# Fast denoising of surface meshes with intrinsic texture

**H Huang and U. Ascher**

Department of Mathematics, University of British Columbia, Room 121, 1984 Mathematics Road, Vancouver, B.C., Canada V6T 1Z2
Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver, B.C., Canada V6T 1Z4

E-mail: hhzhiyan@math.ubc.ca ascher@cs.ubc.ca

**Abstract.** We describe a fast, dynamic, multiscale iterative method that is designed to smooth, but not over-smooth, noisy triangle meshes. Our method not only preserves sharp features but also retains visually meaningful fine scale components or details, referred to as *intrinsic texture*. An anisotropic Laplacian (AL) operator is first developed. It is then embedded in an iteration that gradually and adaptively increases the importance of data fidelity, yielding a highly efficient multiscale algorithm (MSAL) that is capable of handling both intrinsic texture and mesh sampling irregularity without any significant cost increase.

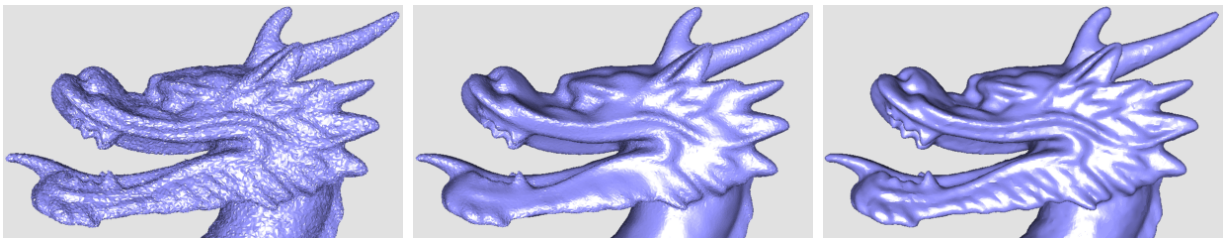
*Keywords*: Surface mesh, anisotropic Laplacian, multiscale denoising, intrinsic texture, mesh irregularity, volume shrinkage.

## 1. Introduction

The problem of denoising, or smoothing 3D meshes has received a lot of attention in recent years due to the increasing use of 3D scanning and acquisition technology. For example, meshes supplied by laser scanning devices often carry high-frequency noise in the position of the vertices, so a mesh smoothing algorithm is required to rapidly remove noise while preserving real artifacts in the acquired data.

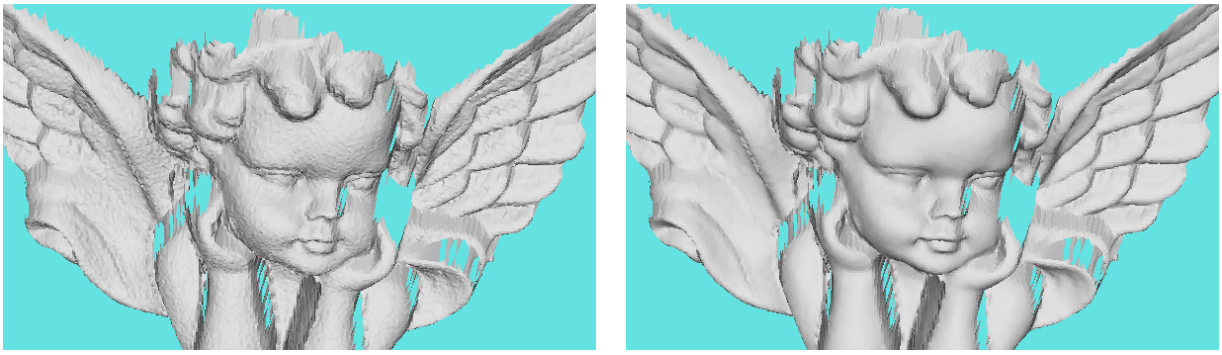
Most techniques for 3D mesh smoothing have predecessors in the literature on the significantly simpler image denoising problem [27, 23, 10, 28, 5, 29, 25]. However, "lifting" image processing methods up to 3D surface meshes is not trivial or automatic, see e.g. [14]. Specifically, whereas in image processing the data as well as the sought surface consist of scalar intensity height maps corresponding to an underlying regular grid of pixels, here the data as well as the sought surface are defined by sets of points $\mathbf{x}_i$ irregularly placed in 3D. A preprocessing step constructs a triangular mesh with the data points as vertices, and this mesh connectivity is retained throughout the smoothing process. We denote such a surface mesh by $S$, with its set of vertices $V(S) = \mathbf{x} = \{\mathbf{x}_i;\ i = 1, \ldots, N\}$ and set of directed edges $E(S)$. The noise in the data is therefore not only in intensity heights at known pixel values but also in what corresponds to the location of the data. Indeed, the following inter-related issues arise:

- Some sort of separation between the surface location of the data points and their value in normal direction is required. Thus, it is often desired to denoise the surface by adjusting data values at each vertex in the normal direction but not in directions tangential to the surface. Motion of vertices in tangential directions is referred to as vertex drifting.
- The mesh is irregular, and this gives rise to several issues addressed further below that do not arise in the corresponding image processing problem.
- The mesh describes a volume in 3D, and it is important to ensure that this volume does not shrink noticeably during the denoising process.



(a) (b) (c)

**Figure 1.** Comparing with non-iterative bilateral filtering [19]: (a) noisy Dragon head model (100K Vertices); (b) smoothed model presented in [19] (80 sec on a 1.4Ghz Athlon); (c) smoothed model by our multiscale algorithm (4 iters, $K = 1/2$, 10 sec on our laptop).



(a) (b)

**Figure 2.** Rapid unknown noise removal while preserving fine scale details: (a) scanned Angel model (25K Vertices); (b) smoothed by our multiscale scheme (3 iters, $K = 1/2$, 1.8 sec).

Several algorithms for smoothing 3D meshes were proposed in the 1990s involving geometric diffusion [31, 16]. Later it was recognized that smoothing based on isotropic diffusion inherently smears out important features such as edges, which correspond to height discontinuities in image processing, so methods based on anisotropic diffusion were introduced for various purposes such as fairing height fields and bivariate data, and smoothing surfaces and level sets [10, 32, 7, 3, 30, 4, 26, 17, 8]. Irregularities in mesh sampling introduce additional difficulties [15, 20, 9] that occasionally distort results and significantly slow algorithms down.

The most recent of the above methods reconstruct large sharp edges and corners of surfaces particularly well (see especially [17]). Also, the more elaborate surface representation methods of [24, 13], that are designed for other purposes, generally can handle sharp features well. However, these methods typically come with a relatively high price tag, both in terms of cost per iteration and in the number of iterations required to achieve satisfactory results. Moreover, they often require the user to provide unintuitive parameter values, including a threshold value for the anisotropic diffusion process. Such a value may highly depend on the given model, with different threshold values possibly yielding very different fairing results. Thus, an automatic selection formula that can adaptively determine the threshold value for different meshes is desirable.

Another approach, that of bilateral filtering (BF), has given rise to methods that more rapidly yield results of a quality similar to anisotropic diffusion, albeit with less theoretical justification [33, 14, 19, 35]. These methods usually require very few, cheap iterations, and can cost but a tiny fraction of the computational effort required to carry out an elaborate anisotropic diffusion process. In fact it can be argued that an algorithm that terminates after what amounts to 2 or 3 forward Euler time steps does not capture the nonlinear dynamics of a complex diffusion process, i.e. that methods based on accurately

capturing such dynamics are inherently slower. However, the results generated by these bilateral filtering variants may strongly depend on the range of a vertex neighborhood and the manner in which tangent planes are approximated. It is not obvious, and not clearly specified in the references, how to adaptively and automatically determine these important algorithm components for different models.

Moreover, none of these algorithms perform very well for problems with significant fine scale features. For instance, the hair and scales of the famous Stanford dragon typically disappear when applying a fast bilateral filtering type method, see Figs. 1(b), 7(d) and 7(e) as well as the relevant examples in [30, 19, 35]. We refer to such visually meaningful fine scale components or details as *intrinsic texture*. Such texture, which may appear either regularly or irregularly, is harder to deal with because its separation from noise depends on the *local* mesh resolution. (A different notion of texture is considered in [8].) And yet, this is where the simpler and faster methods often are most visibly unsatisfactory.

