RC-20404(#90237) 3/12/96
Computer Sciences 20 pages

Research Report

OPTIMAL SURFACE SMOOTHING AS FILTER DESIGN

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

.

Tong Zhang and *Gene Golub*
Computer Science Department
Stanford University
Stanford, CA 94305
email:{tzhang,golub}@cs.stanford.edu

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

OPTIMAL SURFACE SMOOTHING AS FILTER DESIGN

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

.

Tong Zhang and *Gene Golub*
Computer Science Department
Stanford University
Stanford, CA 94305
email:{tzhang,golub}@cs.stanford.edu

ABSTRACT:

For a number of computational purposes, including visualization, smooth surfaces are approximated by polyhedral surfaces. An inherent problem of these approximation algorithms is that the resulting polyhedral surfaces appear faceted. A signal processing approach to smoothing polyhedral surfaces was recently introduced [10, 11]. Within this framework surface smoothing corresponds to low-pass filtering. In this paper we look at the filter design problem in more detail. We analyze the stability properties of the low-pass filter described in [10, 11], and show how to minimize its running time. Then we show that most classical techniques used to design finite impulse response (FIR) digital filters can also be used to design significantly faster smoothing filters. Finally, we describe an algorithm to estimate the power spectrum of a signal, and use it to evaluate the performance of the different filter design techniques described in the paper.

1. Introduction

The signal processing framework introduced in [10, 11], extends Fourier analysis to *discrete surface signals*, functions defined on the vertices of polyhedral surfaces. As in the method of Fourier Descriptors [12], where a closed curve is smoothed by truncating the Fourier series of its coordinate signals, a very large polyhedral surface of arbitrary topology is smoothed here by low-pass filtering its three surface coordinate signals. And although the formulation was developed mainly for signals defined on surfaces, it is in fact valid for *discrete graph signals*, functions defined on the vertices of directed graphs. Since this general formulation provides a unified treatment of polygonal curves, polyhedral surfaces, and even three-dimensional finite elements meshes, we start this paper by reviewing this formulation in its full generality.

Then we look at the filter design problem in more detail, with the main goal of minimizing the execution time of the low-pass filtering algorithm, given a desired frequency response specification. But we also take into consideration numerical issues, such as stability. We first study the tradeoffs that exists between minimizing execution time and maintaining the filter stable for the low-pass filter design of [10, 11]. Then we show that most classical finite impulse response (FIR) digital filter design techniques can be applied, with minor or no modifications in most cases, to the design of discrete graph signal filters. FIR filters, which in this framework correspond to sparse matrix multiplication, yield acceptable linear time and space complexity algorithms. Five to ten-fold speedups with respect to the low-pass filter design of [10, 11] can easily be obtained.

Then, we compare the performance of the different filter design methodologies with an algorithm to estimate the power spectrum of a discrete graph signal. This power spectrum estimator is implemented as a bank of high order band-pass filters, designed with the same techniques as the surface low-pass smoothing filters. However, the goal here is to design very sharp band-pass filters, not necessarily to minimize the order of the filter. We also use the power spectrum estimator to determine the pass-band frequency of the filter in such a way that shrinkage is prevented.

We end the paper with some experimental results and our conclusions.

2. Fourier Analysis of Discrete Graph Signals

In this section we describe the signal processing formulation of [10, 11] in its most general form, i.e., for *discrete graph signals*, functions defined on directed graphs. We represent a *directed graph* on the set $\{1,\ldots,n\}$ of n nodes as a set of *neighborhoods* $\{i^\star : i = 1,\ldots,n\}$, where $i^\star$ is a subset of nodes that we call the neighborhood of node i. If j is an element of $i^\star$ we say that j is a *neighbor* of i, and we visualize it as an arrow from i to j. In principle, except for prohibiting a node from being a neighbor of itself, we do not impose any other constraint on the neighborhoods. Note that j is allowed to be a neighbor of i without requiring i to be a neighbor of j, and neighborhoods can also be empty. We call a vector $x = (x_1,\ldots,x_n)^t$, with one component per node of the graph, a *discrete graph signal.*

We represent a polyhedral surface as a pair of arrays $S = \{V, F\}$, an array of n vertices

V, and an array of faces F. A vertex is a three-dimensional vector of real coordinates, and a face is a sequence of non-repeated indices of vertices representing a closed three-dimensional polygon. *Triangulated surfaces* are the most common, where all faces are triangles. We look at a polyhedral surface of n vertices as a directed graph, by labeling the vertices with distinct node numbers ranging from 1 to n, and defining a neighborhood for each node. We normally use *first order neighborhoods*, were node j is a neighbor of node i if i and j share an edge (or face), but other neighborhood structures can be used for other purposes, such as to impose certain types of constraints [11]. A *discrete surface signal* is a discrete graph signal defined on the associated graph. We visualize a discrete surface signal defined on a polyhedral surface as a piece-wise linear function defined on the surface. Discrete surface signals defined on polygonal curves, and on simplicial complexes of higher dimension, can be interpreted in a similar way.

The Fourier transform of a discrete graph signal cannot be defined in the traditional way because there is no notion of convolution. However, there is an alternative definition that can be generalized. Computing the Discrete Fourier Transform (DFT) of a signal defined on a closed polygon of n vertices is equivalent to decomposing the signal as a linear combination of the eigenvectors of the Laplacian operator

$$\Delta x_i = \frac{1}{2}(x_{i-1} - x_i) + \frac{1}{2}(x_{i+1} - x_i) \ , \tag{2.1}$$

were the Fourier transform is the vector of coefficients of the sum. To define the Fourier transform of a signal defined on an arbitrary directed graph we only have to define a linear operator that we will call the Laplacian operator. This is the same idea behind the method of eigenfunctions of Mathematical Physics [1].

We define the *Laplacian* of a discrete graph signal x by the formula

$$\Delta x_i = \sum_{j \in i^\star} w_{ij} \, (x_j - x_i) \tag{2.2}$$

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

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

These weights can be chosen in many different ways taking into consideration the neighborhoods, but in this paper we will assume that they are not functions of the signal x. Otherwise, the resulting operator is non-linear, and so, beyond the scope of this paper. 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 node i, for each element j of $i^\star$. Other choices of weights are discussed in [10, 11]. Note that the Laplacian of a signal defined on a closed polygon, described in equation (2.1), is a particular case of these definitions, with $w_{ij} = 1/2$, for $j \in i^\star = \{i-1, i+1\}$, for each node i.

If $W = (w_{ij})$ denotes the matrix of weights, with $w_{ij} = 0$ when j is not a neighbor of i, and $K = I - W$, the Laplacian of a discrete signal can be written in matrix form as

$$\Delta x = -Kx \ . \tag{2.3}$$

Although the method applies to general neighborhood structures, in this paper we will restrict our analysis to those cases where the matrix W can be factorized as a product of a symmetric matrix times a diagonal matrix $W = ED$. In this case the matrix W is a *normal matrix* [4], because the matrix

$$D^{1/2} W D^{-1/2} = D^{1/2} E D^{1/2} \tag{2.4}$$

is symmetric. Note that such is the case for the first order neighborhoods of a surface with equal weights $w_{ij} = 1/|i^\star|$ in each neighborhood $i^\star$, where E is the *incidence matrix* of the neighborhood structure (a symmetric matrix for first order neighborhoods), the matrix whose ij-th. element is equal to 1 if the nodes i and j are neighbors, and 0 otherwise; and D is the diagonal positive definite matrix whose i-th. diagonal element is $1/|i^\star|$. When W is a normal matrix it has all real eigenvalues, and sets of n left and right eigenvectors that form dual 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 [9]. The eigenvalues of a stochastic matrix are bounded above in magnitude by 1. It follows that the eigenvalues of the matrix K are real, bounded below by 0, and above by 2. Seen as discrete graph signals, the right eigenvectors of the matrix K can be considered as the *natural vibration modes*, and the corresponding eigenvalues as the associated *natural frequencies*. In our case, a vibration mode of high natural frequency corresponds to a rapid oscillation in the space domain. For example, for any directed graph, the constant signal $(1, \ldots, 1)$ is an eigenvector of K associated with the frequency $k = 0$, and the values of a natural vibration mode associated with a low natural frequency varies slowly when we move from a vertex to a neighbor vertex.

