CURVE AND SURFACE SMOOTHING WITHOUT SHRINKAGE

Gabriel Taubin

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

Abstract

For a number of computational purposes, including visualization of scientific data and registration of multimodal medical data, smooth curves must be approximated by polygonal curves, and surfaces by polyhedral surfaces. An inherent problem of these approximation algorithms is that the resulting curves and surfaces appear faceted. Boundary-following and iso-surface construction algorithms are typical examples. To reduce the apparent faceting, smoothing methods are used. In this paper we introduce a new method for smoothing piece-wise linear shapes of arbitrary dimension and topology. This new method is in fact a linear low-pass filter that removes high curvature variations, and does not produce shrinkage. Its computational complexity is linear in the number of edges or faces of the shape, and the required storage is linear in the number of vertices.

Keywords

Medical Computer Vision, Shape and Object Representation, Low-Level Processing.

1 Introduction

In this paper, curves and surfaces will be referred to as *shapes*, and polygonal curves and polyhedral surfaces will be referred to as *piece-wise linear shapes*, respectively.

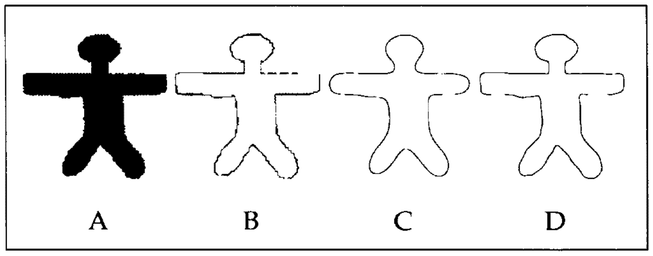

Figure 1: (A) A binary image. (B) The boundary curve of (A) appears faceted. (C) The result of applying Gaussian smoothing to (B). (D) The result of applying the smoothing method of this article to (B).

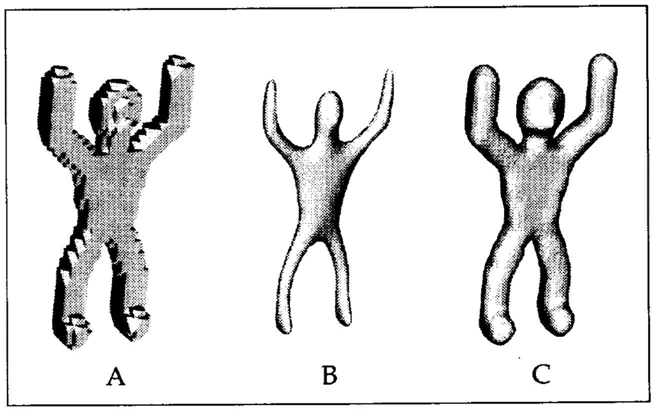

Figure 2: (A) A surface constructed with an iso-surface construction algorithm appears faceted. (B) The result of applying Gaussian smoothing to (A). (C) The result of applying the smoothing method of this article to (A).

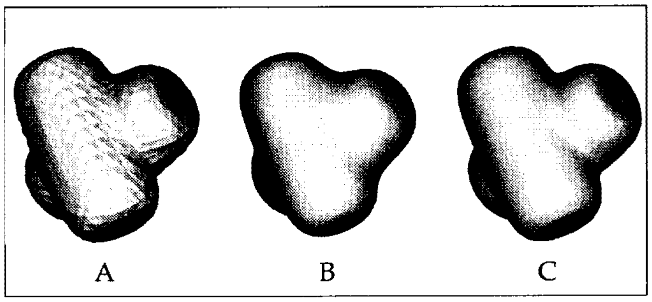

Figure 3: (A) A surface constructed with an iso-surface construction algorithm appears faceted. (B) The result of applying Gaussian smoothing to (A). (C) The result of applying the smoothing method of this article to (A).

Most existing geometric smoothing methods suffer from a number of problems. Perhaps the most important one is the *shrinkage* problem: when the smoothing method is applied iteratively a large number of times, a shape eventually collapses to a point.

