Adaptive Mesh Compression in 3D Computer Graphics using Multiscale Manifold Learning

Sridhar Mahadevan MAHADEVA@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01003 USA

Abstract

This paper investigates compression of 3D objects in computer graphics using manifold learning. *Spectral compression* uses the eigenvectors of the graph Laplacian of an object's topology to adaptively compress 3D objects. 3D compression is a challenging application domain: object models can have $> 10^5$ vertices, and reliably computing the basis functions on large graphs is numerically challenging. In this paper, we introduce a novel multiscale manifold learning approach to 3D mesh compression using *diffusion wavelets*, a general extension of wavelets to graphs with arbitrary topology. Unlike the "global" nature of Laplacian bases, diffusion wavelet bases are compact, and multiscale in nature. We decompose large graphs using a fast graph partitioning method, and combine local multiscale wavelet bases computed on each subgraph. We present results showing that multiscale diffusion wavelets bases are superior to the Laplacian bases for adaptive compression of large 3D objects.

1. Introduction

JPEG compression is widely used to capture, store, and distribute images (Wallace, 1991). Formally, JPEG uses the discrete cosine transform to convert images from a "spatial" basis to a Fourier basis, where much of the image "energy" is concentrated in the low frequency eigenvectors (Amhed et al., 1974). However, DCT assumes a fixed 2D topology and cannot be directly applied to compress 3D objects in computer animation and graphics. Consequently, the problem of compression of 3D objects is of much interest in computer graphics (Taubin, 1995).

(Karni & Gotsman, 2000) introduced an *adaptive* spectral compression method, where the compression is customized to specific 3D objects by deriving basis functions from the object's known graph topology. This approach is a natural generalization of "Fourier" analysis to discrete graphs, in particular the eigenvectors of the *graph Laplacian* are essentially Fourier bases on graphs (Chung, 1997). The graph Laplacian has started to play an increasingly prominent role in machine learning in the areas of spectral clustering, nonlinear dimensionality reduction, semi-supervised learning, and basis construction in reinforcement learning, as well as applications such as segmentation in computer vision (Shi & Malik, 2000; Ng et al., 2002; Belkin & Niyogi, 2004; Meila & Shi, 2001; Mahadevan & Maggioni, 2006). The set of instances is represented by vertices of a graph, where an edge is used to connect instances x and y using a "local" distance measure, such as if y is among the k-nearest neighbors of x. The weight of the edge is specified typically using the heat kernel $e^{-\frac{\|x-y\|^2}{2\sigma^2}}$. Given such an undirected graph G, the *graph Laplacian* operator can be defined in a number of ways, including the combinatorial Laplacian $L = D - W$, where W is the weight matrix, and D is a diagonal "valency" matrix of the row-sums of W. Several recent theoretical studies have shown that the graph Laplacian asymptotically converges to the Laplace-Beltrami operator on the underlying manifold under certain conditions on the sampling distribution (Belkin & Niyogi, 2004; Singer, 2006).

However, compression of 3D objects is a challenging problem for current techniques from manifold and spectral learning. 3D objects can be very large, resulting in graphs with 10^5 or more vertices, and millions of edges. Existing algorithms for regression on graphs (Niyogi et al., 2003), as well as semi-supervised learning on graphs (Belkin & Niyogi, 2004; Zhu & Ghahramani, 2002) typically involve inversion of the graph Laplacian matrix on either the whole graph, which

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

is $O(|V|^3)$, or on the subgraph defined by the unlabeled examples, which is typically much larger than the set of labeled examples. A similar worst-case analysis holds for approaches that use the eigenvectors of the graph Laplacian as bases (Mahadevan & Maggioni, 2006). Direct application of diagonalization or inversion methods to Laplacian matrices of size 10^5 seem infeasible, even if in many cases, these matrices are quite sparse.

In this paper, we introduce a novel framework for the adaptive mesh compression of 3D objects. Our approach, first and foremost, addresses the limitations of existing Fourier methods which are based on global eigenvector representations. Classically, the limitation of Fourier bases are well-known and led to the development of wavelets (Mallat, 1989): Fourier basis vectors are *global*, they do not provide a multiresolution analysis, and poorly capture "transients" and "local discontinuities". As Figure 1 illustrates, these limitations have tangible consequences: it is hard to efficiently approximate piecewise smooth mesh geometries, such as the nonlinearities represented by "horns". We build on the framework of multiscale diffusion wavelet bases (Coifman & Maggioni, 2006; Mahadevan & Maggioni, 2006). To deal with the challenge of large graphs, we use the "divide-and-conquer" approach suggested in (Karni & Gotsman, 2000) of decomposing large graphs into a set of subgraphs, and compute local basis functions on each subgraph.

