

Smoothing of meshes and point clouds using weighted geometry-aware bases

Tim Volodine

Denis Vanderstraeten

Dirk Roose

Report TW 451, March 2005



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Smoothing of meshes and point clouds using weighted geometry-aware bases

Tim Volodine
Denis Vanderstraeten
Dirk Roose

Report TW451, March 2005

Department of Computer Science, K.U.Leuven

Abstract

Recently, Sorkine et al. proposed a least squares based representation of meshes, which is suitable for compression and modeling. In this paper we look at this representation from the viewpoint of Tikhonov regularization. We show that this viewpoint yields a smoothing algorithm, which can be seen as an approximation of the shape using *weighted* geometry aware bases, where the weighting factor is determined by the algorithm. The algorithm combines the Laplacian smoothing approach with the smoothing spline approach, where a global deviation constraint is imposed on the approximation. We use the generalized Laplacian matrix to measure smoothness and show how it can be modified in order to obtain smoothing behavior similar to that of curvature flow and feature preserving smoothing algorithms. The method is applicable to meshes, polygonal curves and point clouds in arbitrary dimensional spaces.

Keywords : smoothing, point clouds, constrained least squares, tikhonov regularization.

CR Subject Classification : G.1.3, I.3.5.

AMS(MOS) Classification : Primary : 65Y20, 65D10 Secondary : 65F22, 65F20, 93E24.

Smoothing of meshes and point clouds using weighted geometry-aware bases

Tim Volodine¹, Denis Vanderstraeten², and Dirk Roose¹

¹ KULeuven, Dept Computer Science, Celestijnenlaan 200A, B-3001, Belgium
{timv,dirkr}@cs.kuleuven.be

² Metris Belgium, Interleuvenlaan 86, B-3001, Belgium
dv@metris.com

Abstract. In [1] Sorkine et al. proposed a least squares based representation of meshes, which is suitable for compression and modeling. In this paper we look at this representation from the viewpoint of Tikhonov regularization. We show that this viewpoint yields a smoothing algorithm, which can be seen as an approximation of the shape using *weighted* geometry aware bases, where the weighting factor is determined by the algorithm. The algorithm combines the Laplacian smoothing approach with the smoothing spline approach, where a global deviation constraint is imposed on the approximation. We use the generalized Laplacian matrix to measure smoothness and show how it can be modified in order to obtain smoothing behavior similar to that of curvature flow and feature preserving smoothing algorithms. The method is applicable to meshes, polygonal curves and point clouds in arbitrary dimensional spaces.

1 Introduction

Shape smoothing is often a necessary pre-processing step in the acquisition and manipulation of 3D models. Many 3D models today are obtained by laser scanners, touch probes and structural light range-finding techniques, which produce noisy point clouds and meshes. Obviously, for further processing and visualization it is desirable to reduce the noise acquired during measurement. In addition smoothing algorithms often have the effect of data reduction, since it is easier to encode a smooth shape, than a noisy one. Smoothing also greatly improves the reliability of derived shape properties such as normals and curvatures.

An important class of mesh smoothing algorithms is based on the discrete Laplacian smoothing approach [2–4]. These algorithms are typically iterative and can be seen as an approximation of a diffusion process, where at each iteration each vertex moves in the direction of the Laplacian vector at that vertex. Since the Laplacian vector at a vertex is usually estimated using a local neighborhood, the resulting algorithms are efficient and simple to implement.

The use of uniform approximation of the Laplacian operator [2] results in what we call the *classical* Laplacian smoothing algorithm, where each vertex moves towards the center of its neighbors. When the mesh is irregular, weighted combinations of neighbors can be used to obtain better approximations of the

Laplacian vector [4] or to discretize the curvature flow approach [3]. We refer to these methods as the *weighted* Laplacian smoothing algorithms.

In this paper we describe an algorithm similar to the weighted Laplacian smoothing approach but with additional features. While the Laplacian smoothing algorithms are iterative and do not have a natural stopping criterion, the proposed algorithm is non-iterative and provides a global deviation constraint by means of a user-specified parameter τ , called the *smoothing factor*. This parameter controls the extent of smoothing, such that the *average quadratic deviation* of each smoothed point from its original position is limited to τ/n , with n the number of points. This kind of deviation constraint is widely used in the literature on smoothing splines and requires a solution of a non-linear problem. Fortunately, the optimal solution can be found efficiently by solving several least squares systems (see section 6).

Furthermore we use the *generalized Laplacian* operator (section 2.1) with weights chosen such, as to minimize vertex drift and triangle overlap. We also provide scaled versions of the generalized Laplacian operator, which make the proposed smoothing algorithm behave like *curvature flow* smoothing and *feature preserving* smoothing methods.

The proposed method is closely related to the concept of geometry aware bases (section 3) and the Tikhonov regularization process [5]. In section 5 we show that from the viewpoint of Tikhonov regularization the proposed smoothing algorithm has the effect of a low-pass filter.

