3D Mesh Compression Using Fixed Spectral Bases

Zachi Karni Craig Gotsman

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

{zachik|gotsman}@cs.technion.ac.il

Abstract

We show how to use fixed bases for efficient spectral compression of 3D meshes. In contrast with compression using variable bases, this permits efficient decoding of the mesh. The coding procedure involves efficient mesh augmentation and generation of a neighborhood-preserving mapping between the vertices of a 3D mesh with arbitrary connectivity and those of a 6-regular mesh.

Keywords: Computer Graphics, Mesh Compression, Spectral Decomposition.

1. Introduction

Due to the need to transmit 3D data sets rapidly over the Internet, 3D mesh compression is an important and active research topic. Since most of the 3D content in use are polygonal meshes, most of the published work concentrates on coding that type of data, consisting of two main components: mesh connectivity and mesh geometry. Mesh connectivity has a combinatorial graph structure, so can be coded losslessly, as has been done by a variety of methods (e.g. [6,7]). Mesh geometry consists of the floating-point coordinates of the mesh vertices in 3D space, so may incur loss when coded. The simplest way of implementing this is to uniformly quantize each coordinate to an integer grid, and then losslessly code the result. Hence the sole loss incurred is due to the quantization.

The mesh geometry component dominates the connectivity component in terms of information content, so any effort made to make it more compact is justified. The conventional way to compress mesh geometry is by spatial prediction methods, for example, by predicting the coordinates of a vertex from the coordinates of neighboring vertices. A particularly effective method is the *parallelogram* predictor [7], which operates under the (quite correct) assumption that adjacent triangles in a mesh tend to form a parallelogram. This is a local method, which captures reasonably well the correlations present in the mesh geometry. Other local methods [4] predict the vertex coordinates by finding other sorts of patterns in the mesh geometry.

Karni and Gotsman [2] proposed the use of spectral methods for compressing 3D mesh geometry. In a nutshell, this involves projecting the mesh geometry vector(s) onto basis vectors, which are the eigenvectors of the mesh Laplacian matrix, derived from the mesh connectivity as follows: If A is the connectivity (adjacency) matrix:

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

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:

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

Smooth meshes, i.e. those whose vertex coordinates are close to the average of their neighbors, concentrate most of their energy on the vectors with small eigenvalues (analogous to low "frequency" Fourier basis functions) when expanded on this basis. Karni and Gotsman show that this method yields a progressive compression technique providing a good tradeoff between bit rate and approximation error. However, despite the encouraging results, the method suffers from a number of drawbacks which limit the practicality of the method. One of these drawbacks is that the Laplacian matrix is different for each mesh connectivity, hence the basis must be computed separately for each mesh. Computing the eigenvalues of a *nxn* matrix, even a sparse one such as the Laplacian, requires $O(n^3)$ operations, which is prohibitive for large n's. This problem is amplified by the need to compute these eigenvectors *also at the decoder*, meaning that efficient (real-time) decoding using this method is probably impossible.

This paper explores the possibility of using fixed basis functions, independent of the mesh connectivity, for spectral mesh coding. While these basis functions will

obviously not be optimal, the hope is that on the average, most of the mesh energy will still be concentrated on the low frequency basis functions. The fixed basis functions that we propose are those derived from a 2D 6-regular connectivity. The eigenvectors associated with this connectivity are identical to those associated with the traditional 4-regular connected (grid) mesh, which are none other than the discrete Fourier basis functions. The spectrum (set of eigenvalues) is different, however, between the two connectivities. The fact that the eigenvectors are the Fourier basis functions means that the encoding and decoding procedures may be efficiently implemented using the FFT. This paper shows that even though these basis functions are not precisely what the true mesh connectivity prescribes, they are still good enough for the purpose of mesh compression.

The remainder of this paper is organized as follows: Section 2 describes the technique we use to map a mesh with arbitrary connectivity to a 6-regular mesh in order to encode it and Section 3 describes the decoder. Section 4 presents the experimental results we obtained with these methods. We conclude in Section 5.

2. The Encoder

The main problem that arises when using bases derived from regular grids is the mapping of the vertices of the given mesh to the vertices of the regular mesh. This mapping problem is known in general as *embedding* a *candidate* mesh into a *host* mesh. Our host is the 6-regular mesh – the mesh in which each interior vertex has exactly 6 neighbors. The problem is further complicated if the number of boundary and interior vertices in the candidate is different from that of the host. In order for it to be useful in our application, the embedding e should be such that it preserves as much as possible the neighborhood relationships of the mesh, namely, if u and v are vertices at (topological) distance d in the candidate, $e(u)$ and $e(v)$ should also be at a distance as close to d as possible in the host. A close relative of the graph embedding problem is the *geometric* embedding problem [1], where each vertex of the candidate is assigned a location in R^m. The embedding is considered good if the Euclidean distances between the locations of the vertices in R^m is similar to the topological distances between the same vertices in the candidate.

