Eurographics Symposium on Geometry Processing (2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

High-Pass Quantization for Mesh Encoding

Olga Sorkine, Daniel Cohen-Or and Sivan Toledo

Raymond and Beverly Sackler Faculty of Exact Sciences, School of Computer Science
Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978 Israel

Abstract

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 "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 some control over the shape and magnitude of the low-frequency quantization errors, we extend the Laplacian matrix by adding a number of spatial constraints. This enables us to tailor the quantization process to specific visual requirements, and to strongly quantize the δ-coordinates.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—curve, surface, solid and object representations

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 [3, 22], 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 quantization 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 [20, 22]. 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 [16]. 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 Carte-

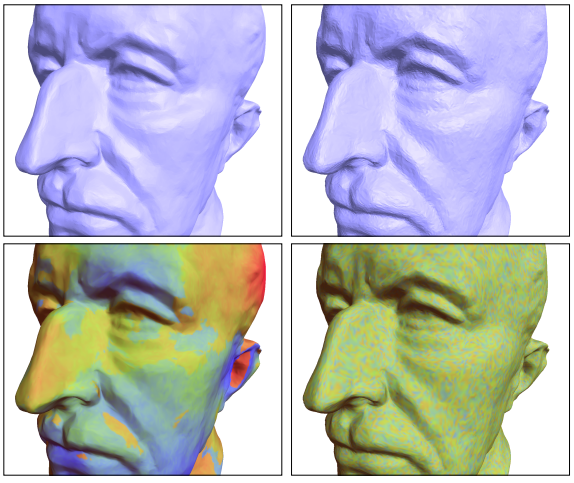

Figure 1: *The δ-coordinates quantization to 5 bits/coordinate (left) introduces low-frequency errors, whereas Cartesian quantization to 11 bits/coordinate (right) introduces noticeable errors. The upper row shows the quantized model, and the bottom figures use color to visualize the corresponding quantization errors.*

sian 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. Aggressive quantization significantly alters the surface normals, causing the irritating "jaggies" effect. 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 [8]) 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 4 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 rest of the paper is organized as follows. We review previous work related to the mesh Laplacian and the field of geometry compression in Section 2. Section 3 shows the properties of the Laplacian matrix and the δ-coordinates, which enable their strong quantization. In Section 4 we describe the visual error metric that better captures the visual distance between meshes. Section 5 presents implementation details and the results, and we conclude in Section 6.

2. Related work

Following the pioneering work of Deering [9] and Taubin and Rossignac [20], numerous mesh-compression techniques have been developed, which focus mainly on connectivity encoding (e.g., [13, 19, 22]). It has been shown that the efficiency of the connectivity encoding has reached near-optimal level [3, 14]. 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 $v_1, ..., v_n$ and each vertex v_i is encoded based on the known locations of the previous vertices in the sequence $v_1, ..., v_{i-1}$. The unknown location of v_i is predicted to be at $\hat{v}_i$, and the displacement $e_i = \hat{v}_i - v_i$ is encoded. Usually linear predictors are used; the most common one is known as the parallelogram predictor [22].

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 [6] 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 [15], where the compression results are measured in terms of visual quality.

The mesh-compression method of Karni and Gotsman [15] 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 8 in the color section). 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 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 [17]. 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 [21] 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 [1, 2] 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. [18] 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 bit-string. 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 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,

$$x = c_1u_1 + c_2u_2 + \cdots + c_nu_n,$$

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 $\max_i |(Ax)_i| \ll \max_i |x_i|$. Assuming a uniform quantization, this property 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.