The goal of this paper is to design a fast, robust denoising method that performs well also in the presence of significant intrinsic texture. A basic anisotropic Laplacian (AL) iteration is developed first that costs essentially half that of the basic bilateral filtering iteration and requires no user-specified parameters. Then, since intrinsic texture crucially depends on the local scale on which it noticeably varies, an adaptive procedure is developed in which the AL operator is repeatedly applied while the local fidelity to the data is gradually increased. The resulting multiscale anisotropic Laplacian (MSAL) algorithm produces significantly better results than AL or bilateral filtering type methods at a fraction of the cost of sophisticated methods such as [17, 8, 24]. See Figs. 1, 2, 6, 7 and 8, where important fine scale features are well-preserved. All our final results are achieved using very few mesh sweeps of the simplest and most directly parallelizable sort. Thus, the entire cost of the linear algebra portion of our algorithm is lower than, for instance, one V-cycle in a multigrid preconditioner for solving a linear system arising when carrying out just one time step of an implicit scheme such as used in [7, 8, 17].

To distinguish the two components of smoothing and data fidelity, we first revisit anisotropic geometric diffusion in Section 2, starting with the continuous model given in (1). We then proceed to develop an AL operator that can be viewed as applying image processing techniques in local coordinates at each mesh vertex, yielding a procedure that adaptively determines the filter parameters. The resulting quality of reconstruction of this AL operator is much better than that of isotropic Laplacian operators [31, 16, 15, 20, 9, 32] while the cost remains low. In Section 3 we then build our multiscale iterative scheme, which has a distinct advantage when intrinsic texture is important. The MSAL method is also advantageous in cases where an approximated vertex normal near a boundary erroneously points to an averaged direction, which may lead to a rounded edge or to a significant volume shrinkage. Implementation details, results, discussion and conclusions follow.

## 2. Geometric Laplacian operators

### *2.1. Continuous and discrete diffusion models*

Starting from a continuous geometric diffusion model, one seeks a one-parameter family of embedded manifolds $\{\mathcal{M}(t)_{0\le t\le T}\}$ in $\mathbb{R}^3$ and corresponding parameterizations $\mathbf{x}(t)$, such that for $0 < t \le T$

$$\begin{aligned} \partial_t \mathbf{x}(t) - \mathtt{div}_{\mathcal{M}}(G\ \mathbf{grad}_{\mathcal{M}}\,\mathbf{x}(t)) &= \lambda(\mathbf{x}(0) - \mathbf{x}(t)), \\ \mathcal{M}(\mathbf{x}(0)) &= \mathcal{M}_0. \end{aligned} \tag{1}$$

Here $\mathcal{M}_0$ is the given, noisy data and $G$ is usually a diffusion tensor acting on the tangent space of $\mathcal{M}$ to enhance large sharp edges. A detailed discussion on choosing this diffusion tensor for surface meshes can be found in [7, 4, 8]. For isotropic diffusion on surfaces (where $G$ is an identity), the counterpart of the Laplacian is the Laplace-Beltrami operator $\Delta_{\mathcal{M}} = \mathtt{div}_{\mathcal{M}}\ \mathbf{grad}_{\mathcal{M}}$, and $-\Delta_{\mathcal{M}}\mathbf{x} = 2\kappa(\mathbf{x})\mathbf{n}(\mathbf{x})$, where $\kappa$ is the mean curvature and $\mathbf{n}$ is the normal direction (see [11, 7] for details). The final integration time $T$ controls the amount of diffusion administered. The nonnegative parameter function $\lambda(\mathbf{x}(t))$ controls data fidelity. This equation can be viewed as describing a composition of mean curvature normal motion caused by $\kappa(\mathbf{x})$, smoothing out noise, and a retrieving force toward the original surface caused by $(\mathbf{x}(0) - \mathbf{x}(t))$, recapturing fine scale texture. The magnitude of $\lambda$ determines the weights in this composition, see [34].

Next, a manifold $\mathcal{M}$ is replaced by a triangular surface mesh $S$ with its sets of vertices $\mathbf{x}_i$ and directed edges $\mathbf{e}_{i,k}$ as defined before. If two distinct vertices $\mathbf{x}_i$ and $\mathbf{x}_k$ are linked by an edge $\mathbf{e}_{i,k} = \mathbf{x}_k - \mathbf{x}_i$ then we denote $k \in \mathcal{N}(i)$ and the edge length $l_{i,k} = |\mathbf{e}_{i,k}|$. The given data is a noisy mesh of this sort, and we denote its vertices $\mathbf{x}(0) = \mathbf{v} = \{\mathbf{v}_i;\ i = 1, \ldots, N\}$.

For all methods considered in this article an iteration can be written as updating each vertex $i$ by

$$\mathbf{x}_i \longleftarrow \mathbf{x}_i + \tau\Delta\mathbf{x}_i + \lambda_i(\mathbf{v}_i - \mathbf{x}_i), \tag{2}$$

in obvious analogy to (1). The first iteration starts with the given data. Unless otherwise noted we set $\tau = 1$. Moreover, for all calculations of this section we set $\lambda \equiv 0$, the model thus becoming comparable to [27, 10, 14, 19]. The rest of this section describes various possibilities for defining $\Delta\mathbf{x}_i$ in (2).

Note that in general $\Delta\mathbf{x}_i$ is not intended to be simply an approximation to a Laplacian, but we find this notation convenient nonetheless. Moreover, it is important to realize that (1) is artificial and must be regarded as providing a guidance, rather than describing a true continuous physical process for denoising, unlike e.g. the Navier-Stokes equations which are indeed widely believed to describe physical fluid flow. The iteration (2) on the surface mesh $S$ does not necessarily derive its validity as an approximation of (1) [2].

### *2.2. Time discretization and discrete isotropic Laplacian*

Recall next three discrete isotropic Laplacian operators. The first is the umbrella operator [31]

$$\Delta \mathbf{x}_i = \frac{1}{m_i} \sum_{k \in \mathcal{N}(i)} \mathbf{e}_{i,k}, \tag{3}$$

where $m_i = |\mathcal{N}(i)|$, the cardinal number of one-ring neighbor set $\mathcal{N}(i)$. This is a linear form implying the assumption that all neighboring edge lengths are equal to one. Hence it can serve as an effective smoother if the targeted mesh is close enough to being regular. However, when the model has different discretization rates, significant local distortions are introduced by this umbrella operator, and a better choice is the scale-dependent version [15].

Still, this weighting scheme does not solve problems arising from unequal face angles. Also, the method tends to equalize the lengths of the edges, which may or may not be desirable. Based on a better approximation to the mean curvature normal, a third scheme was proposed in [9, 22] that doesn't produce vertex tangential drifting when surfaces are relatively flat and compensates both for unequal edge lengths and for unequal face angles. All three schemes are based on isotropic diffusion, though, and may easily smear sharp features.

The basic iterative process of (2) is related to (1) using the forward Euler method in "time". This, however, may work much less effectively on irregular meshes, requiring significantly more computational effort because of the larger number of time steps (or iterations) as well as additional cost per iteration for the mean curvature method. One could think of implicit time integration methods, such as backward Euler, that do not have similar stability restrictions on their time step. However, our multiscale algorithm below requires such a small amount of computational work as to be much faster than any truly implicit method, and its stability restriction does not give rise to stiffness issues. Moreover, the time step that the finer part of the mesh mandates when using forward Euler may not necessarily produce an effective smoother for coarser parts of the mesh when using geometric diffusion, even though there is no stability problem.

Thus, it is the *discrete dynamics*, rather than the continuous dynamics corresponding to (1), that counts [2]. For geometric diffusion the time step should ideally be adjusted locally, corresponding to local edge lengths, but that is hard to do economically. Alternatively, it is possible to view the forward Euler discretization as a steepest descent method for the minimization of

$$\mathbf{x}^* = \mathtt{argmin}\{\int_{\mathcal{M}} |\mathbf{grad}_{\mathcal{M}}\mathbf{x}| + \lambda \|\mathbf{x} - \mathbf{x}(0)\|^2\}$$

in suitable norms. The "time step" is determined by a line search that somehow averages

contributions of different mean curvatures, and the iteration is better conditioned hence more effective the closer the mesh is to uniformity.

