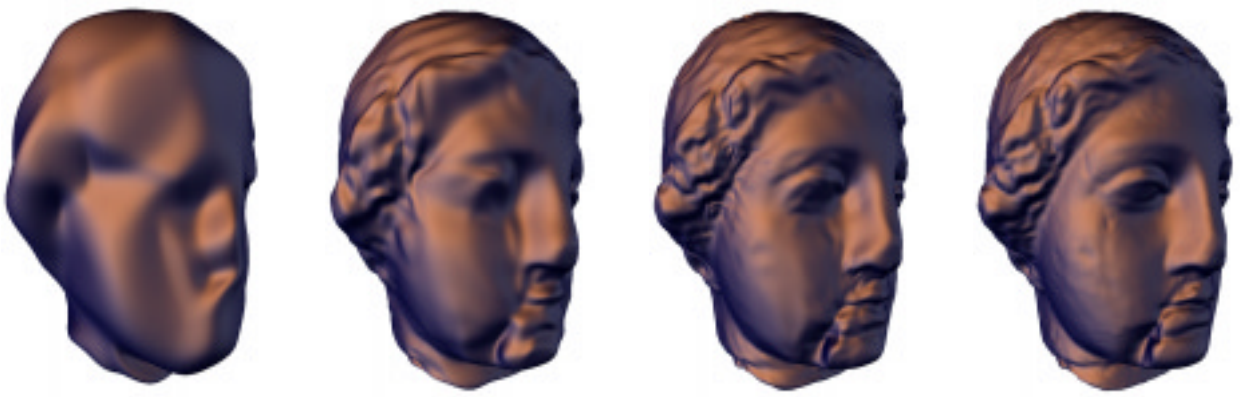
Progressive Geometry Compression

Andrei Khodakovsky
Caltech

Peter Schröder
Caltech

Wim Sweldens
Bell Laboratories



476B (e:40) 1528B (e: 12) 4163B (e: 4.7) 26800B (e: 0.82)

Figure 1: *Partial bit-stream reconstructions from a progressive encoding of the Venus head model. File sizes are given in bytes and relative L^2 reconstruction error in multiples of 10^{-4}. The rightmost reconstruction is indistinguishable from the original.*

Abstract

We propose a new progressive compression scheme for arbitrary topology, highly detailed and densely sampled meshes arising from geometry scanning. We observe that meshes consist of three distinct components: geometry, parameter, and connectivity information. The latter two do not contribute to the reduction of error in a compression setting. Using semi-regular meshes, parameter and connectivity information can be virtually eliminated. Coupled with semi-regular wavelet transforms, zerotree coding, and subdivision based reconstruction we see improvements in error by a factor four (12dB) compared to other progressive coding schemes.

CR Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling - *hierarchy and geometric transformations*; G.1.2 [**Numerical Analysis**]: Approximation - *approximation of surfaces and contours, wavelets and fractals*; I.4.2 [**Image Processing and Computer Vision**]: Compression (Coding) - *Approximate methods*

Additional Keywords: Compression algorithms, signal processing, wavelets, subdivision surfaces, semi-regular meshes, zerotree coding, hierarchical representations

1 Introduction

Today we can accurately acquire finely detailed, arbitrary topology surfaces with millions and most recently billions [22] of vertices. Such models place large strains on computation, storage, transmission, and display resources. Compression is essential in these settings and in particular *progressive* compression, where an early, coarse approximation can subsequently be improved through additional bits. While compression of *images* has a long history and has achieved a high level of sophistication, compression of *surfaces* is relatively new and still evolving rapidly.

Compression is always a tradeoff between accuracy and bit rate, i.e., bits per vertex. This tradeoff is the subject of classical rate-distortion theory. While rate-distortion curves are common in the image coding literature they have only recently appeared in geometry coding. This is partially due to the fact that the error for images is easily measured using the L^2 norm of the difference between original and approximation, while measuring error for surfaces is more involved. Since there is no immediate correspondence between the original and compressed surface, one cannot simply subtract one surface from another. This difficulty is typically addressed by computing a geometry error using, for example, Hausdorff distance. Such error metrics do not depend on the particular sample locations or connectivity, but instead measure the distance between the geometric shapes. This is important since the original and compressed mesh may have very different sample locations and connectivity, especially in a progressive setting. By sample location we mean the precise location of the vertex *within* the surface.

How low can such errors be? Consider a continuous physical surface, such as the Venus sculpture whose scan generated the mesh in Figure 1. Given that the source geometry is continuous, any digital representation, such as a triangle mesh, has some error E associated with it. This error has three components due to sampling, discretization, and quantization. Sampling error E_s arises from acquisition noise. Discretization error E_d is due to the fact that a triangulation with edge length h can approximate a smooth geometry no better than $O(h^2)$. Finally, a finite bit representation for the vertex positions leads to quantization error E_q. The sampling and triangulation of the model fix E_s and E_d. A standard float representation typically leads to a quantization error much smaller than $E_s + E_d$. All existing single rate coders proceed by first quantizing the vertex positions more coarsely leading to a quantization error $E'_q \approx E_s + E_d$ followed by lossless encoding of the connectivity and quantized vertex positions. Existing progressive coders aim

to eventually recover the quantized sample locations and original connectivity. For small meshes with carefully layed out connectivity and sample locations this is very appropriate. The situation is different for highly detailed, densely sampled meshes coming from 3D scanning: Since distortion is measured as geometric distance the sample locations and connectivity can be treated as additional degrees of freedom to improve the rate-distortion performance. As long as the final result has geometric error on the order of the original E, the actual sample locations and connectivity do not matter. We will call the information contained in the sample locations, the *parameter* information. For example, by letting the vertices slide *within* the surface we only change the parameter information and not the geometric fidelity.

In particular, we propose a new progressive geometry compression method which is based on smooth semi-regular meshes, i.e., meshes built by successive triangle quadrisection starting from a coarse irregular mesh. Almost all vertices in a semi-regular mesh have valence six and their sample locations can easily be estimated. Hence, semi-regular meshes allow us to eliminate almost all *parameter* and connectivity information. As we illustrate below, parameter and connectivity information make up a considerable fraction of the bit budget in existing coders, but do not contribute at all to reducing geometric error. Consequently our rate-distortion curves are significantly better than those of existing coders. For most models, our error is about four times smaller at comparable bit rates, a remarkable 12 dB improvement!