2. Spectral Mesh Compression: Fourier vs Wavelet Bases

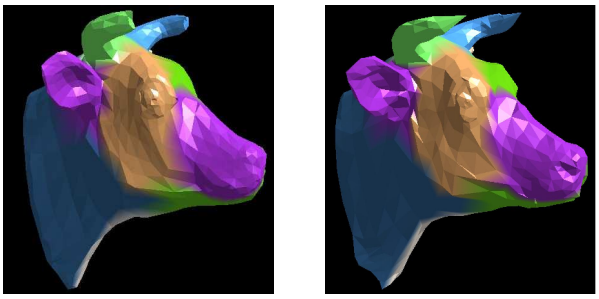

Figure 1. Spectral approximation of the mesh geometry of a 3D object using Laplacian eigenvectors (left) versus diffusion wavelet bases (right). Laplacian eigenvectors poorly reconstruct local nonlinearities represented by the "horns" or the "nose", which are rendered with much higher fidelity by diffusion wavelet bases. This object has 1107 vertices, which were partitioned into 10 subgraphs, and 20 basis functions were used to approximate the mesh geometry on each subgraph. Colors indicate partitions of the object on which both basis functions were computed.

The problem of compression of 3D objects has long been of interest in computer graphics (Taubin, 1995). (Karni & Gotsman, 2000) introduced an *adaptive* spectral compression method, where the compression is customized to specific 3D objects by deriving basis functions from the object's known graph topology. Figure 1 illustrates their approach, the picture on the left showing that Laplacian eigenvectors provide an efficient basis for compression of mesh geometry. However, they observed that their results on smooth models were significantly better than those on models "containing sharp edges and folds... due to the very high frequencies present..." (Karni & Gotsman, 2000). Figure 1 clearly illustrates these problems. In this paper, we apply a powerful new class of nonlinear function approximation techniques termed *diffusion wavelets* (Coifman & Maggioni, 2006), which generalize classical wavelets to graphs and manifolds. They are named diffusion wavelets, because they are associated with a diffusion process (similar to the graph Laplacian) that defines the different scales. As Figure 1 shows, they handily outperform Laplacian eigenvectors in mesh approximation, reconstructing local discontinuities almost perfectly even with a low number of basis functions. Figure 2 illustrates some sample multiscale diffusion bases.

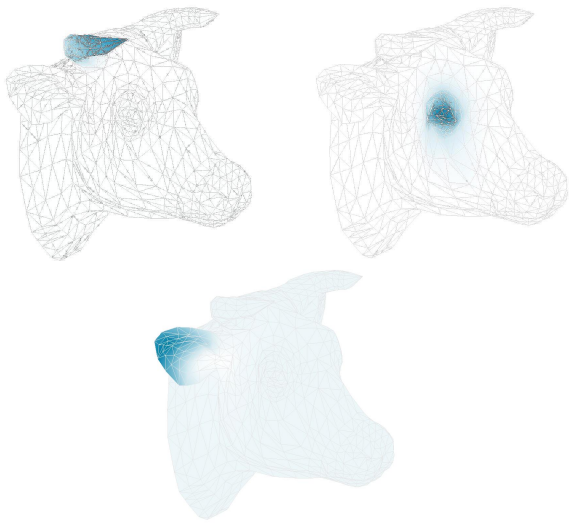

Figure 2. Diffusion wavelet bases are able to capture "semantic" regions of objects. This figure shows sample basis functions for a "cowhead" model where the basis functions are shown darkly shaded over the set of vertices from level 5 of the diffusion wavelet hierarchy: top left shows a basis function defined over the left "horn"; top right shows a basis function defined over the left "eye"; and bottom shows a basis function over the left "ear".

3. Approximation of Mesh Geometry using Laplacian Bases

The problem of mesh compression is to approximate the $3D$ coordinate functions mapping each vertex to its $3D$ position $V \to \mathcal{R}^3$. More specifically, a 3D object is specified by a graph $G = (V, E, W, M)$, where the 3D mesh coordinates $M(v) \in \mathcal{R}^3$. The weight matrix W is a set of weights on each edge $e \in E$. In our experiments below, we used binary weights so that $W(i,j) = 1$ if $(i,j) \in E$. The problem is to approximate the mesh coordinates using a set of basis functions that are not precomputed or stored, but instead adaptively learned from the topology of the graph. More precisely, let v_x, v_y, v_z be the coordinates of a vertex $v \in G$. Each of these coordinate functions can be approximated by projecting them on the subspace spanned by some orthogonal basis set, namely either a Fourier (Laplacian) or a (diffusion) wavelet basis set.