2 Notation and definitions

2.1 Generalized discrete Laplacian

We represent the input to the smoothing algorithm as a directed graph $G = (V, E)$, where $V = \{\mathbf{p}_i \in \mathbb{R}^{s \times 1}, i = 1..n\}$ is the set of vertices, and E is the set of directed edges, i.e. $E = \{E_i, i = 1..n\}$, with $E_i = \{\overline{ij} \mid j \in \mathcal{I}_i\}$ and $\mathcal{I} = \{j \mid \mathbf{p}_j \text{ is a neighbor of } \mathbf{p}_i\}$. Note that we only need to know the local neighborhoods of each point. A mesh can be used to define neighborhoods, but it is not required. Using the graph G we can construct a *generalized discrete Laplacian* matrix $L \in \mathbb{R}^{n \times n}$, which is defined as

$$L_{ij} = \begin{cases} -\sum_{j \in \mathcal{I}_i} w_{ij} & i = j \\ w_{ij} & i \neq j, \end{cases} \quad (1)$$

with weights $w_{ij} \neq 0$ if points \mathbf{p}_i and \mathbf{p}_j are neighbors and zero otherwise. When all $w_{ij} \geq 0$, the matrix L can be viewed as a coefficient matrix of a system of *convex combination equations*, written as $L\mathbf{x} = 0$. We say that L is *normalized* if it has unit diagonal.

We denote by $P = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n)^\top$ the $n \times s$ matrix of all input points coordinates and by $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^\top$ the $n \times s$ matrix of all smoothed points. L_i , P_i and X_i denote the i -th row of L , P and X respectively. By $P^{(k)}$ and $X^{(k)}$ we denote the k -th column of P and X . With the introduced notation the generalized discrete Laplace operator at \mathbf{p}_i can be written as $\Delta\mathbf{p}_i = (L_i P)^\top$.

2.2 Smoothing using Tikhonov regularization

Our smoothing approach can be formulated as a linear algebra problem, where we find the solution matrix $X^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_n^*)^\top$, such that

$$X^* = \arg \min_X \{\|LX\|_F^2\} \quad (2)$$

$$\text{subject to } \|D(X - P)\|_F^2 \leq \tau, \quad (3)$$

where L is the *generalized discrete Laplacian* as in (1), P is the matrix with the coordinates of the input points, $D = \text{diag}(\omega_1, \omega_2, \dots, \omega_n)$ the diagonal matrix of weights and τ is the *smoothing factor*. Equation (2) represents the *smoothness condition* and the inequality (3) defines a constraint on the *average displacement* of the smoothed points. Introducing a Lagrange multiplier λ , we reformulate (2) as

$$X^* = \arg \min_X \{\|LX\|_F^2 + \lambda\|D(X - P)\|_F^2\}, \quad (4)$$

where λ can be determined *a priori*, such that constraint (3) is satisfied.

Equation (4) can be visually represented by a mass-spring system shown in figure 1. The displacement term $\|D(X - P)\|_F^2$ represents the potential energy of the springs between the original points \mathbf{p}_i and the smoothed points \mathbf{x}_i . The smoothness term $\|LX\|_F^2$ minimizes some energy derived from the springs between the smoothed points themselves.

The discrete weighted least squares problem (4) is also known as linear *Tikhonov regularization* method in numerical literature [5]. Since the generalized discrete Laplacian is singular, minimizing $\|LX\|_F^2$ as such, is not well-defined. By supplying additional information about the solution, in the form of a λ -weighted displacement term, the solution to (4) will be unique for any $\lambda > 0$.

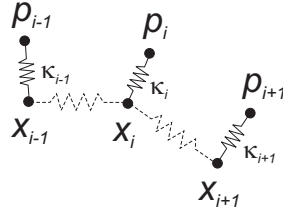


Fig. 1. Physical interpretation of the equation (4) as a mass-spring system, the stiffness of springs $\kappa_i = \omega_i^2$.

3 Related Work

There is extensive literature available on both smoothing of meshes and smoothing splines [6, 7]. We do not attempt to give an exhaustive overview of the ref-

erences, instead we only consider the references which are, to our opinion, most relevant in this case.

A large class of smoothing algorithms can be written using the generalized Laplacian, where in each iteration a new position \mathbf{p}'_i is computed for each vertex \mathbf{p}_i ,

$$\mathbf{p}'_i = \mathbf{p}_i + \mu(L_i P)^\top, \quad (5)$$

with $\mu \in (0, 1)$. Different weights w_{ij} for the generalized Laplacian matrix L , result in different smoothing algorithms. For example, *equal weights* and *reciprocal weights* were used by Taubin [2] to construct a low-pass filter. In the irregular setting, Desbrun et al. [3] uses Fujiwara weights, which are more accurate than the classical Laplacian approximation. Guskov et al. [4] propose a non-uniform generalization of the discrete Laplacian operator using weights based on second order differences. Finally, the so-called *cotangent weights* can be used to discretize the mean curvature flow approach [3].

Recently, several *feature preserving* smoothing algorithms were proposed, based on concepts from robust statistics. In particular an iterative version of the *bilateral filtering* approach for meshes has been proposed by Fleishman et al. [8]. In [9] Jones et al. present a non-iterative generalization of the bilateral filtering technique, applicable to triangle soups.

In [1], Sorkine et al. introduces the concept of *geometry aware bases*, as a new means of shape approximation. The geometry aware bases are vectors that assign a real value to each vertex in the mesh. They depend on both the connectivity, represented by the combinatorial Laplacian, and some geometric information, given by carefully selected mesh vertices, called the anchors. An approximation to the original mesh can be obtained by computing a certain linear combination of the geometry aware vectors. Since the number of anchors is typically small, this representation appears to be very suitable for compression, progressive transmission and editing.

The geometry aware bases can be used for smoothing as well. This comes from the fact that the reconstruction using geometry aware bases is related to the Tikhonov regularization approach. In fact, each column of X^* in (4) can be expressed as a linear combination of the geometry aware basis vectors \mathbf{v}_i , which minimize

$$\|L\mathbf{v}_i\|_2^2 + \lambda\|D(\mathbf{v}_i - \mathbf{e}_i)\|_2^2, \quad (6)$$

with $\{\mathbf{e}_i\}$ the canonical basis in \mathbb{R}^n .