Semi-regular meshes additionally allow for wavelet transforms and zerotree coders. Zerotrees are amongst the best image coding algorithms today. Wavelets have superior decorrelation properties and allow for subdivision based reconstruction. This means that in regions where the encoder sets wavelet coefficients to zero the decoder uses subdivision to reconstruct the geometry. Hence even highly compressed surfaces are still smooth and visually pleasing. Figure 1 shows a sequence of progressive reconstructions of the compressed Venus model at different bitrates.

Goals and Contributions The main contribution of this paper is the observation that parameter information makes up a significant fraction of the bit budget while not contributing to error reduction at all. This motivates our compression algorithm based on semi-regular meshes.

As input our algorithm takes an irregular mesh describing a 2-manifold (possibly with boundary) and produces successive approximations employing semi-regular meshes with little parameter and connectivity information. The coder first produces a hierarchical approximation of the surface which is subsequently encoded with a zerotree progressive coder. Novel aspects of the algorithm include

- reducing parameter information through the use of semi-regular meshes;
- a Loop based wavelet transform for high order decorrelation and subdivision based reconstruction;
- a novel zerotree hierarchy for primal semi-regular triangle meshes of arbitrary topology.

We emphasize that our target application is the compression of densely sampled, highly detailed surfaces. Our algorithm is not effective when the input geometry is well described by a small, carefully layed out mesh. In this case progressive coding is generally questionable and non-progressive coders are more appropriate and perform exceedingly well.

1.1 Review of Related Work

Mesh Compression: Algorithms for efficient encoding of arbitrary connectivity meshes have been described both for the progressive and non-progressive setting (for an excellent overview of 3D geometry compression see [36]). Most of the early efforts concentrated on finding efficient encodings for mesh connectivity with the current state of the art at around 2-6b/v (bits per vertex) [37, 13, 35, 29, 28]. Vertex positions are dealt with by performing an initial quantization followed by predictive coding induced by the traversal order of the connectivity encoding.

In contrast to single target rate coders, progressive coders aim to code for a range of rates by allowing reconstruction of intermediate shapes using a prefix of the encoded bit stream. Such coding schemes are typically based on mesh simplification techniques. Examples include progressive meshes [26, 23, 16], independent set vertex removal strategies [4], topological surgery [34], and topological layering [1]. Connectivity bits increase to around 4-10b/v in these schemes. Prediction of vertex positions is now more naturally performed in a hierarchical fashion as induced by the associated mesh simplification. Examples include centroid predictors [34, 4] as well as higher order predictors [26]. To date, progressivity in these coders has typically been focused on connectivity encoding. Rate-distortion theory however says that coordinate values should be progressively quantized [23, 17] as well: to minimize error at a given rate one must trade off additional quantization bits for already present vertices against bits for new vertices and their connectivity.

Wavelets It is well known from image coding that wavelet representations are very effective in decorrelating the original data [8, 6], greatly facilitating subsequent entropy coding. In essence, coarser level data provides excellent predictors for finer level data, leaving only generally small prediction residuals for the coding step. For tensor product surfaces many of these ideas can be applied in a straightforward fashion [8, 33, 12]. However, the arbitrary topology surface case is much more challenging. To begin with, wavelet decompositions of general surfaces were not known until the pioneering work in [25]. These constructions were subsequently applied to progressive approximation of surfaces [2] as well as data *on* surfaces [31, 19].

Multiresolution surface representations based on subdivision [39] and local frame details are closely related to our wavelet constructions and have proven to be very powerful in a variety of circumstances. However, they require the initial surface to be represented by a semi-regular mesh. This has led to the development of a number of algorithms for remeshing [10, 20, 21, 18].

Zerotree Coders Some of the best wavelet based progressive coders are based on zerotrees [5, 32, 30]. They effectively exploit the fact that wavelet coefficients at finer scales tend to be smaller in magnitude than coefficients at coarser scales in the same region. A zerotree coder encodes the location of coefficients below threshold in subtrees. Standard zerotree coders for images are based on a dual formulation, i.e., coefficients are associated with faces. For primal hierarchical mesh decompositions using face splits (e.g., quadrisection of triangles) the data however lives at vertices, not faces. We show in Section 3.4 how to build zerotree coders for primal hierarchies.

Irregular Subdivision Our separation of parameter versus geometry information is partially inspired by the work done on irregular subdivision [14] and intrinsic curvature normal flow [7]. They point out that without the parameter side information, it is impossible to build high order schemes converging to smooth meshes. Irregular parameter information is inherently hard to encode and hinders the performance of irregular mesh coders.

2 Geometry, Parameter, and Connectivity Information

Elimination of parameter and connectivity information is a key ingredient of our algorithm. In this section we go into more detail

regarding parameter and connectivity information and how to eliminate it.

Previous compression approaches have typically treated triangle meshes as consisting of *two* distinct components: connectivity and vertex positions. State of the art coders are able to encode connectivity of irregular meshes with 2b/v or even less. Hence, it is argued, vertex positions are much more expensive and their coding needs further advancement, for example through better predictors.

The main insight of this paper is that there are actually *three* components: connectivity, geometry, and *parameter* information. The parameter information captures where the sample locations are *within* the surface while the geometry information captures the geometry *independent* of the sample locations used. So far parameter and geometry information were treated together.

Consider a vertex of a particular Venus head triangulation. Moving this vertex slightly *within* the surface, does not change the discretization error or geometry information. It only affects the parameter information. Alternatively, moving the vertex normal to the surface clearly changes the error and geometry information, but leaves parameter information unchanged. This illustrates that while geometry and parameter information are globally intertwined they disconnect locally: infinitesimally, we may think of parameter information as being described by displacements in the tangent plane to the surface. Geometry information on the other hand is normal to the surface. This implies that from a rate distortion point of view bits should be allocated preferentially to the local normal direction. For smooth parameterizations this occurs naturally since prediction residuals in the tangent plane will be small.