Smoothing polygonal curves is simpler than smoothing polyhedral surfaces because curves have an intrinsic linear ordering which allows for the application of Fourier analysis. The so-called *Fourier descriptors*, the use of the coefficients in a Fourier series expansion of the tangent-angle versus arc-length description of a curve, provide a multi-resolution representation of continuous curves. To smooth a polygonal curve it is sufficient to truncate the Discrete Fourier Transforms of its coordinates, which are looked upon as discrete periodic signals. Fourier descriptors date back to the early 1960's [18], and have been widely used since then in the computer vision literature as multi-resolution shape descriptors for object recognition. The method of Fourier descriptors does not produce shrinkage, but nevertheless, it does have two important problems. First, it is computationally expensive. Even using the Fast Fourier Transform algorithm, the number of arithmetic operations is of the order of $n\log(n)$, where n is the number of vertices. Linear algorithms, those which require in the order of n arithmetic operations, are more desirable, particularly for surfaces, where the number of vertices is large. The second problem is that it does not extend to surfaces of arbitrary topological type.

Perhaps the most popular linear technique of geometric smoothing parameterized curves is the so-called *Gaussian smoothing* method, associated with scale-space theory [17]. In the continuous case, Gaussian smoothing is performed by convolving the vector function that parameterizes the curve with a Gaussian kernel. In section 2 we show how to define Gaussian smoothing on polyhedral surfaces of arbitrary topology. It is well known though, that Gaussian smoothing produces shrinkage. Some heuristic solutions to this problem have been proposed [6, 9], and more recently Oliensis [10] Oliensis pointed out that Gaussian smoothing produces shrinkage because the convolution with a Gaussian kernel is not a low-pass filter operation. Except for the zero frequency, all the frequencies are attenuated. To prevent shrinkage, the smoothing algorithm must produce a low-pass filter effect. However, Oliensis' solution, based on Fourier analysis and formulated for the continuous case, only extends to images, not to surfaces of arbitrary topological type.

Lindeberg [8] formulated a scale-space theory for closed polygonal curves. Within this framework Gaussian smoothing becomes a discrete convolution of the vertex coordinates with a discrete approximation of a Gaussian kernel. Lindeberg's smoothing method also produces shrinkage. The analysis techniques of section 4 generalize Lindeberg's analysis to general piece-wise linear shapes of arbitrary dimension and topology.

2 Gaussian Smoothing

In the Gaussian smoothing method the new position of each vertex is computed as a weighted average of the current positions of the vertex itself, and its first order neighbors, those vertices that share an edge (or face) with the current vertex. This process is repeated a number of times. The Gaussian smoothing method has a number of advantages with respect to the existing methods discussed above, but still produces shrinkage. The first advantage is that it applies to piece-wise linear surfaces of arbitrary topological type, not only those that can be parameterized by functions defined on a rectangular domain. The second advantage is that, since first order neighbors are defined implicitly in the list of edges or faces of the curve or surface, no storage is required to encode the neighborhood structures. The third advantage is that the number of operations is a linear function of the total number of vertices, edges, and faces. However, since the method is very local, to obtain a long range smoothing effect, the Gaussian smoothing methods has to be applied iteratively a large number of times, producing significant shrinkage as a by-product. Figures 1, 2, and 3 illustrate the problem of shrinkage that the Gaussian smoothing method shares with most existing geometric smoothing algorithms.

The main innovation of the method introduced in this article is how to modify the Gaussian smoothing method to prevent shrinkage, obtaining a simple and general method to smooth general and arbitrary polygonal curves and polyhedral surfaces that has all the good properties of the other methods described above, but none of their disadvantages.

In the rest of this section we first discuss how to represent piece-wise linear shapes as lists of vertices and edges, or vertices and faces. Then we define what we mean by a *neighborhood* of a vertex, and by a *neighborhood structure* of a shape. Finally, we describe the Gaussian smoothing algorithm in detail.