In the simple cases of signals defined on regular polygons, or more generally on graphs with group structure [3], the eigenvectors and eigenvalues of K have analytic expressions. The Fast Fourier Transform algorithm for signals defined on closed polygons is a good example of how this structure can be exploited. However, for the typical large graphs that we are interested in processing here, there are no analytic expressions for the eigenvalues and eigenvectors of K. And although a few extremal eigenvalues and eigenvectors of K can be computed with the Lanczos method [4], it is numerically impossible to reliably compute all of them. However, and this is the most significant observation, for filtering operations it is not necessary to compute the eigenvectors explicitly.

If $0 \leq k_1 \leq \cdots \leq k_n \leq 2$ are the eigenvalues of K, $e_1, \ldots, e_n$ a set of corresponding right eigenvectors, and $\delta_1, \ldots, \delta_n$ the associated dual basis of $e_1, \ldots, e_n$, the identity matrix I, and the matrix K can be written as follows

$$I = \sum_{i=1}^{n} e_i \delta_i^t \qquad K = \sum_{i=1}^{n} k_i \, e_i \delta_i^t \ ,$$

and every discrete graph signal x has a unique decomposition as a linear combination of $e_1, \ldots, e_n$

$$x = I\, x = \sum_{i=1}^{n} \hat{x}_i \, e_i \ , \tag{2.5}$$

where $\hat{x}_i = \delta_i^t x$. We call the vector $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)^t$ the Discrete Fourier Transform (DFT) of x, generalizing the classical definition for signals defined on closed polygons. Note, however, that this definition does not identify a unique object yet. If a different set of right eigenvectors of K is chosen, a different DFT is obtained. To complete the definition, if $W = ED$ with E symmetric, and D diagonal, The formula $\langle x, y\rangle_D = x^t D y$ defines an inner product in our space of signals, and normalizing the right eigenvectors of K to unit length with respect to the associated norm is equivalent to orthonormalizing them with respect to the inner product, and Parseval's formula is satisfied

$$\|x\|_D^2 = \|\hat{x}\|^2 \,, \tag{2.6}$$

where the norm on the right hand side is the Euclidean norm. That is, the frequency components $\hat{x}_i e_i$ of the signal x are orthogonal with respect to the inner product defined by D. We will assume from now on that the right eigenvectors of K are normalized in this fashion. These results will be used in sections 7 and 8.

To filter the signal x is to change its frequency distribution according to a transfer function $f(k)$

$$x' = \sum_{i=1}^{n} f(k_i)\, \hat{x}_i e_i = \left(\sum_{i=1}^{n} f(k_i)\, e_i \delta_i^t \right) x \,. \tag{2.7}$$

The frequency component of x corresponding the the natural frequency k_i is enhanced or attenuated by a factor $f(k_i)$. For example, the transfer function of an ideal low-pass filter, illustrated in figure 1, is

$$f_{\mathrm{LP}} = \begin{cases} 1 & \text{for } 0 \le k \le k_{\mathrm{PB}} \\ 0 & \text{for } k_{\mathrm{PB}} < k \le 2 \end{cases} \,, \tag{2.8}$$

where k_{PB} is the *pass-band frequency.* In this case, all the frequencies above the pass-band

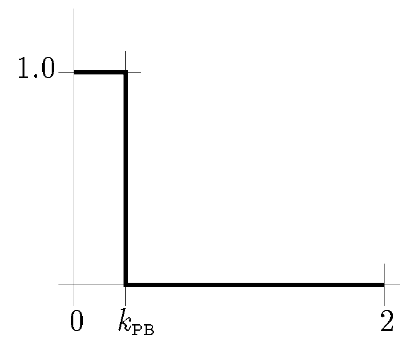

Figure 1: Graph of the ideal low-pass filter f_{LP}.

frequencies are removed, leaving only the low frequency components. The method of Fourier Descriptors [12] consists in filtering a discrete graph signal with an ideal low-pass filter transfer function. An efficient algorithm ($O(n \log(n))$) to ideal low-pass filter a signal defined on a closed polygon can be implemented using the Fast Fourier Transform algorithm. But in the general case of discrete graph signals, there is no efficient numerical method to compute its DFT, particularly when the number of nodes of the graph is very large. The computation can only be performed approximately, which is the main subject of this paper. To do this the ideal low-pass filter transfer function is replaced by an analytic approximation, usually a polynomial or rational function, for which the computation can be performed in an efficient manner. A wide range of analytic functions of one variable $f(k)$ can be evaluated in a matrix such as K [4]. The result is another matrix $f(K)$ with the same left and right eigenvectors, but with eigenvalues $f(k_1), \ldots, f(k_n)$

$$f(K) = \sum_{i=1}^{n} f(k_i)\, e_i \delta_i^t \ .$$

The main reason why the filtering operation $x' = f(K)\, x$ of equation (2.7) can be performed efficiently for a polynomial transfer function of low degree, is that when K is sparse, which is the case here, the matrix $f(K)$ is also sparse (but of wider bandwidth), and so, the filtering operation becomes the multiplication of a vector by a sparse matrix.

In Gaussian smoothing the transfer function is the polynomial $f_N(k) = (1 - \lambda k)^N$, with $0 < \lambda < 1$. But this transfer function produces shrinkage

$$\lim_{N \to \infty} (1 - \lambda k)^N = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{for } 0 < k \leq 2 \ . \end{cases}$$

That is, as N grows, the shape asymptotically converges to its centroid.

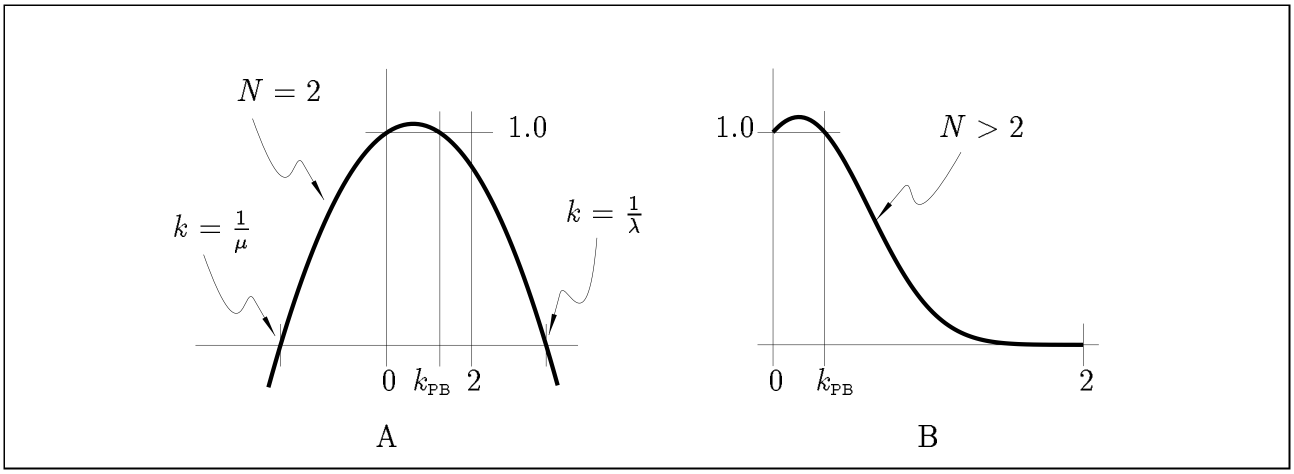

Figure 2: Graph of transfer function $f(k) = ((1 - \mu k)(1 - \lambda k))^{N/2}$. (A) $N = 2$. (B) $N > 2$. (out of scale)

The algorithm introduced in [10, 11] is escentially Gaussian smoothing with the difference that the scale factor λ changes from iteration to iteration, alternating between a positive value λ and a negative value μ. This simple modification still produces smoothing, but prevents shrinkage. The transfer function is the polynomial $f_N(k) = ((1 - \lambda k)(1 - \mu k))^{N/2}$, with $0 < \lambda < -\mu$ and N even, illustrated in figure 2-A for $N = 2$, and in 2-B for $N > 2$. This displays a typical *low-pass filter* shape in the region of interest, from $k = 0$ to $k = 2$. The *pass-band frequency* of this filter is defined as the unique value of k in the interval $(0, 2)$ such that $f_N(k) = 1$. Such a value exists when $0 < \lambda < -\mu$, and turns out to be equal to $k_{\text{PB}} = 1/\lambda + 1/\mu$. This polynomial transfer function of degree N results in a linear time and space complexity algorithm, which is very simple to implement, and produces smoothing without shrinkage. From now on we will refer to this algorithm as the $\lambda\!-\!\mu$ algorithm. However, as we will see below, faster algorithms can be achieved by choosing as transfer function a beter polynomial approximation of the same degree of the ideal low-pass filter.

