Spectral Compression of Mesh Geometry

Zachi Karni[1] Craig Gotsman[2]

Computer Science Department
Technion – Israel Institute of Technology
Haifa 32000, Israel

Abstract

We show how spectral methods may be applied to 3D mesh data to obtain compact representations. This is achieved by projecting the mesh geometry onto an orthonormal basis derived from the mesh topology. To reduce complexity, the mesh is partitioned into a number of balanced submeshes with minimal interaction, each of which are compressed independently. Our methods may be used for compression and progressive transmission of 3D content, and are shown to be vastly superior to existing methods using spatial techniques, if slight loss can be tolerated.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation.

Additional Keywords: Compression Algorithms, Signal Processing.

1. INTRODUCTION

With the advent of the Web and the increase in demand for 3D content, it is becoming very important to compress 3D mesh data for efficient transmission. The basic content of a 3D mesh dataset is the topology, i.e. the connectivity information of the mesh structure, and the geometry, i.e. the 3D coordinates of the mesh vertices. All of the works to date on mesh compression [1,3,4,5,8,10,11,13,17,18,21,22] have concentrated mostly on efficient coding of the mesh topology, and the secondary coding of the geometry is driven by this. For example, the early mesh compression schemes of Deering [5] and Chow [3] and the later scheme of Taubin and Rossignac [21] order the vertices according to the topological information, and then code them using a simple linear predictor. Similarly, the mesh compression scheme of Touma and Gotsman [22] codes the topology as a traversal of the vertices, and the vertex coordinates are coded by predicting them along this traversal using the so-called "parallelogram rule", which better captures the geometry of the mesh surface. In all cases, the prediction errors are then entropy-coded. Due to the topology coding driving the geometry coding, and not the opposite, the geometry code is not optimal. Ironically, the geometric data contains far more information than the topological data (15 bits/vertex vs. 3 bits/vertex on the average), so more effort should be invested in reducing the geometry code than the topology code, but this seems to have been neglected for the most part by contemporary mesh compression algorithms.

While the existing mesh compression algorithms are advertised as being lossless, from a pure theoretical point of view, they are actually lossy. This is because the vertex coordinates are quantized to finite precision before the actual coding. Typical 3D mesh geometry is quantized to 10-14 bits per coordinate, predictive coding subsequently reducing this to approximately half. At these quantization levels, the decoded mesh is usually visually indistinguishable from the original, justifying the use of the term "lossless". More significant loss may be introduced, and the code size reduced, by performing coarser quantization, but this results in a model with a blocky structure, which is very different from the original. Hence these algorithms are not suitable for lossy compression, due to their non-graceful degradation. This paper proposes a mesh geometry compression technique which degrades gracefully, based on spectral methods.

Many compression techniques for traditional media, such as images, employ spectral methods to achieve impressive lossy compression ratios, e.g. the popular JPEG method which relies on the discrete cosine transform. These involve expressing the data as a linear combination of a set of orthogonal basis functions, each basis function characterized by a "frequency". The underlying assumption is that a relatively good approximation may be obtained using only a small number of low-frequency basis functions. JPEG typically reduces image storage requirements by a factor of 20 relative to the raw RGB data. The next section shows how to extend classical Fourier theory to the case of 3D meshes. Subsequent sections show how to apply this to 3D mesh coding. Similarly to JPEG, we are able to obtain very significant compression ratios at the expense of a very small loss in mesh quality. For example, in many cases the code size may be reduced by a factor of 2 or 3 relative to the lossless version, with an almost unnoticeable damage to mesh appearance.

Due to the assumption that the "low frequency" coefficients contribute more to the mesh data than the "high frequency" ones, the

[1] zachi_k@cs.technion.ac.il
[2] gotsman@cs.technion.ac.il

codes generated by our algorithm may be employed in a progressive manner. For instance, a rough approximation of the model may be reconstructed using a small number of spectral coefficients, and this progressively refined by increasing the number of coefficients used in the reconstruction.

Spectral codes are useful also in other applications where precise accuracy is not important, e.g. rapid previewing and product visualization for e-commerce.

2. MESH SPECTRA

We start by showing how to extend the classical Fourier analysis to 3D mesh data. Imagine a simple graph consisting of n vertices connected in a cycle. The adjacency matrix A of this graph is the circulant n x n matrix induced by the vector [1 0 1] (where the zero coincides with the diagonal). The so-called Laplacian operator associated with A is $L=I-\frac{1}{2}A$, and is the analog of the second spatial derivative. As is well-known in matrix theory [16], the traditional cosine basis functions of the one-dimensional Fourier transform are none other than the eigenvectors of this L. The associated eigenvalues are the squared frequencies. Since the rows of L sum to zero, L is singular and has a vanishing eigenvalue, which corresponds to an eigenvector of constant values. The projection of any real n-dimensional vector on this basis vector is just the DC component (mean) of the vector.

