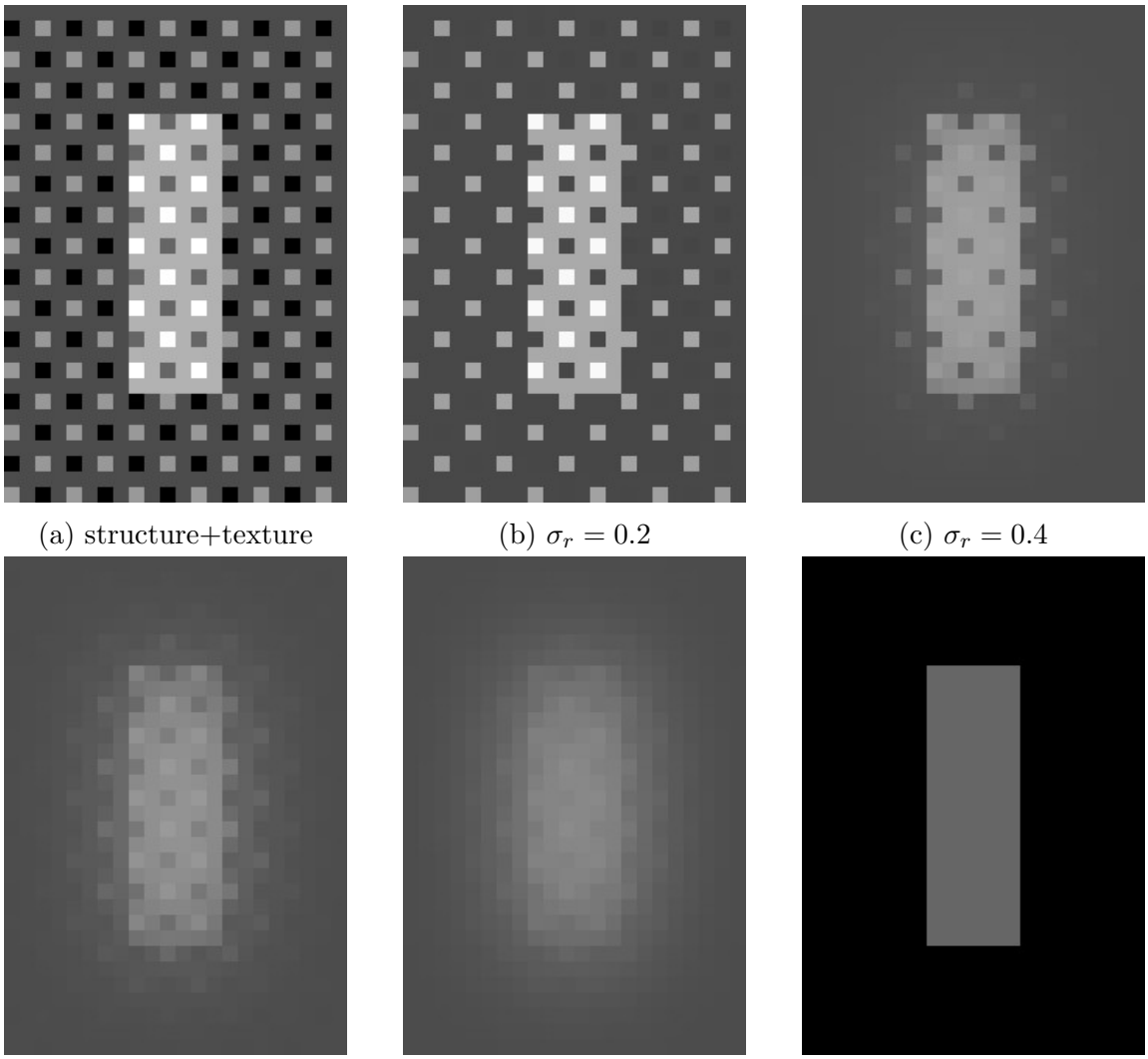
The Gaussian kernel function is defined as

$$G_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}}. \tag{3.3}$$

The entire filtering is controlled by two parameters σ_s and σ_r. As the range parameter σ_r increases, the filter gradually approximate a Gaussian filter and fewer edge features are expected to be preserved. The spatial parameter σ_s is depending on the scale of the textures. While the users may choose these parameters arbitrarily, it is also possible to adaptively select parameters [33] by estimating the local noise level. Figure 3.3 shows some results of the bilateral filter with different parameters. This example demonstrates that it is difficult if not impossible to find an appropriate parameter for the bilateral filter to clearly separate the texture and structure.

3.1.2 Total Variation Model

Aujol et al. [7] have studied four total variation models and advocate the $TV - L_2$ model in a general case, when no prior knowledge of the texture is known. This TV model introduces a quadratic data fidelity term to enforce the similarity between the



(a) structure+texture (b) $\sigma_r = 0.2$ (c) $\sigma_r = 0.4$

(d) $\sigma_r = 0.6$ (e) $\sigma_r = 0.8$ (f) ground truth

Figure 3.3: a an input image, b−e results of BF with different range parameters, f ground truth.

original textured image and the resulting structure image. It solves the following minimization problem:

$$\min_I \sum_p |I_p - I_p^*|^2 + \lambda |\nabla I_p|, \tag{3.4}$$

where $\sum_p |\nabla I_p|$ is the total variation. Here I^* is the original image to decompose, I is the unknown structural image. The regularizer can be written as

$$\sum_p |\nabla I_p| = \sum_p |(\partial_x I)_p| + |(\partial_y I)_p|,$$

where $(\partial_x I)_p$ and $(\partial_y I)_p$ are the gradients at p in two directions.

Xu et al. [62] modified this model to achieve much better performance on distinguishing strong edges from textures by introducing *relative total variation* as the regularizer. The authors minimize

