High-Pass Quantization with Laplacian Coordinates

OLGA SORKINE, DORON CHEN, DANIEL COHEN-OR, and SIVAN TOLEDO
Tel-Aviv University

Any quantization introduces errors. An important question is how to suppress their visual effect. In this paper we present a new quantization method for the geometry of 3D meshes, which enables aggressive quantization without significant loss of visual quality. Conventionally, quantization is applied directly to the 3-space coordinates. This form of quantization introduces high-frequency errors into the model. Since high-frequency errors modify the appearance of the surface, they are highly noticeable, and commonly, this form of quantization must be done conservatively to preserve the precision of the coordinates. Our method first multiplies the coordinates by the Laplacian matrix of the mesh and quantizes the transformed coordinates which we call Laplacian coordinates or "δ-coordinates". We show that the high-frequency quantization errors in the δ-coordinates are transformed into low-frequency errors when the quantized δ-coordinates are transformed back into standard Cartesian coordinates. These low-frequency errors in the model are much less noticeable than the high-frequency errors. We call our strategy *high-pass quantization*, to emphasize the fact that it tends to concentrate the quantization error at the low-frequency end of the spectrum. To allow control over the shape and magnitude of the low-frequency quantization errors, we extend the Laplacian matrix by adding a number of spatial constraints. We analyze the singular values of the extended matrix and derive bounds on the quantization and rounding errors. We show that the small singular values, and hence the errors, are related in a specific way to the number and location of the spatial constraints.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*surface, solid, and object representations*

General Terms: Algorithms

Additional Key Words and Phrases: topological Laplacian operator, quantization, mesh geometry, linear least squares, spectral methods

1. INTRODUCTION

Polygonal meshes are widely used for representation of 3D objects. Compression of 3D meshes is today an active research area, important for web-based applications, efficient storage and archiving. Mesh compression involves two problems that are usually solved, at least conceptually, separately: the mesh *connectivity* encoding and the *geometry* encoding. While state-of-the-art connectivity encoding techniques are extremely effective [Touma and Gotsman 1998; Alliez and Desbrun 2001], compressing the geometry remains a challenge. The encoded geometry is, on average, at least five times larger than the encoded connectivity, even when the coordinates are pre-quantized to 10–12 bits. Finer quanti-

Authors' address: School of Computer Science, Tel-Aviv University, Tel-Aviv 69978 Israel.
{sorkine|mycroft|dcor|stoledo}@tau.ac.il

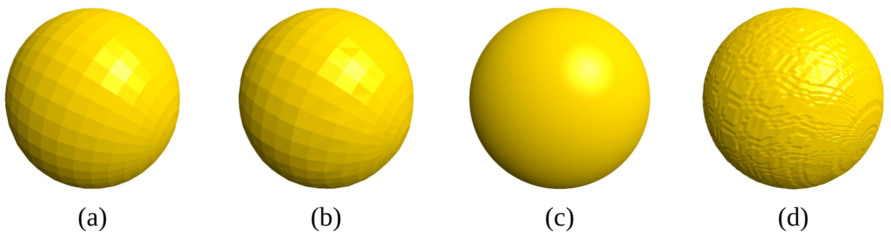


Fig. 1. The counter-intuitive effect of quantization. Fine-sampled surfaces suffer more than coarse ones. (a) A coarse mesh representation of a sphere with high-precision coordinates. (b) The same mesh with Cartesian coordinates quantized to 8 bits/coordinate. (c) A fine mesh representation of the sphere. (d) Quantization of the fine mesh to 8 bits/coordinate yields a jaggy surface. Note that the RMS of vertex displacements is roughly the same in both the coarse mesh and the fine mesh, although the visual error is clearly larger in the fine mesh.

zation for higher precision increases the importance of effective geometry encoding even further.

The raw geometry data, whether originating from scanned real-world objects or synthetic modeling applications, usually comes in high-precision floating-point representation. Such data cannot be significantly compressed by standard techniques such as dictionary-based coding (e.g., LZ), or entropy coding; therefore, most geometry encoding schemes involve quantization. Normally, the Cartesian coordinates of each vertex are uniformly quantized, and the resulting integer values are encoded using predictive approaches that rely on local surface smoothness assumptions [Touma and Gotsman 1998; Taubin and Rossignac 1998]. Another possibility is to alter the surface representation; for instance, to treat the geometry as a surface signal and employ signal processing and compression techniques, such as wavelet compression [Khodakovsky et al. 2000]. However, such approaches require modification of the connectivity of the mesh into a regular or semi-regular network. While the new mesh might be close enough to the original surface, some important local features that are well represented by a specific connectivity might get washed out. Thus, in many cases it is desirable to keep the original connectivity intact.

Quantization necessarily introduces errors and causes a certain loss of data. Loosely speaking, quantizing the Cartesian coordinates of the mesh produces high-frequency errors across the surface. This especially damages the fine-sampled areas, since the relative error is greater when the polygons are smaller (see Figure 1). Aggressive quantization significantly alters the surface normals, causing the irritating "jaggies" effect (see Figure 1(d)). Thus, only mild quantization of Cartesian coordinates is possible without causing visible artifacts (usually between 10 and 16 bits per coordinate).

In this paper, we investigate a different approach to geometry quantization. Instead of directly quantizing the Cartesian coordinates, we first transform them to another space by applying the Laplacian operator associated with the mesh topology. We call these transformed coordinates "δ-coordinates". The quantization is applied to the δ-coordinates, and the geometry of the mesh can be restored on the decoder side by solving a linear least-squares system defined by the extended Laplacian matrix (discussed in Section 3). We show that introducing high-frequency errors by quantizing the δ-coordinates results in *low-frequency* errors in the reconstructed Cartesian coordinates. By considering a visual error metric between meshes, that takes into account not only the Euclidean distance between corresponding vertices (or the "Metro" distance [Cignoni et al. 1998]) but also the smoothness error, we argue that low-frequency displacements in the surface geometry are less noticeable than high-frequency displacements which modify the local characteristics of the surface such as normals and curvature. Consequently, strong quantization of the δ-coordinates yields a small visual error, in contrast to standard Cartesian coordinate quantization.

We call our strategy *high-pass quantization*, to emphasize the fact that it tends to concentrate the quantization error at the low-frequency end of the spectrum. A high-pass filter also concentrates the error at the low end of the spectrum, and the form of the error is known: damping. In high-pass quantization, the high-end of the spectrum is preserved, as in high-pass filtering. The low-frequency errors that high-pass quantization introduces, however, are essentially random. They do not necessarily correspond to damping. The randomness is an outcome of the quantization process, which always introduces random errors.

Lossy compression methods are evaluated by rate-distortion curves, which correlate bitrates with signal distortion. We claim that there is not yet a visual distortion metric for 3D models that can objectively rank compression methods. One of our main contributions is the observation that visual distortion is highly influenced by the spectrum of the error, in ways that are not captured well by existing distortion metrics. We address the quantitative evaluation issue using a two-pronged approach: (i) In Section 5 we propose a distortion metric and show that it captures our visual perception well; then we demonstrate the effectiveness of our method by means of rate-distortion curves; (ii) We show in a visual metric-independent way that the rate-distortion of our method is better than that of direct quantization methods.

The paper presents two main contributions. The first is the observation that lossy mesh compression should introduce low-frequency errors but almost no high-frequency errors. We assume that high-frequency information below the visual threshold has already been filtered from the coordinates. Therefore, compression should aim to preserve the remaining significant high-frequency information. The second contribution is a computational method, based on extended Laplacian matrix, that achieves this objective.

The key ideas of this paper were first presented in [Sorkine et al. 2003]. Here[1] we provide a complete mathematical analysis of the method supported by an extended set of experiments. We show that the shape and magnitude of the quantization error are governed by the singular values and singular vectors of matrices associated with the mesh. We analyze the extreme singular values of these matrices and bound them in terms of topological properties of the mesh. We also bound the rounding errors introduced by the method, again in terms of topological mesh properties. Our analysis shows that both the quantization and rounding errors can be reduced and controlled by introducing spatial constraints. We propose algorithms to select appropriate constraints in order to meet desired error bounds. The present paper also expands the discussion on efficiently solving the optimization problem that arises in our method, and expands the empirical study of the compression-ratios versus visual-quality tradeoff.

Section 2 reviews previous work related to the mesh Laplacian and the field of geometry

[1]**A note to the reviewers:** This version of the paper is significantly larger than the conference version (about 50% larger). It contains much more details, examples, experiments and rigorous mathematical analysis.

compression. Section 3 shows the properties of the Laplacian matrix and the δ-coordinates, which enable their strong quantization. Section 4 presents two ways to add constraints to the Laplacian, and investigates the spectral properties of the extended Laplacians. In Section 5 we describe the visual error metric that better captures the visual distance between meshes. Section 6 presents implementation details and the results, and we conclude in Section 8. Our research raises a number of interesting and important open problems, which we describe in Section 7. We present our conclusions in Section 8.

2. RELATED WORK

Following the pioneering work of Deering [1995] and Taubin and Rossignac [1998], numerous mesh-compression techniques have been developed, which focus mainly on connectivity encoding (e.g., [Gumhold and Straßer 1998; Touma and Gotsman 1998; Rossignac 1999]. It has been shown that the efficiency of the connectivity encoding has reached near-optimal level [Gumhold 2000; Alliez and Desbrun 2001]. Our work is based on these results since the encoded connectivity is in fact an efficient encoding of our extended Laplacian matrix.

As mentioned above, the geometry data size is significantly larger than the encoded connectivity data size. In recent years the focus has been shifted to efficient encoding of the geometry. In earlier works, the geometry was encoded by a predictive coding paradigm. The vertices of the mesh are traversed in some order $v_1,\ldots,v_n$ and each vertex v_i is encoded based on the known locations of the previous vertices in the sequence $v_1,\ldots,v_{i-1}$. The unknown location of v_i is predicted to be at $\hat{v}_i$, and the displacement (the residual error) $e_i = \hat{v}_i - v_i$ is encoded. Usually linear predictors are used; the most common one is known as the parallelogram predictor [Touma and Gotsman 1998].

The above methods first quantize the mesh vertices and then losslessly encode the displacements. Our approach is different: we compute the displacements on the *exact* geometry and then quantize them in a lossy manner.

In all the above methods, the displacements are compressed by some entropy encoder. Chou and Meng [2002] use vector quantization instead to gain speed. Their paper, as well as others, does not measure the relation between the quantization error and the visual quality of the decoded mesh. Most works consider the Metro-like measure, rather than a visual error metric. A notable exception is the work of Karni and Gotsman [2000], where the compression results are measured in terms of visual quality.

The mesh-compression method of Karni and Gotsman [2000] is based on the spectral properties of Laplacians, as well as our work, but it is fundamentally and computationally different from our method. Karni and Gotsman propose to compute the eigenvectors of the Laplacian of the mesh, expand the mesh functions (the *x*, *y* and *z* vectors) in this basis, and drop the coefficients of high-frequency modes from the representation. The rationale is that smooth shapes can be well represented by a linear combination of low-frequency modes (the same applies to other bases, such as wavelet bases). The fundamental difference between their method and ours is that the error in their method consists entirely of high-frequency modes, since these are the modes that suffer from the lossy representation, whereas the error in our method consists mostly of low-frequency modes. In models that have some high-frequency components, such as folds, corners, or rough surfaces, their method wipes out these features, whereas ours preserves them almost perfectly (see Figure 9). In other words, both methods exploit the fact that 3D models can be well approximated by a combination of low-frequency Laplacian eigenvectors, but the compression errors in the two methods are entirely different.

Another important difference between our method and Karni and Gotsman's lies in computational efficiency. Their method requires computing eigenvectors of the Laplacian, which is computationally more expensive than solving a sparse least-squares problem, which is the computational kernel in our method.

An alternative to quantization as means of geometry compression is mesh simplification, which removes vertices and changes the connectivity of the mesh. The trade-off between simplification and quantization is extensively studied by King and Rossignac [King and Rossignac 1999]. They define a *shape complexity* measure and use it to estimate the optimal number of vertices and bits per vertex, given an error bound or file size bound. In this work, our goal is to investigate a different way to perform geometry quantization, while preserving the connectivity. However, it would be interesting to combine our findings on quantization with the above study.

The mesh Laplacian has other applications in Computer Graphics. Taubin [Taubin 1995] showed the use of the Laplacian matrix as a tool for spectral analysis of the 3D mesh. In his work, Taubin designs a mesh smoothing filter and a modeling tool. Alexa [Alexa 2001; 2003] uses Laplacian coordinates for 3D morphing. He shows that by interpolating Laplacian coordinates locally, the intermediate surfaces remain smoother and tend to deform less than linearly interpolated Cartesian coordinates. Ohbuchi et al. [Ohbuchi et al. 2002] use spectral decomposition of the Laplacian to watermark 3D models. The amplitude of the spectral coefficients of low-frequency modes is modulated to embed a watermark bitstring. Their work, like ours, exploits the observation that low-frequency errors are almost invisible.

3. LAPLACIAN MATRIX AND δ-COORDINATES

3.1 Algebraic background

Quantizing a vector x with continuous coefficients introduces an error q_x, where $x + q_x$ is the quantized vector. In this section we show how to control the spectral behavior of the error using linear transformations. We assume that a simple fixed-point quantization is used, so that the maximum quantization error $\max_i |q_i|$ is bounded by $2^{-p}(\max_i x_i - \min_j x_j)$, using p-bit quantized coefficients.