Let us first consider embedding a given n-vertex mesh M with connectivity graph G(M) in the n-vertex 6-regular triangle graph H. Not only do we assume that the two meshes have the same number of vertices, but also the same number of vertices on their boundaries ($I_M=I_H$, $B_M=B_H$, $I_M+B_M=n$). Graph embedding is known to be a difficult problem, so we reduce the problem to a geometric problem of matching two point sets in the plane. Tutte [8] proposed the following simple method of embedding ("drawing") a mesh in the plane: Map the mesh boundary vertices onto a convex shape, and position each interior vertex such that its coordinates are the centroid of its neighbor's coordinates. This involves iteratively solving a set of linear equations for the coordinates of the interior points.

We use the Tutte method to embed both M and H into the unit disk, yielding e_H:V(H)$\rightarrow R^2$ and e_M:V(M)$\rightarrow R^2$, where V(M) is the vertex set of the mesh M. Now, given these embeddings, find a mapping m:V(M)$\rightarrow$V(H) such that the sum of the Euclidean mapping distances in the plane is minimized:

$$m = \arg\min \sum_{v \varepsilon V(M)} \left\| e_M(v) - e_H(m(v)) \right\|$$

For this we use the following recursive algorithm: Partition each of the point sets e_M(V(M)) and e_M(V(H)) into two sets of equal size (as much as possible) by a (separate) vertical line in the plane. Such a partition is commonly known as a *median cut*. Now recursively map each half between themselves. The recursion terminates when the point set sizes drop below a threshold. In this case, the points are matched greedily, i.e. first the pair of closest points, then the pair of second closest, and so on until all points are matched. The median cut is chosen to be either a vertical or a horizontal partitioning, depending on which results in an aspect ratio which is closer to square. Figure 1 shows the 6-regular host mesh and a candidate embedded in the plane, and the matching resulting from running the recursive median cut algorithm. Note that at first glance it seems that this matching is far from optimal. It turns out however, that is hard to improve on this result if the objective is to minimize the *average* matching distance (as opposed to the minimal matching distance). The quality of the embedding may be visualized by drawing the candidate mesh with the connectivity of the host. The result is good if all resulting edges are short, even though they may now intersect.

For the more general case where the candidate mesh has fewer interior vertices than the host, i.e. $I_M<I_H$, and also fewer vertices along its boundary $B_M<B_H$, we must *augment* the candidate by adding these missing vertices to the boundary and the interior and prescribing their connectivity and geometry. In essence, since all interior vertices of the regular host have degree 6, we search for triangles whose sum of vertex degrees is small, and insert a vertex into that triangle, connected to its three vertices. We avoid boundary triangles (which will usually have lower vertex degrees), and of all triangles with the same sum of vertex degrees, we prefer that with less variance among the degrees. Adding a new vertex to a triangle increases the degree of the triangle vertices by one, and introduces a new vertex of degree 3 into the

mesh. The new vertex is assigned geometry (3D coordinates) which are just the average of its three neighbors coordinates. This keeps the mesh smooth. A similar procedure is followed for the boundary vertices. Figure 2 illustrates this.

After the candidate has been augmented and mapped to the fixed host, the candidate vertex geometry is projected onto the fixed spectral basis functions associated with the host. These are just the inner product of the mesh coordinate vector with each of the basis functions. If we take the host to be the 6-regular grid (including the boundary vertices, namely, use a closed toroidal mesh), these basis functions are just the 2D Fourier basis functions, and the projections may be computed (together) efficiently using the FFT. If the boundary vertices of the host have degree 4, the basis functions differ slightly from the Fourier basis. After being computed, the projection coefficients are quantized, entropy coded, and transmitted progressively from low frequency to high frequency to the decoder.

