

A Bayesian Method for Probable Surface Reconstruction and Decimation

JAMES R. DIEBEL and SEBASTIAN THRUN
Stanford University
and
MICHAEL BRUENIG
Robert Bosch Corporation

Preprint

To Appear: *ACM Transactions on Graphics* 25(1), Jan 2006

We present a Bayesian technique for the reconstruction and subsequent decimation of 3D surface models from noisy sensor data. The method uses oriented probabilistic models of the measurement noise, and combines them with feature-enhancing prior probabilities over 3D surfaces. When applied to surface reconstruction, the method simultaneously smooths noisy regions while enhancing features, such as corners. When applied to surface decimation, it finds models that closely approximate the original mesh when rendered. The method is applied in the context of computer animation, where it finds decimations that minimize the visual error even under nonrigid deformations.

Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry; Object Modeling—*Boundary representations*

General Terms: Measurement, Theory

Additional Key Words and Phrases: fairing, denoising, smoothing, decimation, remeshing, bayesian, learning

1. INTRODUCTION

We provide a unified Bayesian method for probable surface reconstruction and decimation from noisy surface meshes. Recent research has led to a flurry of acquisition devices of 3D shape models of physical objects. However, the noise in the data acquisition process induces models that have rough and uneven surfaces.

The Bayesian method is based on a mathematical model of the noisy measurement process, and on a prior over surface shapes. With an appropriate prior, the method smooths away noise in the sensed surface while enhancing visible surface features, such as edges. We show how an appropriate prior can be learned from data, yielding superior reconstruction results in application domains where specific features (e.g., right angle corners) are paramount. We also show how advanced optimization techniques can be applied to recover a probable surface reconstruction efficiently.

In the context of decimation, the Bayesian method utilizes a mathematical model that characterizes the appearance of a surface model when rendered. We show how to recover faithful decimations of detailed surface meshes using a compactness prior. We apply this method to static meshes, and also to animated meshes, where the optimal decimation is a function of the deformation process that occurs during animation. Empirically, we find that the Bayesian method produces decimated models which closely resemble in appearance the corresponding dense models when rendered, even when used for animation.

Author's address: Stanford University, 353 Serra Mall, Gates Building Room 116, Stanford, CA 94305-9010; email: diebel@stanford.edu.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2005 ACM 0730-0301/2005/0100-0001 \$5.00

A detailed analysis of the results show that the Bayesian approach yields empirical results comparable to or, in many instances, noticeably superior to previously published works. The use of a Bayesian technique for smoothing and decimation enables us to separate our assumptions on the noise of the sensor, from the statistical properties of actual 3D objects. It also enables us to bring to bear efficient optimization techniques for finding a probable reconstruction or decimation.

The method provides a number of advances, all documented in the text. For example, when smoothing range images, the smoothing can be oriented in the direction of the measurement axis along which the expected error is largest. When decimating meshes, the specific deformation can easily be taken into account to determine the best decimation.

2. RELATED WORK

The topic of denoising has a long history in computer graphics and the related field of image processing. Taubin [1995] generalized frequency-domain image filtering to the problem of fair surface design, yielding a linear-time algorithm. Desbrun et al. [1999] introduced the use of diffusion processes for mesh smoothing, using an implicit integration scheme to solve the a curvature flow partial differential equation. Both of these approaches are quite fast but fail to distinguish between noise and sharp structure in the underlying object. To solve this problem, anisotropic diffusion, inspired by work in image processing, was applied to the problem of denoising height fields by Desbrun et al. [2000]. Related techniques were applied to more general geometric representations by Clarenz et al. [2000] and Bajaj and Xu [2003]. Tasdizen et al. [2002] uses a volumetric level set representation and performs anisotropic diffusion on the normal vector field; an update step keeps the surface shape consistent with the evolving normal vector field. The results are impressive but the computation times are very large and the memory requirements limit the size of the models that can be processed. Most recently Jones et al. [2003] proposed an efficient non-iterative technique for feature-preserving mesh smoothing, and Fleishman et al. [2003] adapted bilateral filtering for application to mesh denoising.

The Bayesian technique presented here is very different in flavor: It enables the user to explicitly model the sensor noise, and to tune the method to the specific application domain through adapting the Bayesian priors. However, the use of Bayesian and energy-optimization techniques is not new to other subfields of computer graphics. The connection between Bayesian techniques and optimization was first pointed out by Szeliski and Terzopoulos [1989]. Normal potentials were introduced in [Terzopoulos et al. 1987]. Building on this work, Szeliski and Tonnesen [1992] applied energy minimization techniques to surface reconstruction with sparse data using quadratic normal potentials; see also Stahl et al. [2002]. However, as we shall show, quadratic potentials tend to over-smooth and therefore destroy small surface features—which our approach enhances. We draw our motivation to use non-quadratic potentials from a recent study in computer vision by Levin et al. [2002], who showed that the natural potentials in image space are sub-linear. Finally, our method is also related to techniques for surface deformation and finite element techniques by Molino et al. [2004] (who also use quadratic potentials).

Some existing work in the machine learning community is tangentially related to our own work, but we are unaware of any papers that seek to address the same problems. Jeong et al. [2003] uses a neural network to create a simplified mesh directly from an unorganized point cloud. Barhak and Fischer [2001] uses a neural network Self Organizing Map to achieve the same goal. And Saul and Roweis [2003] approach the more general problem of fitting low dimensional manifolds to high dimensional data. In contrast, our method is specifically designed to work on existing manifolds (i.e., surfaces) that have been found by other means.

The ‘classical’ paper on mesh decimation is Garland and Heckbert [1997], whose QSlim algorithm has been used widely. More recently, surface decimation has been studied in Alliez et al. [2003] and Cohen-Steiner et al. [2004], again using non-Bayesian techniques. These works apply all to static meshes only; here we extended it to decimation with the specific purpose of animating the decimated meshes. Other approaches have been taken to the related problem of compressing animated 3D models. Alexa and Müller [2000] accomplishes this goal by performing principal components analysis on an animated sequence. Briceno et al. [2003] uses a video in which each frame is a geometry image, a novel and useful geometric representation [Gu et al. 2002]. Both of these methods are effective at reducing

the storage requirements but they do not reduce the total number of primitives that need to be rendered by the graphics hardware, so the cost of rendering an animated sequence remains essentially unchanged.

A number of these techniques are compared to our work in later sections of this paper.

3. BAYESIAN MODEL

This section describes the basic Bayesian model used for surface reconstruction, and introduces the resulting optimization problem. The model combines a probabilistic noise model of the data acquisition device with a probabilistic prior on typical surfaces.

3.1 Surface Posterior

Let x be the true surface of the object being modeled. Instead of x , our sensors perceives a noisy measurement of the surface. This measurement could be a triangular mesh or a depth image; both are covered by our analysis.

Let us denote the measurement by z . In probabilistic terms, we seek to reconstruct the most likely surface x given the measurement z . Bayes rule enables us to “invert” this probability:

$$p(x|z) = \eta p(z|x) p(x). \quad (1)$$

Here $\eta = 1/p(z)$ is a normalization constant that is independent of x ; $p(z|x)$ is a probabilistic model of the measurement formation process, and $p(x)$ is a prior probability distribution over surfaces in the world.

The process of *most probable surface reconstruction* is simply the process of finding the surface x that maximizes the posterior probability in Eq. 1. We adopt the standard technique of optimizing the negative logarithm of the likelihood. The result is equivalent, since the logarithm is a monotonic function:

$$\hat{x} = \underset{x}{\operatorname{argmin}} \underbrace{-\log p(z|x)}_{:= \Phi(x,z)} - \underbrace{\log p(x)}_{:= \Psi(x)}. \quad (2)$$

Thus, the argument to be minimized decomposes into two additive terms, one for the measurement formation process denoted Φ , and one for the prior denoted Ψ . These different terms will now be discussed in more detail.

3.2 Probabilistic Measurement Formation

The model of measurement formation (in logarithmic form) is given by $\Phi(x, z)$. We denote the ‘true’ surface by x . The actual measurement z is a sample of this surface, called the *measurement mesh*. The measurement mesh is a collection of 3-D points, denoted $z = \{z_i\}$. (It may also contain edges or faces, but this is irrelevant for the analysis here).

We assume that the sensor noise follows a Gaussian distribution. Hence the probability for each individual measurement point $z_i \in \mathcal{R}^3$ given the true surface x is given by the following Gaussian:

$$p(z_i|x) = |2\pi\Sigma_i|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_i - x_k)^T \Sigma_i^{-1} (z_i - x_k)\right\}. \quad (3)$$

Here x_k is the point on the surface closest to z_i , according to the Mahalanobis distance defined through Σ_i^{-1} .

