A Signal Processing Approach To Fair Surface Design

Gabriel Taubin
IBM T.J.Watson Research Center

ABSTRACT

In this paper we describe a new tool for interactive free-form fair surface design. By generalizing classical discrete Fourier analysis to two-dimensional discrete surface signals – functions defined on polyhedral surfaces of arbitrary topology –, we reduce the problem of surface smoothing, or fairing, to low-pass filtering. We describe a very simple surface signal low-pass filter algorithm that applies to surfaces of arbitrary topology. As opposed to other existing optimization-based fairing methods, which are computationally more expensive, this is a linear time and space complexity algorithm. With this algorithm, fairing very large surfaces, such as those obtained from volumetric medical data, becomes affordable. By combining this algorithm with surface subdivision methods we obtain a very effective fair surface design technique. We then extend the analysis, and modify the algorithm accordingly, to accommodate different types of constraints. Some constraints can be imposed without any modification of the algorithm, while others require the solution of a small associated linear system of equations. In particular, vertex location constraints, vertex normal constraints, and surface normal discontinuities across curves embedded in the surface, can be imposed with this technique.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/image generation - display algorithms; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - curve, surface, solid, and object representations; J.6 [Computer Applications]: Computer-Aided Engineering - computer-aided design


1 INTRODUCTION

The signal processing approach described in this paper was originally motivated by the problem of how to fair large polyhedral surfaces of arbitrary topology, such as those extracted from volumetric medical data by iso-surface construction algorithms [21, 2, 11, 15], or constructed by integration of multiple range images [36].

Since most existing algorithms based on fairness norm optimization [37, 24, 12, 38] are prohibitively expensive for very large surfaces – a million vertices is not unusual in medical images –, we decided to look for new algorithms with linear time and space complexity [31]. Unless these large surfaces are first simplified [29, 13, 11], or re-meshed using far fewer faces [35], methods based on patch technology, whether parametric [28, 22, 10, 20, 19] or implicit [1, 23], are not acceptable either. Although curvature continuous, a patch-based surface interpolant is far more complex than the original surface, more expensive to render, and worst of all, does not remove the high curvature variation present in the original mesh.

As in the fairness norm optimization methods and physics-based deformable models [16, 34, 30, 26], our approach is to move the vertices of the polyhedral surface without changing the connectivity of the faces. The faired surface has exactly the same number of vertices and faces as the original one. However, our signal processing formulation results in much less expensive computations. In these variational formulations [5, 24, 38, 12], after finite element discretization, 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 [9], and usually result in quadratic time complexity algorithms. In our case, the problem of surface fairing is reduced to sparse matrix multiplication instead, a linear time complexity operation.

The paper is organized as follows. In section 2 we describe how to extend signal processing to signals defined on polyhedral surfaces of arbitrary topology, reducing the problem of surface smoothing to low-pass filtering, and we describe a particularly simple linear time and space complexity surface signal low-pass filter algorithm. Then we concentrate on the applications of this algorithm to interactive free-form fair surface design. As Welch and Witkin [38], in section 3 we design more detailed fair surfaces by combining our fairing algorithm with subdivision techniques. In section 4 we modify our fairing algorithm to accommodate different kinds of constraints. Finally, in section 5 we present some closing remarks.

2 THE SIGNAL PROCESSING APPROACH

Fourier analysis is a natural tool to solve the problem of signal smoothing. The space of signals – functions defined on certain domain – 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.

2.1 CLOSED CURVE FAIRING

To smooth a closed curve it is sufficient to remove the noise from the coordinate signals, i.e., to project the coordinate signals onto the subspace of low frequencies. This is what the method of Fourier descriptors, which dates back to the early 60’s, does [40]. Our approach to extend Fourier analysis to signals defined on polyhedral surfaces of arbitrary topology is based on the observation that the classical Fourier transform of a signal can be seen as the decomposition of the signal into a linear combination of the eigenvectors of the Laplacian operator. To extend Fourier analysis to surfaces of arbitrary topology we only have to define a new operator that takes the place of the Laplacian.