Analogously, the Fourier basis functions for 2D signals are obtained as the eigenvectors of the Laplacian matrix of the graph with the topology of a 2D grid: $A_{ij} = 1$ for entries (i,j) such that vertex i and vertex j are neighbors on the grid and $L = I-\frac{1}{4}A$. Because of the regular structure of the 2D grid, these are just the 2D cosine functions used in JPEG.

This classical spectral theory may be extended naturally to more general graph topologies [2], and, in particular, to arbitrary 3D mesh structures [20]. If A is the adjacency matrix as defined by the n-vertex mesh topology, i.e.

$$A_{ij} = \begin{cases} 1 & i \text{ and } j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases}$$

and D is the diagonal matrix such that $D_{ii}=1/d_i$, where d_i is the degree (valence) of vertex i, then $L=I-DA$ is the mesh Laplacian:

$$L_{ij} = \begin{cases} 1 & i = j \\ -1/d_i & i \text{ and } j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases}$$

See Fig. 1 for an example. If vertex i has no neighbors, L_{ii} is set to zero. The eigenvectors of L form an orthogonal basis of R^n. The associated eigenvalues may be considered *frequencies*, and the three projections of each of the coordinate vectors of a 3D mesh geometry vector on the basis functions are the *spectrum* of the geometry. The essential observation is that geometries that are smooth relative to the mesh topology should yield spectra dominated by the low-frequency components. By "smooth relative to the mesh topology" we mean that the local geometry, as defined by topological neighborhoods in the mesh, is such that the coordinates of a vertex are very close to the average coordinates of the vertex's neighbors, hence the Laplacian operator, when applied to the mesh geometry, will yield very small absolute values. Note that there is a separate spectrum for each of the x,y and z components of the geometry, and they could behave differently, depending on the directional geometric properties (e.g. curvature) of the mesh.

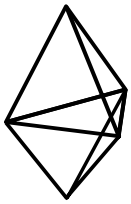
Recent years have seen an increase of interest in signal processing approaches [20,6] to 3D mesh manipulation, and their extension to multiresolution analyses [15,9]. However, to the best of our knowledge, this work is the first in which they are exploited for compression purposes.[1]

Fig. 2 shows a decimated *horse* mesh containing 2,978 vertices, a visualization of the some of the basis functions, and reconstruction of the horse using a small number of the low-frequency basis functions. Note how smooth these reconstructions are, due to the smooth nature of the low-frequency basis functions.

3. MESH PARTITIONING

Computing the spectral basis functions involves computing the eigenvectors of a n x n matrix. Ordinarily, this would require $O(n^3)$ time, which is prohibitive. However, since the Laplacian matrix is sparse (each row has only six non-zero entries on the average), this can be done in $O(n)$ time using multi-resolution methods [9,15]. Nonetheless, when n is large, the numerical stability of these methods breaks down, due to adjacent eigenvalues becoming too close. Hence it is practically impossible to compute Laplacian eigenvectors for meshes containing more than 1,000 vertices, and the mesh must be partitioned into submeshes, each of which is treated separately. This, of course, may result in a degradation in coding quality due to "edge-effects" along submesh boundaries, but has the advantage that local properties of the mesh may be captured better. In order to minimize damage, the partition should be well balanced, i.e. that each submesh contain approximately the same number of vertices, and also the number of edges straddling the different submeshes, the *edgecut*, be minimized. A optimal solution to this problem is NP-Complete [7], and algorithms approximating the optimum are an active branch of graph algorithmic research. An algorithm that performs reasonably well on meshes of up to 100,000 vertices is MeTiS [14], for which an optimized linear-time implementation is available. It seems that METIS gives some preference to minimizing the edge-cut over balancing the partition, which is the preference in our application as well. In particular, if the mesh consists of a number of connected components, MeTiS will prefer to partition into these components, unless the balance is significantly violated. Fig. 3 shows partitions generated by MeTiS for the (non-decimated) *horse* and *bunny* models. On these models MeTiS requires less than a second to run on a 350 MHz machine.

[1] As this paper went to press, we discovered that [12] addresses this issue.



(a)

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 1.00 & -0.25 & -0.25 & -0.25 & -0.25 \\ -0.25 & 1.00 & -0.25 & -0.25 & -0.25 \\ -0.25 & -0.25 & 1.00 & -0.25 & -0.25 \\ -0.33 & -0.33 & -0.33 & 1.00 & 0 \\ -0.33 & -0.33 & -0.33 & 0 & 1.00 \end{bmatrix}$$

(c)