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 [10]. Laplacians of meshes have been extensively studied [7], 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. We will later explore 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

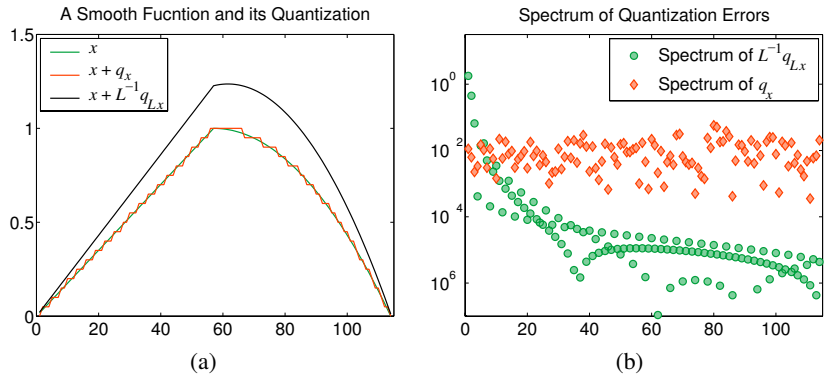

Figure 2: *An example showing quantization errors in a one-dimensional mesh. The mesh here is a simple linear chain with 114 vertices. (a) shows a smooth function x defined on the mesh, its direct quantization, and a Laplacian-transform quantization. The quantizations were performed with 20 discrete 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.*

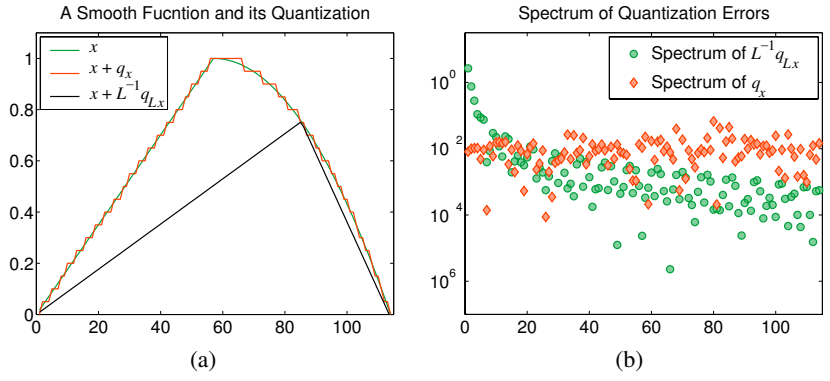

Figure 3: *The same mesh as in Figure 2, but with an additional anchor point at vertex 86. The transformed quantization error is no longer smooth at the anchor point, even though the vector x is smooth there.*

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. 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 Lapacian 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) [5, 11]. For a typical n-vertex 3D mesh, the small eigenvalue is there-

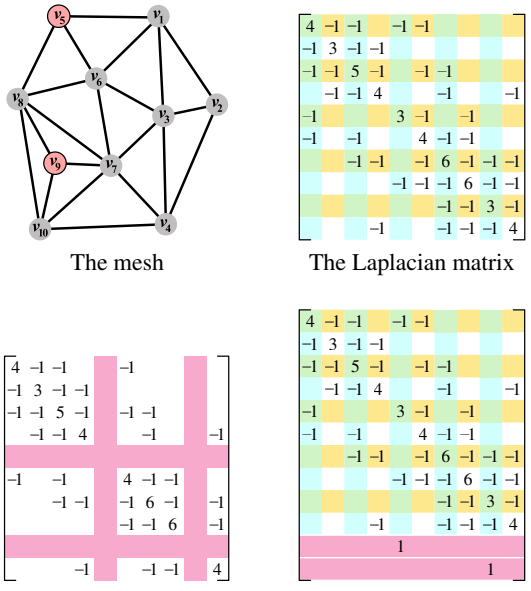

Figure 4: *A small example of a triangular mesh and its associated Laplacian matrix (top right). Second row: a 2-anchor invertible Laplacian and a 2-anchor rectangular Laplacian. The anchors are denoted in the mesh by circles.*

fore likely to be $\Theta(n^{-1.5})$. This causes large low-frequency errors which are clearly visible in the example in Figure 2.