As a motivation, let us consider the simple case of a discrete time \( \pi \)-periodic signal – a function defined on a regular polygon of \( \pi \) vertices –, which we represent as a column vector \( x = (x_1, \ldots, x_n)^T \). The components of this vector are the values of the signal at the
Explicitly, the real eigenvalues \( u_i \) increases, the corresponding eigenvector.

Note that since the vertices. (B) Then a second step with negative scale factor \( \mu \) is applied to all the vertices.

Since \( x \) is symmetric, it has real eigenvalues and eigenvectors. Explicitly, the real eigenvalues \( k_1, \ldots, k_n \) of \( K \), sorted in non-decreasing order, are

\[
k_i = 1 - \cos \left( 2\pi \left\lfloor j/2 \right\rfloor / n \right),
\]

and the corresponding unit length real eigenvectors, \( u_1, \ldots, u_n \), are

\[
(u_j)_k = \begin{cases} 
\sqrt{1/n} & \text{if } j = 1 \\
\sqrt{2/n} \sin \left( 2\pi h \left\lfloor j/2 \right\rfloor / n \right) & \text{if } j \text{ is even} \\
\sqrt{2/n} \cos \left( 2\pi h \left\lfloor j/2 \right\rfloor / n \right) & \text{if } j \text{ is odd}
\end{cases}
\]

Note that \( 0 \leq k_1 \leq \ldots \leq k_n \leq 2 \), and as the frequency \( k_j \) increases, the corresponding eigenvector \( u_j \), as a \( n \)-periodic signal, changes more rapidly from vertex to vertex.

To decompose the signal \( x \) as a linear combination of the real eigenvectors \( u_1, \ldots, u_n \),

\[
x = \sum_{i=1}^{n} \xi_i u_i,
\]

is computationally equivalent to the evaluation of the Discrete Fourier Transform of \( x \). To smooth the signal \( x \) with the method of Fourier descriptors, this decomposition has to be computed, and then the high frequency terms of the sum must be discarded. But this is computationally expensive. Even using the Fast Fourier Transform algorithm, the computational complexity is in the order of \( n \log(n) \) operations.

An alternative is to do the projection onto the space of low frequencies only approximately. This is what a low-pass filter does. We will only consider here low-pass filters implemented as a convolution. A more detailed analysis of other filter methodologies is beyond the scope of this paper, and will be done elsewhere [33]. Perhaps the most popular convolution-based smoothing method for parameterized curves is the so-called Gaussian filtering method, associated with scale-space theory [39, 17]. In its simplest form, it can be described by the following formula

\[
x_i' = x_i + \lambda \Delta x_i,
\]

where \( 0 < \lambda < 1 \) is a scale factor (for \( \lambda < 0 \) and \( \lambda \geq 1 \) the algorithm enhances high frequencies instead of attenuating them).

This can be written in matrix form as

\[
x' = (I - \lambda K) x.
\]

It is well known though, that Gaussian filtering produces shrinkage, and this is so because the Gaussian kernel is not a low-pass filter kernel [25]. To define a low-pass filter, the matrix \( I - \lambda K \) must be replaced by some other function \( f(K) \) of the matrix \( K \). Our non-shrinking fairing algorithm, described in the next section, is one particularly efficient choice.

We now extend this formulation to functions defined on surfaces of arbitrary topology.

### 2.2 Surface Signal Fairing

At this point we need a few definitions. We represent a polyhedral surface as a pair of lists \( S = \{V, F\} \), a list of \( n \) vertices \( V \), and a list of polygonal faces \( F \). Although in our current implementation, only triangulated surfaces, and surfaces with quadrilateral faces are allowed, the algorithm is defined for any polyhedral surface.

Both for curves and for surfaces, a neighborhood of a vertex \( v_i \) is a set \( i' \) of indices of vertices. If the index \( j \) belongs to the neighborhood \( i' \), we say that \( v_j \) is a neighbor of \( v_i \). The neighborhood structure of a polygonal curve or polyhedral surface is the family of all its neighborhoods \( \{i' : i = 1, 2, \ldots, n\} \). A neighborhood structure is symmetric if every time that a vertex \( v_j \) is a neighbor of vertex \( v_i \), also \( v_i \) is a neighbor of \( v_j \). With non-symmetric neighborhoods certain constraints can be imposed. We discuss this issue in detail in section 4.