Sphere Example To illustrate the power of the distinction between geometry, parameter, and connectivity information we consider three triangulations of a sphere (Figure 2). All three meshes contain the same geometry information and carry the same discretization error E_d with no sampling noise. The first two meshes have semi-regular connectivity but different parameter information. The middle one was generated by jiggling the sample locations within the sphere, thereby adding significant parameter information. The rightmost has irregular connectivity and parameter information.

Figure 3 shows the respective rate-distortion curves when using the state of the art non-progressive coder of Touma and Gotsman (TG) [37]. We always show non-progressive curves dashed since these points are not achievable in a progressive manner. In case of the smooth semi-regular mesh, the TG coder correctly noticed that it contains almost no connectivity information (0.1 b/v) and almost no parameter information. Its performance is essentially limited by the quality of the predictor used. The TG coder for the non-smooth semi-regular sphere is worse illustrating the bit penalty for parameter information. The TG coder for the irregular mesh (right) illustrates the additional overhead from irregular connectivity. This example demonstrates the tremendous pay off of reducing both connectivity and parameter information in a mesh.

Finally the small curve near the y-axis shows the result of applying our coder to the smooth semi-regular mesh. It can approximate the sphere with a relative error of $5 \cdot 10^{-5}$ using 166 bytes or .5 b/v. This it not surprising since a sphere has very little geometric information and a smooth semi-regular mesh is essentially optimal for our coder. This is where the high order decorrelation and subdivision based reconstruction really pays off. The same effect we see here so pronounced for the sphere, can also be observed in smooth, regularly sampled regions of more general surfaces, see Section 4.

3 Algorithm Components

The algorithm accepts as input an arbitrary connectivity 2-manifold (with boundary) triangulation. In a first step we compute a smooth

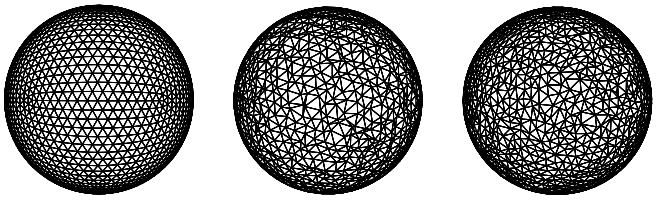


Figure 2: *Three spherical meshes each with 2562 vertices: smooth semi-regular (left), non-smooth semi-regular (middle), irregular (right). They have the same geometry information. The middle one also has parameter information while the right one has parameter and connectivity information.*

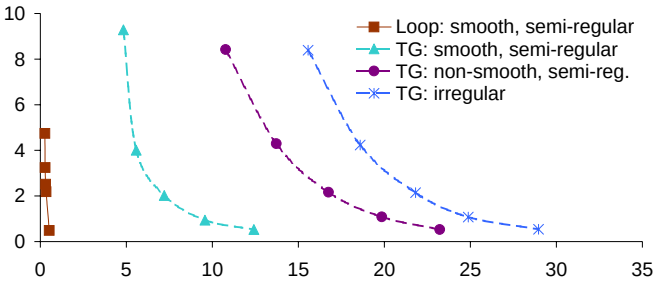


Figure 3: *Rate distortion curves for the triangle meshes from Figure 2 measured in relative L^2 error on a scale of 10^{-4} as a function of rate in b/v for TG coordinate quantization levels of $8 - 12b$.*

global parameterization using the MAPS algorithm [21]. This allows us to compute successive adaptive approximations with semi-regular connectivity. These semi-regular approximations are subsequently wavelet transformed and progressively compressed using zerotrees. The coarsest level connectivity is encoded using a standard non-progressive mesh encoder [37]. The decoder may produce intermediate approximations from any prefix of the bitstream.

We need to define the distance $d(X, Y)$ between two surfaces X and Y. Let $d(x, Y)$ be the Euclidean distance from a point x on X to the closest point on Y. Then the L^2 distance $d(X, Y)$ is given by

$$d(X,Y) = \left(\frac{1}{\text{area}(X)} \int_{x \in X} d(x,Y)^2 dx \right)^{1/2}.$$

This distance is not symmetric and we symmetrized it by taking the max of $d(X, Y)$ and $d(Y, X)$. For triangulations this distance can be computed using the METRO tool [3]. All the L^2 errors reported here are relative with respect to the bounding box diagonal on a scale of 10^{-4}, while rate is reported in b/v with respect to the number of vertices in the original input mesh.

3.1 Parameterization

As a first step, we compute a smooth parameterization of our input triangulation using MAPS [21]. An important feature of MAPS is its ability to automatically align iso-parameter lines of the semi-regular mesh with sharp features of the original input surface helping to avoid large wavelet coefficients near creases.

MAPS builds a bijective map between the input mesh T and a coarse base domain B. One can then apply quadrisection in the base domain B and use the mapping to build semi-regular approximations of T. These approximations have some remeshing error E_r with respect to T. While this error can be made arbitrarily small, it does not make sense to make the remeshing error E_r smaller than the discretization error E_d. This roughly occurs when the triangles from the semi-regular mesh are about the same size as the triangles of the input mesh. Using smaller triangles only serves to produce a better approximation of the input mesh, not necessarily of the original unknown geometry.

Of course one does not know E_d. An order estimate of E_d can be computed by measuring the distance between the input mesh T and a much finer mesh S obtained by Butterfly subdividing T. The latter serves as a proxy for the unknown original geometry. Once our semi-regular mesh error E_r is below the estimated discretization error E_d there is no need to further refine the semi-regular mesh. Hence our rate distortion curves will asymptotically not go to zero, but converge to the E_d estimate. Table 1 gives the E_d estimate, the minimum remeshing error, and the connectivity coding cost in bytes of the base domain B for various models. The connectivity was encoded using the TG coder.

	Feline	Bunny	Horse	Venus	Fandisk
# Vert.	49864	34835	48485	50002	6475
E_d (10^{-5})	7.3	9.4	6.0	5.5	28
E_r (10^{-5})	6.3	7.4	5.1	4.2	4.8
# Base Vert.	250	127	112	196	73
Base conn. (B)	122	76	62	72	46

Table 1: *Statistics for example meshes.*

3.2 Wavelet Transform

The wavelet transform replaces the original mesh with a coarsest mesh and a sequence of wavelet coefficients expressing the difference between successive levels. Since we deal with piecewise smooth models, neighboring vertices are highly correlated. The wavelet transform removes a large amount of this correlation. The distribution of wavelet coefficients is centered around zero and their magnitude decays at finer levels with the rate of decay related to the smoothness of the original surface. This behavior of the magnitude of wavelet coefficients is the key to progressive coding and justifies the choice of the zerotree coder for the bit encoding of coefficients.