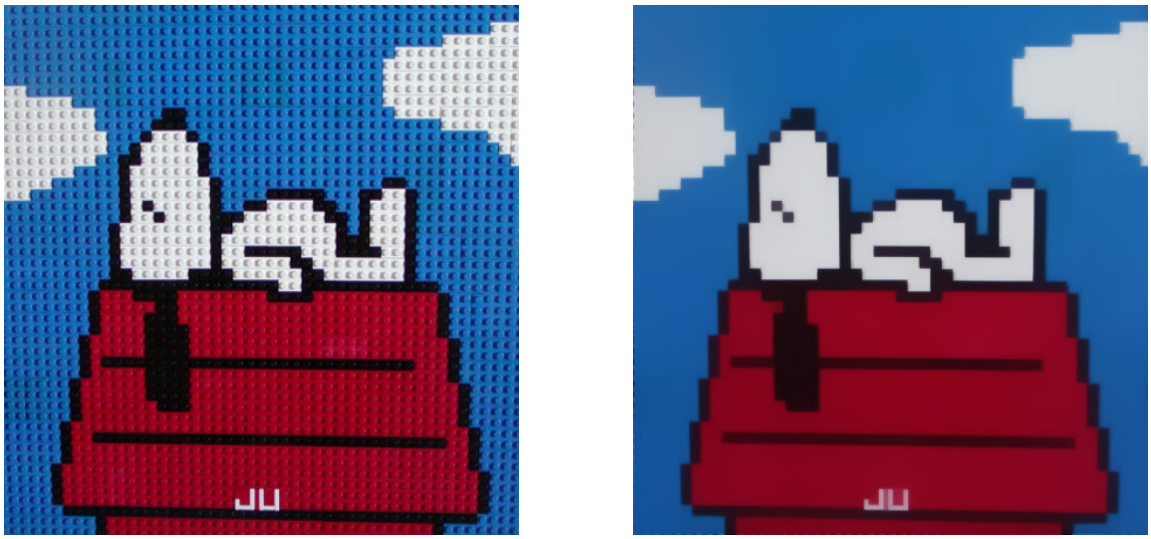
$$\min_I \sum_p (I_p - I_p^*)^2 + \lambda \cdot (\frac{D_x(p)}{L_x(p) + \epsilon} + \frac{D_y(p)}{L_y(p) + \epsilon}), \tag{3.5}$$

where $\frac{D_x(p)}{L_x(p)+\epsilon} + \frac{D_y(p)}{L_y(p)+\epsilon}$ is the new regularizer. $D_x(p)$ and $D_y(p)$ measure the *windowed total variation* at pixel p in two directions, which are written as

$$D_x(p) = \textstyle\sum_{q \in N(p)} G_\sigma(\|p - q\|) |(\partial_x I)_q|,$$
$$D_y(p) = \textstyle\sum_{q \in N(p)} G_\sigma(\|p - q\|) |(\partial_y I)_q|,$$

where the Gaussian kernel $G_\sigma(\cdot)$ is defined in Equation 3.3 and $N(p)$ is the neighborhood of pixel p.

In Equation 3.5, the functions $L_x(p)$ and $L_y(p)$ are the *windowed inherent varia-*



(a) (b)

Figure 3.4: (a) A textured image [63], (b) a detextured image by the RTV model.

tion and used to better distinguish sharp structures from the textured image.

$$L_x(p) = |\sum_{q \in N(p)} G_\sigma(\|p - q\|)(\partial_x I)_q|,$$
$$L_y(p) = |\sum_{q \in N(p)} G_\sigma(\|p - q\|)(\partial_y I)_q|.$$

While the windowed total variation has the ability to identify textures just like the total variation introduced in Equation 3.4, the windowed inherent variation helps to further prohibit textures. The key point is that the windowed inherent variation of a texture region is much smaller than that of the region with structures as well. Xu et al. [62] has verified the superiority of this new regularizer on removing textures by various tests. Figure 3.4 shows a result of this method.

3.2 Image Detexturing via Filtering

In this section, we first introduce how the bilateral filter can be modified to a joint bilateral filter (JBF) to handle the image detexturing problem. Then we show

how to filter images in practice.

3.2.1 Modified Bilateral Filter

We focus on processing general images with unknown texture patterns, so we assume that no prior knowledge about the textures can be used. Given an input image I, our goal is to decompose it into the structure I_s and the texture I_t such that $I = I_s + I_t$. While directly applying bilateral filter to I is difficult to remove the textures, we use a joint bilateral filter, in which instead of I, a different image I' has been used to compute the range kernel. This filter is defined as

$$I_p^{jbf} = \frac{1}{W_p} \sum_q G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I'_p - I'_q|) I_q. \tag{3.6}$$

When $I' = I$, Equation 3.6 reduces to the bilateral filter, which cannot satisfyingly remove textures. Hence, we need to find the optimal image I', so that I^{jbf} approaches I_s. Note that under a fixed range image I', the joint bilateral filter is linear in I_p, so Equation 3.6 can be rewritten as

$$I^{jbf} = I_s^{jbf} + I_t^{jbf}. \tag{3.7}$$

Directly seeking the optimal I' is quite complicated. Instead, we attempt to answer if there is a suboptimal I' that is solely depending on the structure image I_s and independent of I_t. If there exists such a suboptimal range image I'_{opt} for our filter to produce a resulting image I^{jbf} that can approach the ground truth I_s, the second part I_t^{jbf} in Equation 3.7 should approach 0. However, filtering a texture image without any structural information would ideally result in an image with constant color. Note that this constant is not necessarily 0 in practice. Therefore, we claim that if and only if the mean of the texture image is 0, the resulting image then

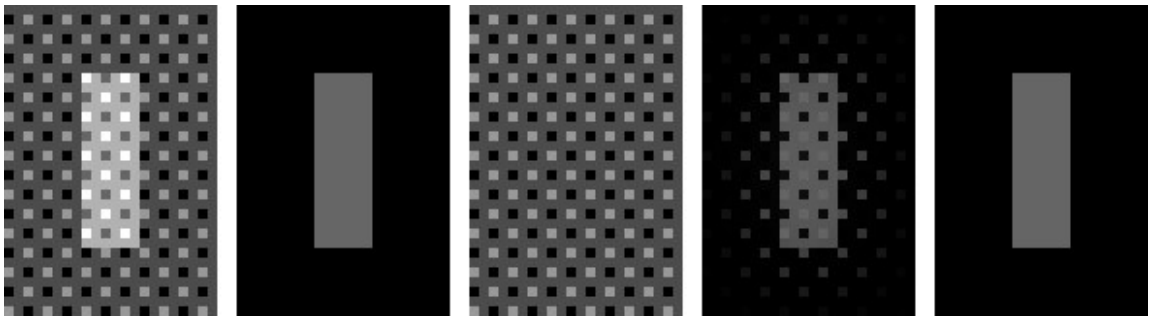


Figure 3.5: From left to right: structure+texture image I, structure image I_s, texture image I_t, results of filtering I_s using I and I_s as the range images. $\sigma_s = 5.0, \sigma_r = 0.25$.

approaches the structural image I_s. Otherwise, it approaches the latent structural image plus an image with constant color $I_s + C$, in which C is an image with constant color. Because it is impossible to infer C, our filter will always assume that $C = \mathbf{0}$, which means I_t is a zero-mean texture image. This is a very natural choice given no prior information about the textures. The suboptimal I'_{opt} is then constrained by

$$\begin{aligned} I_s^{jbf} &= I_s, \\ I_t^{jbf} &= \mathbf{0}. \end{aligned} \tag{3.8}$$

The first constraint says that filtering the structure image should result in the same structure image, while the second one says that filtering the pure-texture image should just simply remove it. Based on the constraints in Equation 3.8, we claim that the unknown structure image $I' = I_s$ is a good choice for estimating the range kernel.