Suppose that instead of quantizing the input vector x, we first transform x into a vector Ax using a nonsingular matrix A, and then quantize Ax. We denote the quantization error by q_{Ax}, so that the new quantized vector is $Ax + q_{Ax}$. The elements of the quantized vector are now discrete, as are those of $x + q_x$. We can recover an approximation of x from this representation, by multiplying the quantized vector by A^{-1}:

$$A^{-1}(Ax + q_{Ax}) = x + A^{-1}q_{Ax}.$$

The error in this approximation is $A^{-1}q_{Ax}$, and we will see shortly that under certain conditions, it behaves quite differently than q_x.

Assume that A has an orthonormal eigen-decomposition $AU = U\Lambda$, where U is unitary (has orthonormal columns) and Λ is diagonal. This assumption is satisfied when A is real and symmetric, and more generally, if and only if $AA^* = A^*A$, where A^* is the Hermitian adjoint of A. Without loss of generality, we assume that

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|,$$

where $\lambda_i = \Lambda_{ii}$ are the eigenvalues of A. Since the processes we are concerned with are invariant to scaling A, we also assume that $|\lambda_1| = 1$. We express x as a linear combination of A's orthonormal eigenvectors,

$$x = c_1 u_1 + c_2 u_2 + \cdots + c_n u_n,$$

where u_i are the columns of U. We also have

$$Ax = c_1 \lambda_1 u_1 + c_2 \lambda_2 u_2 + \cdots + c_n \lambda_n u_n.$$

Similarly, since $A^{-1}U = U\Lambda^{-1}$, we can express the quantization error as a linear combination of eigenvectors,

$$q_{Ax} = c'_1 u_1 + c'_2 u_2 + \cdots + c'_n u_n,$$

so

$$A^{-1} q_{Ax} = c'_1 \lambda_1^{-1} u_1 + c'_2 \lambda_2^{-1} u_2 + \cdots + c'_n \lambda_n^{-1} u_n.$$

We now reach the first fundamental point of the discussion. The transformation A is useful for quantization when three conditions hold:

(1) For typical inputs x, the norm of Ax is much smaller than the norm of x,
(2) Quantization errors with large $c'_i \lambda_i^{-1}$ for large i (that is, with strong representation for the last eigenvectors) are not disturbing,
(3) $|\lambda_n|$ is not too small.

The first point is important since it implies that $\max_i |(Ax)_i| \ll \max_i |x_i|$, which allows us to achieve a given quantization error with fewer bits. The best choice of norm for this purpose is, of course, the max norm, but since norms are essentially equivalent, the implication also holds if

$$\|Ax\|_2 \ll \|x\|_2.$$

Since $\|x\|_2^2 = \sum_i c_i^2$ and $\|Ax\|_2^2 = \sum_i c_i^2 \lambda_i^2$, the above condition occurs if and only if the first c_i's are small compared to the last ones. In other words, the first point holds if A, viewed as a filter, filters out strong components of typical x's.

The importance of the second and third points stems from the fact that A^{-1} amplifies the components of q_{Ax} in the direction of the last eigenvalues. If A has tiny eigenvalues, the amplification by a factor λ_i^{-1} is significant for large i. Even if the small eigenvalues of A are not tiny, the error may be unacceptable. The quantization error $A^{-1}q_{Ax}$ always contains moderate components in the direction of eigenvectors that correspond to the small eigenvalues of A. When small error components in these directions distort the signal perceptively, the error will be unacceptable. Therefore, the last two points must hold for the quantization error to be acceptable.

3.2 Laplacian transformations

This section discusses the Laplacian matrix of the mesh and its variants and shows that these linear transformations work well as quantization transforms.

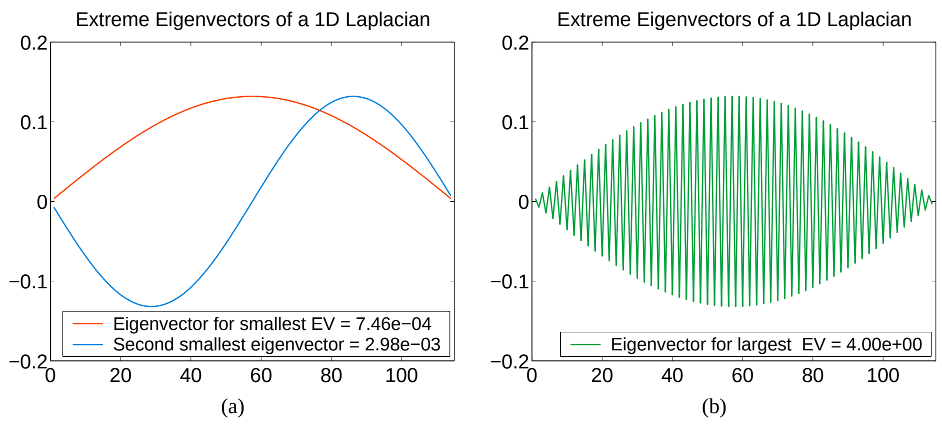


Fig. 2. Extreme eigenvectors of the Laplacian of a one-dimensional mesh. Plot (a) shows u_n and u_{n-1}, the two eigenvectors corresponding to the smallest eigenvalues $\lambda_n = \lambda_{\min}$ and λ_{n-1}. Plot (b) shows u_1, the eigenvector corresponding to the largest eigenvalue λ_1.

Let M be a given triangular mesh with n vertices. Each vertex $i \in M$ is conventionally represented using absolute Cartesian coordinates, denoted by $v_i = (x_i, y_i, z_i)$. We define the *relative* or *δ-coordinates* of v_i to be the difference between the absolute coordinates of v_i and the center of mass of its immediate neighbors in the mesh,

$$\delta_i = (\delta_i^{(x)}, \delta_i^{(y)}, \delta_i^{(z)}) = v_i - \frac{1}{d_i} \sum_{k=1}^{d} v_{i_k} \,,$$

where d_i is the number of immediate neighbors of i (the degree or valence of i) and i_k is i's kth neighbor.

The transformation of the vector of absolute Cartesian coordinates to the vector of relative coordinates can be represented in matrix form. Let A be the adjacency (connectivity) matrix of the mesh:

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

and let D be the diagonal matrix such that $D_{ii} = d_i$. Then the matrix transforming the absolute coordinates to relative coordinates (scaled by D) is $L = D - A$,

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

That is, $Lx = D\delta^{(x)}$, $Ly = D\delta^{(y)}$, and $Lz = D\delta^{(z)}$, where x is an n-vector containing the x absolute coordinates of all the vertices and so on. Without loss of generality, we now focus on the vectors x and $\delta = D\delta^{(x)}$.

The matrix L is called the *Laplacian* of the mesh [Fiedler 1973]. Laplacians of meshes have been extensively studied [Chung 1997], primarily because their algebraic properties are related to the combinatorial properties of the meshes they represent. The Laplacian is symmetric, singular and positive semidefinite. The singularity stems from the fact that the system $Lx = \delta$ has an infinite number of solutions which differ from each other by a vector that is constant on each connected component of the mesh. Thus, we can actually recover x from δ if we know, in addition to δ, the Cartesian coordinate of one x_i in each connected component. We can formalize this method by dropping from L the rows and columns that correspond to one vertex in each connected component, called the *anchor* of the component. The resulting matrix (see Figure 4), which we call the *basic invertible Laplacian*, generates all the δ's that we need and is nonsingular. The next section explores other nonsingular variants of the Laplacian.

To explain why variants of the Laplacian are effective quantization transforms, we first have to introduce the notion of mesh frequencies (spectrum). The *frequency* of a real function x defined on the vertices of a mesh M is the number of zero crossings along edges,

$$f(x) = \sum_{(i,j)\in E(M)} \left\{ \begin{array}{ll} 1 & x_i x_j < 0 \\ \\ 0 & \text{otherwise} \end{array} \right\} ,$$

where $E(M)$ is the set of edges of M, so the summation is over adjacent vertices. It turns out that for many classes of graphs, including 3D meshes, eigenvectors of the Laplacian (and related matrices, such as our basic invertible Laplacian) corresponding to large eigenvalues are high-frequency mesh functions, and eigenvectors corresponding to small eigenvalues are low-frequency mesh functions (see example in Figure 2. In other words, when $i \ll j$, $\lambda_i > \lambda_j$ and $f(u_i) \gg f(u_j)$. Furthermore, since 3D models are typically smooth, possibly with some relatively small high-frequency perturbation, the coordinate vectors x, y, and z often have a large low-frequency and a small high-frequency content. That is, the first c_i's are often very small relative to the last ones.

This behavior of the eigenvectors of Laplacians and of typical 3D models implies that the first property we need for effective quantization holds, namely, the 2-norm of Lx is typically much smaller than the norm of x, and therefore the dynamic range of Lx is smaller than that of x.

Laplacians also satisfy the second requirement. As stated above, eigenvectors associated with small eigenvalues are low-frequency functions that are typically very smooth. When we add such smooth low-frequency errors to a 3D model, large features of the model may slightly shift, scale, or rotate, but the local features and curvature are maintained. Thus, errors consisting mainly of small-eigenvalue low-frequency eigenvectors are not visually disturbing.

However, simple Laplacian transformations do not satisfy our third requirement. The small eigenvalue of a basic invertible Laplacian is typically tiny; a good estimate for $|\lambda_n^{-1}|$ is the product of the maximum topological distance of a vertex from the anchor vertex, and the number of vertices in the mesh (assuming there is one connected component; otherwise the maximum of the estimate over all components) [Guattery and Miller 2000; Boman and Hendrickson 2001]. For a typical n-vertex 3D mesh, the small eigenvalue is therefore likely to be $\Theta(n^{-1.5})$. This causes large low-frequency errors which are clearly visible in the example in Figure 3.

4. THE K-ANCHOR LAPLACIAN

An effective way to increase the small eigenvalue of a Laplacian is to add more anchor points. This section analyzes the effect of two algorithm parameters on the magnitude and shape of the quantization error. One parameter is the number and location of the

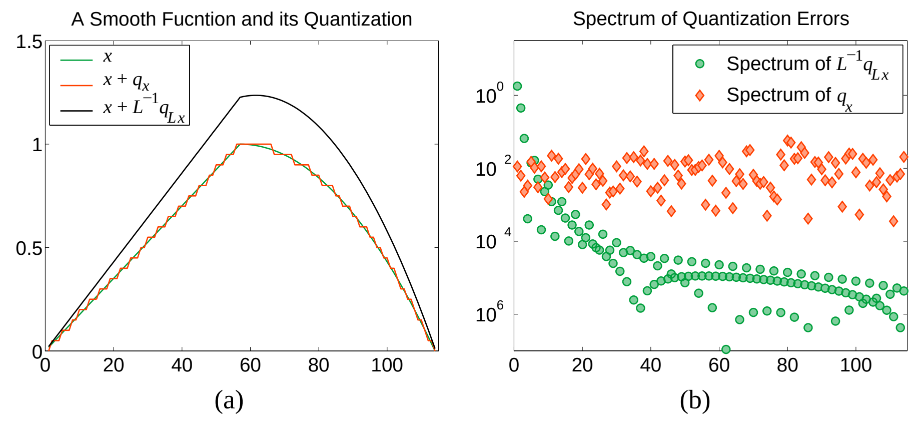


Fig. 3. An example showing quantization errors in a one-dimensional mesh. The mesh here is a simple linear chain with 114 vertices. (a) shows a smooth function x defined on the mesh, its direct quantization, and a Laplacian-transform quantization. The quantizations were performed with 20 discrete values uniformly distributed between the minimum and maximum absolute values of the vectors. The direct error vector is smaller in magnitude, but has a strong high-frequency oscillatory nature, whereas the Laplacian-transformed error vector is smooth. (b) explains this observation by plotting, on a log scale, the spectrum of the two errors. We can see that the direct quantization has moderate components in the direction of all eigenvectors of the Laplacian (i.e., all frequencies), whereas the Laplacian-transformed error has strong components in the direction of the smooth eigenvectors, but very small components in the direction of high-frequency eigenvectors.

anchor points. The second parameter is the algorithm that transforms the relative (or δ) coordinates to the original coordinates.

The relationship between the original coordinates x and the relative coordinates δ is given, up to a shift, by the linear system of equations $Lx = \delta$. When we add anchors, we essentially add constraints to this system of equations. Without loss of generality, we assume that the anchors are $x_1, \dots, x_k$, the first k vertices of the mesh. For each anchor point x_{i_j}, $j = 1, \dots, k$, we add the constraint $x_i = x_i$, where the left-hand side is taken to be an unknown and the right-hand side a known constant.

It may seem strange that we do not immediately substitute the known constant for the unknown, but the reason for this will become apparent later. The full system of constraints that defines the relationship between the absolute and relative coordinates is therefore

$$\left(\frac{L}{I_{k\times k} \,|\, 0} \right) x = \begin{pmatrix} Lx \\ x_{1:k} \end{pmatrix} = \begin{pmatrix} \delta \\ x_{1:k} \end{pmatrix} . \tag{1}$$

We denote this $(n+k)$-by-n matrix by $\tilde{L}$,

$$\tilde{L} = \left(\frac{L}{I_{k\times k} \,|\, 0} \right) , \tag{2}$$

and call it the *k-anchor rectangular Laplacian*. Figure 4 shows a small mesh, its Laplacian, along with a 2-anchor rectangular Laplacian.

