Geometric Signal Processing on Polygonal Meshes

G. Taubin†

IBM T.J. Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598
http://www.research.ibm.com/people/t/taubin

Abstract

Very large polygonal models, which are used in more and more graphics applications today, are routinely generated by a variety of methods such as surface reconstruction algorithms from 3D scanned data, isosurface construction algorithms from volumetric data, and photogrametric methods from aerial photography. In this report we provide an overview of several closely related methods developed during the last few yers, to smooth, denoise, edit, compress, transmit, and animate very large polygonal models.

1. Introduction

The geometric signal processing approach was originally motivated by the problem of smoothing large irregular polygonal meshes of arbitrary topology [36], such as those extracted from volumetric medical data by iso-surface construction algorithms, or constructed by integration of multiple range images, and the related problem of fair surface design. 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. This constraint on the complexity of the algorithms discards most early algorithms based on fairness norm optimization [42, 28, 13, 43], parametric [31, 26, 11, 25, 24] and implicit [1, 27] patch technology, physics-based deformable models [20, 41, 33, 30], and variational formulations [5, 28, 43, 13]. In these approaches, the problem is often reduced to the solution of a large sparse linear system, or a more expensive global optimization problem. Large sparse linear systems are solved using iterative methods [10], and usually result in quadratic time complexity algorithms. However, more recent work formulations have shown efficient solutions to the variational formulation based on multigrid algorithms [21, 22], and stable implicit sparse solvers that are competitive when agressive smoothing is required [7].

Most smoothing algorithms move the vertices of the polygonal mesh without changing the connectivity of the faces. The smoothed mesh has exactly the same number of vertices and faces as the original one. The simplest smoothing algorithm that satisfies the linear complexity requirement is Laplacian smoothing, described in detail in section 2. Laplacian smoothing is an iterative process, where in each step every vertex of the mesh is moved to the barycenter of its neighbors.

The only problem with Laplacian smoothing is *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 algorithm introduced by Taubin [36] solves this problem and introduced the signal processing machinery necessary to analyze the behavior of these smoothing processes. This work was followed by a number of extensions [40, 7] and applications to interactive shape design [46, 23, 21, 44, 22, 12], 3D geometry compression [37, 2, 19], and shape reconstruction from multiple 3D scans [3].

Within the context of interactive shape design, Zorin [46] defines a multi-resolution subdivision structure over an irregular mesh, using the signal processing smoothing algorithms as the basis of his analysis process.

Guskov [12] follows a different signal processing approach over the Progressive Meshes [16] structure, where *frequency* has a completely different meaning. He is able to perform similar filtering operations, as with the methods described in this paper.

† On sabbatical from 08/01/2000 to 07/31/2001, Dept. of Electrical Engineering, California Institute of Technology Mail Code 136-93, Pasadena, CA 91125

In 3D geometry compression[38, 39], Taubin et.al. [37] use these signal processing smoothing algorithms to predict the position of high resolution vertices from their low resolution counterparts in their progressive transmission scheme. Balan and Taubin [2], study the problem of constructing optimal filters in this context. Karni and Gotsman [19] use the partial Fourier expansion applied to the vertices of a mesh partition to define a JPEG-like compression scheme for meshes.

In the area of shpe reconstruction from multiple 3D scans, Bernardini et.al. [3] define a *conforming* process to estimate the average shape of several overlaping meshes by allowing them to deform at very low frequency, while preserving the details. This process is based on applying a very aggressive smoothing filter to the deformation field that would make each vertex of each overlapping mesh move to the average position of vertices of other meshes in a neighborhood.

The paper is organized as follows. In section 2 we introduce Laplacian smoothing withing the context of meshes. In section 3 we show how Fourier Analysis can be performed on signals defined on meshes and graphs. In section 4 we discuss methods to smooth or denoise signals defined on meshes and graphs as low-pass filtering. In section 5 we describe Taubin's $\lambda|\mu$ algorithm. In section 6 we discuss how edge weights can be manipulated to compensate for irregular edge lengths and face angles. In section 7 we show that classic filter design methods can be used to construct faster smoothing algorithms, and other feature enhancing filters. In section 8 we discuss how different constraints can be imposed to the smoothing algorithms and their relation to interactive shape design. Finally, in section 9 we present our conclusions.

2. Laplacian Smoothing

Laplacian smoothing is a well established technique to improve the geometric irregularity of a 2D mesh in the field of finite-elements meshing [15]. In this context, boundary vertices of the mesh are constrained not to move, but internal vertices are simultaneously moved to the barycenter of its neighboring vertices. And then the process is iterated a number of times.

When Laplacian smoothing is applied to a noisy 3D polygonal mesh without constraints, noise is removed, but significant shape distortion may be introduced. The main problem is that Laplacian smoothing produces *shrinkage*, because in the limit, all the vertices of the mesh converge to their barycenter.

To understand why the Laplacian smoothing algorithm removes high frequency noise, why it produces shrinkage, and how to solve the shrinkage problem, we need to develop the basic concepts of signal processing on meshes, or more generally, on graphs.

3. Fourier Analysis on Meshes and Graphs

A graph $G = (V, E)$, composed of a set of n vertices V, and a set of edges E can be directed or undirected. 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 look at the vertices of M as a three-dimensional *graph signal* $v = (v_1, \ldots, v_n)^t$ defined on G. In general, a d-dimensional graph signal on a graph G is a $d \times n$ matrix $x = (x_1, \ldots, x_n)^t$, where each row of x is regarded as the signal value at the i-th. vertex of the graph.