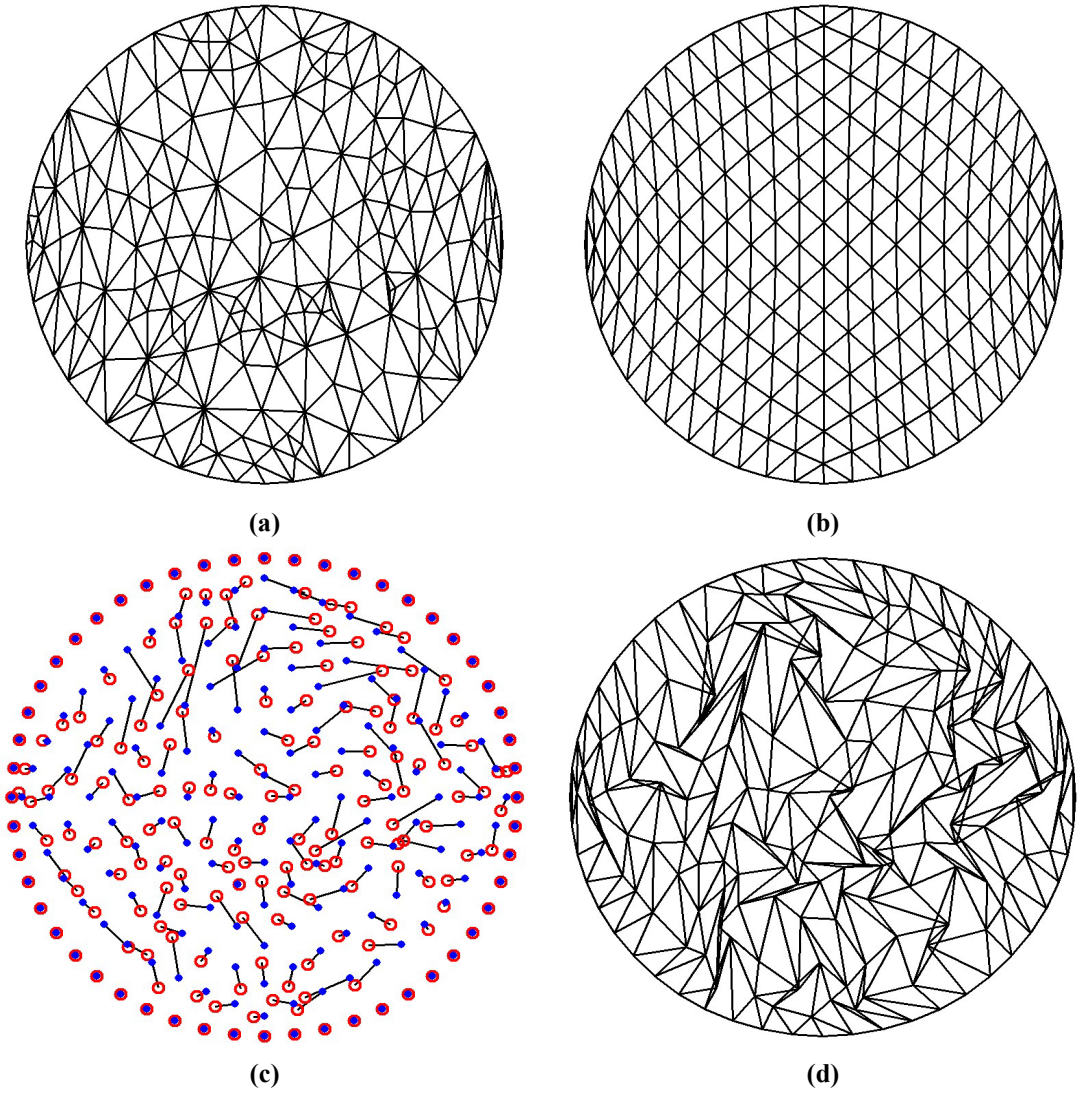

Figure 1: *Embedding a candidate mesh in a 6-regular host. Both meshes have 52 boundary vertices and 144 interior vertices.* ***(a)*** *The candidate M embedded in the planar unit disk using the Tutte method.* ***(b)*** *The host H embedded in the planar unit disk using the Tutte method.* ***(c)*** *The results of our geometric mapping procedure. Lines indicate mapped pairs. The average mapping distance is 0.08.* ***(d)*** *The embedding of M with the connectivity of H.*

3. The Decoder

The decoder receives the connectivity code separately from the geometry, which arrives in the form of coded spectral coefficients. The geometry is recovered progressively by summing the precomputed (or hardwired) host basis functions weighted by the coded coefficients as they arrive, or decoded in one shot using the FFT (if the Fourier basis is used) after the entire code has arrived. Since the encoding and decoding of the mesh connectivity may in general permute the vertex ordering relative to the original, but the encoder is aware of this and also knows the permutation, we assume the mapping of the mesh to the host was done at the encoder after the permutation was applied, hence is consistent with the connectivity generated by the decoder. To map the decoded geometry of the host vertices to the vertices of the decoded connectivity, the decoder must first determine the correct ordering of the host vertices, and then eliminate the extra dummy vertices introduced at the encoder. The first is easily achieved by running at the decoder the *same* linear time (Tutte embedding + recursive partitioning) procedures run at the encoder, and the latter by just discarding the appropriate suffix of the host vertex list after ordering.