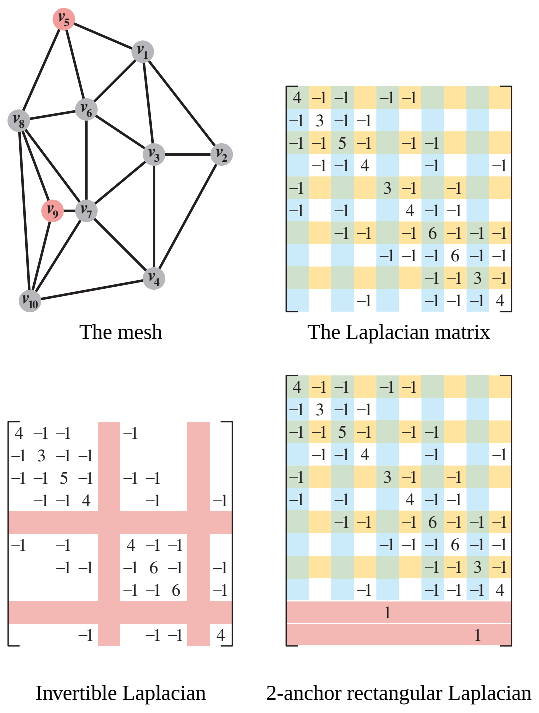


Fig. 4. A small example of a triangular mesh and its associated Laplacian matrix (top right). Second row: a 2-anchor invertible Laplacian and a 2-anchor rectangular Laplacian. The anchors are denoted in the mesh in red.

With k anchors, the quantized representation of the mesh consists of the quantized δ's and of the absolute coordinates of the anchors. Since we take k to be much smaller than n, there is no need to aggressively quantize the coordinates of the anchors, but they can be quantized as well. The quantized vector that represents the mesh is, therefore,

$$\begin{pmatrix} Lx + q_{Lx} \\ x_{1:k} + q_{x_{1:k}} \end{pmatrix} = \begin{pmatrix} Lx \\ x_{1:k} \end{pmatrix} + q_{\tilde{L}x} = \tilde{L}x + q_{\tilde{L}x} \; . \tag{3}$$

The matrix $\tilde{L}$ is rectangular and full rank. Hence, trying to recover x from $\tilde{L}x + q_{\tilde{L}x}$ by "solving" the constraint system $\tilde{L}x' = \tilde{L}x + q_{\tilde{L}x}$ for x' will fail, since this system is overdetermined, and therefore most likely inconsistent. An approximation x' can be computed in (at least) two ways. The simplest is to eliminate the last k rows from the system. By adding row $n+j$ to row j, for $j = 1, \ldots, k$ and deleting row $n+j$, we obtain a square symmetric positive definite linear system of equations $\hat{L}x' = b$, which can be solved for x'. This transformation corresponds to multiplying both sides of the system $\tilde{L}x' = \tilde{L}x + q_{\tilde{L}x}$ by an n-by-$(n+k)$ matrix

$$J = \left(I_{n\times n} \;\middle|\; \begin{array}{c} I_{k\times k} \\ \hline 0 \end{array} \right) \; , \tag{4}$$

so

$$\hat{L} = J\tilde{L} \tag{5}$$

and $b = J(\tilde{L}x + q_{\tilde{L}x})$. We call $\hat{L}$ the *k-anchor invertible Laplacian*. For a small example, see Figure 4.

The second method to obtain an approximation x' is to find the least-square solution x' to the full rectangular system $\tilde{L}x' \approx \tilde{L}x + q_{\tilde{L}x}$.

It turns out that the norm of the quantization error is essentially the same in the two approximation methods, but the shape of the error is not. The shape of the error when using a least-squares solution to the rectangular system is smoother and more visually pleasing than the shape of the error resulting from the solution of the square invertible system.

The next subsection analyzes the eigenvalues of $\hat{L}$, which determine the quantization errors in the square invertible case; the following subsection, Section 4.2 relates the error in the rectangular case to the singular values of $\tilde{L}$, and Section 4.3 relates the singular values of $\tilde{L}$ to the eigenvalues of $\hat{L}$.

4.1 The eigenvalues of $\hat{L}$

In this subsection we show how to bound from below the smallest eigenvalue of $\hat{L}$. Bounding the small eigenvalue from below ensures that the transformation $\hat{L}$ satisfies condition (3) in Section 3.1.

The largest eigenvalue $\lambda_{\max}(\tilde{L})$ is at most $2d_{\max} + 1$, where $d_{\max}$ is the maximal degree in the mesh. This bound is less important than the lower bound on the small eigenvalue, since it only ensures that the norm of the transformed coordinates is never much larger than the norm of the absolute coordinates. We expect the transformed norm to be much smaller, so this bound is not particularly important. We include it for completeness, and also to show that even when our quantization method is not very effective, it does not cause much harm either.

We first show that bounding the spectrum of $\hat{L}$ proves a lower bound on the quantization error $x - x'$. The bound is similar to the analysis of the quantization error in Section 3.1, but it is not identical. The difference, which turns out to be quite minor, stems from the fact that we now quantize an $(n+k)$-vector, not an n-vector.

LEMMA 4.1. *The norm of the quantization error $x - x'$ resulting from solving*

$$\hat{L}x' = J\tilde{L}x' = J(\tilde{L}x + q_{\tilde{L}x})$$

is bounded by

$$\|x - x'\|_2 \le \sqrt{2}\lambda_{\min}^{-1}(\hat{L})\|q_{\tilde{L}x}\|_2 \ .$$

PROOF. We add the quantization error $q_{\tilde{L}x}$ to the right-hand side of Equation 1, and multiply both sides by J,

$$J\tilde{L}x' = J(\tilde{L}x + q_{\tilde{L}x}) \ . \tag{6}$$

Because $J\tilde{L}x' = \hat{L}x'$, we can multiply both sides by $\hat{L}^{-1}$ to obtain

$$\begin{aligned} x' &= \hat{L}^{-1}J(\tilde{L}x + q_{\tilde{L}x}) \\ &= \hat{L}^{-1}(\hat{L}x + Jq_{\tilde{L}x}) \\ &= x + \hat{L}^{-1}Jq_{\tilde{L}x} \end{aligned}$$

,

so

$$\|x - x'\|_2 \le \|\hat{L}^{-1}\|_2\|J\|_2\|q_{\tilde{L}x}\|_2 \ .$$

We now bound the first two factors in the right-hand-side product. Because $\hat{L}$ is symmetric positive definite,

$$\|\hat{L}^{-1}\|_2 = \lambda_{\max}(\hat{L}^{-1}) = 1/\lambda_{\min}(\hat{L}) = \lambda_{\min}^{-1}(\hat{L}) \ .$$

By the definition of the 1 and infinity norms,

$$\|J\|_2^2 \leq \|J\|_1 \|J\|_\infty = 1 \cdot 2 = 2 \ ,$$

which completes the proof. □

This lemma shows that to preserve the bound on the norm of $x - x'$, the quantization error $q_{x_{1:k}}$ for the anchor points should be no larger than the quantization error q_{Lx} of the relative coordinates.

We now bound the small eigenvalue of $\hat{L}$. We express the lower bound in terms of a set of paths in the mesh. Given a set of anchor points, we assign each vertex a path to an anchor point. The bound uses the following three metrics of the set of paths.

Definition 4.2. The *dilation* ϑ of the set of paths is the length, in edges, of the longest path in the set. The *congestion* φ of the set is the maximal number of paths that use a single edge in the mesh. The *contention* ρ of the set is the maximal number of vertices whose paths lead to a single anchor point. The maximum is taken over all vertices for dilation, over all edges for congestion, and over all anchors for contention.

The smaller the dilation, congestion, and contention, the better the bound on the small eigenvalue of $\hat{L}$. Note that for a single set of anchor points, we can assign many different sets of paths, some of which yield tighter bounds than others. In addition, even the best set of paths does not, in general, provide a completely tight bound. For more details, see [Boman and Hendrickson 2001]. But the dependence of the bound on the dilation, congestion and contention does provide us with guidelines as to how to select the anchor points. The next theorem is the main result of this subsection.

THEOREM 4.3. *The smallest eigenvalue of $\hat{L}$ satisfies*

$$\lambda_{\min}(\hat{L}) \geq \frac{1}{\varphi \cdot \vartheta + \rho} \ .$$

We use the following strategy to prove this theorem. We will show how to factor $\hat{L}$ into $\hat{L} = VV^T$. The eigenvalues of $\hat{L}$ are the squares of the singular values of V, so it suffices to bound the small singular value of V. The factor V will have a special structure, in which each column corresponds to one edge of the mesh or to one anchor point. We will then use the given set of paths from vertices to anchor points, to construct a matrix W such that $VW = I$, and show how the norm of W is related to the path structure. The equation $VW = I$ will allow us to relate the 2-norm of W, which we can bound using the path set, to the small singular value of V, which we seek to bound.

The following definitions are used in the construction of the factor V.

Definition 4.4. The *edge-vector* $\langle ij \rangle$ has exactly two non-zeros, $\langle ij \rangle_{\min(i,j)} = 1$ and $\langle ij \rangle_{\max(i,j)} = -1$. The *vertex-vector* $\langle i \rangle$ has exactly one non-zero, $\langle i \rangle_i = 1$.

We associate an edge-vector $\langle ij \rangle$ with an edge connecting vertex i with vertex j. The following lemma demonstrates one of the connection between edges and their corresponding vectors.

LEMMA 4.5. *The edge-vectors of a simple path between vertices i and j span the edge-vector $\langle ij \rangle$ with coefficients ± 1.*

The following lemma describes a factorization of k-anchor Laplacian matrices:

LEMMA 4.6. *A k-anchor Laplacian matrix $\hat{L}$ can be factored into $\hat{L} = VV^T$, $V = \left(V_1 \;\; V_2 \right)$, where V_2 is a matrix of unscaled edge-vectors, each column corresponding to one non-zero off-diagonal in $\hat{L}$, and V_1 is a matrix of vertex-vectors, each column corresponding to an anchor point.*

PROOF. For each offdiagonal nonzero $\hat{l}_{ij} = -1$ (each edge of the mesh), V has a column containing the edge vector $\langle ij \rangle$, and for each anchor j, V has a vertex vector $\langle j \rangle$. The edge vectors constitute V_1 and the vertex vectors constitute V_2. It is easy to verify that $\hat{L} = VV^T$. For a more detailed proof, see [Boman et al. 2001]. □

Given the above factorization, we bound the smallest singular value of V. Our course of action in bounding the smallest singular value of V is as follows: we shall find a matrix $W \in R^{m \times n}$ such that $VW = I_{n \times n}$. As the next lemma shows, the matrix G with the smallest 2-norm satisfying $VG = I_{n \times n}$ is the Moore-Penrose pseudo-inverse $G = V^+$ of V [Golub and Loan 1996, pages 257–258]. Therefore, any matrix W satisfying $VW = I_{n \times n}$ has the property $\|W\| \geq \|V^+\|$. We shall then find an upper bound C on $\|W\|$. Since $C \geq \|W\| \geq \|V^+\| = \frac{1}{\sigma_{\min}(V)}$ we will be able to conclude that $\sigma_{\min(V)} \geq \frac{1}{C}$. We first prove a technical lemma concerning the pseudo-inverse (this result is probably well-known, but we have not found it in the literature).

LEMMA 4.7. *Let V be a full-rank n-by-m real matrix, and let G be an m-by-n real matrix such that $VG = I_{n \times n}$. Then $\|G\|_2 \geq \|V^+\|_2$.*

PROOF. The singular values of V^+V are n ones and $m-n$ zeros, so its 2-norm is 1. We now show that for any x with unit 2-norm we have $\|V^+x\|_2 \leq \|Gx\|_2$. Let $c = \|Gx\|_2$, and let $y = Gx/c$, so $\|y\|_2 = 1$. We have $Gx = cy$, and multiplying V from the left on both sides we get $x = Ix = VGx = cVy$. Multiplying now from the left by V^+ we get $V^+x = cV^+Vy$, so

$$\begin{aligned} \left\|V^+x\right\|_2 &= \left\|cV^+Vy\right\|_2 \\ &\leq c\left\|V^+Vy\right\|_2 \\ &\leq c\left\|V^+V\right\|_2 \|y\|_2 \\ &= c \cdot 1 \cdot 1 \\ &= c \\ &= \|Gx\|_2 \ . \end{aligned}$$

□

We are now ready to bound the singular values of V.

LEMMA 4.8. *Given a k-anchor Laplacian $\hat{L}$ with a factorization into edge and vertex vectors $\hat{L} = VV^T$ as in Lemma 4.6, and a set of paths*

$$\Pi = \left\{ \pi_i = (i, i_1, i_2, \ldots, j) | i = 1, \ldots, n \text{ and } j \text{ is an anchor} \right\} \ ,$$

we have

$$\sigma_{\min(V)} \geq \frac{1}{\sqrt{\varphi(\Pi)\cdot\vartheta(\Pi)+\rho(\Pi)}} \ .$$

PROOF. Finding a matrix W satisfying $VW = I_{n\times n}$ is equivalent to finding, for $i = 1,\ldots,n$, a vector w_i such that $Vw_i = e_i$, where $e_i = \langle i\rangle$ is the ith unit vector.

Let j_i be the anchor endpoint of π_i. It is easy to verify that

$$\left\langle ij_i\right\rangle = (-1)^{(i>j_i)} \sum_{(\ell_1,\ell_2)\in\pi_i} (-1)^{(\ell_1>\ell_2)} \left\langle \ell_1\ell_2\right\rangle \ .$$

(we use the convention that a boolean predicate such as $(i > j)$ evaluates to 1 if it is true and to 0 otherwise.) By Lemma 4.6, all the edge vectors in the summation are columns of V. To obtain w_i, all that remains is to add or subtract $\left\langle j_i\right\rangle$, and perhaps to multiply by -1,

$$\langle i\rangle = (-1)^{(i>j_i)}\left\langle ij_i\right\rangle + \left\langle j_i\right\rangle \ .$$

The last two equations together specify w_i, which contains only 1's, -1's, and 0's.

