RC22213 (W0110-051) October 18, 2001
Computer Science

IBM Research Report

Linear Anisotropic Mesh Filtering

Gabriel Taubin
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

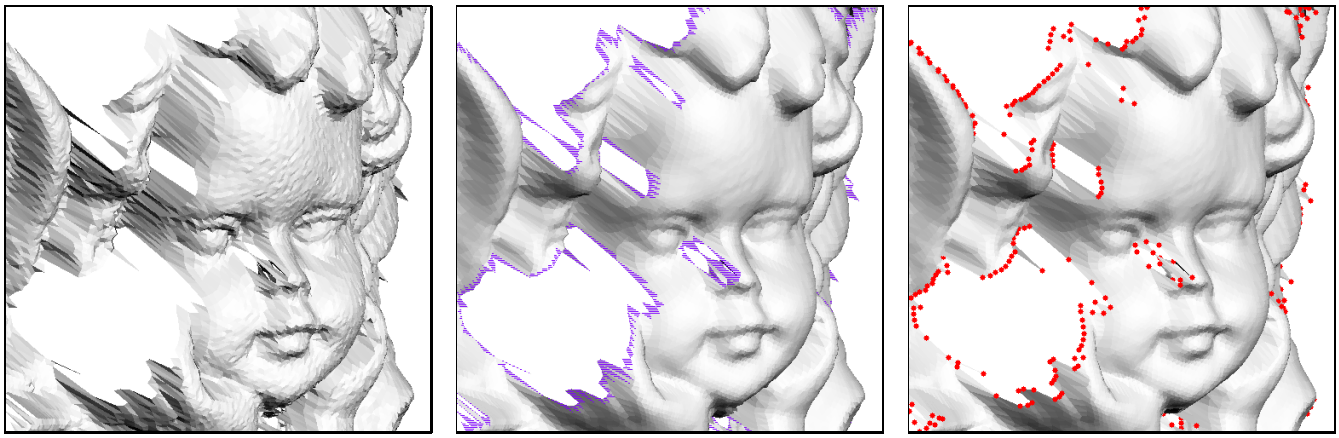
Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

October 2001 (under review)

Linear Anisotropic Mesh Filtering

Gabriel Taubin*

IBM T.J. Watson Research Center



A B C

Figure 1: Application to smoothing without tangential drift. (A) Noisy mesh with irregular edge length distribution. (B) After 4 steps of unit-length isotropic Laplacian smoothing of face normals with $\lambda = 0.5$, constrained boundary faces (blue), and fix vertices. (C) After 10 subsequent steps of face normal integrating anisotropic Laplacian smoothing on vertex positions with $\lambda = 0.5$, constrained boundary vertices (red), and fix face normals. Flat-shading used to enhance the faceting effect. Connectivity is kept constant.

Abstract

No algorithm for unconstrained polygon mesh denoising can beat Laplacian smoothing in simplicity and ease of implementation. In this paper we introduce a new algorithm for polygon meshes smoothing with vertex position and face normal interpolatory constraints composed of two phases. First the face normals are filtered independently of vertex positions. Then the vertex positions are filtered integrating the face normals in the least squares sense. Laplacian smoothing is used to smooth both the face normal field and the vertex positions, with a properly defined Laplacian operator in each case. We define isotropic, anisotropic, linear, and non-linear Laplacian operators for signals defined in Euclidean space, and on the unit sphere. In addition to the obvious applications to shape design, our algorithm constitutes a new fast and linear solution to the tangential drift problem, observed in meshes with irregular edge length and angle distribution. We show how classical linear filter design techniques can be applied in both cases, and conclude the paper determining integrability conditions for face normal fields.

CR Categories and Subject Descriptors:
I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modelling - surface, solid, and object representations.

General Terms: Mesh Signal Processing, Smoothing, Denoising, Anisotropic Diffusion, Algorithms, Graphics.

*http://www.research.ibm.com/people/t/taubin IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 taubin@us.ibm.com

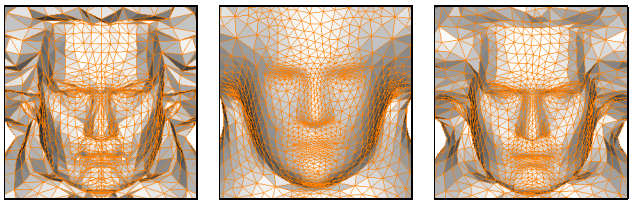
1 Introduction

The problem of smoothing or denoising large irregular polygon meshes of arbitrary topology, such as those extracted from volumetric medical data by iso-surface construction algorithms, or constructed by integration of multiple range images, has motivated most of the recent work in geometric signal processing [19].

1.1 Previous Work

Because of the size of the typical data sets, only linear time and space algorithms can be considered, particularly for applications such as surface design and mesh editing, where interactive rates are a primary concern. The simplest smoothing algorithm that satisfies the linear complexity requirement is Laplacian smoothing, a well established iterative algorithm introduced in the mesh generation literature to improve the quality of meshes used for finite element computations [5, 8]. In this context boundary vertices of the mesh are constrained not to move, but internal vertices are simultaneously moved in the direction of the barycenter of their neighboring vertices. And then the process is iterated a number of times.

Laplacian smoothing is not problem-free, though. When Laplacian smoothing is applied to a noisy 3D polygon mesh without constraints, noise is removed, but significant shape distortion may be introduced. This problem is called *shrinkage*. When a large number of Laplacian smoothing steps are iteratively performed, the shape undergoes significant deformations, eventually converging to the centroid of the original data. The $\lambda|\mu$ algorithm introduced by Taubin [18] solves



A B C

Figure 2: Laplacian vs. Taubin smoothing. FIR Filters based on the linear isotropic Laplacian operator. (A) Noisy mesh. (B) Result of Laplacian smoothing (10 steps with $\lambda = 0.6307$). (C) Result of Taubin Smoothing (10 steps with $\lambda_1 = 0.6307$ and $\lambda_2 = -0.6732$).

the shrinkage problem with a simple alternating sign modification that converts the Laplacian smoothing algorithm into a low-pass filter. Figure 2 shows the result of smoothing a noisy mesh with Laplacian and Taubin smoothing. In both cases the algorithms perform exactly the same number of operations. In the same paper, Taubin introduced Fourier Analysis on mesh signals as a tool to understand and predict the behavior of these linear smoothing processes [18]. This work was followed by a number of extensions, improvements, applications, and closely related algorithms, which addressed the following existing and/or open problems with mesh filters to some extent: enhancement and prevention of tangential drift, boundary and crease curve detection and enhancement, and introduction and intuitive control of position and normal constraints.

Taubin et.al. [21] introduced efficient and robust algorithms to evaluate any finite impulse response (FIR) linear filter on mesh signals. Zorin et.al. [24] showed an application to multiresolution interactive shape design. Kuriyama and Tachibana [13] used an anisotropic diffusion mechanism to generate smooth interpolatory subdivision surfaces with vertex position and normal constraints, and tension control. Hausler and Karbacher [9, 10] describe a non-linear method for smoothing polygon meshes based on circular arc approximations. Kobbelt et.al. [11, 12] introduced a multiresolution approach to denoising and shape design based on linear filters and constrained minimization. Desbrun et. al. [4] addressed the tangential drift problem observed in meshes with irregular edge length and angle distribution with a non-linear Laplacian operator and an infinite impulse response (IIR) filter with rational transfer function. And more recently, several authors [3, 14, 2, 15] have proposed closely related non-linear algorithms that extend to meshes the anisotropic diffusion approach to image segmentation introduced by Perona and Malik [16].

The ability to impose constraints to the smoothing process, such as specifying the positions of some vertices, or normal vectors, specifying ridge curves, or the behavior of the smoothing process along the boundaries of the mesh, is needed in the context of free-form interactive shape design. Taubin [18] shows that by modifying the neighborhood structure certain kind of constraints can be imposed without any modification of the algorithm, while other constraints that require minor modifications and the solution of small linear systems. Bierman et.al. [1] show how to construct subdivision surfaces with vertex normal constraints. Kobbelt et. al. [11, 12] formulate the problem as an energy minimization problem, and solve it efficiently with a multi-resolution approach on levels of detail hierarchies generated by decimation. Kuriyama [13] and Yamada et. al. [23] introduced soft vertex normal constraints trough a spring-based approach, but report slow convergence.