3.3. The k-anchor rectangular Laplacian

An effective way to increase the small eigenvalue of a Laplacian is to add more anchor points. With k anchor points in a connected mesh, we can bound $|\lambda_n^{-1}|$ in the following way. Assign each vertex i to an anchor point m, denote the neighborhood of m by $N_m = \{i : m \text{ is the anchor of } i\}$, and denote $R_m = \max_{i \in N_m}\{\text{distance between } i \text{ and } m\}$. We then have $|\lambda_n^{-1}| \leq \max_m |N_m| R_m$. Given a set of anchor points, this inequality provides an easily-computed a-priori upper bound on the norm of the quantization error.

Adding anchor points and dropping the corresponding rows and columns from the Laplacian increases the smallest eigenvalue of the transformation and reduces the magnitude of the transformed quantization errors, but this method has a serious defect. At the anchor points, the transformed quantization errors are not smooth, as illustrated in Figures 3 and 5.

We propose to fix this defect. The crucial observation is that the k-anchor invertible Laplacian essentially forces the transformed quantization error to be zero at the anchors, and allows the error to grow as we get further and further away from the anchor points. We suggest using a *k-anchor rectangular Laplacian*, which is an $(n+k)$-by-n matrix $\tilde{L}$, with the singular Laplacian in the first n rows, and unit vectors corresponding to the anchor points in the last k rows (see Figure 4). We transform the coordinates of the mesh simply by multiplying x by $\tilde{L}$, $w = \tilde{L}x$. The vector that represents x is now a vector of n Laplacian coordinates, and k standard coordinates of the anchor points. We quantize the two groups of coordinates separately, to reap the benefits of the small values of the Laplacian coordinates.

To recover x from $w + q_w$, we solve the least-squares problem $\min_x \left\|\tilde{L}x - (w+q_w)\right\|_2$. It is well known that the solution of the least-squares problem is the solution of the so-called normal equations $\tilde{L}^T\tilde{L}x = \tilde{L}^T(w+q_w)$, and that it is unique if there is at least one anchor point in each connected component of the mesh. Intuitively, the least-squares solution generates smooth errors since it allows the computed solution to differ from the given input even at the anchor points. The magnitude of the quantization error is similar to that of the k-anchor invertible Laplacian transform, but it is smooth where x is smooth. We will provide a provable bound on the norm of the quantization error that this technique introduces in a companion paper.

To place k anchor points we use a greedy algorithm. We begin by placing one random anchor point and generating a 1-anchor rectangular Laplacian, denoted by $\tilde{L}_1$. We proceed iteratively: given $\tilde{L}_j$ we place the $j+1$ anchor at the vertex with the largest visual error, to yield $\tilde{L}_{j+1}$. We can either place enough anchors to satisfy some prescribed visual error, or place k anchors. This can be accelerated by adding some anchors in random locations. However, since typically only a small portion of the vertices are used as anchors (about $0.1\% - 0.7\%$), we found the greedy algorithm to be effective.

Figures 1 and 7 (see the color section) visualize 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.

4. 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:

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

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.

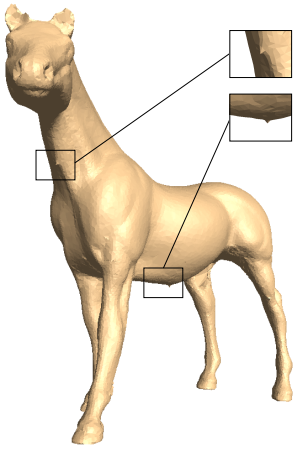

Figure 5: *Reconstruction of the mesh from quantized δ-coordinates using k-anchor invertible Laplacian is not smooth at the anchor vertices.*

Karni and Gotsman [15] 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 a local measure of smoothness at v_i:

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

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 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 8 (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 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 8.

This agrees with the claim that our perception is more sensitive to lighting than geometry. When the tessellation is finer, the same displacements have more effect on the normals and so the distortion of the lighting is stronger.

