Multiresolution Signal Processing for Meshes

Igor Guskov
Princeton University

Wim Sweldens
Bell Laboratories

Peter Schröder
Caltech

Figure 1: *Mount Meshmore (Design Khrysaundt Koenig).*

Abstract

We generalize basic signal processing tools such as downsampling, upsampling, and filters to irregular connectivity triangle meshes. This is accomplished through the design of a non-uniform relaxation procedure whose weights depend on the geometry and we show its superiority over existing schemes whose weights depend only on connectivity. This is combined with known mesh simplification methods to build subdivision and pyramid algorithms. We demonstrate the power of these algorithms through a number of application examples including smoothing, enhancement, editing, and texture mapping.

CR Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling - *hierarchy and geometric transformations, object hierarchies*; I.4.3 [**Image Processing and Computer Vision**]: Enhancement - *filtering, geometric correction, sharpening and deblurring, smoothing*; G.1.2 [**Numerical Analysis**]: Approximation - *approximation of surfaces and contours, wavelets and fractals*

Additional Keywords: Meshes, subdivision, irregular connectivity, surface parameterization, multiresolution, wavelets, Laplacian Pyramid.

1 Introduction

3D range sensing is capable of producing detailed and densely sampled triangular meshes of high quality. Increasing deployment of this technology in the automotive and entertainment industries, as well as many other areas, has fueled the need for algorithms to process such datasets. Examples include editing, simplification, denoising, compression, and finite element simulation.

In the case of regularly sampled data, for example images, basic signal processing tools such as filtering, subsampling, and upsampling exist. These can be used to build subdivision and pyramid algorithms, which are useful in many applications. Our goal is the construction of signal processing style analyses and algorithms for triangle meshes.

Building the elements of a signal processing toolbox for meshes is not immediately straightforward since there are essential differences between images, for example, and meshes. Images are functions defined on Euclidean ("flat") geometry and are almost always sampled on a regular grid. Consequently, algorithms such as subsampling and upsampling are straightforward to define, and uniform filtering methods are appropriate. This makes Fourier analysis an elegant and efficient tool for the construction and analysis of signal processing algorithms.

In contrast, triangle meshes of arbitrary connectivity form an inherently irregular sampling setting. Additionally we are dealing with general 2-manifolds as opposed to a Euclidean space. Consequently new algorithms need to be developed which incorporate the fundamental differences between images and meshes.

A crucial first observation concerns the difference between geometric and parametric smoothness. *Geometric* smoothness measures how much triangle normals vary over the mesh. Geometric

Terminology

In order to describe our contribution and its relationship to existing work we need to set some terminology. Among triangulations we distinguish three types

- **Regular:** every vertex has degree six;
- **Irregular:** vertices can have any degree;
- **Semi-regular:** formed by starting with a coarse irregular triangulation and performing repeated quadrisection on all triangles. Coarse vertices have arbitrary degree while all other vertices have degree six.

In all cases we assume that any triangulation is a proper 2-manifold with boundary. On the boundary regular vertices have degree four. Each of these triangulations call for different filtering and subdivision algorithms:

- **Uniform:** fixed coefficient stencils everywhere; typically used only on regular triangulations;
- **Non-uniform:** filter coefficients depend on the connectivity *and* geometry of the triangulation;
- **Semi-uniform:** coefficients of filters depend only on the (local) connectivity of the triangulation; typically used on semi-regular triangulations.

Using our terminology, for example, traditional subdivision [22] uses semi-uniform filters on semi-regular meshes.

smoothness implies that there exists *some* smooth (differentiable) parameterization of the mesh. However, any particular parameterization may well be non-smooth. The smoothness of the parameterizations is important in most numerical algorithms, which work only with the coordinate functions the user provides. The algorithms' behavior, such as convergence rates or the quality of the results, generally depends strongly on the smoothness of the coordinate functions.

In the regular setting of an image, or the knots of a uniform tensor product spline, we may simply use a uniform parameterization and will get parametric smoothness wherever there is geometric smoothness. In the irregular triangle mesh setting there is a priori no such "obvious" parameterization. In this case using a uniformity assumption leads to parametric non-smoothness with undesirable consequences for further processing. One approach to remedy this situation is the use of remeshing [8, 19], which maintains the original geometric smoothness, but improves the sampling to vary smoothly. This enables subsequent treatment with a uniform parameter assumption without detrimental effects. Here we wish to build tools which work on the original meshes directly.

To understand the role of the parameterization further, consider traditional subdivision [22], such as Loop or Catmull-Clark. In the signal processing context, subdivision can be seen as upsampling followed by filtering. One starts with an arbitrary connectivity mesh and uses regular upsampling techniques such as triangle quadrisection to obtain a semi-regular triangulation. The subdivision weights depend only on connectivity, not geometry. Such stencils can be designed with existing Fourier or spectral techniques. These schemes result in geometrically smooth limit surfaces with smooth semi-uniform parameterizations. Because traditional subdivision is only concerned with *refinement* one has the freedom to choose regular upsampling, and semi-uniform schemes suffice.

The picture changes entirely if we wish to compute a mesh pyramid, i.e., we want to be able to *coarsify* a given fine irregular mesh and later *refine* it again. We then need to filter, downsample, upsample and filter again. The downsampling typically involves a standard mesh simplification hierarchy. When subdividing back, we want to build a mesh with the *same* connectivity as the original mesh and a smooth geometry. This time the upsampling procedure is determined by reversing the previously computed simplification hierarchy. We no longer have a choice as in the classical subdivision setting. Consequently the filters used before downsampling and after upsampling should use non-uniform weights, which depend on the local parameterization. The challenge is to ensure that these local parameterizations are smooth so that subsequent algorithms act on the geometry and not some potentially bad parameterization.

1.1 Contributions

In this paper we present a series of non-uniform signal processing algorithms designed for irregular triangulations and show their usefulness in several application areas. Specifically, we make the following contributions:

- We show how the non-uniform subdivision algorithm of Guskov [12] can be used for geometric smoothing of triangle meshes. Our scheme is fast, local, and straightforward to implement.
- We use the smoothing algorithm combined with existing hierarchy methods to build subdivision, pyramid, and wavelet algorithms for irregular connectivity meshes.
- We show how these signal processing algorithms can be used in applications such as smoothing, enhancement, editing, animation, and texture mapping.

1.2 Related Work

In our approach we draw on observations made by researchers in several different areas. These include classical subdivision [22], which we generalize to the irregular setting with the help of mesh simplification [13] and careful attention to the role of smooth parameterizations. Parameterizations were examined in the context of remeshing [19, 8, 9], texture mapping (e.g., [20]), and variational modeling [16, 28, 21]. One area which employs these elements is hierarchical editing for semi-regular [29] and irregular meshes [18].

Signal processing as an approach to surface fairing in the irregular setting was first considered by Taubin [26, 27]. He defines frequencies as the eigenvectors of a discrete Laplacian generalized to irregular triangulations. The resulting smoothing schemes were used to denoise meshes, apply smooth deformations, and build semi-uniform subdivision over irregular meshes. Our approach is related to Taubin's and can be seen as a generalization to the non-uniform setting. In particular we build a smoothing method by minimizing multivariate finite differences. Together with progressive mesh simplification [14] we use these to define a non-uniform subdivision scheme and pyramid algorithm on top of an irregular mesh hierarchy.

Progressive meshes and a semi-uniform discrete Laplacian were used by Kobbelt et al. [18] to perform multiresolution editing on irregular meshes. Given some region of the mesh, discrete fairing is used to compute a smoothed version with the same connectivity. This smoothed region is deformed and offsets to the original mesh are added back in. Kobbelt discusses the issue of geometric vs. parametric smoothing. Smoothing of irregular meshes based on uniform approximations of the Laplacian results in vertex motion "within" the surface, even in a perfectly planar triangulation. It is geometrically smooth, yet the parameter functions appear non-smooth due to a non-uniform parameterization. This has undesirable effects in a hierarchical setting in which fine levels are defined as offsets ("details") from a coarse level: using the difference between topologically corresponding vertices in the original and smoothed mesh can lead to large detail vectors [18, Figure 4]. To minimize the size of detail vectors they employed a search procedure to find the nearest vertex on the smoothed mesh to a given vertex on the original mesh. This diminishes the advantage of having a smoothed version with the exact same connectivity. In contrast, our non-uniform smoothing scheme affects only geometric smoothness and so does not need a search procedure. We will present two ways

in which our scheme can be used for editing: one is based on multiresolution and combines the work of Kobbelt et al. [18] with the ideas of Zorin et al. [29]. The other method relies on defining vector displacement fields with controllable decay similar to the ideas presented in the work of Singh and Fiume [23].