1.2 Contributions

In this paper we present a simpler and unified solution to some of the problems listed above based on new modifications of the Laplacian smoothing algorithm and the closely related FIR linear filters. As a first contribution we introduce an *anisotropic Laplacian operator* in which the weighting of displacements to neighboring vertices is given by matrices. These matrices defined on the vertices of the mesh can be regarded as a *discrete tensor field*. In the classical *isotropic Laplacian operator* the weights are scalars. To prevent tangential drift we compute the weights as functions of a field of face normals, in such a way that the resulting Laplacian smoothing algorithm integrates the face normals in the least squares sense. Other anisotropic Laplacian operators can be constructed to produce different effects, but we will not study them here.

If the field of face normals is smooth, a linear low-pass filter based on this anisotropic Laplacian operator removes noise while preventing tangential drift. Since computing face normals from noisy data produces much noisier face normal fields, we need to smooth the field of face normals before using it to generate the anisotropic weights for the vertex position smoothing process. As a second contribution we introduce an algorithm to evaluate FIR linear filters on signals defined on graphs and with values in the unit sphere. We then apply this algorithm to a field of face normals defined on the dual graph of a mesh. Again, our algorithm is Laplacian smoothing with a properly defined Laplacian operator for unit-length vector signals.

In our algorithm for denoising without tangential drift the face normals are first filtered independently of the vertex positions. Then the matrix weights of the linear anisotropic Laplacian operator are computed, and finally the vertex positions are filtered using the resulting linear anisotropic filter with constant weights. This algorithm is related to the *non-linear anisotropic diffusion* crease-enhancing algorithm introduced by Ohtake et.al. [14, 15], where a non-linear coupled diffusion process is introduced to simultaneously process face normals and vertex positions. But here the two processes are decoupled. We filter the face normals independently of the vertex positions, and then, as in the linear isotropic case, we compute the weights once and keep them fixed during the vertex position filtering process.

In the now classical Laplacian smoothing algorithm and its derivatives, imposing interpolatory vertex position constraints is easy, but imposing interpolatory normal constraints is not. In our new algorithm both kinds of interpolatory constraints are easy to apply. Interpolatory face normal constraints are imposed as in the classical case by not moving the constrained face normals in the direction determined by the Laplacian operator. Similarly, the constrained vertices are not moved in the second phase of the algorithm.

1.3 Paper Organization

The paper is organized as follows. In section 2 we basic mesh signal processing concepts and establish some notation. In section 3 we define the different types of Laplacian operators: linear, non-linear, isotropic, and anisotropic. In section 4 we extend the Fourier Analysis of mesh signals [18] to the linear

anisotropic case, and in section 5 we review the design of linear filters. In section 6 we describe a solution of the boundary shrinkage problem of classical Laplacian smoothing as a motivation for the algorithm introduced in this paper. In section 7 we describe the existing non-linear isotropic solutions to the tangential drift problem, and our new linear anisotropic solution. In section 8 we define an isotropic Laplacian operator for signals with values on the unit sphere, which allows us to extend the Laplacian smoothing algorithm to these signals. In section 9 we review the formulation of Rodrigues formula required in the previous section. In section 10 we determine integrability conditions for vector fields defined on the faces of a mesh, we formulate an orthogonal decomposition theorem, and show that our algorithm integrates the field of face normals in the least squares sense. In section 11 we show some experimental results, and we discuss applications and extensions of our algorithm, other than denoising. Finally, in section 12 we present our conclusions.

2 Mesh Signals

A graph G, composed of a set of vertices V, and a set of edges E can be directed or undirected. In addition to vertices and edges, a polygon mesh M includes a set of faces F. For simplicity, we only consider faces defined as cycles of vertices. The undirected graph of a mesh M is composed of the set of mesh vertices and the set of mesh edges as unordered pairs. In the directed case, where the edges of G are ordered pairs of vertices, every edge of M corresponds to two oriented edges of G. We use the symbols V, E, and F not only to denote the *sets* of vertices, edges, and faces, but also for the *number* of vertices, edges, and faces. The correct meaning can always be deduced from the context.

In this paper we look at the vertices of M in two ways. One way is as a three-dimensional *graph signal* $x = (x_1, \ldots, x_V)^t$ defined on G. In general, a D-dimensional graph signal on a graph G is a $D \times V$ matrix $x = (x_1, \ldots, x_V)^t$, where each row of x is regarded as the signal value at the i-th. vertex of the graph. We are mainly interested in the cases $D = 1$ and $D = 3$. The other way we look at a D-dimensional graph signal on a graph G is as a DV column vector x constructed concatenating the V vectors $x_1, \ldots, x_V$ of dimension D on top of each other.

In this paper we will also look at the *face normals* of M as a graph signal on the *dual graph* of M and with values in the unit sphere, and more generally on vector fields of arbitrary dimensionality defined on the faces of the mesh.

The edges of M are classified as *boundary*, *regular*, or *singular* depending on the number of incident faces. A boundary edge has one incident face, a regular edge has two incident faces, and a singular edge has three or more incident faces. The dual graph of M has the faces of M as vertices, and the regular edges of M as edges. The *boundary faces* of M are those incident to one or more boundary edges.

A *neighborhood* or *star* of a vertex index i in the graph G is the set $i^\star$ of vertex indices j connected to i by an edge (i, j).

$$i^\star = \{j : (i, j) \in E\} \ .$$

If the index j belongs to the neighborhood $i^\star$, we say that j is a *neighbor* of i. The neighborhood structure of an undirected graph, such as the graph of a mesh defined above, are symmetric. That is, a vertex j is a neighbor of a vertex i, if and only if i is a neighbor of j.

```
LaplacianOperator (G, W, x)
  if (non-linear)
    W = ComputeWeights (x)
  new Δx = 0
  for (e = (i, j) ∈ E)
    Δx_i = Δx_i + w_ij (x_j − x_i)
  end
return Δx
```

Figure 3: Algorithm to evaluate the Laplacian operator. $G = (V, E)$ directed graph, W matrix of weights defined on the edges of G, x input signal on G, Δx output signal. The isotropic scalar weight w_{ij} is replaced by a matrix weight W_{ij} in the anisotropic case.

3 Anisotropic Laplacian Operator

The Laplacian operator can be *isotropic* or *anisotropic*, and independently *linear* or *non-linear*. The four combinations are possible. As Weickert points out [22], sometimes in the image processing literature a filter that we would call non-linear isotropic, is called anisotropic. This is also the case in some of the recent work on anisotropic mesh processing [4, 2, 14, 15]

In this paper the *isotropic Laplacian operator* is defined on a graph signal x with values in Euclidean space by weighted averages over the neighborhoods

$$\Delta x_i = \sum_{j \in i^\star} w_{ij} \, (x_j - x_i) \ , \tag{1}$$

where the *edge weights* w_{ij} are non-negative *scalars* that add up to one for each vertex star

$$\sum_{j \in i^\star} w_{ij} = 1 \ . \tag{2}$$

One way to impose these constraints is to define the weights as normalized *edge costs* $w_{ij} = c_{ij}/c_i$, where $c_{ij} \geq 0$ is an edge cost, and $c_i = \sum_{j \in i^\star} c_{ij} > 0$ is a *vertex cost* equal to the total cost of edges incident to the i-th. vertex. The simplest choice here is to make all the edge costs equal to one $c_{ij} = 1$, i.e., to make the weight w_{ij} equal to the inverse of the number of neighbors $1/|i^\star|$ of the i-th. vertex. Other non-linear choices of isotropic weights are discussed in section 7.