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, every time that a vertex j is a neighbor of vertex i, also i is a neighbor of j. With non-symmetric neighborhoods, which are associated with directed graphs, certain constraints can be imposed. We discuss this issue in detail in section 8.

The set of displacements Δv_i produced by the Laplacian smoothing step that moves each vertex to the barycenter of its neighbors can be described as the result of applying the Laplacian operator to the vertices of the mesh.

The Laplacian operator is defined on a graph signal x by weighted averages over the neighborhoods

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

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

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

Since the Laplacian operator $x \to \Delta x$ is linear on the space of graph signals defined on G, and operates on the coordinates of x independently, it is sufficient to consider the case of one-dimensional graph signals.

In section 6 we discuss in detail different ways of choosing weights. For the time being, lets assume that the edge weights are determined by first choosing an edge cost $c_{ij} = c_{ji} \geq 0$ for each graph edge, and then setting $w_{ij} = c_{ij}/c_i$, where c_i is the average cost of edges incident to i

$$c_i = \sum_{j \in i^\star} c_{ij} > 0 .$$

For example, if all the edges have unit cost $c_{ij} = 1$, then for each neighbor j of i, the weight w_{ij} is equal to the inverse of the number of neighbors $1/|i^\star|$ of v. We organize the edge

costs and weights as matrices $C = (c_{ij})$, $W = (w_{ij})$, with elements equal to zero if j is not a neighbor of i. We also assume that once set, the weights are kept constant during the iterative smoothing process. We will relax this asumption in section 6.

This choice of weights is independent of the vertex positions, or *geometry*, of the mesh, and only function of the structure of the graph G, i.e. the *connectivity* of the mesh. Note that as a result of the neighborhood normalization constraint of equation 2, although the $n \times n$ matrix of edge costs C is symmetric, in general the matrix of edge weigths W is not. We consider edge weights that are function of the geometry in section 6.

If we define the matrix $K = I - W$, with I the identity matrix, the Laplacian operator applied to a graph signal x can be written in matrix form as follows

$$\Delta x = -Kx\,. \tag{3}$$

For undirected graphs and the choice of weights described above, the matrix K has real eigenvalues $0 \le k_1 \le k_2 \le \cdots \le k_n \le 2$ with corresponding linearly independent real unit length right eigenvectors $e^1, \ldots, e^n$ [36]. In matrix form

$$K\mathbf{E} = \mathbf{E}\,\mathrm{diag}(k)\,, \tag{4}$$

with $\mathbf{E} = (e^1, \ldots, e^n)$, $k = (k_1, \ldots, k_n)^t$, and $\mathrm{diag}(k)$ 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 $e^1, \ldots, e^n$ form a basis of n-dimensional space, every graph signal x can be written as a linear combination

$$x = \sum_{j=1}^{n} \hat{x}_j\, e^j = \mathbf{E}\hat{x}\,. \tag{5}$$

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

If instead of being derived from the vertices and edges of a mesh, the graph G is a closed polygonal curve with n vertices and edges, i.e., a cycle, we are in the classical case of discrete-time n-periodic signals.

Fourier analysis is a natural tool to solve the problem of signal smoothing. The space of signals is decomposed into orthogonal subspaces associated with different frequencies, with the low frequency content of a signal regarded as subjacent data, and the high frequency content as noise. To denoise a signal it is sufficient to compute its DFT, discard its high frequency coefficients, and compute the linear combination of remaning terms as the result. This is exatly what the method of *Fourier descriptors* [45] does to smooth a closed curve.

In the case of closed polygonal curves the DFT of a signal x can be computed very efficiently using the Fast Fourier Transform (FFT) algorithm [32], and the eigenvalues and eigenvectors of K can be computed analytically. In general, the matrix K is large, and although sparse, it is almost impossible to reliably compute its eigenvalues and eigenvectors. This makes it impractical to smooth vertex positions of large meshes with the Fourier descriptors method.

Note that even using the FFT algorithm in the closed polygonal curve case, the computational complexity is $O(n\log(n))$, i.e., not linear.

```
Laplacian(G,W,x)
  new Δx = 0;
  for(e = (i, j) ∈ E)
    Δx_i = x_i + w_ij(x_i − x_j);
  end;
return Δx;
```

Figure 1: *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.*

```
LaplacianSmoothing(G,W,N,λ,x)
  new Δx
  for(i = 0; i < N; i = i + 1)
    Δx = Laplacian(G,W,x);
    x = x + λΔx;
  end;
return;
```

Figure 2: *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 smoothd.*

4. Smoothing as Low Pass Filtering

Figure 4 describes the algorithm to evaluate the Laplacian operator on a signal x defined on a directed graph G, with given weight matrix W. And 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 Laplacian smoothing algorithm can be described in matrix form as follows

$$x^1 = x + \lambda\Delta x = (I - \lambda K)\,x = f(K)\,x\,, \tag{6}$$

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 $f(k) = (1 - \lambda k)^N$.

