RC-19923(#88210) 02/01/95
Computer Sciences 8 pages

Research Report

A SIGNAL PROCESSING APPROACH TO FAIR SURFACE DESIGN

Gabriel Taubin

IBM T.J.Watson Research Center
P.O.Box 704, Yorktown Heights, NY 10598
email:taubin@watson.ibm.com

IBM Research Division
Yorktown Heights, New York • San Jose, California • Zurich, Switzerland

A Signal Processing Approach To Fair Surface Design

Gabriel Taubin[1]
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*

General Terms: Algorithms, Graphics.

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 [33].

Since most existing algorithms based on fairness norm optimization [34, 24, 12, 35] 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. Unless these large surfaces are first simplified [29, 13, 11], or re-meshed using far fewer faces [32], 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, 31, 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, 35, 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 [35], 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 [37]. 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 n-periodic signal – a function defined on a regular polygon of n 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

[1]IBM T.J.Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598, taubin@watson.ibm.com

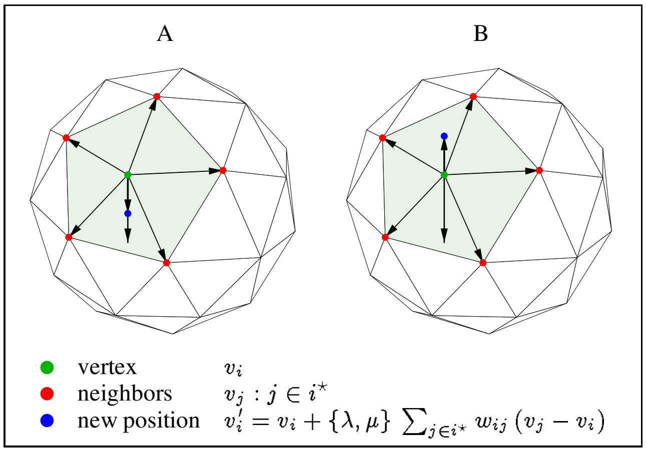

Figure 1: The two weighted averaging steps of our fairing algorithm. (A) A first step with positive scale factor λ is applied to all the vertices. (B) Then a second step with negative scale factor μ is applied to all the vertices.

vertices of the polygon. The discrete Laplacian of x is defined as

$$\Delta x_i = \frac{1}{2}(x_{i-1} - x_i) + \frac{1}{2}(x_{i+1} - x_i) \ , \tag{1}$$

where the indices are incremented and decremented modulo n. In matrix form it can be written as follows

$$\Delta x = -K\, x \ , \tag{2}$$

where K is the circulant matrix

$$K = \frac{1}{2}\begin{pmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{pmatrix} .$$

Since K 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

$$k_j = 1 - \cos(2\pi \lfloor j/2 \rfloor / n) \ ,$$

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

$$(u_j)_h = \begin{cases} \sqrt{1/n} & \text{if } j = 1 \\ \sqrt{2/n}\, \sin(2\pi h \lfloor j/2 \rfloor / n) & \text{if } j \text{ is even} \\ \sqrt{2/n}\, \cos(2\pi h \lfloor j/2 \rfloor / n) & \text{if } j \text{ is odd} . \end{cases}$$

Note that $0 \le k_1 \le \cdots \le k_n \le 2$, 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 $u_1, \ldots, u_n$

$$x = \sum_{i=1}^{n} \xi_i\, u_i \ , \tag{3}$$

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.

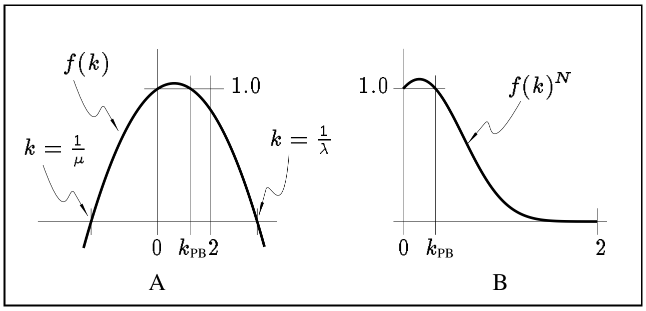

Figure 2: (A) Graph of transfer function $f(k) = (1 - \mu k)(1 - \lambda k)$ of non-shrinking smoothing algorithm.