### *2.3. Discrete anisotropic Laplacian*

To better reconstruct sharp features, consider next using an anisotropic Laplacian (AL) operator on a surface mesh $S$, where updates are made at each vertex in the direction of its normal. This is similar to bilateral filtering (BF) and different from (3) and the schemes of [17, 32].

The idea behind the AL operator is simple. A given normal $\mathbf{n}_i$ at a vertex $\mathbf{x}_i$ defines a tangent plane at this vertex. Restricting the update $\Delta\mathbf{x}_i$ to be in the direction of $\mathbf{n}_i$, we can view it as updating the height intensity of a 2D image defined on this tangent plane, i.e. in local coordinates, where the offsets of the neighboring vertices, denoted by $\{h_{i,k} = \mathbf{e}_{i,k}^T\mathbf{n}_i, k \in \mathcal{N}(i)\}$, act as image intensities at pixels. Let us consider one vertex $\mathbf{x}_i$ with its one-ring neighborhood first. Thus, we define

$$\Delta\mathbf{x}_i = \frac{1}{C_i}\left(\sum_{k\in\mathcal{N}(i)} g(h_{i,k})h_{i,k}\right)\mathbf{n}_i,$$

where $C_i = \sum_{k\in\mathcal{N}(i)} g(h_{i,k})$ is a normalization factor and $g$ is the feature indicator. In determining $g$ we then follow image processing sources, specifically [6, 28].

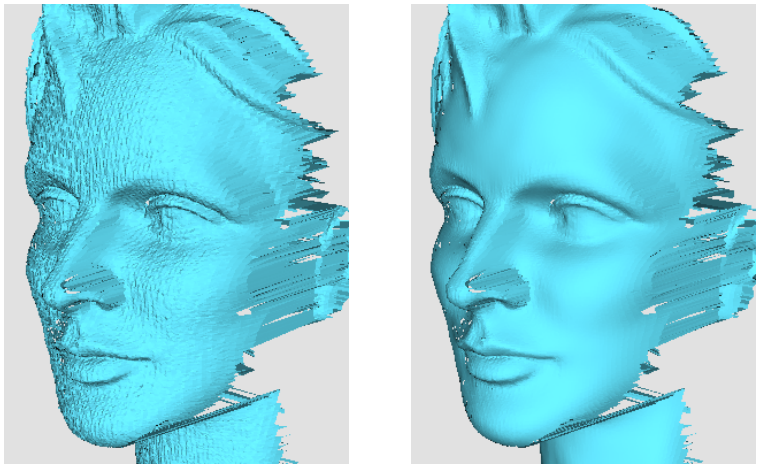
Several robust filter choices, e.g. Huber (a variant of total variation [1]), Lorentz, Gauss and Tukey, have been discussed and compared in [6, 28, 12] for images. These studies conclude that Gaussian or Tukey filters are more robust in the presence of outliers, hence better preserve important features. According to our numerical experiments, this conclusion holds also for surface meshes. So, we employ the Gaussian filter

$$g(h_{i,k}) = exp(-\frac{h_{i,k}^2}{2\sigma_i^2}),$$

due to its robustness, stability and simplicity. This filter clearly reduces the influence of neighbors that contain large discontinuities in normal space during the smoothing process. The point from which neighboring vertices are treated as outliers depends on the parameter $\sigma_i$.

The image processing literature [6, 28] uses some tools from robust statistics to automatically estimate this *robust scale* $\sigma_i$ as the mean or median absolute deviation of the given image intensity gradient. The main idea is that $\sigma_i$ should characterize the variance of the majority of data within a region, so outliers would then be determined relative to this background variation. Since $h_{i,k}$ is considered as height intensity of a local image, we set

$$\sigma_i = 2\,\mathtt{mean}(\mathtt{abs}(\mathbf{h}_i - \mathtt{mean}(\mathbf{h}_i))), \tag{4}$$



(a) (b)

**Figure 3.** Fast denoising of data with unknown noise without losing significant features: (a) scanned Lady face model (41K Vertices); (b) smoothed model by AL (3 iters, 3.0 sec).

where $\mathbf{h}_i = (h_{i,k} \ \ldots \ , k \in \mathcal{N}(i))$ and the leading constant is derived from the fact that the inverse of mean absolute deviation of a zero-mean normal distribution with unit variance is near 2.

Thus, our discrete anisotropic Laplacian (AL) operator is finally defined and can be rewritten as

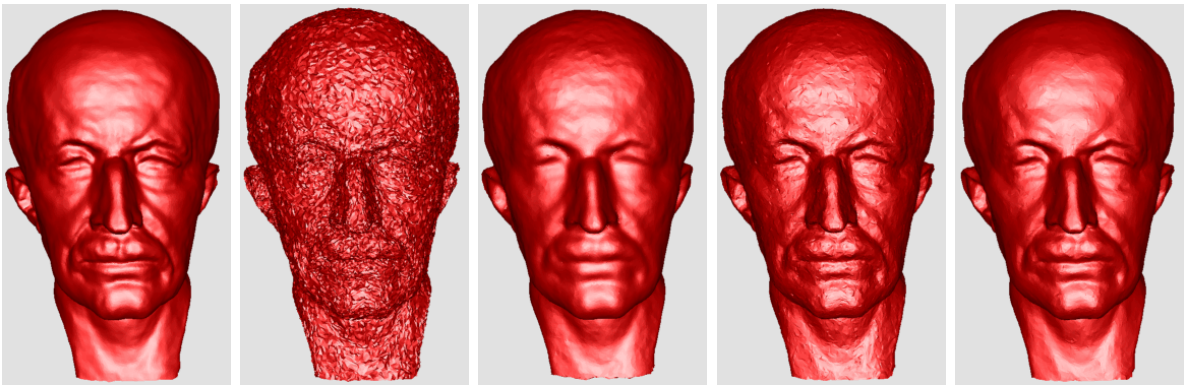
$$\Delta \mathbf{x}_i = \frac{1}{C_i} \left( \sum_{k \in \mathcal{N}(i)} g(h_{i,k}) \mathbf{n}_i^T \mathbf{e}_{i,k} \right) \mathbf{n}_i = \left( \frac{\sum_{k \in \mathcal{N}(i)} exp(-\frac{|h_{i,k}|^2}{2\sigma_i^2}) \, h_{i,k}}{\sum_{k \in \mathcal{N}(i)} exp(-\frac{|h_{i,k}|^2}{2\sigma_i^2})} \right) \mathbf{n}_i, \tag{5}$$

with $\sigma_i$ given by (4). The normalization scales the operator such that a step size $\tau = 1$ can be stably used in the explicit method (2). From Figs. 3 and 4, it is clear that the above scheme is able to rapidly remove noise while preserving significant surface features. To connect, at least formally, between the AL operator and the anisotropic diffusion operator $\mathtt{div}_{\mathcal{M}}(G \ \mathbf{grad}_{\mathcal{M}} \mathbf{x})$, we can write (cf. [32])

$$\Delta \mathbf{x}_i = \sum_{k \in \mathcal{N}(i)} W_{i,k} \mathbf{e}_{i,k}, \qquad W_{i,k} = \frac{1}{C_i} g(h_{i,k}) \mathbf{n}_i \mathbf{n}_i^T,$$

where the edge weights $W_{i,k}$ are symmetric, semi-definite, $3 \times 3$, rank-1 matrices. However, the validity of the AL operator is not derived from (1) directly.

The method obtained by employing the AL (5) in (2) with $\tau = 1$ may be viewed as a variant of BF where the domain filter used to measure closeness there is defined as the identity here. We refer to the resulting BF variant as *our BF*. Straightforward operation count shows that our BF iteration is basically twice as expensive as AL. Our



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

**Figure 4.** Removing heavy Gaussian noise from a corrupted model: (a) original Max Planck model (50K Vertices); (b) model is corrupted by about 20% noise (1/5 of the mean edge length) in the normal direction; (c) smoothed by explicit mean curvature flow (20 iters, 128 sec); (d) smoothed by AL without normal mollification; (e) smoothed by AL with normal mollification (4 iters, 5.1 sec).