The covariance of the measurement noise is Σ_i . It enables us to express knowledge of the noise characteristics of our sensor. Even though any covariance can be used here, we generally model sensor noise by an elongated covariance of the type

$$\Sigma = R^T \underbrace{\begin{pmatrix} \sigma_{\text{large}}^2 & 0 & 0 \\ 0 & \sigma_{\text{small}}^2 & 0 \\ 0 & 0 & \sigma_{\text{small}}^2 \end{pmatrix}}_{(*)} R. \quad (4)$$

Here $\sigma_{\text{large}} > \sigma_{\text{small}} > 0$. The matrix labeled $(*)$ permits for large variation in the x -axis. R is a rotation matrix, which enables us to transform the Gaussian into any orientation.

The oriented Gaussian enables us to model the sensor noise of the imaging device. In a stereo range image, for example, R is the orientation of the ray that corresponds to the respective pixels in the reference image (e.g., the left image). The resulting noise covariance is then elongated along the direction of the measurement ray. This is plausible because within a camera image, the actual pixel locations are usually accurate. Depth, however, is determined through disparity, which is a process characterized by high error. Hence the measurement model assumes higher noise along the direction of the light beam along which a surface was imaged.

In a general mesh of unknown origin, we chose the values of σ_{large}^2 and σ_{small}^2 to be closer to each other, so that the Gaussian is not as elongated. The Gaussian is then oriented towards the mean surface normal of the adjacent faces.

The function $\Phi(x, z)$ Eq. 2 is now obtained as the negative logarithm of the joint probability over all measurements $z = \{z_i\}$. This joint probability is given by

$$p(z | x) = \prod_i p(z_i | x). \quad (5)$$

This distribution can be thought of as a high-dimensional multivariate Gaussian, whose dimensionality equals the number of measurement points z_i times three; however, it is not an arbitrary Gaussian, in that its covariance is block-diagonal.

We now have

$$-\log p(z | x) = \text{const.} + \sum_i (z_i - x_i)^T \Omega_i (z_i - x_i). \quad (6)$$

Here we wrote $\Omega_i = \frac{1}{2}\Sigma_i^{-1}$ for convenience. The constant term in Eq. 6 is the logarithm of the normalizer $\|2\pi\Sigma_i\|^{-\frac{1}{2}}$ in Eq. 3. However, this normalizer does not depend on the target variables x . Hence it plays no role in the overall optimization, and it can safely be omitted. This consideration now leads us to the final definition of the function Φ :

$$\Phi(x, z) = \sum_i (z_i - x_i)^T \Omega_i (z_i - x_i). \quad (7)$$

We note that Φ is a weighted quadratic distance (Mahalanobis distance) between the measurements z_i and the actual surface points x_i . The matrix Ω_i is positive definite. Consequently, $\Phi(x, z)$ is minimal for $z_i = x_i$. Or put differently, maximizing Φ leads to an unaltered clone of the measured mesh z .

3.3 Surface Prior

The (log-)surface prior is encapsulated in the function $\Psi(x)$. The importance of a good prior cannot be overemphasized. We just noted that without a prior, the most probable surface reconstruction would be the measurement itself. Thus, any smoothing or feature enhancing is the result of applying the prior distribution.

Remarkably, we can get the desired effects with a simple ‘weak’ smoothness prior. The smoothness of a surface is a function of its curvature. Our approach assesses the curvature through the relation of adjacent surface normals. Let $x_i, x_{i'}, x_{i''}$ be adjacent nodes in the mesh that define a triangular face. The normal of this face is given by

$$n_k = \frac{(x_{i'} - x_i) \times (x_{i''} - x_i)}{\|(x_{i'} - x_i) \times (x_{i''} - x_i)\|}. \quad (8)$$

Let n_k and n_j be two adjacent normals in the surface mesh. Then the surface prior function Ψ is of the form

$$\Psi(x) = \sum_{k,j} f(n_k - n_j), \quad (9)$$

where each $f(n_k - n_j) \geq 0$ is a *potential* that is zero if $n_k = n_j$, and positive otherwise.

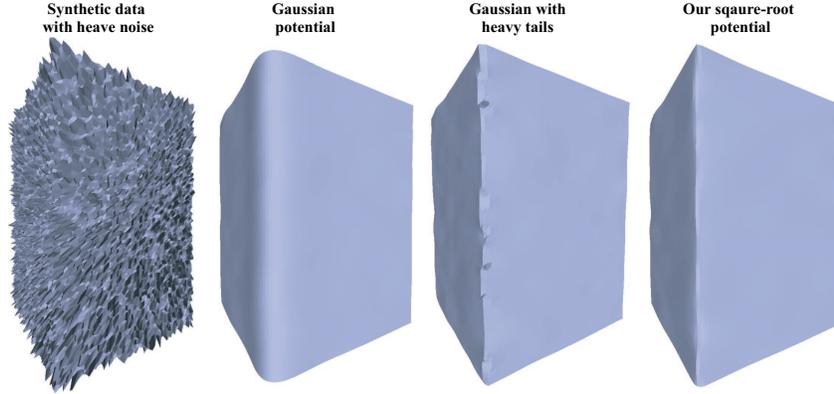


Fig. 1. Investigation of different potential functions f , for a synthetic data set with heavy noise levels.

3.4 Square-Root Priors

The actual choice of a prior is essential for feature enhancement. In prior work, [Szeliski and Tonnesen 1992] developed *quadratic potentials* of the form $f_{\text{sq}}(n_k - n_j) = \|n_k - n_j\|_2^2$. Such potentials arise from Gaussian probability distributions and we will freely refer to them as both quadratic and Gaussian potentials. Unfortunately, the quadratic function tends to over-smooth the edges and remove fine features from the mesh. This is illustrated in Figure 1, which compares the effect of those potentials. This figure also depicts the result of a quadratic function with heavy tails, which is commonly used in the field of robust statistics.

In computer vision, a recent empirical study identified that natural image priors are sub-linear, not quadratic [Levin et al. 2002]. Here we propose the use of the square-root function (Euclidean distance in normal space):

$$f_{\text{sqrt}}(n_k - n_j) = \|n_k - n_j\|_2 = \sqrt{(n_k - n_j)^T (n_k - n_j)}. \quad (10)$$

As shall be shown, the square-root potential smooths the mesh while actually *enhancing* features such as edges. The enhancement effect is due to the fact $f_{\text{sqrt}}(n_k - n_j)$ grows sub-linearly in its arguments, hence it tends *not* to distribute curvature over multiple vertices, as would a quadratic potential. With the exception of certain special-purpose learned potentials discussed below, we use the square-root potential throughout this work. Figure 1 shows the result of this prior: The edge is not only retained; it is enhanced.

4. FINDING PROBABLE RECONSTRUCTIONS

As stated in Eq. 2, the probable reconstruction of the actual surface is obtained through the minimization of $\Phi(x, z) + \Psi(x)$:

$$\hat{x} = \underset{x}{\operatorname{argmin}} \left[\sum_i (z_i - x_i)^T \Omega_i (z_i - x_i) \right] + \left[\sum_{k,j} f_{\text{sqrt}}(n_k - n_j) \right]. \quad (11)$$

This minimization requires us to adjust the location of the vertices x in the reconstructed mesh.

This problem is a *sparse energy minimization optimization problem*, for which a rich family of efficient algorithms exists. Thus, through our Bayesian formulation, we can now leverage existing optimization algorithms to perform the minimization. This is in stark contrast to prior work on *diffusion techniques*, which specify a specific algorithm for mesh smoothing, instead of explicitly stating the objective.

Descriptions of the conjugate gradient (CG) algorithm can be found in contemporary textbooks on optimiza-

tion [Press 1988]. We use the Polak-Ribiere formulation [Krzitek et al. 2004] of the conjugate gradient technique for non-linear functions to minimize this function with respect to the state vector x , which we shall now describe.