We construct our subdivision scheme by designing a non-uniform relaxation operator which minimizes second differences. This is motivated by the smoothness analysis of the 1D irregular setting [2]. This analysis relies on the decay of divided differences, carefully designed to respect the underlying parameterization. These ideas were extended to the multi-variate setting in [12] and we employ them here. While the schemes we present have many nice properties and work very well in practice, we note that their analytic smoothness is currently unknown.

2 Signal Processing Algorithms

Before describing the actual numerical algorithms we begin with some remarks regarding different settings and establish our notation for triangulations and difference operators defined on them.

Coordinate Functions To describe our algorithms we must distinguish between two settings: the functional and the surface setting. The *functional setting* deals with a function $g(u, v)$ of two independent variables in the plane. The dependent variable g can be visualized as height above the (u, v) parameter plane. In practice we only have discrete data $g_i = g(u_i, v_i)$. The sample points (u_i, v_i) are triangulated in the plane and this connectivity can be transferred to the corresponding points (u_i, v_i, g_i) in $\mathbf{R}^3$. The canonical example of this is a terrain model.

The *surface setting* deals with a triangle mesh of arbitrary topology and connectivity embedded in $\mathbf{R}^3$ with vertices $p_i = (x_i, y_i, z_i)$. It is important to treat all three coordinates x, y, and z as *dependent* variables with independent parameters u and v, giving us three functional settings. The independent parameters are typically unknown and must be estimated. Algorithms to estimate *global* smooth parameterizations are described in [19, 8, 9, 20]. We require only *local* parameterizations which are consistent over the support of a small filter stencil.

Triangulations To talk about local neighborhoods of vertices within the mesh it is convenient to describe the topological and geometric aspects of a mesh separately. We use notation inspired by [24]. A triangle mesh is denoted as a pair $(\mathcal{P}, \mathcal{K})$, where $\mathcal{P}$ is a set of N point positions $\mathcal{P} = \{p_i \in \mathbf{R}^3 \mid 1 \le i \le N\}$ (either $p_i = (u_i, v_i, f_i)$ in a functional setting or $p_i = (x_i, y_i, z_i)$ in the surface setting), and $\mathcal{K}$ is an *abstract simplicial complex* which contains all the topological, i.e., adjacency information. The complex $\mathcal{K}$ is a set of subsets of $\{1, \ldots, N\}$. These subsets are called simplices and come in three types: vertices $v = \{i\} \in \mathcal{V}$, edges $e = \{i, j\} \in \mathcal{E}$, and faces $f = \{i, j, k\} \in \mathcal{F}$, so that $\mathcal{K} = \mathcal{V} \cup \mathcal{E} \cup \mathcal{F}$. Two vertices i and j are *neighbors* if $\{i, j\} \in \mathcal{E}$. The 1-ring neighbors of a vertex i form a set $\mathcal{V}_1(i) = \{j \mid \{i, j\} \in \mathcal{E}\}$ (see Figure 2, left). $K_i = \#\mathcal{V}_1(i)$ is the *degree* of i. The edges from i to its neighbors form a set $\mathcal{E}_1(i) = \{\{i, j\} \mid j \in \mathcal{V}_1(i)\}$. A 1-ring neighborhood with flaps is shown in Figure 2 (middle). Its vertices except the center vertex form a set $\mathcal{V}_2(i)$ and its interior edges form a set $\mathcal{E}_2(i)$. Finally, the neighborhood $\omega(e)$ of an edge (see Figure 2) is formed by the 4 vertices of its incident triangles.

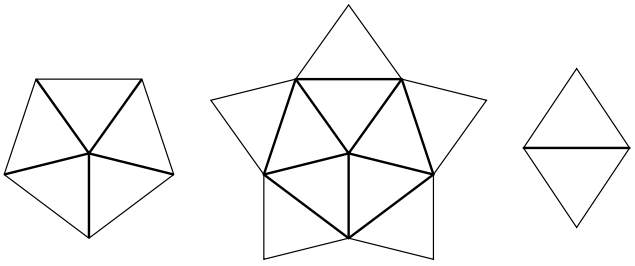

Figure 2: *Left: 1-ring neighborhood. The vertices except the center one form $\mathcal{V}_1(i)$ and the bold edges form $\mathcal{E}_1(i)$. Middle: 1-ring with flaps. The vertices except the center one form $\mathcal{V}_2(i)$ and the bold edges form $\mathcal{E}_2(i)$. Right: Edge neighborhood. The four vertices of the incident triangles form $\omega(e)$.*

The *geometric realization* $\varphi(s)$ of a simplex s is defined as the strictly convex hull of the points p_i with $i \in s$. The polyhedron $\varphi(\mathcal{K})$ is defined as $\cup_{s \in \mathcal{K}} \varphi(s)$ and consists of points, segments, and triangles in $\mathbf{R}^3$.

2.1 Divided Differences in the Functional Setting

Our relaxation algorithm relies on minimizing divided differences. In the one dimensional setting divided differences are straightforward to define, but for multivariate settings many approaches are possible (see for example [10, 4, 3]). An approach that was developed specifically with subdivision in mind is described in [12] and we use it here for our purposes.

Consider a face $f = \{i, j, k\}$ and the triangle $t = \varphi(f)$ where $p_i = (u_i, v_i, g_i)$. Then the first order divided difference of g at f is simply the gradient of the piecewise linear spline interpolating g denoted by $\nabla_f g = (\partial g/\partial u, \partial g/\partial v)$. Note that the gradient depends on the parameter locations (u_i, v_i) and converges in the limit to the first partial derivatives. If we create a three vector by adding a third component equal to 1, we obtain the normal $n_f = (-\partial g/\partial u, -\partial g/\partial v, 1)$ to the triangle t. Conversely, the gradient is the projection of the normal in the parameter plane. Consequently the gradient is zero only if the triangle t is horizontal ($g_i = g_j = g_k$).

Second order differences are defined as the difference between two normals on neighboring triangles and can be associated with the common edge (see Figure 3, left). Consider an edge $e = \{j, k\}$ with its two incident faces $f_1 = \{j, k, l_1\}$ and $f_2 = \{j, k, l_2\}$ (see Figure 2, right). Compute the difference between the two normals $m_e = n_{f_2} - n_{f_1}$. Given that the two normals are orthogonal to $\varphi(e)$ so is their difference m_e (see Figure 3, right). But the third component of m_e is zero, hence m_e itself lies in the parameter plane, which also contains the segment between $(u_j, v_j, 0)$ and $(u_k, v_k, 0)$. This implies that m_e is orthogonal to the segment and hence only its signed magnitude matters (see Figure 3).