A particularly important neighborhood structure is the first order neighborhood structure, where for each pair of vertices \( v_i \) and \( v_j \) that share a face (edge for a curve), we make \( v_j \) a neighbor of \( v_i \), and \( v_i \) a neighbor of \( v_j \). For example, for a polygonal curve represented as a list of consecutive vertices, the first order neighborhood of a vertex \( v_i \) is \( i' = \{i-1, i+1\} \). The first order neighborhood

![Figure 2: (A) Graph of transfer function \( f(k) = (1 - \mu k)(1 - \lambda k) \) of non-shrinking smoothing algorithm.](image-url)
structure is symmetric, and since it is implicitly given by the list of faces of the surface, no extra storage is required to represent it. This is the default neighborhood structure used in our current implementation.

A discrete surface signal is a function \( x = (x_1, \ldots, x_n)^T \) defined on the vertices of a polyhedral surface. We define the discrete Laplacian of a discrete surface signal by weighted averages over the neighborhoods

\[
\Delta x_i = \sum_{j \in \mathcal{N}(i)} w_{ij} (x_j - x_i),
\]

where the weights \( w_{ij} \) are positive numbers that add up to one, \( \sum_{j \in \mathcal{N}(i)} w_{ij} = 1 \), for each \( i \). The weights can be chosen in many different ways taking into consideration the neighborhood structures. One particularly simple choice that produces good results is to set \( w_{ij} \) equal to the inverse of the number of neighbors \( |\mathcal{N}_{ij}| \) of vertex \( v_i \), for each element \( j \) of \( i \). Note that the case of the Laplacian of a periodic signal (1) is a particular case of these definitions. A more general way of choosing weights for a surface with a first order neighborhood structure, is using a positive function \( \phi(v_i, v_j) \) defined on the edges of the surface

\[
w_{ij} = \frac{\phi(v_i, v_j)}{\sum_{k \in \mathcal{E}} \phi(v_i, v_k)}.
\]

For example, the function can be the surface area of the two faces that share the edge, or some power of the length of the edge \( \phi(v_i, v_j) = \|v_i - v_j\|^\alpha \). In our implementation the user can choose any one of these weighting schemes. They produce similar results when the surface has faces of roughly uniform size. When using a power of the length of the edges as weighting function, the exponent \( \alpha = -1 \) produces good results.

If \( W = (w_{ij}) \) is the matrix of weights, with \( w_{ij} = 0 \) when \( j \) is not a neighbor of \( i \), the matrix \( K \) can now be defined as

\[
K = I - W.
\]

In the appendix we show that for a first order neighborhood structure, and for all the choices of weights described above, the matrix \( K \) has real eigenvalues \( 0 \leq k_1 \leq k_2 \leq \cdots \leq k_n \leq 2 \) with corresponding linearly independent real unit length right eigenvectors \( u_1, \ldots, u_n \). Seen as discrete surface signals, these eigenvectors should be considered as the natural vibration modes of the surface, and the corresponding eigenvalues as the associated natural frequencies.

The decomposition of equation (3), of the signal \( x \) into a linear combination of the eigenvectors \( u_1, \ldots, u_n \), is still valid with these definitions, but there is no extension of the Fast Fourier Transform algorithm to compute it. The method of Fourier descriptors – the exact projection onto the subspace of low frequencies – is just not longer feasible, particularly for very large surfaces. On the other hand, low-pass filtering – the approximate projection – can be formulated in exactly the same way as for \( n \)-periodic signals, as the multiplication of a function \( f(K) \) of the matrix \( K \) by the original signal

\[
x' = f(K)x,
\]

and this process can be iterated \( N \) times

\[
x'^N = f(K)^N x.
\]

The function of one variable \( f(k) \) is the transfer function of the filter. Although many functions of one variable can be evaluated in matrices [9], we will only consider polynomials here. For example, in the case of Gaussian smoothing the transfer function is \( f(k) = 1 - \lambda k \). Since for any polynomial transfer function we have

\[
x' = f(K)x = \sum_{i=1}^n \xi_i f(k_i) u_i,
\]

because \( Ku_i = k_i u_i \), to define a low-pass filter we need to find a polynomial such that \( f(k_i)^N \approx 1 \) for low frequencies, and
The polynomial transfer function of equation (7) can be designed by first beyond the scope of this paper, and will be treated elsewhere [33].

