Notes on Mesh Smoothing

Nicholas Bray

October 18, 2004

Motivation

There are a number of applications in which increasing the smoothness of a mesh is desirable. A compelling example is noise removal. When meshes are constructed from acquired data, measurement error can result in a smooth surface being approximated with a rough mesh. Smoothing the mesh can attenuate this noise. Another application is surface subdivision. Taubin [1] cites a class of algorithms that smooth the surface after subdividing it to improve the appearance. In general, mesh processing algorithms may introduce undesirable artifacts that can be attenuated with smoothing.

Laplacian Smoothing

A common method for smoothing meshes is Laplacian smoothing. This process is also known as diffusion. The basic idea is that the vertices of a mesh are incrementally moved in the direction of the Laplacian. The differential version of the smoothing equation is:

$$\frac{\partial X}{\partial t} = \lambda L(X) \tag{1}$$

where X is the vertices of the mesh, L is the Laplacian, and λ is a scalar that controls the diffusion speed. Assuming the Laplacian operator is linear, the smoothing equation can be rewritten as the following forward difference equation:

$$X(n+1) = (I + \lambda dt L)X(n) \tag{2}$$

Diffusion smoothing has a couple of desirable properties. First, it doesn't modify the connectivity of a mesh. Each smoothing step modifies the position of the vertices, but no vertices are added or removed, and the triangulation remains unchanged. Second, the discrete Laplacian can be formulated so that smoothing a given vertex only requires information about its immediate neighbors. The size of this neighborhood tends to be small and does not increase as the size of the mesh increases. As such, the complexity of Laplacian smoothing can be linear in time and space. Taubin [1] cites a number of optimization-based smoothing algorithms, and notes their higher-order complexities make them impractical to use on large meshes.

Diffusion smoothing also has a few undesirable properties. Unless the approximation of the Laplacian is constructed carefully, triangles will slide across the mesh. This sliding pushes the mesh towards a more regular triangulation. This effect may in of itself be desirable, but surface properties, such as color and texture coordinates, may need to be reparameterized. Further, sliding tends to distort the shape of the mesh. Shape distortion, for most applications, is extremely undesirable. Figure 1 and 2 demonstrate shape distortion resulting from Laplacian smoothing. Finally, meshes subjected to Laplacian smoothing shrink. In the asymptotic limit, applying Laplacian smoothing to a mesh will cause it to collapse to its barycenter. Volume loss will not be this extreme for reasonable amounts of smoothing, but it can still be quite noticeable.

Laplacian Approximations

There are a number of different approximations for the Laplacian operator, each with their own advantages and disadvantages. In the continuous case, the Laplacian is defined as:

$$\Delta f = \nabla^2 f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{3}$$

Desbrun et al. [3] describe three discrete approximations, each of which boil down to a weighted sum of the one-ring neighbors of a vertex:

$$L(x_i) = \sum_{j \in N_1(i)} w_{ij}(x_j - x_i) \tag{4}$$

The only difference between the operators is how they calculate the weights.

Simple Laplacian

A simple approximation of the Laplacian uses equal weights:

$$w_{ij} = \frac{1}{m} \tag{5}$$

where m is the number of neighbors of x_i. In effect, this scheme defines the Laplacian as the vector from the vertex in question to the barycenter of its one-ring neighbors. This is also known as the umbrella operator.

The biggest advantage of using the simple Laplacian operator is that it depends only on the topology of the mesh, and not the position of the vertices. As such, it is a linear operator, and matches the assumptions made in the formulation of the smoothing equation. Further, it does not change during the smoothing process and therefore never needs to be recalculated.

The biggest problem with the umbrella operator is that it results in significant sliding and shape distortion when applied to meshes with irregular triangulations

Scale-dependent Laplacian

A slightly more complex approximation of the Laplacian uses weights proportional to the inverse distance between the vertices. These weights are also known as Fujiwara weights. This formulation of the Laplacian is also known as the scale-dependent umbrella operator.

$$w_{ij} = \frac{1}{|e_{ij}|} \tag{6}$$

Like the previous approximation, vertex sliding is still an issue, although it is less pronounced. The primary benefit of this operator is that it better preserves the distribution of triangle sizes.

On the downside, the operator is no longer linear. Further, the operator must be recalculated as the vertices move. For reasonable amounts of smoothing, however, edge lengths should not change too radically. As such, it is safe to approximate the scale-dependent Laplacian operator as constant over short time intervals. Desbrun et al. [3] claim good results without recalculating the operator. Scale dependence puts an additional stability constraint on explicit methods, and can force small time steps for large meshes with fine details.