3. Fast Smoothing as Filter Design

We are faced with the classical problem of digital filter design in signal processing [8, 5], but with some restrictions. Note that because of the linear complexity constraint discussed above, only polynomial transfer functions (FIR filters) are allowed. With rational transfer functions (IIR filters) better approximations of the ideal low-pass filter could be achieved with lower degrees of polynomials, but in our context a rational transfer function $f(k)=g(k)/h(k)$ involves solving the sparse linear system $h(K)\,x' = g(K)\,x$, which is not a linear complexity operation. Because of this reason, we leave the study of rational transfer functions for the future.

Because of space restrictions, of all the traditional FIR filter design methods available in the signal processing literature, we only cover here in some detail the method of windows, which is the simplest one. With this method we can design filters which are significantly faster, or sharper, than those obtain with the $\lambda-\mu$ algorithm for the same degree.

4. Optimizing the $\lambda\!-\!\mu$ algorithm

The $\lambda\!-\!\mu$ algorithm can be described in a recursive fashion as follows

$$f_N(k) = \begin{cases} 1 & N=0 \\ (1-\lambda_N\, k)\, f_{N-1}(k) & N>0 \end{cases}$$

where $\lambda_N = \lambda$, for N odd, and $\lambda_N = \mu$ for N even. Note that this algorithm requires minimum storage, only one array of dimension n to store the Laplacian of a signal if computed in place, and two arrays of dimension n in general. The algorithm is described in figure 3, where x is the input signal, and x' is the result of the filtering operation.

To maintain the minimum storage property and the same simple algorithmic structure, one could try to generalize by changing the scale factors λ_N from iteration to iteration in a different way. But if we start with a given pass-band frequency $k_{\rm PB} = 1/\lambda + 1/\mu$, as it is usually the case when one wants to *design* the filter, there are many values of λ and μ such that $0 < \lambda < -\mu$, that define a filter with the same pass-band frequency. In order for the polynomial $f(k) = (1-\lambda k)(1-\mu k)$ to define a low-pass filter in the interval $[0,2]$ it is necessary that $|f(k)| < 1$ in the stop-band region, so that $f_N(k) = f(k)^N \to 0$ when N grows. Since $f(k_{\rm PB}) = 1$ and $f(k)$ is strictly decreasing for $k > k_{\rm PB}$, this condition is equivalent to $f(2) > -1$, which translates into the following constraint on λ

$$\lambda < \frac{-k_{\rm PB} + \sqrt{(2-k_{\rm PB})^2+4}}{2(2-k_{\rm PB})}\,. \tag{4.1}$$

```
filter(N, λ_1, ..., λ_N, K, x, x')
    x^0 = x
    for j = 1 to N step 1 do
        x^1 = K x^0
        x^0 = x^0 - λ_j K x^1
    end
    x' = x^0
return
```

Figure 3: The $\lambda - \mu$ filtering algorithm.

Figure 4 shows examples of transfer functions of filters designed for the same pass-band frequency, but with different values of λ. As λ increases, the slope of the filter immediately after the pass-band frequency increases, i.e., the filter becomes sharper, but at the same time instability starts to develop at the other end of the spectrum, close to $k = 2$. If the maximum eigenvalue k_n of the matrix K is significantly less than 2 (which is not usually the case) we only need the filter to be stable in the interval $[0, k_n]$ (i.e., $1 > f(k_n) > -1$), and larger values of λ are acceptable. A good estimate of the maximum eigenvalue of K can be obtained with the Lanczos method [4]. Even if the maximum eigenvalue k_n is not known, the signal x to be smoothed might be band-limited, i.e., the coefficients $\hat{x}_i$ in equation (2.5) associated with high frequencies are all zero, or very close to zero. This condition might be difficult to determine in practice for a particular signal, but if we apply the algorithm with small λ for a certain number of iterations, the resulting signal becomes in effect band-limited. At this point λ can be increased keeping the pass-band frequency constant, maybe even making the filter unstable, and the algorithm can be applied again with the new values of λ and μ for more iterations. This process of increasing λ keeping the pass-band frequency constant can now be repeated again and again. A moderate speed-up is obtained in this way. Figure 5 show some examples of this process. All the filters in this figure produce almost the same response, but filter (F) is five times faster than filter (A).

5. Filter Design with Windows

The most straightforward approach to traditional digital filter design is to obtain a trigonometric polynomial approximation of the ideal filter transfer function by truncating its Fourier series. The resulting trigonometric polynomial minimizes the L_2 distance to the ideal filter transfer function among all the trigonometric polynomials of the same degree. Note that it is sufficient to know how to construct low-pass filters. A band-pass filter can be constructed as the difference of two low pass filters, and a high-pass filter can be constructed in a similar way. To obtain regular polynomials, not trigonometric ones, we first apply the change of variable $k = 2(1 - \cos(\theta))$. This change of variable is a $1 - 1$ mapping $[0, \pi/2] \rightarrow [0, 2]$. Then we extend

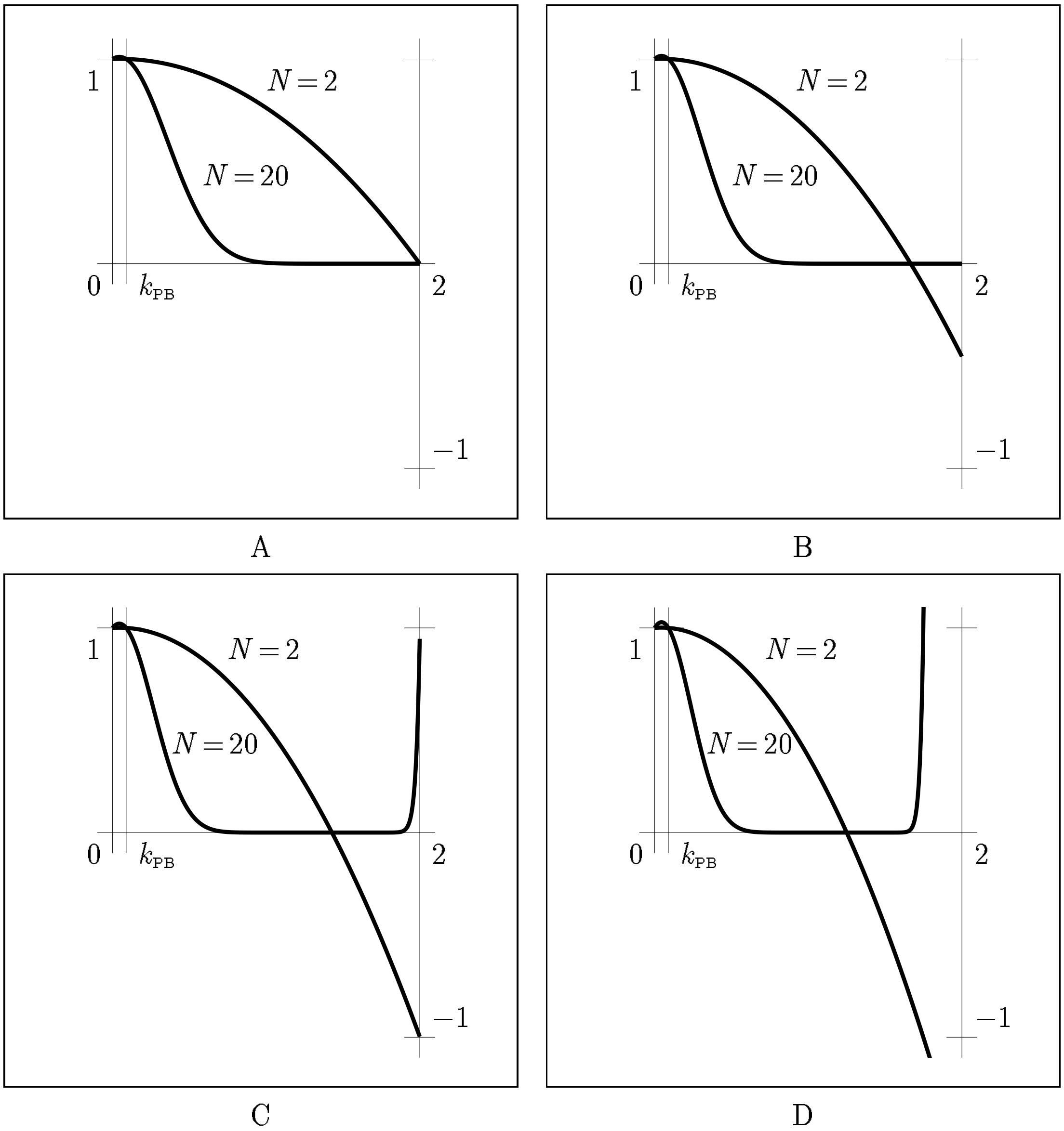

Figure 4: Graphs of transfer function $((1-\lambda k)(1-\mu k))^{N/2}$ for $N=2$ and $N=20$ and pass-band frequency $k_{\text{PB}} = 1/\lambda + 1/\mu = 0.09$. (A) $\lambda = 0.5$: stable. (B) $\lambda = 0.6$: stable. (C) $\lambda = 0.7$: limit case. (D) $\lambda = 0.8$: unstable.