In our approach, the optimal λ is determined automatically during the smoothing algorithm, resulting in a method which can be viewed as an approximation method using λ -weighted geometry aware bases.

4 Construction of the Laplacian matrices

In our smoothing algorithm we use positive weights for the construction of the generalized Laplacian. This choice is related to the aim to preserve the topology of the original mesh, because positive weights result in *convex combinations* of

neighboring points. During smoothing we would like to avoid triangle flips in the case of meshes or self-intersections in the case of curves. In general, however, solving (4) with positive weights does not guarantee intersection-free results, because the deviation from the convex combination constraint $\|LX\|_F^2$ is minimized in the least squares sense. As λ approaches zero we expect this constraint to be increasingly satisfied, reducing the possibility of flips.

4.1 Generalized Laplacian matrix variants

We introduce three different variants of the generalized Laplacian matrix:

- $L^{(n)}$ – denotes the normalized Laplacian with unit diagonal (1).
- $L^{(\kappa)}$ – is the curvature scaled Laplacian.
- $L^{(f)}$ – is the feature preserving Laplacian.

In this section we show how the matrices $L^{(\kappa)}$ and $L^{(f)}$ are derived from $L^{(n)}$. First, let us define the *length normalized* Laplacian matrix, for which the i -th row is given by

$$\bar{L}_i^{(n)} = L_i^{(n)} g\left(\|L_i^{(n)} P\|_2\right). \quad (7)$$

The scalar function $g(x)$ serves the purpose to scale each row $L_i^{(n)}$ such that $(\bar{L}_i^{(n)} P)^\top$ has approximately unit length. In regions of low curvature $\|L_i^{(n)} P\|_2$ can be very small or even zero, so simply taking $g(x) = 1/x$ would lead to very large entries in $\bar{L}_i^{(n)}$, yielding a very ill-conditioned matrix. To avoid numerical problems we use the robust Huber *edge stopping* function, which is defined for positive x as

$$g(x) = \begin{cases} 1/\sigma_h & x \leq \sigma_h \\ 1/x & x > \sigma_h, \end{cases} \quad (8)$$

with some small $\sigma_h \approx 1e - 7$.

The normalized Laplacian represents a weighted version of the classical Laplacian smoothing. Since the normalized Laplacian is scale-independent, it does not always properly reflect the frequencies of the mesh. If the sampling is non-uniform, the normalized Laplacian can be the same for small neighborhoods as well as for large ones. In order to solve this problem we introduce the scale-dependent version of the Laplacian, i.e. $L^{(\kappa)}$ with

$$L_i^{(\kappa)} = \bar{L}_i^{(n)} \tilde{\kappa}_i, \quad (9)$$

where $\tilde{\kappa}_i$ denotes an approximation of the curvature at point \mathbf{p}_i (see sections 4.2, 4.3). This matrix $L^{(\kappa)}$ actually mimics the *curvature flow* approach [3]. The effect of the curvature dependent smoothing can be seen in figure 3. The noisy left half of the circle is smoothed out faster in figure 3(c), compared to the normalized smoothing in figure 3(b).

In some cases, however, it is desirable not to smooth points of high curvature, e.g. the corners of the square in figure 4. This kind of feature preservation can

be achieved by using the feature preserving Laplacian $L^{(f)}$, which is obtained from $\bar{L}^{(n)}$ by weighting each row with a feature preserving function, $\psi(\tilde{\kappa}_i)$, i.e.

$$L_i^{(f)} = \bar{L}_i^{(n)} \psi(\tilde{\kappa}_i). \quad (10)$$

For the feature preserving function ψ we use the Gaussian $\psi(\tilde{\kappa}_i) = e^{-\frac{1}{2}\tilde{\kappa}_i^2/\sigma_f^2}$, because it tends to assign zero weight to outliers, points in high curvature regions, which we consider to belong to features. The curvatures are rescaled such that $\tilde{\kappa}_i \in [0, 1]$, and σ_f is a user-specified parameter. Figure 4(c) shows the result of smoothing using $L^{(f)}$ where the corners are clearly preserved.

4.2 Construction of L from polygons

When working with polygonal curves and polygonal curve networks (i.e. graphs with degree larger than two at some vertices) we use the *normalized reciprocal distance weights* for the construction of $L^{(n)}$. Let $l_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|_2$, the normalized weights corresponding to the inner vertices are given by $w_{ij} = l_{ij}^{-1} / \sum_{k \in \mathcal{I}_i} l_{ik}^{-1}$, and $\mathcal{I}_i = \{i-1, i+1\}$. The geometric interpretation of this discretization of the Laplace operator is shown in figure 2.

The curvature at the polygon vertex \mathbf{p}_i is estimated using the osculating circle approach. The osculating circle is approximated by the circle through the successive points $\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$. This definition gives the approximate $\tilde{\kappa}_i = 4A/abc$, where A is the area of the triangle in figure 2. Using this approximation of the curvature, the $L^{(\kappa)}$ -matrix is equivalent to the discretization of the Laplacian using 1D Fujiwara weights [3], up to a factor of $\cos((\beta - \alpha)/2)$.

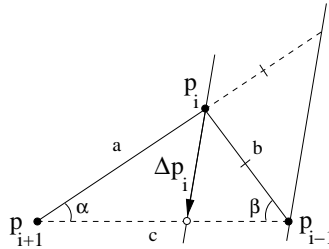


Fig. 2. Geometric interpretation of the reciprocal weights, $\mathbf{p}_i + \Delta\mathbf{p}_i = \frac{b}{a+b}\mathbf{p}_{i+1} + \frac{a}{a+b}\mathbf{p}_{i-1}$.