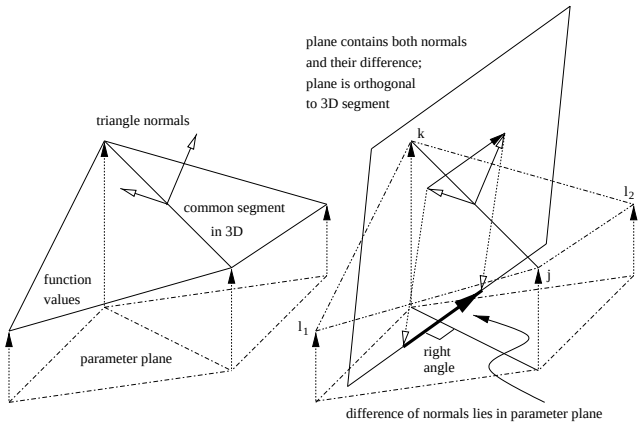

Figure 3: *In the functional setting triangles are erected over the parameter plane. Their normals generate a plane orthogonal to the edge in 3-space. Any vector in that plane which is also in the parameter plane must be at right angles with the parameter plane segment. Hence D_e^2 is orthogonal to $(u_j, v_j) - (u_k, v_k)$.*

This argument justifies defining the second order difference $D_e^2 g$ as the component of m_e orthogonal to the segment in the parameter plane. $D_e^2 g$ depends on four function values at vertices $\omega(e) = \{j, k, l_1, l_2\}$. Since all operations to compute $D_e^2 g$ are linear (gradient, difference, and projection) so is the entire expression

$$D_e^2 g = \sum_{l \in \omega(e)} c_{e,l} g_l.$$

The coefficients are given by

$$c_{e,l_1} = \frac{L_e}{A_{[l_1,k,j]}}, \quad c_{e,l_2} = \frac{L_e}{A_{[l_2,j,k]}},$$

$$c_{e,j} = -\frac{L_e A_{[k,l_2,l_1]}}{A_{[l_1,k,j]}\, A_{[l_2,j,k]}}, \quad c_{e,k} = -\frac{L_e A_{[j,l_1,l_2]}}{A_{[l_1,k,j]}\, A_{[l_2,j,k]}}, \quad (1)$$

where $A_{[k_1,k_2,k_3]}$ is the signed area of the triangle formed by (u_{k_1}, v_{k_1}), (u_{k_2}, v_{k_2}), (u_{k_3}, v_{k_3}); and L_e is the length of the segment between (u_j, v_j) and (u_k, v_k) [12]. All the parameterization information is captured in the edge length and signed triangle areas. Given that we later only use squares of D_e^2 the actual sign of the areas is not important as long as the orientations prescribed by (1) are consistent. Also, note that the second order difference operator is zero only if the two triangles lie in the same plane.

2.2 Relaxation in the Functional Setting

The central ingredient in our signal processing toolbox is a non-uniform relaxation operator. It generalizes the usual notion of a low pass filter. We begin by discussing the construction of such a relaxation operator in the functional setting.

The purpose of the relaxation operation is the minimization of second order differences. To this end we define a quadratic energy, which is an instance of a discrete fairing functional [16]

$$\mathbf{E} = \textstyle\sum_{e \in \mathcal{E}} (D_e^2 g)^2.$$

The relaxation is computed locally, i.e., for a given vertex i we compute a relaxed function value Rg_i based on neighboring function values g_j. Treating $\mathbf{E}$ as a function of a given g_i the relaxed value Rg_i is defined as the minimizer of $\mathbf{E}(g_i)$. Given that the stencil for D_e^2 consists of two triangles, all edges which affect $\mathbf{E}(g_i)$ belong to $\mathcal{E}_2(i)$ (see Figure 2, middle)

$$Rg_i = \arg\min \mathbf{E}(g_i) = \arg\min \textstyle\sum_{e \in \mathcal{E}_2(i)} (D_e^2 g)^2. \quad (2)$$

Since the functional is quadratic the relaxation operator is linear in the function values. To find the expression, write each of the $D_e^2 g$ with $e \in \mathcal{E}_2(i)$, i.e., all second differences depending on g_i, as a linear function of g_i

$$D_e^2 g = c_{e,i} g_i + \alpha_e \;\text{ with }\; \alpha_e = \textstyle\sum_{l \in \omega(e) \setminus \{i\}} c_{e,l} g_l.$$

Setting the partial derivative of $\mathbf{E}$ with respect to g_i equal to zero yields

$$Rg_i = -\left(\textstyle\sum_{e \in \mathcal{E}_2(i)} c_{e,i}\, \alpha_e\right) / \left(\textstyle\sum_{e \in \mathcal{E}_2(i)} c_{e,i}^2\right), \quad (3)$$

which can be rewritten as

$$Rg_i = \textstyle\sum_{j \in \mathcal{V}_2(i)} w_{i,j} g_j, \;\; w_{i,j} = -\frac{\sum_{\{e \in \mathcal{E}_2(i) | j \in \omega(e)\}} c_{e,i} c_{e,j}}{\sum_{e \in \mathcal{E}_2(i)} c_{e,i}^2}.$$

There are two ways to implement R which trade off speed versus memory. One can either precompute and store the $w_{i,j}$ and use the above expression or one can use (3) and compute R on the fly.

Note that if g is a linear function, i.e., all triangles lie in one plane, the fairing functional $\mathbf{E}$ is zero. Consequently linear functions are invariant under R. In particular R preserves constants from which we deduce that the $w_{i,j}$ sum to one.

To summarize, given an arbitrary but fixed triangulation in the parameter plane and function values g_i with the associated (u_i, v_i) coordinates, simple linear expressions describe first and second differences. The coefficients of these expressions depend on the parameterization. The relaxation operator R acts on individual function values to minimize the discrete second difference energy over the $\mathcal{E}_2(i)$ neighborhood of a given $p_i = (u_i, v_i, g_i)$, leaving linears invariant.

2.3 Relaxation in the Surface Setting

To apply the above relaxation in the surface setting we need to have parameter values (u, v) associated with every point in our mesh. Typically such parameter values are not available and we must compute them. One possible solution is to compute a global parameterization to a coarse base domain using approaches such as those described in [8, 19]. However, specifying parameter values for an entire region is equivalent to flattening that region and thus invariably introduces distortion. Therefore we wish to keep the parameter regions as small as possible. Typically one computes parameter values for a certain local neighborhood like a 1-ring. We propose an even more local scheme in which parameter values are specified *separately* for each of the D_e^2 stencils. The two triangles of the D_e^2 stencil get flattened with the so-called *hinge map*: using the common edge as a hinge, rotate one triangle until it lies in the plane defined by the other triangle and compute the needed edge lengths and areas from (1). Note that the hinge map leaves the areas of the triangles $\varphi(f_1)$ and $\varphi(f_2)$ unchanged and only affects the faces $\{j, k, l_1\}$ and $\{j, k, l_2\}$. The surface relaxation operator is defined as before, but acts on points in $\mathbf{R}^3$

$$Rp_i = \textstyle\sum_{j \in \mathcal{V}_2(i)} w_{i,j} p_j.$$

Our minimization is similar to minimizing dihedral angles [21]. However, minimizing exact dihedral angles is difficult as the expressions depend non-linearly on the points. Instead one can think of the D_e^2 as a linear expression which behaves like the dihedral angle.

Features With our scheme it is particularly easy to deal with features in the mesh. Examples include sharp edges across which one does not wish to smooth. In that case the D_e^2 associated with those edges are simply removed from the functional.

One may worry what happens with the equations in (1) in case one of the triangles is degenerate, i.e., two of its points coincide and its area is zero. Then the D_e^2 that use this triangle are not defined and simply can be left out from the optimization. This is similar to coinciding knots in the case of splines.

Comparison with Existing Schemes The approach followed in [18] is to assume that the 1-ring neighborhood of a vertex i is parameterized over a regular K_i-gon. Using this approximation a discrete Laplacian, dubbed umbrella, is computed as

$$Lp_i = K_i^{-1} \textstyle\sum_{j \in \mathcal{V}_1(i)} p_j - p_i.$$

This discrete Laplacian was used in a relaxation operator $R = I + L$ which replaces a vertex with the average of its 1-ring neighbors.

In our setting, we can build a 1-ring relaxation scheme by only taking the minimum in (2) over $\mathcal{E}_1(i)$. The relaxation operator is then computed as in (3) with summations over $\mathcal{E}_1(i)$ rather than $\mathcal{E}_2(i)$. Our 1-ring scheme parameterized on a *regular* K_i-gon leads to the same relaxation operator as used by Kobbelt. Our scheme can thus be seen as a natural non-uniform generalization of the umbrella

which is still linear. In general we use the $\mathcal{E}_2(i)$ (1-ring with flaps) scheme as it yields visually smoother surfaces.

Taubin [26] presents a two step relaxation operator $R = (I + \mu L)(I + \lambda L)$, with μ and λ tuned to minimize shrinkage of the mesh.