A closed polygon in two or three dimensions is usually represented as an ordered list of vertices $V = \{v_i : 1 \le i \le n_V\}$, with $v_i = (x_i, y_i)^t$ or $v_i = (x_i, y_i, z_i)^t$ depending on whether the curve is two or three dimensional. No more information is needed because curves have an intrinsic linear order. But for an open curve or a curve composed of several connected components, it is desirable to represent it as a pair of lists $C = \{V, E\}$, a list of vertices V and a list of edges $E = \{e_k : 1 \le k \le n_E\}$, with each edge $e_k = (i_1^k, i_2^k)$ being a pair of non-repeated indices of vertices. A surface is usually represented as a pair of lists $S = \{V, F\}$, a list of vertices $V = \{v_i : 1 \le i \le n_V\}$, and a list of faces $F = \{f_k : 1 \le k \le n_F\}$, with each face $f_k = (i_1^k, \ldots, i_{n_{f_k}}^k)$ being a sequence of non-repeated indices of vertices, and representing itself a closed three dimensional polygon, not necessarily flat. In some cases, the number of vertices n_{f_k} varies from face to face, while in others all the faces have the same number of vertices. *Triangulated surfaces* are the most common, where all the faces are triangles $f_k = (i_1^k, i_2^k, i_3^k)$. There are other popular representations for piece-wise linear shapes [2], but those just described are the most appropriate for our algorithm.

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. No vertex is allowed to be a neighbor of itself, but otherwise no further restrictions are imposed on the neighborhoods. In particular, it is permitted that a vertex v_j be a neighbor of vertex v_i without vertex v_i being a neighbor of vertex v_j. A neighborhood structure is *symmetric* if the situation just described never happens, i.e., every time that a vertex v_j is a neighbor of vertex v_i, also v_i is a neighbor of v_j. The *neighborhood structure* of a shape is defined as the family of all the neighborhoods $\{i^\star : i = 1, 2, \ldots, n_V\}$. A particularly important neighborhood structure is the *first order* neighborhood structure, where for each pair of vertices v_i and v_j that share an edge (or face), we make v_j a neighbor of v_i, and v_i a neighbor of v_j. For example, for a polygonal curve represented just by 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 represented in the list of vertices, edges, or faces of the shape, no extra storage is required to represent it.

In the Gaussian smoothing algorithm the position of each vertex is replaced by a convex combination of the positions of itself and its neighbors. Alternatively, Gaussian smoothing can also be reformulated as follows. First, for each vertex v_i, a *vector average*

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

is computed as a weighted average of the vectors $v_j - v_i$, that extend from the current vertex v_i to a neighbor vertex v_j. For each vertex v_i the weights w_{ij} are positive and add up to one, but otherwise they 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 $i^\star$. Once all the vector averages are computed, the vertices are updated by adding to each vertex current position v_i its corresponding *displacement vector*

$$v_i' = v_i + \lambda \Delta v_i$$

computed as the product of the vector average Δv_i and the *scale factor* λ, obtaining the new position v_i'. The scale factor, which can be a common value for all the vertices or be vertex dependent, is a positive number $0 < \lambda < 1$.

The advantage of the Gaussian smoothing method is that it produces geometric smoothing. The main disadvantage is that to produce significant smoothing, the Gaussian smoothing algorithm must be applied iteratively a large number of times using first order neighborhoods. However, by doing so a significant shrinkage effect is also introduced.

3 The New Algorithm

The smoothing algorithm introduced in this article consists of two consecutive Gaussian smoothing steps. After a first Gaussian smoothing step with a positive scale factor λ is applied to all the vertices of the shape, a second Gaussian smoothing step is applied to all the vertices, but with a negative scale factor μ, greater in magnitude than the first scale factor ($0 < \lambda < -\mu$). To produce a significant smoothing effect, these two steps must be repeated, alternating the positive and negative scale factors, a number of times.

In fact this method produces a *low pass filter* effect, where curve or surface *curvature* takes the place of frequency. The original non-smooth curve or surface is modeled as an underlying smooth curve or surface, plus a normal perturbation vector field. The underlying curve or surface is bounded above in curvature, and the perturbation that needs to be filtered out is regarded as zero mean high curvature noise. The two scale factors determine the pass-band and stop-band curvatures. For higher attenuation in the stop-band, the two Gaussian smoothing steps must be repeated alternating the two scale factors λ and μ. The amount of attenuation is then determined by the number of iterations N.

Figures 4, 5, and 6 show more examples of applying this new smoothing algorithm to large piece-wise surfaces extracted from volumetric data.