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, where the filtered signal is the convolution of the original signal with a low-pass filter kernel. A more detailed analysis of other filter methodologies is beyond the scope of this paper, and will be done elsewhere. Perhaps the most popular convolution-based smoothing method for parameterized curves is the so-called *Gaussian filtering* method, associated with scale-space theory [36, 17]. In its simplest form, it can be described by the following formula

$$x_i' = x_i + \lambda\, \Delta x_i \ , \tag{4}$$

where $0 < \lambda < 1$ is a scale factor (for $\lambda < 0$ and $\lambda \ge 1$ the algorithm enhances high frequencies instead of attenuating them). This can be written in matrix form as

$$x' = (I - \lambda K)\, x \ . \tag{5}$$

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 $i^\star$ of indices of vertices. If the index j belongs to the neighborhood $i^\star$, 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 $\{i^\star : i = 1, 2, \ldots, n\}$. 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

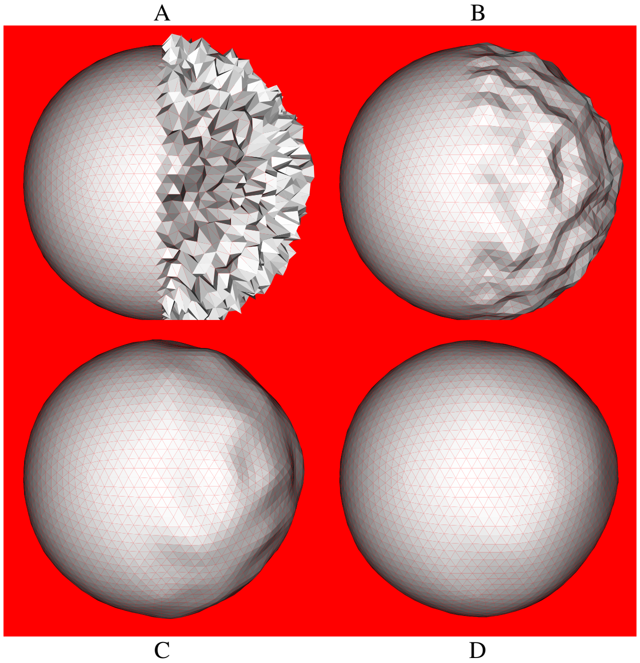

Figure 3: (A) Sphere partially corrupted by normal noise. (B) Sphere (A) after 10 non-shrinking smoothing steps. (C) Sphere (A) after 50 non-shrinking smoothing steps. (D) Sphere (A) after 200 non-shrinking smoothing steps. Surfaces are flat-shaded to enhance the faceting effect.