It turns out that the visual quality of the decoded

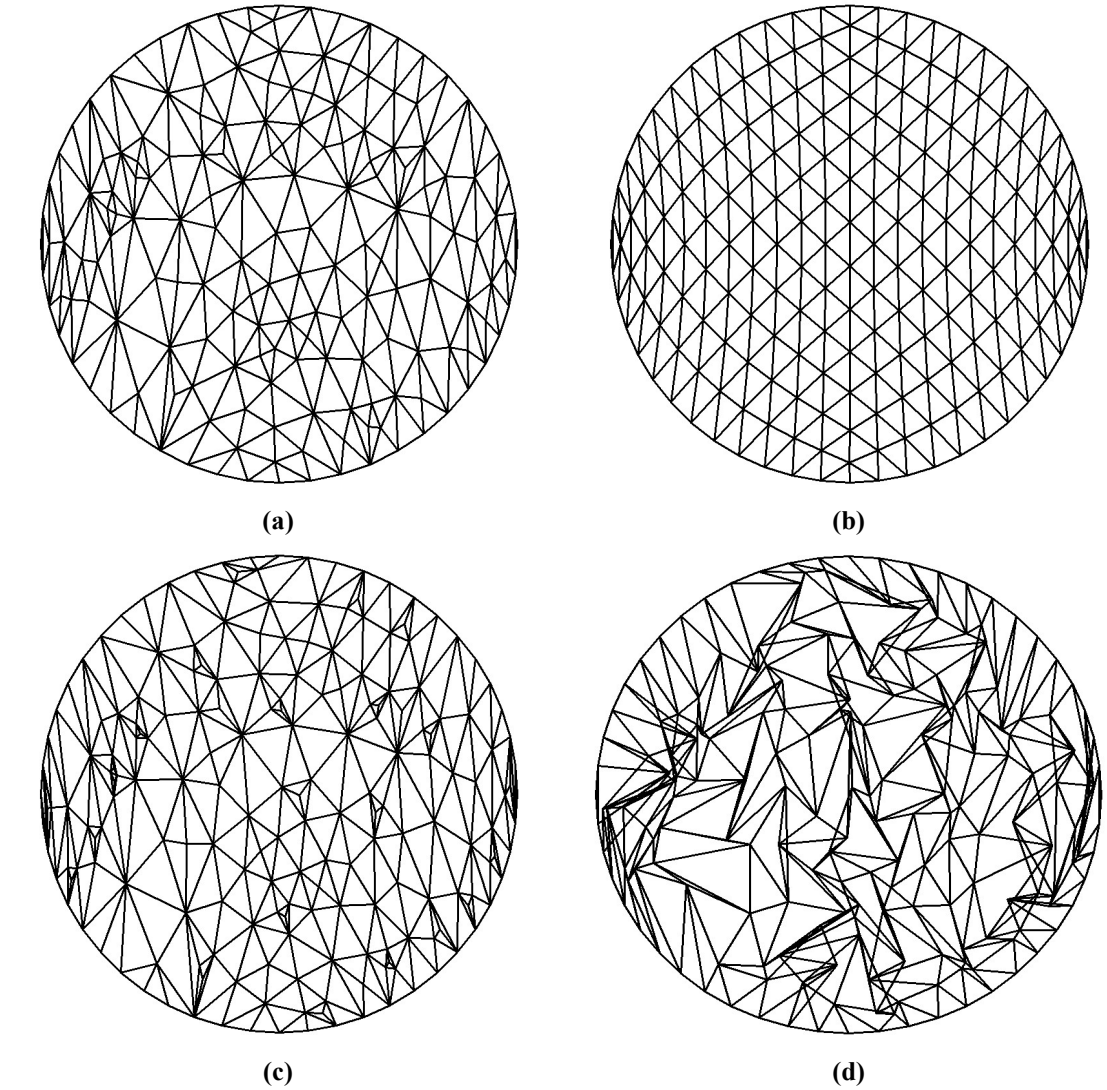

Figure 2: *Embedding a candidate mesh in a 6-regular host by augmentation.* ***(a)*** *The candidate M has 48 boundary vertices and 121 interior vertices.* ***(b)*** *The host H has 52 boundary vertices and 144 interior vertices.* ***(c)*** *The augmented candidate M'. All meshes are embedded in the unit disk using the Tutte method.* ***(d)*** *The embedding of M' with the connectivity of H. The average mapping distance resulting from the mapping procedure was 0.107.*

meshes may be somewhat improved by smoothing them using the Laplacian smoothener derived from the candidate's original connectivity (which has already been received and decoded). This reduces high frequencies relative to the optimal basis which were introduced in the transition to the fixed basis. This will be elaborated on in the next section.