Now that we have found, column by column, a matrix W such that $VW = I_{n\times n}$, we partition the rows of W such that

$$VW = (V_1 V_2)\begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \ .$$

The rows of W_1 correspond to the columns of V_1, the vertex vectors in V, and the rows of W_2 corresponds to the columns of V_2, the edge vectors in V. We will bound the norm of W by bounding separately the norms of W_1 and of W_2.

We first bound $\left\|W_1\right\|_2$.

$$\begin{aligned}
\left\|W_1\right\|_2^2 &\leq \left\|W_1\right\|_1 \left\|W_1\right\|_\infty \\
&= \left(\max_j \sum_i \left|[W_1]_{ij}\right|\right)\left(\max_i \sum_j \left|[W_1]_{ij}\right|\right) \\
&= 1\cdot\rho(\Pi) \ .
\end{aligned}$$

The 1-norm of W_1 is one since there is exactly one nonzero in each column of i, in position j_i, and its value is 1. The ∞-norm of W_1 is the contention of the path set, since each row of W_1 corresponds to one anchor point, and it appears with value 1 in each path (column) that ends in it. Therefore, each row in W_1 contains at most $\rho(\Pi)$ 1's, and the other entries are all 0.

Bounding $\left\|W_2\right\|_2$ is similar. Each row in W_2 corresponds to one edge of the mesh and each column to a path in Π. Each edge is used in at most $\varphi(\Pi)$ paths, so $\left\|W_2\right\|_\infty = \varphi(\Pi)$. Each path contains at most $\vartheta(\Pi)$ edges, so $\left\|W_2\right\|_1 = \vartheta(\Pi)$.

We now bound $\|W\|_2$.

$$\begin{aligned}
\|W\|_2^2 &= \max_{\|x\|_2=1} \|Wx\|_2^2 \\
&= \max_{\|x\|_2=1} \left\| \begin{matrix} W_1 x \\ W_2 x \end{matrix} \right\|_2^2 \\
&= \max_{\|x\|_2=1} \left(\left\|W_1 x\right\|_2^2 + \left\|W_2 x\right\|_2^2\right)
\end{aligned}$$

$$
\begin{aligned}
&\leq \max_{\|x_1\|_2=1} \left\|W_1 x_1\right\|_2^2 + \max_{\|x_2\|_2=1} \left\|W_2 x_2\right\|_2^2 \\
&= \left\|W_1\right\|_2^2 + \left\|W_2\right\|_2^2 \\
&\leq \varphi(\Pi)\vartheta(\Pi) + \rho(\Pi) \,.
\end{aligned}
$$

The bound on $\sigma_{\min(V)}$ follows immediately from the bound on $\|W\|_2$ and from the discussion preceding the statement of the lemma.

□

Now we can conclude that $\lambda_{\min}(\hat{L}) \geq \frac{1}{\varphi\cdot\vartheta+\rho}$. This follows from two facts: (1) in a symmetric positive definite matrix the singular values are the same as the eigenvectors, therefore $\lambda_{\min}(\hat{L}) = \sigma_{\min}(\hat{L})$. (2) if $\hat{L} = VV^T$ then the singular values of $\hat{L}$ are the squares of the singular values of V (this follows directly from V's SVD decomposition).

We can now easily prove the theorem:

PROOF. $\lambda_{\min}(\hat{L}) = \sigma_{\min}(\hat{L}) = \sigma^2_{\min}(V) \geq \frac{1}{\varphi\cdot\vartheta+\rho}$. □

4.2 Bounding the quantization error using the singular values of $\tilde{L}$

We now show that if we define x' as the least-squares minimizer of $\|\tilde{L}x' - \tilde{L}x + q_{\tilde{L}x}\|_2$, the norm of the error $x - x'$ can be bounded using estimates on the singular values of $\tilde{L}$. The analysis below is the equivalent of Lemma 4.1, but for the case of $\tilde{L}$, the k-anchor rectangular Laplacian rather than for the case of $\hat{L}$, the square k-anchor invertible Laplacian.

LEMMA 4.9. *Let x' be the least-squares minimizer of $\|\tilde{L}x' - \tilde{L}x + q_{\tilde{L}x}\|_2$. The norm of the error $x - x'$ is bounded by*

$$
\|x - x'\|_2 \leq \sigma^{-1}_{\min}(\tilde{L}) \|q_{\tilde{L}x}\|_2 \,,
$$

where $\sigma_{\min}(\tilde{L})$ denotes the nth and smallest singular value of $\tilde{L}$.

PROOF. We express x' in terms of the Moore-Penrose pseudo-inverse $\tilde{L}^+$ of $\tilde{L}$,

$$
\begin{aligned}
x' &= \tilde{L}^+ \left(\tilde{L}x + q_{\tilde{L}x}\right) \\
&= x + \tilde{L}^+ q_{\tilde{L}x} \,.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\|x - x'\|_2 &\leq \|\tilde{L}^+\|_2 \|q_{\tilde{L}x}\|_2 \\
&= \sigma^{-1}_{\min}(\tilde{L}) \|q_{\tilde{L}x}\|_2 \,.
\end{aligned}
$$

□

4.3 The singular values of $\tilde{L}$

The next step is to show that the singular values of $\tilde{L}$ cannot be much smaller than the smallest eigenvalue of $\hat{L}$. In fact, we show that they are at most a factor of $\sqrt{2}$ smaller.

The proof of Lemma 4.1 shows that the 2-norm of J is at most $\sqrt{2}$. It is easy to show that the norm is, in fact, exactly $\sqrt{2}$, and that all the singular values of J are either 1 or $\sqrt{2}$. The next lemma shows that the $\sqrt{2}$ bound on the norm of J ensures that $\sigma_{\min}(\tilde{L}) \geq \lambda_{\min}(\hat{L})/\sqrt{2}$.

LEMMA 4.10. *Let A, B, and C be matrices such that $AB = C$. Then*

$$\sigma_{\min}(B) \geq \frac{\sigma_{\min}(C)}{\sigma_{\max}(A)} .$$

PROOF. Suppose for contradiction that $\sigma_{\min}(B) = \varepsilon < \sigma_{\min}(C)/\sigma_{\max}(A)$. Then there exist vectors x and y such that $\|x\|_2 = \|y\|_2 = 1$ and $Bx = \varepsilon y$. (x and y are the right and left singular vectors corresponding to $\sigma_{\min}(B)$.) Therefore,

$$\|Cx\|_2 = \|ABx\|_2 = \|A\varepsilon y\|_2 = \varepsilon\|Ay\|_2 \leq \varepsilon\sigma_{\max}(A)\|y\|_2 = \varepsilon\sigma_{\max}(A) < \sigma_{\min}(C) ,$$

a contradiction. □

We can now prove the main theorem of this subsection.

THEOREM 4.11.

$$\sigma_{\min}(\tilde{L}) \geq \frac{\lambda_{\min}(\hat{L})}{\sqrt{2}} .$$

PROOF. Since $J\tilde{L} = \hat{L}$, by the previous lemma

$$\begin{aligned}\sigma_{\min}(\tilde{L}) &\geq \frac{\sigma_{\min}(\hat{L})}{\sigma_{\max}(J)} \\ &= \frac{\sigma_{\min}(\hat{L})}{\sqrt{2}} \\ &= \frac{\lambda_{\min}(\hat{L})}{\sqrt{2}} .\end{aligned}$$

□

4.4 Singular vectors and the shape of the error

Why do we propose to use a rectangular Laplacian rather than a square invertible one? The reason lies in the shape of the quantization error that each method generates.

We have already seen that adding anchor points increases the smallest singular value of both the invertible and the rectangular Laplacians. Furthermore, in both cases the 2-norm of the error $x - x'$ is bounded by $\sqrt{2}\lambda_{\min}^{-1}(\hat{L}) \left\| q_{\tilde{L}x} \right\|_2$, exactly the same bound. (The actual errors will differ and the norms will most likely differ, since the bounds are not tight, but the bounds we proved are exactly the same.)

We have found, however, that the shape of the error is visually better when we obtain the approximation x' from the rectangular Laplacian. The main difference between the two errors is that the rectangular approximation x' is usually smooth where x is smooth, but the invertible approximation is not. The invertible approximation is almost always non-smooth at the anchors, where "spikes" seem to always appear. This phenomenon is illustrated in Figures 5 and 6.

The crucial observation is that the k-anchor invertible Laplacian essentially forces the error $x - x'$ to zero at the anchors, and allows the error to grow as we get farther and farther away from the anchor points. When we obtain x' from solving a least-squares problem whose coefficient matrix is $\tilde{L}$, x' can differ from x everywhere, including at the anchor points. This allows x' to be smooth.

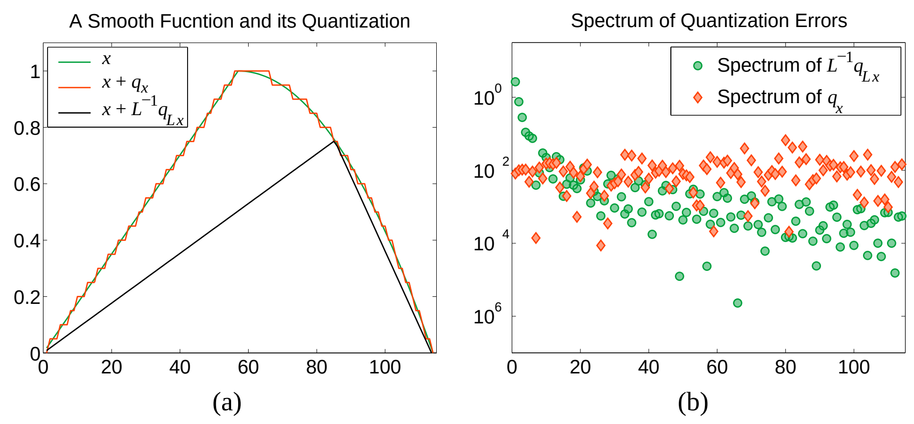


Fig. 5. The same mesh as in Figure 3, but with an additional anchor point at vertex 86. The transformed quantization error is no longer smooth at the anchor point, even though the vector x is smooth there.

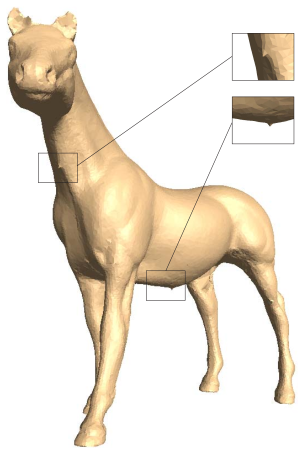


Fig. 6. Reconstruction of the mesh from quantized δ-coordinates using k-anchor invertible Laplacian is not smooth at the anchor vertices.

Formalizing this explanation is hard and is beyond the scope of this paper. The error $x' - x$ consists, in both cases, mainly of the singular vectors of $\tilde{L}$ or $\hat{L}$ that correspond to the smallest singular values. If these singular vectors are smooth, the error $x - x'$ will be smooth, so x' will be smooth where x is smooth. Are these vectors smooth?

The numerical examples in Figures 5 and 6 indicate that the relevant singular/eigen vectors of $\hat{L}$ are *not* smooth. Our experiments also indicate that the singular vectors of $\tilde{L}$ that correspond to small singular values are smooth.

In this paper we do not attempt to prove these statements about the shape of the singular vectors. In general, the singular *vectors* of Laplacian and Laplacian-like matrices have not been researched as much as the singular *values*. It is generally believed that the vectors corresponding to small singular values are indeed smooth. This belief underlies important algorithms such as multigrid [Briggs et al. 2000] and spectral separators [Pothen et al. 1990]. Some additional progress towards an understanding of the relationships between the graph and the eigenvectors of its Laplacian were made recently by Ben-Chen and Gotsman [Ben-Chen and Gotsman 2003]. On the other hand, there is also research that indicates that these vectors are not always well-behaved [Guattery and Miller 2000].

We leave the full mathematical analysis of the shape of the errors as an open problem in this paper, and provide instead empirical evidence that the errors indeed behave as we claim.

4.5 The effect of anchor points on numerical accuracy

So far we have analyzed the norm of the error assuming that x' is the exact solution of $\hat{L}x' = J(\tilde{L}x + q_{\tilde{L}x})$ or exact minimizer of $\left\|\tilde{L}x' - (\tilde{L}x + q_{\tilde{L}x})\right\|_2$. Since we cannot determine x' exactly using floating-point arithmetic, what we actually obtain is an approximation x'' to x'. The total error $x - x''$ depends on both $x - x'$ and $x' - x''$. In this section we analyze the numerical error $x' - x''$, and show that it too depends primarily on the small singular values of the coefficient matrices $\hat{L}$ and $\tilde{L}$, and hence on the anchor points. The results in this section rely on standard error bounds from numerical linear algebra. For details on these error bounds, see for example [Higham 2002] or [Trefethen and Bau 2000]; the first reference is an encyclopedic monograph, the second a readable textbook.

We assume that the approximation x'' is obtained using a *backward stable* algorithm. For the invertible problem, this means that x'' is the exact solution of $(\hat{L} + \delta\hat{L})x'' = J(\tilde{L}x + q_{\tilde{L}x})$, where $\delta\hat{L}$ is a small perturbation such that $\|\delta\hat{L}\|/\|\delta\hat{L}\| = O(\varepsilon_{\text{machine}})$, where $\varepsilon_{\text{machine}}$ is a small constant depending on the floating-point arithmetic, about 10^{-16} for double-precision IEEE-754 arithmetic, which is now used on virtually all computers. For the rectangular problem, backward stability means that x'' is the exact minimizer of $(\tilde{L} + \delta\tilde{L})x'' - (\tilde{L}x + q_{\tilde{L}x})$ for a similarly small perturbation.