BF, in turn, is a variant of the BF method presented in [14]. In numerical experiments carried out for all examples reported in this article, our BF does not generate significantly better results than AL. In order to pursue a better reconstruction by using BF, one has to set an appropriate neighborhood radius $\rho$ and then get the neighboring set of vertices, i.e. $\mathcal{N}(i)$ is defined by the set of points $\mathbf{x}_k$ that satisfy $||\mathbf{x}_k - \mathbf{x}_i|| < \rho$. So the included neighbors in [14] depend on Euclidean distance, approximating geodesic distance, rather than on connectivity. Unfortunately, the questions of how to automatically choose $\rho$ for different models and how to better approximate geodesic distances between vertices are far from clear and may easily result in heavy computation, even if there is no problem with recapturing intrinsic texture.

The description of the AL operator is not complete until we specify how to approximate the normal $\mathbf{n}_i$ at vertex $\mathbf{x}_i$. Since we want to work on a smooth but not over-smooth vertex normal field, we estimate the vertex normal $\mathbf{n}_i$ as the average of the adjacent face normals, leaving it unchanged during the fairing process

$$\mathbf{n}_i = normalize \left( \sum_{k \in \mathcal{N}(i)} \frac{\mathbf{e}_{i,k} \times \mathbf{e}_{i,k+1}}{|\mathbf{e}_{i,k} \times \mathbf{e}_{i,k+1}|} \right).$$

Another possible choice is to weigh adjacent face normals by their face areas, but there has been no visible improvement in our numerical experiments upon introducing this. More importantly, we must be aware that very noisy given data may lead to an unstable computation of initial vertex normals, since the normals are first-order properties of the mesh. Thus, for extremely noisy data, such as in the Max Planck example of Fig. 4, we apply a mollification step, employing the normalized average of neighboring vertex normals

as initial $\mathbf{n} = \{\mathbf{n}_i;\ i = 1, \ldots, N\}$, i.e. running the umbrella operator on the normal field. Compare Fig. 4(d) vs Fig. 4(e). This guarantees enough smoothing over relatively large non-feature surfaces and requires much less computation than using the two-ring or a larger neighborhood as in [14]. For common scanned models without very heavy noise, such mollification turns out not to be necessary.

## 3. Handling intrinsic texture

Although AL is very efficient for smoothing some models as demonstrated in Figs. 3 and 4, it cannot avoid over-smoothing in the presence of essential intrinsic texture, especially when using $\lambda = 0$ in (2). The same holds for various BF variants, see Figs. 1(b), 7(e) and 7(f). A simple first step toward a better solution is then to use $\lambda > 0$, thus increasing the influence of the given data during the denoising process at all iterates [34]. This also helps to reduce the effect of volume shrinkage over several iterations [31, 9] so as to be essentially unnoticeable in the method developed below.

Rewriting (2) as

$$\mathbf{x}_i \longleftarrow (1 - \lambda_i)\mathbf{x}_i + \tau\Delta\mathbf{x}_i + \lambda_i\mathbf{v}_i,$$

we may view $\Delta\mathbf{x}_i$ as the smoothing contribution of the current iteration. It is clear that for stability we must require $0 < \lambda_i < 2$. However, here we require nonnegativity as well, which means $0 < \lambda_i \leq 1$ at all vertices. This ensures that the smoothing contributions are accumulated monotonically, preventing undesirable mesh deformation. Moreover, when $\tau$ is relatively smaller, the influence of the mesh smoothing modification gets smaller too, hence finer scale components are recaptured from the given mesh. Thus, we keep $0 < \lambda \leq 1$ and reduce $\tau$ gradually by setting at the $j$th iteration

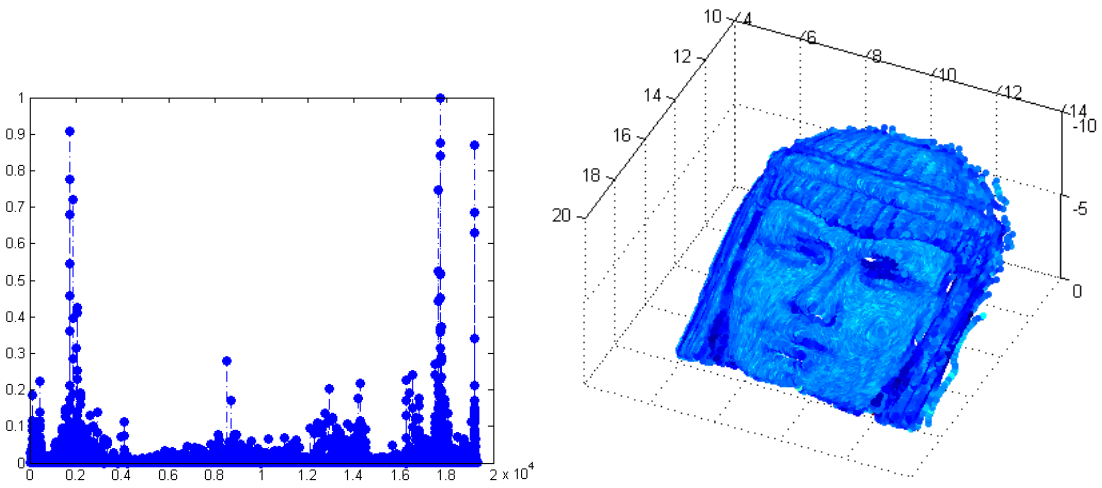
$$\tau = K^j, \tag{6}$$

where $K$ is a parameter, $0 < K < 1$.

To see how the iteration progresses, let us denote $\mathbf{x} = \mathbf{x}^{(j)}$ at the $j$th iteration, with $\mathbf{x}^{(0)} = \mathbf{v}$. We then unroll the recursion, obtaining

$$\begin{aligned}
\mathbf{x}_i^{(j+1)} - \mathbf{v}_i &= (1 - \lambda_i^{(j)})(\mathbf{x}_i^{(j)} - \mathbf{v}_i) + K^j\Delta\mathbf{x}_i^{(j)} \\
&= (1 - \lambda_i^{(j)})\left[(1 - \lambda_i^{(j-1)})(\mathbf{x}_i^{(j-1)} - \mathbf{v}_i) + K^{j-1}\Delta\mathbf{x}_i^{(j-1)}\right] + K^j\Delta\mathbf{x}_i^{(j)} \\
&= \sum_{k=0}^{j} c_i^{(j-k)} K^{j-k}\Delta\mathbf{x}_i^{(j-k)},
\end{aligned}$$

where $c_i^j = 1$, $k = 0$ and $c_i^{(j-k)} = \prod_{m=0}^{k-1}(1 - \lambda_i^{(j-m)})$, $k > 0$. Note that with $0 < \lambda \leq 1$ everywhere we have nonnegative weights $c^{(j-k)}$ for the smoothing contributions. Moreover, these weights shrink as $k$ increases. Furthermore, for $K < 1$ the later smoothing




(a) (b)

**Figure 5.** Illustrating the function $\lambda$ at each vertex of the Monk model (see also Fig. 9): (a) magnitude of $\lambda_i$ at vertex $\mathbf{x}_i$ in the first smoothing run; (b) the values of $\lambda$ specify the colors of the vertices as RGB values.

contributions weigh less than the earlier ones in the sum, and more strongly so the smaller $K$ is. Our default value is $K = 1/2$, and this often works well, as evident in the numerical examples.