4.3 Construction of L from triangular meshes

When G represents a triangular mesh a natural choice for the construction of the Laplacian is to use convex combinations of the 1-ring neighborhoods. In the examples we use the *mean value* weights for inner vertices, because for general

neighborhoods they are the only barycentric weights [10]. An alternative would be to use cotangent weights, because they can be used to discretize the Laplace-Beltrami operator. However, in general, cotangent weights can be negative, while mean value weights are always positive. In addition, both the reciprocal weights (see section 4.2) and the mean value weights have the *reproduction property*: points lying on a line or a plane are left invariant by smoothing.

In the examples, the curvature is approximated by the magnitude of the discrete *mean curvature normal* [3],

$$\tilde{\kappa}_i \mathbf{n} = \frac{1}{4A} \sum_{j \in \mathcal{I}_i} (\cot \beta_j + \cot \gamma_j) (\mathbf{p}_i - \mathbf{p}_j), \quad (11)$$

where β_j, γ_j are the 3D angles opposite to the edge $\mathbf{p}_i \mathbf{p}_j$. In case noise prevents to use formula (11), other more robust techniques for estimating curvature can be used [11].

4.4 Construction of L from point clouds

For point clouds, the choice of the neighborhood is an important issue, because, contrary to the previous two cases, no connectivity information is available. It is possible to use k -nearest neighbors, but in practice it is better to use local neighborhoods, especially in the presence of anisotropy. When the points are sampled from a manifold surface we use a local meshing approach, based on the so-called *Delaunay neighborhoods* [12]. A Delaunay neighborhood of a point is obtained by projecting k -nearest neighbors on their least squares plane and computing the Delaunay triangulation. The resulting 1-ring neighborhood in the plane defines the 1-ring neighborhood in the point cloud. In this way the smoothing problem reduces to the mesh case. Since the cotangent weights are positive when computed on a Delaunay triangulation, it is possible to use them on the Delaunay neighborhoods as well. In practice the results using mean value weights and cotangent weights are visually similar.

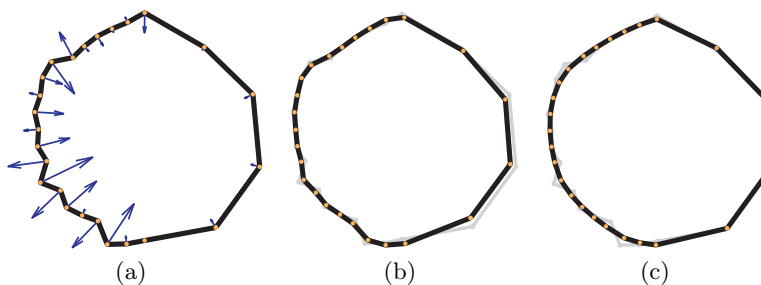


Fig. 3. Smoothing of a noisy non-uniformly sampled circle (a) the arrows represent the column vectors of $(L^{(\kappa)} P)^T$; (b) smoothing using $L^{(n)}$; (c) smoothing using $L^{(\kappa)}$.

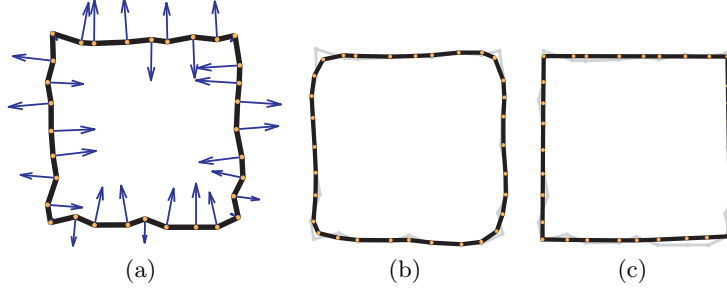


Fig. 4. Corner preserving smoothing of a noisy square (a) the arrows represent the columns of $(L^{(f)}P)^\top$; (b) smoothing using $L^{(n)}$; (c) smoothing using $L^{(f)}$.

5 Smoothing from the Tikhonov regularization viewpoint

The smoothing problem (4) can be viewed as a Tikhonov regularization process, in which the smoothness term is regularized by the error term. This view provides insight into the properties of the smoothing algorithm, such as filtering properties, convergence and computational aspects.

5.1 Low pass filtering

The eigenvectors of the combinatorial Laplacian matrix can be seen as an extension of the Fourier transform basis functions, with eigenvalues representing frequencies [2, 1]. We argue that the same idea applies to the *right singular vectors* of the generalized Laplacian. In fact, if the generalized Laplacian matrix is symmetric and positive semidefinite its eigenvalues are equal to the singular values and the eigenvectors are equal to the singular vectors up to a sign. Although the Laplacian with mean value weights is generally not symmetric, its right singular vectors have a typical frequency distribution as can be seen in figure 5(b), which shows the power spectrum of the right singular vectors of $L^{(n)}$, corresponding to the simplified cow model 5(a). The highest frequency content is carried by the singular vectors with lowest index, i.e. the singular vectors corresponding to the largest singular values σ_i .