4.1 Conjugate Gradient Implementation

For notational brevity, we write $J(x) = \Phi(x, z) + \Psi(x)$. CG initializes the mesh $x^{[0]}$ with the measured mesh z . It then iteratively alters the vertex locations in x by following the conjugate gradient search direction, which linearly combines the negative gradient of the function $J(x)$ with previous such gradients. Specifically, we have for each vertex x_k in the mesh x :

$$\frac{\partial J(x)}{\partial x_k} = \frac{\partial \Phi(x, z)}{\partial x_k} + \frac{\partial \Psi(x)}{\partial x_k}. \quad (12)$$

The gradient of the log-measurement constraint resolves to

$$\begin{aligned} \frac{\partial \Phi(x, z)}{\partial x_k} &\stackrel{\text{Eq. 7}}{=} \sum_i \frac{\partial (z_i - x_i)^T \Omega (z_i - x_i)}{\partial x_k} \\ &= -2 \Omega (z_k - x_k). \end{aligned} \quad (13)$$

The gradient of the log-prior is given by

$$\begin{aligned} \frac{\partial \Psi(x, z)}{\partial x_k} &\stackrel{\text{Eq. 9}}{=} \sum_{i,j} \frac{\partial f(n_i - n_j)}{\partial x_k} \\ &= \sum_{i,j} \frac{\partial f(n_i - n_j)}{\partial (n_i - n_j)} \left[\frac{\partial n_i}{\partial x_k} - \frac{\partial n_j}{\partial x_k} \right]. \end{aligned} \quad (14)$$

For $f = f_{\text{sqrt}}$, this derivative resolves to

$$\frac{\partial f_{\text{sqrt}}(n_i - n_j)}{\partial (n_i - n_j)} = (f_{\text{sqrt}}(n_i - n_j))^{-1} (n_i - n_j). \quad (15)$$

Lastly, we need to calculate the derivatives of the type $\frac{\partial n_i}{\partial x_k}$ in Eq. 14. Those derivatives are only non-zero if x_k is a vertex of the surface associated with n_i . Let such a surface be defined through three nodes, x_k , $x_{k'}$, and $x_{k''}$ (c.f. Eq. 8). Then we have

$$\frac{\partial n_i}{\partial x_k} = \frac{I - n_i n_i^T}{|(x_{k'} - x_k) \times (x_{k''} - x_k)|} (x_{k''} - x_{k'}) \times x_k. \quad (16)$$

The gradient $\frac{\partial J(x)}{\partial x_k}$ is now fully defined, and the actual implementation of the gradient should now be straightforward. The details of the CG algorithm are omitted for brevity, but can be found in contemporary texts. Our implementation uses a Newton-Raphson line search step for determining the step size of the update. The resulting algorithm for probable surface reconstructions is now remarkably simple: Simply set $x^{[0]} = z$ and iterate the CG update rule. The result is a probable reconstruction of the object surface.

4.2 Results

Figure 2 shows results for three different data sets collected by an active space-time stereo vision system [Davis et al. 2005]. The most important aspect of these results pertains to the fact that sharp edges are retained (or even enhanced), while the remaining surface is smoothed in the Bayesian reconstruction process. Note that even small features such as the lip of the cup (center), which spans only a few triangles, are faithfully reconstructed. All of these models, which contain tens of thousands of triangles each, require less than one minute on a 3-GHz Pentium for the conjugate gradient algorithm to converge. Good results can be obtained in approximately 10 seconds if the convergence thresholds are selected liberally.

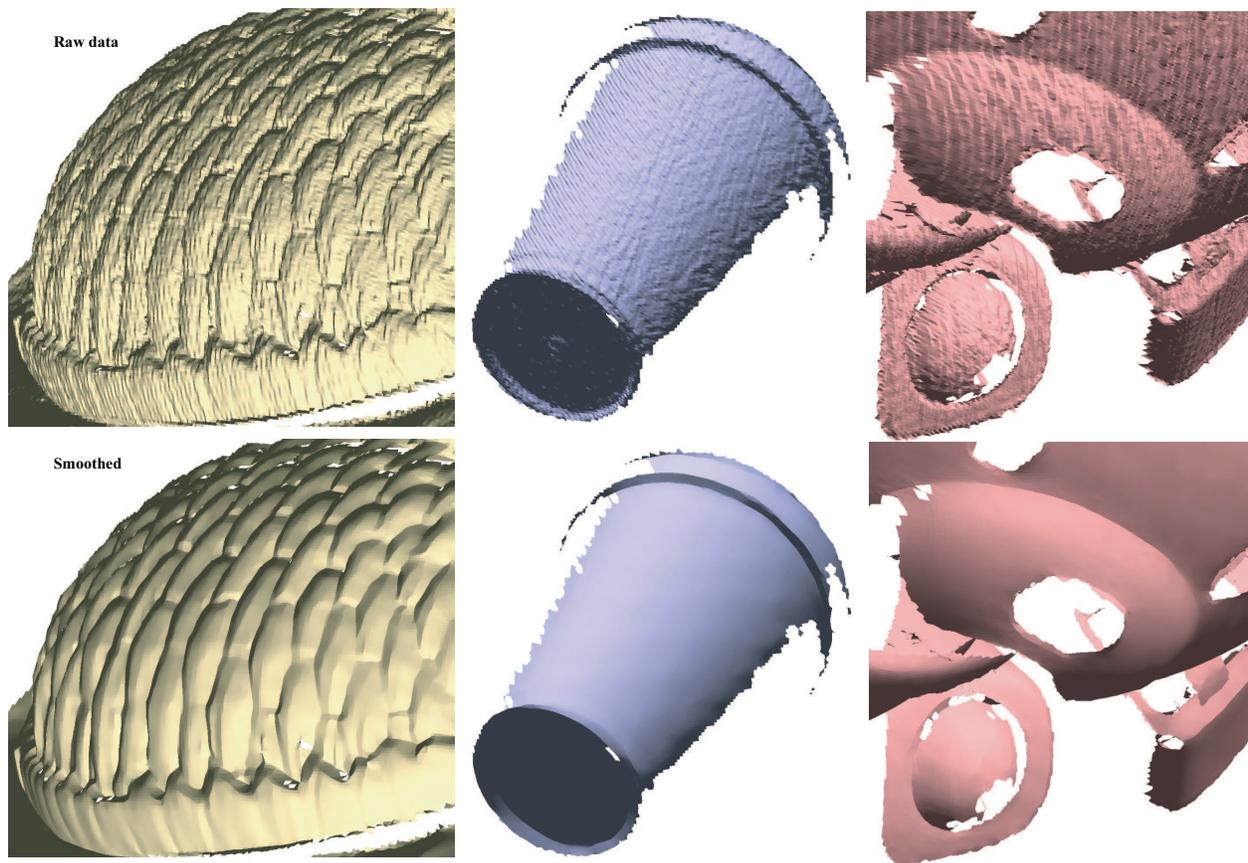


Fig. 2. Examples of Bayesian smoothing on several real datasets collected with a space-time stereo vision system. Notice how the method smooths the surface while enhancing the features such as edges.

Next we compare these results to the existing literature. Figure 3 compares the Bayesian reconstruction to the published results of Jones et al. [2003]. Our algorithm compares favorably with these results. For the dragon model their algorithm fails to smooth out areas where high natural curvature exists in the underlying model. This is illustrated in the detailed views. Also note the leveling lines in the view of the dragon's tooth. Here you can see that their algorithm shrinks the tooth while failing to eliminate all the noise. The Venus model is less stark of a contrast. Both algorithms perform well on this model, though careful examination reveals that the Bayesian reconstruction is slightly sharper and less blurred around areas of high-curvature, such as the chip in her chin. Processing times for these models, both of which contain about 100k vertices, were less than two minutes on a 3-GHz Pentium, compared to 54 seconds (Venus) and 80 seconds (dragon) for Jones et al. on a 1.4-GHz Athlon.

Figure 4 compares our algorithm to the work of Fleishman et al. [2003]. Here we consider two models that contain sharp corners. The first is the Fandisk model. Just as with Jones et al. the technique of Fleishman et al. fails in areas of high curvature. In particular, notches can be seen in the sharp edges of the model. This is a natural consequence of the bilateral filtering technique, which filters less in areas of high curvature. The results of the Bayesian algorithm did not exhibit any such failures. The pyramid model is even more striking. A limitation of our algorithm can be seen in this result, however: the square-root prior does allow for some slight blurring on the edge of the pyramid. This defect

will be addressed in the next section, which deals with learning multi-modal surface priors.

Finally, in Figure 5 we compare our results to the isotropic fairing algorithm of Desbrun et al. [1999], which is a baseline algorithm for mesh smoothing. The subject here is the scanned face of a mannequin. The advantages of the non-linear prior are clear.

In all cases of published results for which we were able to obtain example data, we find our algorithm to perform better or equal. Generally speaking, the algorithm scales with the number of vertices in the mesh, although memory consumption has been an issue for models with greater than one million vertices.