We define the *anisotropic Laplacian operator* on a $3D$ graph signal x also by weighted averages over the neighborhoods

$$\Delta x_i = \sum_{j \in i^\star} W_{ij} \, (x_j - x_i) \ , \tag{3}$$

but here the edge weights W_{ij} are symmetric and non-negative definite 3×3 matrices such that their sum $\sum_{j \in i^\star} W_{ij}$ has eigenvalues in the interval $[0, 1]$. One way to construct these weight matrices is as functions of edge and vertex costs, as in the isotropic case. Here the edge costs C_{ij}, and the vertex cost C_i, are 3×3 symmetric and non-negative definite matrices such that $C_i \geq \sum_{j \in i^\star} C_{ij}$ as quadratic forms (i.e., such that $\forall x \in \mathbb{R}^3 : x^t C_i x \geq \sum_{j \in i^\star} x^t C_{ij} x$). For example, if c_i is an upper bound for the eigenvalues of $\sum_{j \in i^\star} C_{ij}$, we can take $C_i = c_i I$, which is what we have used in our implementation (see section 7). To simplify the formulation, we will also assume that C_i is non-singular, which is true in the

previous example, and set $W_{ij} = L_i^{-1} C_{ij} L_i^{-t}$, where L_i is the Cholesky decomposition [7] of C_i, i.e. the unique lower triangular matrix such that $L_i L_i^t = C_i$.

Both in the isotropic and anisotropic cases we call the Laplacian operator *linear* when the weights are constants, and *non-linear* when the weights are computed as an additional function of the signal values. Figure 3 describes the algorithm to evaluate the different Laplacian operators defined in this section on a signal x defined on a directed graph G, with given weight matrix W. The isotropic weights discussed in section 7 are all non-linear.

4 Fourier Analysis on Meshes

Fourier analysis can be used to understand and predict the behavior of linear operators [18, 21]. Since the isotropic Laplacian operator $x \mapsto \Delta x$ is linear on the space of graph signals defined on G, and operates independently on each of the coordinates of x, it is sufficient to analyze the case of one-dimensional graph signals. If we define the matrix $K = I - W$, with I the identity matrix, the isotropic Laplacian operator applied to a graph signal x can be written in matrix form as follows

$$\Delta x = -K\, x\ . \tag{4}$$

For undirected graphs and weights defined by symmetric costs ($c_{ij} = c_{ji}$), the matrix K has real eigenvalues $0 \leq k_1 \leq k_2 \leq \cdots \leq k_V \leq 2$ with corresponding linearly independent real unit length right eigenvectors $e^1, \ldots, e^V$. In matrix form

$$K\, E = E\, D\ , \tag{5}$$

with $E = (e^1, \ldots, e^V)$, $k = (k_1, \ldots, k_V)^t$, and D the diagonal matrix with k_i in its i-th. diagonal position. Seen as one-dimensional graph signals, these eigenvectors can be considered as the *natural vibration modes* of the graph, and the corresponding eigenvalues as the associated *natural frequencies*.

Since $e^1, \ldots, e^V$ form a basis of V-dimensional space, every graph signal x can be written as a linear combination

$$x = \sum_{j=1}^{V} \hat{x}_j\, e^j = E\, \hat{x}\ . \tag{6}$$

The vector of coefficients $\hat{x}$ is the Discrete Fourier Transform (DFT) of x, and E is the Fourier Matrix.

The analysis for the anisotropic Laplacian operator is similar, but requires the treatment of the 3 coordinates of the graph signal at once. In this case we look at the signal x not as a matrix but as a $3V$-dimensional vector; the matrix K is a $V \times V$ block matrix, with each block a 3×3 matrix:

$$K_{ij} = \begin{cases} I - W_{ii} & \text{if} \quad i = j \\ -W_{ij} & \text{if} \quad i \neq j \end{cases}$$

The rest of the analysis is exactly the same as for the isotropic case, except that here K has $3V$ eigenvalues and eigenvectors, and each eigenvector simultaneously defines the 3 coordinates of all the vertices of the mesh.

5 Mesh Filters

A *Linear Filter* is defined by a univariate function $f(k)$ that can be evaluated on the square matrix K to produce another matrix $f(K)$ of the same size. Although many functions of one variable can be evaluated in matrices [7], in this paper we only consider polynomials [21]. The function $f(k)$ is the *transfer function* of the filter.

```
LaplacianSmoothing(G, W, N, λ, x)
  new Δx
  for (h = 0 ... N − 1)
    Δx = LaplacianOperator(G, W, x)
    for (i = 0 ... V − 1)
      x_i = x_i + λ Δx_i
    end
  end
return
```

Figure 4: The Laplacian Smoothing Algorithm. G graph, W matrix of weights defined on the edges of G, N number of iterations, λ scaling factor, x signal on G to be smoothed. The algorithm is the same, whether the Laplacian operator is linear or non-linear cases, and isotropic or anisotropic.

It is well known that for any of these functions, the matrix $f(K)$ has the eigenvectors $e^1, e^2, \ldots$ of the matrix K as eigenvectors, and the result $f(k_1), f(k_2), \ldots$ of evaluating the function on the eigenvalues of K as eigenvalues.

Since for any polynomial transfer function

$$x' = f(K)\, x = \sum_i f(k_i)\, \hat{x}_i\, e^i\ ,$$

because $K e^i = k_i e^i$, to define a low-pass filter we need to find a polynomial such that $f(k_i) \approx 1$ for low frequencies, and $f(k_i) \approx 0$ for high frequencies in the region of interest $k \in [0, 2]$.

Figure 4 describes the Laplacian smoothing algorithm, with a scaling factor $0 < \lambda < 1$ which is used to control the speed of the diffusion process. With this parameter, one step of the linear isotropic Laplacian smoothing algorithm can be described in matrix form as follows

$$x^1 = x + \lambda \Delta x = (I - \lambda K)\, x = f(K)\, x\ , \tag{7}$$

where $f(K)$ is a matrix obtained by evaluating the univariate polynomial $f(k) = 1 - \lambda k$ in the matrix K. If the process is iterated N times, the output can still be expressed as $x^N = f(K)\, x$, but with a different univariate polynomial $f(k) = (1 - \lambda k)^N$. When $0 < \lambda < 1$, we see that for every $k \in (0, 2]$, we have $(1 - \lambda k)^N \to 0$ when $N \to \infty$ because $|1 - \lambda k| < 1$. This means that all the frequency components, other than the zero frequency component (the barycenter of all the vertices), are attenuated for large N. On the other hand, the neighborhood normalization constraint of equation 2 implies that the matrix K always has 0 as its first eigenvalue with associated eigenvector $(1, \ldots, 1)^t$, and the zero frequency component is preserved without changes because $f(0) = 1$ independently of the values of λ and N. A similar analysis applies to the linear anisotropic case.

Taubin [18] proposed the following second degree transfer function to solve the problem of shrinkage

$$f(k) = (1 - \lambda_0 k)(1 - \lambda_1 k)\ , \tag{8}$$

which can be implemented as two consecutive steps of Laplacian smoothing with different scaling factors; the first one with $\lambda_0 > 0$, and the second one with $\lambda_1 < -\lambda_0 < 0$.

```
MeshFilter(G, W, N, λ, x)
  new Δx
  for (h = 0 ... N − 1)
    Δx =LaplacianOperator (G, W, x)
    for (i = 0 ... V − 1)
      x_i = x_i + λ_i Δx_i
    end
  end
return
```

Figure 5: A mesh filter more general than Laplacian smoothing can be implemented by making the scaling factor λ function of the iteration number. Compare to the algorithm in figure 4. Again, the same algorithm applies to the linear or non-linear Laplacian operator, and to the isotropic or anisotropic cases.

To implement a more general linear filter, and by doing so being able to solve the shrinkage problem, we make the scaling factor λ dependent on the iteration number i. That is, the scaling factor is now a vector $\lambda = (\lambda_0, \ldots, \lambda_{N-1})^t$, and the transfer function is

$$f(k) = \prod_{i=0}^{N-1} (1 - \lambda_i \, k) \; . \tag{9}$$

A different way to implement FIR linear filters based on classical digital filter design techniques and Chebyshev polynomials was introduced by Taubin et.al. [21].