4 Why the New Algorithm Works

In this section we show why the algorithm described in the previous section produces a low-pass filter effect. In the next section we show how to design the low pass-filter, i.e., how to determine the two scale factors λ and μ, and the number of iterations N, as functions of the low-pass filter parameters.

For a two-dimensional shape, let X be the $n_V \times 2$ matrix with i-th. row equal to the coordinates (x_i, y_i) of the current position of vertex v_i. For a three-dimensional shape, let X be the $n_V \times 3$ matrix with i-th. row equal to the coordinates (x_i, y_i, z_i) of the current position of vertex v_i. Let X' and X'' be matrices constructed in the same way, but with the coordinates of the first and second positions of the vertices, respectively. And let X^N be yet another matrix constructed in the same way, but with the coordinates of the vertices when the algorithm stops, after N iterations. The relation between the matrices X and X' can be described in matrix form as

$$X' = (I - \lambda K) X \ ,$$

where λ is the first scale factor, K is the square $n_V \times n_V$ matrix $K = I - W$, I is the $n_V \times n_V$ identity matrix, and W is the square $n_V \times n_V$ matrix of weights $\{w_{ij} : i, j = 1, 2, \ldots, n_V\}$, with the convention that weight w_{ij} is equal to 0 if vertex v_j is not a neighbor of vertex v_i. Similarly, the relation between the matrices X' and X''

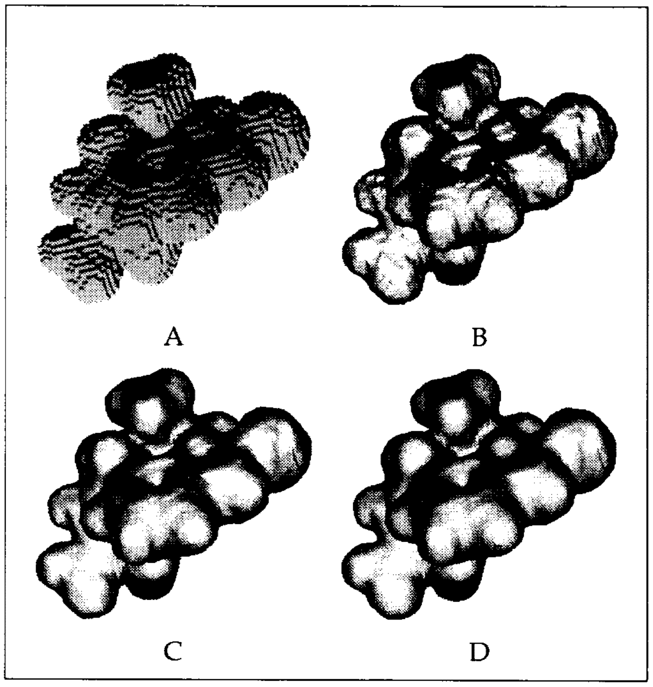

Figure 4: (A) A very irregular polyhedral surface computed as the boundary of a set of volume elements. The result of applying 30 steps (B), 60 steps (C), and 90 steps (D) of the smoothing method of this article to A with parameters $\lambda = 0.33$ and $\mu = -0.34$ (see text for explanation of parameters).

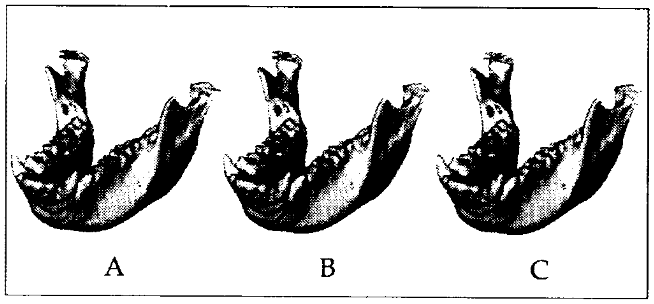

Figure 5: Smoothing piece-wise linear surfaces created by a good iso-surface construction algorithm. (A) Iso-surface constructed with Kalvin's Alligator algorithm [7] from CT data (74,760 vertices, 149,776 faces). The result of applying 30 steps (B), and 100 steps (C), of the smoothing method of this article to A with parameters $\lambda = 0.33$ and $\mu = -0.34$ (see text for explanation of parameters). Note the faceting in A near the teeth produced by the greater separation between slices, has disappeared in C