A *Linear Filter* is defined by a univariate function $f(k)$ that can be evaluated on the square matrix K to produce another matrix of the same size. Although many functions of one variable can be evaluated in matrices [10], in this section

```
TaubinSmoothing(G, W, N, λ, μ, x)
  new Δx
  for(i = 0 ; i < N ; i = i + 1)
    Δx = Laplacian(G, W, x);
    if i is even
      x = x + λΔx;
    else
      x = x + μΔx;
  end;
return;
```

Figure 3: *The Taubin Smoothing Algorithm. G graph, W matrix of weights defined on the edges of G, N number of iterations, λ and μ scaling factors, x signal on G to be smoothd.*

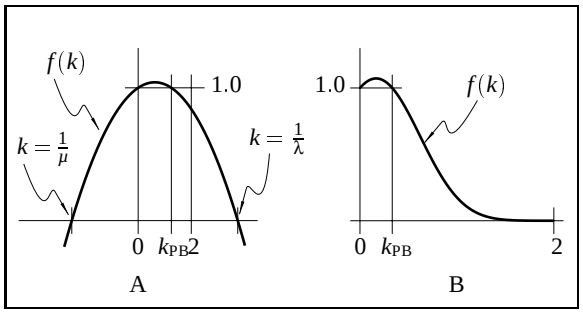

Figure 4: *Graph of transfer functions for the λ|μ algorithm. (A) $f(k) = (1-\mu k)(1-\lambda k)$. (B) $f(k) = ((1-\mu k)(1-\lambda k))^{N/2}$ with $N > 1$.*

we only consider polynomials. In section 7 we also consider rational functions. The function $f(k)$ is the *transfer function* of the filter. It is well known that for any of these functions, the matrix $f(K)$ has as eigenvectors the eigenvectors $e^1, \ldots, e^n$ of the matrix K, and as eigenvalues the result $f(k_1), \ldots, f(k_n)$ of evaluating the function on the eigenvalues of K. Since for any polynomial transfer function

$$x' = f(K)\,x = \sum_{i=1}^{n} f(k_i)\hat{x}_i e^i \,,$$

because $Ke^i = k_i e^i$, 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 $k \in [0,2]$.

In the case of Laplacian smoothing, where the transfer function is $f(k) = (1-\lambda k)^N$, with $0 < \lambda < 1$, 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 atenuated 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. In conclusion Laplacian smoothing filters out too many frequencies.

5. The λ|μ Algorithm

Taubin [36] proposed the following second degree transfer function to solve the problem of shrinkage

$$f(k) = (1-\lambda k)(1-\mu k) \,, \tag{7}$$

which can be implemented as two consecutive steps of Laplacian smoothing with different scaling factors; the first one with $\lambda > 0$, and the second one with $\mu < -\lambda < 0$. That is, after the Laplacian smoothing step with positive scale factor λ is performed (shrinking step), a second Laplacian smoothing step with negative scale factor μ is performed (unshrinking step). Figure 5 describes the algorithm.

The graph of the transfer function of equation (7) is illustrated in figure 4-A. Figure 4-B shows the resulting transfer function after N iterations of the algorithm. Since $f(0) = 1$ and $\mu + \lambda < 0$, there is a positive value of k, let us denote it k_{PB} (the *pass-band frequency*), such that $f(k_{PB}) = 1$. The value of k_{PB} is

$$k_{PB} = \frac{1}{\lambda} + \frac{1}{\mu} > 0 \,. \tag{8}$$

The graph of the transfer function $f(k)$ shown in Figure 4-B displays a typical *low-pass filter* shape in the region of interest $k \in [0,2]$. The *pass-band region* extends from $k = 0$ to $k = k_{PB}$, where $f(k) \approx 1$. As k increases from $k = k_{PB}$ to $k = 2$, the transfer function decreases to zero. The faster the transfer function decreases in this region, the better. The rate of decrease is controlled by the number of iterations N.

For example, choosing λ so that $f(1) = -f(2)$, i.e.,

$$0 = f(1) + f(2) = 1 - 3(\lambda + \mu) + 5\lambda\mu \,, \tag{9}$$

ensures a stable and fast filter [40]. A typical value for k_{PB} is 0.1. The corresponding typical scaling factor values are then computed from equations 8 and 9.

Figures 5 and 6 show examples of large surfaces smoothed with this algorithm. Figures 5 is a synthetic example, where noise has been added to one half of a polyhedral approximation of a sphere. Note that while the algorithm progresses the half without noise does not change. Figure 6 was constructed from a CT scan of a spine. The boundary surface of the set of voxels with intensity value above a certain threshold is used as the input signal. Note that there is not much difference between the results after 50 and 100 iterations.

6. Weights

With *Equal weights*, determined by unit edge costs, very satisfactory results are obtained on meshes which display very

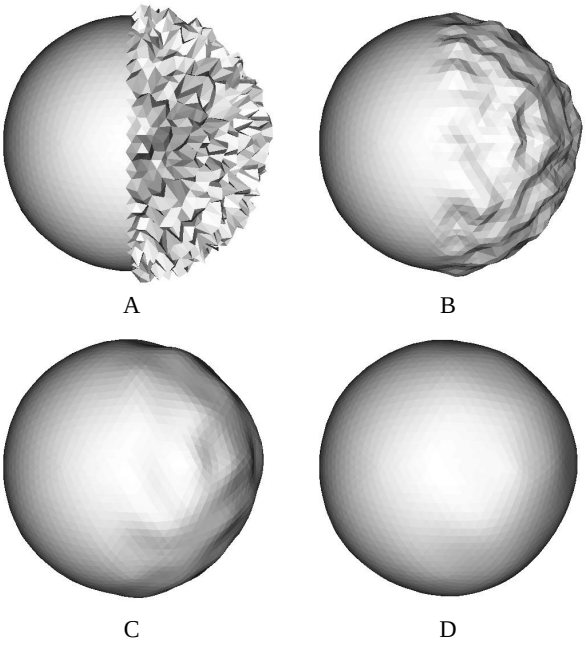

Figure 5: *(A) Sphere partially corrupted by normal noise. (B) Sphere (A) after 10 non-shrinking smoothing steps. (C) Sphere (A) after 50 non-shrinking smoothing steps. (D) Sphere (A) after 200 non-shrinking smoothing steps. Surfaces are flat-shaded to enhance the faceting effect.*