Ignoring numerical errors, in the linear case, the N Laplacian smoothing steps of equation 9 can be performed in any order, without affecting the result. Since the weights of the non-linear Laplacian operator are computed as a function of the signal values, in the non-linear case we have to be more precise. The same equation 9 can be used, but the non-linear Laplacian smoothing steps have to be performed in the specified order, as shown in the algorithm of figure 5.

6 Preventing Boundary Shrinkage

Although the normal vector to a polygon mesh is not defined at a vertex, it is customary to define it by averaging some local information for shading purposes. When the signal x in equation (1) corresponds to the 3-dimensional mesh vertex positions, the Laplacian operator can be used to define a normal vector

$$n_i = \begin{cases} \frac{\Delta x_i}{\| \Delta x_i \|} & \text{if} \quad \Delta x_i \neq 0 \\ 0 & \text{if} \quad \Delta x_i = 0 \end{cases} \tag{10}$$

When the length of edges incident to vertex i and the angles formed by these edges are all similar, the mesh is a discretization of a smooth surface, and the vertex is not on the boundary of the surface, this vector average approximates the *curvature normal*, modulo a scale factor function of the average edge length (sampling wavelength). In fact, the following expression can be used as the definition of the *mean curvature* κ_i [17] of the graph signal x at the vertex i

$$\frac{1}{2} \kappa_i \, n_i \, e_i^2 = \sum_{j \in i^\star} w_{ij} \, (x_j - x_i) \; , \tag{11}$$

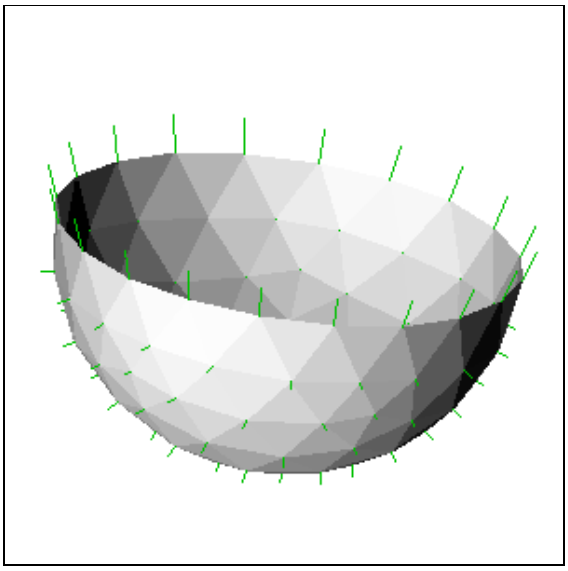


Figure 6: The boundary shrinkage problem. Laplacian operator defines vertex displacements that approximate curvature normal at non-boundary vertices. The displacements have strong tangential components at boundary vertices. The negative of the Laplacian operator displacements $-\Delta x_i$ are plotted because they easier to visualize.

independently of the subjacent smooth surface, where e_i^2 is the average square length from vertex i to its neighbors

$$e_i^2 = \sum_{j \in i^\star} w_{ij} \, \| \, x_j - x_i \, \|^2 \; .$$

When the vertex i is on the boundary of the mesh, the vector Δx_i has a very strong tangential component, and the vector of equation 10 cannot be considered a good approximation of the surface normal. This problem is illustrated in figure 6. As a result, when an isotropic filter is applied to a mesh with boundary, while non-boundary vertices tend to move along the normal direction, boundary vertices tend to move in a tangential direction, producing shrinkage, even when the filter is a low-pass filter such as the $\lambda|\mu$ algorithm.

The hierarchical smoothing approach described by Taubin [18], where the boundary vertices are filtered as a curve independently of the rest of the mesh, is one way to prevent boundary shrinkage. But because the boundary and mesh smoothing processes proceed at different speeds, the boundary curves may be over-smoothed, and there is no control on the surface normals along the boundary.

The following alternative solution to the boundary shrinkage problem produces better results, and is a good motivation to the definition of the anisotropic Laplacian operator given above. In this case we need an additional vertex normal vector n_i for each boundary vertex i, and we redefine the Laplacian operator for the boundary vertex i as the projection of the old Laplacian operator onto line defined by the normal vector n_i

$$\Delta x_i = n_i \, n_i^t \sum_{j \in i^\star} w_{ij} \, (x_j - x_i) \; .$$

The boundary vertex normal vectors can either be explicitly provided as constraints, or computed by averaging face normal vectors to incident faces.

Equivalently, we can define the following anisotropic weights

$$W_{ij} = \begin{cases} w_{ij}\, n_i\, n_i^t & \text{if} \quad i \text{ is a boundary vertex} \\ w_{ij} \quad I & \text{if} \quad i \text{ is not a boundary vertex} \end{cases}$$

where w_{ij} is the old isotropic weight. The resulting linear anisotropic diffusion algorithm removes noise but the boundaries do not shrink.

7 Preventing Tangential Drift

The boundary shrinkage problem mentioned above is a particular case of the tangential drift problem. With unit edge costs, very satisfactory results are obtained with linear isotropic FIR filters [21] on meshes with no boundaries displaying very small variation in edge length and face angles across the whole mesh. When these assumptions are not met, local distortions are introduced, mainly because the Laplacian operator defines vertex displacement vectors with strong tangential components. The edge weights can be used to compensate for the irregularities of the tessellation, and produce results which are function of the local geometry of the signal, rather than the local parameterization. For example, shorter edges have to be given larger weights. The following *non-linear* strategies to compute weights have been proposed.

Fujiwara weights [6] try to compensate for irregular edge lengths by determining the edge costs as a function of the edge length $c_{ij} = \phi(\|v_j - v_i\|)$. For example, Fujiwara proposed the inverse of the edge length $\phi(t) = 1/t$ as the function, which makes the Laplacian operator independent of the edge lengths, and only dependent on the directions of the vectors pointing to the neighboring vertices. This weighting scheme does not solve the problems arising from unequal face angles.

Desbrun weights [4] compensate not only for unequal edge lengths, but also for unequal face angles. Laplacian smoothing with equal edge costs tends to equalize the lengths of the edges, and so, tends to make the triangular faces equilateral. The vertex displacements produced by the Laplacian operator can be decomposed into a normal and a tangential component. In some cases the edge equalization is the desired effect, such as when mesh smoothing is used to improve the quality of finite-elements mesh. But in other cases, such as when a texture is mapped onto the mesh, having a non-zero tangential component is undesirable. Based on a better approximation to the curvature normal, Desbrun et.al. proposed the following choice of edge costs

$$c_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \ , \tag{12}$$

where α_{ij} and β_{ij} are the two angles opposite to the edge $e = (i, j)$ in the two triangles having e in common. This choice of weights produces no tangential drift when all the faces incident to the vertex are coplanar.

The weighting schemes described above can be applied to a FIR filter, but both Fujiwara weights and Desbrun weights must be recomputed after each iteration, or at least after a small number of iterations. This makes the whole smoothing process a nonlinear operation, and computationally more expensive. The alternative solution that we propose in this paper is based on using a linear anisotropic Laplacian operator where the edge costs are computed as a function of the face normals, in such a way that tangential displacements are penalized. For example, if vertex j is a neighbor of vertex i, we can define

$$C_{ij} = \sum_{f \in F_{ij}} c_{fij}\, n_f\, n_f^t \ , \tag{13}$$

where F_{ij} is the set of faces incident to the edge joining vertices i and j, n_f is the unit length normal vector to face f, and $c_{fij} \geq 0$ is a non-negative cost. In this paper we use $c_{fij} = 1$ independently of f, i, or j, $C_i = c_i I$, and $c_i = \sum_{j \in i^\star} \sum_{f \in F_{ij}} c_{fij}$, which produces excellent results. Note that for manifold meshes the set F_{ij} is composed of only one or two elements. Another possible choice is to make c_{fij} equal to the area of face f, but this would make the cost a non-linear function of the vertex coordinates, which we have been trying to avoid in this paper.

```
LaplacianOperatorNormals(G, W, n)
  new Σ = 0
  for (e = (i, j) ∈ E)
    Σ_i = Σ_i + w_ij n_j
  end
  for (i = 0 ... N − 1)
    Σ_i = n_i × Σ_i
  end
return Σ

MeshFilterNormals(G, W, N, λ, n)
  new Σ
  for (h = 0 ... N − 1)
    Σ =LaplacianOperatorNormals(G, W, n)
    for (i = 0 ... V − 1)
      n_i = R(λ_h Σ_i) n_i
    end
  end
return
```

Figure 7: Algorithms to evaluate the Laplacian operator, and a general linear filter on a graph signal with values the unit sphere. Laplacian smoothing corresponds to the case of constant λ_h.