Structure preserving. When we apply the joint bilateral filter to the pure structure image I_s, with I_s as the range image, this filter is identical with the bilateral filter. The performance of our filter relies on the performance of bilateral filter on structural images. Note that bilateral filter has been proved to be good at preserving

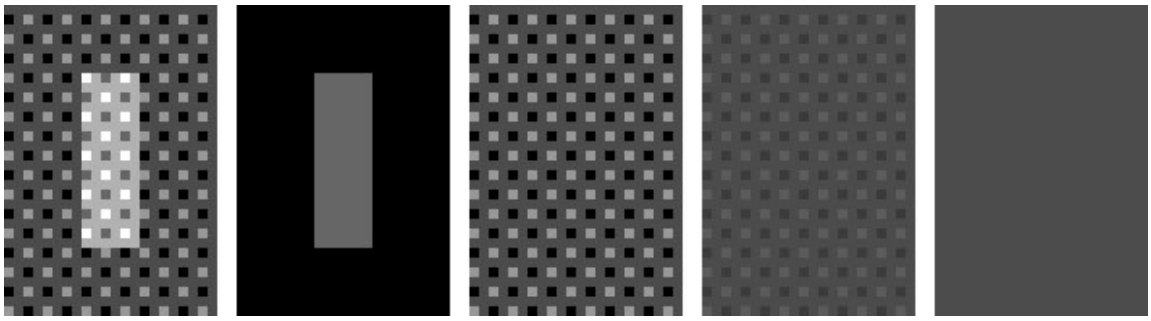


Figure 3.6: From left to right: structure+texture image I, structure image I_s, texture image I_t, results of filtering I_t using I and I_s as the range images. Both filters are with the same parameters: $\sigma_s = 5.0, \sigma_r = 0.25$.

features when the input image is noise free. Figure 3.5 shows the performance of bilateral filter on an artificial structural image using either I or I_s as the range image. We can see that the texture in the range image interrupts the structure information, and it prevents the filter from estimating a precise range kernel to measure the range difference. The first constraint in Equation 3.8 is approximately satisfied in practice when using structure image to estimate the range kernel.

Texture removal. We use different range images to filter the texture image I_t to evaluate their performance, Figure 3.6 shows that using the input image I cannot remove the textures very well and even tends to preserve the textures. We propose to use I_s as the range image, the advantage is two-fold. First, I_s contains the essential structural information in order to preserve edge features. Second, it helps to distinguish textures clearly and prevent the filter from incorrectly parsing the textures as structural content, so that textures can be removed completely. Note that the mean of the texture image in Figure 3.6 is not zero but some positive constant. Figure 3.7 compares three different filters: Gaussian filter, BF and the proposed JBF. We also show the operator masks of a pixel for each method to demonstrate that our filter outperforms the other two filters. Compared to the Gaussian filter,