5. SUPERVISED LEARNING OF THE SHAPE POTENTIALS

5.1 Parameterizing the Prior

As noted, the smoothing is a direct result of applying the shape prior $p(x)$, as manifest through the pairwise normal potentials f . This makes it possible to tailor the potentials to the specific types of shapes commonly encountered in the application.

For example, when imaging man-made objects, one may find a dominant use of edges describing right angles. Thus, a potential f favoring right angles (along with smoothness) sometimes works better in such situations.

A nice feature of the Bayesian approach is that we can *learn* the potential function from data, through a *supervised learning technique*. In supervised learning, we are given the true shape y along with its measurement z . The true shape y is easily obtained for simple man-made objects, such as card boxes of known size (e.g., a cereal box), or cylinders with known diameter and height (e.g., a soup can). It is also obtainable for objects manufactured through FDM machines.

To learn such an application-specific prior, we need a potential f with free parameters. Our method implements f via a *cubic spline* over the difference $\|n_k - n_j\|_2$ of any two adjacent surface normals. This spline is defined over N points, spaced equally in the interval $[0, 2]$. Thus, this spline has exactly N parameters, denoted $\theta = \{\theta_1, \dots, \theta_N\}$.

5.2 Learning the Prior Parameters

The method uses another probabilistic technique to identify the correct potential. This methodology is applicable to any application, not just applications with frequent right-angled edges. Specifically, let

$$\theta^* = \operatorname{argmax}_{\theta} p(\hat{x}_{z,\theta} | y)p(\theta), \quad (17)$$

where $\hat{x}_{z,\theta}$ is the most probable surface reconstruction corresponding to the measurement vector z and the prior parameter set θ . We define the expression $p(\hat{x}_{z,\theta} | y)$ implicitly by defining its log-likelihood potential function

$$\Gamma(\hat{x}_{z,\theta}, y) := -\log(p(\hat{x}_{z,\theta} | y)) = (\hat{x}_{z,\theta} - y)^T (\hat{x}_{z,\theta} - y). \quad (18)$$

Likewise, we define the regularization potential corresponding to $p(\theta)$ by

$$\Upsilon(\theta) := -\log(p(\theta)) = \sum_{j=2}^{N-1} w(\theta_{j-1} - 2\theta_j + \theta_{j+1})^2. \quad (19)$$

This is a second order curvature penalty, weighted by w , that is designed to prevent over-sensitivity of the prior estimate parameter set to the specific training data. In this work we have manually selected w to be one. We have also clamped the boundary points of the spline to match the value and derivative of the square-root potential. In order to avoid the singularity of that function at the origin we have added a small offset of 10^{-4} to the argument of the square root.

The maximum of the probability distribution corresponds to the minimum of the potential function, hence

$$\theta^* = \operatorname{argmin}_{\theta} \Gamma(\hat{x}_{z,\theta}, y) + \Upsilon(\theta). \quad (20)$$

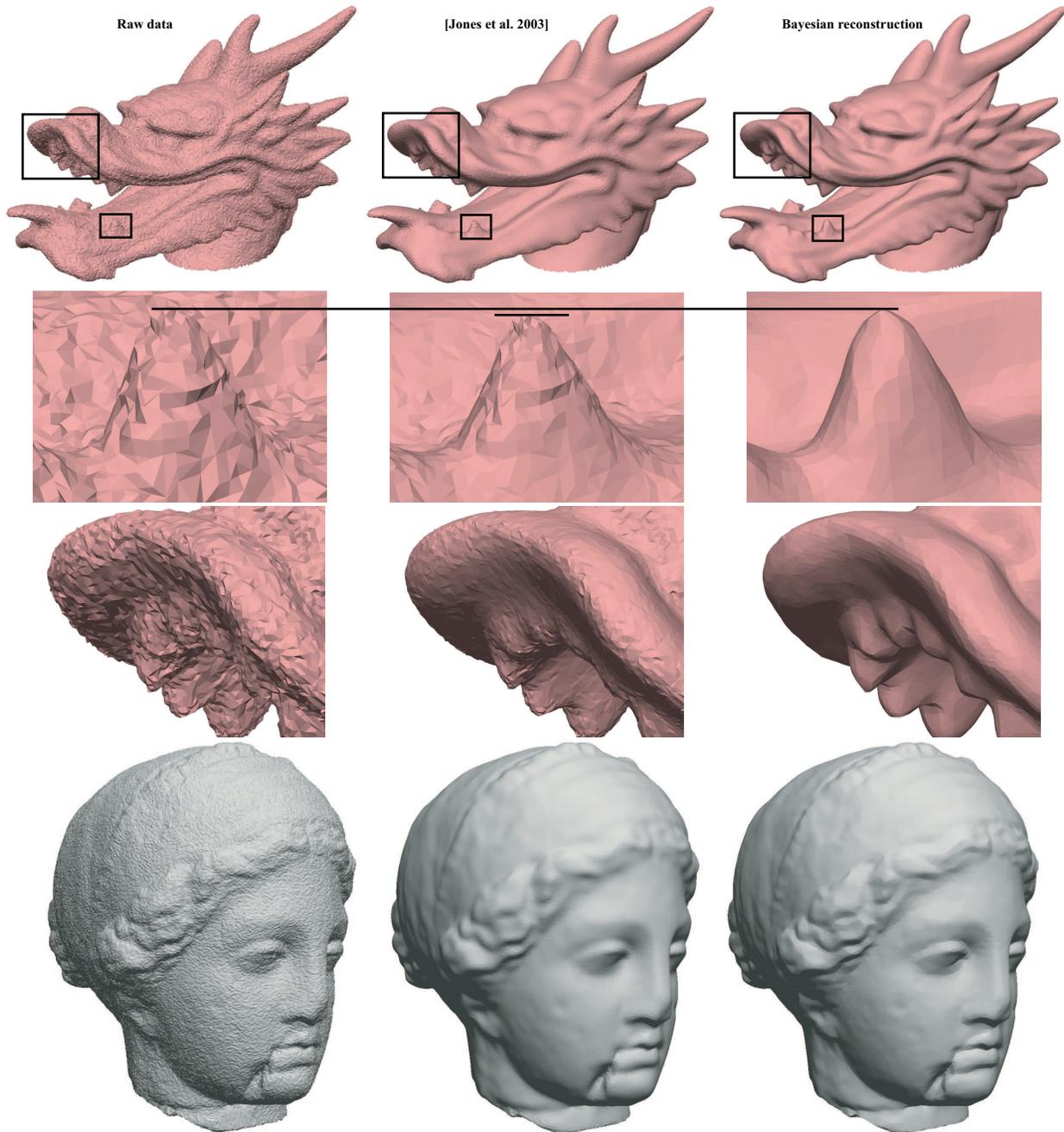


Fig. 3. A comparison to the published results of Jones et al. [2003] Notice the detail view of the tooth, which is shrunken despite remaining rough in the work of Jones et al.

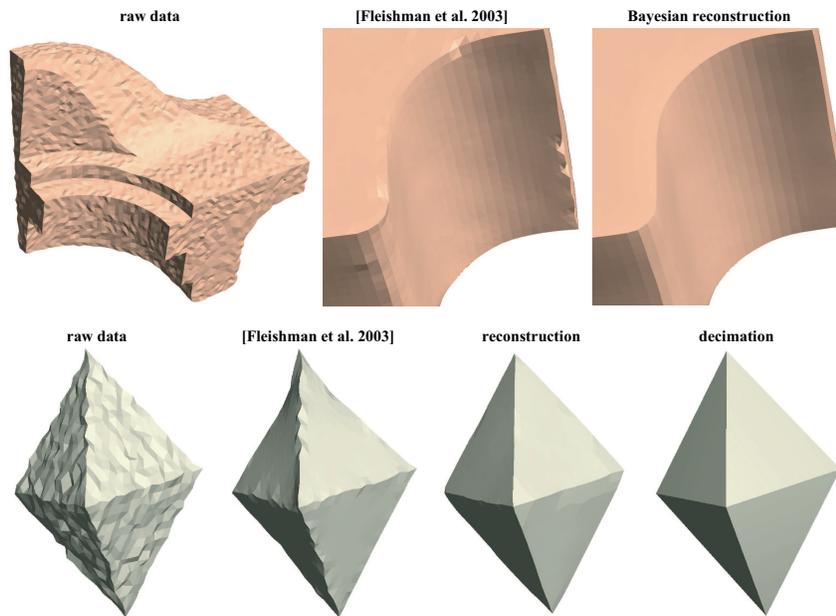


Fig. 4. A comparison to the published results of Fleishman et al. [2003]