8 Laplacian Smoothing on the Sphere

In order for this linear anisotropic filter to produce the desired results, we need a smooth face normal field. Since the field of face normals of a noisy mesh is even noisier, we now have to solve the problem of smoothing a field of unit-length face normal vectors. We regard such vector field as a signal defined on the dual graph of the mesh, and with values in the unit sphere. We solve this problem again with Laplacian smoothing, or a more general FIR filter, based on a redefined Laplacian operator for signals with values in the unit sphere, and then use the algorithms described in figure 7.

The Laplacian operator can be applied to unit length vectors, but the result of one Laplacian smoothing step

$$n_i' = n_i + \lambda \sum_{j \in i^\star} w_{ij}\, (n_j - n_i)$$

is no longer a unit length vector. This can be fixed by normalizing the length of the resulting vector, but the results are poor, particularly when the angles between neighboring normals are large. A better solution is to look at the expression $\lambda\, w_{ij}\, (n_j - n_i)$ as a displacement along a *geodesic* on the

sphere, i.e., along a great circle, and at the scaled Laplacian vector $\lambda\Delta n_i$ as the average of these displacements in a well-defined local chart.

The resulting displacement $\lambda\Delta n_i$ is represented by a rotation $R_i(\lambda)$, where the function $R_i(\lambda)$ is a continuous function of λ such that when $\lambda = 0$ the rotation is the identity $R_i(\lambda) = I$. For each neighbor j of i such that n_i and n_j are linearly independent, there are two rotations that keep the plane defined by the vectors n_i and n_j invariant . Of these two let R_{ij} be the rotation of minimum angle. If $n_i = n_j$ we define $R_{ij} = I$, and we will assume that the case $n_i = -n_j$ never happens. In section 9 we show that the rotation R_{ij} can be computed using Rodrigues' formula as

$$R_{ij} = R(n_i \times n_j) \ ,$$

where $R(\omega)$ is the Rodrigues function that maps vectors of length not larger than one onto rotation matrices

$$R : \{\omega : |\omega| \leq 1\} \rightarrow \mathrm{SO}(3) \tag{14}$$

Now we can define the Laplacian operator for signals with values in the unit sphere as follows

$$R_i(\lambda) = R\left(\lambda\, n_i \ \times \sum_{j \in i^\star} w_{ij}\, n_j\right) \tag{15}$$

for $\lambda \leq 1$. We do all the scaling and averaging in the domain of $R(\omega)$ which is a convex subset of Euclidean space that contains the origin, and so close under scaling ($|\lambda| \leq 1$) and convex averaging ($\sum_j w_{ij} = 1$).

Finally, we define a Laplacian smoothing step as follows

$$n_i' = R_i(\lambda)\, n_i \tag{16}$$

Now we can implement any FIR filter using the polynomial factorization of equation 9. For example, one step of Taubin smoothing becomes

$$n_i'' = R_i(\lambda_1)\, R_i(\lambda_0)\, n_i$$

9 Rodrigues Formula

A 3D rotation can be represented as a 3×3 matrix

$$R = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}$$

orthogonal $RR^t = I$ and with unit determinant $|R| = 1$. In particular, the inverse rotation is represented by the transposed matrix $R^{-1} = R^t$. The result of applying the rotation represented by the matrix R to a vector v is computed by multiplying the matrix by the vector Rv. The group of 3D rotation matrices, also called *special orthogonal group of dimension 3*, is usually denoted $\mathrm{SO}(3)$.

A more intuitive way of representing a 3D rotation is by specifying an *axis of rotation* with a unit length vector r, and an *angle of rotation* α in radians. Usually, these two values are jointly specified as a single vector $\omega = \alpha r$. The angle of rotation is the length of ω, and the axis of rotation is determined by normalization $r = \omega/|\omega|$. In the case $|\omega| = 0$, which corresponds to the identity matrix, the unit vector is not uniquely determined.

Since this representation produces multiple representatives for each rotation, and the conversion to matrix form requires computation of trigonometric functions, it is better to represent the rotation as a vector with length equal to the positive *sine* of the angle of rotation. The conversion from vector to matrix representation is given by *Rodrigues formula*

$$R(\omega) = c\, I + (1 - c)\, rr^t + s\, r^\Lambda \ , \tag{17}$$

where $s = (\|\omega\|^2)^{1/2} = \sin(\alpha)$, $c = (1 - \|\omega\|^2)^{1/2} = \cos(\alpha)$, $sr = \omega$, and

$$r^\Lambda = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} \ .$$

This last matrix is *skew-symmetric* $(r^\Lambda)^t = -r^\Lambda$, and represents the *vector product*: when we multiply the matrix r^Λ by a vector v we obtain the vector product $r \times v$

$$r^\Lambda\, v = r \times v = \begin{pmatrix} r_y\, v_z - r_z\, v_y \\ r_z\, v_x - r_x\, v_z \\ r_x\, v_y - r_y\, v_x \end{pmatrix} \ .$$

Rodrigues' formula defines the function $R : \{\omega : |\omega| \leq 1\} \rightarrow \mathrm{SO}(3)$ which is clearly $1-1$ and continuous for $\omega \neq 0$, but it is not difficult to show that it is well defined and continuous at $\omega = 0$ as well.

To convert from matrix to vector representation we first note that, since $c\, I + (1 - c)\, rr^t$ is symmetric, and $s\, r^\Lambda$ skew-symmetric, the transpose of the matrix R in equation 17 is

$$R^t = c\, I + (1 - c)\, rr^t - s\, r^\Lambda \ ,$$

and so, the skew-symmetric part of R is

$$(R - R^t)/2 = s\, r^\Lambda = (s\, r)^\Lambda \ . \tag{18}$$

The value of s is obtained as the length of the vector $\omega = s\, r$. The value of c is obtained by computing a square root $c = \sqrt{1 - s^2}$. With c and s we can determine α, if needed. And the unit length vector r is obtained by normalization, except when $R - R^t = 0$, which corresponds to the identity rotation $R = I$. This case has multiple solutions. So, the inverse of Rodrigues' formula of equation 18 defines the inverse function

$$\omega = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \frac{1}{2} \begin{pmatrix} R_{zx} - R_{xz} \\ R_{xz} - R_{zx} \\ R_{yx} - R_{xy} \end{pmatrix} \ .$$

If n_i and n_j are two unit length vectors such that $n_i + n_j \neq 0$, it is not difficult to show that

$$R(n_i \times n_j)\, n_i = n_j \ .$$

10 Integration of Face Normal Fields

In this section we first study the existence and uniqueness of solution to the problem of reconstructing the mesh vertex positions from a given field of face normals. Then we formulate an orthogonal decomposition theorem that explains the least squares solution of the previous problem in the case where an exact solution does not exists. Finally, we discuss how our algorithm relates to the solution of these problems. Our approach is totally self-contained and based on basic concepts from linear algebra.