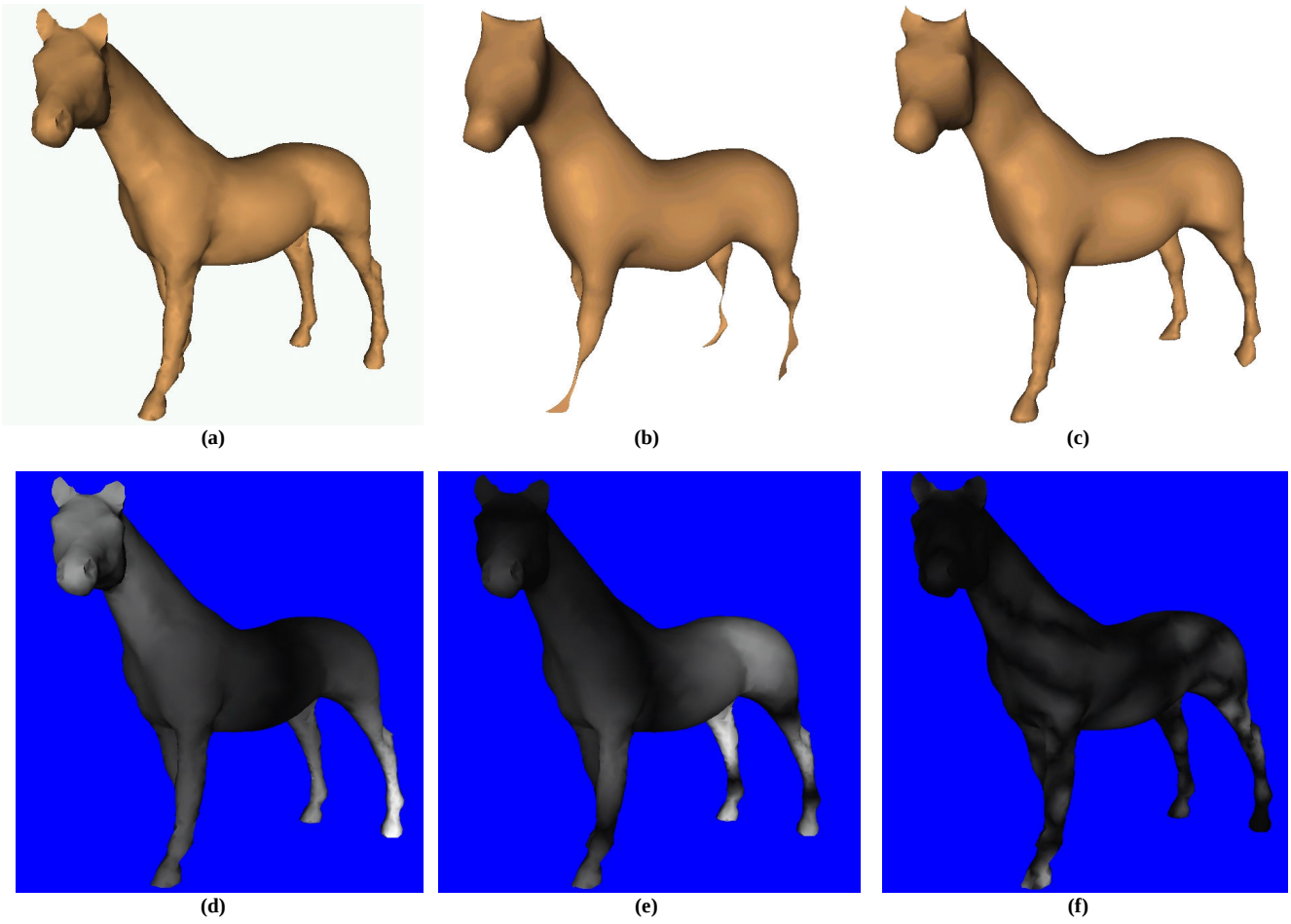
$$\begin{bmatrix} -0.447 & 0.000 & -0.817 & 0.000 & 0.366 \\ -0.447 & 0.000 & 0.408 & -0.707 & 0.366 \\ -0.447 & 0.000 & 0.408 & 0.707 & 0.366 \\ -0.447 & 0.707 & 0.000 & 0.000 & -0.548 \\ -0.447 & -0.707 & 0.000 & 0.000 & -0.548 \end{bmatrix}$$

(d)

$$\begin{bmatrix} 0 & 1 & 1.25 & 1.25 & 1.51 \end{bmatrix}$$

(e)

Figure 1: Spectral analysis of a simple 3D mesh containing 5 vertices. **(a)** Mesh. **(b)** Adjacency matrix A. **(c)** Laplacian L. **(d)** Laplacian (column) eigenvectors and **(e)** eigenvalues. Note that the first eigenvector has constant (DC) values and a vanishing eigenvalue.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 2: Approximation of the decimated *horse* model containing 2,978 vertices. **(a)** The original. **(b)** Reconstruction of the horse using 100 of the 2,978 basis functions. **(c)** Reconstruction of the horse using 200 basis functions. Note that both are smooth, and how the fine details gradually appear as more basis functions are used. **(d)** Second basis function. Eigenvalue = 4.9×10^{-4}. The grayscale intensity of a vertex is proportional to the scalar value of the basis function at that coordinate. (e) Tenth basis function. Eigenvalue = 6.5×10^{-2}. **(f)** Hundredth basis function. Eigenvalue = 1.2×10^{-1}. Note how higher frequencies (color bands) appear in the higher-order basis functions.