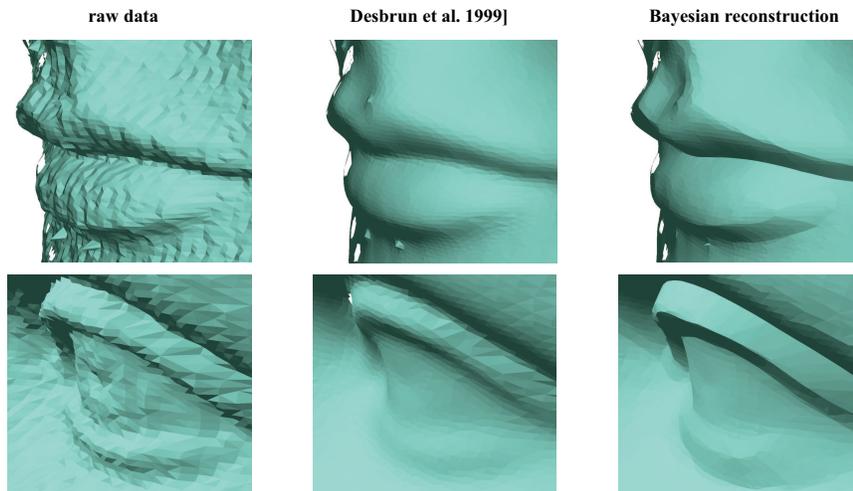


Fig. 5. A comparison to the published results of Desbrun et al. [1999]

In short, θ^* is the parameter vector that describes the prior estimate potential whose most probable surface reconstruction best matches the given ground truth without over-fitting the data.

Once again, this minimization can be carried out using any of the standard optimization techniques and since this is done only once for each application domain, computational efficiency is less important than in the surface

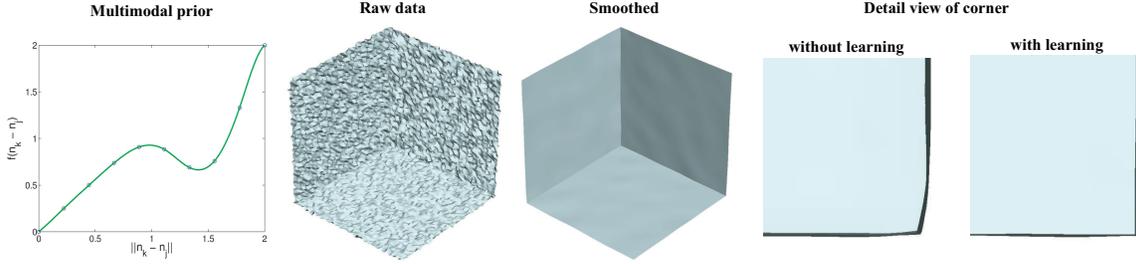


Fig. 6. The potential specialized for right angles better reconstructs this artificially-constructed corner.

reconstruction using this learned prior. Nevertheless the computation of the most probable surface reconstruction $\hat{x}_{z,\theta}$ for a given parameter set θ is itself expensive, so it is desirable to find a way to interleave the two optimizations in this learning process.

To effect this goal we modify the conjugate gradient algorithm to include the modification of the parameters θ at each iteration. In particular, for each iteration n we have the current best estimate of the prior estimate parameter set $\theta^{[n]}$ and the intermediate state vector $x_{z,\theta^{[n]}}^{[n]}$. We compute by finite difference

$$\frac{\partial \Gamma(x_{z,\theta^{[n]}}^{[n]}, y)}{\partial \theta_j} \approx h_j^{[n]} = \frac{\Gamma(x_{z,\theta^{[n]}+\Delta\theta_j}^{[n+1]}, y) - \Gamma(x_{z,\theta^{[n]}}^{[n]}, y)}{\varepsilon}, \quad (21)$$

where $\Delta\theta_j$ is the small increment ε applied to the j^{th} parameter of θ . That is, for each of the N θ_j 's we replacing θ_j by $\theta_j + \varepsilon$ and perform a conjugate gradient update, reverting the state vector after each. From this we obtain the gradient we require to update the parameters of θ . In this case we apply a hill-climbing technique with a manually-selected step size, as follows:

$$\theta_j^{[n+1]} = \theta_j^{[n]} - \alpha \left(h_j^{[n]} + \frac{\partial \Upsilon(\theta^{[n]})}{\partial \theta_j^{[n]}} \right). \quad (22)$$

The partial of $\Upsilon(\theta^{[n]})$ with respect to $\theta_j^{[n]}$ may be computed analytically directly from Eq. 19. The process terminates with the standard CG convergence tests. The resulting set of parameters θ^* may not be optimal, in that the optimization may get stuck in a local minimum. This is the universal problem of non-convex optimizations and only extensive experience can indicate whether this is a substantive or merely a theoretical shortcoming of this approach. So far we do not have enough experience to comment.

Figure 6 shows the results of a prior specialized to deal with many right angles and flat surfaces. The learned potential shown in Figure 6 is multi-modal: It possesses a mode for parallel normals, and a second one for orthogonal normals. This leads to superior reconstructions of right angles, as shown in Figure 6.

6. BAYESIAN SURFACE DECIMATION

We now apply the Bayesian method to the problem of *surface decimation*. This problem pertains to the identification of a sparse mesh that, when rendered, is close in appearance to the dense mesh from which it was derived.

At first glance, the surface decimation problem differs from the surface reconstruction problem in two significant ways. First, surface decimation is a *discrete* problem: either two surface patches are merged into one, or they are not. Second, the decimation problem is not a reconstruction problem. The “sensor” is not an imaging sensor. Instead,

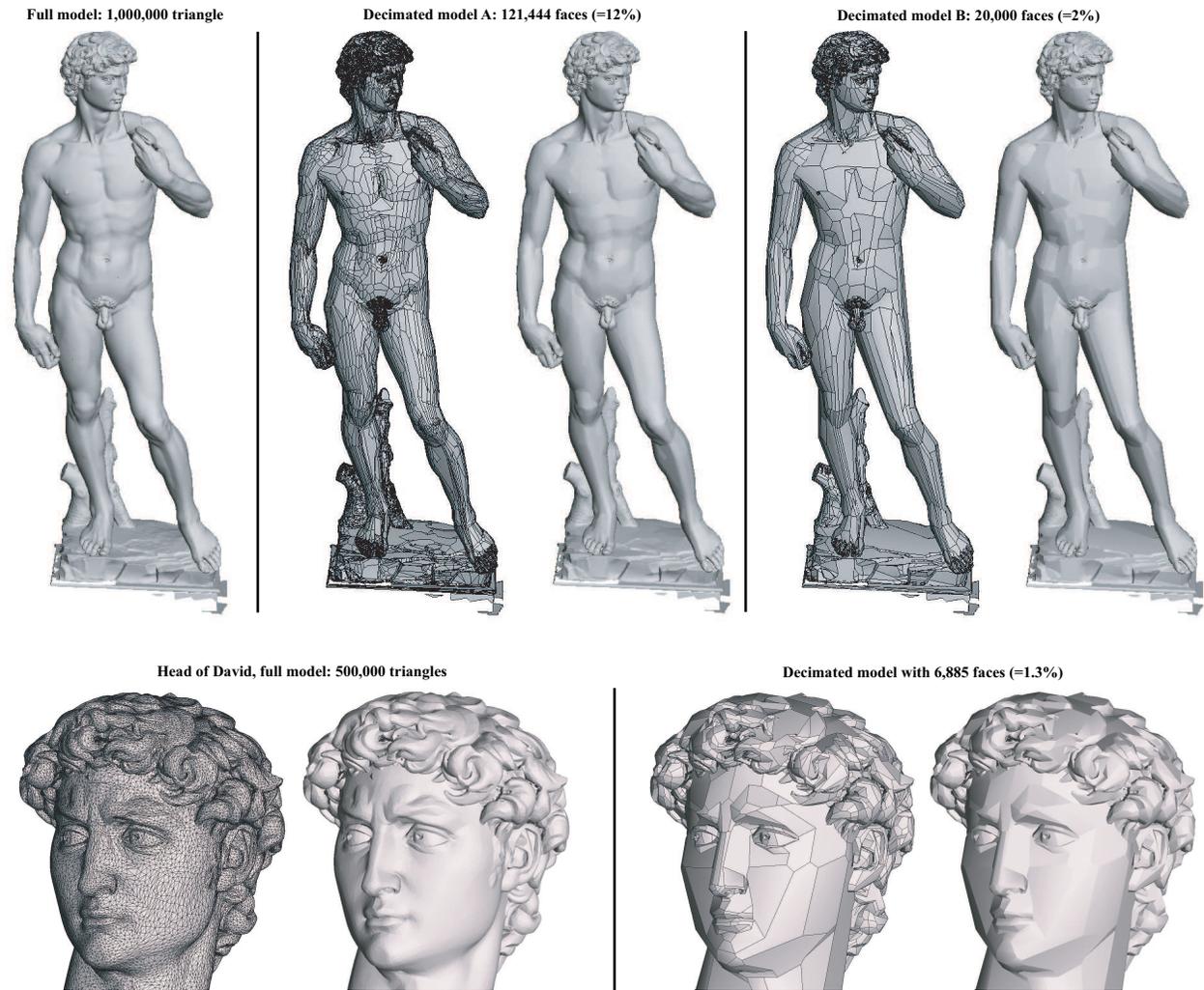


Fig. 7. The David model shown at various levels of decimation.

it is the rendering algorithm that generates a 2-D view of the 3-D surface mesh. This leads to a redefinition of the basic ingredients of the probabilistic model, and the meaning of the variables z and x . It also necessitates the use of discrete optimization techniques instead of the conjugate gradient algorithm. Nevertheless, we find that with minimal adaptation of the basic paradigm, the Bayesian approach is applicable to the surface decimation problem.