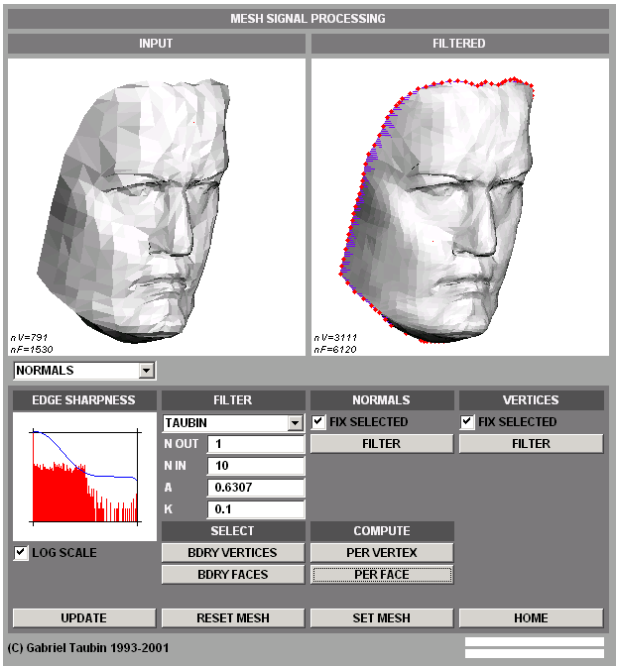


Figure 8: Java implementation of the algorithms introduced in this paper.

10.1 Integrability

We restrict our analysis here to the case of triangle meshes, where the notation is simpler. The results can be extend to general polygon meshes, but we leave the details of the extension to the reader, or a subsequent paper.

The problem is this: given a non-zero vector u_f for each face $f = (i, j, k)$ of a triangular mesh M, we want to describe the set S of vertex positions $(x_1, \ldots, x_n)$ so that, for each triangular face f, the vector u_f is orthogonal to the plane defined by the points $\{x_i, x_j, x_k\}$, and in what sense our algorithm helps find the elements of S. In principle, the set S may be empty (i.e., the field is not integrable), may be composed of a single element (i.e., the problem has a unique solution), may be composed of a finite number of element (i.e., the problem has a multiple solutions), or may be composed of an infinite number of elements (i.e., the problem has free degrees of freedom).

Unconstrained case We first consider the case where no further constraints are imposed on the vertex positions. The face orthogonality conditions described above are equivalent to the following family of simultaneous linear equations

$$\forall\, f = (i,j,k) \quad : \quad \left\{ \begin{array}{rcl} u_f^t\,(x_j - x_i) & = & 0 \\ u_f^t\,(x_k - x_j) & = & 0 \\ u_f^t\,(x_i - x_k) & = & 0 \end{array} \right. \tag{19}$$

Note that for each face f, the corresponding three equations are linearly dependent because their sum is equal to zero. As a result, we can discard one of the three equations, and obtain an homogeneous system of $2F$ equations in $3V$ variables, which we rewrite in matrix form as follows

$$\begin{pmatrix} & \vdots & \vdots & \\ \cdots & -u_f & u_f & \cdots \\ & \vdots & \vdots & \\ \cdots & u_f & 0 & \cdots \\ & \vdots & \vdots & \\ \cdots & 0 & -u_f & \cdots \\ & \vdots & \vdots & \end{pmatrix}^t \begin{pmatrix} \vdots \\ x_i \\ \vdots \\ x_j \\ \vdots \\ x_k \\ \vdots \end{pmatrix} = 0\,. \tag{20}$$

If U denotes the $3V \times 2F$ left matrix, and X the $3V$-dimensional right vector, we have $U^t X = 0$. Since the equations are linear an homogeneous, the space of solutions is a subspace in $\mathbb{R}^{3V}$. Note that the dimension of this subspace is at least 3, because the constant vector $x_i = x$ for $i = 1, \ldots, V$ is a solution of our problem for any point $x \in \mathbb{R}^3$. Since the problem is invariant with respect to translations, we can reduce its dimensionality by imposing the constraint $x_{V-1} = 0$. A solution of the original system can be decomposed as a solution of the reduced system plus a 3D translation. Let $\tilde{U}$ be the $(3V-3) \times 2F$ matrix obtained from U by deleting the last row, and let $\tilde{X}$ be the $(3V-3)$-dimensional right vector obtained from X by deleting the last three coordinates corresponding to x_{V-1}. Now, our original problem has a solution if and only if the system $\tilde{U}^t \tilde{X} = 0$ has a non-trivial solution, and the solution is unique if and only if the dimension of the subspace of solutions is 1. Equivalently, if $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots$ are the singular values of $\tilde{U}$, the problem has a solution if $\lambda_1 = 0$ as well, and the solution is unique if $\lambda_2 > 0$. If $\lambda_1 = \cdots = \lambda_h = 0$ and $\lambda_{h+1} > 0$, the problem has multiple solutions, and the subspace of solutions has dimension h. That is, the problem has h degrees of freedom in addition to the 3 degrees of freedom corresponding to the invariance to translation. A unique solution can be obtained by imposing further independent linear constraints. Also note that since the problem is homogeneous, the solutions can only be obtained modulo scale.

In the overconstrained case $3V - 3 \leq 2F$, the problem may have no solution, a unique solution (modulo scale), or multiple degrees of freedom. In the last case, the general solution can be represented as a linear combination of the singular vectors corresponding to the zero singular values. In the underconstrained case $3V - 3 > 2F$, when we have fewer equations than unknowns, the dimension of the subspace of solutions is at least $3V - 3 - 2F > 0$, and the problem always has at least one solution.

If the triangular mesh is manifold without boundary, the numbers of vertices V, edges E, and faces F satisfy the Euler formula $V - E + F = 2(1 - G)$ where $G \geq 0$ is the genus of the mesh. For example, a sphere has genus 0 and a torus has genus 1. Since each triangular face of a manifold mesh without boundary corresponds to three half-edges, or $3F = 2E$, we obtain a necessary and sufficient condition for the problem to be overconstrained

$$V - 3 \leq 2F \quad \Leftrightarrow \quad 2 - 4G \leq F\,.$$

For example, for a mesh with sphere topology we get $F \geq 2$, which is always satisfied because the tetrahedron is the smallest in this class with $F = 4$. For higher genus, the left term is negative, and the condition is always satisfied. If the triangular mesh is manifold with boundary, a similar necessary and sufficient condition can be derived involving in

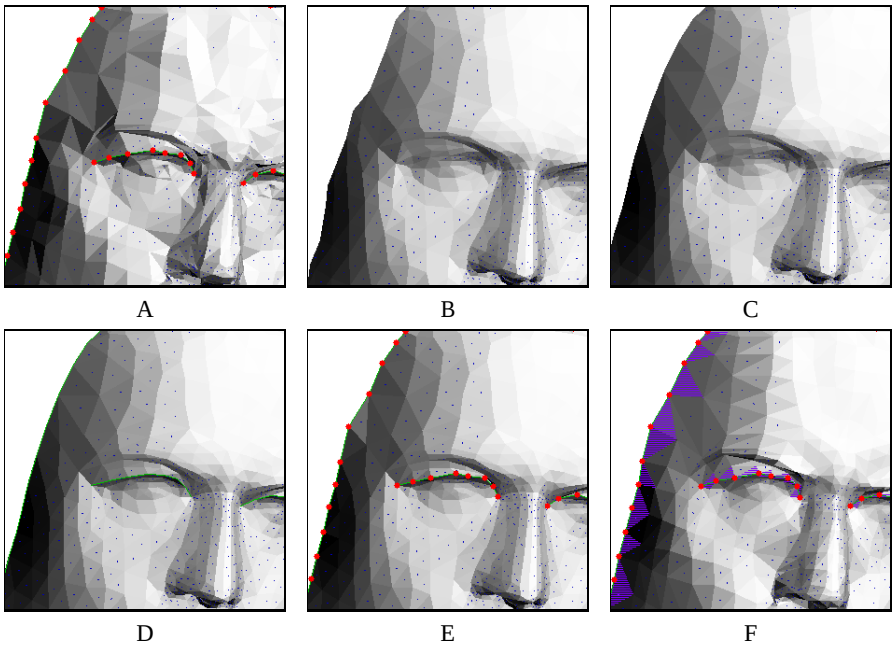


Figure 9: Hierarchical smoothing can be used to introduce crease lines and to control their shape. (A) Mesh with tagged boundary and crease edges. (B) The result of applying 10 steps of Taubin smoothing without constraints. Observe the shrinkage of the boundary and the over-smoothing of the crease lines. (C) The result of applying 10 steps of Taubin smoothing with hierarchical constraints, but ignoring the crease lines. The boundary curve is smoothed independently of the interior vertices. (D) Same as (C) but taking the crease edges into account. (E) The result of applying 10 steps of Taubin smoothing with fixed vertices on the tagged edges. Noise present on the boundary vertices remains. (F) The result of applying the new algorithm introduced in this paper with fixed face normals and vertices incident to tagged and boundary edges.

addition the number of boundary loops L and the number of boundary edges B, but it is a lot easier just to check the original inequality. In this case the system can be underconstrained. For example, look at the extreme case of F disconnected triangles with $V = 3F$ vertices.