Transfer functions, but the analysis of the filter design problem is simple to implement, and produces smoothing without shrinkage. Our choice is

\[ f(k) = (1 - \lambda k)(1 - \mu k) \] (7)

where \( 0 < \lambda \), and \( \mu \) is a new negative scale factor such that \( \mu < -\lambda \). That is, after we perform the Gaussian smoothing step of equation (4) with positive scale factor \( \lambda \) for all the vertices – the shrinking step –, we then perform another similar step

\[ x'_i = x_i + \mu \Delta x_i \] (8)

for all the vertices, but with negative scale factor \( \mu \) instead of \( \lambda \) – the un-shrinking step –. These steps are illustrated in figure 1.

The graph of the transfer function of equation (7) is illustrated in figure 2-A. Figure 2-B shows the resulting transfer function after \( N \) iterations of the algorithm, the graph of the function \( f(k)^N \). Since \( f(0) = 1 \) and \( \mu + \lambda < 0 \), there is a positive value of \( k \), the pass-band frequency \( k_{PB} \), such that \( f(k_{PB}) = 1 \). The value of \( k_{PB} \) is

\[ k_{PB} = \frac{1}{\lambda} + \frac{1}{\mu} > 0 \]. (9)

The graph of the transfer function \( f(k)^N \) 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 \( f(k)^N \approx 1 \). 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 \( N \).

This algorithm is fast (linear both in time and space), extremely simple to implement, and produces smoothing without shrinkage. Faster algorithms can be achieved by choosing other polynomial transfer functions, but the analysis of the filter design problem is beyond the scope of this paper, and will be treated elsewhere [33]. However, as a rule of thumb, the filter based on the second degree polynomial transfer function of equation (7) can be designed by first choosing a value of \( k_{PB} \). Values from 0.01 to 0.1 produce good results, and all the examples shown in the paper where computed with \( k_{PB} \approx 0.1 \). Once \( k_{PB} \) has been chosen, we have to choose \( \lambda \) and \( N \) (\( \mu \) comes out of equation (9) afterwards). Of course we want to minimize \( N \), the number of iterations. To do so, \( \lambda \) must be chosen as large as possible, while keeping \( |f(k)| < 1 \) for \( k_{PB} < k \leq 2 \) (if \( |f(k)| \geq 1 \) in \([k_{PB}, 2] \), the filter will enhance high frequencies instead of attenuating them). In some of the examples, we have chosen \( \lambda \) so that \( f(1) = -f(2) \). For \( k_{PB} < 1 \) this choice of \( \lambda \) ensures a stable and fast filter.

Figures 3 and 4 show examples of large surfaces fairied with this algorithm. Figures 3 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 significantly. Figure 4 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.

### 3 SUBDIVISION

A subdivision surface is a smooth surface defined as the limit of a sequence of polyhedral surfaces, where the next surface in the sequence is constructed from the previous one by a refinement process. In practice, since the number of faces grows very fast, only a few levels of subdivision are computed. Once the faces are smaller than the resolution of the display, it is not necessary to continue. As Welch and Witkin [38], we are not interested in the limit surfaces, but rather in using subdivision and smoothing steps as tools to design fair polyhedral surfaces in an interactive environment. The classical subdivision schemes [8, 4, 12] are rigid, in the sense that they have no free parameters that influence the behavior of the algorithm as it progresses through the subdivision process. By using our fairing algorithm in conjunction with subdivision steps, we achieve more flexibility in the design process. In this way our fairing algorithm can be seen as a complement of the existing subdivision strategies.