the resulting function to the interval $[-\pi, \pi]$ as follows

$$h_{\text{LP}}(\theta) = \begin{cases} 0 & \pi/2 \leq \theta \leq \pi \\ f_{\text{LP}}(2(1-\cos(\theta))) & 0 \leq \theta \leq \pi/2 \\ h(-\theta) & -\pi \leq \theta \leq 0\,. \end{cases}$$

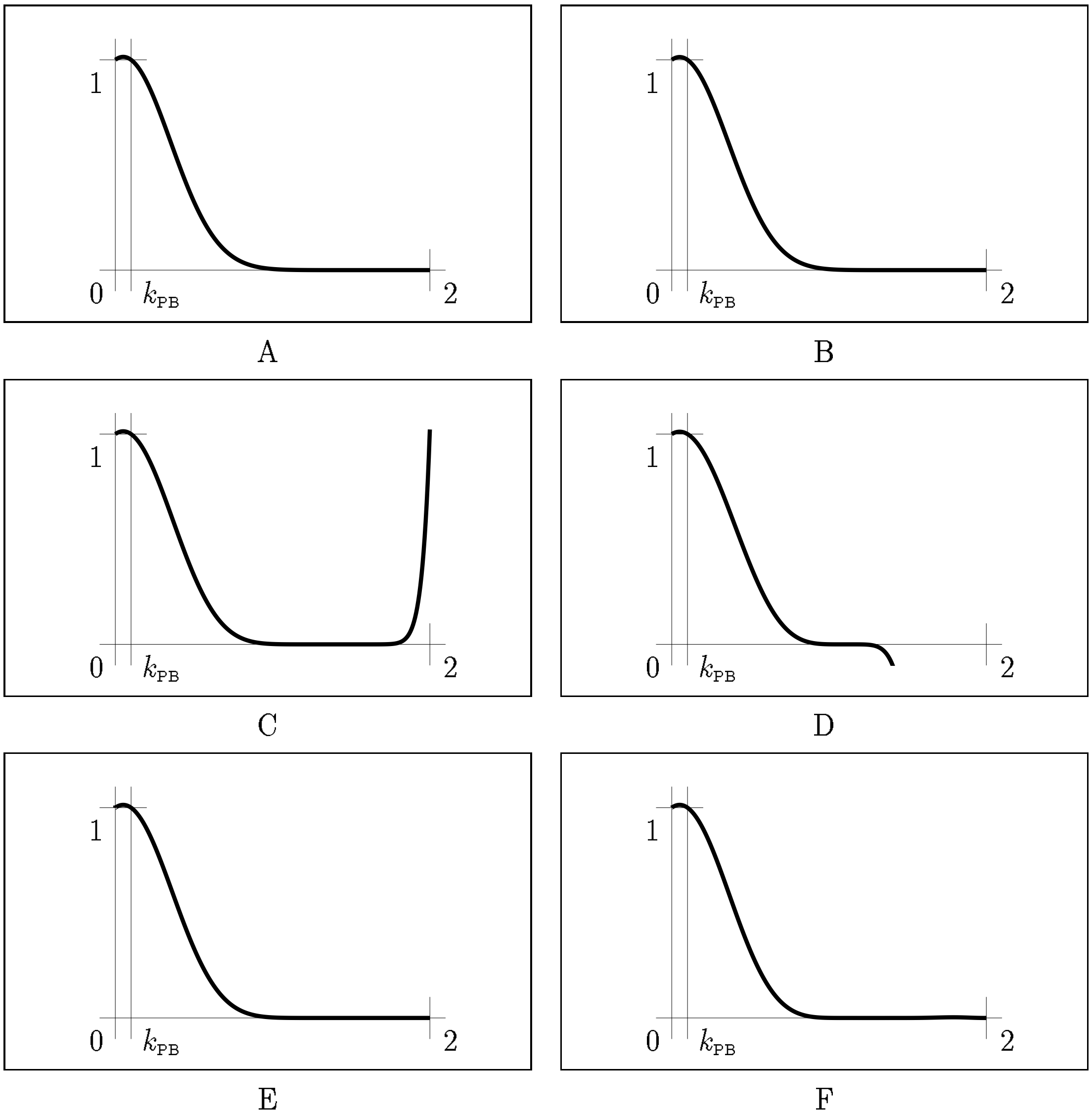

Figure 5: Different combinations of parameters in $((1-\lambda k)(1-\mu k))^{N/2}$ produce almost indistinguishable transfer functions. The pass-band frequency $k_{PB} = 1/\lambda + 1/\mu = 0.1$ is the same in the four cases. (A) $\lambda = 0.3$, $\mu = -0.3093$, $N = 120$. (B) $\lambda = 0.5$, $\mu = -0.5263$, $N = 40$. (C) $\lambda = 0.7$, $\mu = -0.7527$, $N = 20$. (D) $\lambda = 0.9$, $\mu = -0.9890$, $N = 12$. (E) $\lambda = 0.3$, $\mu = -0.3093$, $N = 12$, followed by $\lambda = 0.5$, $\mu = -0.5263$, $N = 12$, followed by $\lambda = 0.7$, $\mu = -0.7527$, $N = 12$. (F) $\lambda = 0.3$, $\mu = -0.3093$, $N = 6$, followed by $\lambda = 0.5$, $\mu = -0.5263$, $N = 6$, followed by $\lambda = 0.7$, $\mu = -0.7527$, $N = 6$, followed by $\lambda = 0.9$, $\mu = -0.9890$, $N = 6$. Note that (C) and (D) are unstable by themselves, but preceded by stable filters become stable. The degrees of the polynomials are (A) 120, (B) 40, (C) 20, (D) 12, (E) 36, and (F) 24.

Note that this function, periodic of period 2π and even, is also an ideal low-pass filter as a function of θ

$$h_{LP}(\theta) = \begin{cases} 1 & \text{if } |\theta| < \theta_{PB} \\ 0 & \text{otherwise} \end{cases},$$

where θ_{PB} is the unique solution of $k_{\text{PB}} = 2(1-\cos(\theta_{\text{PB}}))$ in $[0,\pi/2]$. Since $h(\theta)$ is an even function, it has a Fourier series expansion in terms of cosines only

$$h_{\text{LP}}(\theta) = h_0 + 2\sum_{n=0}^{\infty} h_n \cos(n\theta) \ ,$$

where h_n is

$$h_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} h(\theta)\,\cos(n\theta)\,d\theta = \begin{cases} \theta_{\text{PB}}/\pi & n=0 \\ \sin(n\,\theta_{\text{PB}})/n\,\pi & n>0 \end{cases} .$$

Now, it is well known that $\cos(n\,\theta) = T_n(\cos(\theta))$, where T_n is the n-th. Chebyshev polynomial [2], defined by the three term recursion

$$T_n(w) = \begin{cases} 1 & n=0 \\ w & n=1 \\ 2\,w\,T_{n-1}(w) - T_{n-2}(w) & n>1 \end{cases}$$

The N-th. polynomial approximation of f_{LP} for $k \in [0,2]$ is then

$$f_N(k) = \frac{\theta_{\text{PB}}}{\pi}\,T_0(1-k/2) + \sum_{n=1}^{N} \frac{2\sin(n\,\theta_{\text{PB}})}{n\,\pi}\,T_n(1-k/2) \ . \tag{5.1}$$

Figure 6 shows some of these polynomials compared with the polynomials $((1-\lambda k)(1-\mu k))^{N/2}$ for the same pass-band frequency.