Both of these schemes are semi-uniform filters since the weights only depend on K_i and not the geometry. Consequently they affect both geometry and parameterization. Consider again an irregular triangulation of a plane. Semi-uniform schemes try to make each 1-ring look as much as possible like a regular K-gon. Thus the triangulation may change globally while the plane remains the same. As we will see, this will lead to unwanted effects in applications such as editing and texture mapping. On the other hand our non-uniform scheme is linearly invariant, leaves the triangles unchanged, and does not suffer from the problems concerning movement "inside" the surface observed in [18, Figure 4].

Figure 4 shows the effect on a non-planar triangulation like the eye of the mannequin head. Our non-uniform scheme (right) smoothes the geometry without affecting the triangle shapes much. The semi-uniform scheme (middle) tries to make edge lengths as uniform as possible which can only be done by effectively destroying the delicate mesh structure around the eye. This effect also applies to any other attributes that vertices may carry such as detail vectors for editing or texture map coordinates causing distortion during smoothing (see Figure 8).

Taubin [26] also uses a non-uniform discrete Laplacian in which the weights vary as the powers of the respective edge lengths. While such an operator can greatly reduce the triangle distortions, it can be shown that such a scheme can never be linearly invariant.

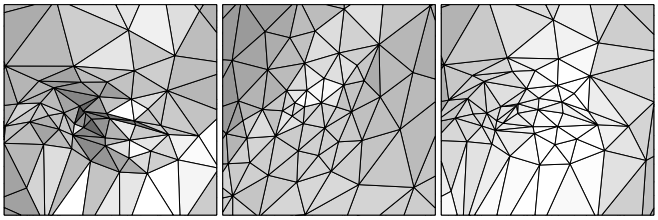

Figure 4: *Smoothing of the eye (left) with our non-uniform (right) and a semi-uniform scheme (middle). The semi-uniform scheme tries to make edge lengths as uniform as possible and severely distorts the geometry, while the non-uniform scheme only smoothes the geometry and does not affect the triangle shapes much.*

3 Multiresolution Signal Processing

Up to this point we have only considered operators which act on a scale comparable to their small finite support. To build more powerful signal processing tools we now consider a multiresolution setting.

Multiresolution algorithms such as subdivision, pyramids, and wavelets require decimation and upsampling procedures. For images decimation comes down to removing every other row or column. The situation for meshes is more complex, but a considerable body of work is available [13].

We employ Hoppe's Progressive Mesh (PM) approach [14]. In the PM setting, an edge collapse provides the atomic decimation step, while a vertex split becomes the atomic upsampling step. For simplicity we only employ half-edge collapses in our implementation. As a priority criterion we use a combination of the Garland-Heckbert quadric error metric [11] and edge length to favor removal of long edges (see also [17]).

Each half edge collapse removes one vertex and we number them in reverse so that the one with highest index gets removed first. This gives a sequence of N meshes $(\mathcal{P}^n, \mathcal{K}^n)$, $1 \le n \le N$, and $\mathcal{P}^n = \{p_i \mid 1 \le i \le n\}$. Later we will consider mesh sequences $(\mathcal{Q}^{(n)}, \mathcal{K}^{(n)})$ where the points on coarser meshes do move from their finest mesh position. These are denoted $q_i^{(n)}$, $i \le n$.

In traditional signal processing, downsampling creates a coarser level through the removal of a constant fraction of samples. This leads to a logarithmic number of levels. A PM does not have such a notion of levels. However, one may think of each removed vertex as living on its own level, and the number of levels being linear.

3.1 Subdivision

Subdivision starts from a coarse mesh and successively builds finer and smoother versions [22]. In signal processing terms it consists of upsampling followed by relaxation. So far the word subdivision has been associated in the literature with either regular or semi-regular meshes with corresponding uniform or semi-uniform operators. If one only has an original, coarse mesh and cares about building a smooth version, then semi-regular is the correct approach.

Our setting is different. The coarse mesh comes from a PM started at the original, finest level. Hence the connectivity of the finer levels is fixed and determined by the reverse PM. Our goal is to use non-uniform subdivision to build a *geometrically* smooth mesh with the *same* connectivity as the original mesh and with as little triangle shape distortion as possible. Such smoothed meshes can subsequently be used to build pyramid algorithms.

Subdivision is computed one level at a time starting from level n_0 in the progressive mesh $\mathcal{Q}^{(n_0)} = \mathcal{P}^{(n_0)}$. Since the reverse PM adds one vertex per level, our non-uniform subdivision is computed one vertex at a time. We denote the vertex positions as $\mathcal{Q}^{(n)} = \{q_i^{(n)} \mid 1 \le i \le n\}$ $(n \ge n_0)$ and use meshes $(\mathcal{Q}^{(n)}, \mathcal{K}^{(n)})$ with the same connectivity as the PM meshes.

Going from $\mathcal{Q}^{(n-1)}$ to $\mathcal{Q}^{(n)}$ involves three groups of vertices. (I) the new vertex n, which is introduced together with a point position $q_n^{(n)}$ to be computed. (II) certain points from the $\mathcal{Q}^{(n-1)}$ mesh change position; these correspond to *even* vertices. There is only a small number of them. (III) the remainder of the points of $\mathcal{Q}^{(n-1)}$, typically the majority, remains unchanged. Specifically:

- The new position $q_n^{(n)}$ is computed after upsampling from $\mathcal{K}^{n-1}$ to $\mathcal{K}^n$:
$$q_n^{(n)} = \textstyle\sum_{j \in \mathcal{V}_2^n(j)} w_{n,j}^{(n)} q_j^{(n-1)}.$$
The position of the new vertex is computed to satisfy the relaxation operator using points from $\mathcal{Q}^{n-1}$ with weights using areas and lengths of mesh $(\mathcal{P}^n, \mathcal{K}^{(n)})$.
- The even points of $\mathcal{Q}^{n-1}$ form a 1-ring neighborhood of n. Their respective $\mathcal{V}_2^n$ neighborhoods contain n, which has just received an updated position $q_n^{(n)}$
$$\forall j \in \mathcal{V}_1^n(n): \quad q_j^{(n)} = \textstyle\sum_{k \in \mathcal{V}_2^n(j) \setminus \{n\}} w_{j,k}^{(n)} q_k^{(n-1)} + w_{j,n}^{(n)} q_n^{(n)}.$$
The even vertices are relaxed using the point positions from $\mathcal{Q}^{(n-1)}$ (except for $q_n^{(n)}$), using weights coming from $(\mathcal{P}^n, \mathcal{K}^{(n)})$.
- Finally, the remainder of the positions do not change
$$\forall j \in \mathcal{V}^{n-1} \setminus \mathcal{V}_1^n(n): \quad q_j^{(n)} = q_j^{(n-1)}.$$

A central ingredient in our construction is the fact that the weights $w_{i,j}^{(n)}$ depend on parameter information from the mesh $\mathcal{P}^{(n)}$. No globally or even locally consistent parameterization is required. For each D_e^2 stencil we use the hinge map as described above. In effect the original mesh provides the parameterizations and in this way enters into the subdivision procedure. The actual *areas* and *lengths*, which make up the expressions for $w_{i,j}^{(n)}$ are assembled based on the connectivity $\mathcal{K}^{(n)}$ of level n, and hence induce the level dependence

of the weights. As a result all $w_{i,j}^{(n)}$ may be precomputed during the PM construction and can be stored if desired for later use during repeated subdivision. It is easy to see that the storage is linear in the total degree, $\sum_i K_i$, of the mesh.