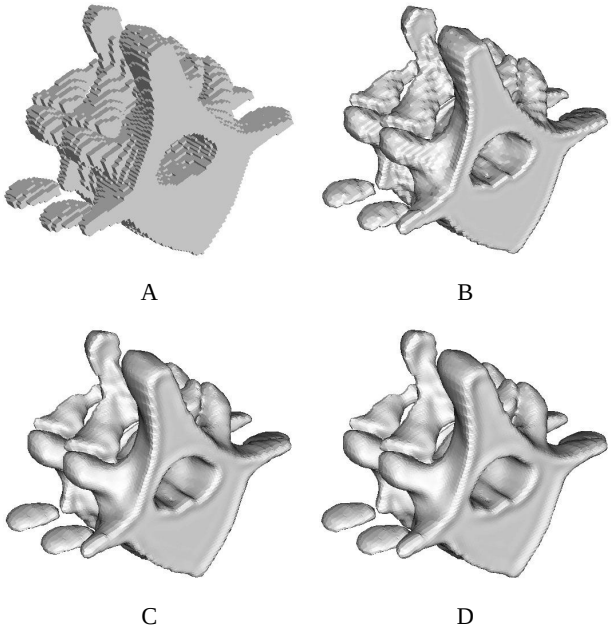

Figure 6: *(A) Boundary surface of voxels from a CT scan. (B) Surface (A) after 10 non-shrinking smoothing steps. (C) Surface (A) after 50 non-shrinking smoothing steps. (D) Surface (A) after 100 non-shrinking smoothing steps.* $k_{PB} = 0.1$ *and* $\lambda = 0.6307$ *in (B), (C), and (D). Surfaces are flat-shaded to enhance the faceting effect.*

small variation in edge length and face angles across the whole mesh, such as those shown in figures 5 and 6. When these assumptions are not met, local distortions are introduced. The edge weights can be used to compensate for the irregularities of the teselation, and produce results which are function of the local geometry of the signal, rather than the local parameterization.

Fujiwara weights 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, both Taubin [36] and Fujiwara [9] propose choosing 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 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 tangencial component. In some cases the edge equalization may be the desired effect. This is the case 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 tangencial component is undesirable. Based on a better approximation to the curvature normal, Desbrun[7] proposes the following choice of edge costs

$$c_{ij} = \cot\alpha_i j + \cot\beta_i j \,, \tag{10}$$

where $\alpha_i j$ and $\beta_i j$ 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 tangencial drift when all the faces incident to the vertex are coplanar.

The three weighting schemes described in this section can be applied to both Laplacian smoothing and Taubin smoothing, but bot Fujiwara weights and Desbrun weights must be recomputed after each iteration, or after a small number of iterations. This makes the whole smoothing process a non-linear operation, and computationally more expensive.

An interactive implementation of these techniques is available as a Java applet[34]. Figure 7 shows a screen shot of this applet.

Guskov [12] proposed another weighting scheme based on divided differences, but applies to a smoothing process based on a second order neighborhood.

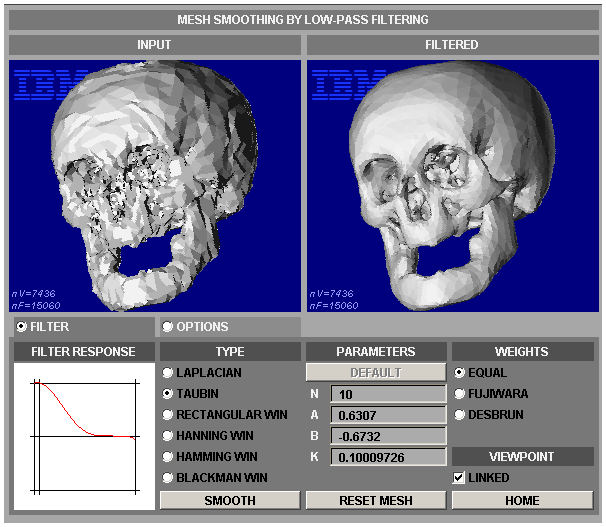

Figure 7: *Implementation of some of the techniques described in this paper as a Java applet* [34].

```
FirFilter(G,W,N, f,x)
  new x^0 = x
  new x^1 = Laplacian(G,W,x^0);
  new x^2 = x^0 − 0.5x^1
  new x = f_0 x^0 + f_1 x^1
  for(i = 2; i < N; i = i + 1)
    x^2 =Laplacian(G,W,x^1);
    x = x + f_i x^2;
    x^0 = x^1;
    x^1 = x^2;
  end;
return;
```

Figure 8: *The FIR Filter Algorithm of Taubin et.al.* [40]. *G graph, W matrix of weights defined on the edges of G, N number of iterations, $f = (f_0, \ldots, f_{N-1})$ polynomial coefficients in Chebyshev basis, x signal on G to be filtered.*

7. Fast Smoothing as Filter Design

In the $\lambda|\mu$ algorithm different combinations of the parameters λ, μ, and N produce almost identical transfer functions $f(k)$. For example if the scaling factors λ is reduced in magnitude, and then μ is recomputed to keep the pass-band frequency unchanged using equation 8, an equivalent result can be achieved with more iterations [40].