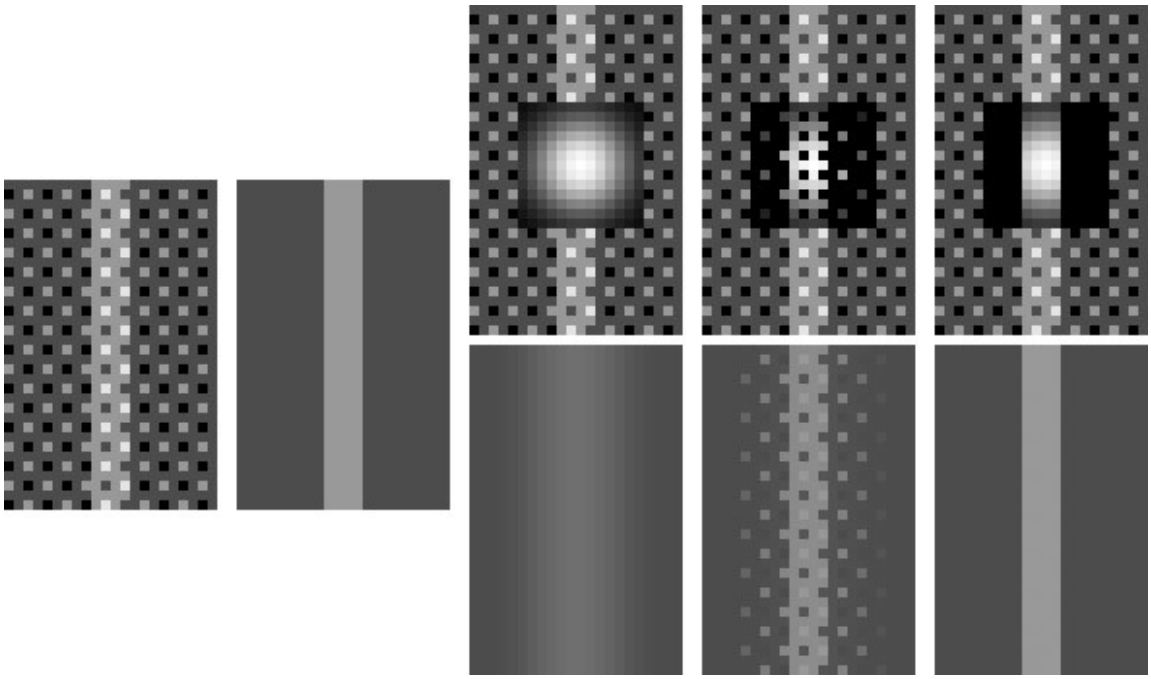


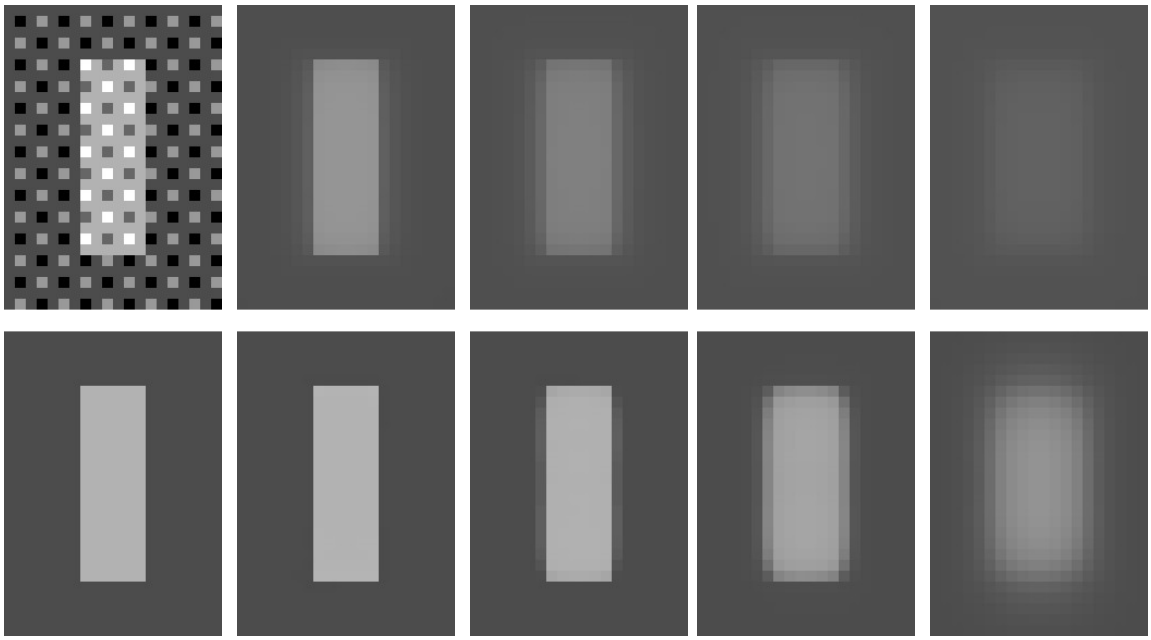
Figure 3.7: From left to right: structure+texture image $I_s + I_t$, structure image (ground truth) I_s, operator masks of a pixel (top) and results (bottom) for Gaussian filter, bilateral filter and our filter. $\sigma_s = 5.0, \sigma_r = 0.25$.

our filter precisely captures the edges and preserves them better. Compared to the BF, our filter overcomes the obstruction from the textures by using the structure image itself to compute the range kernel.

3.2.2 Approximate Range Image

Section 3.2.1 has reasoned the effectiveness of choosing the structure image I_s for computing the range kernel. However, the structure image is unknown and is the ideal result that we are actually looking for. We propose to acquire an approximate range image first, then fit it into our filter.

Figure 3.8 shows the effect of the approximate range image. From 3.8 (b) to 3.8 (e), As the structure information of range images is gradually reduced, the result becomes blurrier. Figure 3.8 (b) demonstrates that it is not necessary to have



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

Figure 3.8: (a) An input image and its texture image [63], (b)-(e) each includes a range image (top) and a resulting image (bottom).

$I' = I_s$ to achieve good results. It is also clear that all the resulting texture images have higher quality than their corresponding range images.

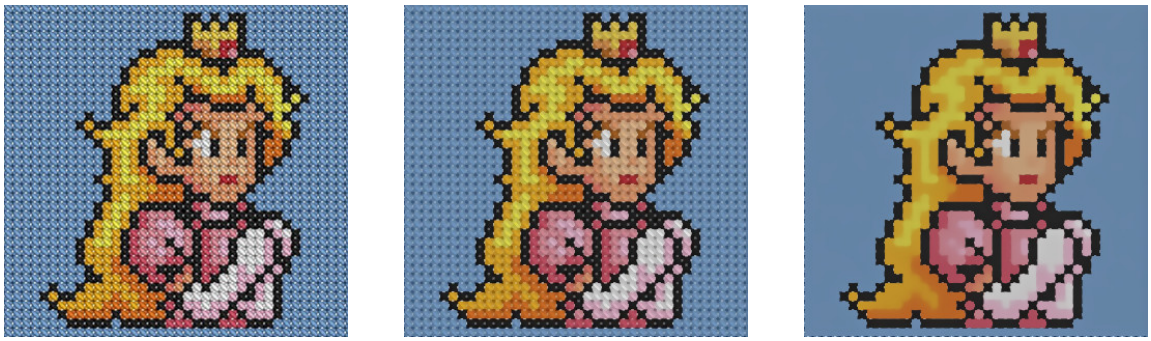
Figure 3.4 shows the effectiveness of the relative total variation model [62] on removing textures and preserving edge features, we choose this method to estimate the range image for our filter. However, other methods can be also used to estimate the range image to produce high quality results. Figure 3.9 shows the results of our filter by applying different range images, which are produced by different detexturing methods. In all of our other results, unless noted, we use RTV model to estimate the range image. Figure 3.9 also shows the superiority of our filter over the bilateral filter with a better range image. Certain parameters are provided for most results: the spatial parameter σ and regularizer weight λ for the RTV model, the spatial parameter σ_s and range paramter σ_r for BF and JBF and number of iterations n for

(a) (b) BF

(c) TV (d) RTV

(e) JBF using TV. (f) JBF using RTV.

Figure 3.9: (a) An input image with textures [63], (b) BF with $\sigma_s = 5.0, \sigma_r = 0.4, n = 4$, (c) TV with $\sigma = 1.0, \lambda = 0.1, n = 3$, (d) RTV with $\sigma = 1.0, \lambda = 0.01, n = 3$, (e)(f) our method with different range images by TV and RTV models, $\sigma_s = 5.0, \sigma_r = 0.1, n = 2$.



(a) (b) (c)

Figure 3.10: (a) input, (b) our filter with $\sigma_s = 1.0$, (c) our filter with $\sigma_s = 5.0$. For both, $\sigma_r = 0.1, n = 2$.