4. Experimental Results

Figures 3, 4 and 5 show the results of our methods run on a single candidate patch containing 965 vertices, rendered in Figures 5a and 5b. The candidate was projected onto the fixed basis functions associated with the host - a 32x32 grid with 6-regular connectivity, and, to compare, onto the "optimal" spectral basis which are the eigenvectors of the Laplacian associated with the candidate's connectivity matrix.

Figure 3 shows various spectra related to the candidate manipulation at the encoder: the spectra when projecting the candidate and its augmented version onto the optimal bases associated with the mesh connectivities. Note that the effect of the augmentation is negligible. As is to be expected, most of the spectral energy is concentrated in the lower "frequencies". Also shown are the spectra of the augmented candidate when projected onto the fixed basis using an arbitrary mapping of the mesh vertices to the 6-regular grid vertices. Not surprisingly, high frequencies are now very significant. This contrasts with what happens when our vertex mapping algorithm is employed, where the candidate energy is forced back onto the low frequencies of the fixed basis.

Figure 4 shows the spectra of the mesh reconstructed at the decoder. The spectrum of the reconstructed mesh on the fixed basis is identical to that of the original augmented candidate, except for the truncated high frequencies, which were not transmitted. On the optimal basis, this mesh has high frequencies, which are reduced thru the Laplacian smoothing procedure. At the end, the spectrum of the reconstructed mesh, after eliminating the extraneous vertices from the augmented reconstruction, is quite similar to that of the original.

All the spectral coefficients were calculated at 16 bit precision, and the code consisted of a small number of these coefficients, which were then entropy coded. Of course, the more coefficients used, the longer the code, and the closer the reconstructed mesh to the original. Figures 5c, 5f and 5i show the reconstructions possible using 200, 400 and 800 coefficients of the optimal basis, and Figures 5d, 5g and 5j show the reconstructions possible using the same number of coefficients of the fixed basis. It is possible (and expected) that the reconstructions for the fixed basis are not as good as those of the optimal basis. This is because the spectral decomposition of the augmented candidate on the fixed basis contains non-negligible high frequencies, which are lost

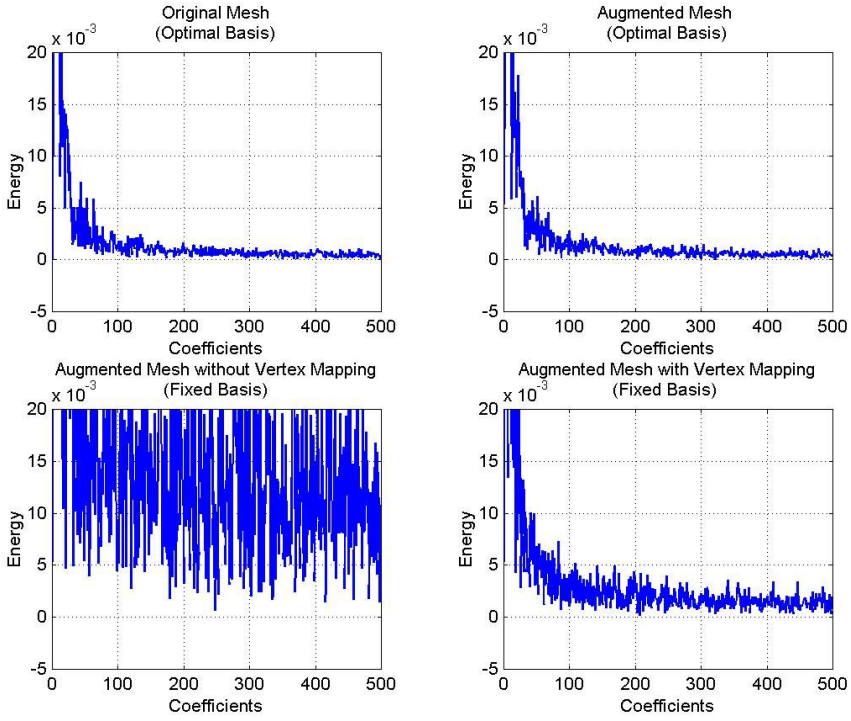

Figure 3: *Mesh spectra at the encoder.*

when the coefficient sequence is truncated, damaging the result more than in the case of the optimal basis. This is alleviated by smoothening the result using the smoothening operator associated with the candidate connectivity (which is known at the decoder). Figures 5e, 5h and 5k show that this indeed produces much better results, without paying any penalty in code length.