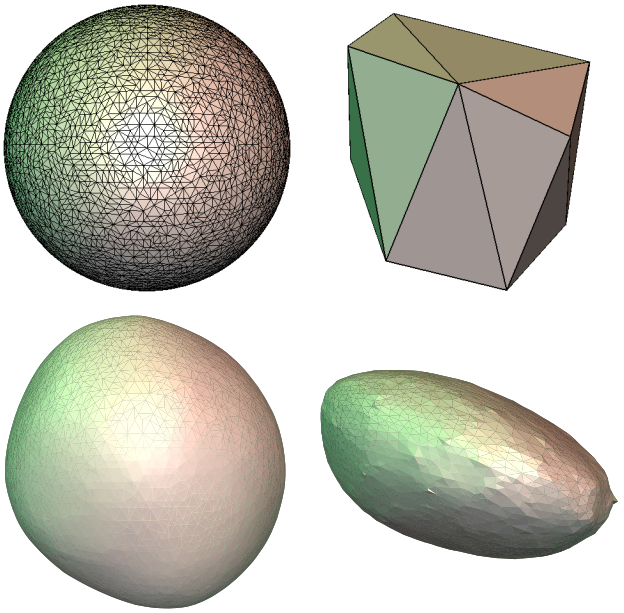

Figure 5: *Starting with the irregular triangulation of a sphere (upper left) we compute a PM down to 16 triangles (upper right). We then compute our non-uniform subdivision scheme back to the finest level (lower left) and obtain a smooth mesh which approximates the original. For comparison the lower right shows the limit surface of a semi-uniform subdivision scheme.*

To illustrate the behavior of uniform functional subdivision schemes one considers the so called *scaling function* or fundamental solution obtained from starting with a Kronecker sequence on the coarsest level. For surface subdivision, there is no equivalent to this. To illustrate the behavior of the surface scheme we perform the following experiment (see Figure 5). We start with an irregular triangulation of a sphere with 12000 triangles (upper left) and compute a PM down to 16 triangles (upper right). Next the non-uniform surface subdivision scheme starting from the 16 triangles back to the original mesh is computed (lower left). We clearly get a smooth mesh. For comparison the lower right shows the limit function using a semi-uniform scheme. It is important to understand that the non-uniform scheme has access to the parameterization information of the original finest mesh whereas the semi-uniform scheme does not use this additional information.

While for uniform and semi-uniform subdivision, extensive literature on regularity of limit functions exists, few results are known for non-uniform subdivision [2, 12]. The goal of our strategy of minimizing D_e^2 is to obtain C^1 smoothness. However, there is currently no regularity result for our scheme in either the functional or surface setting.

3.2 Burt-Adelson Pyramid

The pyramid proposed by Burt and Adelson [1] (BA) is another important signal processing tool. We show how to generalize it to a mesh pyramid. We start from the finest level points $\mathcal{S}^N = \mathcal{P}$ and compute a sequence of meshes $(\mathcal{S}^n, \mathcal{K}^n)$ $(1 \leq n \leq N)$ as well as oversampled differences $d_i^{(n)}$ between levels.

To go from $\mathcal{S}^n$ to $\mathcal{S}^{n-1}$, i.e., to remove vertex n, we follow the diagram in Figure 6. The top wire represents the points of $\mathcal{S}^{n-1}$ while the bottom wire represent the points of $\mathcal{S}^n$. There are four

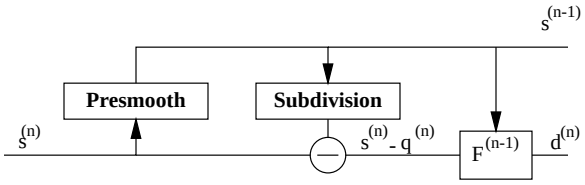

Figure 6: *Burt-Adelson style pyramid scheme.*

stages: presmoothing, downsampling, subdivision, and computation of details.

- **Presmoothing:** Presmoothing in the original BA pyramid is important to avoid aliasing. We have found that in a PM the presmoothing step can often be omitted because the downsampling steps (edge collapses) are chosen carefully, depending heavily on the data. In essence vertices are removed mostly in smooth regions, where presmoothing does not make a big difference. Thus, no presmoothing was used in our implementation.
- **Downsampling:** n is removed in a half-edge collapse.
- **Subdivision:** Using the points from $\mathcal{S}^{n-1}$ we compute subdivided points $q_j^{(n)}$ for the vertex just removed and the surrounding even vertices exactly as described in Section 3.1
- **Detail Computation:** Finally, detail values are computed for all even vertices as well as the vertex n. These detail vectors are expressed in a local frame $F_j^{(n-1)}$ which depends on the coarser level:

$$\forall j \in \mathcal{V}_1^n(n) \cup \{n\}: \quad d_j^{(n)} = F_j^{(n-1)}(s_j^{(n)} - q_j^{(n)}).$$

  We refer to the entire group of $d_j^{(n)}$ as an array $d^{(n)}$. In the implementation this array is stored with n.

One of the features of the BA pyramid is that the above procedure can always be inverted independent of which presmoothing operator or subdivision scheme is used. For reconstruction, we start with the points of $\mathcal{S}^{n-1}$, subdivide values $q_j^{(n)}$ for both the new and even vertices and add in the details to recover the original values $s_j^{(n)}$.

To see the potential of a mesh pyramid in applications it is important to understand that the details $d^{(n)}$ can be seen as an approximate frequency spectrum of the mesh. The details $d^{(n)}$ with large n come from edge collapses on the finer levels and thus correspond to small scales and high frequencies, while the details $d^{(n)}$ with small n come from edge collapses on the coarser levels and thus correspond to large scales and low frequencies.

Oversampling factor A standard image pyramid has an oversampling factor of 4/3, while we have an expected oversampling factor of 7. The advantage of oversampling is that the details are quite small and lead to natural editing behavior [29]. If needed, a technique exists to reduce the oversampling factor. The idea is to use levels with more than one vertex. Say, we divide the N vertices of $\mathcal{V}$ into M levels with $M \ll N$:

$$\mathcal{V} = \mathcal{V}_0 \cup \bigcup_{1 \leq m \leq M} \mathcal{W}_m \quad \text{and} \quad \mathcal{V}_m = \mathcal{V}_{m-1} \cup \mathcal{W}_m.$$

This can be done, for example, so that the sizes of the $\mathcal{V}_m$ grow with a constant factor [7]. The BA pyramid then goes from $\mathcal{V}_m$ to $\mathcal{V}_{m-1}$. First presmooth all even vertices in $\mathcal{V}_m$, then compute subdivided values for all vertices in $\mathcal{W}_m$ and their 1-ring neighbors in $\mathcal{V}_m$. For the subdivided points, which need not be all vertices of $\mathcal{V}_m$, compute the details as differences with the original values from $\mathcal{V}_m$. One can see that the above algorithm with oversampling factor 7 is a special case when $\mathcal{W}_m = \{m\}$. The other extreme is the case with only one level containing all vertices. In that case

there is no multiresolution as all details live on the same level. The oversampling factor is 1. By choosing the levels appropriately one can obtain any oversampling between 1 and 7. It is theoretically possible to build a wavelet-like, i.e, critically sampled multiresolution transform based on the Lifting scheme [25]. However, at this point it is not clear how to design filters that make the transform stable.

Caveat Often in this paper we use signal processing terminology such as frequency, low pass filter, aliasing, to describe operations on 2-manifolds. One has to be extremely careful with this and keep in mind that unlike in the Euclidean setting, there is no formal definition of these terms in the manifold setting. For example in a mesh the notion of a DC component strictly does not exist. Also in connection with the pyramid we often talk about frequency bands. Again one has to be careful as even in the Euclidean setting the coefficients in a a pyramid do not represent exact frequencies due to the Heisenberg uncertainty principle.

4 Applications

The algorithms we described above provide a powerful signal processing toolbox. In this section we demonstrate this claim by considering a variety of applications that use them. These include smoothing and filtering, enhancement, texture coordinate generation, vector displacement field editing, and multiresolution editing.

4.1 Smoothing and Filtering

One way to smooth a mesh is through repeated application of the relaxation operator R. Numerically this behaves similarly to traditional Jacobi iterations for an elliptic PDE solver. The relaxation rapidly attenuates the highest frequencies in the mesh, but has little impact on low frequencies. Even though each iteration of the operator is linear in the number of vertices, the number of iterations to attenuate a fixed frequency band grows linearly with the mesh size. This results in quadratically increasing run times as the sample density increases relative to a fixed geometric scale. One way to combat this behavior is through the use of appropriate preconditioners, as was done in [18], or through the use of implicit solvers [6].