Let us define $x \sim y$ to mean an edge between x and y, and the degree of x to be $d(x) = \sum_{x \sim y} w(x,y)$. D will denote the diagonal matrix defined by $D_{xx} = d(x)$, and W the matrix defined by $W_{xy} = w(x,y) = w(y,x)$. The $\mathcal{L}^2$ norm of a function on G is $||f||_2^2 = \sum_{x \in G} |f(x)|^2 d(x)$. The gradient of a function is $\nabla f(i,j) = w(i,j)(f(i) - f(j))$ if there is an edge e connecting i to j, 0 otherwise. The *smoothness* of a function on a graph, can be measured by the Sobolev norm

$$||f||_{\mathcal{H}^2}^2 = ||f||_2^2 + ||\nabla f||_2^2 \quad (1)$$

$$= \sum_x |f(x)|^2 d(x) + \sum_{x \sim y} |f(x) - f(y)|^2 w(x,y)\,. \quad (2)$$

The first term in this norm controls the size (in terms of $\mathcal{L}^2$-norm) for the function f, and the second term controls the size of the gradient. The smaller $||f||_{\mathcal{H}^2}$, the smoother is f. We will assume that the functions we consider have small $\mathcal{H}^2$ norms, except at a few points, where the gradient may be large. Figure 1 illustrates this typical variability in smoothness, where the mesh geometry is smooth in some areas (face), but highly nonsmooth in other areas (horns).

3.1. Global Laplacian Eigenfunctions

Global basis functions can be constructed on a graph $G = (V, E, W)$ by diagonalizing the combinatorial graph Laplacian L (Chung, 1997), which is defined as

$$Lf(x) = \sum_{y \sim x} w(x,y)(f(x) - f(y)) = (D - W)f \quad (3)$$

These basis functions are of size $|V| = n$, which can be problematic if n is large. To address this, we will actually compute the Laplacian bases on subgraphs of much smaller size. In our experiments, we used the *normalized* Laplacian $\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ which has spectrum in $[0, 2]$. Th normalized Laplacian is related to the notion of smoothness as above, since $\langle f, \mathcal{L}f \rangle = \sum_x f(x)\, Lf(x) = \sum_{x,y} w(x,y)(f(x) - f(y))^2 = ||\nabla f||_2^2$, which should be compared with (2). The Spectral Theorem can be applied to $\mathcal{L}$ (or L), yielding a discrete set of eigenvalues $0 \le \lambda_0 \le \lambda_1 \le \ldots \lambda_i \le \ldots$ and a corresponding orthonormal basis of eigenfunctions $\{\xi_i\}_{i \ge 0}$, solutions to the eigenvalue problem $\mathcal{L}\xi_i = \lambda_i \xi_i$. The eigenfunctions of the Laplacian can be viewed as an orthonormal basis of global Fourier smooth functions that can be used for approximating any value function on a graph. Observe that ξ_i satisfies $||\nabla \xi_i||_2^2 = \lambda_i$. In fact, the variational characterization of eigenvectors shows that ξ_i is the normalized function orthogonal to $\xi_0, \ldots, \xi_{i-1}$ with minimal $||\nabla \xi_i||_2$. Hence the projection of a function f on S onto the top k eigenvectors of the Laplacian is the smoothest approximation to f, in the sense of the norm in $\mathcal{H}^2$. A potential drawback of Laplacian approximation is that it detects only global smoothness, and may poorly approximate a function which is not globally smooth but only piecewise smooth, or with different smoothness in different regions (as in Figure 1). Diffusion wavelets were primarily designed to address these drawbacks.

4. Beyond Eigenvectors: Multiscale Diffusion Wavelets

Diffusion wavelets generalize wavelet analysis to functions on manifolds and graphs (Coifman & Maggioni, 2006; Bremer et al., 2006; Mahadevan & Maggioni, 2006). The input to the algorithm is a "precision" parameter $\epsilon > 0$, and a weighted graph (G, E, W). The construction is based on using the natural random walk $P = D^{-1}W$ on a graph and its powers to "dilate", or "diffuse" functions on the graph, and then defining an associated coarse-graining of the graph. We symmetrize P by conjugation and take powers to obtain

$$H^t = D^{\frac{1}{2}} P^t D^{-\frac{1}{2}} = \sum_{i \ge 0} (1 - \lambda_i)^t \xi_i(\cdot)\xi_i(\cdot) \quad (4)$$

where $\{\lambda_i\}$ and $\{\xi_i\}$ are the eigenvalues and eigenfunctions of the Laplacian as above. Hence the eigenfunctions of H^t are again ξ_i and the i^{th} eigenvalue is $(1 - \lambda_i)^t$. We assume that H^1 is a sparse matrix, and that the spectrum of H^1 has rapid decay.