Constrained case In the case of triangular meshes with boundary, where a large number of boundary edges may result in an underconstrained system of equations, further constraints can be imposed by specifying, for example, the positions of some or all of the boundary vertices. In general, we will consider the case when C of the vertices are constrained to be in specified locations. Since the ordering of the vertices is irrelevant, we will assume without loss of generality that the vertices $x_1, \ldots, x_C$ are constrained, and the vertices $x_{C+1}, \ldots, x_V$ are free. Since the invariance to translation no longer applies here, we need to look at the original system of equations $U^t X = 0$. We partition the matrix U into two matrices U_1 and U_2. The first one composed of the first $3C$ rows of V, and the second one composed of the remaining rows. We also partition the vector X into X_1 and X_2 in a similar manner. The system to be solved can be written as follows

$$U_1^t X_1 + U_2^t X_2 = 0 \ ,$$

where U_1, U_2, and X_1 are given. For the system not to be underconstrained, we need to have at least as many equations as variables, i.e., $3(V - C) \leq 2F$. In this case the problem has a unique solution if and only if the rank of the matrix U_2 is equal to $3(V - C)$, and the solution is

$$\hat{X}_2 = -U_2^{t\dagger} V_1^t X_1$$

with $U_2^{t\dagger} = (U_2 U_2^t)^{-1} U_2$ the *pseudo-inverse* of U_2^t. If the rank of the matrix U_2 is less than $3(V - C)$ the problem still has free degrees of freedom, and if the rank is larger than $3(V - C)$ the problem has no solution. In this case the least squares solution is also described by the previous equation.

10.2 Orthogonal Decomposition

Helmholtz decomposition theorem, used extensively in electromagnetism, states that every smooth vector field in $\mathbb{R}^3$ can be decomposed as the sum of two components: a rotation-free vector field (gradient of a scalar potential) and a divergence-free vector field (curl of a potential vector field)

$$F = \nabla \phi + \nabla \times B \ .$$

For vector fields defined on surfaces, Hodge decomposition theorem states that every vector field on a surface can be decomposed into three components: a rotation-free vector field, a divergence-free vector field, and a harmonic vector field (with zero Laplacian). We seek a similar decomposition of vector fields defined on the faces of a mesh, where the concept of integrable face vector field on a mesh correspond to rotation-free smooth vector field on a surface.

In general, in the unconstrained case, given an arbitrary non-zero vector u_f for each face $f = (i, j, k)$ of the triangular mesh M, the system of equations 19 will have no solution. Equivalently, the first singular value λ_1 of the matrix $\tilde{U}$ is positive.

If vertex positions $(x_1, \ldots, x_n)$ are given and no triangular face is degenerate (zero surface area), the face vector field u_f can be decomposed as the sum of a normal face field and a

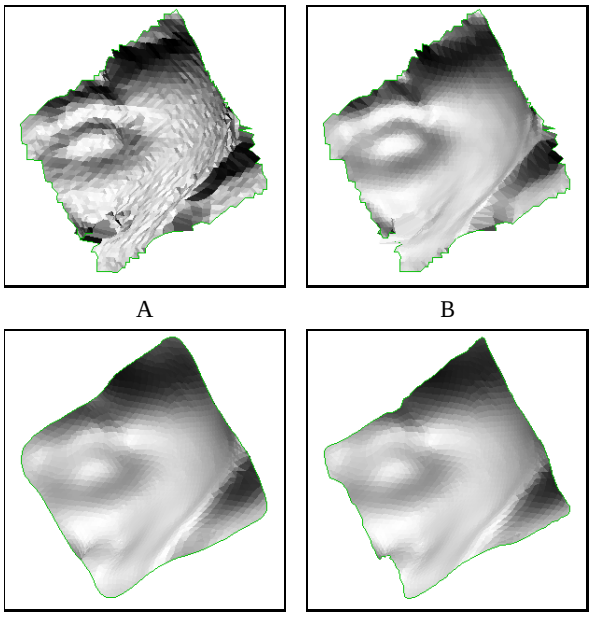


Figure 10: Pre-smoothing of boundary curves. (A) Noisy mesh with noisy boundary curve. (B) The result of applying the algorithm introduced in this paper with fixed boundary vertices and faces. (C) Same as (B) but with 2 steps of Laplacian smoothing along the boundary as a pre-processing phase. (D) (C) Same as (B) but with 2 steps of Taubin smoothing along the boundary as a pre-processing phase.

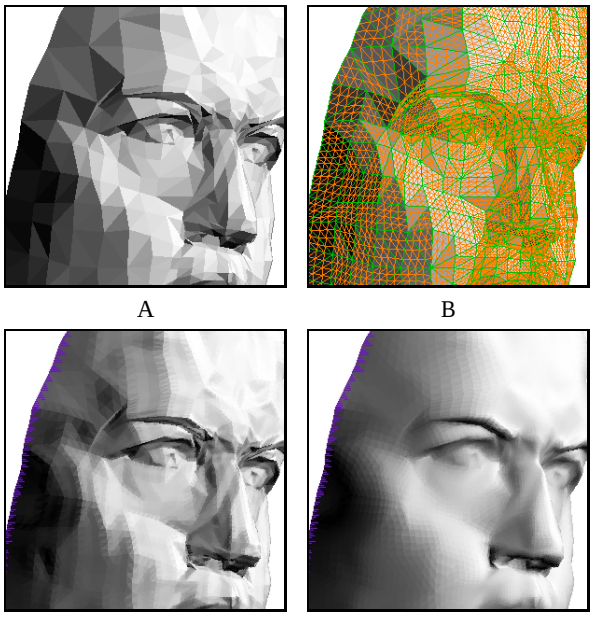


Figure 11: Application to mesh design in conjunction with recursive connectivity subdivision operators. (A) Coarse triangular mesh. (B) After two steps of linear triangle quadrisection with vertices inserted at the midpoints of edges. (C) After applying the algorithm introduced in this paper with 10 steps of Taubin smoothing on the face normal field, followed by 10 steps of linear anisotropic Laplacian smoothing on the vertex positions. (D) Same as (C) but using 10 steps of isotropic Laplacian smoothing to smooth the field of face normals.

tangent vector field. In general, even if some faces defined by the vertex positions are degenerate, we can define the following face vector field

$$n_f = \begin{cases} \frac{(x_j - x_i) \times (x_k - x_j)}{\|(x_j - x_i) \times (x_k - x_j)\|} & \text{if} \quad (x_j - x_i) \times (x_k - x_j) \neq 0 \\ 0 & \text{if} \quad (x_j - x_i) \times (x_k - x_j) = 0 \end{cases} \tag{21}$$

For a non-degenerate face n_f is a unit length normal vector to face f defined by the points $\{x_i, x_j, x_k\}$. Otherwise it is the zero vector. Now we can define the normal component of u_f as $u_{Nf} = n_f n_f^t u_f$ and the tangential component by $u_{Tf} = (I - n_f n_f^t) u_f$. If U_N and U_F are the $3V \times 2F$ matrices constructed as U in equation 20 from the face vector fields u_{Nf} and u_{Tf}, we have $U = U_N + U_T$ and $U_N^t U_T = 0$. In addition, by construction $U_N^t X = 0$ and the first singular value of $\tilde{U}_N$ is zero $\lambda_{N1} = 0$ corresponding to the singular vector $\tilde{X}$. The problem we are facing here is to find vertex positions X minimizing the tangential error $\|U_T^t X\|^2 / \|X\|^2$ defined by the decomposition described above.