4. SPECTRAL CODING

4.1 A Visual Metric

In order to measure the loss resulting from a non-perfect reconstruction, a metric is required which captures well the visual difference between the original M^1 and its approximation M^2. The simplest measure is just the RMS geometric distance between corresponding vertices in both models. While this does give some indication of geometric closeness, it does not capture the more subtle visual properties the human eye appreciates, such as smoothness. This may be captured by a Laplacian operator, which takes into account both the topology and geometry. The value of this geometric Laplacian at vertex v_i is

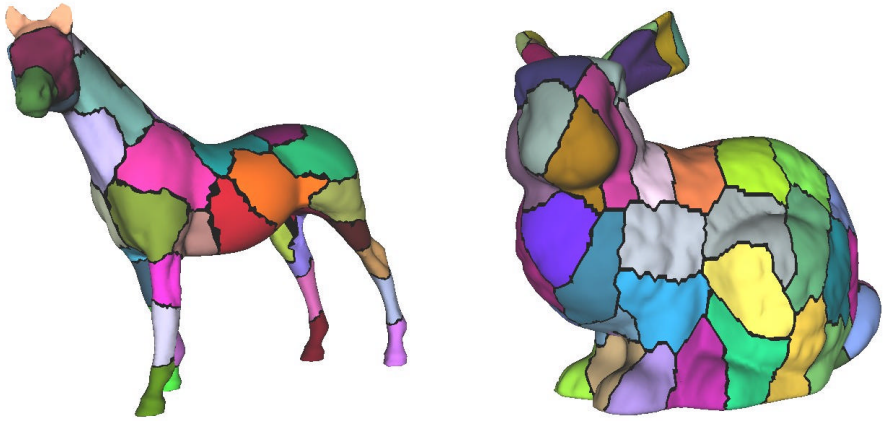
$$GL(v_i) = v_i - \frac{\sum_{j \in n(i)} l_{ij}^{-1} v_j}{\sum_{j \in n(i)} l_{ij}^{-1}},$$

where $n(i)$ is the set of indices of the neighbors of vertex i, and l_{ij} is the geometric distance between vertices i and j.

Hence we have chosen to use a metric which is the simple average of the norm of the geometric distance between models and the norm of the Laplacian difference (v is the vertex set of M):

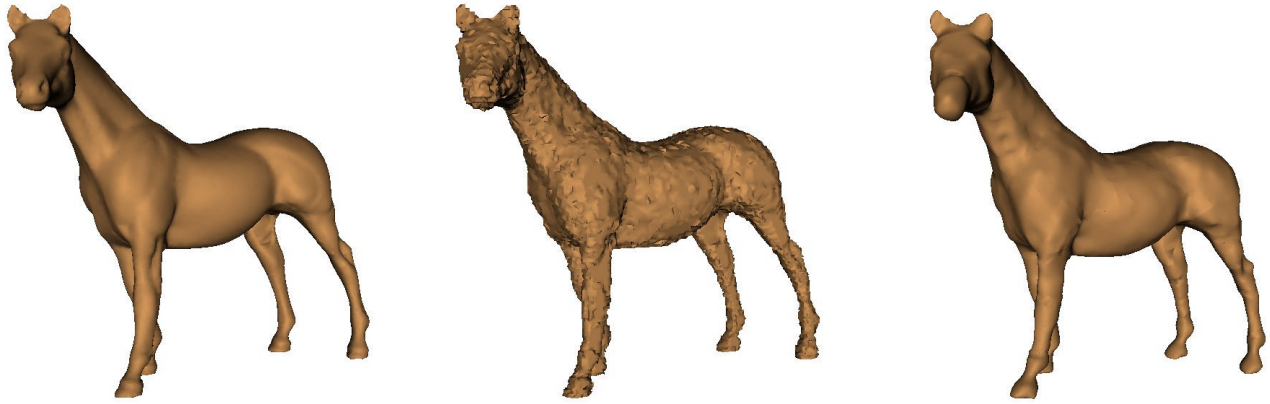
$$\left\| M^1 - M^2 \right\| = \frac{1}{2n} \left(\left\| v^1 - v^2 \right\| + \left\| GL(v^1) - GL(v^2) \right\| \right). \quad (1)$$

Fig. 4 shows the original *horse* model, and two reconstructions of it. It is clear that the second reconstruction is closer to the original than the first. A simple geometric difference does not capture this, but our visual metric does.



(a) **(b)**

Figure 3: MeTiS partitioning of meshes. Each submesh is colored in a random color. Black triangles straddle submeshes. *Edgecut* is the percentage of edges straddling submeshes. The smaller, the better. *Balance* is the ratio of the largest submesh to the average submesh size (in vertices). The closer to unity, the better. **(a)** *Horse* model: 19,851 vertices, 59,547 edges, 40 submeshes, edgecut = 4.2%, balance = 1.03. Runtime = 0.28 sec on a 350 MHz machine. **(b)** *Bunny* model: 34,834 vertices, 104,288 edges, 70 submeshes, edgecut =5.5%, balance = 1.03. Runtime = 0.47 sec on a 350 MHz machine.