A diffusion wavelet tree consist of orthogonal diffusion scaling functions Φ_j that are smooth bump functions, with some oscillations, at scale roughly 2^j (measured

`DiffusionWaveletTree` $(H_0, \Phi_0, J, \epsilon)$:

// H_0: symmetric conjugate to random walk matrix, represented on the basis Φ_0
// Φ_0 : initial basis (usually Dirac's δ-function basis), one function per column
// J : number of levels to compute
// ϵ: precision

for j **from** 0 **to** J **do**,

1. Determine sparse factorization $H_j \sim_\epsilon Q_j R_j$, with Q_j orthogonal.
2. Compute the scaling function bases by setting $\Phi_{j+1} \leftarrow Q_j = H_j R_j^{-1}$ and $[H_0^{2^j}]_{\Phi_{j+1}}^{\Phi_{j+1}} \sim_{j\epsilon} H_{j+1} \leftarrow R_j R_j^*$.
3. Compute sparse factorization $I - \Phi_{j+1}\Phi_{j+1}^* = Q_j' R_j'$, with Q_j' orthogonal.
4. Compute the wavelet bases $\Psi_{j+1} \leftarrow Q_j'$.

end

Figure 3. Pseudo-code for building a Diffusion Wavelet Tree.

with respect to geodesic distance, for small j), and orthogonal wavelets Ψ_j that are smooth localized oscillatory functions at the same scale. The scaling functions Φ_j span a subspace V_j, with the property that $V_{j+1} \subseteq V_j$, and the span of Ψ_{j+1}, W_j, is the orthogonal complement of V_j into V_{j+1}. This is achieved by using the dyadic powers H^{2^j} as "dilations", to create smoother and wider (always in a geodesic sense) "bump" functions (which represent densities for the symmetrized random walk after 2^j steps), and orthogonalizing and downsampling appropriately to transform sets of "bumps" into orthonormal scaling functions.

The algorithm is described in Figure 3. We start with the basis $\Phi_0 = I$ and the matrix $H_0 := H^1$, which we assume is sparse, and construct an orthonormal basis of well-localized functions for its range (the space spanned by the columns), up to precision ϵ, through a variation of the Gram-Schmidt orthonormalization scheme, described in (Coifman & Maggioni, 2006). In matrix form, this is a sparse factorization $H_0 \sim_\epsilon \mathrm{Q_0R_0}$, with Q_0 orthonormal. Notice that H_0 is $|G| \times |G|$, but in general Q_0 is $|G| \times |G^{(1)}|$ and R_0 is $|G^{(1)}| \times |G|$, with $|G^{(1)}| \leq |G|$. In fact $|G^{(1)}|$ is approximately equal to the number of singular values of H_0 larger than ϵ. The columns of Q_0 are an orthonormal basis of scaling functions Φ_1 for the range of H_0, written as a linear combination of the initial basis Φ_0. We can now write H_0^2 on the basis Φ_1: $H_1 := [H^2]_{\Phi_1}^{\Phi_1} = Q_0^* H_0 H_0 Q_0 = R_0 R_0^*$, where we used $H_0 = H_0^*$. This is a compressed representation of H_0^2 acting on the range of H_0, and it is a $|G^{(1)}| \times |G^{(1)}|$ matrix. We proceed by induction: at scale j we have an orthonormal basis Φ_j for the rank of H^{2^j-1} up to precision $j\epsilon$, represented as a linear combination of elements in Φ_{j-1}. This basis contains $|G^{(j)}|$ functions, where $|G^{(j)}|$ is comparable with the number of eigenvalues λ_j of H_0 such that $\lambda_j^{2^j-1} \geq \epsilon$. We have the operator $H_0^{2^j}$ represented on Φ_j by a $|G^{(j)}| \times |G^{(j)}|$ matrix H_j, up to precision $j\epsilon$. We compute a sparse decomposition of $H_j \sim_\epsilon Q_j R_j$, and obtain the next basis $\Phi_{j+1} = Q_j = H_j R_j^{-1}$ and represent $H^{2^{j+1}}$ on this basis by the matrix $H_{j+1} := [H^{2^j}]_{\Phi_{j+1}}^{\Phi_{j+1}} = Q_j^* H_j H_j Q_j = R_j R_j^*$. Wavelet bases for the spaces W_j can be built analogously by factorizing $I_{V_j} - Q_{j+1} Q_{j+1}^*$, which is the orthogonal projection on the complement of V_{j+1} into V_j. The spaces can be further split to obtain wavelet packets (Bremer et al., 2006). A Fast Diffusion Wavelet Transform allows expanding in $\mathcal{O}(n)$ (where n is the number of vertices) computations any function in the wavelet, or wavelet packet, basis, and efficiently search for the most suitable basis set.