This observation is interesting, because Tikhonov regularization is related to Wiener filtering in the Fourier domain of the signal [5]. When using unit weights ($D = I$), the solution to the least squares problem (4) can be expressed in terms of the singular values σ_i and right singular vectors \mathbf{v}_i of L ,

$$X^* = \sum_i \mathbf{v}_i \left(\frac{\lambda}{\sigma_i^2 + \lambda} \mathbf{v}_i^\top P \right). \quad (12)$$

We call $\gamma_i = \lambda/(\sigma_i^2 + \lambda)$ the filter factors. When λ tends to infinity, all $\gamma_i \approx 1$, so we obtain the original points expressed as a linear combination of the right singular vectors of L . When λ tends to zero, the filter factors γ_i corresponding

to large σ_i tend to zero (figure 6). Since large singular values σ_i correspond to highly oscillating \mathbf{v}_i , we can say that the ‘higher’ frequencies are filtered out by the Tikhonov regularization process.

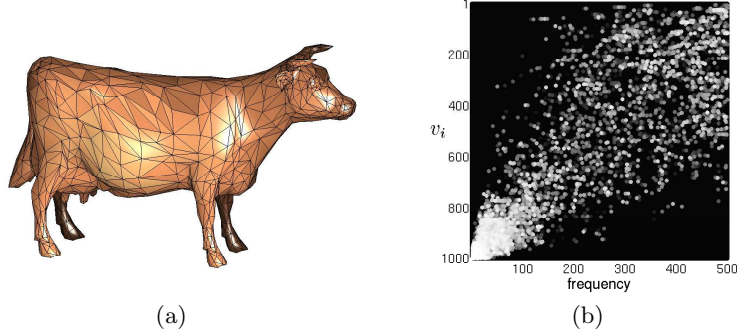


Fig. 5. (a) simplified COW model with 1000 points (b) power spectrum image of the right singular vectors, where the i -th horizontal line in the image represents the power spectrum of \mathbf{v}_i . The dots represent the distribution of the 1% of the highest power content.

5.2 Convergence properties

When λ tends to infinity, we obviously obtain the original points as the solution to the regularization problem (4). A more interesting situation occurs when λ goes to zero. When $\lambda = 0$, equation (4) accepts more than one solution, because L is singular. However, since the Laplacian has $\sigma_n = 0$, *in the limit* only one γ_i will be nonzero, i.e. $\lim_{\lambda \rightarrow 0} \gamma_n = 1$. We also know that the last singular vector is a constant vector $v_{n,i} = c$, because L has zero row-sums. Therefore the solution in each dimension, as λ approaches zero, is a matrix X^* with rows

$$X_i^* = v_{n,i}(\mathbf{v}_n^\top P) = c^2 \sum_{i=1..n} \mathbf{p}_i^\top. \quad (13)$$

Since \mathbf{v}_n is normalized, $c^2 = 1/n$, meaning that the limit value for each smoothed point \mathbf{x}_i^* is the *centroid* of the point cloud P .

Using the *generalized singular value decomposition* (GSVD) of (L, D) , it can be shown that for general weights the solution of (4) converges to the *weighted centroid* \bar{P} of P , for which the i -th row is given by

$$\bar{P}_i = \left(\sum_i \omega_i^2 \mathbf{p}_i^\top \right) / \sum_i \omega_i^2. \quad (14)$$

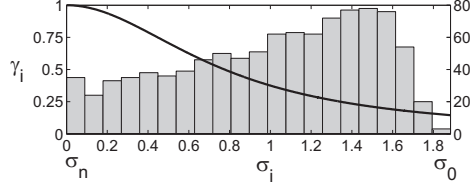


Fig. 6. Plot of the filter factors γ_i for $\lambda = 0.6$ and the histogram of the distribution of the singular values of $L^{(n)}$ derived from the COW mesh.

6 Determination of the regularization parameter λ

In this section we provide a method for determining the regularization parameter λ in equation (4).

6.1 Constraint on the sum of squared errors

Consider the deviation function

$$\phi(\lambda) = \|D(X(\lambda) - P)\|_F^2, \quad (15)$$

where $X(\lambda) = (x_1, x_2, \dots, x_n)^\top$ is the solution of (4) for a given λ . This function is just the *sum of squared errors* (SSE) of the smoothed points \mathbf{x}_i w.r.t. the original points \mathbf{p}_i , i.e. $\phi(\lambda) = \sum_{i=1..n} \omega_i^2 \|\mathbf{p}_i - \mathbf{x}_i\|_2^2$. It can be shown that $\phi(\lambda)$ is a convex and strictly decreasing function [13, chapter 12]. Due to this convexity, the minimum of (4) is achieved for $\lambda = \lambda^*$, satisfying $\phi(\lambda^*) = \tau$.

6.2 Determination of λ using GSVD

The generalized singular value decomposition of matrices L and D can be used to find λ^* , but for a large number of points n , it is *impractical*. In our algorithm we prefer to compute an approximation of the optimal λ^* , using a rational interpolation scheme, analogous to the one described by Dierckx in [7].

6.3 Determination of λ using rational interpolation

By using rational interpolation to determine λ , as described below, the computation of the GSVD is not required. It is sufficient to have a least squares solver. In fact, the solution to the minimization problem (4) for a given λ can be obtained by solving s independent least squares problems,

$$\min_{X^{(k)}} \left\| \begin{bmatrix} L \\ \sqrt{\lambda} D \end{bmatrix} X^{(k)} - \begin{bmatrix} \mathbf{0}_{n \times 1} \\ \sqrt{\lambda} D P^{(k)} \end{bmatrix} \right\|_2^2, \quad k = 1..s. \quad (16)$$

By solving these systems for certain λ values and using rational interpolation on the corresponding solutions we can find the optimal λ with the desired accuracy.

In addition, for smoothing purposes, it is not necessary to find λ^* to full precision. The function $\phi(\lambda)$ flattens very rapidly and therefore a small deviation from the optimum will be indistinguishable in visualization applications.