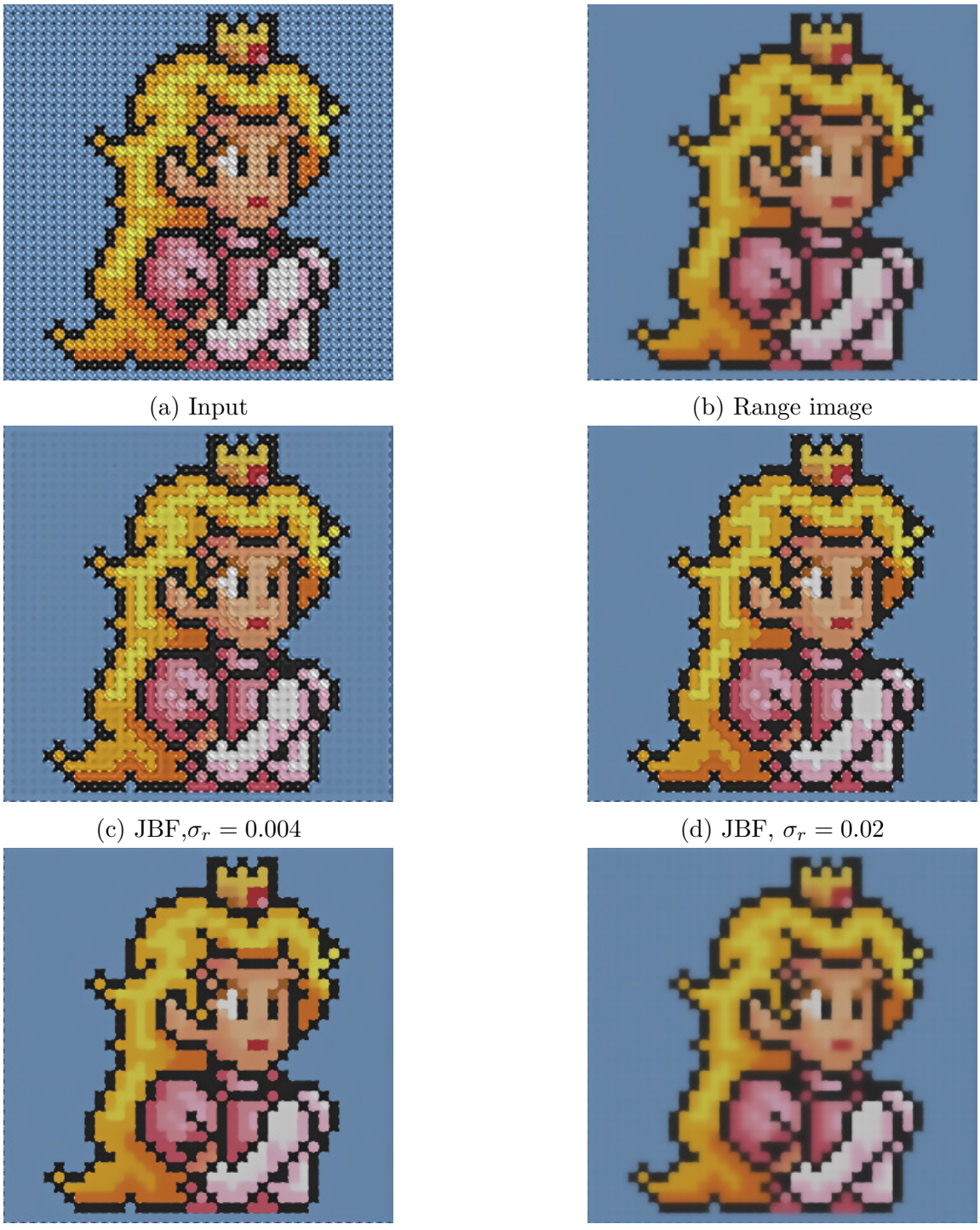
all of them.

3.3 Results

3.3.1 Parameters

We scale all the pixel values to the range $[0, 1]$. The spatial parameter σ_s controls the local shape of the spatial Gaussian kernel, it depends on the scale of the textures. Figure 3.10 shows the effect of this parameter, σ_s needs to be large enough to effectively cover enough amount of texture elements, empirically $\sigma_s = 5.0$ is a good choice for most examples provided. Since the range kernel is computed based on an approximate structure image, it is easier to find a proper range parameter σ_r. Figure 3.11 shows the effect of the range parameter. While a small value for σ_r is not enough to remove textures, a too large σ_r over-blurs the structures.

In terms of the parameters for RTV and TV models, we follow the parameter choice proposed by Xu et al. [62]. The only exception is the detexturing strength λ in Equation 3.5. Our values of λ are generally smaller, this difference may be due to the discretization of the Gaussian kernel. According to our experience on these two models, without the inherent total variation as the denominator in the regularizer,

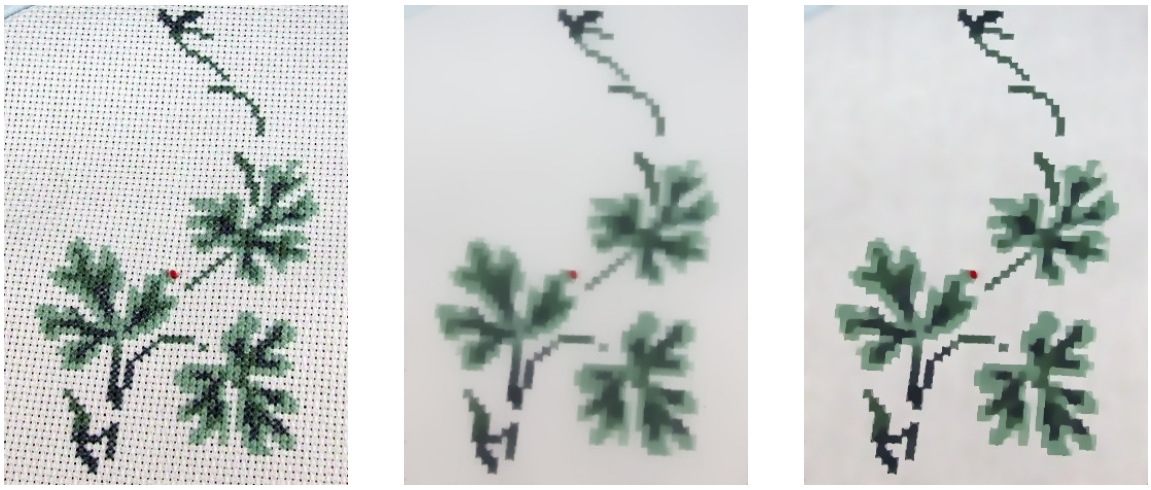


(a) Input (b) Range image

(c) JBF,$\sigma_r = 0.004$ (d) JBF, $\sigma_r = 0.02$

(e) JBF, $\sigma_r = 0.1$ (f) JBF, $\sigma_r = 0.5$

Figure 3.11: Top: (a) input [63], (b) RTV with $\sigma = 2.0, \lambda = 0.004, n = 2$, (c)-(f) our filter with increasing range parameters.



(a) (b) (c)

Figure 3.12: (a) input [63], (b) RTV with $\sigma = 2.0, \lambda = 0.01, n = 2$, (c) our filter with $\sigma_s = 5.0, \sigma_r = 0.1, n = 2$.

the TV model requires relatively higher λ.

3.3.2 Quality

We compare our method mainly with the RTV model [62] since it outperforms other methods on image detexturing. We have tuned the parameters a bit for all the methods to generate best results possible. Various examples with both uniform (Figure 3.12, 3.13) and non-uniform textures (Figure 3.14, 3.15 and 3.16) demonstrate that the proposed JBF can better avoid over-blurring and preserve the edge features, see the close-ups in Figure 3.13 and 3.14. In terms of the RTV model, each iteration of the filtering process is much like a weighted Laplacian smoothing. Consequently, there must be a trade-off between saving the original colors and removing textures. As a result, the recovered image loses its original contrast. In all the tested images, our filter is able to recover the original colors greatly. Note that both RTV model and the proposed JBF assumes that the mean of the texture image is zero; so color loss is unavoidable due to the local averaging if the actual mean of texture intensity