(a) **(b)** **(c)**

Figure 4: A simple visual metric: **(a)** Original model M^1. **(b)** Approximation M^2 to M^1. **(c)** Approximation M^3 to M^1. It is obvious that M^3 is visually closer to M^1 than M^2, yet $\|v^1-v^2\|=\|v^1-v^3\| = 0.10$. However introducing the geometric Laplacian component of Eq. (1) yields $\|M^1- M^2\| = 0.16$ and $\|M^1- M^3\| = 0.07$, which better reflects the visual distance.

4.2 Coefficient Coding

The first step in coding the spectral coefficients is to uniformly quantize them to finite precision. This, of course, introduces some loss into the code. Typical quantization levels are between 10 and 16 bits. The second step is to truncate the coefficient vector. The resulting set of integers is then entropy coded using a Huffman or arithmetic coder [19]. It is possible to optimize the tradeoff between coefficient quantization level and the truncation by taking either a small number of high-precision coefficients, or a large number of low-precision coefficients. In practice, the input parameter to the compression procedure is the desired visual distance d from the original. The mesh is partitioned into k submeshes, such that each submesh contains approximately 500 vertices. The visual distance d is divided by $\sqrt{k}$, and the number of retained coefficients per coordinate per submesh chosen so that the RMS of the truncated coefficients does not exceed this value.

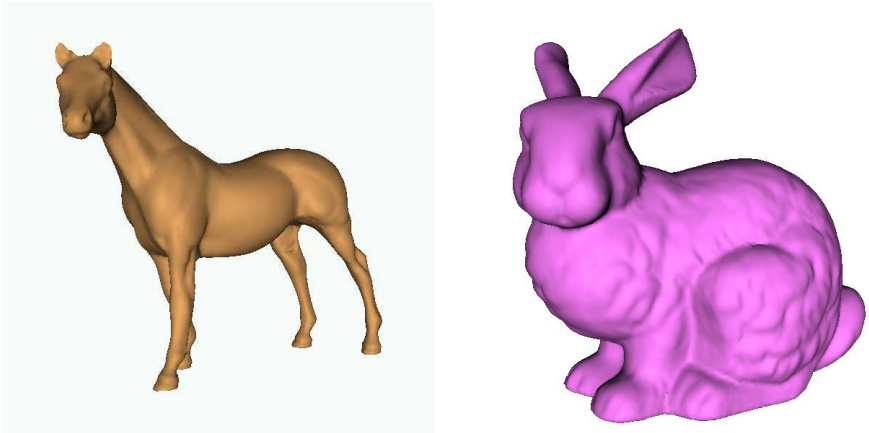
5. EXPERIMENTAL RESULTS

We present numerical results for the *horse* and *bunny* models shown in Fig. 5. These models are relatively smooth, but also have some fine details. Our compression results are compared with results obtained using commercial compression software of Virtue Ltd. (www.virtue3d.com), which incorporates the Touma-Gotsman (TG) compression algorithm [22], and is widely considered to do a very good job. This software is meant primarily for lossless compression, but can introduce loss by aggressively quantizing the geometry before coding.

Fig. 6 shows a "rate-distortion" graph for each of the models. There we plot the visual distance between the original and the decoded version of the model, as described in Eq. (1), vs. the code size. The main result is that in the lossy domain, we are able to achieve codes of approximately half to a third the size of those of the TG method for comparable visual distance. In the lossless domain, the code lengths converge to comparable values. Fig. 7 shows lossy versions of the *horse* and *bunny* models reconstructed by our algorithm, and models reconstructed by the TG algorithm, when compressed with significant quantization in order to achieve the same code size. The superiority of our reconstructions is obvious. Additionally, when our lossy reconstructions are losslessly compressed using the TG algorithm, the code size is still 1.6 and 2.5 times larger than ours.

We should emphasize that although we compare our results to those of the TG algorithm, it is not a fair comparison, since the TG algorithm is not progressive, and there is no immediate way to continuously increase model quality by increasing the number of code bits.

Animated GIFs showing progressive reconstruction of the horse and bunny models as the number of spectral coefficients is increased may be found at http://www.cs.technion.ac.il/~gotsman/siggraph2000/demos



(a) **(b)**

Figure 5: Models used in our experiments. **(a)** *Horse*: 19,851 vertices, 59,547 edges. **(b)** *Bunny*: 34,834 vertices, 104,288 edges.