Taubin et.al. [40] showed how to efficiently implement any polynomial transfer function expressed as a linear combination of Chebyshev polynomials[6]. Figure 7 describes the algorithm. Chebyshev polynomials are numerically more stable than the power basis, and are defined by a three term recursion that results in an algorithm with low storage use

```
IirFilter(G,W,N_g,g,N_h,h,x)
  FirFilter(G,W,N_g,g,x)
  new x^1 = x;
  new H = h(K);
  solve Hx = x^1;
return;
```

Figure 9: *The IIR Filter Algorithm Taubin et.al.* [40]. *G graph, W matrix of weights defined on the edges of G, N number of iterations, $g = (g_0, \ldots, g_{N_g-1})$ and $h = (h_0, \ldots, h_{N_h-1})$ polynomial coefficients in Chebyshev basis, x signal on G to be filtered.*

and linear complexity

$$\begin{cases} T_0(w) &= 1 \\ T_1(w) &= w \\ T_j(w) &= 2wT_{j-1}(w) - T_{j-2}(w) \end{cases} \qquad (11)$$

Since the domain of Chebyshev polynomials is $w \in [0,1]$, the following change of variable is necessary $w = 1 - k/2$.

The ability to efficiently implement any polynomial transfer function, reduces the problem of minimizing the number of iterations to a univariate polynomial approximation problem, i.e., to the classical problem of Finite Impulse Response (FIR) filter desgin in signal processing [29]. As an example, Taubin et.al. [40] showed how to design filters based on the classical Window-based method [14], but other polynomial approximation technique can be used to design stable FIR filters. For example, The Parks-McClellan algorithm [18] uses the Remez exchange algorithm and Chebyshev approximation theory to design filters with an optimal fit between the desired and actual frequency responses. The filters are optimal in the sense that the maximum error between the desired frequency response and the actual frequency response is minimized. Filters designed this way exhibit an equiripple behavior in their frequency responses and are sometimes called equiripple filters.

The only problem with FIR filters is that high degrees are usually needed to obtain a good approximations of ideal frequency responses with sharp transitions, such as low-pass filters with a narrow pass-band. Infinite Impulse Response filters (IIR), with rational transfer functions with polynomials of low degree, solve this problem. In our case, if the transfer function is a ratio of two polynomials $f(k) = g(k)/h(k)$, with $h(k) \neq 0$ for $k \in [0,2]$, filtering a signal x corresponds to solving the following system of equations

$$h(K)\,x' = g(K)\,x\,. \qquad (12)$$

Evaluation of this filter can be performed in three steps. First, if $g(k)$ is not constant, the right hand side of this equation is evaluated with the FIR algorithm of Taubin et.al. $x^1 = g(K)x$. Then the the matrix $H = h(K)$ has to be constructed, and finally the linear system of equations $Hx = x^1$ is solved. Figure 7 describes this algorithm. In this context, IIR filters only

makes sense if the polynomial $h(k)$ is of very low degree, i.e., if the matrix H is sparse. Some sparse linear solvers only need the to evaluate the product of the matrix H by a vector. In that case the matrix H does not need to be constructed explicitly, and the FIR algorithm of Taubin et.al. can be used again to evaluate this filter as many times as necessary by the linear solver.

The Implicit Fairing method of Desbrun et.al. [7] is a particular case this type of filter. It corresponds to the classical Butterworth filter with transfer function

$$f(k) = \frac{1}{1+(k/k_{\mathrm{PB}})^N} \,. \tag{13}$$

Desbrun et.al. development is based on a PDE formulation. They show that the Laplacian smoothing algorithm corresponds the solution of the diffusion process

$$\frac{\partial x}{\partial t} = \lambda dt \Delta x \,,$$

using the *forward Euler method*

$$x' = x + \lambda dt \Delta x = (I + \lambda dt \Delta) x \,,$$

with unit time step $dt = 1$. They use the *backward Euler method* instead, which requires the solution of the linear system

$$(I - \lambda dt \Delta) x' = x \,,$$

but is stable for arbitrary large time steps, as opposd to the explicit scheme which is stable only for $|\lambda dt| < 1$. Although having to solve a sparse linear system per step, as apposed to multiplying by a sparse matrix, seems to slow down the computation, they report computational time similar or better than the explicit method.

8. Constraints

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.

All the methods described so far allows the signals to freely evolve without imposing any constraint. For example, although shrinkage prevention minimizes the problem in the $\lambda|\mu$ algorithm, all the smooth signal values are different from the original ones.

Taubin [36] 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.

Kobbelt [21, 22] formulates the problem as an energy minimization problem, and solves it efficiently with a multiresolution approach on levels of detail hierachies generated by decimation.

Kuriyama [23] and Yamada [44] impose hard constraints on vertex positions, but modify the displacement produced by the Laplacian operator to impose soft normal constraints.

We will only discuss here some of these methods.

8.1. Interpolatory Constraints

A simple way to introduce interpolatory constraints in the smoothing algorithm is by using non-symmetric neighborhood structures. If no other vertex is a neighbor of a certain vertex v_1, i.e., if the neighborhood of v_1 is empty, then the value x_1 of any signal x does not change during the smoothing process, because the Laplacian operator Δx_1 is equal to zero by definition of empty sum. Other vertices are allowed to have v_1 as a neighbor, though.