Taubin [21] 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 6.

5. Implementation and results

5.1. 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 [3, 22]. 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 $n+1, \ldots, n+k$ 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.

5.2. Rate-distortion

To evaluate the performance of our scheme we generated a number of rate-distortion curves comparing our technique to the parallelogram scheme [22], 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 6) 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 6 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 4, the S_q measure is more visually important. This is further supported by Figure 8 (see the color section) in a metric-independent way. The figure shows 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.

5.3. Solving least-squares problems

Decoding in our method requires solving a linear least-squares problem. There are efficient algorithms that can solve such problems for sparse systems [4]. We have experimented with two direct solvers, one based on the $\tilde{L} = QR$ decomposition, and one based on triangular factorization of the coefficient matrix $\tilde{L}^T\tilde{L}$ of the so-called normal equations. The second solver is faster than the QR procedure, but produces less accurate solutions. However, the accuracy of the solutions depends on the condition number of $\tilde{L}$ (ratio of extreme singular values), and in our case $\tilde{L}$ is well-conditioned, thanks to the anchors, and yields sufficiently accurate solutions in terms of visual quality.

These methods are fast enough for decoding moderately large models. For example, computing the triangular factorization of the normal equations for the horse model (19,851 vertices) took 0.98 seconds on a 2 GHz P4 computer with RDRAM, and solving for a single mesh function took 0.04 seconds once the factorization was computed (see Table 1).

In all direct methods, the factorization is computed once and is used to solve for multiple mesh functions. Most of the time is spent on 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).

Model	#vertices	Factorization (sec.)	Solving (sec.)
Eight	2,718	0.127	0.006
Horse	19,851	0.980	0.040
Fandisk	20,111	1.595	0.056
Venus	50,002	4.803	0.151
Max Planck	100,086	10.790	0.318

Table 1: *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).*

5.4. 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 [6]. 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 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 2, 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 [12]). 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.

6. Conclusions

This paper addressed the issue of reducing the visual effects of aggressive quantization of 3D meshes. We argue

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

Table 2: *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.*

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.

To improve the decoding times even further, we plan to compare the performance of iterative and direct solvers for our application. Iterative least-squares solvers do not factor the coefficient matrix, but iteratively improve an approximate solution. 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 case, $\tilde{L}$ is always well conditioned, so we can expect reasonably rapid convergence. Furthermore, the decoder knows the values of the mesh functions at the anchor vertices. By interpolating these values, the decoder can quickly produce a good initial approximation. We expect that for large models, iterative methods would outperform direct solvers.

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 [18]. We believe that the ability to manipulate 3D models in the frequency domain 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.

References

1. Alexa, M. 2001. Local control for mesh morphing. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI-01)*, IEEE Computer Society, 209–215.

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

3. Alliez, P., and Desbrun, M. 2001. Valence-driven connectivity encoding for 3D meshes. *Computer Graphics Forum 20*, 3, 480–489.

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

5. Boman, E. G., and Hendrickson, B. 2001. Support theory for preconditioning. Tech. Rep. SAND-01-1794J, Sandia National Labs, Mar.

6. 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.

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

8. Cignoni, P., Rocchini, C., and Scopigno, R. 1998. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2, 167–174.

9. Deering, M. F. 1995. Geometry compression. In *Proceedings of ACM SIGGRAPH 95*, 13–20.

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

11. Guattery, S., and Miller, G. L. 2000. Graph embeddings and laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications 21*.

12. Gumhold, S., and Amjoun, R. 2003. Higher order prediction for geometry compression. *International Conference On Shape Modelling And Applications*. Accepted for publication.

13. Gumhold, S., and Straßer, W. 1998. Real time compression of triangle mesh connectivity. In *Proceedings of ACM SIGGRAPH 98*, 133–140.

14. 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.