Since $\hat{L}$ is a symmetric positive-definite matrix and since $\tilde{L}$ is full rank, most linear-equation solvers and most least-squares solvers are backward stable when applied to them. This includes sparse direct Cholesky factorization solvers for the square problem, sparse QR solvers for the rectangular least-squares problem, and most iterative algorithms for these problems.

When we obtain an approximation x'' using a backward stable algorithm, the relative norm of the so-called forward error $x' - x''$ is bounded by the *condition number* κ of the problem times $\varepsilon_{\text{machine}}$,

$$\frac{\|x'' - x'\|_2}{\|x'\|_2} = O(\kappa\varepsilon_{\text{machine}}) \,.$$

For the invertible problem, the condition number is simply the condition number of the coefficient matrix,

$$\kappa_{\text{inv}} = \kappa_2(L) = \|\hat{L}\| \, \|\hat{L}^{-1}\| \,.$$

When we use the 2-norm,

$$\kappa_{\text{inv}} = \frac{\sigma_{\max}(\hat{L})}{\sigma_{\min}(\hat{L})} \,.$$

The condition number of least-squares problems is a little more complicated. We denote by θ the angle between the right-hand side $(\tilde{L}x+q)$ (here $q=q_{\tilde{L}x}$) and its projection into the column space of $\tilde{L}$. Since $\tilde{L}x$ is in this column space, the size of $\tan\theta$ is roughly proportional to $\|q\|_2/\|\|\tilde{L}x\|$, which is proportional to how aggressive the quantization is. Therefore, $\tan\theta$ will be usually small. We denote by η the quantity

$$\eta = \frac{\|\tilde{L}\|_2\|x\|_2}{\|\tilde{L}x\|_2} .$$

This quantity is bounded by $1 \le \eta \le \kappa_2(\tilde{L})$. In our case, unfortunately, η will not be large, because $\tilde{L}x$ contains some values of x, namely the anchors, so its norm will not be much smaller than the norm of x. Given θ and η, we can express the condition number of solving least squares problems,

$$\kappa_{\text{rect}} = \kappa_2(\tilde{L}) + \frac{\kappa_2(\tilde{L})^2\tan\theta}{\eta} .$$

In our case, $\kappa_2(\tilde{L})$ and $\kappa_2(\hat{L})$ depend only on the small eigenvalue of $\hat{L}$, which we have already shown to be strongly influenced by the anchor points. Since $\sigma_{\max}(\hat{L}) = \lambda_{\max}(\hat{L}) \le 2d_{\max}+1$, where $d_{\max}$ is the maximal degree of a vertex in the mesh, and since $\sigma_{\max}(\tilde{L}) \le \sqrt{2}\lambda_{\max}(\hat{L})$, in both cases the largest singular value is bounded by a small constant, so $\kappa(L) = O(\lambda_{\min}^{-1}(\hat{L}))$ for both L's.

THEOREM 4.12. *Let $\lambda = \lambda_{\min}(\hat{L})$, $\varepsilon = \varepsilon_{machine}$, and $q = q_{\tilde{L}x}$. The 2-norm of the error $x - x''$, when x'' is computed from the invertible Laplacian using a backward-stable algorithm, is bounded by*

$$\|x-x''\|_2 \le O(\lambda^{-1}\|q\|_2 + \lambda^{-1}\varepsilon\|x\|_2 + \lambda^{-2}\varepsilon\|q\|_2) .$$

PROOF. By Lemma 4.9 we have

$$\begin{aligned}\|x'\|_2 &= \|x'+x-x\|_2 \\ &\le \|x-x'\|_2 + \|x\|_2 \\ &\le \lambda^{-1}\|q\|_2 + \|x\|_2 .\end{aligned}$$

The inequality and the discussion preceding the theorem yield

$$\begin{aligned}\|x-x''\|_2 &= \|x-x'+x'-x''\|_2 \\ &\le \|x-x'\| + \|x'-x''\|_2 \\ &\le \lambda^{-1}\|q\|_2 + \|x'-x''\|_2 \quad \text{by Lemma 4.9} \\ &\le \lambda^{-1}\|q\|_2 + O(\kappa_2(\hat{L})\varepsilon\|x'\|_2) \\ &= \lambda^{-1}\|q\|_2 + O(\lambda^{-1}\varepsilon(\lambda^{-1}\|q\|_2 + \|x\|_2)) \\ &= \lambda^{-1}\|q\|_2 + O(\lambda^{-1}\varepsilon\|x\|_2 + \lambda^{-2}\varepsilon\|q\|_2)\end{aligned}$$

□

We now state the corresponding theorem for the least squares case. The proof, which we omit, is identical except for the expression of the condition number.

THEOREM 4.13. *Let $\lambda = \lambda_{\min}(\hat{L})$, $\varepsilon = \varepsilon_{machine}$, and $q = q_{\tilde{L}x}$. The 2-norm of the error $x - x''$, when x'' is computed from the rectangular least-squares problem using a backward-*

stable algorithm, is bounded by

$$\|x-x''\|_2 \le O(\lambda^{-1}\|q\|_2+\lambda^{-1}\varepsilon\|x\|_2+\frac{\lambda^{-2}\tan\theta\varepsilon\|x\|_2}{\eta}+\lambda^{-2}\varepsilon\|q\|_2+\frac{\lambda^{-3}\tan\theta\varepsilon\|q\|_2}{\eta})\ .$$

One way of solving the least-squares problem is by constructing and solving the so-called *normal equations*. This solution method relies on the fact that the least-squares minimizer x' is also the solution of the symmetric positive-definite linear system $\tilde{L}^T\tilde{L}x' = \tilde{L}^T(\tilde{L}x+q_{\tilde{L}x})$. Even when the normal equations are solved using a backward-stable algorithm, the whole algorithm is not backward stable with respect to the original least-squares problem. The computed solution satisfies only

$$\frac{\|x''-x'\|_2}{\|x'\|_2} = O(\kappa_2(\hat{L})^2\varepsilon_{\text{machine}})\ ,$$

Because the error bound is much larger in this case (and usually much larger in practice), this method is usually not recommended.

However, since in our application we can control and estimate $\kappa_2(\hat{L})$ by adding anchor points, we can ensure that even the normal-equations forward error is acceptable.

4.6 Algorithms for placing anchor points

We considered two classes of algorithms for placing anchor points. The first algorithm, which is the one that we use in the experiments, is adaptive and greedy. We begin by placing one random anchor point and generating a 1-anchor rectangular Laplacian, denoted by $\tilde{L}_1$. We use this matrix to transform the coordinates, we quantize the relative coordinates, compute an approximation x_1'', and compute the error $x-x_1''$. We then place the second anchor at the vertex with the largest visual error, to yield $\tilde{L}_2$. We continue iteratively either until we obtain a satisfactory visual error, or until we place a given number k of anchors. We can accelerate this method to some extend by adding more anchors at random points, although the quality of the approximation will probably suffer. However, since typically we use only a small number of anchors, we found the fully adaptive algorithm to be effective.

Figure 7 visualizes the errors over the mesh. In Figure 7(a) only two anchors are used; the legs and the head of the horse exhibit some shift, indicated by the strong red and blue colors. In (c) twenty anchors are used, “nailing” the horse in place. As expected, the errors are nicely distributed.

It is also possible to devise anchor placement heuristics that select k anchors at once, and which aim to minimize the congestion-dilation-contention bound. We have not experimented with such heuristics.

5. THE VISUAL QUALITY MEASURE

The geometric error of a lossy mesh compression scheme can be measured by a per-vertex Euclidean distance $M_q(v_i) = \|v_i - Q(v_i)\|$, where $Q(v_i)$ is the decompressed position of vertex $v_i \in \mathbb{R}^3$. Then the root-mean-square (RMS) error that measures the “physical” distance between the original mesh and the decompressed mesh is:

$$M_q = \left(\sum_{i=1}^{n}\|v_i-Q(v_i)\|^2\right)^{1/2}\ .$$

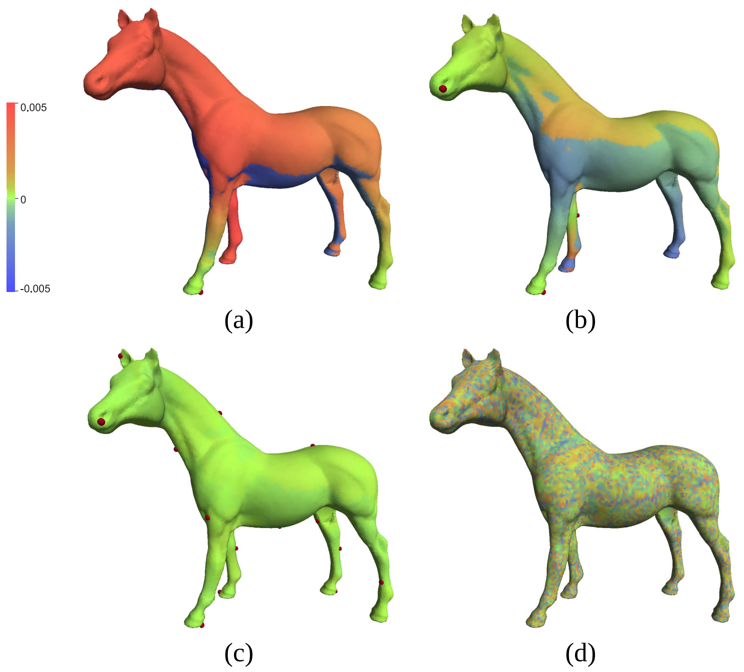


Fig. 7. Visualization of the visual error across the mesh surface. The surface was reconstructed from δ-coordinates quantized to 7 bits/coordinate using 2 anchors (a), 4 anchors (b) and 20 anchors (c). The anchor points are shown as small red balls. Each vertex v is colored according to its visual error value $E_{vis}(v)$. We have also added a sign to these values, so that vertices that move outside of the surface have positive error values (colored by red), and vertices that move inwards have negative error values (colored by blue). In (d), the visual error of direct quantization of the Cartesian coordinates is shown.

The M_q measure does not represent the visual quality of the mesh well, since the visual quality should be more sensitive to the surface local differential properties that define the surface appearance.

Karni and Gotsman [2000] suggest using a "geometric Laplacian" of a vertex v_i, that is, $S_q(v_i) = \|S(v_i) - S(Q(v_i))\|$, where $S(v_i)$ is the local differential measure of the smoothness at v_i:

$$S(v_i) = v_i - \frac{\sum\limits_{j \in N(i)} l_{ij}^{-1} v_j}{\sum\limits_{j \in N(i)} l_{ij}^{-1}}.$$

In this formula, l_{ij} is the Euclidean distance between v_i and v_j, and $N(i)$ is the set of indices of the neighbors of vertex v_i. Then:

$$S_q = \left(\sum_{i=1}^{n} \|S(v_i) - S(Q(v_i))\|^2 \right)^{1/2}.$$

They further combine M_q and S_q as the visual quality measure:

$$E_{vis} = \alpha M_q + (1-\alpha) S_q.$$

Karni and Gotsman used $\alpha = 0.5$. While this is a step towards a better visual measure, we argue that α must be smaller since S_q has a more significant visual effect. This is quite evident in Figure 9 (see the color section), where the surfaces in the middle column clearly look visually closer to the original models than the surfaces in the right column. From our informal empirical experiments, only when $\alpha \leqslant 0.15$, does E_{vis} agree with our perception. We acknowledge that this metric and this particular α value are ad-hoc. Ideally, compression methods should be evaluated using a quantitative visual measure backed by psychophysical evidence. Unfortunately, such evidence is not available, so an ad-hoc measure must be used. Our results can also be evaluated in a visual-metric-independent way as shown in Figure 9.

This agrees with the claim that our perception is more sensitive to lighting than geometry. A lighting distortion model would explain the phenomena depicted in Figure 1. When the tessellation is finer, the same displacements have more effect on the normals and so the distortion of the lighting is stronger.

Taubin [1995] shows that the δ-coordinates are an approximation of the vertex normal for curvature-continuous surfaces, and the norm of the vector of the δ-coordinates is an approximation of the mean curvature. Thus, quantization of the δ-coordinates provides direct control over the normals and curvatures, and consequently on the shading of the model.

It should be emphasized that for various CAD and engineering applications the geometric distance M_q must be accurate, and no loss of precision can be accepted. Moreover, defining a visual error that measures the human perception of 3D models is an open problem. We believe that, just like a similarity metric among shapes, the perception problem remains open, as it is subjective to the observers. We further discuss this in Section 7.

6. IMPLEMENTATION AND RESULTS

6.1 Quantization and compression

Our technique encodes the geometry of a 3D mesh by quantizing the δ-coordinates. We now explain how this fits into an overall coding/decoding framework.

We assume that the connectivity of the mesh is first encoded using a state-of-the-art connectivity encoder [Touma and Gotsman 1998; Alliez and Desbrun 2001]. Since the Laplacian is a trivial function of the connectivity, this encoding also represents the Laplacian of the mesh.

Next, we attach to the encoding the indices of the k anchor points, which are necessary for constructing rows $n+1,\dots,n+k$ of the k-rectangular Laplacian. This requires $k \log_2 n$ bits, which for reasonable values of k is insignificant.

The encoder then transforms the Cartesian coordinates x into δ-coordinates $\delta = Lx$. These transformed coordinates are quantized and compressed using an entropy-based encoder. Finally, we attach to the encoded δ-coordinates the original Cartesian coordinates of the k anchors, separately quantized and encoded. The compression ratio of the encoding of the anchors is fairly irrelevant, since k is typically less than one percent of n.