In the subdivision surfaces of Catmull and Clark [4, 12] and Loop [18, 6], the subdivision process involves two steps. A refinement step, where a new surface with more vertices and faces is created, and a smoothing step, where the vertices of the new sur-

---

**Figure 5:** Surfaces created alternating subdivision and different smoothing steps. (A) Skeleton surface. (B) One Gaussian smoothing step (\( \lambda = 0.5 \)). Note the hexagonal symmetry because of insufficient smoothing. (C) Five Gaussian smoothing steps (\( \lambda = 0.5 \)). Note the shrinkage effect. (D) Five non-shrinking smoothing steps (\( k_{PB} = 0.1 \) and \( \lambda = 0.6307 \)) of this paper. (B), (C), and (D) are the surfaces obtained after two levels of refinement and smoothing. Surfaces are flat-shaded to enhance the faceting effect.
face are moved. The Catmull and Clark refinement process creates polyhedral surfaces with quadrilateral faces, and Loop refinement process subdivides each triangular face into four similar triangular faces. In both cases the smoothing step can be described by equation (4). The weights are chosen to ensure tangent or curvature continuity of the limit surface.

These subdivision surfaces have the problem of shrinkage, though. The limit surface is significantly smaller overall than the initial skeleton mesh — the first surface of the sequence —. This is so because the smoothing step is essentially Gaussian smoothing, and as we have pointed out, Gaussian smoothing produces shrinkage. Because of the refinement steps, the surfaces do not collapse to the centroid of the initial skeleton, but the shrinkage effect can be quite significant.

The problem of shrinkage can be solved by a global operation. If the amount of shrinkage can be predicted in closed form, the skeleton surface can be expanded before the subdivision process is applied. This is what Halstead, Kass, and DeRose [12] do. They show how to modify the skeleton mesh so that the subdivision surface associated with the modified skeleton interpolates the vertices of the original skeleton.

The subdivision surfaces of Halstead, Kass, and DeRose interpolate the vertices of the original skeleton, and are curvature continuous. However, they show a significant high curvature content, even when the original skeleton mesh does not have such undulations. The shrinkage problem is solved, but a new problem is introduced. Their solution to this second problem is to stop the subdivision process after a certain number of steps, and fair the resulting polyhedral surface based on a variational approach. Their fairness norm minimization procedure reduces to the solution of a large sparse linear system, and they report quadratic running times.

The result of this modified algorithm is no longer a curvature continuous surface that interpolates the vertices of the skeleton, but a more detailed fair polyhedral surface that usually does not interpolate the vertices of the skeleton unless the interpolatory constraints are imposed during the fairing process.

We argue that the source of the unwanted undulations in the Catmull-Clark surface generated from the modified skeleton is the smoothing step of the subdivision process. Only one Gaussian smoothing step does not produce enough smoothing, i.e., it does not produce sufficient attenuation of the high frequency components of the surfaces, and these high frequency components persist during the subdivision process. Figure 5-B shows an example of such a subdivision surface created with the triangular refinement step of Loop, and one Gaussian smoothing step of equation (4). The hexagonal symmetry of the skeleton remains during the subdivision process. Figure 5-C shows the same example, but where five Gaussian smoothing steps are performed after each refinement step. The hexagonal symmetry has been removed at the expense of significant shrinkage effect. Figure 5-D shows the same example where the five non-shrinking fairing steps are performed after each refinement step. Neither hexagonal symmetry nor shrinkage can be observed.

4 CONSTRAINTS

Although surfaces created by a sequence of subdivision and smoothing steps based on our fairing algorithm do not shrink much, they usually do not interpolate the vertices of the original skeleton. In this section we show that by modifying the neighborhood structure certain kind of constraints can be imposed without any modification of the algorithm. Then we study other constraints that require minor modifications.

4.1 INTERPOLATORY CONSTRAINTS

As we mentioned in section 2.2, a simple way to introduce interpolatory constraints in our fairing algorithm is by using non-symmetric neighborhood structures. If no other vertex is a neighbor of a certain vertex vj, i.e., if the neighborhood of vj is empty, then the value xj of any discrete surface signal z does not change during the fairing process, because the discrete Laplacian Δzj is equal to zero by definition of empty sum. Other vertices are allowed to have vj as a neighbor, though. Figure 6-A shows a skeleton surface. Figure 6-B shows the surface generated after two levels of refinement and smoothing using our fairing 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 6-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 fairing 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.