6.1 Mathematical Model

The decimation model parallels the smoothing model. In the surface decimation problem, z is a detailed surface description (dense mesh), and x shall be its low-dimensional approximation. We begin with Eq.2:

$$\hat{x} = \underset{x}{\operatorname{argmin}} \Phi(x, z) + \Psi(x). \quad (23)$$

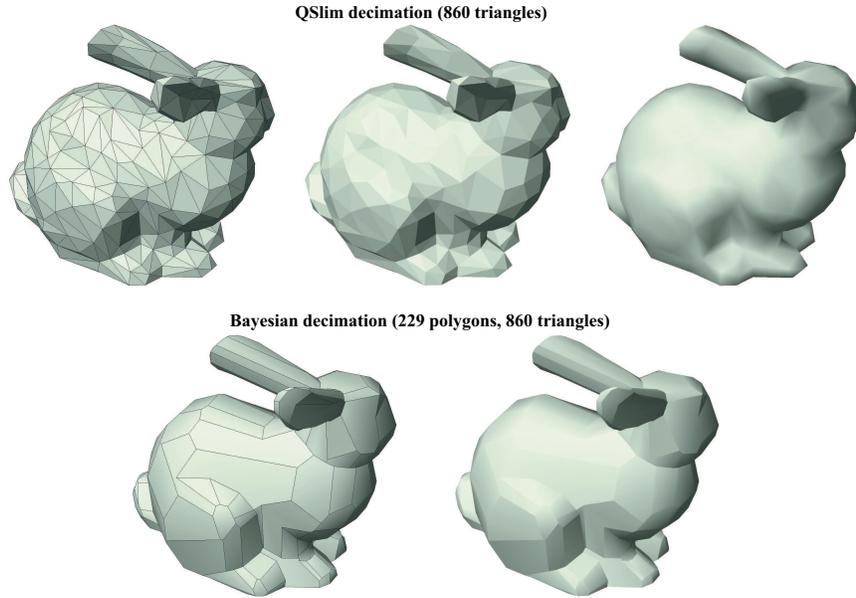


Fig. 8. Comparison of QSlim and the Bayesian decimation. Our method generates polygons, whereas QSlim generates triangles. However, when re-meshing our solution using triangles, both decimations are of equal complexity (860 triangles). The right-most QSlim rendering uses vertex-averaged normal vectors.

As before, $\Phi(x, z)$ is a term relating the dense mesh z and the coarse mesh x . We shall call the underlying function $p(z | x)$ the *appearance model*. $\Psi(x)$ is a term that corresponds to the prior $p(x)$.

We begin with the description of the potential $\Phi(x, z)$. It shall prove useful to think of a decimated mesh x as a mesh in which each individual surface patch subsumes one or more triangles in the dense mesh z . Hence it makes sense to define a projective *decimation function*. We write

$$k = g(i) \quad (24)$$

to indicate that the i -th triangle in the dense mesh z is projected into the k -th surface patch of the decimated mesh x . In the interest of concise notation, we will silently assume this equality throughout the rest of this section.

The *appearance probability* $p(z | x)$ is now given by a Gaussian over the triangle normals in the dense mesh z and the corresponding surface patch normals in the decimated mesh x . To make the two different sets of normals explicit, we denote normals in the dense mesh z by u_i ; and normals in the decimated mesh x by n_k . For each corresponding normals in the original and decimated mesh, the measurement model is defined via the probability

$$p(u_i | x) = |2\pi\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (u_i - n_k)^T \Sigma^{-1} (u_i - n_k) \right\}. \quad (25)$$

The matrix Σ is the familiar covariance which in our implementation is simply a 3-by-3 identity matrix.

The motivation to compare normals in the decimation problem is a direct result of the fact that we seek an approximation which mimics the original in appearance when rendered. Rendering uses the surface normal to determine the shading. Hence the difference in normals accounts for the difference in appearance.

The model in Eq. 25 is now developed into a log-likelihood over all surface patches, analogously to Eqs. 5 and 6

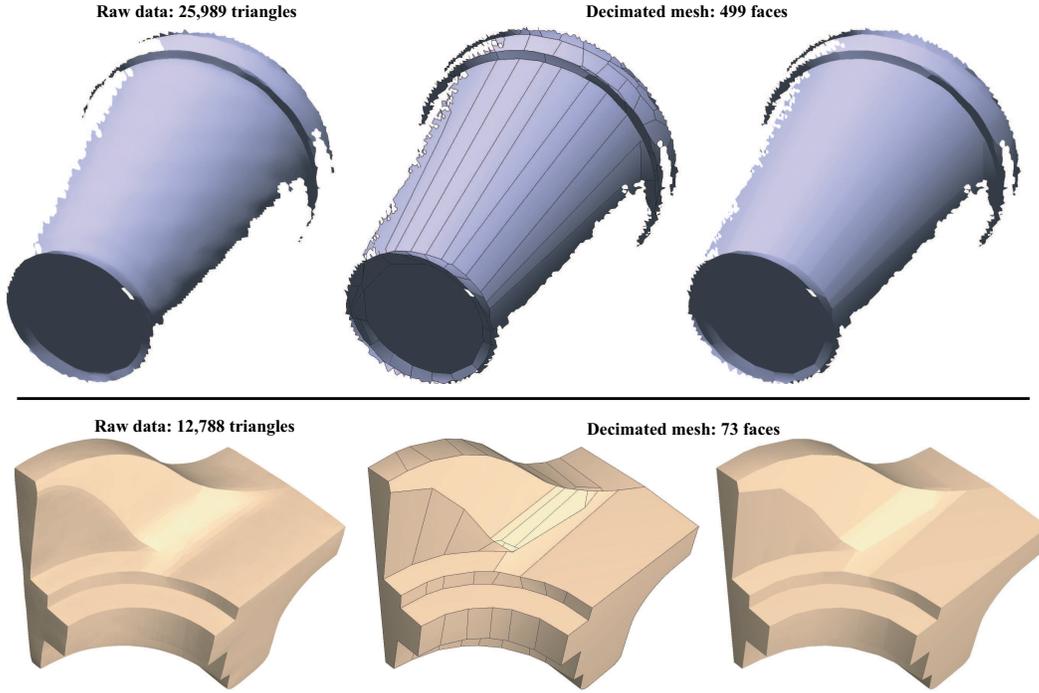


Fig. 9. Two examples of our decimation results.

above. This gives us

$$p(z | x) = \prod_i p(u_i | x) \quad (26)$$

and hence

$$-\log p(z | x) = \text{const.} + \sum_i (u_i - n_k)^T \Omega (u_i - n_k) \quad (27)$$

with $\Omega = \frac{1}{2}\Sigma^{-1}$. This expression leads to the final objective Φ :

$$\Phi(x, z) = \sum_i (u_i - n_k)^T \Omega (u_i - n_k). \quad (28)$$

Clearly, without a prior the function $\Phi(x, z)$ attains its optimum at $x = z$, which is the original mesh (there might be other solutions as well). To see, we simply note $\Phi(x, z) = 0$ for $x = z$; and this is indeed the global minimum since $\Phi(x, z) \geq 0$ for any x . Thus, once again we need a prior to achieve the desired effect.