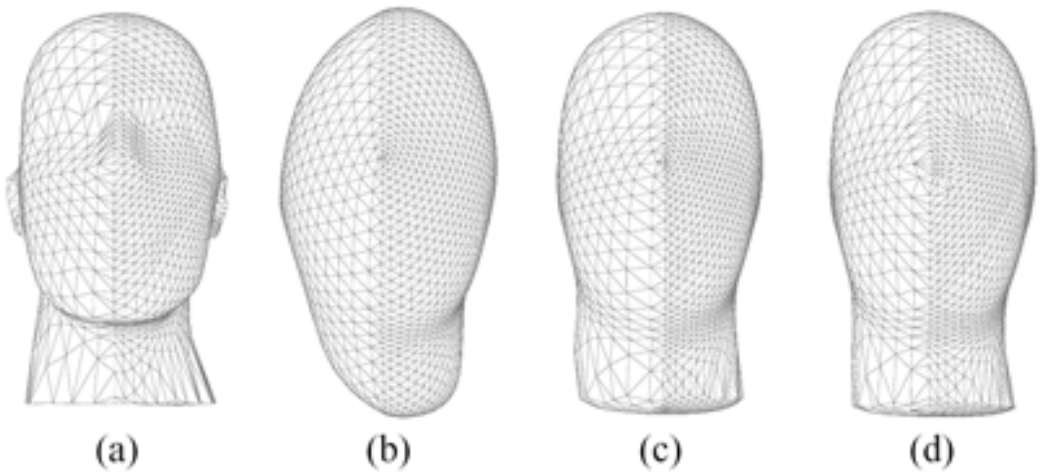

Figure 1: *Application of operators to a mesh with different sampling rates: (a) original mesh, (b) equal weights, (c) Fujiwara weights, and (d) curvature normal. Figure taken from [3].*

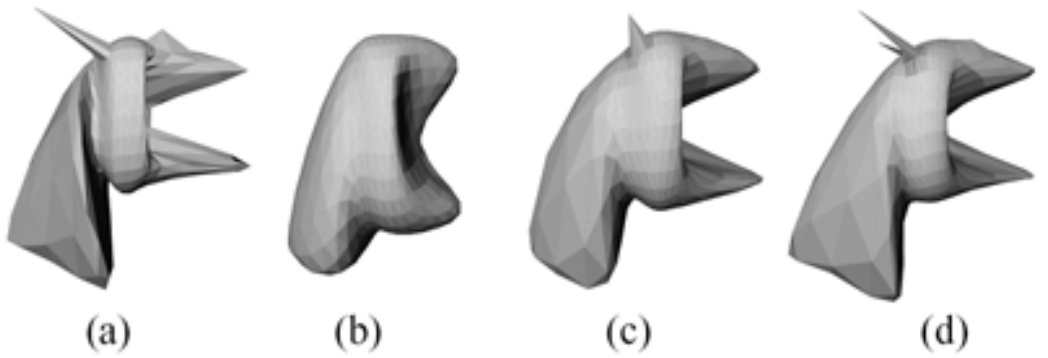

Figure 2: *Significant smoothing of a dragon: (a) original mesh, (b) equal weights, (c) Fujiwara weights, and (d) curvature normal. Figure taken from [3].*

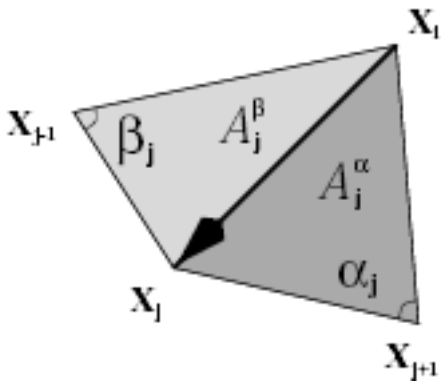

Figure 3: *A diagram of the angles used in equation 7. Figure taken from [3].*

Curvature Normal

To solve the issue of sliding, Desbrun et al. [3] propose a weighting scheme that eliminates the tangential component of the Laplacian:

$$w_{ij} = \cot \alpha_j + \cot \beta_j \tag{7}$$

where α_j and β_j are the angles opposite the edge in the two triangles that share that edge e_{ij}, as shown in Figure 3.

According to Desbrun et al. [3], curvature flow provides the best smoothing with respect to shape. Like the scale-dependent Laplacian, however, the curvature operator is non-linear and changes as the vertices move. This weighting scheme unfortunately breaks down if either α_j or β_j is obtuse, as w_{ij} can become negative. A preprocessing pass can be added to split obtuse triangles and sidestep this issue.

Links to Signal Processing

Taubin [1] notes that Laplacian smoothing is equivalent to lowpass filtering. In traditional signal processing, lowpass filtering is a simple way to remove noise if the desired signal does not have significant high frequency content. As in traditional signal processing, it is possible to tune the frequency response of the filter by constructing a polynomial sum of the Laplacian operator. Higher order polynomials result in a sharper frequency rolloff, and this allows the filter to distinguish between signal and noise with greater precision.