We briefly outline the adopted interpolation scheme, further details can be found in [7]. The proposed rational interpolation scheme is an iterative method which starts with three values for λ , λ_1 , λ_2 and λ_3 , and the corresponding ϕ values. We use $\lambda_1 = 0$ and $\lambda_3 = \infty$ since we know the SSE-error at these points. For $\lambda_1 = 0$ we have $\phi(\lambda_1) = \|P - \bar{P}\|_F^2$, with \bar{P} defined in (14). At $\lambda = \infty$ the SSE-error is zero. There exist refined methods for computing an initial estimate for the optimal λ , like the generalized cross-validation procedure. However, we want to avoid any computationally expensive methods, therefore we use the simple estimate for λ^* borrowed from [14], $\lambda_2 = \|L^\top L\|_F / \|D^2\|_F$. In practice this estimate is sufficient as the starting value of the interpolation scheme. Using these three points on the error-curve we compute the rational interpolating function $R(\lambda) = (u\lambda + v)/(\lambda + w)$, by choosing suitable values for u, v and w (see figure 7). Requiring $R(\lambda) = \tau$ yields a better estimate $\tilde{\lambda}$, which is used in the next iteration. The algorithm terminates when the relative error $|\phi(\tilde{\lambda}) - \tau|/\tau \leq \xi$, where ξ is the tolerance, usually set equal to $1e - 3$. All examples presented in this paper converged in 4 to 8 iterations.

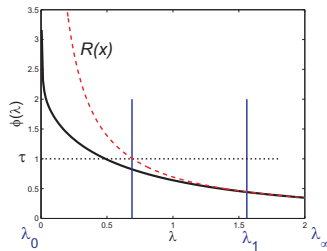


Fig. 7. SSE graph for the DOUBLE_TORUS model with the interpolating $R(x)$ through initial values and one interpolation step.

7 Solution of the least squares system

7.1 Normal equations

In each rational interpolation iteration we have to solve s least squares systems (16). In each dimension k we have a different right hand side but the same coefficient matrix, hence the solution for all dimensions can be found simultaneously provided the factorization of the coefficient matrix is available.

Taking into account the special structure of the systems in (16), it is possible to implement a fast dedicated QR-factorization algorithm. The matrix L can be

reordered such that it has a very small bandwidth, resulting in a solver with almost *linear* complexity in the number of points n .

In this paper we use general purpose sparse solvers to solve (16). We compared 3 suitable solvers available in Matlab, i.e. sparse QR, UMFPACK and CHOLMOD. The built-in sparse QR solver in Matlab can be directly applied to solve (16) while the other two are linear solvers and can be used to solve the corresponding *normal equations*, i.e.

$$(L^\top L + \lambda D^2)X^{(k)} = L^\top \mathbf{b} + \lambda D^2 P^{(k)}. \quad (17)$$

In table 8, the three algorithms running in Matlab are compared w.r.t. computational cost, measured on a P4 2.4 Ghz machine. Cholesky factorization based solver CHOLMOD appears to be the most efficient. It is a factor 5 to 10 faster than UMFPACK. Currently CHOLMOD is not part of Matlab 7.0, but will be included in the future versions.

points	speedup vs UMFPACK	speedup vs sparse QR	time CHOLMOD
3K	5.1	14.3	0.07s
40K	5.7	22.4	2.34s
44K	6.1	28.7	1.96s
56K	6.6	27.3	3.22s
122K	11.2	49.8	5.70s

Fig. 8. Table of the comparative results of the CHOLMOD solver. The speedup columns show the speedup-factor of CHOLMOD compared to the other solvers.

7.2 Condition number of the normal system

When solving the normal equations (17) we inevitably lose accuracy, because the condition number of $L^\top L$ increases quadratically compared to L . In this section we derive an upper bound for the condition number of (17) and argue that in practice the loss of accuracy is acceptable.

Let σ_i be the singular values of L and assume for the ease of explanation that $D = I$. Then the singular values of $L' = L^\top L + \lambda I$ are $\sigma_i^2 + \lambda$. This gives the condition number

$$\text{cond}(L') = \frac{\sigma_1^2 + \lambda}{\sigma_n^2 + \lambda} = 1 + \frac{\sigma_1^2}{\lambda}, \quad (18)$$

because $\sigma_n = 0$. From the matrix theory it is known that

$$\sigma_1^2 \leq \left(\max_i \sum_j |L_{ij}| \right) \left(\max_j \sum_i |L_{ij}| \right). \quad (19)$$

When using normalized Laplacian matrices, the row sum equals 2 and the maximal column sum depends on the maximal valency of the mesh. In practice the maximal column sum is limited, for all tested meshes we found $\sigma^2 \leq 4$. Obviously, the condition goes to infinity as λ approaches zero. However in practice, λ is of the order $\lambda = 1e - 2$ to $1e - 3$. In this case the loss of about 4 significant digits in the smoothed solution is acceptable.

8 Results

In figure 9, $L^{(n)}$ -smoothing is applied to a DOUBLE_TORUS mesh (1K points) with some random noise. In figure 10 we apply smoothing to a real world example, laser scanned mesh of a sheet metal part with 56K points. The natural boundary of the model is kept fixed (figure 10(b)). The interpolation at the boundary vertices is achieved by assigning infinite weights, resulting in the elimination of the corresponding rows in the least squares system (16).

The smoothing method can be applied to curve networks as illustrated in figure 11. The polygonal curves in 11(a) represent the noisy edges of a cube, placed on top of a cylinder. The smoothed version with fixed corner points is shown in figure 11(b). Note that the curve network consists of disconnected components.