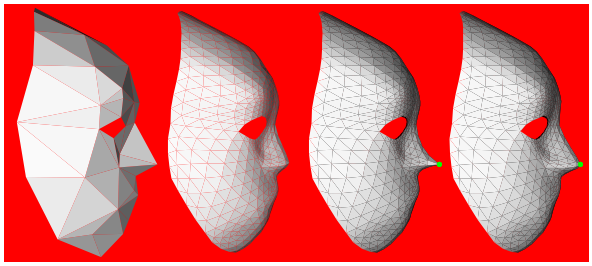

Figure 10: *Example of surfaces designed using subdivision and smoothing steps with one interpolatory constraint. (A) Skeleton. (B) Surface (A) after two levels of subdivision and smoothing without constraints. (C) Same as (B) but with non-smooth interpolatory constraint. (D) Same as (B) but with smooth interpolatory constraint. Surfaces are flat-shaded to enhance the faceting effect.*

Figure 10-A shows a skeleton surface. Figure 10-B shows the surface generated after two levels of refinement and smoothing using our smoothing algorithm without constraints, i.e., with symmetric first-order neighborhoods. Although the surface has not shrunk overall, the nose has been flattened quite significantly. This is so because the nose is made of very few faces in the skeleton, and these faces meet at very sharp angles. Figure 10-C shows the result of applying the same steps, but defining the neighborhood of the vertex at the tip of the nose to be empty. The other neighborhoods are not modified. Now the vertex satisfies the constraint – it has not moved at all during the smoothing process –, but the surface has lost its smoothness at the vertex. This might be the desired effect, but if it is not, instead of the neighborhoods, we have to modify the algorithm.

8.2. Smooth Interpolation

We look at the desired constrained smooth signal x_C^N as a sum of the corresponding unconstrained smooth signal $x^N = Fx$

after N steps of our smoothing algorithm (i.e. $F = f(K)^N$), plus a smooth deformation d_1

$$x_C^N = x^N + (x_1 - x_1^N)\, d_1 \ .$$

The deformation d_1 is itself another discrete surface signal, and the constraint $(x_C^N)_1 = x_1$ is satisfied if $(d_1)_1 = 1$. To construct such a smooth deformation we consider the signal δ_1, where

$$(\delta_i)_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} \ .$$

This is not a smooth signal, but we can apply the smoothing algorithm to it. The result, let us denote it F_{n1}, the first column of the matrix F, is a smooth signal, but its value at the vertex v_1 is not equal to one. However, since the matrix F is diagonally dominated, F_{11}, the first element of its first column, must be non-zero. Therefore, we can scale the signal F_{n1} to make it satisfy the constraint, obtaining the desired smooth deformation

$$d_1 = F_{n1} F_{11}^{-1} \ .$$

Figure 10-D shows the result of applying this process.

When more than one interpolatory constraint must be imposed, the problem is slightly more complicated. For simplicity, we will assume that the vertices have been reordered so that the interpolatory constraints are imposed on the first m vertices, i.e., $(x_C^N)_1 = x_1, \ldots, (x_C^N)_m = x_m$. We now look at the non-smooth signals $\delta_1, \ldots, \delta_m$, and at the corresponding faired signals, the first m columns of the matrix $F = f(K)^N$. These signals are smooth, and so, any linear combination of them is also a smooth signal. Furthermore, since F is non-singular and diagonally dominated, these signals are linearly independent, and there exists a linear combination of them that satisfies the m desired constraints. Explicitly, the constrained smooth signal can be computed as follows

$$x_C^N = x^N + F_{nm} F_{mm}^{-1} \begin{pmatrix} x_1 - x_1^N \\ \vdots \\ x_m - x_m^N \end{pmatrix} \ , \qquad (14)$$

where F_{rs} denotes the sub-matrix of F determined by the first r rows and the first s columns.

To minimize storage requirements, particularly when n is large, and assuming that m is much smaller than n, the computation can be structured as follows. The smoothing algorithm is applied to δ_1 obtaining the first column $F\delta_1$ of the matrix F. The first m elements of this vector are stored as the first column of the matrix F_{mm}. The remaining $m - n$ elements of $F\delta_1$ are discarded. The same process is repeated for $\delta_2, \ldots, \delta_m$, obtaining the remaining columns of F_{mm}. Then the following linear system

$$F_{mm} \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} x_1 - x_1^N \\ \vdots \\ x_m - x_m^N \end{pmatrix}$$

is solved. The matrix F_{mm} is no longer needed. Then the remaining components of the signal y are set to zero $y_{m+1} = \cdots = y_n = 0$. Now the smoothing algorithm is applied to the signal y. The result is the smooth deformation that makes the unconstrained smooth signal x^N satisfy the constraints

$$x_C^N = x^N + F\, y \ .$$

8.3. Smooth Deformations

Note that in the constrained smoothing algorithm described above the fact that the values of the signal at the vertices of interest is constraint to remain constant can be trivially generalized to allow for arbitrary smooth deformations of a surface. To do so, if in equation (14), the values $x_1, \ldots, x_m$ must be replaced by the desired final values of the faired signal at the corresponding vertices. As in in the Free-form deformation approaches of Hsu, Hughes, and Kaufman [17] and Borrel [4], instead of moving control points outside the surface, surfaces can be deformed here by pulling one or more vertices.

Also note that the scope of the deformation can be controlled by changing the number of smoothing steps applied while smoothing the signals $\delta_1, \ldots, \delta_n$. To make the resulting signal satisfy the constraint, the value of N in the definition of the matrix F must be the one used to smooth the deformations. We have observed that good results are obtained when the number of iterations used to smooth the deformations is about five times the number used to fair the original shape.