Next we want to determine the mesh function $\lambda$. In image processing, it can be estimated if some statistical information on the noise is known, especially if we keep $\lambda$ constant. On a surface, since noise is carried in the position of vertices and changes all the neighboring height intensities $h_{i,k}$, we expect that $\lambda_i$ should depend on this noise effect. In other words, we want to apply enough smoothing over domains where the surface varies little on the scale of the mesh while preserving more original information over surfaces with fine scale features. Fortunately, we already have a robust estimator on local mesh features at hand, namely, the Guassian parameter $\sigma_i$ adaptively determined by (4). Recall that $\sigma_i$ is larger over surface patches containing intrinsic features and smaller over non-feature regions, due to the dependence on local height intensity. This naturally gives rise to the formula

$$\lambda_i = \sigma_i/\bar{\sigma}, \quad \text{where } \bar{\sigma} = \max\{\sigma_i;\ i = 1, \ldots, N\}, \tag{7}$$

and all quantities are evaluated at the current mesh $j$. Fig. 5(a) illustrates the magnitude of $\lambda$ at each vertex with $j = 0$. Note how rather far from constant the function $\lambda$ is. In Fig. 5(b), we apply darker color the larger the value of $\lambda_i$ is. This figure clearly shows that $\lambda$ specified by (7) does roughly indicate the feature properties of the mesh surface.

The combination of the normalized formula for $\lambda$ and the geometric decrease of $\tau$ yields a multiscale algorithm. In fact, the iteration (2) with (6) can be viewed as a forward Euler

discretization of the ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = K^t \Delta \mathbf{x} + \lambda(\mathbf{v} - \mathbf{x}), \quad t \geq 0,$$

applied at each vertex $i$ starting from $t = 0$ with step size equal to 1. Thus, at the start of the $j$th iteration, $t = j$. Clearly, as $t \to \infty$ the process converges at steady state to the original data. More importantly, the same holds also for the discrete dynamics. In practice of course we stay well away from this limit by taking only a few iterations. Moreover, an attempt to take a step with large $\lambda$, applying say a backward Euler discretization to this differential equation, would reintroduce the noise without significant smoothing, thus missing the goal. Hence, our gradual multiscale approach is necessary.

## 4. Implementation, numerical results and discussion

### *4.1. Implementation details*

**Algorithm 1** Multiscale Anisotropic Laplacian (MSAL) :

Data: vertices $\{\mathbf{v}_i;\ i = 1, \ldots, N\}$ and normals $\{\mathbf{n}_i;\ i = 1, \ldots, N\}$;
Initialize: $\mathbf{x} = \mathbf{v}$, neighborhood $\{\mathbf{x}_k, k \in \mathcal{N}(i)\}$, $m_i = |\mathcal{N}(i)|$;
Input parameters: $Niter$ and $K$;
**for** $j = 0 : Niter - 1$ **do**
  **for** $i = 1 : N$ **do**
    **for** $k = 1 : m_i$ **do**
      $\mathbf{h}_{i,k} = (\mathbf{x}_k - \mathbf{x}_i) \cdot \mathbf{n}_i$;
    **end for**
  **end for**
  Compute array $\sigma$ by Equation (4);
  Compute array $\lambda$ by Equation (7);
  $\mathbf{g}_k = exp(-\mathbf{h}_k^2./(2\sigma^2))$;
  $d = \sum_{k \in \mathcal{N}(i)} (\mathbf{g}_k. * \mathbf{h}_k)./ \sum_{k \in \mathcal{N}(i)} \mathbf{g}_k$;
  $\mathbf{x} = \mathbf{x} + K^j d. * \mathbf{n} + \lambda. * (\mathbf{v} - \mathbf{x})$;
**end for**

The complete multiscale anisotropic Laplacian process is given in Algorithm 1. We have implemented it in a set of MATLAB routines. Run times are reported on a laptop with a 2.0Ghz Pentium M CPU, except for those in Figs. 1(b) and 8(b).

The instructions following the nested loops in Algorithm 1 involve array operations, $i = 1, \ldots, N$, and we follow MATLAB convention in distinguishing element-wise array products from more general loops. The inner iteration is based on our AL operator

defined by (2) and (5), although a BF variant could easily be employed instead. Clearly, the number of floating point operations needed for one AL iteration is $O(mN)$, where $m = \max\{m_i,\ i = 1, \ldots, N\}$. Note that $m \ll N$.

There are two input parameters in our algorithm. As mentioned earlier, after a few iterations the relative importance of later smoothing contributions decreases, more aggressively the smaller $K$ is. The scale of $K$ must depend on the application. It should be tuned to be small enough to capture intrinsic texture and large enough not to recapture noise too quickly. In practice the default value often works very well. See Fig. 9 below for more on this issue.

Regarding the choice of the total number of iterations $Niter$, the figures displayed generally represent our best results using the "eye norm" and occasionally knowing the ground truth (i.e. the true model). See Fig. 6 below for more on this issue. This approach is of course common; however, it is important to note that the *sensitivity* to the choice, as for the choice of $K$, is low here. For most cases we have found that $Niter \leq 4$ is adequate, and $Niter = 4$ is a stable default value. If we allow $Niter$ to get larger then the results generated by MSAL get noisier, as explained in the previous section.

We emphasize that our algorithm is completely specified and easily programmable. All our results are reproducible by the reader in a straightforward manner.

### *4.2. Comparative results*

Comparing results to those of others in the literature is a delicate task, not only because this involves different platforms and programming levels but also because typically not all algorithm details and relevant data are available. Here, fortunately, in addition to the fact that it only requires two stable input parameters $K$ and $Niter$, the computational complexity of AL is obviously significantly lower than that of leading anisotropic diffusion schemes [3, 4, 8, 17] in terms of computation (assembly) at each node at each iteration, and in either the linear algebra cost per iteration or the number of iterations required. Comparing against the BF alternatives, a basic AL step costs roughly half that of a BF method if the same neighborhood range is defined, and less more generally. Moreover, the multiscale scheme built on AL does not result in any significant extra cost. Hence we are essentially left to demonstrate the quality improvements obtained by the introduction of MSAL.

In the first sets of results presented in Section 1, Fig. 1 compares our method with the non-iterative bilateral filtering method proposed in [19]. Clearly, MSAL recaptures significantly more intrinsic texture with less computational effort. Fig. 2 shows that three multiscale steps can preserve most fine scale features such as the eyelids and the grids on the angel's wings, while sufficiently smoothing flat regions such as the face and arms. The second set of results displayed in Figs. 3 and 4 shows how powerful and efficient our AL operator is for models that do not possess much noticeable fine scale features.