Taubin further points out that projecting a signal onto the basis formed by the eigenvectors of the Laplacian operator is equivalent to the Fourier transform. This opens the door for smoothing a mesh using spectral decomposition and resynthesis. Unfortunately, the time complexity of this approach is usually worse than linear. The complexity for finding all the eigenvalues of a matrix is O(n^3), but sparseness can be exploited to reduce the complexity to O(n). Even if an efficient method is employed, however, numerical instability can become a problem when adjacent eigenvalues are closely spaced.

Variations on Basic Smoothing

Beyond the choice of how to approximate the Laplacian operator, there a number of variations on basic diffusion smoothing that can be useful.

Implicit Integration

Desburn et al. [3] observe that implicit integration can be faster than explicit for smoothing applications. The sparseness of the Laplacian operator allows each iteration to be reasonably efficient, and the stability gained from using an implicit approach enables the smoothing to be done in fewer iterations. The implicit smoothing equation is:

$$(I - \lambda dt L)X(n+1) = X(n) \tag{8}$$

As mentioned previously, the quality of smoothing can be improved by using a polynomial sum of Laplacian operator. Unfortunately, this has the side effect of decreasing the sparseness of the smoothing operator. This is a particular problem for implicit formulations, as the loss of sparcity pushes the computational complexity towards that of solving an N x N system of equations. Desburn et al. [3] claim that L^2 results in the best balance between quality and computational cost when implicit methods are employed.

Taubin Smoothing

Taubin smoothing, also known as the $\lambda|\mu$ algorithm, relies on the use of two diffusion steps, one inwards and one outwards, to approximately preserve the volume of the mesh. As with previous smoothing algorithms, λ controls the amount of

inward diffusion and the new parameter μ plays the same role in the outward diffusion step. In signal processing terms, Taubin smoothing employs a second-order filter with a pass band gain slightly larger than one so as to prevent the shrinkage of low frequency components. In geometric terms, Taubin smoothing diffuses the mesh inwards and outwards to attenuate details while keeping the surface in roughly the same position. Although this approach is not guaranteed to preserve mesh volume, it does a good job if the parameters λ and μ are well chosen. On the down side, Taubin smoothing requires more iterations to achieve a level of smoothing comparable to other methods.

Volume Rescaling

Desbrun et al. [3] propose an alternate approach for preserving the volume of a smoothed mesh. Calculating the volume of a mesh has a linear complexity, so recording the volume of a mesh before and after a smoothing operation is fairly inexpensive. Given these two values, it is possible to scale the smoothed mesh to make its volume match that of the original. This approach requires less computation than Taubin smoothing, and does not require the tuning of an additional parameter. On the down side, volume rescaling does not produce as good of results as the $\lambda|\mu$ algorithm for significant amounts of smoothing.

Anisotropic Smoothing

A problem with smoothing approaches that have been discussed thus far is that they remove sharp features as readily as they remove noise. This is not an algorithmic issue; rather it is a problem with how smoothness has been defined. By definition the curvature at an edge is discontinuous, so any attempt to reduce the variation in curvature will adversely affect the edge. From a signal processing perspective, edges contain high frequency components, so lowpass filtering will round them. Hildebrandt et al. [6] propose an anisotropic smoothing scheme that reduces diffusion across edges. To achieve this, they modify the curvature normal operator by reducing the weight of terms that have a magnitude greater than a threshold.

Other Smoothing Techniques

Not all approaches to smoothing fit within the framework of Laplacian smoothing. Jones et al. [5] propose a statistical method to anisotropically smooth a mesh in one pass. This approach predicts the location of a vertex based on its neighbors. Robust statistics are used to deemphasize the contribution of vertices dissimilar to the one being predicted. Fleishman et al. [4] propose a similar method based on bilateral filtering that is iterative. Bilateral filtering was originally formulated for image processing, and is a non-linear variation of Gaussian smoothing that weights sample points based on their similarity to the one being processed.

Summary

When implementing diffusion smoothing, there are a number of choices that can be made. Different Laplacian approximations can be chosen to balance computational cost with quality. Explicit integration can be used for the sake of simplicity, or implicit integration can be used to improve performance. If volume preservation is desired, Taubin smoothing or volume rescaling can be employed. All of these choices have associated tradeoffs, so there is no best approach to mesh smoothing. Ultimately, the needs of the application determine the best approach to use.

References

[1] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 1995*.

[2] G. Taubin. Geometric signal processing on polygonal meshes. Eurographics 2000 State of the Art Report, August 2000.

[3] M. Desbrun, M. Meyer, P. Schrder, and A. Barr. Implicit fairing of arbitrary meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH 1999*, pp. 317324, 1999.

[4] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. In *Proceedings of SIGGRAPH 2003*, pp. 950953, 2003.

[5] T. R. Jones, F. Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. In *Proceedings of SIGGRAPH 2003*, pp. 943949, 2003.

[6] K. Hildebrandt and K. Polthier. Anisotropic filtering of non-linear surface features. In *Proceedings of Eurographics 2004*.