4.2 SMOOTH INTERPOLATION

We look at the desired constrained smooth signal xC as a sum of the corresponding unconstrained smooth signal x̃C and a first order deformation d1.

\[ x_C = x̃_C + (x_1 - x_1) d_1 \]

The deformation d1 is itself another discrete surface signal, and the constraint (x̃C)1 = x1 is satisfied if (d1)1 = 1. To construct such a smooth deformation we consider the signal 1, where

\[ (1)_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} \]

This is not a smooth signal, but we can apply the fairing algorithm to it. The result, let us denote it F1, the first column of the matrix F, is a smooth signal, but its value at the vertex v1 is not equal to one. However, since the matrix F is diagonally dominated, F1, the first element of its first column, must be non-zero. Therefore, we can scale the signal F1 to make it satisfy the constraint, obtaining the desired smooth deformation

\[ d_1 = F1 \cdot F1^{-1} \]
Figure 6-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., \((x^N_i): = x_1, \ldots, (x^N_m) = x_m\). 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 \( F \). 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

\[
x^N_C = x^N + F_{mm} F_{mm}^{-1} \begin{pmatrix} x_1 - x^N_1 \\ \vdots \\ x_m - x^N_m \end{pmatrix}, \quad (10)
\]

where \( F_{mm} \) denotes the sub-matrix of \( F \) determined by the first \( m \) rows and the first \( m \) columns. Figure 7 shows examples of surfaces constructed using subdivision and smoothing steps and interpolating some vertices of the skeleton using this method. The parameter of the fairing algorithm has been modified to achieve different effects, including shrinkage.

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 fairing algorithm is applied to \( \delta_1 \) 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

\[
F_{mm} \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} x_1 - x^N_1 \\ \vdots \\ x_m - x^N_m \end{pmatrix}
\]

is solved. The matrix \( F_{mm} \) is no longer needed. Then the remaining components of the signal \( y \) are set to zero. Now the fairing algorithm is applied to the signal \( y \). The result is the smooth deformation that makes the unconstrained smooth signal \( x^N \) satisfy the constraints

\[
x^N_C = x^N + F y.
\]

### 4.3 SMOOTH DEFORMATIONS

Note that in the constrained fairing algorithm described above the fact that the values of the signal at the vertices of interest are constrained to remain constant can be trivially generalized to allow for arbitrary smooth deformations of a surface. To do so, the values \( x_1, \ldots, x_m \) in equation (10) 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 [14] and Borrel [3], 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 \( \delta_1, \ldots, \delta_m \). 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. The examples in figure 7 have been generated in this way.

### 4.4 HIERARCHICAL CONSTRAINTS

This is another application of non-symmetric neighborhoods. We start by assigning a numeric label \( l \) to each vertex of the surface. Then we define the neighborhood structure as follows. We make vertex \( v_1 \) a neighbor of vertex \( v_2 \) if \( v_1 \) and \( v_2 \) share an edge (or face), and if \( l_1 \leq l_2 \). Note that if \( v_1 \) is a neighbor of \( v_2 \) and \( l_1 < l_2 \), then \( v_1 \) is not a neighbor of \( v_2 \). The symmetry applies only to vertices with the same label. For example, if we assign label \( l_1 = 1 \) to all the boundary vertices of a surface with boundary, and label \( l_2 = 0 \) 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 \( l_3 = 2 \) to some isolated points along the curves, then those vertices will in fact not move, because they will have empty neighborhoods.