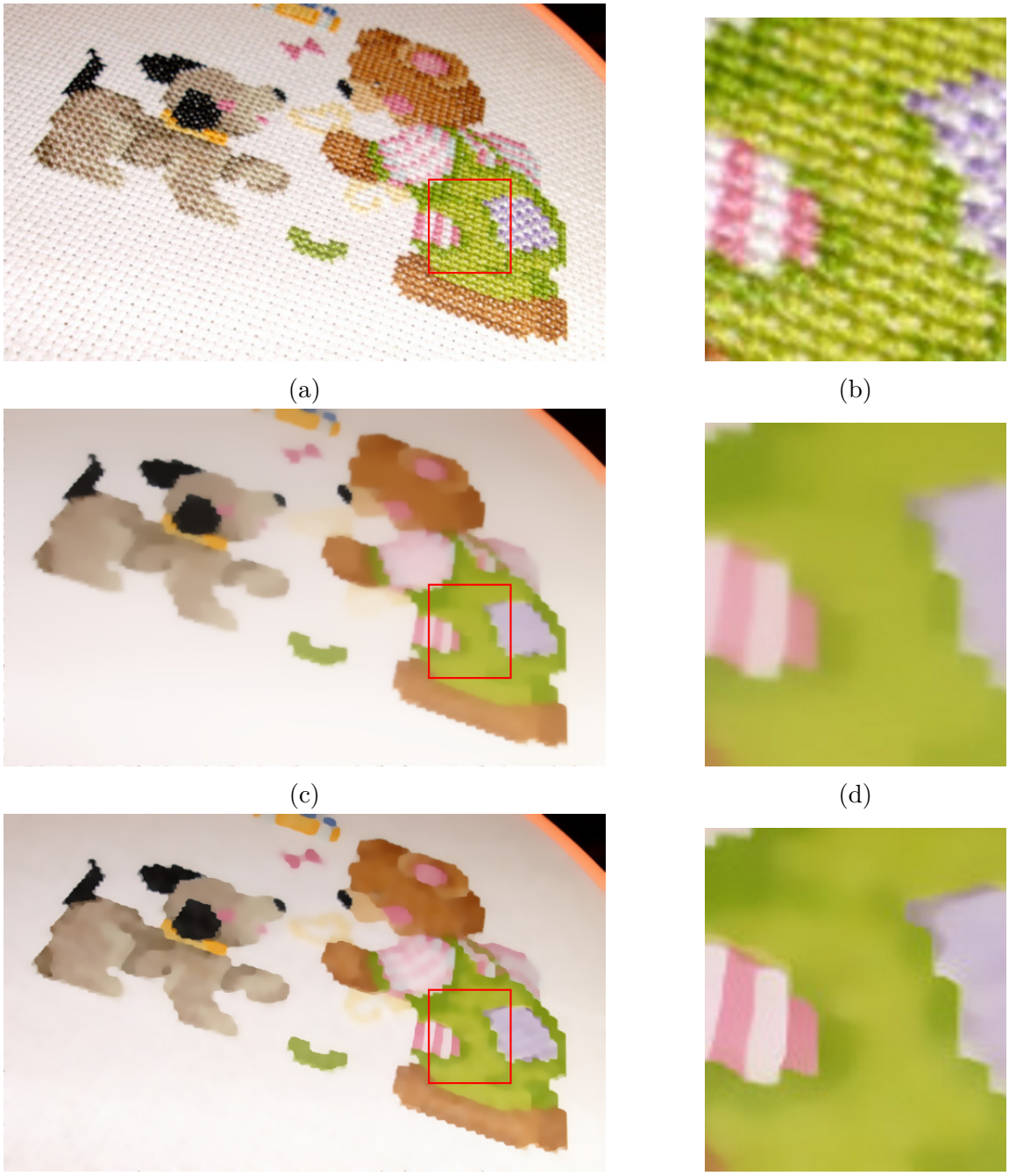


Figure 3.13: (a) input [63], (c) RTV with $\sigma = 4.0, \lambda = 0.002, n = 3$, (e) our filter with $\sigma_s = 5.0, \sigma_r = 0.1, n = 3$, (b)(d)(f) close-ups.

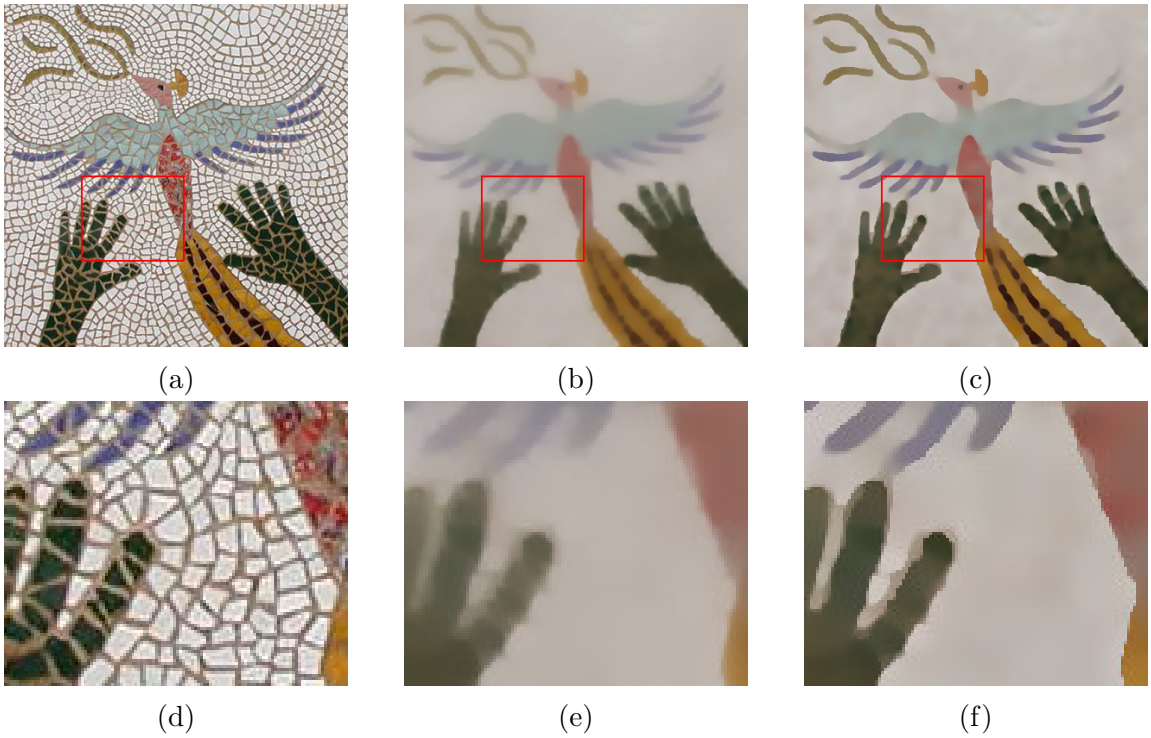


Figure 3.14: Top: (a) input [39], (b) RTV with $\sigma = 2.0, \lambda = 0.008, n = 2$, (c) our filter with $\sigma_s = 5.0, \sigma_r = 0.06, n = 2$. Bottom: Close-ups.

is not zero.