8.4. Hierarchical Constraints

This is another application of non-symmetric neighborhoods. We start by assigning a numeric label l_i to each vertex of the surface. Then we define the neighborhood structure as follows. We make vertex v_j a neighbor of vertex v_i if v_i and v_j share an edge (or face), *and* if $l_i \leq l_j$. Note that if v_j is a neighbor of v_i and $l_i < l_j$, then v_i is not a neighbor of v_j. The symmetry applies only to vertices with the same label. For example, if we assign label $l_i = 1$ to all the boundary vertices of a surface with boundary, and label $l_i = 0$ to all the internal vertices, then the boundary is faired as a curve, independently of the interior vertices, but the interior vertices follow the boundary vertices. If we also assign label $l_i = 1$ to a closed curve composed of internal edges of the surface, then the resulting surface will be smooth *along*, and on both sides of the curve, but not necessarily *across* the curve. Figure 11-D shows examples of subdivision surface designed using this procedure. If we also assign label $l_i = 2$ to some isolated points along the curves, then those vertices will in fact not move, because they will have empty neighborhoods.

8.5. Tangent Plane Constraints

Although the normal vector to a polyhedral surface is not defined at a vertex, it is customary to define it by averaging some local information, say for shading purposes. When the

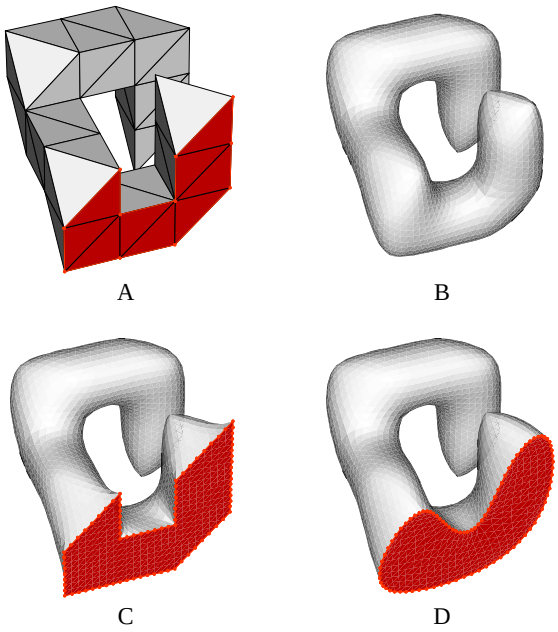

Figure 11: *(A) Skeleton with marked vertices. (B) Surface (A) after three levels of subdivision and smoothing without constraints. (C) Same as (B) but with empty neighborhoods of marked vertices. (D) Same as (B) but with hierarchical neighborhoods, where marked vertices have label 1 and unmarked vertices have label 0. Surfaces are flat-shaded to enhance the faceting effect.*

signal x in equation (1) is replaced by the coordinates of the vertices, the Laplacian becomes a vector

$$\Delta v_i = \sum_{j \in i^*} w_{ij} \, (v_j - v_i) \; .$$

This vector average can be seen as a discrete approximation of the following curvilinear integral

$$\frac{1}{|\gamma|} \int_{v \in \gamma} (v - v_i) \, dl(v) \; ,$$

where γ is a closed curve embedded in the surface which encircles the vertex v_i, and $|\gamma|$ is the length of the curve. It is known that, for a curvature continuous surface, if the curve γ is let to shrink to to the point v_i, the integral converges to the mean curvature $\bar{\kappa}(v_i)$ of the surface at the point v_i times the normal vector N_i at the same point [8]

$$\lim_{\varepsilon \to 0} \frac{1}{|\gamma_\varepsilon|} \int_{v \in \gamma_\varepsilon} (v - v_i) \, dl(v) = \bar{\kappa}(v_i) N_i \; .$$

The expression on the right hand side is the *curvature normal*, where $\bar{\kappa}(v_i)$ is the mean curvature of the surface at v_i and N_i is the surface normal at v_i. It follows that the length of the laplacian vector is equal to the product of the average edge length times the mean curvature

$$\Delta v_i = \left(\sum_{j \in i^*} w_{ij} \, \| (v_j - v_i) \| \right) \bar{\kappa}(v_i) N_i \; ,$$

which can be used as a definition of discrete mean curvature [35].

It follows that imposing normal constraints at v_i is achieved by imposing linear constraints on Δv_i. If N_i is the desired normal direction at vertex v_i after the smoothing process, and S_i and T_i are two linearly independent vectors tangent to N_i, the surface after N iterations of the smoothing algorithm will satisfy the normal desired constraint at the vertex v_i it the following two linear constraints

$$S_i^t \Delta v_i^N = T_i^t \Delta v_i^N = 0$$

are satisfied. This leads us to the problem of smoothing with general linear constraints.

8.6. General Linear Constraints

We consider here the problem of smoothing a discrete surface signal x under general linear constraints $C x_C^N = c$, where C is a $m \times n$ matrix of rank m (m independent constraints), and $c = (c_1, \ldots, c_m)^t$ is a vector. The method described in section 8.1 to impose smooth interpolatory constraints, is a particular case of this problem, where the matrix C is equal the upper m rows of the $m \times m$ identity matrix. Our approach is to reduce the general case to this particular case.

We start by decomposing the matrix C into two blocks. A first $m \times m$ block denoted $C_{(1)}$, composed of m columns of C, and a second block denoted $C_{(2)}$, composed of the remaining columns. The columns that constitute $C_{(1)}$ must be chosen so that $C_{(1)}$ become non-singular, and as well conditioned as possible. In practice this can be done using Gauss elimination with full pivoting [10], but for the sake of simplicity, we will assume here that $C_{(1)}$ is composed of the first m columns of C. We decompose signals in the same way. $x_{(1)}$ denotes here the first m components, and $x_{(2)}$ the last $n - m$ components, of the signal x. We now define a change of basis in the vector space of discrete surface signals as follows

$$\begin{cases} x_{(1)} &= y_{(1)} - C_{(1)}^{-1} C_{(2)} y_{(2)} \\ x_{(2)} &= y_{(2)} \end{cases} \; .$$

If we apply this change of basis to the constraint equation $C_{(1)} x_{(1)} + C_{(2)} x_{(2)} = c$, we obtain $C_{(1)} y_{(1)} = c$, or equivalently

$$y_{(1)} = C_{(1)}^{-1} c \; ,$$

which is the problem solved in section 8.2.