Note that, since by construction $U_N^t X = 0$ and $U_N^t U_T = 0$, we have $\|U^t X\|^2 = \|U_T^t X\|^2$, and the solution to this problem is given by the least squares solution to the original problem. Solving the system of equations 19 in the least squares sense is equivalent to the minimization of the following quadratic function

$$\psi_u(x) = \|U^t X\|^2 = \sum_f \sum_{(i,j) \in \partial f} (u_f^t (x_i - x_j))^2 \ . \tag{22}$$

Since the equations are homogeneous, the first non-trivial solution corresponds to the singular vector associated with the first singular value λ_1 of $\tilde{U}$ (λ_4 of U). Equivalently,

$$\min_X \frac{\|U^t X\|^2}{\|X\|^2} = \lambda_1^2 \ .$$

Note that in general, the solution to this problem obtained in this way may define a mesh with degenerate triangular faces. We leave the problem of determining conditions for non-degenerate solutions for future study.

A similar line of reasoning can be followed in the constrained case, which includes additional linear constraints (fixed boundary vertex positions).

10.3 Integration Algorithms

The linear anisotropic Laplacian smoothing algorithm introduced in this paper to integrate a field of face normals is a descent algorithm for the quadratic function 22. The anisotropic Laplacian operator introduced in previous sections is the negative of the gradient of $\psi_u(X)$, scaled to make it converge in a stable fashion. Since the function being minimized is convex, our algorithm converges to the global minimum.

11 Implementation and Applications

The intended application of these algorithms is as tools in an interactive shape design system. Figure 8 shows a screen dump of our prototype implementation written in

Java, which integrates a number of related mesh processing algorithms. All the illustrations presented in this paper have been generated with this tool.

Overall, in the applications to noise removal a small number of iterations produce satisfactory results, even for very large meshes such as the one shown in figure 1. We envision applications to shape design, where the user interactively modify the face normal field, and then linear anisotropic Laplacian smoothing is used to integrate the field of face normals. In this case the initial position of the vertices may be very far away from their intended destination, and a large number of iterations may be needed to produce satisfactory results. This is so because the length of the displacement vector defined by the Laplacian operators at a vertex is never longer than the average length of edges incident to that vertex. In this case a multi-resolution approach is advisable [11, 12].

11.1 Extensions

We describe in this section a number of potential extensions and applications. Some are implemented and some are not. Those implemented are in different states of maturity.

Hierarchical smoothing The hierarchical smoothing approach described by Taubin [18] can be applied in conjunction with the algorithm introduced in this paper to achieve more precise control along boundaries and tagged discontinuity edges, and non-manifold meshes. In addition to the faces incident to the boundary edges, faces incident to other tagged edges can be constrained during the face normals smoothing process to produce interesting effects such as introduction and shape control of creases. Figure 9 shows an example of this application, comparing different variations of older algorithms and constraints with the new algorithm introduced in this paper.

Boundary smoothing If we constrain boundary vertices and faces not to move at all during the smoothing process, we may obtain undesirable results along noisy boundaries. Figure 10 shows additional ways to smooth boundary curves as a pre-processing step to produce more pleasing shapes.

Vertex normal fields Sometimes mesh normals are specified at the vertices, rather than the faces of the mesh. If the mesh is manifold, we can apply the algorithm introduced in this paper to reconstruct the position of the vertices of the *dual mesh*, and then reconstruct the positions of the primal vertices by dual mesh resampling [20], which is yet another variation of Laplacian smoothing.

Subdivision meshes The algorithm introduced in this paper can be used to design piecewise smooth meshes as a sequence of connectivity refinement and smoothing steps performed within and interactive modelling system [18]. Figure 11 shows some examples of this use in connection with triangle quadrisection as the connectivity refinement operator.

12 Conclusions

In this paper we introduced a new algorithm for polygon meshes smoothing with vertex position and face normal interpolatory constraints. The algorithm, composed of two phases where the face normals are first filtered, and the resulting smooth face normals are integrated, is based on simple extensions of Laplacian smoothing. In addition, we demonstrated explicit control of a variety of constraints for mesh design applications. In our view, the main advantage of this algorithm is its simplicity and its relation with the Laplacian smoothing family of algorithms that allows a clean, simple, efficient and elegant implementation.

References

[1] H. Bierman, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *Siggraph'2000 Conference Proceedings*, 2000.

[2] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing,. In *Proceedings of IEEE Visualization 2000*, October 2000.

[3] M. Desbrun, M. Meyer, P. Schroder, and A. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Proceedings of Graphics Interface 2000*, May 2000.

[4] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Siggraph'99 Conference Proceedings*, pages 317–324, August 1999.

[5] D.A. Field. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods*, 4:709–712, 1984.

[6] K. Fujiwara. Eigenvalues of laplacians on a closed riemannian manifold and its nets. *Proceedings of the AMS*, 123:2585–2594, 1995.

[7] G. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, 2nd. edition, 1989.

[8] K. Ho-Le. Finite element mesh generation methods: A review and classification. *Computer Aided Design*, 20(1):27–38, 1988.

[9] G. Husler and S. Karbacher. Reconstruction of smoothed polyhedral surfaces from multiple range images. In *Proceedings of 3D Image Analysis and Synthesis '97*, pages 191–198, Sankt Augustin, 1997. Infix Verlag.

[10] S. Karbacher and G. Husler. A new aproach for modeling and smoothing of scattered 3d data. In R.N. Ellson and H. Nurre, editors, *Proceedings of the SPIE on Three-Dimensional Image Capture and Applications*, volume 3313, pages 168–177, 1998.

[11] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Siggraph'98 Conference Proceedings*, pages 105–114, July 1998.

[12] L. Kobbelt, J. Vorsatz, and H.-P. Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry Theory and Applications*, 1999. special issue on multi-resolution modeling and 3D geometry compression.

[13] S. Kuriyama and K. Tachibana. Polyhedral surface modeling with a diffusion system. In *Eurographics'97 Conference Proceedings*, pages C39–C46, 1997.

[14] Y. Ohtake, A.G. Belyaev, and I.A. Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization. In *Proceedings of the Geometric Modeling and Processing 2000,* April 2000.

[15] Y. Ohtake, A.G. Belyaev, and I.A. Bogaevski. Mesh regularization and adaptive smoothing. *Computer Aided Design*, 2001.

[16] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 1990.

[17] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings, International Conference on Computer Vision (ICCV)*, 1995.

[18] G. Taubin. A signal processing approach to fair surface design. In *Siggraph'95 Conference Proceedings*, pages 351–358, August 1995.

[19] G. Taubin. Geometric signal processing on polygonal meshes. In *Eurographics 2000 State of The Art Report (STAR)*, September 2000.

[20] G. Taubin. Dual mesh resampling. In *Conference Proceedings, Pacific Graphics 2001*, Tokyo, Japan, October 2001.

[21] G. Taubin, T. Zhang, and G. Golub. Optimal surface smoothing as filter design. In *Fourth European Conference on Computer Vision (ECCV'96)*, 1996. Also as IBM Technical Report RC-20404, March 1996.

[22] J Weickert. *Scale-Space Theory in Computer Vision*, volume 1252 of *Lecture Notes in Computer Science*, chapter A Review of Nonlinear Diffusion Filtering, pages 3–28. Springer-Verlag, 1997.

[23] A. Yamada, T. Furuhata, K. Shimada, and K. Hou. A discrete spring model for generating fair curves and surfaces. In *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications*, pages 270–279, 1998.

[24] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Siggraph'97 Conference Proceedings*, pages 259–268, August 1997.