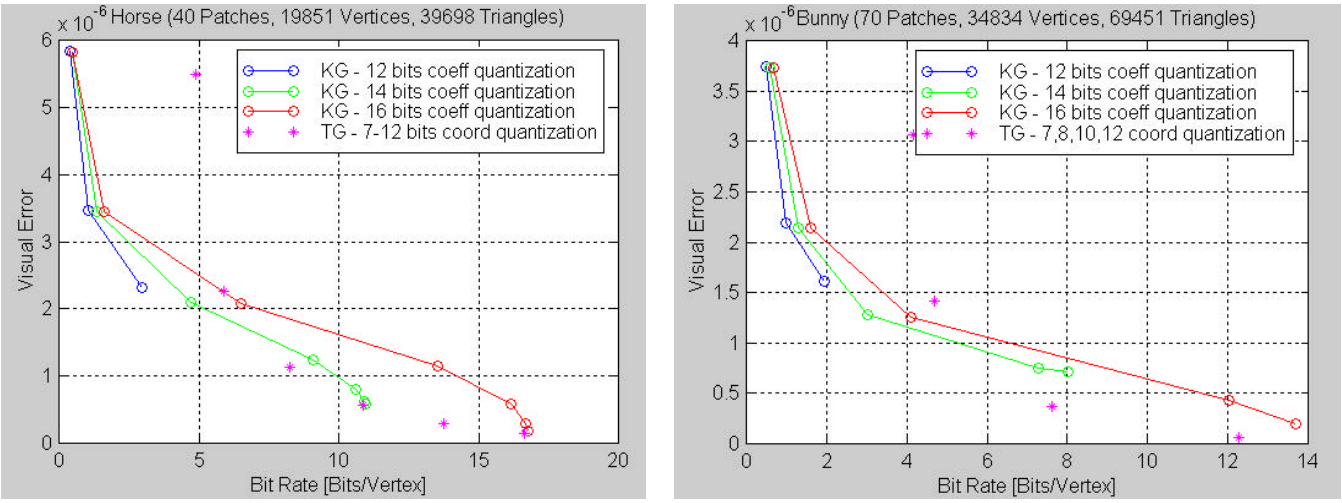


Figure 6: Geometry (only) compression performance comparison between our algorithm and the TG algorithm for the *horse* and *bunny* models. Visual losslessness is achieved at visual error of approximately 0.03 for *horse* and 0.02 for *bunny*. The best performance from our algorithm is achieved when the spectral coefficients are quantized to 14 bits.

6. DISCUSSION AND CONCLUSION

We have presented an algorithm for spectral coding of the geometric information in 3D meshes. This algorithm is similar in spirit to lossy JPEG coding of images, in the sense that it partitions the model into manageable and local submeshes, and represents each as a compact linear combination of orthogonal basis functions. The basis functions are eigenvectors of the topological Laplacian of the submesh.

In constrast to lossless coding, which is relatively well defined, lossy coding introduces a major (open) question of how to measure the loss (sometimes called *distortion*) present in the reconstructed signal. If not carefully addressed, it can be extremely hard to quantify lossy coding algorithm results. We presented a simple metric which we believe captures well the visual distance between models. Although it is far from perfect, and can still be fine-tuned, we do not believe that it will be possible to design a single metric that will be agreeable to the subjective visual systems of multiple observers.

Progressive transmission of 3D meshes is easy using the spectral methods presented here. First the compact mesh topology would be transmitted, and then the spectral coefficients of the geometry streamed, the low order coefficient first. As more and more coefficients are received, they are used to obtain a better approximation to the original.

Our results are seen to be excellent for relatively smooth models. We performed a limited number of experiments also on CAD-type models, containing sharp edges and folds. Here our results were not significantly better than the TG algorithm. This is due to the very high frequencies present in the models, forcing the coding of a very large number of coefficients. More work is required to overcome these difficulties.

We believe that ultimately it will be neccesary to employ mesh partition algorithms which are geometry-dependent, cutting the mesh along sharp folds. Since this partition will no longer be based purely on the mesh topology, and as only the topology is available at the decoder before geometry decoding, it will be neccesary to include this partition information as part of the code. This will theoretically increase the code size, but, fortunately, it seems that this overhead may be minimized by polygonal group coding, such as that of Isenberg and Snoeyink [11], which requires approximately 0.2 bits/vertex.

An alternative way to perform lossy compression of 3D mesh data is through mesh simplification, where the number of mesh vertices is reduced and topology modified, explicitly reducing the information present in the data set. It is also possible to modify even the remaining vertices, further reducing information, as is done in Khodakovsky et al [12]. The relationship between compression thru simplification and our spectral methods, which preserve the number of mesh vertices and mesh topology, has yet to be investigated. On the other hand, it would be interesting to check whether new mesh simplification methods may be obtained using spectral methods.

In practice, the mesh topology is coded and decoded separately from the geometry (e.g. using any of the algorithms of [1,8,11,13,17,18,21,22]). Since some of the topology coding algorithms permute the mesh vertices at the decoder, the geometric coordinate list must be permuted accordingly before coding. The decoder also runs the deterministic mesh partitioning algorithm and eigenvector computation algorithms based only on the topology information, in order to decode the geometry.