5. Scaling to Large Graphs

We now address scaling the approach of adaptive mesh compression using diffusion wavelet bases to deal with large graphs.

5.1. Graph Partitioning

A natural divide-and-conquer strategy suggested by (Karni & Gotsman, 2000) is to decompose the original graph into subgraphs, and then compute local basis functions on each subgraph. A number of graph partitioning methods are available, including spectral methods that use the low-order eigenvectors of the Laplacian to decompose graphs (Meila & Shi, 2001), as well as hybrid methods that combine spectral analysis with other techniques. We used the METIS system (Karypis & Kumar, 1998), which is a fast graph partitioning algorithm that can decompose even very graphs on the order of 10^6 vertices. METIS uses a *multiscale* approach to graph partitioning, where the original graph is "coarsened" by collapsing vertices (and their associated edges) to produce a series of smaller graphs, which are successively partitioned followed by uncoarsening steps mapping the partitions found back to the lower-level graphs.

5.2. Fast Inversion of Diffusion Matrices

Diffusion wavelets compress large powers of the diffusion operator since these often have low-rank. This property can be combined with a principle called *Schultz's expansion* to do a fast inversion of the *Green's function* $(I - T)^{-1}$, where T is a diffusion operator whose large powers have low-rank (Coifman & Maggioni, 2006; Maggioni & Mahadevan, 2006). Figure 4 contrasts the difference in time between computing the direct inverse of the Laplacian, used in the algorithms proposed in (Niyogi et al., 2003; Zhu & Ghahramani, 2002), vs. doing the inverse using the diffusion wavelet tree. The results are shown for the "pig" model illustrated in Figure 7, which has $|V| = 3820$ vertices. As the number of vertices grows (smaller number of partitions), the time required to compute the direct inverse grows much more quickly than the time required for fast inversion using the diffusion wavelet tree.[1]

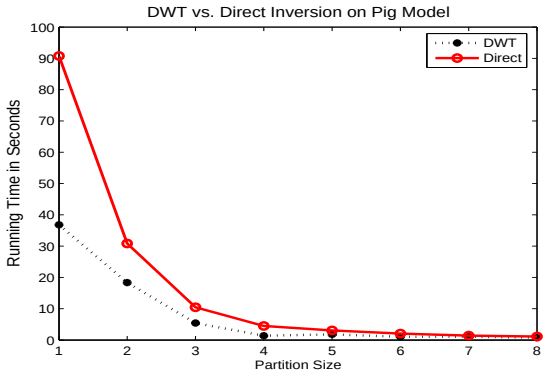

Figure 4. This figure compares the time required to directly invert the Laplacian matrix (top curve) using the pseudo-inverse routine `pinv` in MATLAB vs. computing the inverse using the multiscale diffusion wavelet tree (bottom curve).

5.3. Computing Local Basis Functions

Once a graph $G = (V, E, W)$ has been partitioned into a set of k partitions $G_i = (V_i, E_i, W_i)$, we compute set of basis functions on each subgraph, either using the eigenvectors of the Laplacian on subgraphs, or computing multiscale local diffusion wavelet bases. One subtle issue is the boundary effects that can result from the fact that there are vertices that lie in the intersection of two or more subgraphs. In practice, we have found that boundary effects are not significant enough to cause a problem, although more work is needed to address this issue.

[1] Although other inversion techniques such as `cgs` could be used to invert larger matrices, their numerical precision is quite poor compared to `pinv`.

6. Experimental Results

In this section, we present a series of detailed experiments, evaluating our multiscale approach to adaptive compression of 3D objects. To compare the effectivess of mesh geometry reconstruction by projection onto a set of Laplacian or diffusion wavelet bases, we need to define some notion of error. The most straightforward method is to compare the difference between the predicted mesh coordinates $\hat{v}$ with the true mesh coordinates v, that is the *geometric error* between two models M_1 and M_2 is defined as

$$\|M_1 - M_2\|_g = \sum_{v \in V} \sum_{i \in (x,y,z)} (\hat{v}_i - v_i)^2 \tag{5}$$

where for example $\hat{v}_x$ gives the approximated x coordinate and v_x is the exact known x coordinate of vertex v. Unfortunately, geometric error by itself is not sufficient, since it is possible that a model may be close geometrically, and yet provide a poor "visual" reconstruction. (Karni & Gotsman, 2000) introduced a second metric, called the *geometric Laplacian*, defined as follows:

$$GL(v_i) = v_i - \frac{\sum_{j \in n(i)} l_{ij}^{-1} v_j}{\sum_{j \in n(v)} l_{ij}^{-1}} \tag{6}$$

where $n(v)$ is the set of neighbors of vertex v, and v_i again is the i^{th} index of the mesh coordinate geometry (for $i = x, y, z$). This term intuitively measures the difference between the prediction made by simply averaging the coordinates of the neighbors of a vertex versus the actual prediction. The final error in approximation is then defined as the sum of the normalized geometric Laplacian error and the geometric error:

$$\|M_1 - M_2\| = \frac{1}{2n} \left(\|M_1 - M_2\|_g + \sum_{v \in V} \sum_i GL(v_i) \right) \tag{7}$$