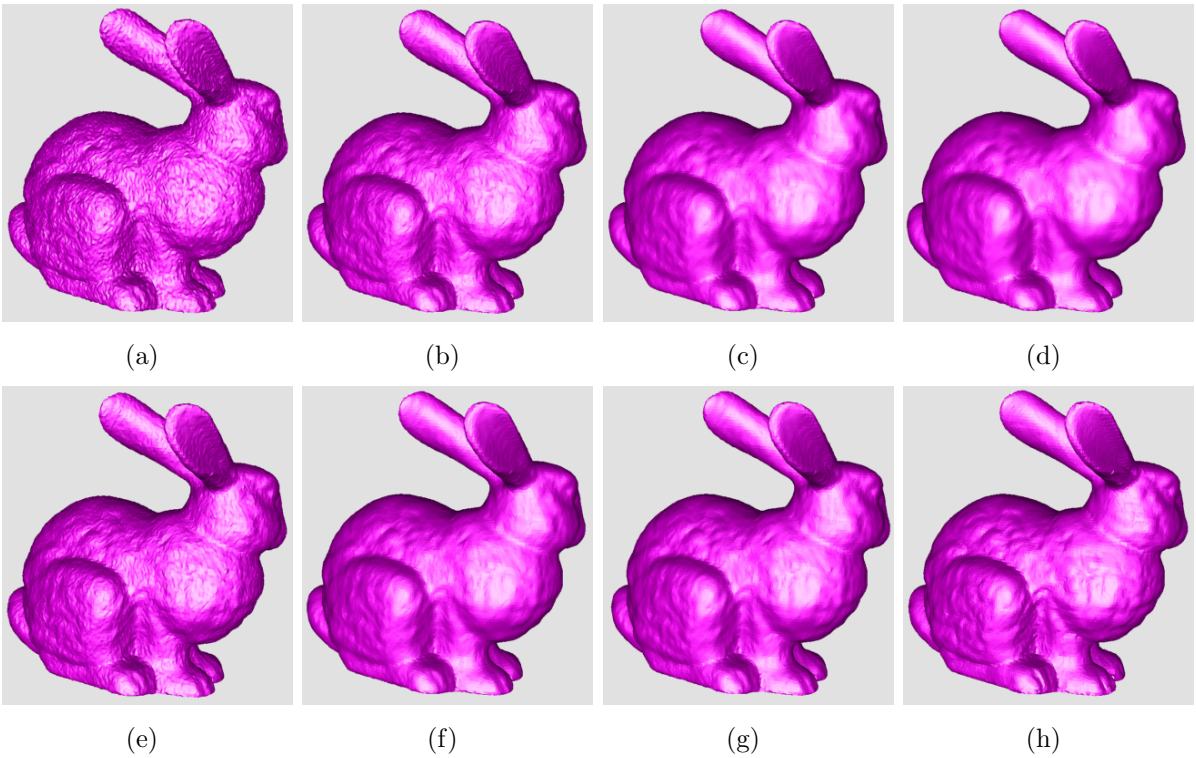



**Figure 6.** Comparing AL with MSAL (each run 3 iters, AL - 2.3 sec, MSAL - 2.6 sec): (a) corrupted Bunny model (35K Vertices); (b) smoothed model after one iteration of AL; (c) smoothed model after two iterations of AL; (d) smoothed model after three iterations of AL; (e) smoothed model after one iteration of MSAL with $K = 1/2$; (f) smoothed model after two iterations of MSAL; (g) smoothed model after three iterations of MSAL; (h) original clean Bunny model with fine scales.

In Fig. 6, we investigate and compare denoising results of AL and MSAL step by step. Observe that AL is able to quickly remove noise, but it also blurs visually distinguished intrinsic texture on the bunny's fur, while MSAL is capable of recapturing fine scales and simultaneously maintaining necessary smoothing with almost the same computational cost, compare Figs. 6(d) vs 6(g).

Fig. 7 compares MSAL against bilateral filtering (BF), both for our BF and as in [14]. Obviously, BF does not produce better results than AL for models with intrinsic texture, while MSAL yields significantly better results than all three variants without a data fidelity term.

Fig. 8 compares smoothing results for a fine Igea model by nonlinear anisotropic diffusion in [17] and by MSAL. Observe the better details of face and hair of the model in Fig. 8(c) obtained using much fewer iterations by our multiscale algorithm. We hasten to say, however, that comparing MSAL to anisotropic diffusion is trickier than comparing BF to AL and MSAL. On one hand our algorithm is not capable of reproducing the results displayed in [17] for CAD-type models on coarse meshes, and on the other hand the choice

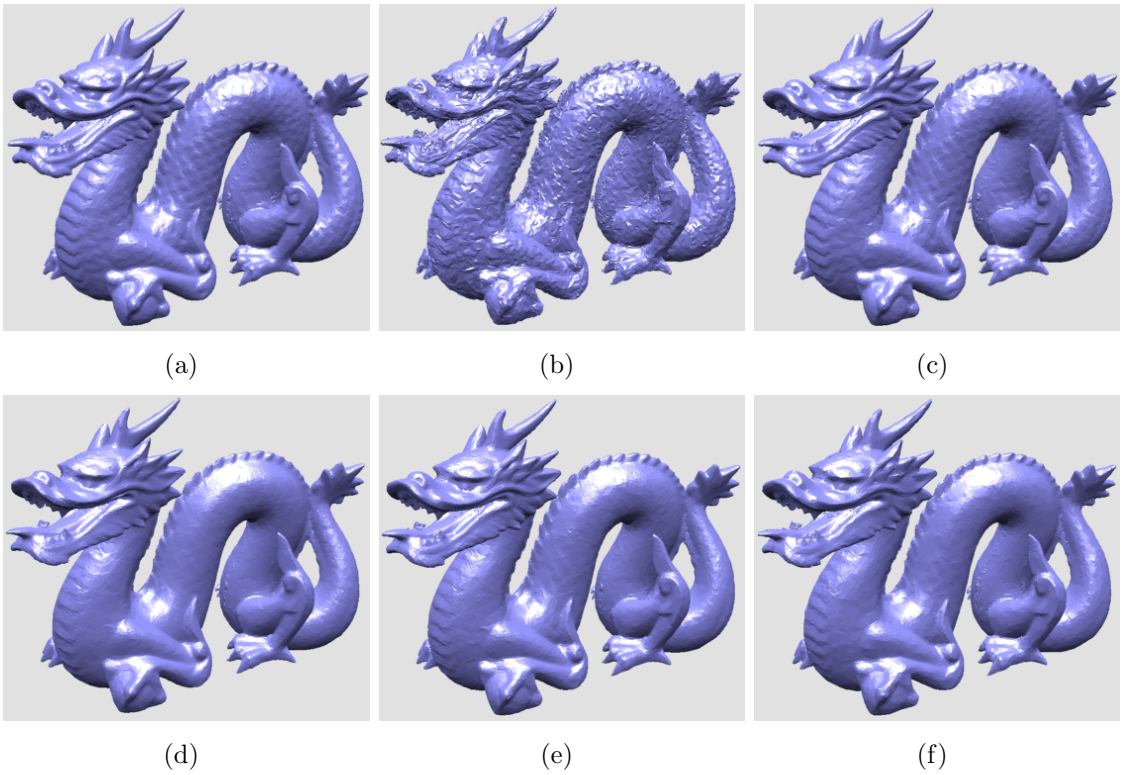
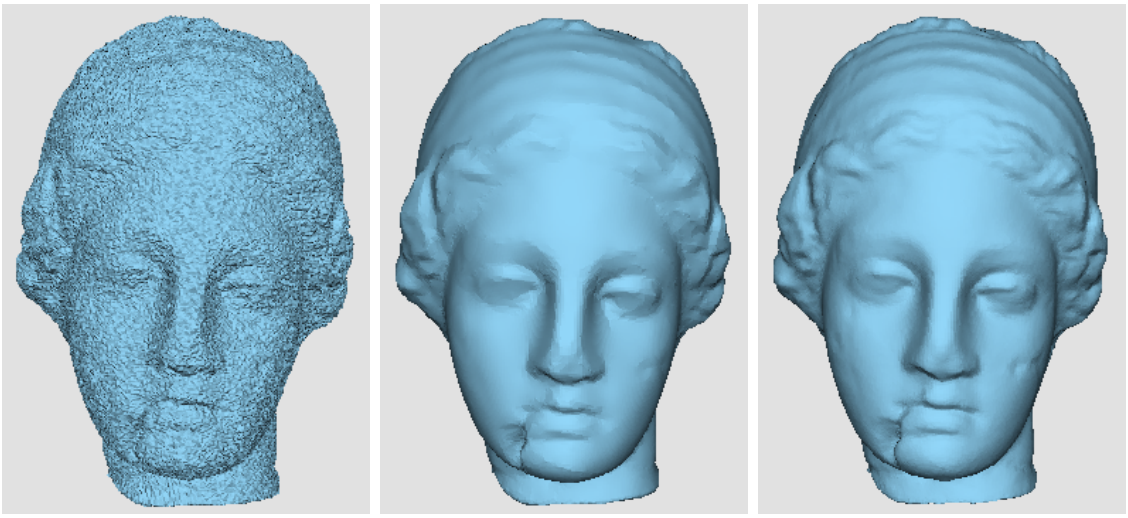



**Figure 7.** Comparing with bilateral filtering: (a) original Dragon model (50K Vertices); (b) noisy model corrupted by about 10% random tangential and normal noise; (c) smoothed model by the multiscale algorithm MSAL (3 iters, $K = 1/5$, 3.8 sec); (d) smoothed model by AL where intrinsic texture has been removed (3 iters, 3.4 sec); (e) smoothed model by our version bilateral filtering with one-ring neighborhood and similarity $\sigma_s$ determined by the standard deviation of local height intensities (3 iters, 5.9 sec); (f) smoothed model by bilateral filtering as in [14] with smooth region radius $\rho = 8.8$ and closeness $\sigma_c = 4.4$ selected by trial and error and similarity $\sigma_s$ determined automatically (3 iters, 8.4 sec).