As can be easily observed in figure 6, direct truncation of the series leads to the well-known Gibbs phenomenon, i.e., a fixed percentage overshoot and ripple before and after the discontinuity. As it is shown in section 9, this is one of the problems that makes this technique unsatisfactory. The other problem is that the resulting polynomial approximation does not necesarily satify the constraint $f_N(0) = 1$, which is required to preserve the average value of the signal (DC level in classical signal processing, centroid in the case of surfaces). Our experiments show that a desirable surface smoothing filter transfer function should be as close as possible to 1 within the pass-band as possible, and then decrease to zero in the stop-band ($[k_{\text{PB}}, 2]$).

Another classical technique to control the convergence of the Fourier series is to use a weighting function to modify the Fourier coefficients. In our case the polynomial approximation of equation (5.1) is modified as follows

$$f_N(k) = w_0\,\frac{\theta_{\text{PB}}}{\pi}\,T_0(1-k/2) + w_n \sum_{n=1}^{N} \frac{2\sin(n\,\theta_{\text{PB}})}{n\,\pi}\,T_n(1-k/2) \ , \tag{5.2}$$

where $w_0, w_1, \ldots, w_N$ are the weights that constitute a so called *window*. Since the multiplication of Fourier coefficients by a window corresponds to convolving the original frequency response with the Fourier series defined by the window, a design criterion for windows is to find a finite window whose Fourier transform has relatively small side lobes. The polynomial approximation of equation (5.1) is a particular case of (5.2), where the weights are all equal to

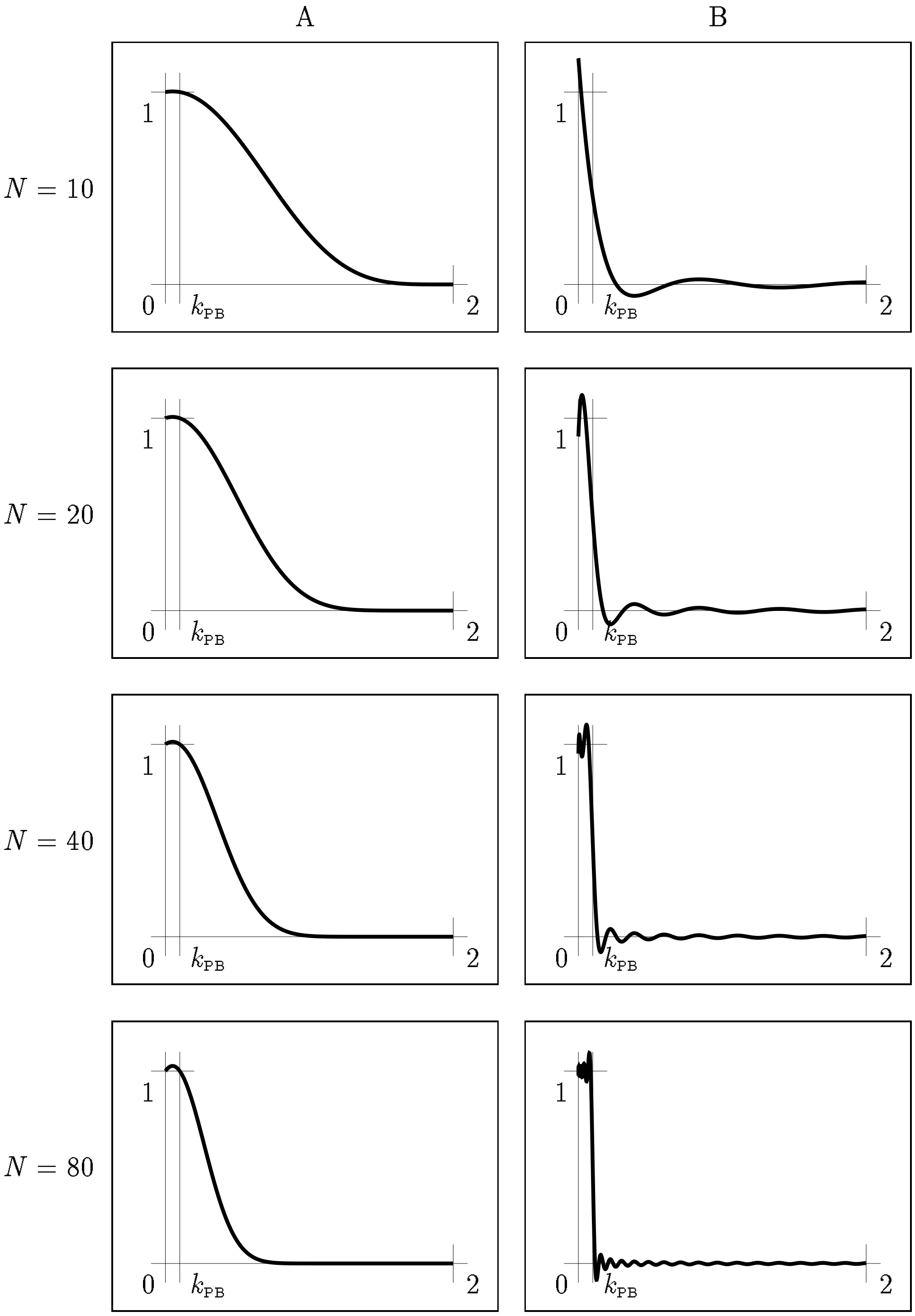

Figure 6: (A) Polynomial transfer function $f(k) = ((1-\lambda k)(1-\mu k))^{N/2}$ with $k_{\mathrm{PB}} = 0.1$ and $\lambda = 0.6307$. (B) Truncated Fourier series approximation of the ideal low-pass filter (Rectangular window).

1. This is called the *Rectangular window*. Other popular windows are, the *Hanning window*, the *Hamming window*, and the *Blackman window*.

$$w_n = \begin{cases} 1.0 & \text{Rectangular} \\ 0.5 + 0.5\ \cos(n\pi/(N+1)) & \text{Hanning} \\ 0.54 + 0.46\ \cos(n\pi/(N+1)) & \text{Hamming} \\ 0.42 + 0.5\ \cos(n\pi/(N+1)) + 0.08\ \cos(2n\pi/(N+1)) & \text{Blackman}\,. \end{cases} \tag{5.3}$$

The Fourier series of the rectangular window has a narrow center lobe, but its side lobes contain a large part of the total energy, and decay very slowly. The Hamming window has 99.96 percent of its energy in its main lobe, but the width of the main lobe is twice the width of the rectangular window's main lobe. The Blackman window further reduces the peak side lobe ripple at the expense of a main lobe whose width is about triple the width of the rectangular window's main lobe. There are other window designs that are optimal in one way or another [7, 6, 5], but the window coefficients are in some cases difficult to compute, and as we will see below, we can design satisfactory filters with the windows described above.

If the low-pass filter must have a very narrow pass-band region, which is usually the case in the surface smoothing application, then a high degree polynomial is necessary to obtain a reasonable approximation. This is in fact a consequence of the uncertainty principle. The phenomenon can be observed even in the case of the rectangular window, illustrated in figure 6. The problem is even worse for the other windows, because they have wider main lobes. To obtain a reasonably good approximation of degree N, the pass-band must be significantly wider than the width of the main lobe of the window. If σ is the width of the main lobe of the window, the resulting filter will be approximately equal to one for $\theta \in [0, \theta_{\text{PB}} - \sigma]$, approximately equal to zero for $\theta \in [\theta_{\text{PB}} + \sigma, \pi]$, and approximately decreasing for $\theta \in [\theta_{\text{PB}} - \sigma, \theta_{\text{PB}} + \sigma]$. Our solution in this case of narrow pass-band frequency, is to design the filter for a small value of N, but with the pass-band frequency increased by σ (no longer the width of the main lobe of the window)

$$f_N(k) = w_0 \frac{(\theta_{\text{PB}} + \sigma)}{\pi} T_0(1 - k/2) + w_n \sum_{n=1}^{N} \frac{2\sin(n(\theta_{\text{PB}} + \sigma))}{n\,\pi} T_n(1 - k/2)\ , \tag{5.4}$$

and then, eventually iterate this filter ($f(k) = f_N(k)^M$). The value of σ can be determined numerically by maximizing $f(k_{\text{PB}})$ under the constraints $|f(k)| < 1$ for $k_{\text{PB}} < k \leq 2$. In our implementation, we compute the optimal σ with a local root finding algorithm (a few Newton iterations) so that $f_N(k_{\text{PB}}) = 1$, starting from an interactively chosen initial value. Figure 7 shows some examples of filters designed in this way, compared with filters of the same degree and $\sigma = 0$, and with $\lambda - \mu$ filters of the same degree. Figure 8 shows several views of the filter design control panel of our interactive surface editing system.