6.1. Compression of Small Objects

We first compare the mesh compression of global Laplacian eigenvectors against multiscale diffusion wavelet bases for "small" $3D$ objects, where by "small" we mean objects with mesh graphs of size < 5000 vertices. Note that these are actually fairly large graphs comparable in size to the standard data set sizes in previous work in manifold or spectral learning (e.g. the "Swissroll" or "Two-Moons" manifolds, or digit recognition). Figure 5, Figure 6, Figure 7, and Figure 8 compare the performance on a "cow", "camel", "pig",

and "Max Planck" model, respectively. Each experiment was carried out using the same set of parameters. The overall graph was partitioned into 50 subgraphs, and then a varying number of basis functions were constructed on each subgraph. The errors introduced in each local subgraph mesh aprpoximation were then added together to produce the final plots shown. In each graph, the horizontal axes measures the number of basis functions, and the vertical axes measures the sum of the geometric error and the geometric Laplacian error, as defined above. In each graph, the bottom curve represents multiscale diffusion bases, and the top curve represents Laplacian bases. It is clear that the multiscale diffusion wavelet bases consistently performs better than the partitioned Laplacian eigenvector bases. The running times shown are the average time for a specific number of bases.[2]

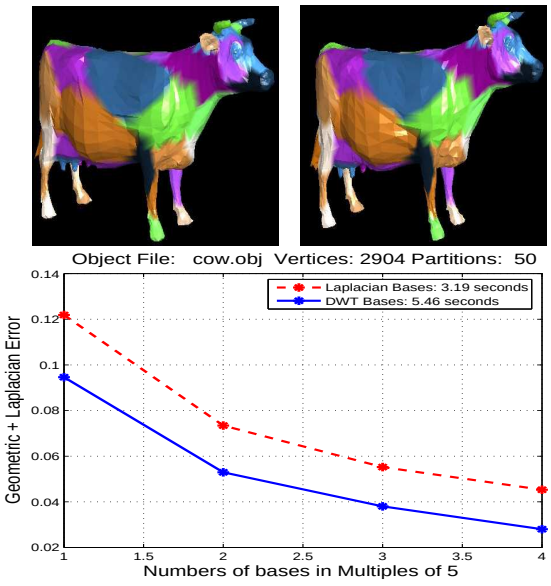

Figure 5. Comparison of Laplacian (top left) and diffusion wavelet (top right) approximation of a 3D "cow" model with $|V| = 2904$ vertices.

6.2. Partition Size vs. Error

The divide-and-conquer approach seems a natural way to make the adaptive spectral compression problem more tractable, but it comes at a price. As the number of partitions grows, the error is likely to increase due to boundary effects, but the running time reduces. We explore this tradeoff for the "pig" model analyzed above. Figure 9 displays the change in error and running times vs. partition size for the "pig" model. The

[2]Unlike the `eigs` package in MATLAB for computing eigenvectors, the diffusion wavelet code is not yet highly optimized, but a faster implementation is currently underway.

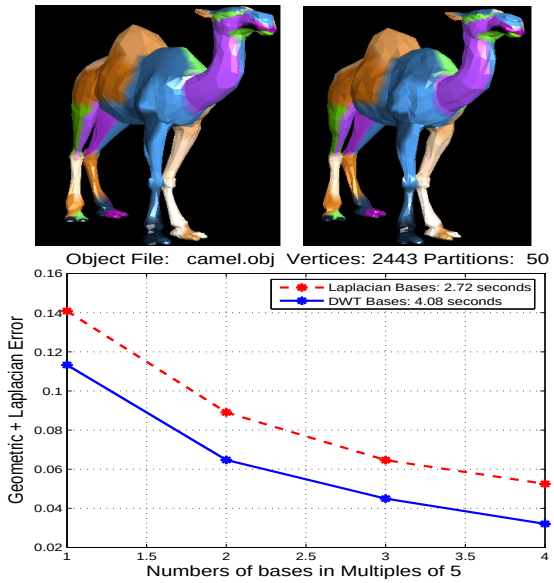

Figure 6. Comparison of Laplacian (top left) and diffusion wavelet (top right) approximation of a 3D "camel" model with $|V| = 2443$ vertices.

figure plots the results only for multiscale diffusion bases. The increase in error turns out to depend on the smoothness of the model: the camel exhibits the worst increase in error whereas the pig exhibits the least. As shown in the figure, the error increase is at beast linear, but the running time shows a significant superlinear decrease.