Several methods for building wavelet transforms on semi-regular meshes exist [25, 31]. These are typically based on interpolating subdivision schemes such as Butterfly [9, 38]. A detailed description of the construction of lifted Butterfly wavelets can be found in [31]. The advantage of lifted wavelets is that both forward and inverse transforms can be computed with finite filters.

We use a novel Loop [24] wavelet transform, which has the advantage that the inverse transform uses Loop subdivision. Experimentally, we found it has rate distortion curves essentially identical to Butterfly, but typically better visual appearance.

The choice of Loop subdivision fixes the low pass reconstruction filter $\mathbf{P}$ in a wavelet construction. We require a high pass reconstruction filter $\mathbf{Q}$. Together they define the inverse wavelet transform

$$\mathbf{p}^{j+1} = \begin{bmatrix} \mathbf{P} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} \mathbf{p}^j \\ \mathbf{d}^j \end{bmatrix}, \qquad (1)$$

where $\mathbf{p}^j$ are the usual control points and $\mathbf{d}^j$ the wavelet coefficients at level j. For performance reasons we would like $\mathbf{Q}$ to have small support. One way to achieve this is to apply a quadrature mirror construction [27], deriving a high pass from a low pass filter. The result is shown in the regular case in Figure 4. Note that a globally consistent choice of the sign-flipping direction is possible only for orientable surfaces. Though we can use the same stencils in the general case, the wavelet subbands corresponding to edges of a certain orientation are well-defined only for orientable surfaces.

Around irregular vertices $\mathbf{P}$ is modified as usual. For edges immediately adjacent to an irregular vertex, $\mathbf{Q}$ must be modified as well. The only taps of the $\mathbf{Q}$ filter that can fall onto irregular vertices are the two -6 coefficients left and right of the center. If one of them is irregular we essentially "open up" that part of the filter and parameterize the coefficients by edge number, counting from the "10" (Figure 4, right). If an irregular vertex has valence less than six this leads to the stencil folding over on itself, while for valences larger than six a gap is left. There is currently no theory available for wavelet constructions around irregular vertices. The only justification of the "trick" we used is that it does not impact the numerically computed condition numbers of our transform. Finally, boundaries are dealt with in the usual way through reflection.

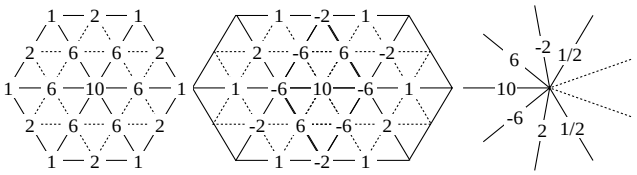


Figure 4: *Low (left) and high (middle) pass Loop reconstruction filters in the regular case. For irregular vertices the high pass filter is opened as indicated on the right.*

The forward wavelet transform, which goes from finer to coarser levels, is defined as the solution $[\mathbf{p}^j, \mathbf{d}^j]$ of the linear system in Eq. 1 for a given $\mathbf{p}^{j+1}$. Consequently computing the forward wavelet transform requires the solution of sparse linear systems. To solve these systems we use a bi-conjugate gradient solver [11] with diagonal preconditioning. We found the condition number for up to a 7 level transform to be no worse than 30 depending on the model.

Of course solving a linear system makes the forward transform slower than the inverse transform. This is acceptable as encoding is typically done once off-line while decoding happens frequently and in real time. For the Venus model the Loop forward transform, for example, takes 30s on a 550Mhz Pentium II Xeon while the inverse transform takes 2.5s. In case symmetry is important one can use a lifted Butterfly wavelet for which both forward and inverse transforms take about 2.5s.

The decorrelating power of the wavelet transform is illustrated in Figure 5. On the left is the histogram of the magnitude of Venus vertex positions. On the right is a histogram of the magnitude of the wavelet coefficients. Clearly a large amount of correlation was removed and the first order entropy has decreased considerably.

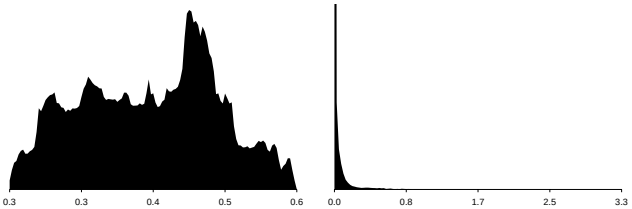


Figure 5: *Left: histogram of vertex position magnitudes for Venus. Right: histogram of the wavelet coefficient magnitudes, showing the decorrelation power of the wavelet transform.*

3.3 Vector Valued Wavelet Coefficients

Since our wavelet coefficients are vector valued, it is not immediately clear how they should be quantized. There is a fair amount of correlation between the x, y, and z wavelet components. We found that representing the wavelet coefficients in a local frame [39] induced by the surface tangent plane makes the components much more independent. In particular, we find that the variance of normal wavelet components is on average twice as large as the variance of the tangential components. Recalling the earlier geometry versus parameter distinction this is exactly what we want. In a smooth semi-regular mesh, the geometry information (normal component)

is much larger than the parameter information (tangential component). Figure 6 illustrates this by showing the histograms of the polar angles θ (the angle from the z of normal axis) of the wavelet coefficients in global and local coordinate frames. The distribution becomes very non-uniform in the local frame with peaks around 0 and π indicating that most of the wavelet vectors lie in the normal direction. The angle along the equator is fairly uniformly distributed both in the global and local frame, hence the choice of basis vectors in the tangent plane is not important. Recall that parameter,

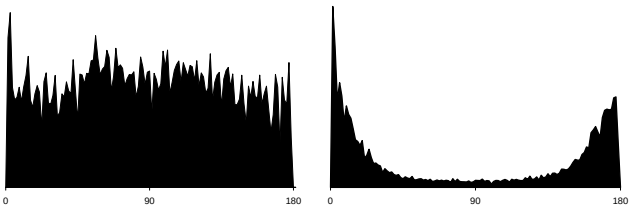


Figure 6: *Histograms of wavelet coefficient polar θ angles for the Venus head model in global (left) and local (right) frames. Coefficients lie mostly in the normal direction.*

i.e., tangential, information does not contribute to the error metric. Unfortunately, we cannot just ignore tangential wavelet components since this argument only holds in the infinitesimal limit. Especially at coarser levels, tangential wavelet coefficients can still contain some geometric information. However, we did find that the error metric is much less sensitive to quantization error of tangential versus normal wavelet components. Thus, we can further improve the error curves by more coarsely quantizing the tangential component.