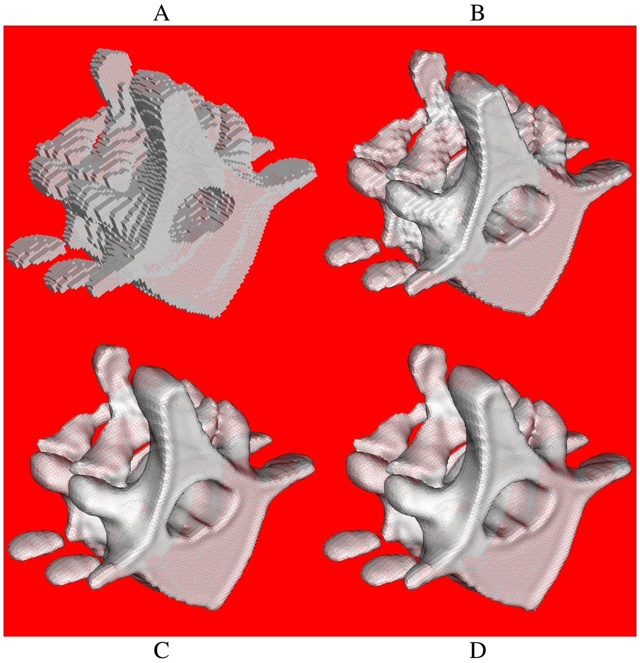

Figure 4: (A) Boundary surface of voxels from a CT scan. (B) Surface (A) after 10 non-shrinking smoothing steps. (C) Surface (A) after 50 non-shrinking smoothing steps. (D) Surface (A) after 100 non-shrinking smoothing steps. $k_{\mathrm{PB}} = 0.1$ and $\lambda = 0.6307$ in (B), (C), and (D). Surfaces are flat-shaded to enhance the faceting effect.

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 $i^\star = \{i-1, i+1\}$. The first order neighborhood 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. The discrete Laplacian of a discrete surface signal is defined by weighted averages over the neighborhoods

$$\Delta x_i = \sum_{j \in i^\star} w_{ij} \, (x_j - x_i) \; , \tag{6}$$

where the weights w_{ij} are positive numbers that add up to one for each i

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

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 $1/|i^\star|$ of vertex v_i, for each element j of $i^\star$. Note that the case of the Laplacian of a n-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 $\phi(v_i, v_j) = \phi(v_j, v_i)$ defined on the edges of the surface

$$w_{ij} = \frac{\phi(v_i, v_j)}{\sum_{h \in i^\star} \phi(v_i, v_h)} \; .$$

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 $0 \le k_1 \le k_2 \le \cdots \le k_n \le 2$ 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

$$x' = f(K)x = \sum_{i=1}^{n} f(k_i)\xi\, u_i \ ,$$

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 $f(k_i)^N \approx 0$ for high frequencies in the region of interest $k \in [0,2]$. Our choice is

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

where $0 < \lambda$, and μ 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 λ for all the vertices – the shrinking step –, we then perform another similar step

$$x_i' = x_i + \mu\, \Delta x_i \tag{8}$$

for all the vertices, but with negative scale factor μ instead of λ – 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, let us denote it k_{PB} (the *pass-band frequency*), such that $f(k_{PB}) = 1$. The value of k_{PB} is

$$k_{PB} = \frac{1}{\lambda} + \frac{1}{\mu} > 0 \ . \tag{9}$$

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. 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 values 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 λ and N (μ comes out of equation (9) afterwards). Of course we want to minimize N, the number of iterations. To do so, λ 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 λ so that $f(1) = -f(2)$. For $k_{PB} < 1$ this choice of λ ensures a stable and fast filter.

Figures 3 and 4 show examples of large surfaces faired 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. 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 [35], 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