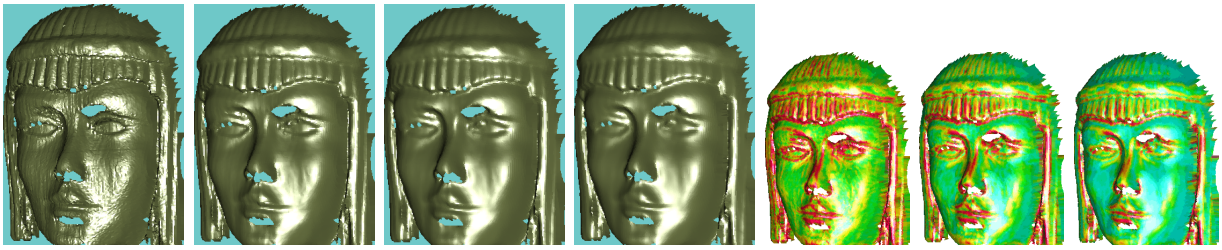
of methods and parameters within an anisotropic diffusion family required to produce a result such as Fig. 8(b) is far more complex and delicate than what is required to produce or reproduce Fig. 8(c).

Different applications may well yield different requirements for retaining intrinsic texture. By simply adjusting the scalar $K$ we can get different flavors. For example, in Fig. 9 one may want to keep wrinkles or other details on the monk's face for a more natural look. For this, we simply decrease $K$, e.g. Figs. 9(b) or 9(c). But caution must be exercised because too small a $K$ may bring back unwanted noise too quickly. On the other hand, a larger value of $K$ yields a smoother model, see e.g. Fig. 9(d). In brief, under the same total number of MSAL iterations we can very easily get different results suited for different goals.



(a) (b) (c)

**Figure 8.** Comparing with nonlinear anisotropic diffusion [17]: (a) fine Igea model (135K vertices) with intrinsic texture corrupted by noise; (b) smoothed by the nonlinear anisotropic diffusion method in [17], (25 itns, 33 secs on the 1.7 GHz Centrino laptop; result courtesy of K. Hilderbrandt); (c) smoothed by MSAL, (K = 1/2, 4 itns, 13 secs).



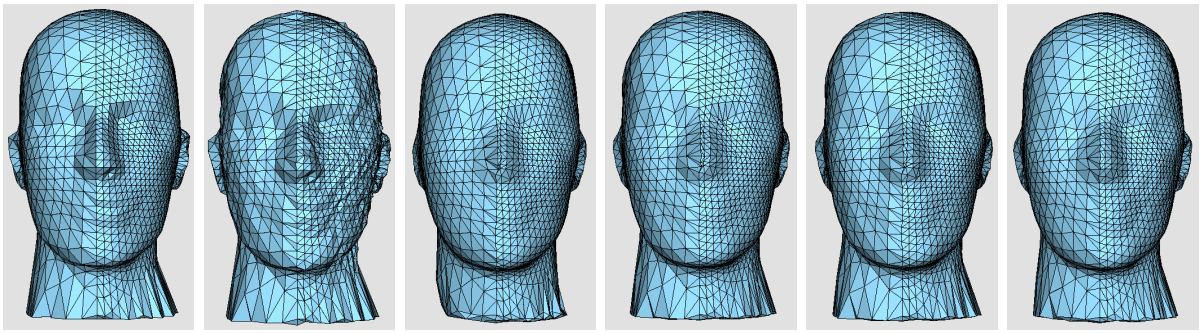
(a) (b) (c) (d) (e) (f) (g)

**Figure 9.** Controlling retained amount of intrinsic texture using MSAL (each run 4 iters, 1.8 sec): (a) scanned Monk model (19K Vertices) contains unknown noise; (b) smoothed model with $K = 1/4$; (c) smoothed model with $K = 1/2$; (d) smoothed model with $K = 1$; subfigures (e)-(g) are mean curvature visualizations of (b)-(d), respectively.

### *4.3. Handing mesh sampling irregularity and volume shrinkage*

Since our discrete anisotropic Laplacian operator (5) is not scaled by the local Voronoi area [9, 22], during the flow the velocity of vertices strongly depends on the geometric discretization. Points adjacent to large triangles move faster than points adjacent to small triangles. For regular meshes such as most real scanned data sets, this is not an issue because one can relate (2) and (1) by a constant rescaling of "time". For highly nonuniform meshes, however, this may deform the surface in an undesirable way, see e.g. Fig. 10(c).

One option to make up for this is to multiply the operator (5) by the inverse of a lumped mass matrix [17] which is a diagonal matrix with elements equal to 1/3 of the one-



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

**Figure 10.** Scaling for mesh irregularity: (a) original Head mesh (2.5K Vertices) contains different discretization rates; (b) mesh is corrupted by 10% Gaussian noise; (c) smoothed by AL (3 iters, $\tau = 1$, 0.29 sec); (d) smoothed by AL with lumped mass matrix scaling (8 iters, $\tau = 0.02$, 3.6 sec); (e) smoothed by edge length scaled-AL (8 iters, $\tau = 0.02$, 1.4 sec); (f) smoothed by MSAL (3 iters, K = 1/3, 0.29 sec).

ring area of the vertices. However, the smaller step size required for stability, resulting in the need for more iterations, and the higher computational cost of each iteration, make this scaling method relatively expensive. A cheaper alternative is to use edge length scaling, similar to the scale-dependent umbrella operator [15], because the edge lengths also contain discretization information and are much easier to compute. Although this scaling does not take angles between edges into account, it is sufficient for examples such as the one depicted in Fig. 10. A scale-dependent AL operator for irregular meshes is given by

$$\Delta \mathbf{x}_i = \left( \frac{\sum_{k \in \mathcal{N}(i)} (exp(-\frac{|h_{i,k}|^2}{2\sigma_i^2})\, h_{i,k}/l_{i,k})}{\sum_{k \in \mathcal{N}(i)} (exp(-\frac{|h_{i,k}|^2}{2\sigma_i^2}) l_{i,k})} \right) \mathbf{n}_i.$$

The results in Figs. 10(d) and 10(e) produced using these two different scalings are very similar. Although the edge scaling still suffers from a smaller step size, hence the need for more iterations, the timing in Fig. 10 indicates that the weight computation in each iteration is significantly cheaper than using lumped mass scaling.

Our MSAL scheme, unchanged, is cheaper than either of these scaling alternatives. In Fig. 10(f) we can detect no significant distortion or unwanted artifacts, almost like when lumped mass matrix or edge length scalings are used and even better at the intersection of finer and coarser segments. We observe that the real contribution comes from the stronger retrieving force toward the original surface on vertices adjacent to larger triangles in the *multiscale* evolution. The algorithm with geometrically decreasing smoothing weight $K^j$ and *multiscale* version of data fidelity term (i.e. $\lambda$ varies on vertices) acts not only for retaining fine features but also for retaining the original structure of the mesh while still removing noise. Of course, if the mesh is much more severely nonuniform, with very large

**Table 1.** Volume comparison on the smoothed meshes.

| | Dragon | Bunny | Fandisk | Head |
|---|---|---|---|---|
| Original | 11175335 | 1550.7 | 20.243 | 215.0 |
| Noisy | 11197273 | 1551.9 | 20.246 | 215.3 |
| AL | 11110735 | 1545.1 | 19.984 | 202.8 |
| MSAL | 11156748 | 1547.5 | 20.180 | 210.5 |

local mesh ratios and highly differing angles, then the above observations may change.

In Table 1 we demonstrate the power for anti-shrinking of this multiscale scheme. The interior volumes are calculated as in [9]. The volumes of four meshes smoothed by AL as well as MSAL are recorded respectively. Clearly, the multiscale algorithm helps toward restoring the original volumes.