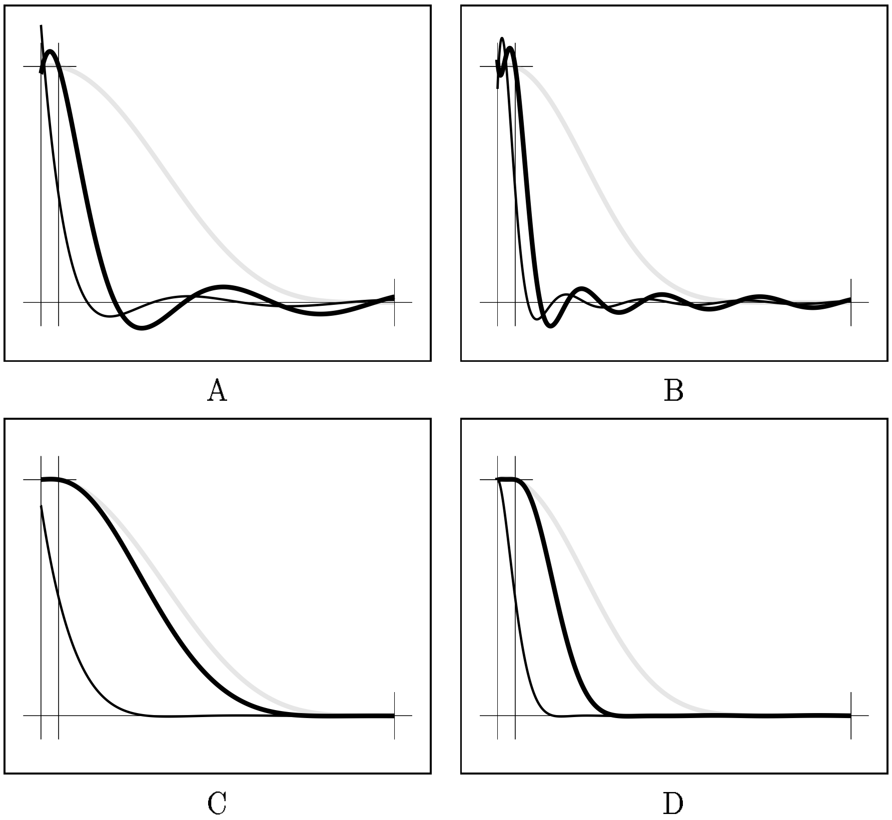

Figure 7: Filters $f_N(k)$ for $k_{\text{PB}} = 0.1$ and $\sigma > 0.0$. (A) Rectangular window, $N = 10$, $\sigma = 0.1353$. (B) Rectangular window, $N = 20$, $\sigma = 0.0637$. (C) Hamming window, $N = 10$, $\sigma = 0.5313$. (D) Hamming window, $N = 20$, $\sigma = 0.2327$. In each of the four cases the thick black line corresponds to the filter described above, the thin black line to the same filter with $\sigma = 0.0$, and the gray line is a $\lambda - \mu$ filter of the same degree and $\lambda = 0.5$.

6. Implementation

Figure 9 describes our algorithmic implementation of the filtering operation $x' = f_N(K)\,x$, where $f(k)$ is the transfer function

$$f(k) = \sum_{j=0}^{N} f_j\, T_j(1 - k/2)\ .$$

This algorithm applies not only to low-pass filters, but to any polynomial transfer function expressed as linear combination of Chebyshev polynomials (every polynomial can be written in this way). From the numerical point of view, the Chebyshev polynomials constitute a better basis than the power basis because they are orthogonal in the interval $[-1, 1]$, Furthermore, the design techniques described above produce polynomial coefficients with respect to this basis. In terms of storage, the algorithm only requires four auxiliary arrays x^0, x^1, x^2, x^3 of dimension n. In terms of computation, the most expensive operation is the multiplication of a vector by

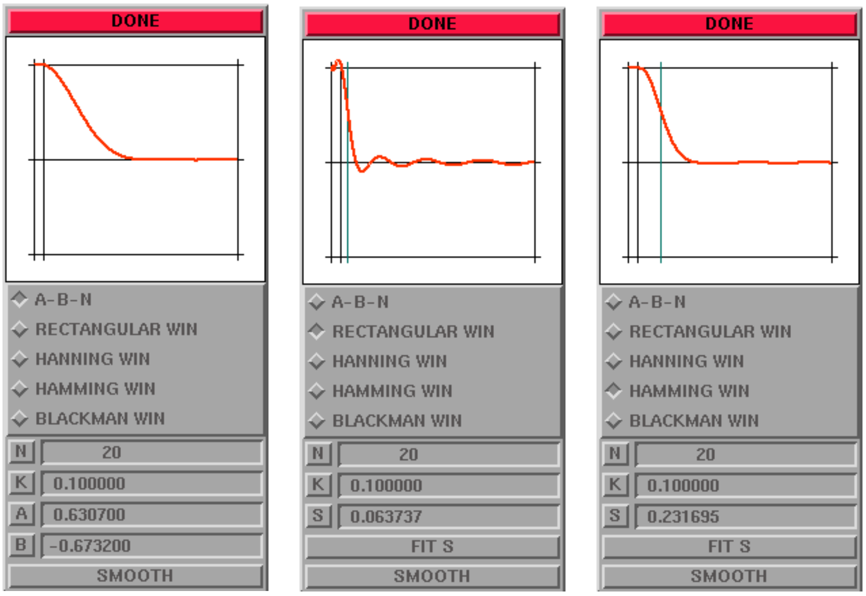

Figure 8: Interactive filter design subsystem.

the matrix K, an operation that is executed N times. This is the evaluation of the Laplacian, described in equation (2.2), which is also a linear complexity operation, because K is sparse.

filter$(N, f_0, \ldots, f_N, K, x, x')$

$\quad x^0 = x$

$\quad x^1 = Kx^0$

$\quad x^1 = x^0 - \frac{1}{2}x^1$

$\quad x^3 = f_0 x^0 + f_1 x^1$

$\quad$ **for** $j = 2$ **to** N **step** 1 **do**

$\qquad x^2 = Kx^1$

$\qquad x^2 = (x^1 - x^0) + (x^1 - x^2)$

$\qquad x^3 = x^3 + f_i x^2$

$\qquad x^0 = x^1$

$\qquad x^1 = x^2$

$\quad$ **end**

$\quad x' = x^3$

return

Figure 9: The filtering algorithm $x' = f(K)x$.

7. How to Choose The Pass-Band Frequency

So far in our discussion of how to design low-pass filters, the pass-band frequency $k_{\rm PB}$ was given. In this section we are concerned with how to choose the pass-band frequency to prevent shrinkage. If the signals are the coordinates of the vertices of a closed surface, preservation of the enclosed volume is a natural criterion. But even in this case, normalizing the filtered signal to make it satisfy the criterion is an expensive global operation that requires the evaluation of a surface integral. And since the criterion does not have a natural generalization to arbitrary discrete graph signals, we will use a different criterion, more related to the signal processing formulation. As in the classical case, since the DFT $\hat{x}$ of a signal x satisfies Parseval's formula, the value of $\hat{x}_i^2$ can be interpreted as the *energy content* of x in the frequency k_i. Similarly, the sum

$$\sum_{k_i \leq k_{\rm PB}} \hat{x}_i^2$$

measures the energy content of x in the pass-band. Our criterion is to choose the minimum pass-band frequency such that most of the energy of the signal falls in the pass-band, i.e., we choose $k_{\rm PB}$ such that

$$\sum_{k_i \leq k_{\rm PB}} \hat{x}_i^2 \geq (1-\epsilon)\,\|x\|_D^2 \ ,$$

where ϵ is a very small number. Of course, since we cannot compute the DFT of x, we cannot minimize this expression exactly. We can only get a rough estimate of the minimizer using the power spectral estimator described in the next section. What value of ϵ to use, and how accurate the estimation should be is application dependent, but in general it should be determined experimentally for a set of typical signals.