Figure 6 shows the compression and decompression of the "Stanford bunny" model, containing 34,834 vertices, used also in the work of Karni and Gotsman [2]. This mesh was partitioned, using the MeTiS software [3], into 36 submeshes, each coded as described above. The figure shows models reconstructed using an average of 300 and 600 coefficients per submesh. It is possible to see that at every working point, the bunny reconstructed from the compression procedure described here is almost visually identical to that reconstructed from coding using the optimal bases. This is a very small penalty to pay for the benefit of a very efficient decoding procedure, which runs in less than a second on a 800MHz Pentium PC.

5. Discussion and Conclusion

This paper has shown how to overcome one of the major obstacles to using spectral mesh compression in practical implementations – that of lengthy decoding times. This is achieved by using fixed spectral bases. While this does incur a penalty in the rate-distortion tradeoff, the reconstruction results are still visually very similar to those obtained using optimal spectral bases at given bit-rates.

There is room for improvement in the algorithmic procedures we have employed here. Our mapping algorithm is probably suboptimal, since it approximates a solution to a combinatorial problem by reduction to a geometric problem. It would be better if we could directly solve the original problem on graphs, such that a connectivity distance function is minimized. Also, the geometric approximation introduces error, since we do not optimize the matching over all possible embeddings resulting from rotating the unit disk. It also uses a very specific (Tutte) embedding, and other embedding methods might be better, possibly into the unit square instead of the unit disk. However, two others we checked [5,8] did not seem to perform any better.

The geometric mapping method might also be prone to numerical errors, which can cause inconsistencies between the encoder and decoder. This can occur when vertices are mapped to locations very close to each other in the plane, which happens when applying the Tutte embedding to a large mesh whose boundary contains very few vertices. However, in practice, our vertex sets are neither large (usually n is a couple of hundred vertices), and the boundaries quite "full", namely the number of boundary vertices is $O(\sqrt{n})$.

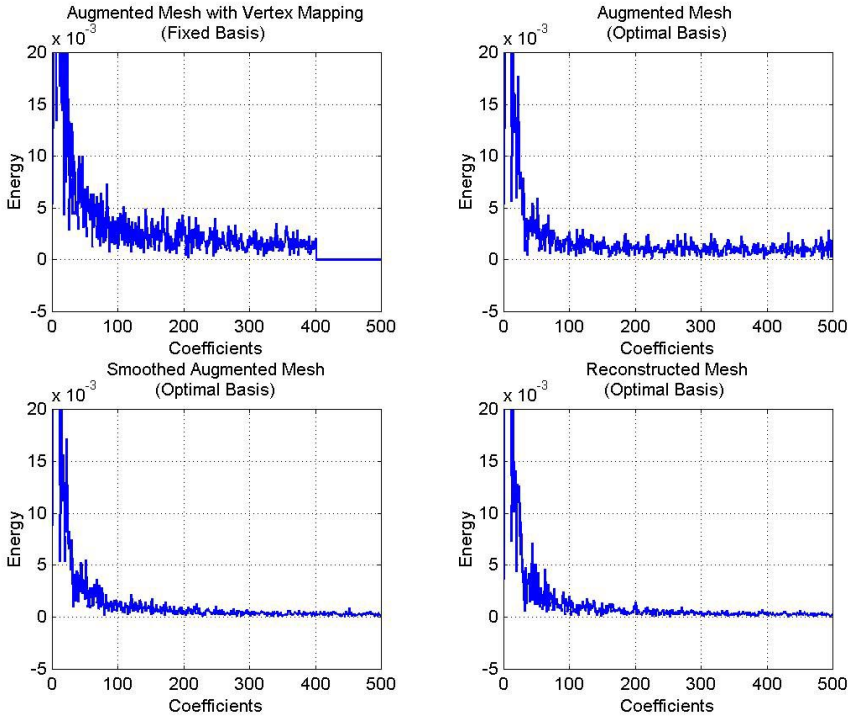

Figure 4: *Mesh spectra at the decoder, after 400 spectral coefficients were transmitted.*