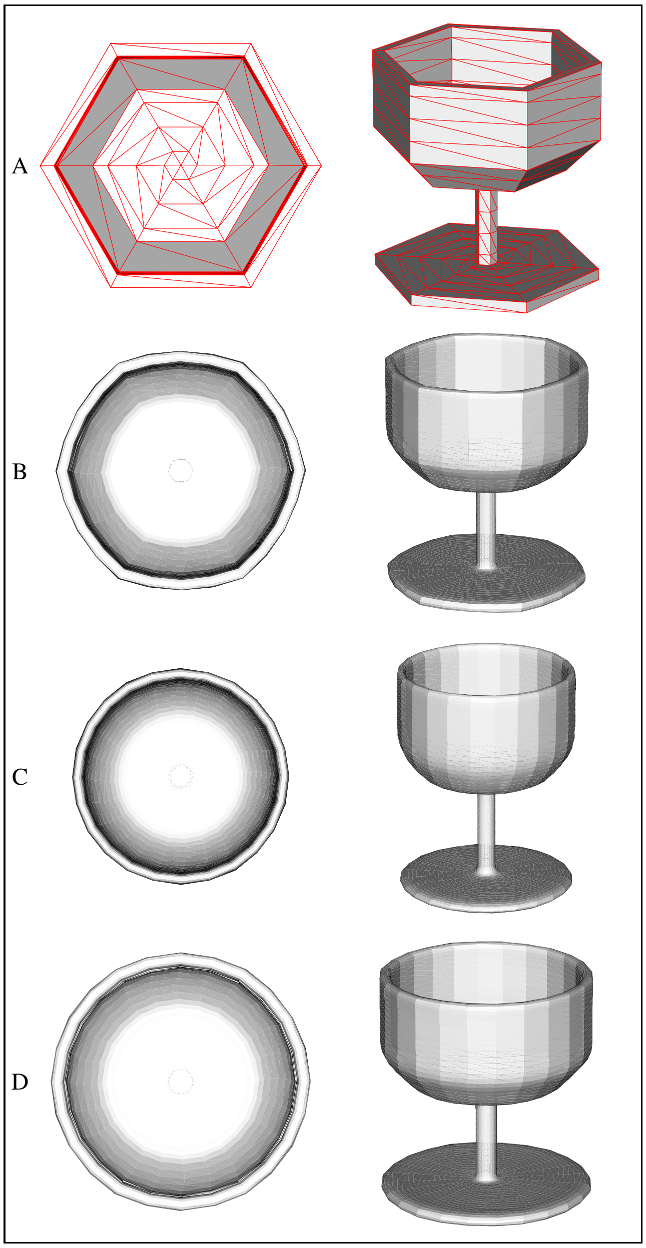

Figure 5: Surfaces created alternating subdivision and different smoothing steps. (A) Skeleton surface. (B) One Gaussian smoothing step ($\lambda = 0.5$). Note the hexagonal symmetry because of insufficient smoothing. (C) Five Gaussian smoothing steps ($\lambda = 0.5$). Note the shrinkage effect. (D) 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.

it progreses trough 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 surface are moved. The Catmull and Clark refinement process creates polyhedral surfaces with quadrilateral faces, and Loop refinement process subdivides each triangular face into for 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 Gausian 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 runing 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 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 vertives of the original skeleton. In this section we show what 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

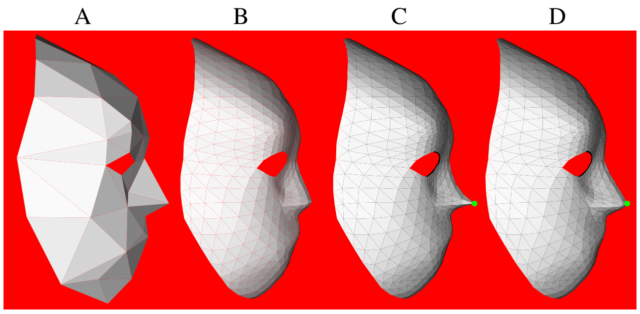

Figure 6: Example of surfaces designed using subdivision and smoothing steps with one interpolatory constraint. (A) Skeleton. (B) Surface (A) after two levels of subdivision and smoothing without constraints. (C) Same as (B) but with non-smooth interpolatory contraint. (D) Same as (B) but with smooth interpolatory contraint. Surfaces are flat-shaded to enhance the faceting effect.

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 v_1, i.e., if the neighborhood of v_1 is empty, then the value x_1 of any discrete surface signal x does not change during the fairing process, because the discrete Laplacian Δx_1 is equal to zero by definition of empty sum. Other vertices are allowed to have v_1 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 satisfyies 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 x_C^N as a sum of the corresponding unconstrained smooth signal $x^N = F\, x$ after N steps of our fairing 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 fairing algorithm to it. The result, let us denote it F_{n1}, the first column of the matrix