In figure 12 we compare the results obtained with $L^{(f)}$ and $L^{(n)}$ smoothing of the DRAGON_HEAD point cloud with 30K points. For both methods we use a smoothing factor $\tau = 0.01$, which is approximately 6% of the bounding box diagonal. For the DRAGON_HEAD point cloud, both methods converge in 3 to 4 iterations in less than 10 seconds.

Figure 13 compares the smoothing spline approximation of a closed polygonal curve of Escher's LIZARD (in grey) with $L^{(n)}$ and $L^{(f)}$ -smoothing. In figures 13(a)-13(c) the smoothing factor τ is the same, meaning that, when measured at the parameter values for the smoothing spline and the polygon vertices for $L^{(n)}$ and $L^{(f)}$ -smoothing, the SSE for the three figures is equal. For the LIZARD we applied area preserving smoothing approach by rescaling the smoothed points, as suggested in [3].

Finally, figure 14 illustrates the $L^{(\kappa)}$ smoothing applied on a noisy, non-uniformly sampled Fermat's spiral, for different values of λ .

9 Conclusion

We presented an algorithm for smoothing of meshes and point clouds inspired on one hand by the *geometry-aware bases*, and on the other hand by the smoothing spline approach. The results indicate that the presented method is suitable both for denoising and for modeling, where we need a smooth approximation, not deviating too much from the original data.

One of the advantages of the presented smoothing algorithm is that, once the Laplacian matrix is available, the solution can be computed very efficiently

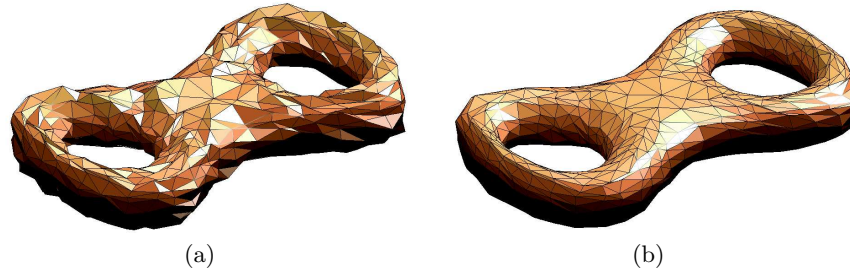


Fig. 9. Smoothing of a noisy double torus using the normalized Laplacian: (a) the original DOUBLE_TORUS model corrupted by noise; (b) after smoothing, $\lambda = 6.42e - 2$, and 7 iterations.

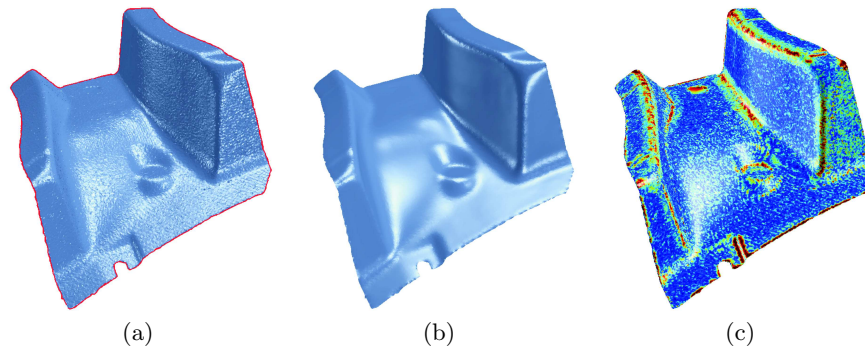


Fig. 10. (a) The original scanned noisy mesh of a sheet metal part; (b) $L^{(n)}$ -smoothed version; (c) difference of the original and smoothed meshes

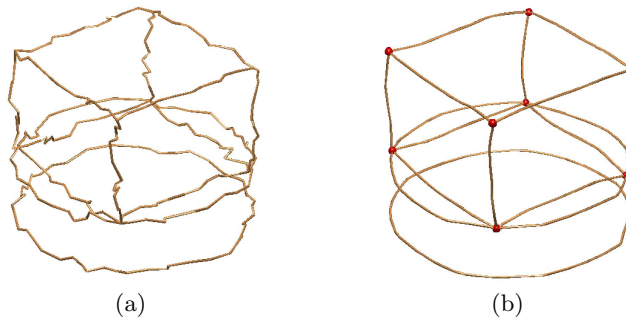


Fig. 11. (a) original noisy curve network (b) result of $L^{(n)}$ -smoothing, with 8 fixed vertices.

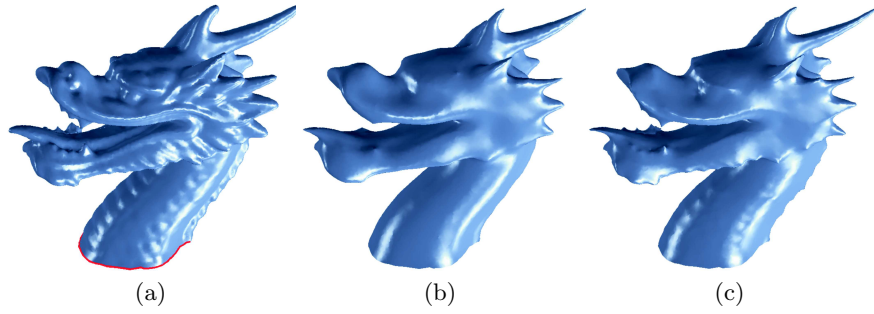


Fig. 12. (a) visualization of the original dragon point cloud (30K points); (b) smoothed version using $L^{(n)}$; (c) smoothing using $L^{(f)}$, $\sigma_h = 1e - 7$, $\sigma_f = 0.5$.