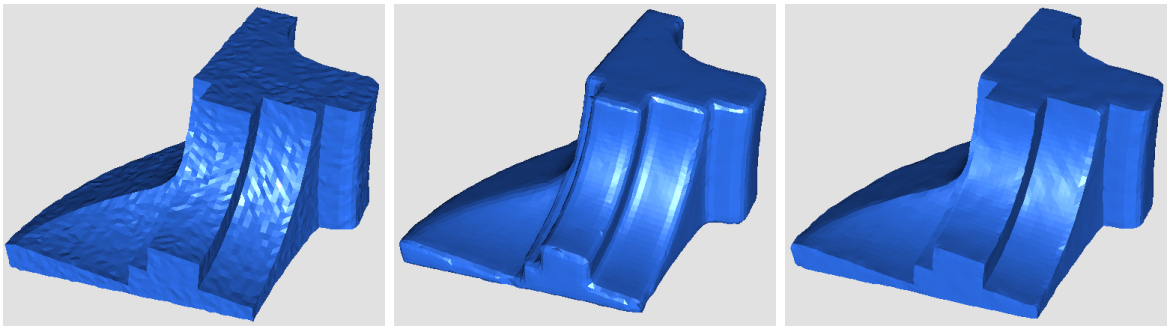
## 5. Conclusions

We have methodically developed a fast adaptive multiscale algorithm that is capable of retaining intrinsic texture as well as handling mesh irregularities while denoising 3D triangular meshes. This approach does not rely directly on the availability of a continuous diffusion process. In addition to the parameter specifying the very small total number of iterations, the only user-specified parameter is $K$ in Algorithm 1. The latter parameter can be easily and intuitively adjusted to obtain purpose-dependent results, as described in Sections 3 and 4.2 (see especially Fig. 9).

The basic iteration employs the anisotropic Laplacian (AL) operator with a new, automatic, justified parameter choice. Then by gradually and locally adding roughness scales to the admitted mesh, a multiscale anisotropic Laplacian (MSAL) smoothing scheme is defined. The high level of efficiency, parallelizability and transparency of the MSAL iteration are expressed in Algorithm 1, yielding a method that can almost automatically achieve high quality fairing results at rapid rates. All our results can be readily reproduced by the reader.

Note that the use of AL as the basic iteration in our multiscale scheme, although usually rather satisfactory, is not mandatory. Not only bilateral filtering methods can immediately replace AL. Also a more sophisticated method such as [17, 8] could be used to replace AL in Algorithm 1, obtaining a deluxe version that could for instance yield distinctively better edge sharpness reconstruction for certain models such as Fig. 11(a). However, the clarity of parameter specification may be lost, and the computational cost of such a combination could be much higher than that of Algorithm 1 as well.

Of course there are some applications where retaining intrinsic texture is less important (e.g. [21]). Also, there are other aspects such as consistency in pattern by which intrinsic texture may differ from random noise. Recovering intrinsic texture based on machine



(a) (b) (c)

**Figure 11.** Multiscale scheme reconstructs large edge sharpness (each run 3 iters, 0.56 sec): (a) noisy Fandisk mesh (6.5K Vertices); (b) smoothing by AL smears sharp edges; (c) MSAL restores edge sharpness with $K = 1/2$.

learning techniques is an interesting, separate challenge.

Besides intrinsic texture, another important surface feature is large sharp edges, typically arising from CAD-like models. The AL algorithm with the mean absolute deviation scaling occasionally does not preserve large edge sharpness very well, see Fig. 11(b). Though this can be essentially improved using MSAL, see Fig. 11(c), some noticeable noise is still brought back on large flat regions by this algorithm, and edge sharpness, especially on the curved ridge, needs to be further enhanced. We have designed a hybrid algorithm which is based on clustering techniques to handle this situation and report on this under separate cover in [18] .

## Acknowledgments

We thank Dr. Alla Sheffer for useful discussions, Dr. Shachar Fleishman for providing his code, Ewout van den Berg for converting some 3D objects and Dan Tulpan for providing a visualization package. We are indebted to K. Hildebrandt for providing both the data in Fig. 8(a) and the result of Fig. 8(b). Other models presented in this paper have been downloaded off the websites of Thouis Jones, Jean-Yves Bouguet, Mathieu Desbrun and the Stanford 3D scanning repository.

## References

[1] U. Ascher, E. Haber, and H. Huang. On effective methods for implicit piecewise smooth surface recovery. *SIAM J. Scient. Comput.*, 28(1):339–358, 2006.

[2] U. Ascher, H. Huang, and K. van den Doel. Artificial time integration. *BIT*, 47:3–25, 2007.

[3] C. Bajaj and G. Xu. Adaptive surface fairing by geometric diffusion. In *Symp. Computer Aided Geometric Design*, pages 731–737. IEEE Computer Society, 2001.

[4] C. Bajaj and G. Xu. Anisotropic diffusion on surfaces and functions on surfaces. *ACM Trans. Graphics (SIGGRAPH)*, 22(1):4–32, 2003.

[5] D. Barash. A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. *IEEE Trans. PAMI*, 24(6):844–847, 2002.

[6] M.J. Black, G. Sapiro, D.H. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE trans. image processing*, 7(3):421–432, 1998.

[7] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing. In *Proceedings IEEE Visualization*, pages 397–405, 2000.

[8] U. Clarenz, U. Diewald, and M. Rumpf. Processing textured surfaces via anisotropic geometric diffusion. *IEEE Transactions on Image Processing*, 13(2):248–261, 2004.

[9] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings SIGGRAPH*, pages 317–324, 1999.

[10] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Anisotropic feature–preserving denoising of height fields and bivariate data. *Graphics Interface*, pages 145–152, 2000.

[11] M. P. Do-Carmo. *Riemannian Geometry*. Birkhauser, Boston-Basel-Berlin, 1992.

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

[13] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graphics (SIGGRAPH)*, 24(3):544–552, 2005.

[14] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Trans. Graphics (SIGGRAPH)*, 22(3):950–953, 2003.

[15] K. Fujiwara. Eigenvalues of laplacians on a closed riemannian manifold and its nets. In *Proceedings AMS*, volume 123, pages 2585–2594, 1995.

[16] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proceedings SIGGRAPH*, pages 325–334, 1999.

[17] K. Hildebrandt and K. Polthier. Anisotropic filtering of non-linear surface features. *EUROGRAPHICS*, 23(3):391–400, 2004.

[18] H. Huang and U. Ascher. Fast 3d surface mesh denoising with edge preservation and mesh regularization. *Submitted*, 2007.

[19] T. Jones, F. Durand, and M. Desbrun. Non-iterative, feature preserving mesh smoothing. *ACM Trans. Graphics (SIGGRAPH)*, 22(3):943–949, 2003.

[20] L. Kobbelt, S. Campagna, J. Vorsatz, and H. P. Seidel. Interactive multiresolution modeling on arbitrary meshes. In *SIGGRAPH Proceedings*, pages 105–114, 1998.

[21] C. Lee, A. Varshney, and D. Jacobs. Mesh saliency. *ACM Trans. Graphics (SIGGRAPH)*, 24(3):659–666, 2005.

[22] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath*, Berlin, Germany, 2002.

[23] N. Nordstrom. Biased anisotropic diffusion - a unified regularization and diffusion approach to edge detection. *Image and Vision Computing*, 8:318–327, 1990.

[24] Y. Ohtake, A. Belyaev, and M. Alexa. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *Proc. Symp. Geometry Processing*, pages 149–158. EUROGRAPHICS Assoc., 2005.

[25] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation based image restoration. *SIAM J. multiscale model. simul.*, 4:460–489, 2005.

[26] J. Peng, V. Sterla, and D. Zorin. A simple algorithm for surface denoising. In *Proceedings IEEE Visualization*, pages 107–112, 2001.

[27] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[28] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge, 2001.

[29] E. Tadmor, S. Nezzar, and L. Vese. A multiscale image representation using hierarchical (bv,$l^2$)

decompositions. *SIAM J. multiscale model. simul.*, 2:554–579, 2004.

[30] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *Proceedings IEEE Visualization*, pages 125–132, 2002.

[31] G. Taubin. A signal processing approach to fair surface design. In *Proceedings SIGGRAPH*, pages 351–358, 1995.

[32] G. Taubin. Linear anisotropic mesh filters. In *IBM Research Technical Report RC-22213*, October 2001.

[33] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *IEEE Intl. Conf. Computer Vision*, pages 839–846, Bombay, India, 1998.

[34] J. Vollmer and R. Mencl. Improved laplacian smoothing of noisy surface meshes. *EUROGRAPHICS*, 18(3):131–139, 1999.

[35] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graphics (SIGGRAPH)*, 23(3):641–648, 2004.