The prior in surface decimation pertains to the number of surface patches in the decimated model. In particular, we seek a prior that ‘penalizes’ complex over more simple decimations. Possibly the most simple such prior is an exponential prior of the type

$$p(x) = \lambda \exp\{-\lambda|x|\}, \quad (29)$$

where $|x|$ is the *cardinality* of x , that is, the number of surface patches in the mesh x . The parameter λ is a user-specified parameter which controls the strength of our desire to decimate the surface (one might derive λ from an economical

perspective of considering the computational costs of rendering individual surface patches; but we simply set it by hand).

The negative logarithm of $p(x)$ is given by

$$-\log p(x) = -\log \lambda + \lambda |x| = \text{const.} + \lambda |x|. \quad (30)$$

Hence we define $\Psi(x) = \lambda |x|$. In other words, under our exponential prior each surface patch incurs a constant penalty in the decimated mesh x .

The problem of finding the optimal decimation is now the problem of minimizing the number of surface patches in x while faithfully reproducing the surface normals in the original mesh z . Mathematically, this is achieved by the following optimization:

$$\begin{aligned} \hat{x} &\stackrel{\text{Eq. 23}}{=} \underset{x}{\operatorname{argmin}} \Phi(x, z) + \Psi(x) \\ &= \underset{x}{\operatorname{argmin}} \lambda |x| + \sum_i (u_i - n_k)^T \Omega (u_i - n_k). \end{aligned} \quad (31)$$

For a decimation to be valid, we also require that each x_k represents a *contiguous region* in the dense mesh z . Put differently, let $\delta(k) := \{i : g(i) = k\}$ be the set of surface patches z_i in the mesh z which are mapped to the surface patch k in the mesh x . This set must consist of a single connected component. This constraint is enforced in the search below.

This approach is notably different than the decimation algorithm used by Cohen-Steiner et al. [2004]. Here we specify an allowable threshold of normal-vector variance within a given surface patch rather than specifying the number of desired output patches. Thus we can use the same threshold for both simple and complex meshes and the number of output patches matches the intrinsic complexity of the input mesh. In contrast the k-means algorithm utilized by Cohen-Steiner et al. [2004] requires the user to guess how many output patches will be needed to achieve a particular level of fidelity to the original mesh.

6.2 Finding Plausible Decimations

The optimization problem now pertains to the problem of finding an optimal decimation g , an optimal set of surface normals n_k for each surface patch in x_k , and an optimal set of vertices in the decimated mesh. As before, our method pursues hill climbing to carry out this optimization.

First, we note that each choice of g uniquely determines the surface normals of each surface patch in x_k . To see, let us consider the set $\delta(k)$. Without loss of generality, we can assume $\delta(k)$ contains at least one element—otherwise we could simply remove the surface patch x_k from the decimated mesh x and increase the score Ψ by λ without incurring any loss.

The optimal surface normal n_k is now obtained by setting the derivative of $\Phi(x, z)$ to zero (the term $\Psi(x)$ does not depend on such a normal):

$$\frac{\partial \Phi(x, z)}{\partial n_k} \stackrel{!}{=} 0. \quad (32)$$

The only contributing normals u_i are those whose δ -value is k (c.f., Eq. 28). Hence we have

$$\begin{aligned} \frac{\partial \Phi(x, z)}{\partial n_k} &= \frac{1}{\partial n_k} \partial \sum_{i \in \delta(k)} (u_i - n_k)^T \Omega (u_i - n_k) \\ &= -2 \sum_{i \in \delta(k)} \Omega (u_i - n_k). \end{aligned} \quad (33)$$

Setting this expression to zero resolves to

$$n_k = \frac{1}{|\delta(k)|} \sum_{i \in \delta(k)} u_i. \quad (34)$$

Here, $|\delta(k)|$ indicates the number of elements in $\delta(k)$. Thus, for any g our method derives the normals of the decimated mesh x (but it does not yet define the actual vertices of the decimated surface). This enables our method to decimate the surface using a hill climbing algorithm in the space of possible functions g .

Our algorithm initially assumes $g(i) = i$ (and hence $x = z$), that is, the initial mesh x is simply a clone of z . The function g is then refined through three hill climbing steps, whose execution is interleaved:

- Fuse.** Our algorithm tests if fusing two adjacent surface patches in the decimated mesh x increase the score. A fusion between the k -th and the j -th surface patches in x implies that we set $g(i) = k$ for any triangle i whose index was previously $g(i) = j$. The j -th surface patch is then safely removed from the mesh x .
- Split.** Our algorithm tests if removing a specific index i from a set $\delta(k)$ improves the score. This involves the creation of a new mesh in x , with a new index k' , and setting $g(i) = k'$.
- Swap.** Finally, our algorithm tests whether shifting a node on the boundary of two surface patches in the decimated mesh x from one to the other adjacent surface patch improves the score. This test involves a simple index change of the type $g(i) = j \rightarrow g(i) = k$.

Although unlikely, the last two steps may jeopardize the requirement that decimated regions are contiguous in z ; hence for each such step we simply check if this condition is still fulfilled. These operations are iterated until convergence.

The result of the hill climbing algorithm is a refined function g , which assumes a reduced set of surface patches in the mesh x . The vertices in x are recovered by a final post-processing step: Simply retain vertices from the original mesh z which, in x , border at least three different surface patches. Such vertices are form the edges in x . If only these vertices are retained, a subsequent triangulation yields the desired decimated mesh. In order to ensure that the output triangulation retains the manifold topology of the input mesh we perform the triangulation within each surface patch individually. Because the contiguity of a given patch is enforced by the decimation algorithm this precaution prevents the introduction of any new topological structures to the mesh.

A final processing step fits a linear shading gradient across each surface patch in the decimated mesh, by minimizing the quadratic distance in appearance space to all subsumed triangles in the original mesh z . To be more specific, we use a local 2D basis vector on the surface patch and fit a triple-valued linear function (representing the surface normal distribution over the patch) by least-squares to the surface normals of the triangles contained in that patch. For the purpose of the fit, each surface normal is placed at the geometric center its triangle. This allows for the pleasant shading effect that is used to render the examples given in this paper. This constitutes an advantage over triangle-based decimation algorithms such as Garland and Heckbert [1997], which do not have a notion of surface patches.

6.3 Decimation for Efficient Animation

The Bayesian approach is easily applied to decimation for mesh animations. To our best knowledge, past work on automatic decimation has only considered static meshes. When meshes are used in animation, the degree by which we seek to decimate is also a function of the deformations which a mesh undergoes during animation. Thus, static decimation techniques tend to produce inferior results when their results are being animated.

The application of the Bayesian approach to this problem is straightforward—with the caveat of a new alignment problem. Suppose we are given M versions of the mesh z in different configurations. Let us denote those meshes by z^1, z^2, \dots, z^M . Let us, for a moment, assume each such mesh possesses its own, decimated mesh. The decimated meshes will be denoted x^1, x^2, \dots, x^M .

The Bayesian formulation now simply states

$$\begin{aligned} p(x^1, \dots, x^M | z^1, \dots, z^M) &= \prod_j p(x^j | z^j) \\ &= \eta \prod_{j=1}^M p(z^j | x^j) p(x^j). \end{aligned} \quad (35)$$

This leads to the following additive decomposition

$$\hat{x} = \underset{x^1}{\operatorname{argmin}} \underset{x^2}{\operatorname{argmin}} \cdots \underset{x^M}{\operatorname{argmin}} \sum_j \Psi(x^j) + \Phi(x^j, z^j). \quad (36)$$

The *key* to the simultaneous decimation of all these meshes is to use an identical function g for *all* deformed meshes z^j and x^j . Thus for any k , the surface patches in $\delta(k)$ are all decimated into a single surface patch, for any of the meshes z^1, z^2, \dots, z^M .

The resulting optimization is analogous to the one just described. The only change pertains to the fact that the score is evaluated as a sum over all deformed meshes; instead of a single mesh only. Once a new function g is tested, the decimated meshes $\{x^j\}$ and their surface normals are describes as stated. The resulting score is then compared, using the sum over all meshes as stated in Eq. 35, instead of the single mesh score in Eq. 23.

We remark that this extension highlights one of the advantages of a Bayesian approach: By separating the Bayesian score from the optimization, it is straightforward to accommodate modifications, such the constraint of finding the optimal decimation relative to a collection of deformed meshes.

7. RESULTS

Decimation results are depicted in Figures 7, 8 and 9. Figure 7 shows that when the number of triangles is reduced by two orders of magnitude, the resulting model is still quite accurate when rendered. Scans of objects with very simple underlying geometry can be reduced greatly, such as the models in Figure 9. Here the reduction ratios are 52 (the cup) and 175 (the Fandisk).