3.3.3 Robustness

Figure 3.9 (e) shows a result of our filter with the range image estimated by the traditional TV model. Even though the range image is over-blurred, it is still able to provide enough structure information for our filter. To further verify the robustness of our filter on the intensity correctness of the range image, we manually scale down and up the range image. As the scaled range image can still capture the major structure changes, the JBF works well.

(a) (b) (c)

Figure 3.15: (a) input [4], (b) RTV with $\sigma = 3.0, \lambda = 0.005, n = 3$, (c) our filter with $\sigma_s = 5.0, \sigma_r = 0.1, n = 3$.

3.3.4 Running Time

The entire filtering process includes two steps: estimating the range image and filtering the image using JBF. In general, the second step takes much less time than the first one. So the total running time of our algorithm depends on the performance of the selected algorithm for estimating the range image. It is safe to say that our method is comparable to most existing image filtering algorithms.

3.4 Conclusions

We present a novel filtering algorithm for extracting main structural information from textured images. We demonstrate that by using an appropriate range image, the proposed joint bilateral filter can remove textures quite effectively. While the unknown texture image itself is a great choice, we propose to preprocess the original textured image to obtain an approximate range image.

3.4.1 Limitations

The performance of our filter relies on the quality of the range image. If the approximate range image cannot exemplify the structural information, our method

(a)

(b)

(c)

Figure 3.16: (a) input [25], (b) RTV with $\sigma = 4.0, \lambda = 0.001, n = 4$, (c) our filter with $\sigma_s = 5.0, \sigma_r = 0.1, n = 3$.

may fail to separate the structures and textures like the traditional bilateral filter. This limitation has also been mentioned by Xu et al. [62] for the RTV model, which may not be able to distinguish between texture and structure that have similar scales.

4. FUTURE WORK

Currently, I have identified following topics for the future work:

Applications of Our Differential Operators Although the potential applications of our differential operator has been roughly discussed in Section 2.5, in-depth research work still needs to be done. For example, compared to the mean curvature, our differential operator can greatly preserve the major features in surface diffusion. We already know that the converged the shape of the modified mean curvature flow on a genus-0 surface is a sphere citeKazhdan12. However, it is unknown that what the converged surface is for the new flow defined by our differential operator. According to our basic test, if the surface converges, it converges to a shape that is dependent on the original shape.

Surface Quality Evaluation While there exist a large number of surface denoising algorithms, it is still hard to evaluate the quality of the results. We have discussed the drawback of a current quantitative evaluation metric in Section 2.5, a desirable evaluation metric has to take both shape similarity and quality into account. A possible choice is to borrow the idea from image domain and use the signal-to-noise ratio (SNR) to evaluate the surface quality. We can use the ground truth as the reference signal for artificial models to compare different algorithms.

Modified Bilateral Filter for Surface Denoising and Detexturing We have already verified the effectiveness of the modified bilateral filter on image detexturing. A natural choice is to extend this filter to either surface denoising or detexturing. Similar to the 2D JBF, a 3D JBF would also need to use a new

range surface for computing the range kernel. However, what the range surface should be or how to obtain such a surface is still an open question.

5. SUMMARY

In this dissertation, I have described two visual data processing techniques: surface denoising and image detexturing. In particular, I have presented:

1. An automatic and robust method for effectively recovering a smooth surface from noisy triangle meshes. The method builds up an energy minimization framework using the L_0 norm of our differential metric. As a part of this work, a novel edge-based discrete differential operator has been developed to measure the planarity locally that is unconditionally stable.

2. A simple and effective method for extracting meaningful structural information from textured images. This method first explores the importance of the range image for a bilateral filter and verifies that a good range image should be able to exemplify the structure changes and neglect the textures in the original input. With an approximate structural image as the range image, the modified bilateral filter can achieve state-of-the-art performance.

REFERENCES

[1] Aimshape shape repository. http://shapes.aim-at-shape.net/viewmodels.php. Accessed: 2014-08-13.

[2] Be the love of life. https://www.betheloveoflife.wordpress.com/. Accessed: 2014-08-13.

[3] Benin oba commemorative heads. http://www.randafricanart.com/Benin_Oba_commemorative_heads.html. Accessed: 2014-08-13.

[4] Crafts; glass & paper. http://www.pinterest.com/susan_fortune/crafts-glass-paper/. Accessed: 2014-08-13.

[5] Princess peach cross stitch. http://www.deviantart.com/morelikethis/artists/316750188?view_mode=2#skins. Accessed: 2014-08-13.

[6] The stanford 3d scanning repositoryy. https://graphics.stanford.edu/data/3Dscanrep/. Accessed: 2014-08-13.

[7] Jean-François Aujol, Guy Gilboa, Tony Chan, and Stanley Osher. Structure-texture image decomposition–modeling, algorithms, and parameter selection. *Int. J. Comput. Vision*, 67(1):111–136, 2006.

[8] Chandrajit L. Bajaj and Guoliang Xu. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans. Graph.*, 22(1):4–32, 2003.

[9] Carissa Bonham. Cupcake dishcloth. http://www.creativegreenliving.com/2007/05/cupcake-dishcloth.html, 2007.

[10] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[11] Jia Chen, Chi-Keung Tang, and Jue Wang. Noise brush: Interactive high quality image-noise separation. *ACM Trans. Graph.*, 28(5):146:1–146:10, 2009.

[12] Ulrich Clarenz, Udo Diewald, and Martin Rumpf. Anisotropic geometric diffusion in surface processing. In *Proceedings of the Conference on Visualization*, pages 397–405, 2000.

[13] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH*, pages 317–324, 1999.

[14] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Proceedings of the Graphics Interface Conference*, pages 145–152, 2000.

[15] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[16] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of ACM SIGGRAPH*, pages 257–266, 2002.

[17] G. Dziuk. An algorithm for evolutionary surfaces. *Numerische Mathematik*, 58(1):603–611, 1990.

[18] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph.*, 23(3):673–678, 2004.

[19] A. F. El Ouafdi, D. Ziou, and H. Krim. A smart stochastic approach for manifolds smoothing. In *Proceedings of the Symposium on Geometry Processing*, pages 1357–1364, 2008.

[20] Hanqi Fan, Yizhou Yu, and Qunsheng Peng. Robust feature-preserving mesh denoising based on consistent subneighborhoods. *IEEE Trans. Vis. Comp. Graph.*, 16(2):312–324, 2010.