Using a mesh pyramid we can build much more direct and flexible filtering operations. Recall that the details in a pyramid measure the local deviation from smoothness at different scales. In that sense they capture the local frequency content of the mesh. This spectrum can be shaped arbitrarily by scaling particular details. Multiresolution filtering operators are built by setting certain ranges of detail coefficients in the pyramid to zero. A low pass filter sets all detail arrays $d^{(n)}$ with $n > n_l$ to zero, while a high pass filter annihilates $d^{(n)}$ for $n < n_h$. However, for meshes it makes little sense to put the coarsest details to zero as this would collapse the mesh. More natural for meshes are stopband filters which zero out detail arrays $d^{(n)}$ in some intermediate range, $n_l < n < n_h$.

Figure 7 shows these procedures applied to the venus head ($N = 50000$). On the upper left the original mesh. The upper right shows the result of applying the non-uniform relaxation operator 20 times at the finest level. High frequency ripples quickly diffuse, but no attenuation is noticeable at larger length scales. The bottom left shows the result of a low pass filter which sets all details above $n_l = 1000$ to zero. Finally the bottom right shows the result of a stopband filter, annihilating all details $1000 < n < 15000$. Note how the last mesh keeps its fine level details, while intermediate frequencies were attenuated. If desired all these filtering operations can be performed in a spatially varying manner due to the space-frequency localization of the mesh pyramid. Figure 8 shows the difference between non-uniform (left) and semi-uniform smoothing (right) on the actual vertex positions. By keeping the original finest level texture coordinates for the vertices of both meshes we can visualize the effect of movement "within" the surface after smoothing. This hints at another application: if one has a scanned mesh with color (r,g,b) attributes per vertex then non-uniform *geometry* smoothing will not distort those colors.

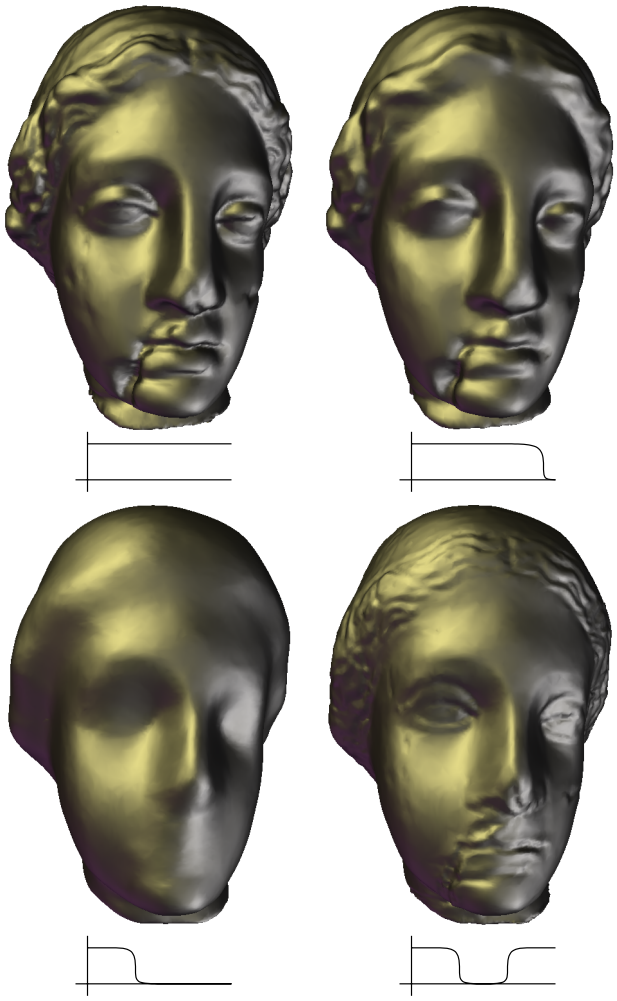

Figure 7: *Smoothing and filtering of the venus head. Original on the top left; 20 finest level relaxation steps on the top right; low pass filter on the bottom left; stopband filter on the bottom right.*

4.2 Enhancement

Enhancement provides the opposite operation to smoothing in that it emphasizes certain frequency ranges. As before this can be done in a single resolution manner as well as in the more flexible multiresolution setup.

The single resolution scheme is easy to compute and typically works best for fairly small meshes, such as those used as control polyhedra for splines or semi-regular subdivision surfaces. The main idea is to extrapolate the difference between the original mesh and a single resolution relaxed mesh. The enhanced points are given by

$$Ep_i = p_i + \xi(R^k p_i - p_i),$$

where $\xi > 1$. Figure 9 illustrates the procedure. On the left the original mannequin head, in the middle the result after 20 relaxation steps, and on the right the enhanced version with $\xi = 2$. The first and last models of Figure 1 show the Loop subdivided meshes of the original and enhanced head. By using combinations of the different algorithms peculiar effects can be obtained. The second

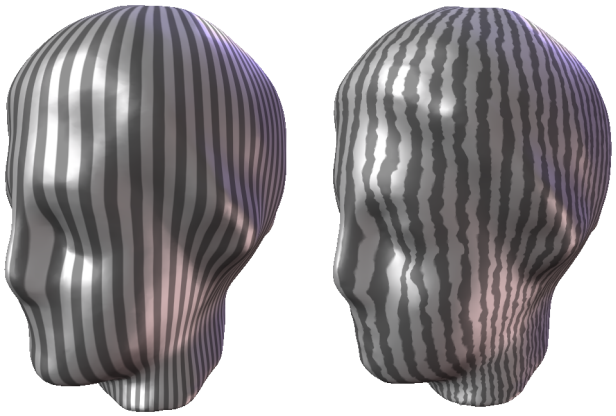

Figure 8: *Movement "within" the surface due to smoothing visualized by letting the vertices keep their original finest level texture coordinates. Left non-uniform smoothing and right semi-uniform smoothing.*

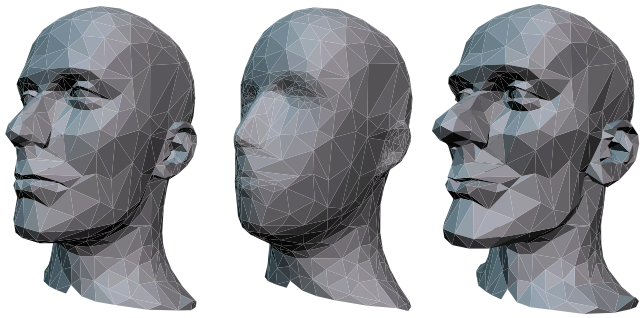

Figure 9: *Enhancement of control mesh. On the left the original, in the middle the smoothed mesh, and on the right the enhanced mesh (see also Figure 1 for the resulting subdivision surfaces).*

model in Figure 1 is obtained by extrapolating from a base model built by 5 semi-uniform relaxation steps followed by 5 non-uniform relaxation steps (needed to recover the parameterization and "pull" features back in place). The third model in Figure 1 is extrapolated from a base built by first simplifying to level 100, then applying 1 relaxation step (which made the chin collapse and ears shrink), and reconstructing.

The single level scheme is simple and easy to compute, but limited in its use. For example, it does not compute offsets with respect to local frames. If the mesh contains fine level detail self intersections quickly appear. As in image enhancement one must be careful not to amplify high frequency noise. For these reasons we need the more flexible setup of multiresolution enhancement. The approach is simple, we compute a mesh pyramid, scale the desired details and then reconstruct. As in the filtering application, the user has control over the different frequency bands. Additionally, the local frames across the many levels of the mesh pyramid tend to stabilize the procedure and lead to a more natural behavior. As a result the multiresolution enhancement scheme deals better with large scanned meshes which usually contain high frequency noise.

Figure 10 shows Loop subdivided versions of the original cow head and an enhanced version obtained by multiplying the details $d^{(n)}$ with $257 < n \leq 2904 = N$ by two (see also Figure 15, right column for an edit of the enhanced model). Finally, Figure 11 shows enhancement on the Stanford bunny ($N = 34835$). Here details with indices $1000 < n < 7000$ were multiplied by 2, and details with indices $7000 < n < 13000$ were multiplied by 1.5.