A basic operation in a scalar zerotree coder is the *coefficient significance test*, i.e., checking its magnitude against a threshold. If it is below, the coefficient is added to a zerotree, else the location and sign of the coefficient need to be transmitted. For the vector case this becomes more difficult and we examined three quantization options. (1) Spherical cells are natural as we can use the magnitude for the significance test. We deal with the quantized angular components as "generalized" signs. (2) For cubical cells we divide the cube into 64 subcubes. Coefficients in the 8 internal cubes are insignificant and all the others are significant; their cell number again is an analog of the angular component. (3) We can deal with each vector component independently and encode it separately, reducing the vector case to three independent scalar passes.

We have compared all three cases and found that three scalar passes results in the best rate distortion curves for all models we considered. Experimentally, we found that quantization cells for the tangential component were best taken to be 4 times larger than those for the normal component.

3.4 Zerotree Coding

Given that we settled on scalar quantization, our coder consists of three independent zerotree coders. The bits from the three coders are interleaved to maintain progressivity.

A general principle of wavelet coefficient encoding is to send the highest order bits of the largest magnitude coefficients first. They will make the most significant contributions towards reducing error. Let $T_0 = \max\{|c_i|\}$ be the maximum magnitude of all coefficients, then in a first pass the coder should send the locations (index i) of *newly significant* coefficients, $|c_i| > T_0/2$. Doing so naïvely is expensive. However, if source and receiver agree on a canonical traversal order the source only has to send the result of the significance test $S(i) = (|c_i| > T)$ and, if true, the sign bit of c_i. If coefficients can be organized into canonical sets such that with high probability all coefficients in a given set are simultaneously below threshold, a few set-based significance tests can enumerate the locations of the relevant coefficients. The decay properties of wavelet coefficients make their hierarchical tree organization the natural set structure [32, 30, 5]. Coding consists of a number of passes with exponentially decreasing thresholds $T_{j+1} = T_j/2$. In each pass significance bits are sent for newly significant coefficients. Additionally, refinement bits are sent for those coefficients which became significant in an earlier pass. Since source and receiver already agreed on locations of the latter, no location bits have to be sent for them. The number of such bit plane passes depends on the final quantization level. The decoder can reconstruct the geometry associated with any prefix of the bitstream by running an inverse wavelet transform on the coefficient bits seen so far.

The main distinction of our setting from the image case is the construction of the zerotrees. For images, one associates the coefficients with a quadrilateral face and the trees follow immediately from the face quadtree. While this works also for dual, i.e., face based subdivision schemes, our triangular transform is primal, i.e, vertex based.

The main insight is that while scale coefficients are associated with vertices, wavelet coefficients have a one-to-one association with edges of the coarser mesh. Vertices do not have a tree structure, but the edges do. Each edge is the parent of four edges of the same orientation in the finer mesh as indicated in Figure 7. Hence, each edge of the base domain forms the root of a zerotree; it groups all the wavelet coefficients of a fixed wavelet subband from its two incident base domain triangles. The grouping is consistent for arbitrary semi-regular meshes, i.e., no coefficient is accounted for multiple times or left out.

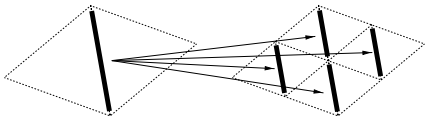


Figure 7: *A coarse edge (left) is parent to four finer edges of the same orientation (right).*

For brevity we do not give the complete coder/decoder algorithm here, but refer the interested reader to the pseudo code in [30], which is identical to our implementation with the above quadtree definition.

A final question concerns the transmission of the scale coefficients from the coarsest level. These are quantized uniformly. Experimentally, we found that it is best to send 4 bit planes initially with the base domain connectivity. Each remaining bitplane is sent as the zerotrees descend another bit plane.

The zerotree encoding (10 passes) of the Venus model takes 1s while decoding takes about 0.6s bringing the total decompression time to about 3.1s. Of course the low rate models can be decompressed faster.

3.5 Entropy Coding

The zerotree algorithm is very effective at exploiting parent-child coefficient correlations, minimizing the amount of encoded significance bits. However, the output of the zerotree coder can still be compressed further through arithmetic coding, which allows for a fractional number of bits per symbol.

The zerotree coder output contains three different types of information, significance bits, refinement bits and sign bits. Refinement and sign bits tend to be uniformly distributed; hence they are not entropy coded. Significance bits on the other hand can be further entropy coded. For early bitplanes most coefficients are insignificant resulting in mostly zero bits. For later bitplanes many coefficients become significant, resulting in mostly one bits. An arithmetic coder naturally takes advantage of this.