Possibly the most important result is the one in Figure 10. Here we illustrate the benefit of finding the optimal decimation for a set of animated objects. In particular, a static decimation might falsely identify a long skinny region on the leg that spans the knee as a single planar patch, which causes a significant error when using the decimated model for animation. The Bayesian technique identifies a different model which deforms nicely.

A comparison to the QSlim technique [Garland and Heckbert 1997] is shown in Figure 8. The ability of our algorithm to fit a shading function over each surface patch allows for these pleasant renderings and constitutes one of the benefits of our algorithm relative to QSlim.

8. DISCUSSION AND FUTURE WORK

We presented a Bayesian approach for surface reconstruction and decimation. The approach uses oriented noise models, and combines them with a sub-linear prior. It also provides a provision for learning the prior. The Bayesian posterior is transformed into an energy minimization problems, and mixed discrete-continuous optimization techniques are applied to achieve the desired result.

There is ample opportunity for future research. One pertains to the popular *Iterative closest point algorithm* [Besl and McKay 1992]: Our Bayesian formulation should make it straightforward to simultaneous align multiple range scans while smoothing them. Another pertains to the learning component of our work: In addition to learning surface priors, it should be possible to learn the actual noise model of a sensor. We also suspect that further improvements can be achieved by better modeling the exact physical noise characteristics of a sensor.

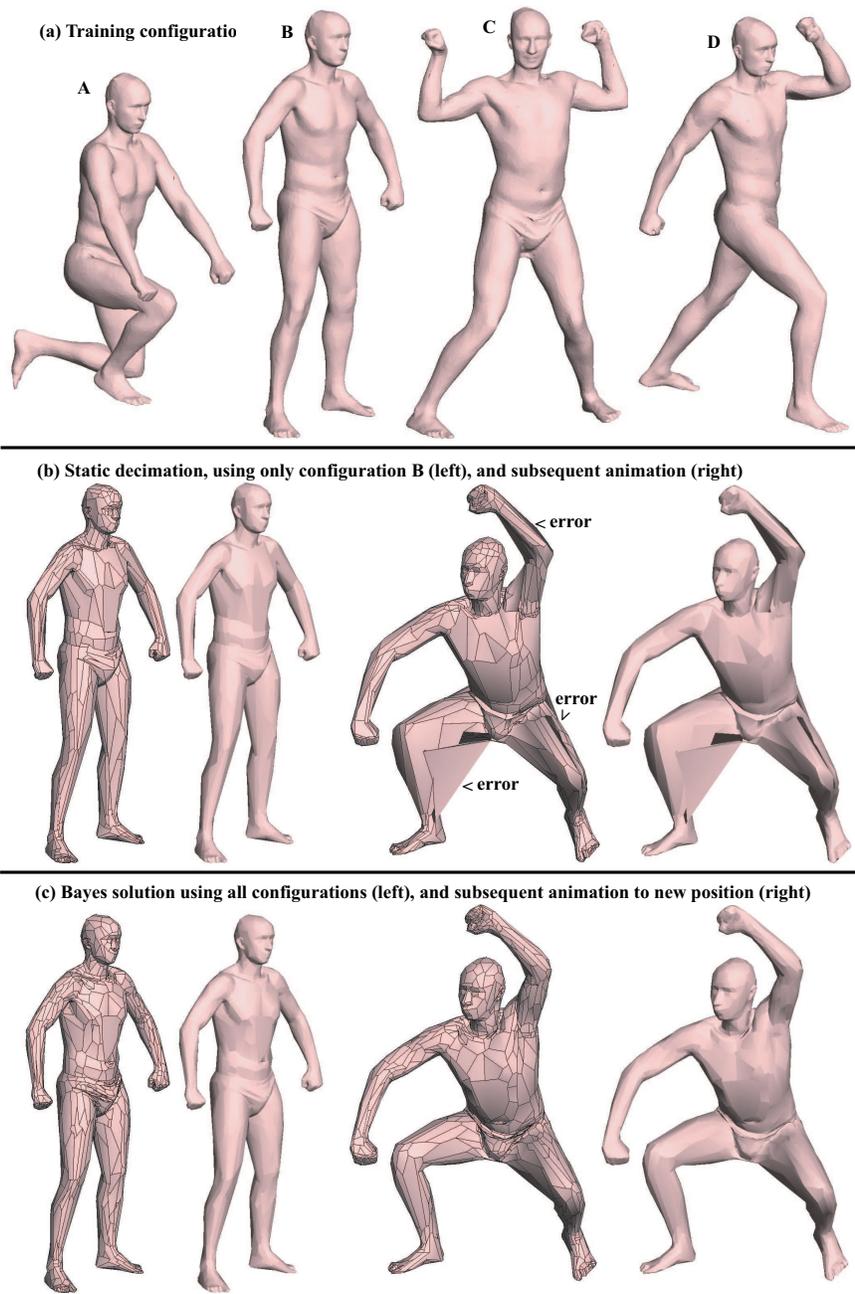


Fig. 10. Comparison of decimation for static versus animated meshes. (a) shows four deformed meshes, which serve as training instances. (b) depicts the static decimation result, obtained using only one of the training instances (Configuration A). When animating this decimated mesh, severe errors occur. In contrast, (c) shows the Bayes decimation optimized for all four training instances. Here such errors are absent in the animation.

Acknowledgment

We are indebted to Drago Anguelov, Jean-Yves Bouguet, James Davis, Mathieu Desbrun, Thouis Jones, Shachar Fleishman, Marc Levoy, and Rick Szeliski for providing us with data, models, and ample advise. All data and models reprinted in this article are reprinted with permission by the respective authors.

REFERENCES

- ALEXA, M. AND MÜLLER, W. 2000. Representing animations by principal components. *Comput. Graph. Forum* 19, 3.
- ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LEVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph.* 22, 3, 485–493.
- BAJAJ, C. L. AND XU, G. 2003. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans. Graph.* 22, 1, 4–32.
- BARHAK, J. AND FISCHER, A. 2001. Adaptive reconstruction of freeform objects with 3d som neural network grids. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, Washington, DC, USA, 97.
- BESL, P. J. AND MCKAY, N. D. 1992. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 2, 239–256.
- BRICENO, H. M., SANDER, P. V., MCMILLAN, L., GORTLER, S., AND HOPPE, H. 2003. Geometry videos: a new representation for 3d animations. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 136–146.
- CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *VIS '00: Proceedings of the conference on Visualization '00*. IEEE Computer Society Press, Los Alamitos, CA, USA, 397–405.
- COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM Trans. Graph.* 23, 3, 905–914.
- DAVIS, J., NEHAB, D., RAMAMOORTHY, R., AND RUSINKIEWICZ, S. 2005. Spacetime stereo: A unifying framework for depth from triangulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 27, 2 (Feb.), 296–302.
- DESBRUN, M., MEYER, M., SCHRODER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 317–324.
- DESBRUN, M., MEYER, M., SCHRODER, P., AND BARR, A. H. 2000. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Graphics Interface*. 145–152.
- FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Trans. Graph.* 22, 3, 950–953.
- GARLAND, M. AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 209–216.
- GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 355–361.
- JEONG, W.-K., IVRISSIMTZIS, I. P., AND SEIDEL, H.-P. 2003. Neural meshes: Statistical learning based on normals. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, Washington, DC, USA, 404.
- JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph.* 22, 3, 943–949.
- KRIZEK, M., NEITTAANMKI, P., GLOWINSKI, R., AND KOROTOV, S. 2004. *Conjugate Gradient Algorithms and Finite Element Methods*. Springer-Verlag, Berlin, New York.
- LEVIN, A., ZOMET, A., AND WEISS, Y. 2002. Learning to perceive transparency from the statistics of natural scenes. In *NIPS*. 1247–1254.
- MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph.* 23, 3, 385–392.
- PRESS, W. H. 1988. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, Cambridge; New York.
- SAUL, L. K. AND ROWEIS, S. T. 2003. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res.* 4, 119–155.
- STAHL, D., EZQUERRA, N., AND TURK, G. 2002. Bag-of-particles as a deformable model. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 141–150.
- SZELISKI, R. AND TERZOPOULOS, D. 1989. From splines to fractals. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 51–60.
- SZELISKI, R. AND TONNESEN, D. 1992. Surface modeling with oriented particle systems. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 185–194.
- TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2002. Geometric surface smoothing via anisotropic diffusion of normals. In *VIS '02: Proceedings of the conference on Visualization '02*. IEEE Computer Society, Washington, DC, USA.
- TAUBIN, G. 1995. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 351–358.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 205–214.

Received October 2005; revised; accepted.