15. Karni, Z., and Gotsman, C. 2000. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000*, 279–286.

16. Khodakovsky, A., Schröder, P., and Sweldens, W. 2000. Progressive geometry compression. In *Proceedings of ACM SIGGRAPH 2000*, 271–278.

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

18. Ohbuchi, R., Mukaiyama, A., and Takahashi, S. 2002. A frequency-domain approach to watermarking 3D shapes. *Computer Graphics Forum 21*, 3, 373–382.

19. Rossignac, J. 1999. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1, 47–61.

20. Taubin, G., and Rossignac, J. 1998. Geometric compression through topological surgery. *ACM Transactions on Graphics 17*, 2, 84–115.

21. Taubin, G. 1995. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH 95*, 351–358.

22. Touma, C., and Gotsman, C. 1998. Triangle mesh compression. In *Graphics Interface '98*, 26–34.

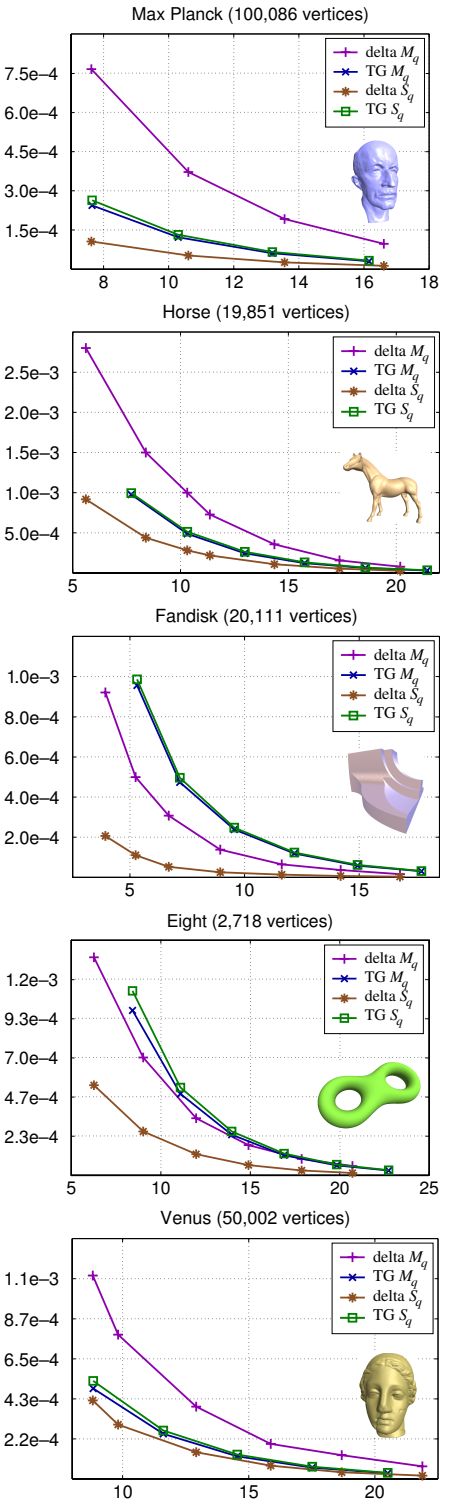

Figure 6: *Rate distortion curves for five known models. The graphs show the M_q and S_q measures as functions of the entropy, for* δ*-coordinates and the TG method.*