The decoder decompresses the connectivity and geometry data and solves the least-squares problem to recover an approximation of the Cartesian coordinates from the δ and anchor coordinates. As explained below, most of the computational effort in a least-squares solver involves a preprocessing step that depends only on $\tilde{L}$, but not on the coordinates data. Therefore, once the connectivity and the indices of the anchors are available to the decoder, it starts working. The bulk of its efforts can be completed before the compressed geometry data is available. This behavior allows the decoder to hide the latency of processing the geometry data.

6.2 Rate-distortion

To evaluate the performance of our scheme we generated a number of rate-distortion curves comparing our technique to the parallelogram scheme [Touma and Gotsman 1998], which we denote by TG. The parallelogram scheme is simple and known to perform well, thus, chosen to represent the general performance of a larger family of traversal-dependent predictors. The rate-distortion curves (see Figure 8) show the error measures as functions of the entropy. The entropy is a reasonable estimate of the compression ratios of high quality predictor-corrector compression schemes. In such schemes, including ours and TG, the predictors capture virtually all the spatial information in the data. The vector of predictors, therefore, behaves essentially like a vector of identically-distributed and independent random variables. For such vectors, the entropy is a provable lower bound on compression ratios. In addition, the publication of entropy data allows for an unbiased comparison of different predictor schemes, whereas actual compression ratios depend on both the prediction scheme and the specific entropy encoding that is used.

Figure 8 shows the curves of the M_q and S_q measures comparing our scheme and TG. The S_q curves of our scheme are clearly below those of TG, while the M_q curves are usually above. As argued in Section 5, the S_q measure is more visually important. This is further supported by Figure 9 in a metric-independent way. The figure shows a series of pairs of models quantized into about the same entropy by the two approaches. The visual comparisons are samples of the rate-distortion curves at a given entropy.

The tables in Figures 10 and 11 demonstrate the behavior of our method when different δ-quantization levels are applied, compared to adding more anchor vertices. Each row of the table depicts the results when various numbers of anchors are used, for a fixed number of bits/coordinate.

6.3 Solving least-squares problems

Decompressing a mesh function in our method requires solving a linear least-squares problem. Fortunately, there are efficient algorithms that can solve such problems very quickly. In this section we briefly mention some of these algorithms, provide some sample performance data, and explain how the quantization and compression methods can be tailored to ensure fast decompression. For a more complete discussion of algorithms for sparse linear least-squares problems, see Björck's monograph [Björck 1996].

Sparse least-squares solvers fall into two categories, direct and iterative. Most direct solvers factor the coefficient matrix $\tilde{L}$ into a product of an orthonormal matrix Q and an upper triangular matrix R, $\tilde{L} = QR$. Once the factorization is computed, the minimizer $\hat{x}$ of $\left\|\tilde{L}x - b\right\|_2$ is found by solving the triangular linear system of equations $R\hat{x} = Q^T b$. This algorithm is backward stable. The matrix R is typically very sparse, although not as sparse as $\tilde{L}$; it is represented explicitly in such algorithms. In particular, since in our case the meshes are almost planar graphs and have small vertex separators, R is guaranteed to remain sparse [George and Ng 1988]. The matrix Q is not as sparse, but it is has a sparse representation as a product of elementary orthogonal factors [George and Heath 1980;

George and Ng 1986]. To reduce the work and storage required for the factorization, the columns of the input matrix $\tilde{L}$ are usually reordered prior to the factorization [Brainman and Toledo 2002; Davis et al. 2000; George and Ng 1983; Heggernes and Matstoms 1996].

Another class of direct solvers, which is normally considered numerically unstable, uses a triangular factorization of the coefficient matrix $\tilde{L}^T\tilde{L}$ of the so-called normal equations. Once triangular factor R is found (it is mathematically the same R as in the $\tilde{L} = QR$ factorization), the minimizer is found by solving two triangular linear systems of equations, $R^T(R\hat{x}) = \tilde{L}^T b$. This procedure is faster than the QR procedure, but produces less accurate solutions, because solving the normal equations is not backward stable. However, the accuracy of the solutions depends on the condition number of $\tilde{L}$ (ratio of extreme singular values), and as we have shown in Section 4.5, the matrix $\tilde{L}$ is well-conditioned thanks to the anchors, so in this case solving the normal-equations procedure yields accurate solutions.

The running times and storage requirements of direct solvers can be further reduced by cutting the mesh into patches, as proposed by Karni and Gotsman [Karni and Gotsman 2000], and solving on each patch separately. All the boundary vertices are then considered anchors, to ensure that the solutions on different patches are consistent. We believe that this optimization would not be usually needed, and that problems involving entire meshes can be solved efficiently, but we mention it as a way of handling extremely large cases. Note that to ensure that the patches are consistent, the k-anchor invertible Laplacian would need to be used here, not the k-anchor rectangular Laplacian.

In all direct methods, the factorization is computed once and used to solve for multiple mesh functions. Most of the time is spent in computing the factorization, and the cost of solving for a minimizer is negligible. Therefore, the cost of decompression using these methods is almost independent of the number of mesh functions (x, y, z, and perhaps other information, such as color).

Direct methods are fast. Table I records the solution times for the models used in our experiments. The table shows the time to decompose the coefficient matrix of the normal equations into its triangular factors, and the subsequent solution time for one mesh function. For example, computing the triangular factorization of the horse, a model with 19,851 vertices, took 0.9 seconds on a 2.4 GHz Pentium 4 computer, and solving for a single mesh function took 0.032 seconds once the factorization has been computed. The linear solver that we used for these experiments is TAUCS version 2.2 [Toledo 2003], which uses internally two additional libraries, ATLAS version 3.4.1 [Whaley et al. 2000] and METIS version 4.0 [Karypis and Kumar 1998]. TAUCS and METIS were compiled using the Intel C/C++ compiler version 7.1 for Linux, and ATLAS was compiled using GCC version 2.95.2. The options to the compilers included optimization options (`-O3`) and Pentium-4-specific instructions (`-xW` for the Intel compiler and inlined assembly language in ATLAS). For additional performance evaluations of TAUCS, see [Irony et al. 2002; Rotkin and Toledo 2003].

We did not have a code of similar performance for computing the sparse QR factorization, but we estimate that it should be about 4–6 times slower.

Even though direct methods are fast, their running times usually scale superlinearly with the size of the mesh. Iterative solvers least-squares solvers, which do not factor the coefficient matrix, sometimes scale better than direct methods. Perhaps the most widely-used least-squares iterative solver is LSQR, which is based on a Krylov bidiagonalization procedure [Paige and Saunders 1982b; 1982a]. Other popular solvers in-

Table I. Running times of solving the linear least-squares systems for the different models. Most time is spent on the factorization of the coefficient matrix, which can be done during the transmission of the δ-coordinates. Solving for a single mesh function (x, y or z) takes only a negligible amount of time (see rightmost column). The experimental setup is described in the text.

Model	Number of vertices	Factorization (sec.)	Solving (sec.)
Eight	2,718	0.085	0.004
Twirl	5,201	0.098	0.006
Horse	19,851	0.900	0.032
Fandisk	20,111	1.091	0.040
Camel	39,074	2.096	0.073
Venus	50,002	3.402	0.112
Max Planck	100,086	7.713	0.240

clude CGLS, a conjugate-gradients algorithm for solving the normal equations [Björck and Elfving 1979; Elfving 1978], and CRAIG, an error-minimization bidiagonalization procedure [Craig 1955]; see also [Paige and Saunders 1982b; Saunders 1995].

The convergence of these methods depends on the distribution of the singular values of the coefficient matrix $\tilde{L}$, as well as on the initial approximation. In our $\tilde{L}$ is always well conditioned, so we can expect reasonably rapid convergence. Furthermore, the decoder knows the the values of the mesh function at the anchor vertices. By interpolating these values at non-anchor vertices, the decoder can quickly produce a good initial approximation (note, however, that even at the anchor points, the known values of the original mesh function at the anchors need not coincide with the values of the least-squares minimizer).

The iterative methods mentioned above can be accelerated by using a preconditioner, (informally, an approximate inverse of $\tilde{L}$). The relationship of our coefficient matrix $\tilde{L}$ to a graph Laplacian can be probably exploited when constructing a preconditioner, since highly effective preconditioners have been discovered for Laplacians. The most important classes of preconditioners for Laplacians are algebraic multigrid preconditioners [Brandt et al. 1984], incomplete Cholesky preconditioners [Gustafsson 1978; Meijerink and van der Vorst 1977], and more recently, support preconditioners [Boman and Hendrickson 2001; Chen and Toledo 2003; Vaidya 1991]. For further information about iterative solvers and preconditioning, see [Axelsson 1994; Saad 1996; Barret et al. 1993], and of course, [Björck 1996].

We have not yet compared the performance of iterative and direct solvers for our application, but we expect that for large models, iterative solvers would outperform direct ones.

For extremely huge systems, the running times and storage requirements of direct solvers can be further reduced by cutting the mesh into patches, as proposed by Karni and Gotsman [2000], and solving on each patch separately. All the boundary vertices are then considered anchors, to ensure that the solutions on different patches are consistent.

6.4 Discussion

Instead of directly pre-quantizing the Cartesian coordinates, one could first compute the predictions in floating-point precision and then quantize the offsets from the predictions [Chou and Meng 2002]. A naive implementation of such scheme would result in accumulating error along the traversal path. Therefore, the offsets are rounded in such a way that after decompression the M_q error is kept bounded.

This method is computationally cheaper than solving a least-squares system, but it does

Table II. Comparison of the entropies of the offsets using different multi-way predictors. All models where pre-quantized to 12 bits/coordinate. Clearly, adding more prediction directions makes the prediction better and lowers the entropy of the offsets.

Model	1-way	2-way	3-way	4-way	5-way	6-way	All-way
Eight	16.9212	15.2487	14.3161	14.2064	13.8455	13.5434	13.5285
Horse	15.7501	14.9716	14.2978	13.7994	14.4517	13.2583	13.1568
Fandisk	12.1724	10.7292	9.8037	9.3536	8.8192	8.3925	8.3369
Venus	14.5413	13.4164	12.7131	12.1663	11.7758	11.5464	11.4519
Max Planck	10.2935	8.5432	7.6266	6.8253	6.1680	5.8708	5.7795

not possess the same properties as the Laplacian transform. In particular, the spectrum of the resulting quantization error will contain high-frequency and not low frequency modes.

Moreover, a predictor that takes into account known locations from many directions, yields better predictions than that based on only one direction or just few. A prediction based on the Laplacian uses all possible directions and in general yields better prediction than the 1-way parallelogram rule. This fact is demonstrated in Table II, where the entropy of the offsets is computed using a single or several known positions. To compute the entropy of a multi-way predictor, we take several 1-way predictions around the vertex and average them. The offset is then taken from that averaged prediction. The more directions are used for prediction, the better the prediction is, and the entropy of the offsets is lower (see also [Gumhold and Amjoun 2003]). However, multi-way predictor cannot be employed by traversal-dependent schemes. Our δ-coordinates serve in a sense as the offsets of all-way predictor.

7. OPEN PROBLEMS

Our research raises a number of interesting open problems for future research.

(1) Evaluating lossy mesh-compression methods requires a quantitative visual-quality metric that is widely accepted and agreed upon, but no such metric exists. Ideally, the validity of such a metric should be established using psychophysic evidence.

(2) Can we compress the quantized δ-coordinates into less space than predicted by their entropy? Our overall compression scheme relies on a transformation of the vertex coordinates, quantization, and compression. If the coordinates of nearby mesh vertices remain correlated in some way even after transformation and quantization, the compressor can probably exploit these correlations to achieve high compression ratios. If there are no remaining correlations, on the other hand, then the entropy of the quantized transformed coordinates, which is the metric that we reported in this paper, indeed determines the best possible compression ratio.

(3) Can one rigorously analyze the behavior of the eigenvectors of the Laplacian of 3D meshes? Our method works because for a vector x of mesh coordinates, the norm of the Lx tends to be much smaller than the norm of x. This happens because x most of the energy of x is concentrated in the subspace of R^n that is spanned by the eigenvectors of L that correspond to small eigenvalues. But does this always happen? The answer depends on the relationship between the eigenvectors of the Laplacian and typical mesh-coordinate vectors. Ben-Chen and Gotsman have done the first step towards resolving this question [Ben-Chen and Gotsman 2003]. They have shown that under certain probabilistic assumptions on the shape of 3D meshes, most of the energy of the mesh-coordinate vectors indeed lies in the subspaces spanned by the small eigen-

vectors. Another analysis of Laplacian eigenvectors, done by Guattery and Miller in a completely different context, may provide another perspective on the issue [Guattery and Miller 2000].

(4) Can one solve the least-squares problems that arise in our method in time linear or almost linear in the size of the mesh? We have shown reasonably small running times even for large meshes, but our solution method scales superlinearly. It would be useful to find solution methods with better scaling. Algebraic multigrid methods can almost certainly solve the invertible k-anchor Laplacian equations in $O(n)$ work, and we have begun experiments in that direction. We are not yet sure whether algebraic multigrid methods can also effectively solve the least-squares problem arising from the rectangular Laplacian. Another direction might be an iterative solver, such as LSQR or CGLS, coupled with an effective preconditioner. In particular, it would be interesting to know whether graph-theoretical preconditioners, such as support-tree [Gremban 1996; Gremban et al. 1995] and support-graph [Bern et al. 2001; Boman et al. 2001; Spielman and Teng 2003; Vaidya 1991] preconditioners could be adapted to this task.

8. CONCLUSIONS

This paper addressed the issue of reducing the visual effects of aggressive quantization of 3D meshes. We argue that low-frequency quantization errors are less distracting than high-frequency errors. We presented a quantization method that achieves this goal and results in mostly low-frequency errors. Our approach is in complete contrast to that of common wisdom and practice, which aims at preserving low-frequency information while discarding high-frequency data.