can be described in matrix form as

$$X'' = (I - \mu K)X \ ,$$

where μ is the second scale factor, and K is the same

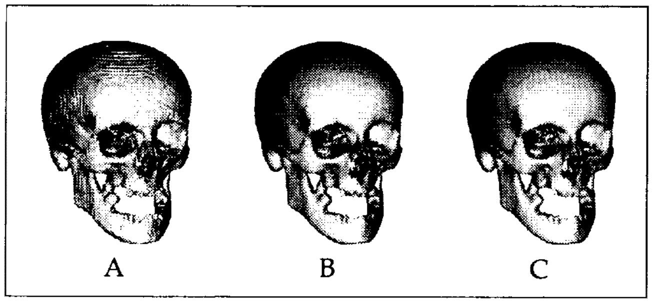

Figure 6: Smoothing simplified piece-wise linear surfaces. (A) Simplified iso-surface constructed with Guéziec's Wrapper algorithm [5, 4] from CT data (27,367 vertices, 55,310 faces). The result of applying 30 steps (B), and 60 steps (C), of the smoothing method of this article to A with parameters $\lambda = 0.330$ and $\mu = -0.331$ (see text for explanation of parameters).

matrix described above. Finally, since the matrices $I - \lambda K$ and $I - \mu K$ commute with each other, the relation between the position of the vertices before and after N iterations can be expressed in matrix form as

$$X^N = ((I - \mu K)(I - \lambda K))^N X \ .$$

Although the method applies to general neighborhood structures, we will restrict our analysis here to 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 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 $1/|i^\star|$ (the incidence matrix of the neighborhood). The study of shapes with more general neighborhood structures is beyond the scope of this paper, and will be carried out in detail in a forthcoming article.

Since the matrix $D^{1/2}WD^{-1/2} = D^{1/2}ED^{1/2}$ is symmetric, W is a so-called *normal matrix* [3], which has all real eigenvalues, and sets of n_V left and right eigenvectors that form respective bases of n_V-dimensional space. Furthermore, by construction, W is also a *stochastic matrix*, a matrix with nonnegative elements and rows that add up to one [12]. 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 (but typically the greatest eigenvalue of K is significantly smaller than 2). Let $0 \leq k_1 \leq k_2 \leq \cdots \leq k_{n_V} \leq 2$ be the eigenvalues of the matrix K, and let $f(k)$ be the polynomial of one variable k defined by

$$f(k) = (1 - \lambda k)(1 - \mu k) \ . \qquad (1)$$

Polynomials of one variable can be evaluated in square matrices. In particular the matrix $((I - \mu K)(I - \lambda K))^N$ can be written as the evaluation $f(K)^N$ of the polynomial $f(k)^N$ in the matrix K. If $u_1, u_2, \ldots, u_{n_V}$ is a set of linearly independent unit length right eigenvectors of the matrix K associated with the eigenvalues $k_1, k_2, \ldots, k_{n_V}$, respectively, then $u_1, u_2, \ldots, u_{n_V}$ are also right eigenvectors of the matrix $f(K)^N$, with associated eigenvalues $f(k_1)^N, \ldots, f(k_{n_V})^N$. Furthermore, since $u_1, u_2, \ldots, u_{n_V}$ constitute a basis of n_V-dimensional space, each column vector x of the matrix X (the vectors of first, second, or third coordinates of the vertices), and in fact any n_V-dimensional vector x, can be written in a unique way as a linear combination of the basis vectors

$$x = \xi_1 u_1 + \cdots + \xi_{n_V} u_{n_V} , \tag{2}$$

where $\xi_1, \ldots, \xi_{n_V}$ are constants. Thus, after applying the smoothing algorithm, we obtain

$$f(K)^N x = (\xi_1 f(k_1)^N)\, u_1 + \cdots + (\xi_{n_V} f(k_n)^N)\, u_{n_V} . \tag{3}$$

Seen as scalar functions defined on the vertices of the shape, the eigenvectors of the matrix K can be considered as the main *free vibration modes*, and the corresponding eigenvalues as the associated *natural frequencies*. There is some similarity here with the modal analysis of Pentland and Sclaroff [11], in the sense that eigenvectors of certain matrices are seen as vibrations modes and the corresponding eigenvalues as natural frequencies, but the similarity ends up here. The matrices are constructed in very different ways. This is also a generalization of traditional Discrete Fourier analysis. For example, if the shape is a closed polygon, where each vertex has exactly two neighbors, the matrix K turns out to be a *circulant matrix*, whose eigenvectors are the columns of the Fourier matrix [1]. Computing the coefficients $\xi_1, \ldots, \xi_{n_V}$ of equation (2) is nothing but computing the Discrete Fourier Transform of x.