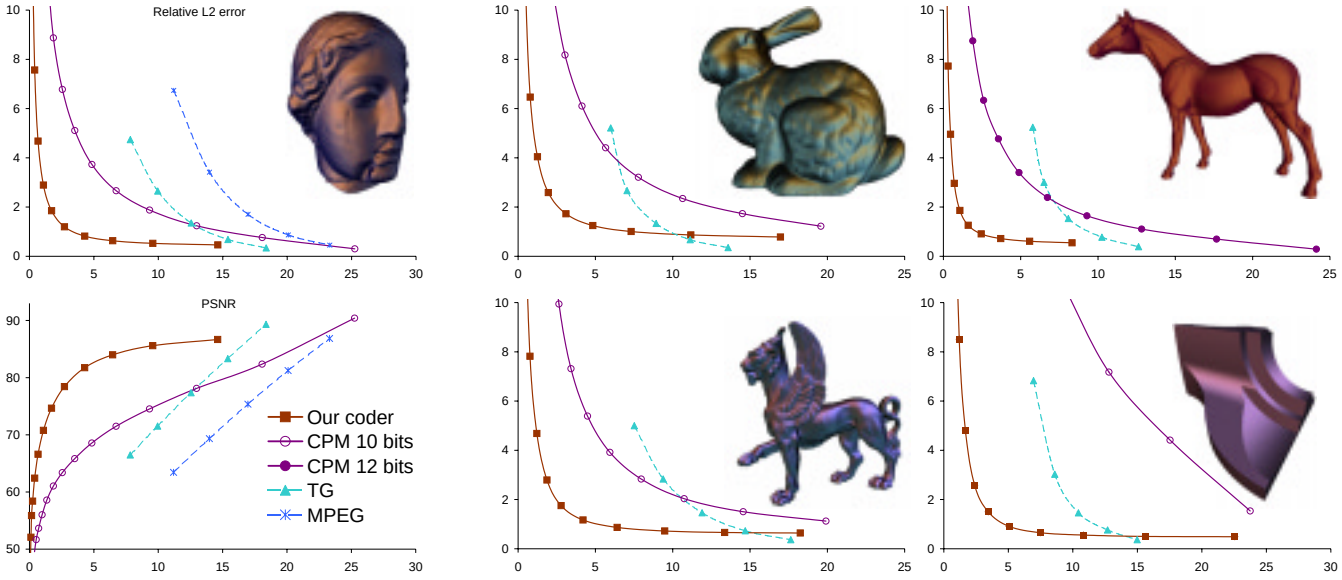


Figure 8: *Rate-distortion curves.*

We found that encoding of the significance bits in groups further improves performance of entropy coding [30]. Because children of any node always appear together during a zerotree pass we group their significance bits to form symbols of a 2^j alphabet ($j = 4, 3, 2, 1$). The actual number of bits of the alphabet is the number of children which were left insignificant at the previous pass. This grouping exploits correlations between magnitudes of spatially close wavelet coefficients.

4 Results

We compare our Loop based coder against known state of the art coders for different models. The coders we used are:

- **TG:** The Touma-Gotsman coder, which is a non progressive coder. It can be operated at different rates by changing the coordinate quantization between 8 and 12 bits.
- **CPM:** The compressed progressive mesh coder of Pajarola and Rossignac [26]. It can start with various quantization sizes. We found 10 or 12 to work best (and always show the best one).
- **MPEG:** The non-progressive coder from the MPEG4 standard which is based on topological surgery [35].

Figure 8 (left) shows the different curves for the Venus model for bitrates up to 25b/v. The top left shows relative L^2 error in units of 10^{-4}. The bottom left shows the same numbers but in a PSNR scale where PSNR $= 20 \log_{10} \text{peak}/d$, peak is the bounding box diagonal and d is the L^2 error. One can see that our progressive coder is about 12dB or a factor 4 better than the progressive CPM coder. As expected the non-progressive coders are much worse at lower rates and slightly better at higher rates. Our curve converges to the remeshing error which is where it crosses the TG curve. Given that the remeshing error is comparable to the discretization error, any compression with smaller error is only resolving a particular triangulation more accurately, but not increasing the geometric fidelity.

Figure 8 (right) shows the rate distortion curves for several additional models. Our curves are again significantly better. Typically the TG coder crosses our curve below the discretization error. For the fandisk, which is a model with creases, we used a tagged Loop transform which preserves the creases. The fandisk does not have that many triangles which is why the TG coder shows better rate-distortion performance than the CPM coder.

Figure 9 shows renderings of the different compressed versions of the model. This demonstrates the visual benefits of using subdivision based reconstruction. Note that the feline dataset has non-trivial genus (tail section), while the bunny has a number of holes on the bottom. For purposes of comparison (in the case of the Venus head) we have also rendered a number of partial bitstream reconstructions produced with the CPM coder (Figure 10) at file sizes comparable to our reconstructions (Figure 1). One could argue that the results of a more traditional progressive mesh coder could be improved by a smoothing post-process. However, even at very low bit rates, bit-plane progressivity in our coder implies that we see high order bits of significant coefficients at fine levels of the hierarchy early on. The resulting reconstructions always carry more detail than a straightforward Loop smoothing of some triangle mesh would capture. Finally Table 2 gives numerical error values for our coder at a variety of bit rates for the different models.

b/v	1/4	1/2	1	2	4	8
venus	15	6.1	3.1	1.60	0.85	0.55
feline	32	13	5.8	2.5	1.25	0.75
horse	9.7	4.5	2.0	1.05	0.70	0.55
bunny	22	10.8	5.1	2.5	1.40	0.95
fandisk		52	11.9	3.5	1.00	0.60

Table 2: *Relative L^2 error in units of 10^{-4} of our coder at various bitrates.*

5 Conclusion and Future Work

In this paper we described a progressive compression algorithm based on semi-regular meshes, wavelet transforms, and zerotree coders. Our rate distortion curves are significantly better than the best known progressive and non-progressive coders. This was achieved by explicitly treating sample locations and mesh connectivity as degrees of freedom of the coder. The progressive reconstructions especially at very low bit rates can be of astonishingly high visual quality.