While it is true that 3D models often contain high-frequency noise that can be safely discarded, further aggressive compression should introduce mostly low-frequency errors. Indeed, models produced by high-resolution input devices are often denoised. Denoising is basically a sophisticated low-pass filter that preserves important high-frequency features, such as folds and corners. We claim that the remaining high-frequency data in the model is an essential part of its visual appearance, and in particular, it is more important to preserve it than to preserve low-frequency data. This is exactly what our quantization method does. In contrast, most other mesh compression techniques continue to erode the high-frequency data, which quickly degrades the appearance of the model, but fail to exploit the insensitivity of the human eye to moderate low-frequency errors.

We feel that it is premature to quantitatively compare the distortion introduced by different lossy mesh-compression methods, since none of the proposed visual-error metrics has yet been convincingly shown to correlate with human perception. We suggest that the spectrum of the error is essential for understanding distortion; that moderate low-frequency errors are usually acceptable, whereas high-frequency errors beyond some threshold are not. Obviously, there are applications and situations in which low-frequency errors are unacceptable, such as mechanical CAD or, for example, almost-touching features. Nonetheless, we have shown that our method performs well using a visual-quality metric based on the one introduced by Karni and Gotsman.

Fundamentally, our main contribution is a technique for manipulating 3D models in the frequency domain. We use this technique to effectively quantize the geometry of models. Others are using similar techniques in other applications, such as watermarking [Ohbuchi et al. 2002]. We believe that the ability to manipulate 3D models in the frequency domain

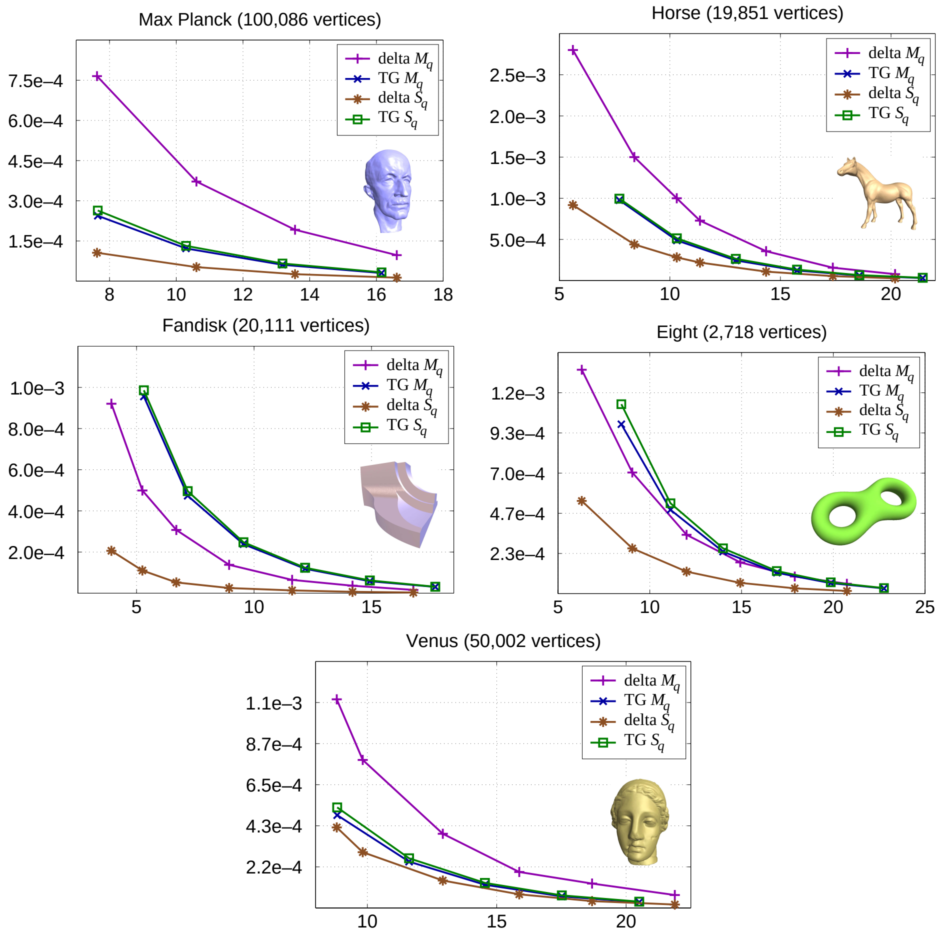
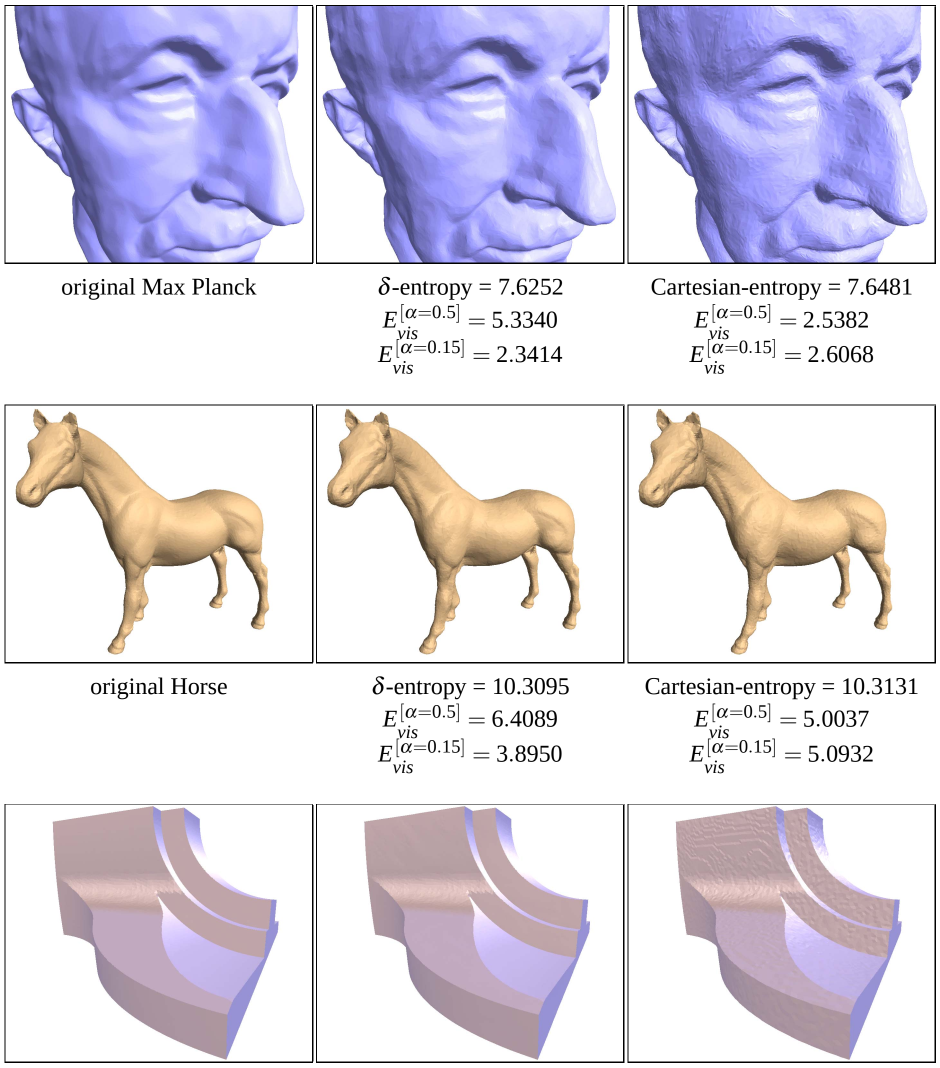


Fig. 8. Rate distortion curves for five known models. The graphs show the M_q and S_q measures as functions of the entropy, for δ-coordinates and the TG method.

will find additional applications in the future.

ACKNOWLEDGMENTS

We would like to thank Tali Irony for helping us with the implementation. The Max-Planck and Fandisk models are courtesy of Christian Rössl and Jens Vorsatz of Max-Planck-Institut für Informatik. This work was supported in part by grants from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), by the Israeli Ministry of Science, by an IBM Faculty Partnership Award, by the German Israel Foundation (GIF) and by the EU research project ‘Multiresolution in Geometric Modelling (MINGLE)’ under grant HPRN-CT-1999-00117.



original Max Planck	δ-entropy = 7.6252 $E_{vis}^{[\alpha=0.5]} = 5.3340$ $E_{vis}^{[\alpha=0.15]} = 2.3414$	Cartesian-entropy = 7.6481 $E_{vis}^{[\alpha=0.5]} = 2.5382$ $E_{vis}^{[\alpha=0.15]} = 2.6068$
original Horse	δ-entropy = 10.3095 $E_{vis}^{[\alpha=0.5]} = 6.4089$ $E_{vis}^{[\alpha=0.15]} = 3.8950$	Cartesian-entropy = 10.3131 $E_{vis}^{[\alpha=0.5]} = 5.0037$ $E_{vis}^{[\alpha=0.15]} = 5.0932$
original Fandisk	δ-entropy = 6.6912 $E_{vis}^{[\alpha=0.5]} = 1.7971$ $E_{vis}^{[\alpha=0.15]} = 0.9479$	Cartesian-entropy = 7.1767 $E_{vis}^{[\alpha=0.5]} = 4.8479$ $E_{vis}^{[\alpha=0.15]} = 4.9263$

Fig. 9. Comparison of visual quality of the meshes using δ-coordinates quantization vs. standard Cartesian coordinates quantization. The original meshes are shown in the left column, meshes with quantized δ-coordinates in the middle column, and meshes with quantized Cartesian coordinates in the right column. Note that the entropy of the δ-coordinates is slightly lower than the entropy of parallelogram-prediction displacements, while visually, the surfaces look closer to the original. Using E_{vis} with $\alpha = 0.5$ (denoted by $E_{vis}^{[\alpha=0.5]}$), most models in the right column have a smaller error, while clearly the ones in the middle column seem to have a better visual quality. Only when using $\alpha = 0.15$, E_{vis} indeed agrees with our perception. The error values are given in units of 10^{-4}.

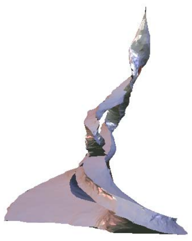
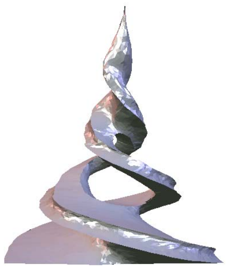
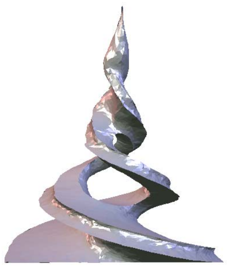
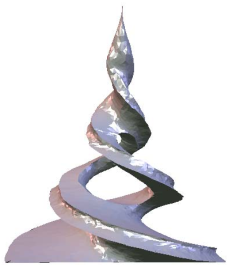
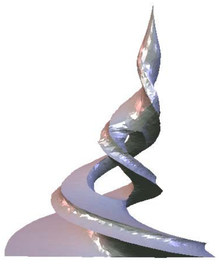
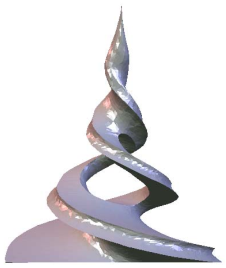
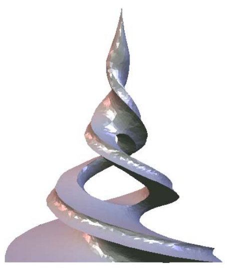
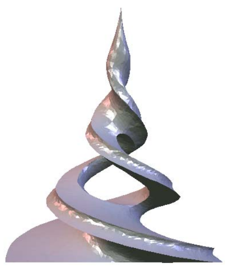
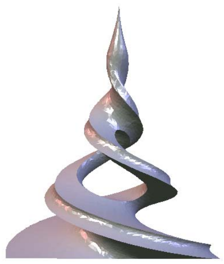
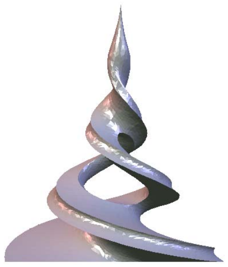
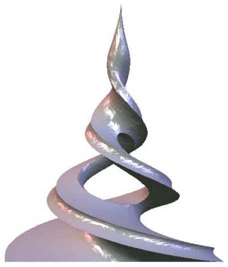
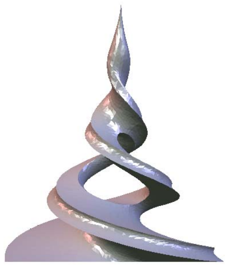
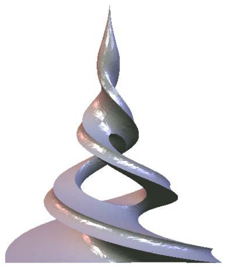
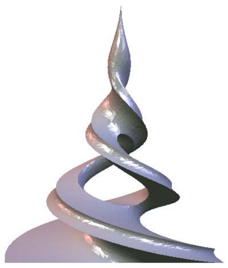
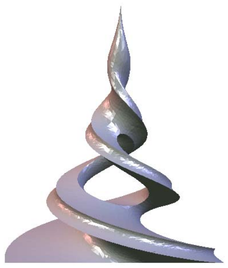
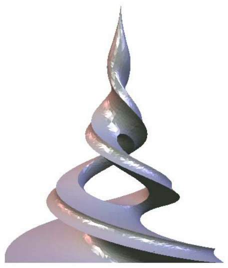
	1 anchor	15 anchors	30 anchors	45 anchors
6 bits				
7 bits				
8 bits				
9 bits				

Fig. 10. Visual table of quantization results for the Twirl model (5201 vertices). The vertical axis corresponds to the number of bits per coordinate used in δ-quantization. The horizontal axis corresponds to the number of anchor points used.