### 4.5 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 signal \( x \) in equation (6) is replaced by the coordinates of the vertices, the Laplacian becomes a vector

\[
\Delta v_i = \sum_{j \in \delta_i} w_{ij}(v_j - v_i).
\]
4.6 GENERAL LINEAR CONSTRAINTS

We consider here the problem of fairing a discrete surface signal \( \pi \) under general linear constraints \( C \pi = c \), where \( C \) is a \( m \times m \) matrix of rank \( m \) (\( m \) independent constraints), and \( c = (c_1, \ldots, c_m)^T \) is a vector. The method described in section 4.1 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 conditionned as possible. In practice this can be done using Gauss elimination with full pivoting [9], 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. \( \pi(1) \) denotes here the first \( m \) components, and \( \pi(2) \) the last \( n - m \) components, of the signal \( \pi \). We now define a change of basis in the vector space of discrete surface signals as follows:

\[
\begin{align*}
\pi(1) &= y(1) - C^{-1}(1) C(2) y(2) \\
\pi(2) &= y(2)
\end{align*}
\]

If we apply this change of basis to the constraint equation \( C(1) \pi(1) + C(2) \pi(2) = c \), we obtain \( C(1) y(1) = c \), or equivalently

\[
y(1) = C(1)^{-1} c,
\]

which is the problem solved in section 4.2.

5 CONCLUSIONS

We have presented a new approach to polyhedral surface fairing based on signal processing ideas, we have demonstrated how to use it as an interactive surface design tool. In our view, this new approach represents a significant improvement over the existing fairness-norm optimization approaches, because of the linear time and space complexity of the resulting fairing algorithm.

Our current implementation of these ideas is a surface modeler that runs at interactive speeds on an IBM RS/6000 class workstation under X-Windows. In this surface modeler we have integrated all the techniques described in this paper and many other popular polyhedral surface manipulation techniques. Among other things, the user can interactively define neighborhood structures, select vertices or edges to impose constraints, subdivide the surfaces, and apply the fairing algorithm with different parameter values. All the illustrations of this paper where generated with this software.

In terms of future work, we plan to investigate how this approach can be extended to provide alternatives solutions for other important graphics and modeling problems that are usually formulated as variational problems, such as surface reconstruction or surface fitting problems solved with physics-based deformable models.

Some related papers [31, 32] can be retrieved from the IBM web server (http://www.watson.ibm.com:8080).

REFERENCES


APPENDIX

We first analyze those cases where the matrix $W$ can be factorized as a product of a symmetric matrix $E$ times a diagonal matrix $D$. Such is the case for the first order neighborhood of a shape with equal weights $w_{ij} = 1/[1^n]$ in each neighborhood $r_x^n$. In this case $E$ is the matrix whose $i$-th element is equal to 1 if vertices $v_i$ and $v_j$ are neighbors, and 0 otherwise, and $D$ is the diagonal matrix whose $i$-th diagonal element is $1/[1^n]$. Since in this case $W$ is a normal matrix [9], because $D^{-1/2}WD^{-1/2} = D^{-1/2}ED^{-1/2}$ is symmetric, $W$ has all real eigenvalues, and sets of $n$ left and right eigenvectors that form respective bases of $n$-dimensional space. Furthermore, by construction, $W$ is also a stochastic matrix, a matrix with nonnegative elements and rows that add up to one [27]. The eigenvalues of a stochastic matrix are bounded above in magnitude by 1, which is the largest magnitude eigenvalue. It follows that the eigenvalues of the matrix $K$ are real, bounded below by 0, and above by 2. Let $0 \leq k_1 \leq k_2 \leq \cdots \leq k_n \leq 2$ be the eigenvalues of the matrix $K$, and let $v_1, v_2, \ldots, v_n$ be a set of linearly independent unit length right eigenvectors associated with them.

When the neighborhood structure is not symmetric, the eigenvalues and eigenvectors of $W$ might not be real, but as long as the eigenvalues are not repeated, the decomposition of equation (3), and the analysis that follows, are still valid. However, the behavior of our fairing algorithm in this case will depend on the distribution of eigenvalues in the complex plane. The matrix $W$ is still stochastic here, and so all the eigenvalues lie on a unit circle $|k_1| = 1$. If all the eigenvalues of $W$ are very close to the real line, the behavior of the fairing algorithm should be essentially the same as in the symmetric case. This seems to be the case when very few neighborhoods are made non-symmetric. But in general, the problem has to be analyzed on a case by case basis.