There are several directions for future work:

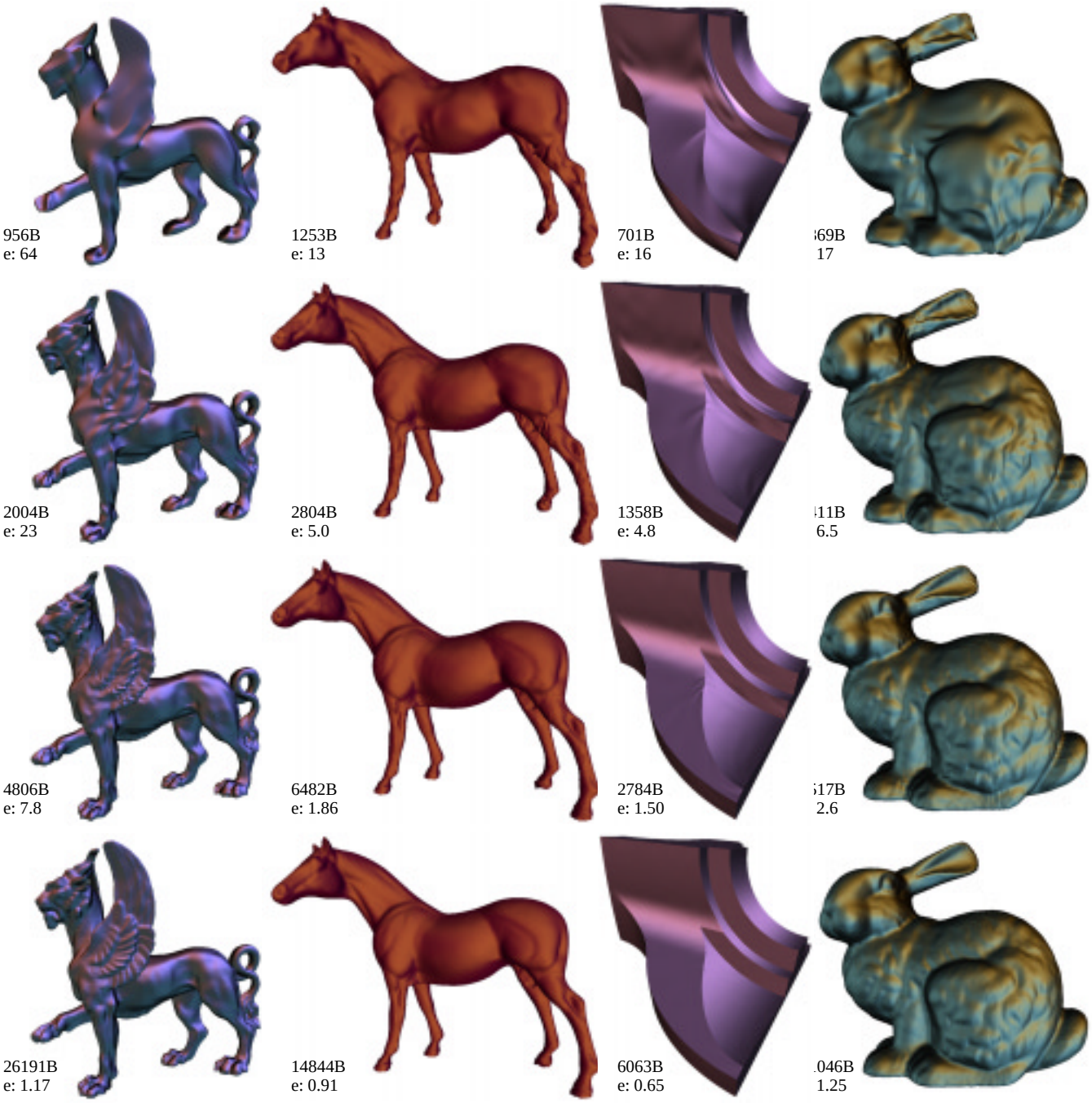


Figure 9: *File size in bytes and errors in units of* 10^{-4}.

- A mathematically sound theory for the construction of Loop wavelets around extraordinary vertices, including stability analysis.
- Construction of Loop wavelet transforms for adaptive semi-regular meshes. While all our reconstructions are performed adaptively, currently only lifted wavelets allow for adaptive analysis.
- Design of wavelet filters more suitable for geometry. Careful examination of reconstructed geometry reveals some ringing artifacts with our current wavelets.
- Even for our semi-regular meshes, there is still a fair amount of tangential information especially on the coarse levels. Recent work by Guskov et al. [15] shows that it is possible to construct *normal meshes*, i.e., meshes in which all wavelet coefficients lie exactly in the normal direction.
- The issues we discuss in this paper regarding geometry versus parameterization led to ideas such as coarsely quantizing the tangential components. These ideas can also be used to further improve irregular mesh coders.

Acknowledgments Andrei Khodakovsky was partially supported through an internship at Lucent Technologies. Other support came from NSF (ACI-9624957, ACI-9721349, DMS-9872890, DMS-9874082), Alias|Wavefront, a Packard Fellowship, and the SGI-Utah Visual Supercomputing Center. Special thanks to Cici Koenig,

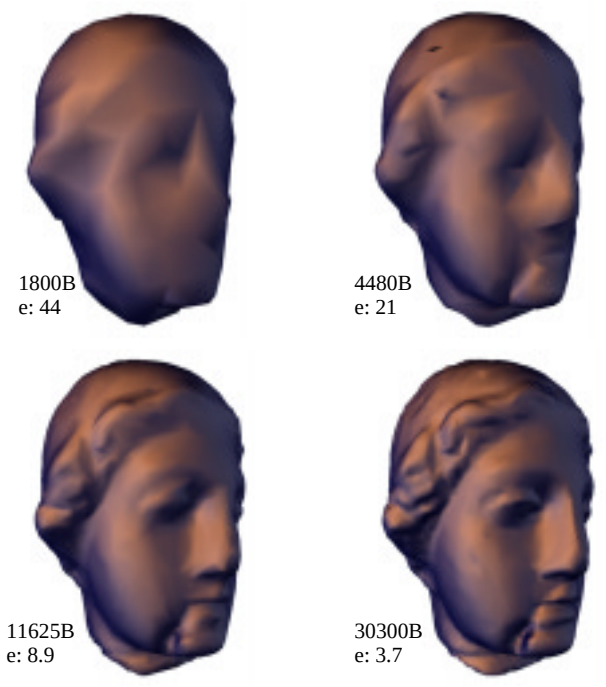


Figure 10: *Partial CPM reconstructions, error in units of* 10^{-4} *(compare to Fig. 1).*

Igor Guskov, Mathieu Desbrun, Aaron Lee, and Martin Vetterli. Datasets are courtesy Cyberware, the Stanford program in Computer Graphics and Hugues Hoppe. Our implementation uses an arithmetic coder of Geoff Davis and John Danskin. We are particularly grateful to Renato Pajarola, Craig Gotsman, and Gabriel Taubin for providing us with executables of their mesh compression algorithms.