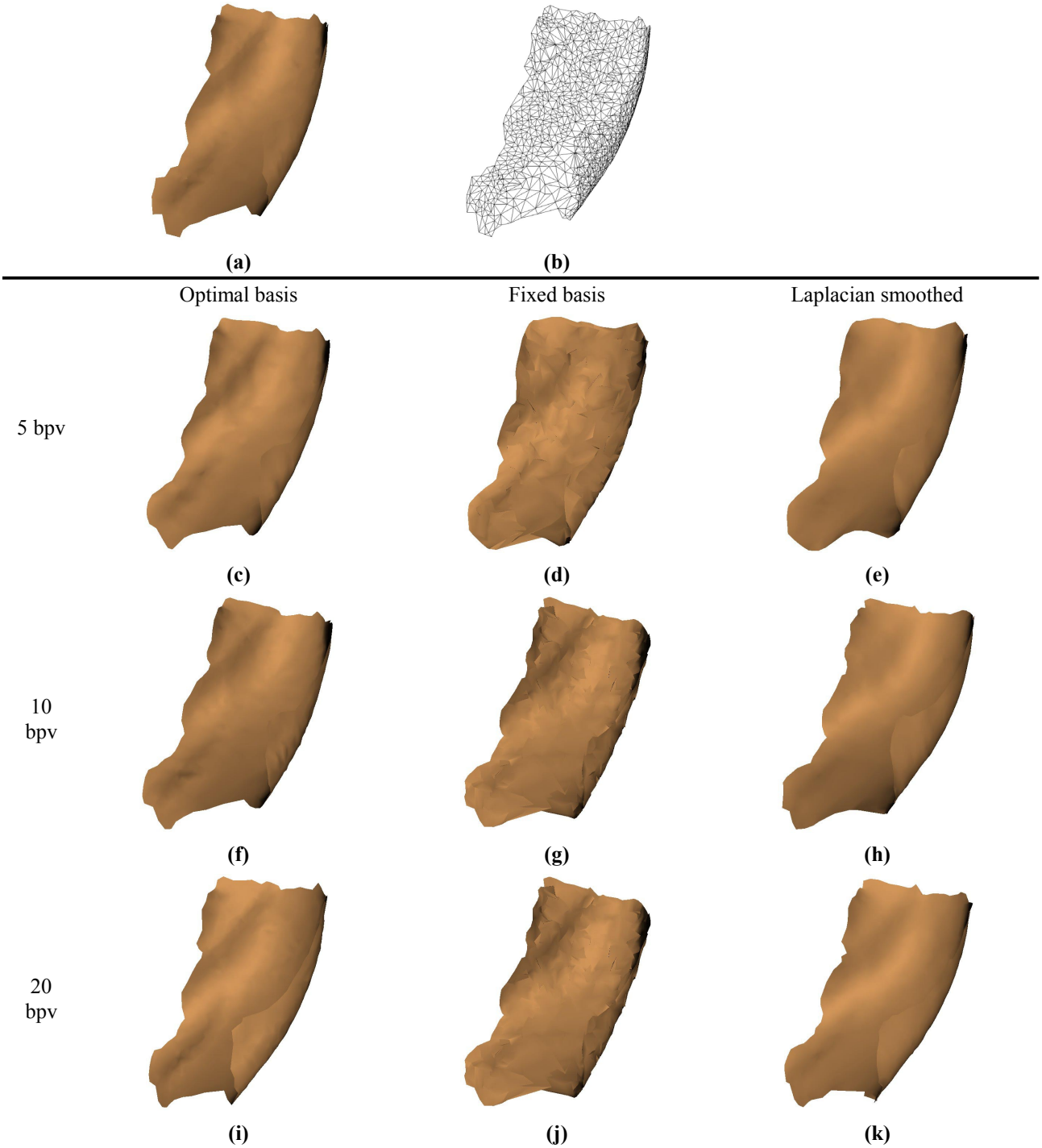

Figure 5: *Reconstructions of a candidate mesh using our methods, at different bit-rates. All spectral coefficients in all cases were quantized to 16 bits.* ***(a)*** *Original shaded mesh, containing 965 vertices, of which 97 are boundary vertices.* ***(b)*** *Original wireframe mesh.* ***(c)*** *Reconstruction using 200 spectral coefficients on optimal basis, equivalent to 5 bits/vertex after entropy coding.* ***(d)*** *Reconstruction using 200 spectral coefficients on fixed basis, equivalent to 5 bits/vertex.* ***(e)*** *Smoothed version of (d).* ***(f)*** *Reconstruction using 400 spectral coefficients on optimal basis, equivalent to 10 bits/vertex.* ***(g)*** *Reconstruction using 400 spectral coefficients on fixed basis, equivalent to 10 bits/vertex.* ***(h)*** *Smoothed version of (g).* ***(i)*** *Reconstruction using 800 spectral coefficients on optimal basis, equivalent to 20 bits/vertex.* ***(j)*** *Reconstruction using 750 spectral coefficients on fixed basis, equivalent to 20 bits/vertex.* ***(k)*** *Smoothed version of (j).*

Acknowledgements

This work was supported by European research project HPRN-CT-99-00117 (MINGLE).

References

[1] M.D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. In Proceedings of the Conference on Foundations of Computer Science, IEEE, pp. 604-609, 1989.

[2] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In Procedings of SIGGRAPH '2000, pp. 279-286, ACM, 2000.

[3] 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.

[4] E. Lee and H. Ko, Vertex data compression for triangular meshes, Proceedings of Pacific Graphics'00, pp. 225-234, ACM, 2000.