Figure 10: *Enhancement of cow head (original on the left).*

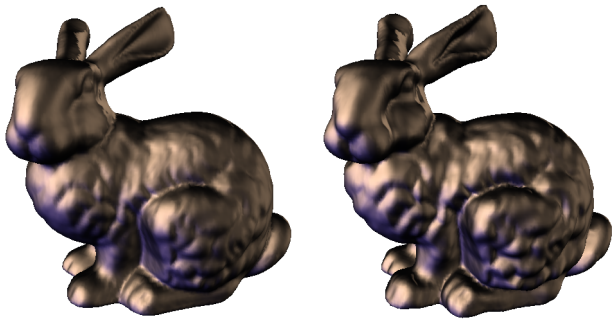

Figure 11: *Enhancement on the bunny. The original is on the left and the frequency enhanced version on the right.*

4.3 Subdivision of Scalar Functions on Manifolds

We can use subdivision to quickly build smooth scalar functions defined *on* a manifold. Simply start with scalar values on a coarse level and use non-uniform subdivision to build a smooth function defined on the finest level.

We present two applications. The first creates smoothly varying texture coordinate assignments for the finest level mesh from some user supplied texture coordinate assignments at a coarse level. The second creates a smoothly varying function over a limited region of an irregular mesh and then uses this function to generate a smooth vector displacement field for shape editing purposes.

Texture Coordinate Generation DeRose et al. [5] discuss this problem in the context of classical, semi-uniform subdivision. Their goal was the construction of smooth texture coordinates for Catmull-Clark surfaces. Beginning with user supplied texture coordinates at some coarse level they subdivide these parameter assignments to the finest subdivision level using the same subdivision operator for texture coordinates as for the vertices.

Figure 12 shows the application of this idea to our setting. Initial texture coordinate assignments were made using a cylindrical projection of all vertices in $\mathcal{P}^{1000}$. The left image shows a test texture on the coarse polygonal mesh. We then reconstruct the original finest level mesh and concurrently subdivide the texture coordinates to the finest level. The resulting mapping is shown on the right. Even though the geometry has much geometric detail and uneven triangle sizes the final texture coordinates vary smoothly over the entire surface.

Displacement Vector Field Editing Singh and Fiume [23] present an algorithm for deformation edits based on vector displacement fields. These fields are defined through a smooth falloff function around a "wire" which drags the surface along. The region of influence is a function of distance in $\mathbf{R}^3$. Controlling this behavior in regions of high curvature or in the vicinity of multiple close objects can be tricky. In our setting we have the opportunity to define the falloff function *only* on the surface itself. A similar idea was used in [15] for feature editing.

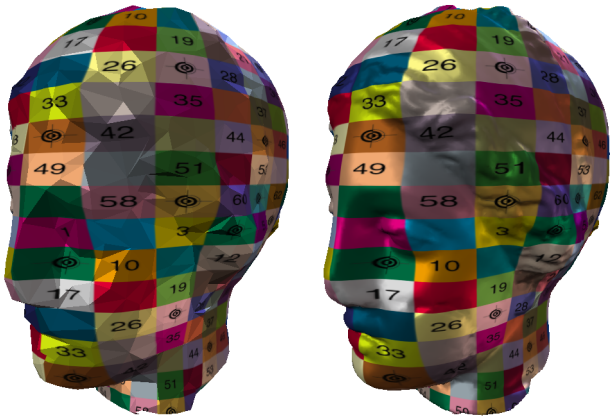

Figure 12: *A test texture is mapped to a coarse level of the mesh pyramid under user control. The resulting texture coordinates are then subdivided to the finest level and the result shown on the right.*

We illustrate this idea with an example. Consider the horse to "giraffe" edit in Figure 13. The user first outlines three regions by drawing closed curves on the mesh. A region that remains unchanged (A); a region that will be gradually stretched (B); and a region that will undergo a translation (C). In our example, region (A) is the back body and the four legs; (B) are the neck and torso; and (C) is the head. The boundary between (A) and (B) consists of three closed curves. Next we define a scalar parameter θ, which is 0 on the boundary between (A) and (B), and 1 on the boundary between (B) and (C). The algorithm computes values for θ that vary smoothly between 0 and 1 in region (B).

This is accomplished by running a PM on the interior of region (B) to a maximally coarse level. Then the initial value $\theta = 1/2$ is assigned to all interior vertices of the coarse region (B). Next we apply relaxation to θ on the coarsest level within (B). This converges quickly because there are few triangles; three steps suffice. These θ values are then used as the starting values for subdivision from the coarsest level back to the original region (B) while keeping the θ values on the boundary fixed. The resulting θ values on the finest region (B) vary smoothly between 0 and 1. The only problem is that at the boundary they meet in a C^0 and not a C^1 fashion. This is because we only imposed Dirichlet like conditions and no Neumann condition. We address this with the following smoothing transformation, $\theta := 1/2 - 1/2\cos(\pi\theta)$.

On the left of Figure 13 the red lines are specified by the user and the black lines show the θ isolines, visualizing how θ varies smoothly. The edit is now done by letting the user drag the head. Every vertex in region B is subjected to θ times the displacement vector of the head. This requires very little computation. The right side of Figure 13 shows the result.

4.4 Multiresolution Editing

The displacement vector editing is simple and fast, but has limited use. We next discuss full fledged multiresolution editing for irregular meshes. Our algorithm combines ideas of Zorin et al. [29] and Kobbelt et al. [18]. The former used multiresolution details and semi-regular meshes, while the latter used single resolution details and irregular meshes. We combine the best of both approaches by using multiresolution details with the irregular mesh setting.

The algorithm is straightforward. The user can manipulate a group of points $s_i^{(n)}$ in the mesh pyramid and the system adds the finer level details back in. This is exactly the same use of the pyramid as Zorin et al. only now for irregular meshes. Kobbelt et al. used a *multiresolution/multigrid* approach to define a smoothed mesh over a user selected region, but then compute *single resolution* details between the original and smoothed mesh.

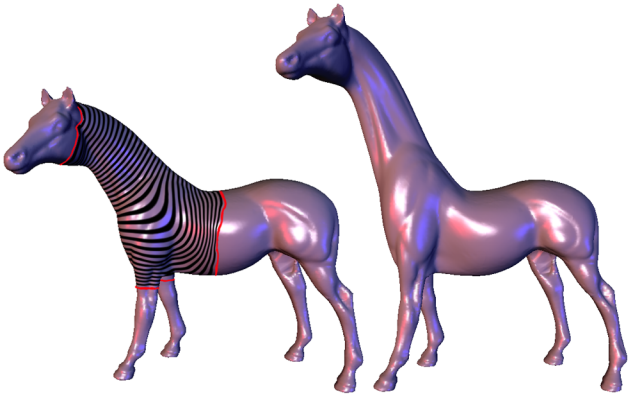

Figure 13: *Horse to giraffe edit using a surface based smooth displacement vector field.*

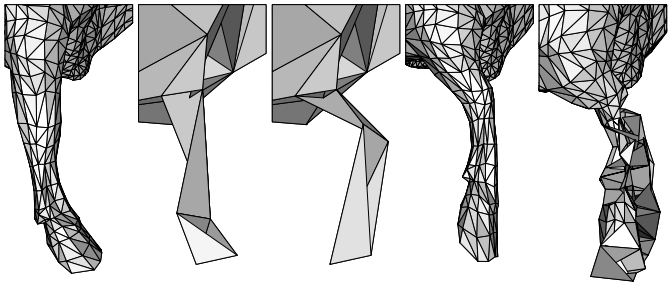

Figure 14: *Cow leg editing sequence: original, coarsest scale, edit, reconstruction with multiresolution details, reconstruction with single resolution details.*

The use of multiresolution details is important when the user wishes to make large scale edits in regions with complicated fine scale geometry. Because the multiresolution details are all described in local frames, they have more flexibility to adjust themselves to a coarse scale edit.