Figure 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.*

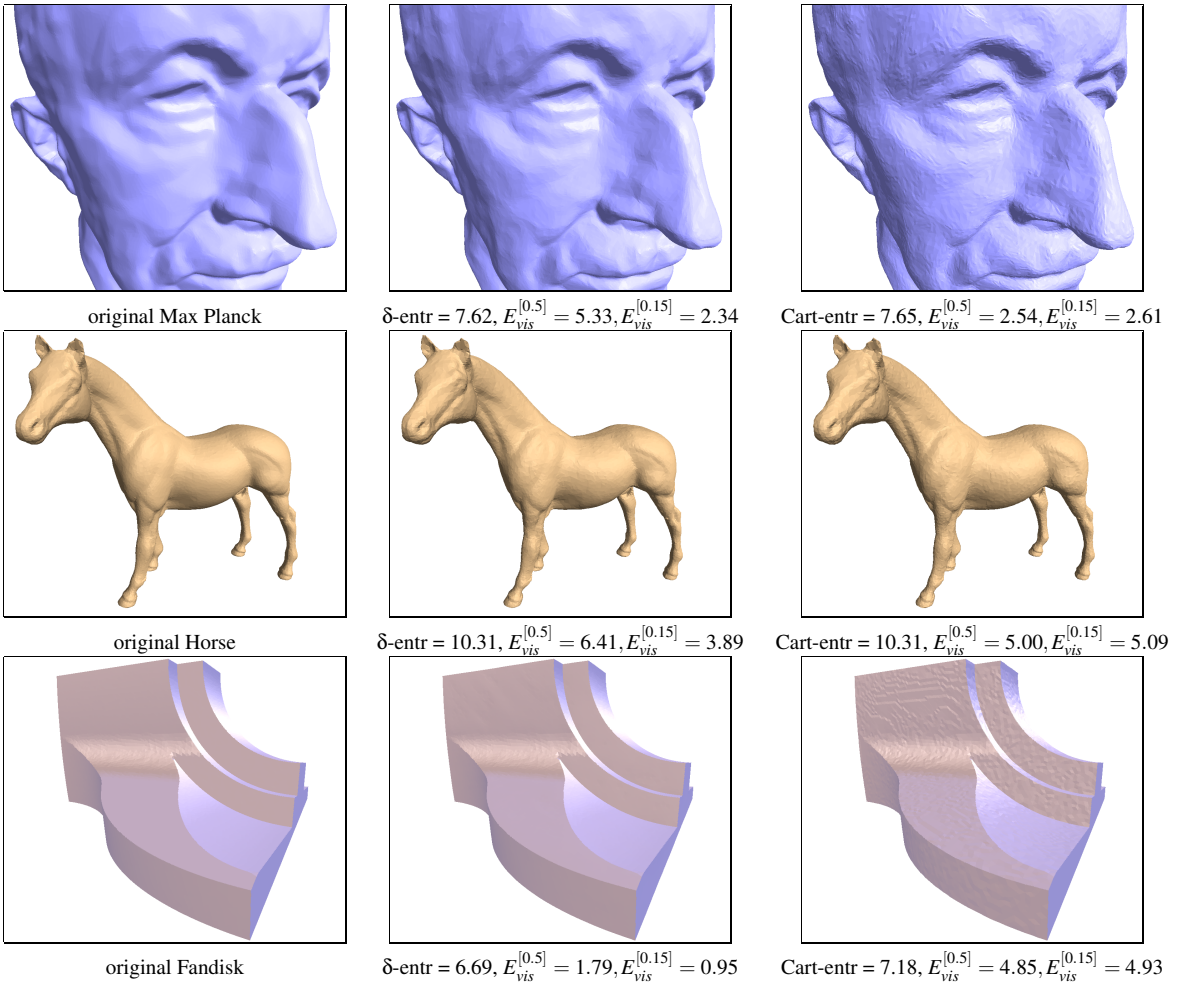

Figure 8: *Comparison of visual quality of the meshes using δ-coordinates quantization vs. standard Cartesian coordinates quantization. The original meshes are shown in the left column, quantized δ-coordinates in the middle column, and quantized Cartesian coordinates in the right column. The entropy of the δ-coordinates is slightly lower than that of parallelogram-prediction displacements, while visually, the surfaces look closer to the original. Using E_{vis} with $\alpha = 0.5$ (denoted by $E_{vis}^{[0.5]}$), 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 $\alpha = 0.15$, does E_{vis} agree with our perception. The error values are given in units of 10^{-4}.*