We only have to make sure now that in equation (3), only the terms associated with low frequencies remain after N iterations. That is, $f(k_i)^N$ must be very close to 1 for a low frequency k_i and large N, and $f(k_i)^N$ must be close to 0 for a high frequency k_i and large N.

Figure 7 is a diagram that shows the graph of the polynomial $f(k)$, and Figure 8 is a diagram that shows the *transfer function* of the algorithm, the graph of the function $f(k)^N$. The graph of the polynomial $f(k)$ is an inverted parabola with roots at $k = 1/\lambda > 0$ and $k = 1/\mu < 0$. The value of the polynomial $f(k)$ is positive for $1/\mu < k < 1/\lambda$, and negative for $k < 1/\mu$ and $k > 1/\lambda$. Furthermore, 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

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

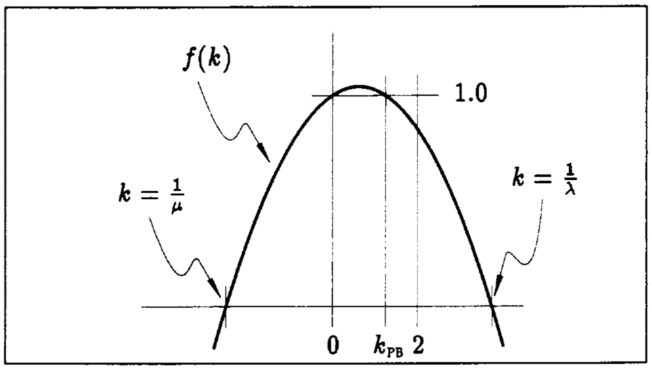

Figure 7: Graph of the polynomial $f(k) = (1 - \lambda k)(1 - \mu k)$.

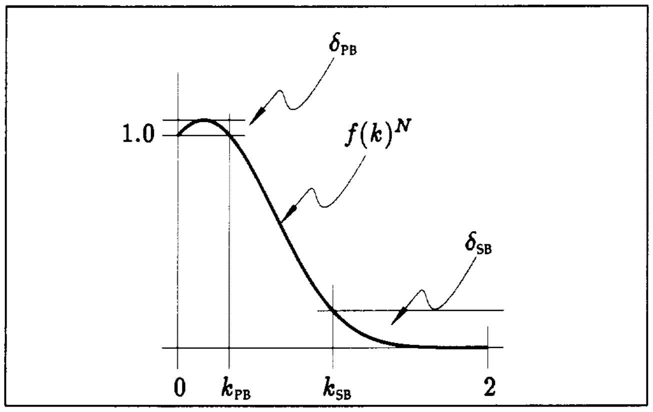

Figure 8: Graph of the transfer function $f(k)^N$.

The graph of the transfer function $f(k)^N$ displays a typical *low-pass filter* shape in the region of interest, i.e., from $k = 0$ to $k = 2$. The *pass-band region* extends from $k = 0$ to $k = k_{PB}$. For values of k in the pass-band region, $f(k)^N \approx 1$. The *transition region* extends from $k = k_{PB}$ to $k = k_{SB}$ (the *stop-band frequency*), where k_{SB} is a parameter defined by the user, that will be discussed in detail in the next section. The *stop-band region* extends from $k = k_{SB}$ to $k = 1/\lambda$. For values of k in the stop-band region, $f(k)^N \approx 0$ for large N.

5 Filter design

The low-pass filter parameters are the pass-band frequency k_{PB}, the pass-band ripple δ_{PB}, the stop-band frequency k_{SB}, and the stop-band ripple δ_{SB}, shown in Figure 8. To design the low-pass filter we have to show how to determine the first scale factor λ, the second scale factor μ, and the number of iterations N as functions of the low-pass filter parameters.