We illustrate this with an edit on the leg of the cow (Figure 14). The sequence shows the original leg, the coarse leg, a coarse edit, and two reconstructions. The first used multiresolution details while the second used single resolution details.

Finally, Figure 15 shows some additional edits. The horse was edited at a level containing only 34 vertices (compare to the original shape shown in Figure 13). The cow edit on the right column involves both manipulation at coarse levels (snout, horns, leg, tail) and overall enhancement.

Dataset	Venus	Horse	Bunny	Cow	Mann.
Size (fine)	50000	48485	34835	2904	689
Size (coarse)	4	34	19	57	5
Timings (s)					
Simpl. & Anal.	79	75	55	3.6	0.8
Reconstruction	9	8	5.8	.37	0.1
Analysis	9	8	5.8	.37	0.1

Table 1: *Timings for mesh pyramid computation assuming storage rather then recomputation of all areas and length needed in stencil weight computations. The size field counts the total vertices (N). Face counts are generally twice as large. All times are given in seconds on an SGI R10k O2 @175Mhz.*

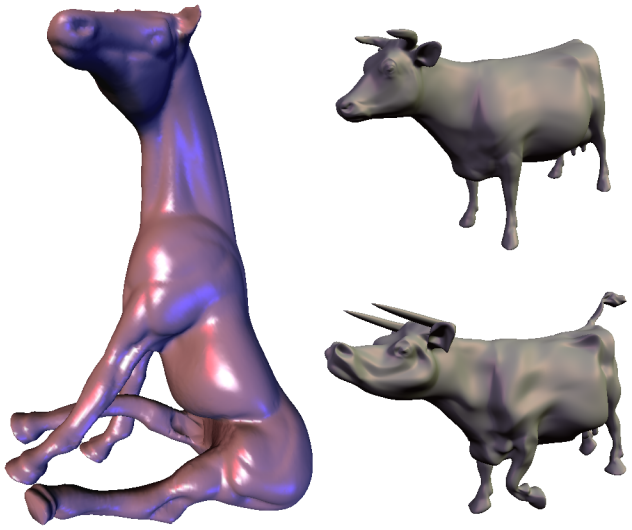

Figure 15: *Multiresolution edits.*

5 Conclusions and Future Work

We have shown how basic signal processing tools such as up and down sampling and filtering can be extended to irregular meshes. These tools can be built into powerful algorithms such as subdivision and mesh pyramids. We have demonstrated their use in texturing, editing, smoothing and enhancement.

Further research can be pursued in several directions. On the algorithms side there is incorporation of various boundary conditions, construction of positive weight schemes, and extensions to tetrahedralizations. On the applications side there is adaptive gridding for time dependent PDE's, computing globally smooth parameterizations, extracting texture maps from scanned textures, and space-frequency morphing.

Compression Another potential future application is compression. However, one needs to be extremely careful: our subdivision weights depend on the parameterization which in turn depends on the geometry of the original mesh. Thus one cannot use the subdivision scheme as a predictor in a compression framework unless sender and receiver share parameter information, i.e., the needed areas and lengths to compute the subdivision. Only a setting where one repeatedly has to communicate functions or attributes defined over a fixed triangulation would justify this overhead.

This touches upon a deeper issue. In some sense for a geometrically smooth irregular mesh only one dimension can effectively be predicted by a subdivision scheme. Even for a geometrically smooth mesh, no subdivision scheme can compress the information implicitly present in the parameterization. Ideally for smooth surfaces one would like to use meshes with as little parametric information as possible.

A typical example are semi-uniform meshes. This argument strongly makes the case for resampling onto semi-regular meshes using smooth parameterizations [8, 19] before compression.

Acknowledgments Igor Guskov was partially supported by a Harold W. Dodds Fellowship and a Summer Internship at Bell Laboratories, Lucent Technologies. Other support was provided by NSF (ACI-9624957, ACI-9721349, DMS-9874082), Alias|wavefront and through a Packard Fellowship. Special thanks to Ingrid Daubechies, Aaron Lee, Adam Finkelstein, Zoë Wood, and Khrysaundt Koenig. Our implementation uses the triangle facet data structure and code of Ernst Mücke, and the priority queue implementation by Michael Garland.

References

[1] BURT, P. J., AND ADELSON, E. H. Laplacian Pyramid as a Compact Image Code. *IEEE Trans. Commun. 31*, 4 (1983), 532–540.

[2] DAUBECHIES, I., GUSKOV, I., AND SWELDENS, W. Regularity of irregular subdivision. *Const. Approx.* (1999), to appear.

[3] DE BOOR, C. A multivariate divided differences. *Approximation Theory VIII 1* (1995), 87–96.

[4] DE BOOR, C., AND RON, A. On multivariate polynomial interpolation. *Constr. Approx. 6* (1990), 287–302.

[5] DEROSE, T., KASS, M., AND TRUONG, T. Subdivision Surfaces in Character Animation. *Computer Graphics (SIGGRAPH '98 Proceedings)* (1998), 85–94.

[6] DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, Aug. 1999.

[7] DOBKIN, D., AND KIRKPATRICK, D. A Linear Algorithm for Determining the Separation of Convex Polyhedra. *Journal of Algorithms 6* (1985), 381–392.

[8] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution Analysis of Arbitrary Meshes. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, 173–182, 1995.

[9] FLOATER, M. S. Parameterization and Smooth Approximation of Surface Triangulations. *Computer Aided Geometric Design 14* (1997), 231–250.

[10] FORNBERG, B. Generation of finite difference formulas on arbitrarily spaced grids. *Math. Comput. 51* (1988), 699–706.

[11] GARLAND, M., AND HECKBERT, P. S. Surface Simplification Using Quadric Error Metrics. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 209–216, 1996.

[12] GUSKOV, I. Multivariate Subdivision Schemes and Divided Differences. Tech. rep., Department of Mathematics, Princeton University, 1998.

[13] HECKBERT, P. S., AND GARLAND, M. Survey of Polygonal Surface Simplification Algorithms. Tech. rep., Carnegie Mellon University, 1997.

[14] HOPPE, H. Progressive Meshes. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 99–108, 1996.

[15] KHODAKOVSKY, A., AND SCHRÖDER, P. Fine Level Feature Editing for Subdivision Surfaces. In *ACM Solid Modeling Symposium*, 1999.

[16] KOBBELT, L. Discrete Fairing. In *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, 101–131, 1997.

[17] KOBBELT, L., CAMPAGNA, S., AND SEIDEL, H.-P. A General Framework for Mesh Decimation. In *Proceedings of Graphics Interface*, 1998.

[18] KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. Interactive Multi-Resolution Modeling on Arbitrary Meshes. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 105–114, 1998.

[19] LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. MAPS: Multiresolution Adaptive Parametrization of Surfaces. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 95–104, 1998.

[20] LÉVY, B., AND MALLET, J. Non-Distorted Texture Mapping for Sheared Triangulated Meshes. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 343–352, July 1998.

[21] MORETON, H. P., AND SÉQUIN, C. H. Functional optimization for fair surface design. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, vol. 26, 167–176, July 1992.

[22] SCHRÖDER, P., AND ZORIN, D., Eds. *Course Notes: Subdivision for Modeling and Animation.* ACM SIGGRAPH, 1998.

[23] SINGH, K., AND FIUME, E. Wires: A Geometric Deformation Technique. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 405–414, 1998.

[24] SPANIER, E. H. *Algebraic Topology*. McGraw-Hill, New York, 1966.

[25] SWELDENS, W. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal. 29*, 2 (1997), 511–546.

[26] TAUBIN, G. A Signal Processing Approach to Fair Surface Design. In *Computer Graphics (SIGGRAPH' 95 Proceedings)*, 351–358, 1995.

[27] TAUBIN, G., ZHANG, T., AND GOLUB, G. Optimal Surface Smoothing as Filter Design. Tech. Rep. 90237, IBM T.J. Watson Research, March 1996.

[28] WELCH, W., AND WITKIN, A. Free–Form Shape Design Using Triangulated Surfaces. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, 247–256, July 1994.

[29] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive Multiresolution Mesh Editing. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 259–268, 1997.