8. Power Spectrum Estimation

Ideally, to evaluate the performance of the different low-pass filter algorithms we should measure the DFT of the filter outputs, and check that the high frequency energy content is very small. Since we do not have any practical way of computing the DFT, we estimate the power spectrum, or energy distribution, of a signal as follows. We partition the interval $[0,2]$ into a small number of non-overlapping intervals $I^1,\ldots,I^M$, and for each one of this intervals we estimate the energy content of the signal within the interval. We do so by designing a very sharp (high degree) pass-band filter $f^j(k)$ for each interval I^j. The energy content of the signal x within the interval I^j can be estimated by measuring the total energy of the output of corresponding filter applied to the signal

$$\|f^j(K)x\|_D^2 \approx \sum_{k_i \in I^j} \hat{x}_i^2 \ .$$

By designing all these FIR filters of the same degree, a filter-bank, we can evaluate all of them simultaneously at a greatly reduced computational cost. The only disadvantage is that we need M arrays of the same dimension as the input signal x to accumulate the filter outputs before their norms are evaluated. If the pass-band filters were ideal, Parseval's formula implies that the sum of the total energies of the filter outputs must be equal to the total energy of the input signal. Since the transfer functions of the filters overlap, this condition is only approximately satisfied. But the error can be made arbitrarily small by increasing the degree of the polynomials.

Figure 8 shows several views of the spectrum estimation control panel of our interactive surface editing system. In this figure N is the degree of the filters in the filter bank, M is the

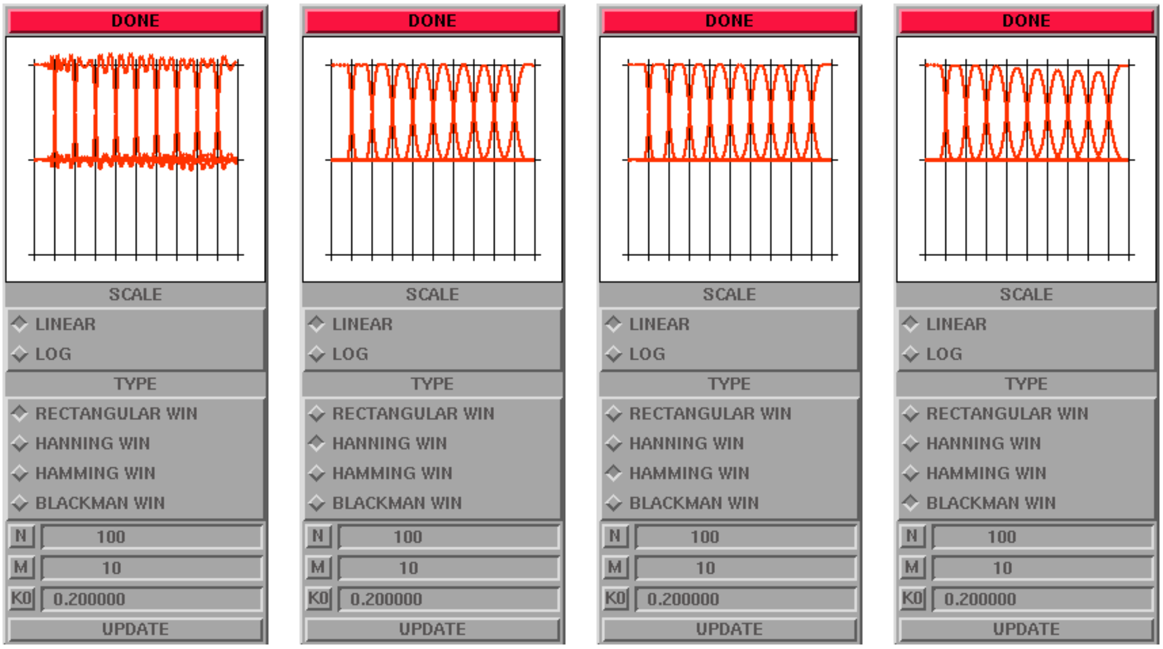

Figure 10: Interactive power spectrum estimation subsystem.

number of bands, and $K0$ is the width of each band. We recommend using filters designed with the Hanning or Hamming windows of a degree at lest ten times the number of spectrum bands.

9. Experimental Results

We have integrated all the methods described above within our surface editing and visualization system, illustrated in figure 11. Figure 12 shows the result of applying the filters of figure 7 to the same input surface. The spectrum estimate for the input surface yields the 99.88% of the energy in the band $[0, 0.1]$. This is a typical result for relatively large surfaces, and we have found that a default value $k_{\mathrm{PB}} = 0.1$ produces very good results. But as we pointed out before, the appropriate value for a family of similar signals must be determined experimentally by estimating the spectrum of a typical sample.

The $\lambda - \mu$ algorithm produces very good results, but to significantly reduce high frequencies, a relatively large number of iterations might be necessary. The results obtained with rectangular filters are unsatisfactory. They are somehow better when we increase the value of σ, as described in section 5, but although they are faster, they change the low frequencies components too much, altering the shape quite significantly.

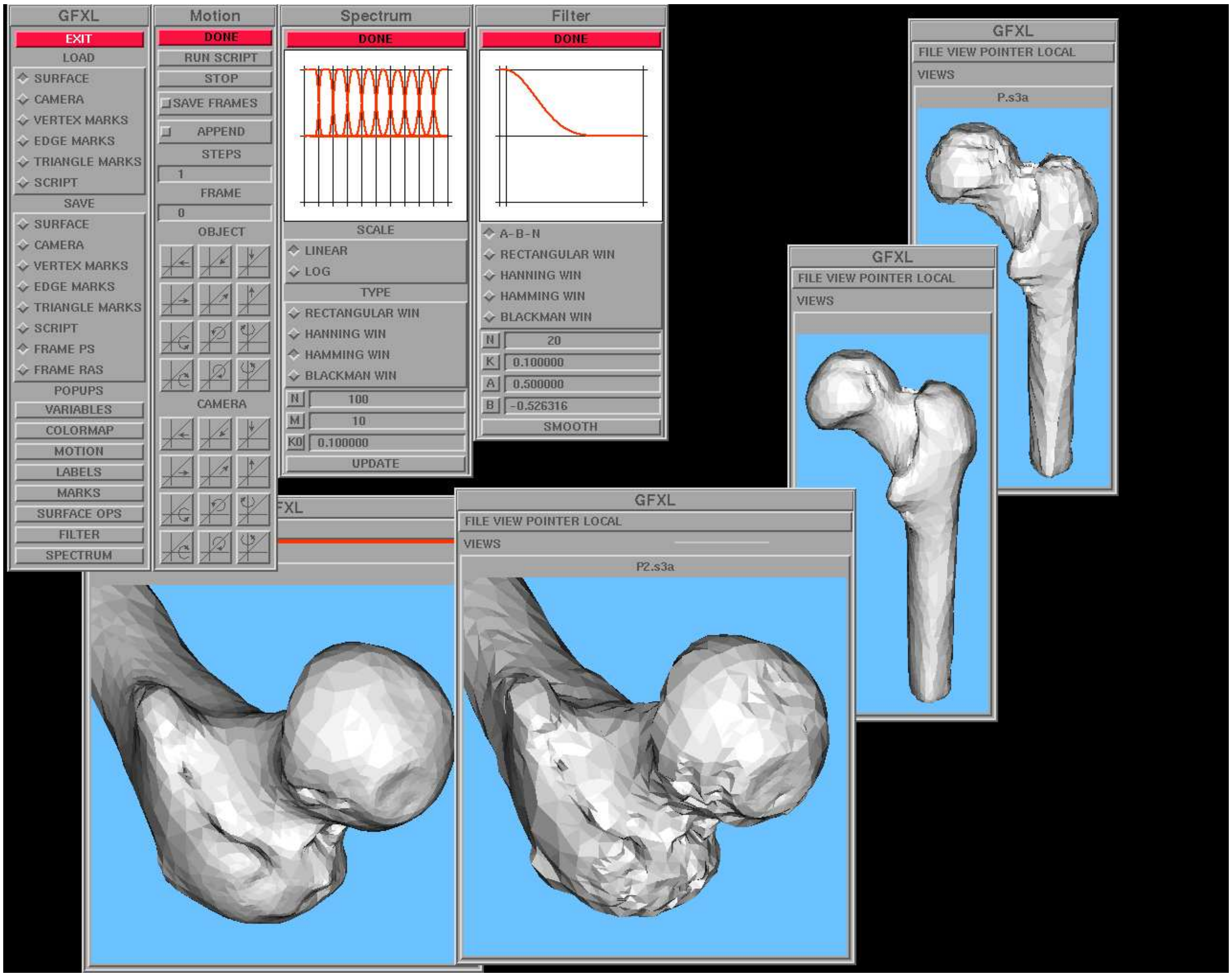

Figure 11: Interactive surface editing system.