6.3. Compression of Large Objects

In this section, we compare the performance of multiscale diffusion bases against Laplacian bases on larger $3D$ objects, where the number of vertices $|V| > 10^4$. Specifically, Figure 10 compares multiscale diffusion wavelet bases vs. global Laplacian bases on an "Elephant" model. The colors indicate the partitions on which local basis functions were computed. As in the earlier "cow" model, sharp features such as the tusks are rendered with much higher fidelity by the diffusion wavelet bases. Figure 11 plots the results for the "Stanford Bunny", a standard benchmark problem in computer graphics.

7. Future Work

Some extensions of this research are briefly summarized. 3D models are sometimes specifed by point sets $\in \mathcal{R}^n$, and can be handled by adding a graph construction phase. The multiscale diffusion bases can also be modified to yield *geometry-aware* bases (Sorkine

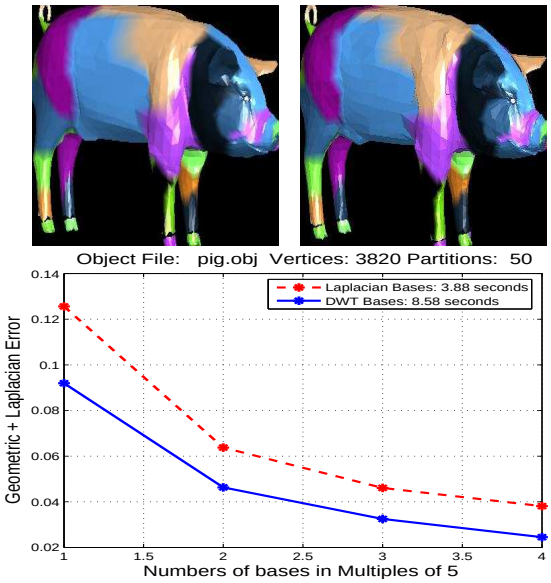

Figure 7. Comparison of Laplacian (top left) and diffusion wavelet (top right) approximation of a 3D "pig" model with $|V| = 3820$ vertices.

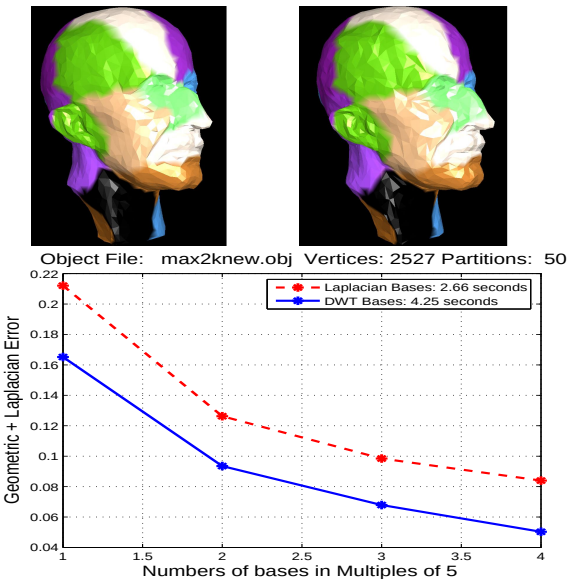

Figure 8. Comparison of Laplacian (top left) and diffusion wavelet (top right) approximation of a 3D model of Max Planck with $|V| = 2527$ vertices.

et al., 2005), where the coordinate function being approximated can influence the bases constructed. Another open problem is to compress *tensor* information stored at each vertex, such as a matrix of illumination or texture values. One strategy is to use homogeneous graphs, defined by a group operator, over which the *vibrational Laplacian* can be constructed (Chung, 1997). Finally, a theoretical analysis of the stability of Laplacian and diffusion wavelet bases under graph partitioning is ongoing.

Acknowledgments

The author thanks Rui Wang for providing some of the models and 3D display software, and Mauro Maggioni for the diffusion wavelet code. This research was supported in part by the National Science Foundation under grant IIS-0534999.

References

Amhed, N., Natarajan, T., & Rao, K. (1974). On image processing and a discrete cosine transform. *IEEE Transactions on Computers*, *C-23*, 90–93.

Belkin, M., & Niyogi, P. (2004). Semi-supervised learning on Riemannian manifolds. *Machine Learning*, *56*, 209–239.

Bremer, J. C., Coifman, R. R., Maggioni, M., & Szlam, A. D. (2006). Diffusion wavelet packets. *Applied and Computational Harmonic Analysis*, *21*, 95–112.