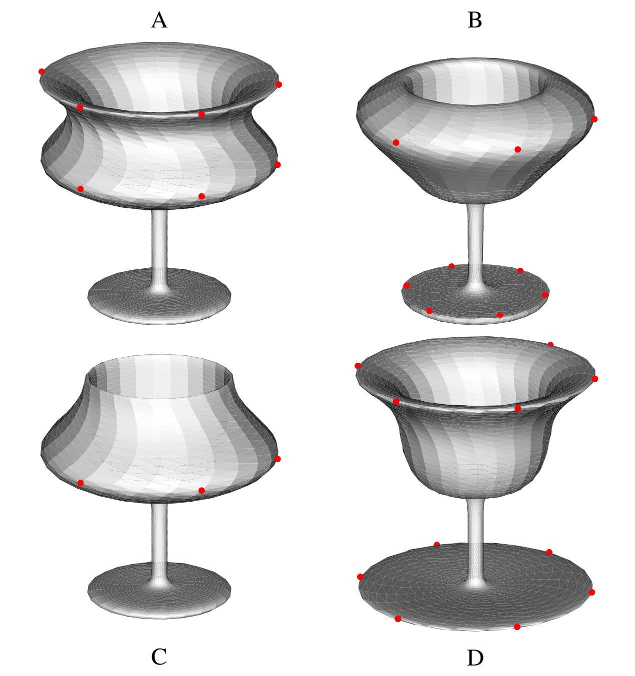

Figure 7: Examples of using subdivision and smoothing with smooth interpolatory constraints as a design tool. All the surfaces have been obtained by applying two levels of subdivision and smoothing with various parameters to the skeleton surface of figure 5-A . Constrained vertices are marked with red dots. Surfaces are flat-shaded to enhance the faceting effect.

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 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., $(x_C^N)_1 = x_1, \ldots, (x_C^N)_m = x_m$. 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

$$x_C^N = x^N + F_{nm} F_{mm}^{-1} \begin{pmatrix} x_1 - x_1^N \\ \vdots \\ x_m - x_m^N \end{pmatrix} \; , \tag{10}$$

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

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

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 is constraint to remain constant can be trivially generalized to allow for arbitrary smooth deformations of a surface. To do so, if in equation (10), 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 [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 $\delta_1, \ldots, \delta_n$. 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. 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_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 8-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.

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

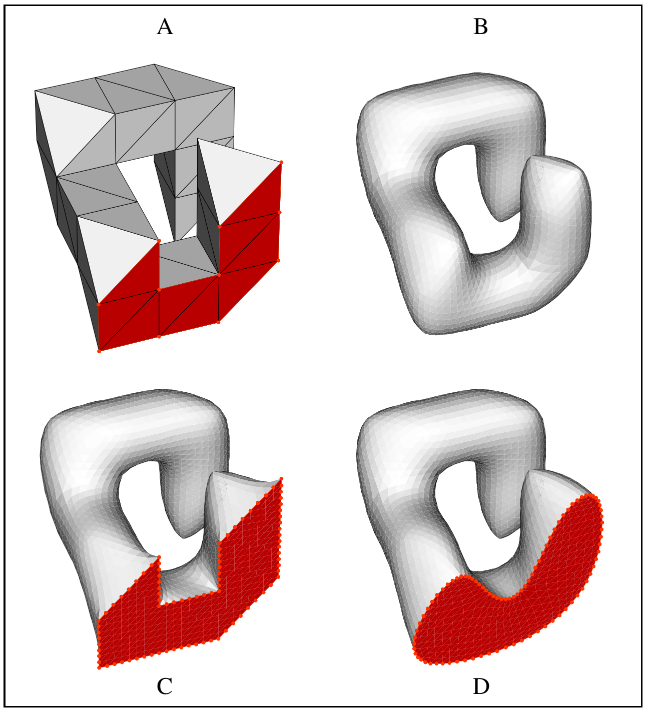

Figure 8: (A) Skeleton with marked vertices. (B) Surface (A) after three levels of subdivision and smoothing without constraints. (C) Same as (B) but with empty neighborhoods of marked vertices. (D) Same as (B) but with hierarchical neighborhoods, where marked vertices have label 1 and unmarked vertices have label 0. Surfaces are flat-shaded to enhance the faceting effect.

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

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

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 [7]

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

Because of this fact, we can define the vector Δv_i as the normal vector to the polyhedral surface at v_i. If N_i is the desired normal direction at vertex v_i after the fairing process, and S_i and T_i are two linearly independent vectors tangent to N_i, The surface after N iterations of the fairing 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 fairing with general linear constraints.