[5] V. Surazhsky. Personal communication, 2000.

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

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

[8] W.T. Tutte. How to draw a graph. Proc. London Math Society, Vol. 10, pp. 304-32, 1960.

[9] G. Zigelman, R. Kimmel and N. Kiryati. Texture mapping using surface flattening via multi-dimensional scaling. Preprint, 2000.

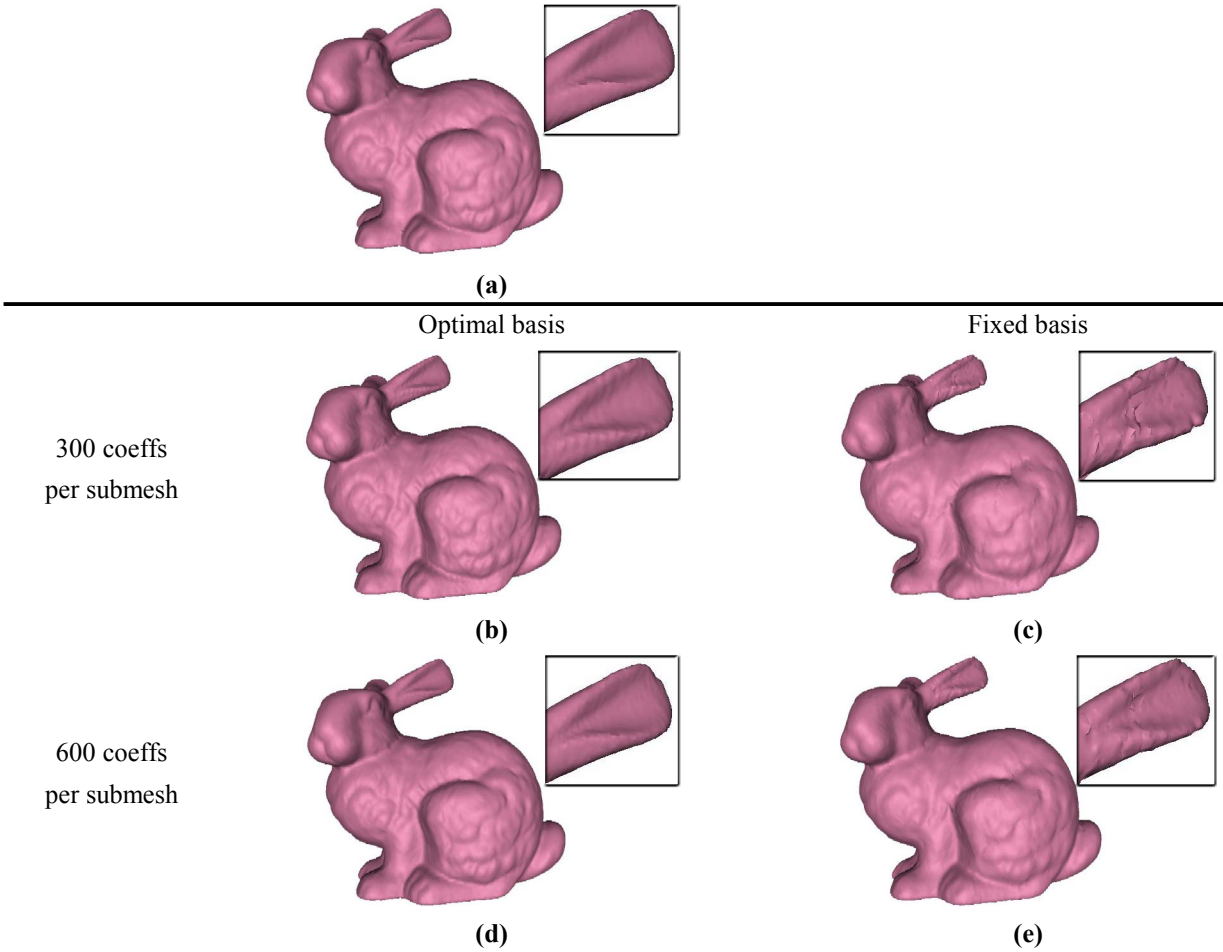

Figure 6*: Comparison of optimal basis compression with fixed basis compression on the Stanford bunny model of 34,834 vertices, partitioned into 36 submeshes. Note the differences in the area of the ears and the overall smoothness of the body.* ***(a)*** *Original mesh.* ***(b)*** *Average of 300 spectral coefficients per submesh with optimal basis.* ***(c)*** *Same as (b) with fixed basis.* ***(d)*** *Average of 600 spectral coefficients per submesh with optimal basis.* ***(e)*** *Same as (d) with fixed basis.*