References

[1] BAJAJ, C. L., PASCUCCI, V., AND ZHUANG, G. Progressive Compression and Transmission of Arbitrary Triangular Meshes. *IEEE Visualization '99* (1999), 307–316.

[2] CERTAIN, A., POPOVIC, J., DEROSE, T., DUCHAMP, T., SALESIN, D., AND STUETZLE, W. Interactive Multiresolution Surface Viewing. *Proceedings of SIGGRAPH 96* (1996), 91–98.

[3] CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[4] COHEN-OR, D., LEVIN, D., AND REMEZ, O. Progressive Compression of Arbitrary Triangular Meshes. *IEEE Visualization '99* (1999), 67–72.

[5] DAVIS, G., AND CHAWLA, S. Image Coding Using Optimized Significance Tree Quantization. In *Prodeedings Data Compression Conference*, 387–396, 1997.

[6] DAVIS, G., AND NOSRATINIA, A. Wavelet-based Image Coding: An Overview. *Applied Computational Control, Signals, and Circuits 1*, 1 (1998).

[7] DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. *Proceedings of SIGGRAPH 99* (1999), 317–324.

[8] DEVORE, R. A., JAWERTH, B., AND LUCIER, B. J. Surface Compression. *Computer Aided Geometric Design 9* (1992), 219–239.

[9] DYN, N., LEVIN, D., AND GREGORY, J. A. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Transactions on Graphics 9*, 2 (1990), 160–169.

[10] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution Analysis of Arbitrary Meshes. *Proceedings of SIGGRAPH 95* (1995), 173–182.

[11] GOLUB, G. H., AND LOAN, C. F. V. *Matrix Computations*, 2nd ed. The John Hopkins University Press, Baltimore, 1983.

[12] GROSS, M. H., STAADT, O. G., AND GATTI, R. Efficient Triangular Surface Approximations Using Wavelets and Quadtree Data Structures. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996).

[13] GUMHOLD, S., AND STRASSER, W. Real Time Compression of Triangle Mesh Connectivity. *Proceedings of SIGGRAPH 98* (1998), 133–140.

[14] GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. Multiresolution Signal Processing for Meshes. *Proceedings of SIGGRAPH 99* (1999), 325–334.

[15] GUSKOV, I., VIDIMCE, K., SWELDENS, W., AND SCHRÖDER, P. Normal Meshes. *Proceedings of SIGGRAPH 00* (2000).

[16] HOPPE, H. Efficient Implementation of Progressive Meshes. *Computers & Graphics 22*, 1 (1998), 27–36.

[17] KING, D., AND ROSSIGNAC, J. Optimal Bit Allocation in 3D Compression. Tech. Rep. GIT-GVU-99-07, Georgia Institute of Technology, 1999.

[18] KOBBELT, L., VORSATZ, J., LABSIK, U., AND SEIDEL, H.-P. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Computer Graphics Forum 18* (1999), 119 – 130.

[19] KOLAROV, K., AND LYNCH, W. Compression of Functions Defined on Surfaces of 3D Objects. In *Proc. of Data Compression Conference*, J. Storer and M. Cohn, Eds., 281–291, 1997.

[20] KRISHNAMURTHY, V., AND LEVOY, M. Fitting Smooth Surfaces to Dense Polygon Meshes. *Proceedings of SIGGRAPH 96* (1996), 313–324.

[21] LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. MAPS: Multiresolution Adaptive Parameterization of Surfaces. *Proceedings of SIGGRAPH 98* (1998), 95–104.

[22] LEVOY, M. The Digital Michelangelo Project. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling*, October 1999.

[23] LI, J., AND KUO, C. Progressive Coding of 3-D Graphic Models. *Proceedings of the IEEE 86*, 6 (1998), 1052–1063.

[24] LOOP, C. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[25] LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *ACM Transactions on Graphics 16*, 1 (1997), 34–73. Originally available as TR-93-10-05, October, 1993, Department of Computer Science and Engineering, University of Washington.

[26] PAJAROLA, R., AND ROSSIGNAC, J. Compressed Progressive Meshes. Tech. Rep. GIT-GVU-99-05, Georgia Institute of Technology, 1999.

[27] RIEMENSCHNEIDER, S. D., AND SHEN, Z. Wavelets and Pre-Wavelets in Low Dimensions. *J. Approx. Th. 71*, 1 (1992), 18–38.

[28] ROSSIGNAC, J. Edgebreaker: Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (1999), 47–61.

[29] ROSSIGNAC, J., AND SZYMCZAK, A. Wrap&Zip: Linear Decoding of Planar Triangle Graphs. Tech. Rep. GIT-GVU-99-08, Georgia Institute of Technology, 1999.

[30] SAID, A., AND PEARLMAN, W. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Transaction on Circuits and Systems for Video Technology 6*, 3 (1996), 243–250.

[31] SCHRÖDER, P., AND SWELDENS, W. Spherical Wavelets: Efficiently Representing Functions on the Sphere. *Proceedings of SIGGRAPH 95* (1995), 161–172.

[32] SHAPIRO, J. Embedded Image-Coding using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing 41*, 12 (1993), 3445–3462.

[33] STAADT, O. G., GROSS, M. H., AND WEBER, R. Multiresolution Compression And Reconstruction. *IEEE Visualization '97* (1997), 337–346.

[34] TAUBIN, G., GUEZIEC, A., HORN, W., AND LAZARUS, F. Progressive Forest Split Compression. *Proceedings of SIGGRAPH 98* (1998), 123–132.

[35] TAUBIN, G., AND ROSSIGNAC, J. Geometric Compression Through Topological Surgery. *ACM Transactions on Graphics 17*, 2 (1998), 84–115.

[36] TAUBIN, G., AND ROSSIGNAC, J., Eds. *3D Geometry Compression*. No. 21 in Course Notes. ACM Siggraph, 1999.

[37] TOUMA, C., AND GOTSMAN, C. Triangle Mesh Compression. *Graphics Interface '98* (1998), 26–34.

[38] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interpolating Subdivision for Meshes with Arbitrary Topology. *Proceedings of SIGGRAPH 96* (1996), 189–192.

[39] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive Multiresolution Mesh Editing. *Proceedings of SIGGRAPH 97* (1997), 259–268.