4.6 GENERAL LINEAR CONSTRAINTS

We consider here the problem of fairing a discrete surface signal x under general linear constraints $Cx_C^N = c$, 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 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 conditioned 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. $x_{(1)}$ 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 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 a IBM RS6000 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.

REFERENCES

[1] C.L. Bajaj and Ihm. Smoothing polyhedra using implicit algebraic splines. *Computer Graphics*, pages 79–88, July 1992. (Proceedings SIGGRAPH'92).

[2] H. Baker. Building surfaces of evolution: The weaving wall. *International Journal of Computer Vision*, 3:51–71, 1989.

[3] P. Borrel. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, April 1994.

[4] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10:350–355, 1978.

[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] T.D. DeRose, M. Lounsbery, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. Technical Report 93-10-05, Department of Computer Science and Enginnering, University of Washington, Seattle, 1993.

[7] M. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[8] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10:356–360, 1978.

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

[10] G. Greiner and H.P. Seidel. Modeling with triangular b-splines. *IEEE Computer Graphics and Applications*, 14(2):56–60, March 1994.

[11] A. Guéziec and R. Hummel. The wrapper algorithm: Surface extraction and simplification. In *IEEE Workshop on Biomedical Image Analysis*, pages 204–213, Seattle, WA, June 24–25 1994.

[12] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surface. *Computer Graphics*, pages 35–44, August 1993. (Proceedings SIGGRAPH'93).

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics*, pages 19–26, August 1993. (Proceedings SIGGRAPH'93).

[14] 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).

[15] A.D. Kalvin. *Segmentation and Surface-Based Modeling of Objects in Three-Dimensional Biomedical Images*. PhD thesis, New York University, New York, March 1991.

[16] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[17] T. Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):234–254, March 1990.

[18] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Dept. of Mathematics, University of Utah, August 1987.

[19] C. Loop. A G^1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, 11:303–330, 1994.

[20] C. Loop. Smooth spline surfaces over irregular meshes. *Computer Graphics*, pages 303–310, July 1994. (Proceedings SIGGRAPH'94).

[21] W. Lorenson and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, pages 163–169, July 1987. (Proceedings SIGGRAPH).

[22] M. Lounsbery, S. Mann, and T. DeRose. Parametric surface interpolation. *IEEE Computer Graphics and Applications*, 12(5):45–52, September 1992.

[23] J. Menon. Constructive shell representations for freeform surfaces and solids. *IEEE Computer Graphics and Applications*, 14(2):24–36, March 1994.

[24] H.P. Moreton and C.H. Séquin. Functional optimization for fair surface design. *Computer Graphics*, pages 167–176, July 1992. (Proceedings SIGGRAPH'92).

[25] J. Oliensis. Local reproducible smoothing without shrinkage. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):307–312, March 1993.

[26] 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.

[27] E. Seneta. *Non-Negative Matrices, An Introduction to Theory and Applications*. John Wiley & Sons, New York, 1973.

[28] L.A. Shirman and C.H. Séquin. Local surface interpolation with bezier patches. *Computer Aided Geometric Design*, 4:279–295, 1987.

[29] W.J. Shroeder, A. Zarge, and W.E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, pages 65–70, 1992. (Proceedings SIGGRAPH'92).

[30] 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.

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

[32] G. Turk. Re-tiling polygonal surfaces. *Computer Graphics*, pages 55–64, July 1992. (Proceedings SIGGRAPH'92).

[33] G. Turk and M. Levoy. Zippered polygon meshes from range data. *Computer Graphics*, pages 311–318, July 1994. (Proceedings SIGGRAPH'94).

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

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

[36] A.P. Witkin. Scale-space filtering. In *Proceedings, 8th. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1019–1022, Karlsruhe, Germany, August 1983.

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

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 $w_{ij} = 1/|i^\star|$ in each neighborhood $i^\star$. In this case E is the matrix whose ij-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 $1/|i^\star|$. 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 $u_1, u_2, \ldots, u_n$ 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 $|k_i - 1| < 1$. 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.