Chung, F. (1997). *Spectral graph theory*. No. 92. American Mathematical Society.

Coifman, R. R., & Maggioni, M. (2006). Diffusion wavelets. *Applied and Computational Harmonic Analysis*, *21*, 53–94.

Karni, Z., & Gotsman, C. (2000). Spectral compression of mesh geometry. *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (pp. 279–286).

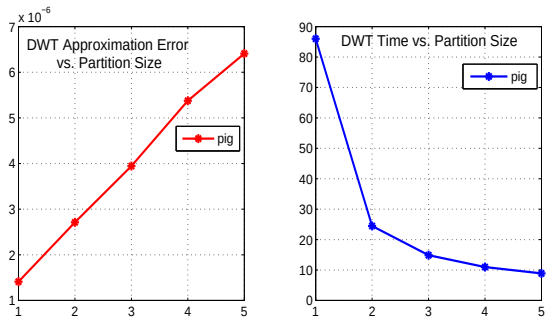

Figure 9. Comparison of average error and running times (seconds) over different partition sizes, showing error increases sublinearly, but running time reduces superlinearly as number of partitions increase.

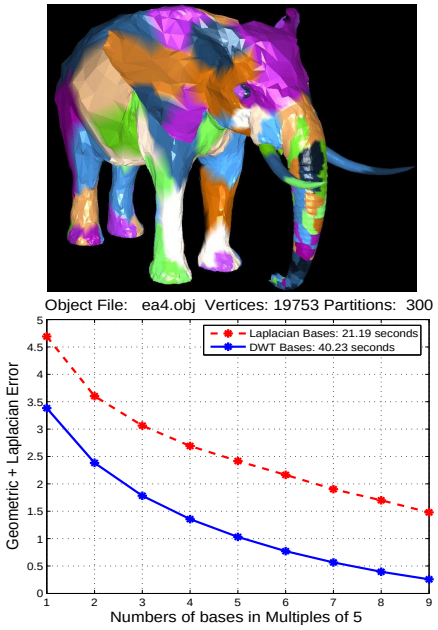

Figure 10. Results for an "Elephant" model, a 3D object with 19,753 vertices and 59,053 edges.

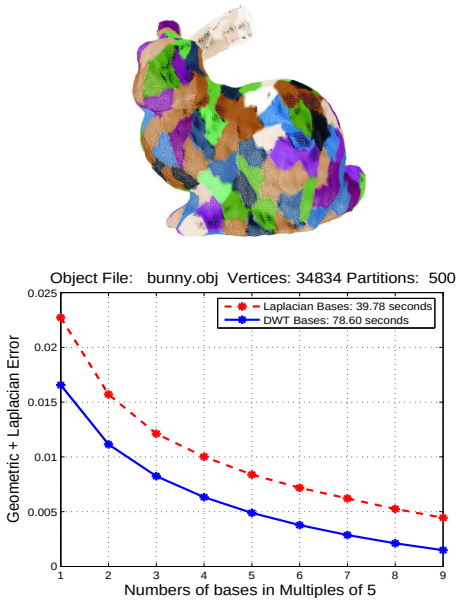

Figure 11. Results for "Stanford Bunny", a 3D object with 34,834 vertices and 104,288 edges.

New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Karypis, G., & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, *20*, 359–392.

Maggioni, M., & Mahadevan, S. (2006). Fast direct policy evaluation using multiscale analysis of markov diffusion processes. *ICML '06: Proceedings of the 23rd international conference on Machine learning* (pp. 601–608). New York, NY, USA: ACM Press.

Mahadevan, S., & Maggioni, M. (2006). Value function approximation with diffusion wavelets and laplacian eigenfunctions. *Proceedings of the Neural Information Processing Systems (NIPS)*. MIT Press.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, *11*, 674–693.

Meila, M., & Shi, J. (2001). Learning segmentation by random walks. *NIPS*.

Ng, A., Jordan, M., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*.

Niyogi, P., Matveeva, I., & Belkin, M. (2003). *Regression and regularization on large graphs* (Technical Report). University of Chicago.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Tran PAMI*, *22*, 888–905.

Singer, A. (2006). From graph to manifold Laplacian: The convergence rate. *Appl. Comp. Harm. Anal.*, *21*, 128–134.

Sorkine, O., Cohen-Or, D., Irony, D., & Toledo, S. (2005). Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics*, *11*, 171–180.

Taubin, G. (1995). A signal processing approach to fair surface design. *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (pp. 351–358). New York, NY, USA: ACM Press.

Wallace, G. (1991). The JPEG still picture compression standard. *Communications of the ACM*, *34*, 30–44.

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation* (Technical Report CMU-CALD-02-107). Carnegie Mellon University.