[21] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. In *Proceedings of ACM SIGGRAPH*, pages 67:1–67:10, 2008.

[22] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. In *Proceedings of ACM SIGGRAPH*, pages 950–953, 2003.

[23] M. Floater, K. Hormann, and G. Kos. A general construction of barycentric coordinates over convex polygons. *Advances in Comp. Math*, 24:311–331, 2006.

[24] Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. *Computer Graphis Forum*, 23(3):391–400, 2004.

[25] Maggy Howarth. Cobblestone designs. `http://maggyhowarth.co.uk/newwork_dragon.html`, 2001.

[26] Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. In *Proceedings of ACM SIGGRAPH*, pages 943–949, 2003.

[27] Michael Kass and Justin Solomon. Smoothed local histogram filters. *ACM Trans. Graph.*, 29(4):100:1–100:10, 2010.

[28] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. Can mean-curvature flow be modified to be non-singular? *Computer Graphics Forum*, 31(5):1745–1754, 2012.

[29] ByungMoon Kim and Jarek Rossignac. Geofilter: Geometric selection of mesh filter parameters. *Computer Graphis Forum*, 24(3):295–302, 2005.

[30] Kai-Wah Lee and Wen-Ping Wang. Feature-preserving mesh denoising via bilateral normal filtering. In *Proceedings of Computer Aided Design and Computer Graphics*, pages 275–280, 2005.

[31] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of ACM SIGGRAPH*, pages 131–144, 2000.

[32] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph.*, 26(3):22:1–22:5, 2007.

[33] Ce Liu, Richard Szeliski, Sing Bing Kang, C. Lawrence Zitnick, and William T. Freeman. Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):299–314, 2008.

[34] Xinguo Liu, Hujun Bao, Heung-Yeung Shum, and Qunsheng Peng. A novel volume constrained smoothing method for meshes. *Graphical Models*, 64:169–182, 2002.

[35] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3):681–689, 2006.

[36] Jean-Laurent Mallet. Discrete smooth interpolation. *ACM Trans. Graph.*, 8(2):121–144, 1989.

[37] Yves Meyer. *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations: The Fifteenth Dean Jacqueline B. Lewis Memorial Lectures.*

American Mathematical Society, Boston, MA, USA, 2001.

[38] Muzukashi Mondai. The metal element in the sahara. http://reckoninsaswewander.blogspot.com/2014/01/downward-metal-element-in-sahara.html, 2014.

[39] Joe Moorman. Riverson fine art. http://www.riversonfineart.com/mosaic-bird.php, 2012.

[40] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of GRAPHITE*, pages 381–389, 2006.

[41] Sylvain Paris, Pierre Kornprobst, and Jack Tumblin. *Bilateral Filtering*. Now Publishers Inc., Hanover, MA, USA, 2009.

[42] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.*, 23(3):664–672, 2004.

[43] Ulrich Pinkall, Strasse Des Juni, and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[44] Ovreiu E. Poranne, R. and C. Gotsman. Interactive planarization and optimization of 3d meshes. *Computer Graphics Forum*, 32(1):152–163, 2013.

[45] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

[46] S. Schaefer, T. Ju, and J. Warren. A unified, integral construction for coordinates over closed curves. *Computer Aided Geometric Design*, 24(8-9):481–493, 2007.

[47] Yuzhong Shen and Kenneth E. Barner. Fuzzy vector median-based surface smoothing. *IEEE Trans. Vis. Comp. Graph.*, 10(3):252–265, 2004.

[48] Doron Chen Sivan Toledo and Vladimir Rotkin. Taucs: A library of sparse linear solvers. `http://www.tau.ac.il/~stoledo/taucs/`, 2001.

[49] Zhixun Su, Hui Wang, and Junjie Cao. Mesh denoising based on differential coordinates. In *Proceedings of Shape Modeling International*, pages 1–6, 2009.

[50] Kartic Subr, Cyril Soler, and Frédo Durand. Edge-preserving multiscale image decomposition based on local extrema. In *Proceedings of ACM SIGGRAPH ASIA*, pages 147:1–147:9, 2009.

[51] Xianfang Sun, Paul L. Rosin, Ralph R. Martin, and Frank C. Langbein. Fast and effective feature-preserving mesh denoising. *IEEE Trans. Vis. Comp. Graph.*, pages 925–938, 2007.

[52] Xianfang Sun, Paul L. Rosin, Ralph R. Martin, and Frank C. Langbein. Random walks for feature-preserving mesh denoising. *Computer Aided Geometric Design*, 25(7):437–456, 2008.

[53] Tolga Tasdizen, Ross Whitaker, Paul Burchard, and Stanley Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *Proceedings of the Conference on Visualization*, pages 125–132, 2002.

[54] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH*, pages 351–358, 1995.

[55] Gabriel Taubin and Gabriel Taubin. Linear anisotropic mesh filtering. In *IBM Research Report RC22213(W0110-051), IBM T.J. Watson Research*, 2001.

[56] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 839–846, 1998.

[57] David Tschumperlé. Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's. *Int. J. Comput. Vision*, 68(1):65–82, 2006.

[58] Luminita A. Vese and Stanley J. Osher. Modeling textures with total variation minimization and oscillating patterns in image processing. *J. Sci. Comput.*, 19(1-3):553–572, 2003.

[59] J. Vollmer, Robert Mencl, and Heinrich Mller. Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, 18(3):131–138, 1999.

[60] Charlie C. L. Wang. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. *IEEE Trans. Vis. Comp. Graph.*, 12(4):629–639, 2006.

[61] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via l0 gradient minimization. *ACM Trans. Graph.*, 30(6):174:1–174:12, 2011.

[62] Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. Structure extraction from texture via natural variation measure. *ACM Trans. Graph.*, 31(6):139:1–139:10, 2012.

[63] Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. Structure-texture dataset. `http://www.cse.cuhk.edu.hk/leojia/projects/texturesep/database.html`, 2012.

[64] Hirokazu Yagou, Yutaka Ohtake, and Alexander Belyaev. Mesh smoothing via mean and median filtering applied to face normals. In *Proceedings of the Geometric Modeling and Processing*, pages 124–131, 2002.

[65] Hirokazu Yagou, Yutaka Ohtake, and Alexander G. Belyaev. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic

thresholding. In *Proceedings of Computer Graphics International Conference*, pages 28–33, 2003.

[66] Wotao Yin, Donald Goldfarb, and Stanley Osher. Image cartoon-texture decomposition and feature selection using the total variation regularized l1 functional. In *Proceedings of VLSM*, pages 73–84, 2005.

[67] Youyi Zheng, Hongbo Fu, Oscar Kin-Chung Au, and Chiew-Lan Tai. Bilateral normal filtering for mesh denoising. *IEEE Trans. Vis. Comp. Graph.*, 17(10):1521–1530, 2011.