There are tradeoffs in many places in the encoding and decoding procedures between the time and space complexities, the code size and visual loss. There is still work to be done in order to fine-tune the parameters and optimize the results. As client CPU power seems to be increasing faster than network bandwidth, we believe that even complex decoding procedures will ultimately

be tolerable, as long as the corresponding encoders produce very short codes. Nonetheless, as we have demonstrated, our results are extremely good even without significant optimizations.

Modern signal processing has embraced multiresolution methods (e.g. wavelets) as an alternative to classical Fourier theory. The main motivation is that basis functions with local support better capture the local features of the signal, hence the need for artificial signal partitioning is eliminated. Much effort has been invested in designing orthogonal multiresolution 2D basis functions, and these will be used for wavelet image coding in JPEG 2000. Designing orthogonal multiresolution basis functions for arbitrary 3D mesh topologies has so far proved to be elusive, but, when discovered, might yield results better than those presented here.

Optimization algorithms for mesh partitioning proved very useful in our compression application, and we believe that partitioning methods which achieve compact edgecuts should be useful for other 3D mesh applications, particularly efficient rendering.

Future work will include the extension of spectral coding theory to 3D mesh animation sequences.

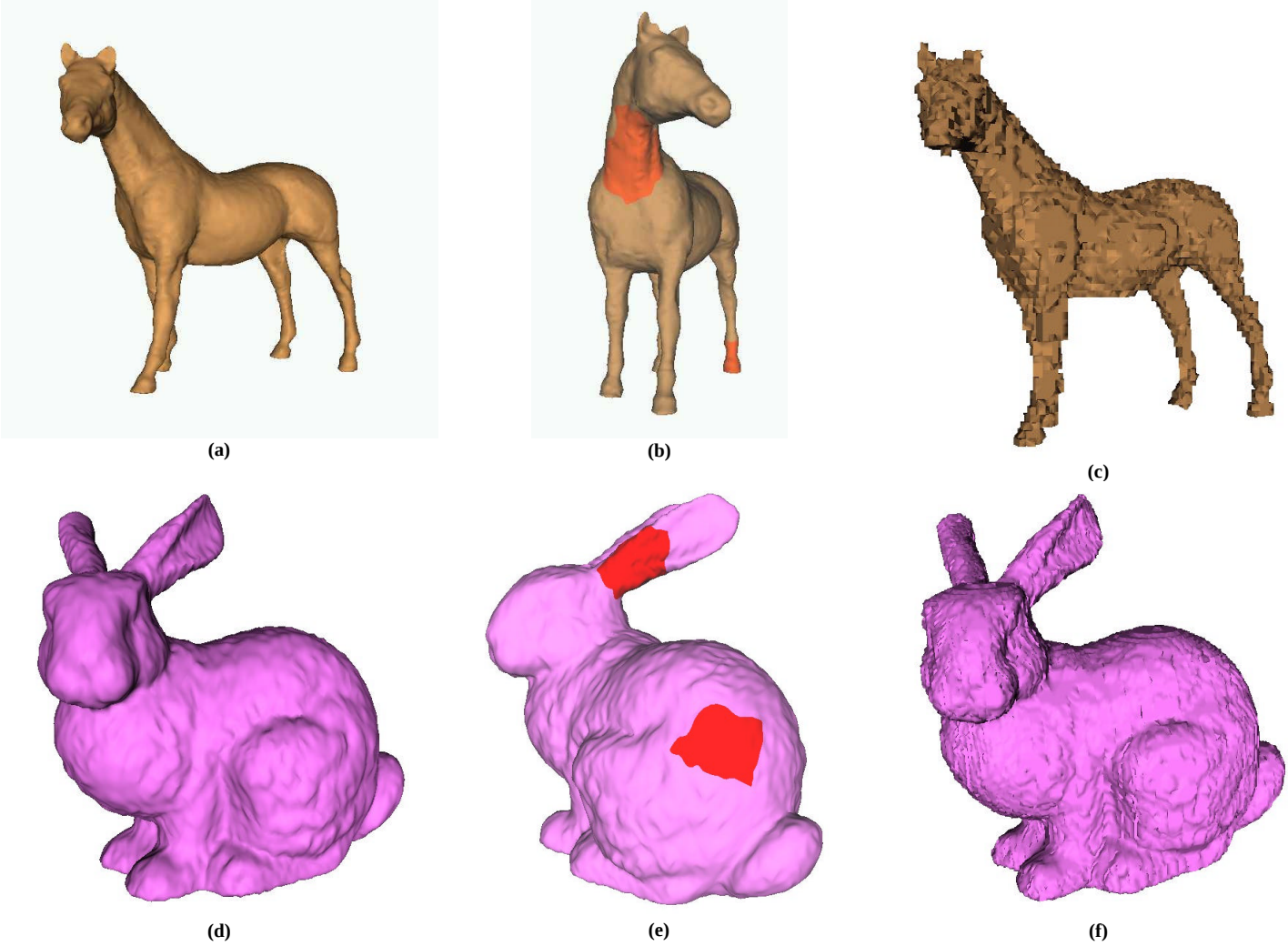


Figure 7: Some representative results of our spectral coding procedure: **(a)** Reconstructed *horse* (original in Fig. 5(a)) from code of 7,400 bytes (3.0 bits/vertex). Number of submeshes = 40, number of coefficients per coordinate per patch ranges from 66 to 221, average is 145. This horse compresses losslessly by the TG algorithm to 18,635 bytes (7.5 bits/vertex). **(b)** Same reconstruction as (a). The minimal (66 coefficients) complexity patch is in the hoof area, and the maximal (221 coefficients) complexity patch is the neck, due to the curvature. **(c)** Reconstructed horse from code of 10,100 bytes (4.0 bits/vertex) generated directly by the TG algorithm with significant geometry quantization. **(d)** Reconstructed *bunny* (original in Fig. 4(b)) from code of 17,900 bytes (4.1 bits/vertex). Number of submeshes = 70, number of coefficients per coordinate per patch ranges from 69 to 144, average is 97. This bunny compresses losslessly by the TG algorithm to 28,361 bytes (6.5 bits/vertex). **(e)** Same reconstruction as (d). The minimal (69) complexity patch is in the lower back area, and the maximal (144) complexity patch is the ear, due to the curvature and sharp edges. **(f)** Reconstructed bunny from code of 18,032 bytes (4.1 bits/vertex) generated directly by TG algorithm with significant geometry quantization.

7. ACKNOWLEDGEMENTS