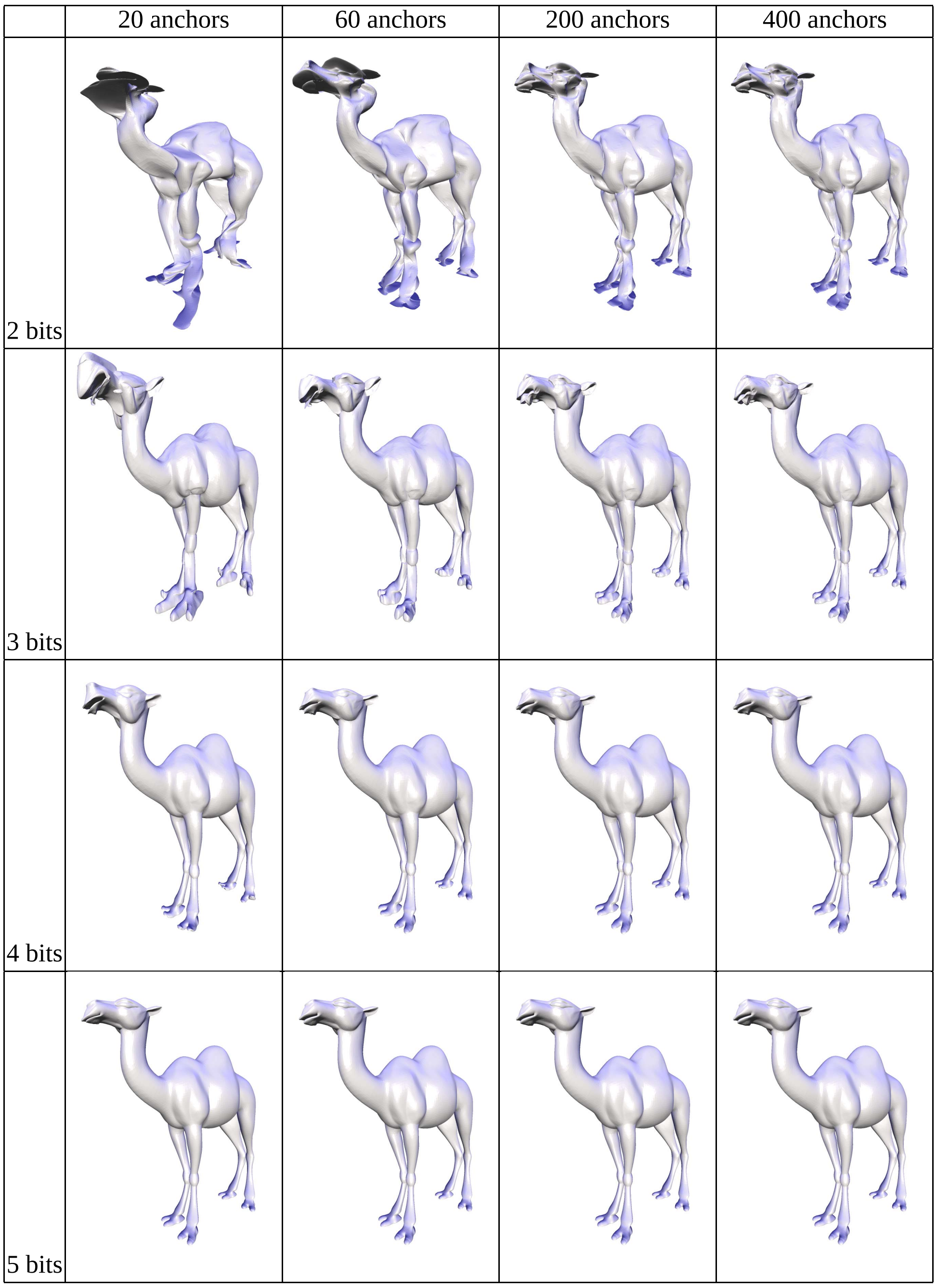


Fig. 11. Visual table of quantization results for the Camel model (39074 vertices). The vertical axis corresponds to the number of bits per coordinate used in δ-quantization. The horizontal axis corresponds to the number of anchor points used.

REFERENCES

ALEXA, M. 2001. Local control for mesh morphing. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI-01)*, B. Werner, Ed. IEEE Computer Society, Los Alamitos, CA, 209–215.

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*. In press.

ALLIEZ, P. AND DESBRUN, M. 2001. Valence-driven connectivity encoding for 3D meshes. *Computer Graphics Forum 20,* 3, 480–489.

AXELSSON, O. 1994. *Iterative Solution Methods*. Cambridge University Press.

BARRET, R., BERRY, M., CHAN, T., DEMMEL, J., DONATO, J., DONGARRA, J., EIJKHOUT, V., POZO, R., ROMINE, C., AND VAN DER VORST, H. 1993. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadeplphia, PA.

BEN-CHEN, M. AND GOTSMAN, C. 2003. On the optimality of spectral compression of meshes. Preprint, available online from `http://www.cs.technion.ac.il/~gotsman/publications.html`.

BERN, M., GILBERT, J. R., HENDRICKSON, B., NGUYEN, N., AND TOLEDO, S. 2001. Support-graph preconditioners. Submitted to the *SIAM Journal on Matrix Analysis and Applications*, 29 pages.

BJÖRCK, Å. 1996. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia.

BJÖRCK, Å. AND ELFVING, T. 1979. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT 19*, 145–163.

BOMAN, E. G., CHEN, D., HENDRICKSON, B., AND TOLEDO, S. 2001. Maximum-weight-basis preconditioners. To appear in *Numerical Linear Algebra with Applications*, 29 pages.

BOMAN, E. G. AND HENDRICKSON, B. 2001. Support theory for preconditioning. Tech. Rep. SAND-01-1794J, Sandia National Labs. Mar.

BRAINMAN, I. AND TOLEDO, S. 2002. Nested-dissection orderings for sparse LU with partial pivoting. *SIAM Journal on Matrix Analysis and Applications 23*, 998–112.

BRANDT, A., MCCORMICK, S. F., AND RUGE, J. 1984. Algebraic multigrid (AMG) for sparse matrix equations. In *Sparsity and its Applications*, D. J. Evans, Ed. Cambridge University Press, 257–284.

BRIGGS, W. L., HENSON, V. E., AND MCCORMICK, S. F. 2000. *A Multigrid Tutorial*, 2 ed. SIAM, Philadelphia.

CHEN, D. AND TOLEDO, S. 2003. Vaidya's preconditioners: Implementation and experimental study. *Electronic Transactions on Numerical Analysis 16*, 30–49.

CHOU, P. H. AND MENG, T. H. 2002. Vertex data compression through vector quantization. *IEEE Transactions on Visualization and Computer Graphics 8,* 4, 373–382.

CHUNG, F. R. K. 1997. *Spectral Graph Theory*. American Methematical Society.

CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. 1998. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17,* 2, 167–174.

CRAIG, E. J. 1955. The *n*-step iteration procedure. *J. Math. Phys. 34*, 65–73.

DAVIS, T. A., GILBERT, J. R., LARIMORE, S. I., AND NG, E. G. 2000. A column approximate minimum degree ordering algorithm. Tech. Rep. TR–00–005, Department of Computer and Information Science and Engineering, University of Florida. Oct. Submitted for publication.

DEERING, M. F. 1995. Geometry compression. In *SIGGRAPH 95 Conference Proceedings,* R. Cook, Ed. Annual Conference Series. ACM SIGGRAPH, Addison Wesley, 13–20. held in Los Angeles, California, 06-11 August 1995.

ELFVING, T. 1978. On the conjugate gradient method for solving linear least squares problems. Tech. Report LiTH-MAT-R-78-3, Linköping University, Sweden.

FIEDLER, M. 1973. Algebraic connectivity of graphs. *Czech. Math. Journal 23*, 298–305.

GEORGE, J. A. AND HEATH, M. T. 1980. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra Appl. 34*, 69–83.

GEORGE, J. A. AND NG, E. G. 1983. On row and column orderings for sparse least squares problems. *SIAM J. Numer. Anal. 20,* 326–344.

GEORGE, J. A. AND NG, E. G. 1986. Orthogonal reduction of sparse matrices to upper triangular form using Householder transformations. *SIAM J. Sci. Statist. Comput. 7,* 460–472.

GEORGE, J. A. AND NG, E. G. 1988. On the complexity of sparse QR and LU factorization of finite-element matrices. *SIAM J. Sci. Statist. Comput. 9,* 849–861.

GOLUB, G. H. AND LOAN, C. F. V. 1996. *Matrix Computations*, 3rd ed. Johns Hopkins University Press.

GREMBAN, K., MILLER, G., AND ZAGHA, M. 1995. Performance evaluation of a parallel preconditioner. In *9th International Parallel Processing Symposium*. IEEE, Santa Barbara, 65–69.

GREMBAN, K. D. 1996. Combinatorial preconditioners for sparse, symmetric, diagonally dominant linear systems. Ph.D. thesis, School of Computer Science, Carnegie Mellon University. Technical Report CMU-CS-96-123.

GUATTERY, S. AND MILLER, G. L. 2000. Graph embeddings and laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications 21*.

GUMHOLD, S. 2000. New bounds on the encoding of planar triangulations. Technical Report WSI–2000–1, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany. January.

GUMHOLD, S. AND AMJOUN, R. 2003. Higher order prediction for geometry compression. *International Conference On Shape Modelling And Applications*. Accepted for publication.

GUMHOLD, S. AND STRASSER, W. 1998. Real time compression of triangle mesh connectivity. In *SIGGRAPH 98 Conference Proceedings*, M. Cohen, Ed. Annual Conference Series. ACM SIGGRAPH, Addison Wesley, 133–140.

GUSTAFSSON, I. 1978. A class of first-order factorization methods. *BIT 18*, 142–156.

HEGGERNES, P. AND MATSTOMS, P. 1996. Finding good column orderings for sparse QR factorizations. Tech. Report LiTH-MAT-1996-20, Department of Mathematics, Linköping University, Sweden.

HIGHAM, N. J. 2002. *Accuracy and Stability of Numerical Algorithms*, 2 ed. SIAM, Philadelphia.

IRONY, D., SHKLARSKI, G., AND TOLEDO, S. 2002. Parallel and fully recursive multifrontal supernodal sparse cholesky. To appear in *Future Generation Computer Systems*, 16 pages.

KARNI, Z. AND GOTSMAN, C. 2000. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000*. Computer Graphics Proceedings, Annual Conference Series. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 279–286.

KARYPIS, G. AND KUMAR, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing 20*, 359–392. The code is available online at `http://www-users.cs.umn.edu/~karypis/metis/`.

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive geometry compression. In *Siggraph 2000, Computer Graphics Proceedings*, K. Akeley, Ed. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 271–278.

KING, D. AND ROSSIGNAC, J. 1999. Optimal bit allocation in compressed 3D models. *Computational Geometry, Theory and Applications 14*, 1-3, 91–118.

MEIJERINK, J. A. AND VAN DER VORST, H. A. 1977. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathemathics of Computation 31*, 148–162.

OHBUCHI, R., MUKAIYAMA, A., AND TAKAHASHI, S. 2002. A frequency-domain approach to watermarking 3d shapes. *Computer Graphics Forum 21*, 3, 373–382.

PAIGE, C. C. AND SAUNDERS, M. A. 1982a. Algorithm 583 LSQR: Sparse linear equations and sparse least squares. *ACM Trans. Math. Software 8*, 195–209.

PAIGE, C. C. AND SAUNDERS, M. A. 1982b. LSQR. An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software 8*, 43–71.

POTHEN, A., SIMON, H. D., AND LIOU, K.-P. 1990. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis 11*, 430–452.

ROSSIGNAC, J. 1999. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5,* 1 (January - March), 47–61. ISSN 1077-2626.

ROTKIN, V. AND TOLEDO, S. 2003. The design and implementation of a new out-of-core sparse Cholesky factorization method. To appear in *ACM Transactions on Mathematical Software*.

SAAD, Y. 1996. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company.

SAUNDERS, M. A. 1995. Solution of sparse rectangular systems using LSQR and CRAIG. *BIT 35*, 588–604.

SORKINE, O., COHEN-OR, D., AND TOLEDO, S. 2003. High-pass quantization for mesh encoding. In *Proceedings of ACM/Eurographics Symposium on Geometry Processing*. Aachen, Germany.

SPIELMAN, D. AND TENG, S.-H. 2003. Solving sparse, symmetric, diagonally-dominant linear systems in time $o(m^{1.31})$. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. Cambridge, MA.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 95*. Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH / Addison Wesley, Los Angeles, California, 351–358.

TAUBIN, G. AND ROSSIGNAC, J. 1998. Geometric compression through topological surgery. *ACM Transactions on Graphics 17,* 2 (April), 84–115.

TOLEDO, S. 2003. TAUCS*: A Library of Sparse Linear Solvers, version 2.2*. Tel-Aviv University, Available online at `http://www.tau.ac.il/~stoledo/taucs/`.

TOUMA, C. AND GOTSMAN, C. 1998. Triangle mesh compression. In *Graphics Interface '98*. 26–34.

TREFETHEN, L. N. AND BAU, III, D. 2000. *Numerical Linear Algebra*. SIAM, Philadelphia.

VAIDYA, P. M. 1991. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Unpublished manuscript. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis.

WHALEY, R. C., PETITET, A., AND DONGARRA, J. J. 2000. Automated empirical optimization of software and the ATLAS project. Tech. rep., Computer Science Department, University Of Tennessee. The code is available online at `http://math-atlas.sourceforge.net`.