We can observe in Figure 8, that the low-pass filter parameters must satisfy the following inequalities

$$0 < k_{PB} < k_{SB} < 2 \; , \; 0 < \delta_{PB} \;\; 0 < \delta_{SB} < 1 \; . \tag{4}$$

From the discussion above, we also know that the first scale factor λ, the second scale factor μ, and the number of iterations N, must satisfy the following equations and inequalities

$$0 < N \ , \ 0 < \lambda < -\mu \ , \ \lambda < \frac{1}{k_{\mathrm{SB}}} \ , \ \frac{1}{\lambda} + \frac{1}{\mu} = k_{\mathrm{PB}} \ . \qquad (5)$$

There are two more inequalities that must be satisfied to design the low-pass filter:

$$((\lambda-\mu)^2)/(-4\lambda\mu))^N \ < \ 1+\delta_{\mathrm{PB}} \qquad (6)$$

$$((1-\lambda\, k_{\mathrm{SB}})(1-\mu\, k_{\mathrm{SB}}))^N \ < \ \delta_{\mathrm{SB}} \ . \qquad (7)$$

The values of the first scale factor λ, the second scale factor μ, and the number of iterations N, are computed as a solution of the system of equations and inequalities (5)-(6)-(7).

Since there are three free variables (λ, μ, and N), and four parameters (k_{PB}, k_{SB}, δ_{PB}, and δ_{SB}), the system might have no solution. That is the case if we push the stop-band frequency very close to the pass-band frequency, or if we set the two ripples very tight. But the system might have more than one solution, in which case we must choose the one corresponding to the minimum number of iterations N to minimize computation time.

6 Conclusion

In this paper we have introduced a method for smoothing piece-wise linear shapes of arbitrary dimension and topology, that generalizes Gaussian smoothing but prevents shrinkage. We have shown that the method produces a low-pass filter effect as a function of the natural frequencies of the shape. The method is efficient both in terms of computational complexity, and in terms of storage requirements. The complexity is linear in the number of edges or faces of the shape, and the storage required is a linear function of the vertices. The method has immediate application to improve a number of computational procedures, including visualization of scientific data and registration of multimodal medical data.

A longer version of this paper [13], as well as other related papers [15, 16], can be retrieved as PostScript files from the IBM Research World Wide Web server (`http://www.watson.ibm.com:8080`).

7 Acknowledgments

Thanks to David Dean for providing CT data sets. Thanks to André Thanks to Alan Kalvin for creating the surface for figure 5. Guéziec for creating the surface for figure 6. And thanks to Louis Percello for helping in writing the patent application [14].

References

[1] P.J. Davis. *Circulant Matrices*. John Wiley & Sons, 1975.

[2] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, Reading, MA, second edition, 1992.

[3] G. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1983.

[4] A. Guéziec and D. Dean. The wrapper algorithm: A surface optimization algorithm that preserves highly curved areas. In *Second Conference on Visualization in Biomedical Computing*, Rochester, Minnesota, October 4-7 1994. SPIE.

[5] 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.

[6] B.K.P. Horn and E.J. Weldon Jr. Filtering closed curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):665–668, September 1986.

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

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

[9] D.G. Lowe. Organization of smooth image curves at multiple scales. *International Journal of Computer Vision*, 3:119–130, 1989.

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

[11] A. Pentland, S. Sclaroff, B. Horowitz, and I. Essa. *Three-Dimensional Object Recognition Systems*, chapter Modal Descriptions for Modeling, Recognition, and Tracking, pages 423–445. Elsevier, 1993.

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

[13] G. Taubin. Curve and surface smoothing without shrinkage. Technical Report RC-19536, IBM Research, April 1994. (also in Proceedings ICCV'95).

[14] G. Taubin. Curve and surface smoothing without shrinkage. U.S. Patent (pending), September 1994.

[15] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. Technical Report RC-19860, IBM Research, December 1994. (also in Proceedings ICCV'95).

[16] G. Taubin. A signal processing approach to fair surface design. Technical Report RC-19923, IBM Research, February 1995. (also in Proceedings SIGGRAPH'95).

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

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