9. Conclusions

In this paper I described the basic elements of the signal processing approach on meshes. It started as a solution to the

shrinkage problem of Laplacian smoothing, and has evolved quite significantly during the last five years, with many important contributions and extensions by many authors, and applications to other areas. In my opinion, the main reason for this interest has been the simplicity of the algorithms and the good qulity of the results produced. I believe that this area will continue evolving in the near future, with theoretical advances, new efficient algorithms, and important applications. Many concepts of classical signal processing may see usefull applications in computer graphics and geometric design, if efficient implementations become available. I look forward to continue contributing to this field myself.

References

1. C.L. Bajaj and Ihm. Smoothing polyhedra using implicit algebraic splines. *Computer Graphics*, pages 79–88, July 1992. (Proceedings SIGGRAPH'92).
2. R. Balan and G. Taubin. 3d mesh geometry filtering algorithms for progressive transmission schemes. *Computer Aided Design*, 2000. Special issue on multiresolution geometric models.
3. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 1999.
4. P. Borrel. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, April 1994.
5. G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics*, pages 257–266, July 1991. (Proceedings SIGGRAPH'91).
6. P.J. Davis. *Interpolation and Approximation*. Dover Publications, Inc., 1975.
7. 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.
8. M. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.
9. K. Fujiwara. Eigenvalues of laplacians on a closed riemannian manifold and its nets. *Proceedings of the AMS*, 123:2585–2594, 1995.
10. G. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, 2nd. edition, 1989.
11. G. Greiner and H.P. Seidel. Modeling with triangular b-splines. *IEEE Computer Graphics and Applications*, 14(2):56–60, March 1994.
12. I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Siggraph'99 Conference Proceedings*, pages 325–334, August 1999.
13. M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surface. *Computer Graphics*, pages 35–44, August 1993. (Proceedings SIGGRAPH'93).
14. R.W. Hamming. *Digital Filters*. Prentice Hall, 1989.
15. K. Ho-Le. Finite element mesh generation methods: A review and classification. *Computer Aided Design*, 20(1):27–38, 1988.
16. H. Hoppe. Progressive meshes. In *Siggraph'96 Conference Proceedings*, pages 99–108, August 1996.
17. W.M. Hsu, J.F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, pages 177–184, July 1992. (Proceedings SIGGRAPH'92).
18. IEEE Press, New York. *Programs for Digital Signal Processing*, 1979. Algorithm 5.1.
19. Z. Karni and C Gotsman. Spectral compression of mesh geometry. In *Siggraph'2000 Conference Proceedings*, 2000.
20. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
21. 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.
22. L. Kobbelt, J. Vorsatz, and H.-P. Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry Theory and Applications*, 1999. special issue on multiresolution modeling and 3D geometry compression.
23. S. Kuriyama and K. Tachibana. Polyhedral surface modeling with a diffusion system. In *Eurographics'97 Conference Proceedings*, pages C39–C46, 1997.
24. C. Loop. A G^1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, 11:303–330, 1994.
25. C. Loop. Smooth spline surfaces over irregular meshes. *Computer Graphics*, pages 303–310, July 1994. (Proceedings SIGGRAPH'94).
26. M. Lounsbery, S. Mann, and T. DeRose. Parametric surface interpolation. *IEEE Computer Graphics and Applications*, 12(5):45–52, September 1992.
27. J. Menon. Constructive shell representations for freeform surfaces and solids. *IEEE Computer Graphics and Applications*, 14(2):24–36, March 1994.
28. H.P. Moreton and C.H. Séquin. Functional optimization for fair surface design. *Computer Graphics*, pages 167–176, July 1992. (Proceedings SIGGRAPH'92).
29. A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing*. Prentice Hall, Englewood Cliffs, NJ, 1975.
30. A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729, July 1991.
31. L.A. Shirman and C.H. Séquin. Local surface interpolation with bezier patches. *Computer Aided Geometric Design*, 4:279–295, 1987.
32. G. Strang. The discrete cosine transform. *SIAM Review*, 41(1):135–147, 1999.

33. R.S. Szeliski, D. Tonnesen, and D. Terzopoulos. Modeling surfaces of arbitrary topology with dynamic particles. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–87, New York, NY, June 15–17 1993.

34. G. Taubin. Mesh smoothing applet. http://www.research.ibm.com/people/t/taubin/smooth.

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

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

37. G. Taubin, A. Guéziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *Siggraph'98 Conference Proceedings*, pages 123–132, July 1998.

38. G. Taubin and J. Rossignac, editors. *3D Geometry Compression*, Siggraph'98 Course Notes 21, July 1998.

39. G. Taubin and J. Rossignac, editors. *3D Geometry Compression*, Siggraph'99 Course Notes 22, August 1999.

40. 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.

41. D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4:306–311, 1988.

42. W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, pages 157–166, July 1992. (Proceedings SIGGRAPH'92).

43. W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. *Computer Graphics*, pages 247–256, July 1994. (Proceedings SIGGRAPH'94).

44. 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.

45. C.T. Zahn and R.Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, 21(3):269–281, March 1972.

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