The ideal transfer function should be as flat as possible in the pass-band region ($f(k) \approx 1$ for $k \in [0, k_{\text{PB}}]$), and then decrease as fast as possible in the stop-band region ($k \in [k_{\text{PB}}, 2]$). The transfer function of the $\lambda - \mu$ algorithm has this shape, but does not decrease fast enough in the stop-band. The filters designed with the other three windows (Hanning, Hamming, and Blackman), and with increased σ produce transfer functions of similar shape. The Blackman window produces transfer functions that are much flatter in the pass-band, but at the expense of a slower rate of decrease in the stop-band. Hanning and Hamming windows produce similar results, but the Hamming window produces transfer functions with less oscillations. As figure 12 shows, filters designed with the Hamming window produce filters of similar quality as the $\lambda - \mu$ algorithm, but much faster.

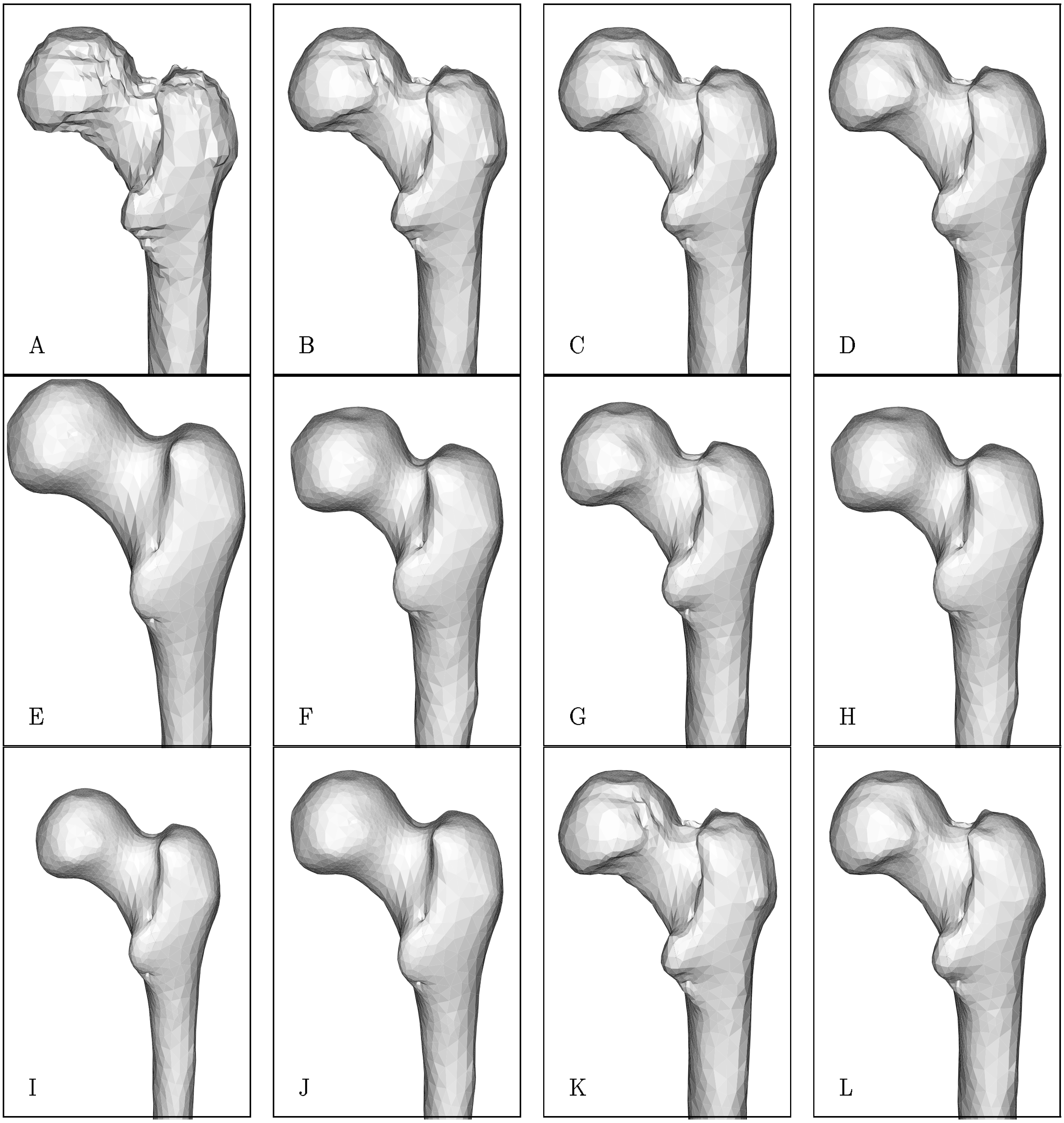

Figure 12: Filters of Figure 7 applied to the same surface. In all these examples $k_{\text{PB}} = 0.1$. (A) Input surface (2565 vertices, 5138 triangles). (B) $\lambda - \mu$ filter $\lambda = 0.5$ $n = 10$. (C) $\lambda - \mu$ filter $\lambda = 0.5$ $n = 20$. (D) $\lambda - \mu$ filter $\lambda = 0.5$ $n = 60$. (E) Rectangular window $\sigma = 0.0$ $n = 10$. (F) Rectangular window $\sigma = 0.0$ $n = 20$. (G) Rectangular window $\sigma = 0.01353$ $n = 10$. (H) Rectangular window $\sigma = 0.06374$ $n = 20$. (I) Hamming window $\sigma = 0.0$ $n = 10$. (J) Hamming window $\sigma = 0.0$ $n = 20$. (K) Hamming window $\sigma = 0.5313$ $n = 10$. (L) Hamming window $\sigma = 0.2327$ $n = 20$.

10. Conclusions

Generalizing the signal processing formulation of [10, 11], in this paper we formulated the most significant concepts of Fourier analysis for signals defined on oriented graphs, and showed that linear filters with polynomial transfer function can be implemented in an efficient manner, and designed with classical digital filter design methods. In particular, we have shown how to design surface smoothing filters that produce almost the same effect as the filter described in [10, 11], but in a fraction of the time. We have also described a method to estimate the power spectrum of a signal, and used this power spectrum estimate to determine the pass-band frequency for a surface smoothing filter, and as a tool to evaluate the performance of different filter designs. We have also given a formal definition of the shrinkage problem, which is valid not only for closed surfaces, but for any signal.

References.

[1] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1. Interscience, 1953.

[2] P.J. Davis. *Interpolation and Approximation.* Dover Publications, Inc., 1975.

[3] H. Dym and H.P. McKean. *Fourier Series and Integrals*, volume 14 of *Probability and Mathematical Statistics.* Academic Press, 1972.

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

[5] R.W. Hamming. *Digital Filters.* Prentice Hall, 1989.

[6] H.D. Helms. Nonrecursive digital filters: Design methods for achieving specifications on frequency response. *IEEE Transactions on Audio and Electroacustics*, AU-16:336–342, September 1968.

[7] J.F. Kaiser. Digital filters. In F.F. Kuo and J.F. Kaiser, editors, *System Analysis by Digital Computer.* Wiley, New York, 1966.

[8] A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing.* Prentice Hall, Englewood Cliffs, NJ, 1975.

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

[10] G. Taubin. Curve and surface smoothing without shrinkage. In *Proceedings, Fifth International Conference on Computer Vision*, pages 852–857, June 1995.

[11] G. Taubin. A signal processing approach to fair surface design. *Computer Graphics*, pages 351–358, August 1995. (Proceedings SIGGRAPH'95).

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