Thanks to Leif Kobbelt and Gabriel Taubin for helpful discussions on the topic of this paper.

8. REFERENCES

[1] V. Bajaj, V. Pascucci and G. Zhuang. Single resolution compression of arbitrary triangular meshes with properties. Proceedings of the Data Compression Conference, Snowbird, 1999.

[2] N. Biggs. Alegbraic Graph Theory (2nd Ed.). Cambridge University Press, 1993.

[3] M. Chow. Geometry compression for real-time graphics. Proceedings of Visualization '97, IEEE, 1997.

[4] D. Cohen-Or , O. Remez, and D. Levin, Progressive compression of arbitrary triangular meshes**.** Proceedings of Visualization '99, IEEE, 1999.

[5] M. Deering, Geometry compression, Proceedings of SIGGRAPH '95, pp. 13-20, ACM, 1995.

[6] M. Desbrun, M. Meyer, P. Schroeder and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. Proceedings of SIGGRAPH '99, pp. 317-324, ACM, 1999.

[7] M. Garey and D. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Addison-Wesley, 1978.

[8] S. Gumhold and W. Strasser. Real time compression of triangle mesh connectivity. Proceedings of SIGGRAPH '98, pp. 133-140, ACM, 1998.

[9] I. Guskov, W. Sweldens and P. Schroeder. Multiresolution signal processing for meshes. Proceedings of SIGGRAPH '99, pp. 325-334, ACM, 1999.

[10] M. Isenburg and J. Snoeyink. Mesh collapse compression. Proceedings of SIBGRAPHI'99 – 12th Brazilian Symposium on Computer Graphics and Image Processing, pp.27-28, 1999.

[11] M. Isenberg and J. Snoeyink. FaceFixer: Compressing polygon meshes with properties. Proceedings of SIGGRAPH 2000.

[12] A. Khodakovsky, P. Schroeder and W. Sweldens. Progressive geometry compression. Proceedings of SIGGRAPH 2000.

[13] D. King and J. Rossignac. Guaranteed 3.67v bit encoding of planar triangle graphs. In Proceedings of 11th Canadian Conference on Computation Geometry, pp.146-149, 1999.

[14] G. Karypis and V. Kumar. MeTiS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Version 4.0, Univ. of Minnesota, Dept. of Computer Science, 1998. Available at http://www-users.cs.umn.edu/~karypis/metis/metis.html

[15] L. Kobbelt, S. Campanga, J. Vorsatz and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. Proceedings of SIGGRAPH '98, pp. 105-114, ACM, 1998.

[16] P. Lancaster and M. Tismenetsky. The theory of matrices. (2nd Ed.), Academic Press, 1985.

[17] J. Li and C.-C. Kuo, A dual graph approach to 3D triangular mesh compression, In Proceedings of the IEEE International Conference on Image Processing, Chicago, 1998.

[18] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics, 5(1), 1999.

[19] D. Salomon. Data compression: The complete reference. Springer Verlag, 1998.

[20] G. Taubin. A signal processing approach to fair surface design. Proceedings of SIGGRAPH '95, pp. 351-358, ACM, 1995.

[21] G. Taubin and J. Rossignac. Geometric compression through topological surgery. ACM Transactions on Graphics, 17(2):84-115, 1998.

[22] C. Touma and C. Gotsman. Triangle mesh compression. In Proceedings of Graphics Interface '98, pp. 26-34, 1998.