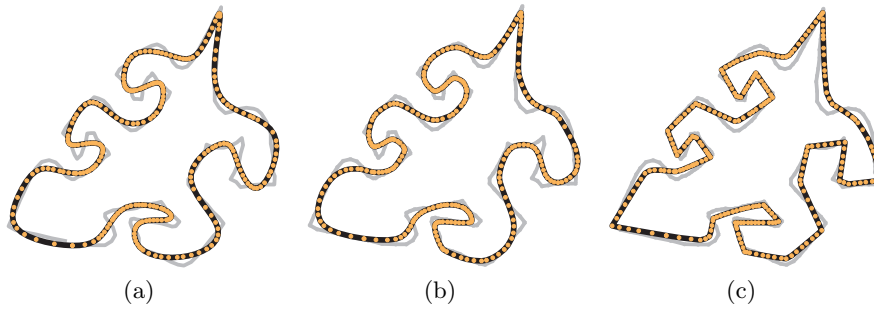


Fig. 13. Smoothing of the Escher's lizard given as a polygon: (a) smoothing spline of degree 3 with 40 knots, using the smoothing criterion of Dierckx; (b) $L^{(n)}$ -smoothing; (c) $L^{(f)}$ -smoothing, $\sigma_f = 0.5$.

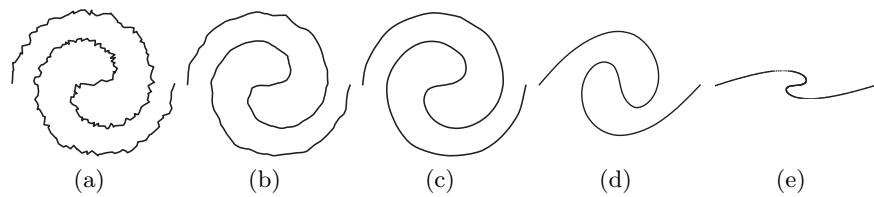


Fig. 14. (a) Fermat's spiral with polar equation $r^2 = \theta$ with noise (b) smoothing using $L^{(\kappa)}$, with $\lambda = 1.8e3$ (c) $\lambda = 9.1e1$ (d) $\lambda = 5.5e - 2$ (e) $\lambda = 2.8e - 3$.

using present-day, widely available sparse linear solvers, e.g. available in Matlab. The method provides a global deviation condition and is guaranteed to converge for any smoothing factor τ . The method is applicable to very general inputs, essentially data points with neighborhood information. Because of its connection to Tikhonov regularization, the method can be viewed as a filter in the space of the right singular vectors corresponding to the generalized singular values. The normalized version of the Laplacian often performs well in practice. It is the easiest version to compute and yields well-conditioned normal equations. We also showed that it is possible to weight the Laplacian, such that it mimics the curvature flow approach, or preserves features.

Shrinkage is a common problem of many smoothing algorithms. The presented algorithm also suffers from shrinkage. However, if desired, volume preservation can be guaranteed by a simple rescaling of the vertices.

Interestingly, when appropriate boundary conditions are imposed and the regularization parameter λ is varied, the resulting family of shapes can be seen as a smooth continuous interpolation between the original data and its parameterization. In figure 14, for example, the spiral converges to its parameterization on a straight line.

In the future, it would be interesting to investigate smoothing under the max-norm error metric, because in some applications it is desirable to limit the *maximal* deviation of the smooth approximation.

References

1. Sorkine, O., Cohen-Or, D., Irony, D., Toledo, S.: Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics* **11**(2) (2005) 171–180
2. Taubin, G.: A signal processing approach to fair surface design. In: *SIGGRAPH '95 Conference Proceedings*, ACM Press (1995) 351–358
3. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: *SIGGRAPH '99 Conference Proceedings*, ACM Press (1999) 317–324
4. Guskov, I., Sweldens, W., Schröder, P.: Multiresolution signal processing for meshes. *Computer Graphics Proceedings (SIGGRAPH 99)* (1999) 325–334
5. Hansen, P.C.: Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1998)
6. de Boor, C.: *A practical guide to splines*. Springer-Verlag (1978)
7. Dierckx, P.: *Curve and Surface Fitting with Splines*. Oxford University Press (1995)
8. Fleishman, S., Drori, I., Cohen-Or, D.: Bilateral mesh denoising. *ACM Trans. Graph.* **22**(3) (2003) 950–953
9. Jones, T.R., Durand, F., Desbrun, M.: Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph.* **22**(3) (2003) 943–949
10. Floater, M.S., Hormann, K., Kós, G.: A general construction of barycentric coordinates over convex polygons. *Advances in Computational Mathematics* (2004) Accepted.

11. Surazhsky, T., Magid, E., Soldea, O., Elber, G., Rivlin, E.: A comparison of gaussian and mean curvatures triangular meshes. In: Proceedings of IEEE International Automation (ICRA2003), Taipei, Taiwan (2003) 1021–1026
12. Floater, M., Reimers, M.: Meshless parameterization and surface reconstruction. *Comp. Aided Geom. Design* **18** (2001) 77–92
13. Golub, G.H., Loan, C.F.V.: *Matrix Computations*. John Hopkins University Press (1996) Third Edition.
14. Floater, M.S.: How to approximate scattered data by least squares. Technical report, SINTEF Report No. STF42 A98013, Oslo (1998)