Spectral Compression of Mesh Geometry

Zachi Karni, Craig Gotsman

SIGGRAPH 2000

Introduction

- Thus far, topology coding drove geometry coding.
- Geometric data contains far more information (15 vs. 3 bits/vertex).
- Quantization methods are not suitable for lossy compression.
  - non-graceful degradation.

Intuition

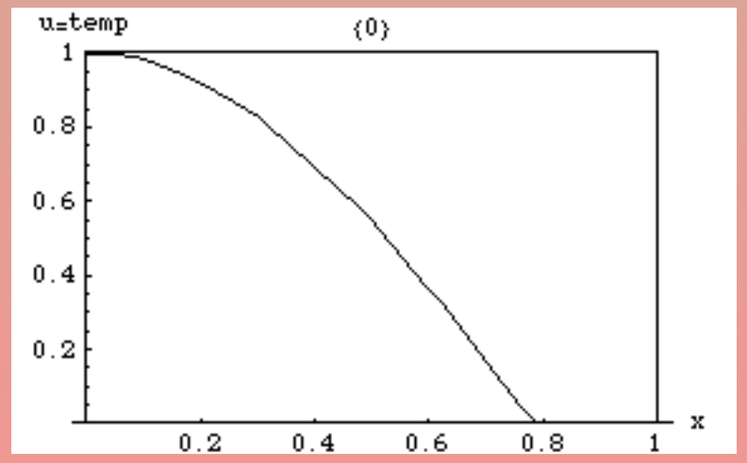
Laplace Operator

- The Laplace operator is a second order differential operator.

$$\Delta f = f_{xx} + f_{yy} + f_{zz}$$

- One-dimensional heat equation:

$$u_t = k\Delta u = ku_{xx}$$



Discrete Laplace Operator

- Defined so that the Laplace operator has meaning on a graph or discrete grid (e.g. 3D mesh).

$$\Delta x_i = \sum_{j \in i^*} w_{ij} (x_j - x_i) ,$$

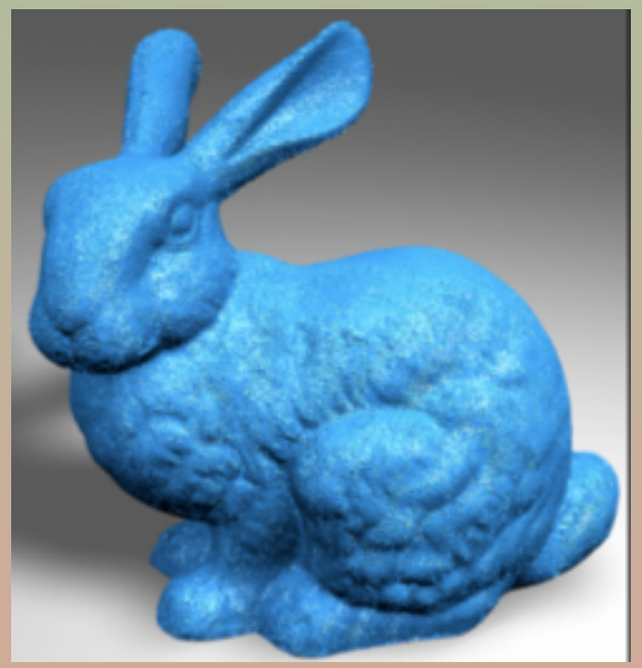
$$\sum_{j \in i^*} w_{ij} = 1 .$$

- Much discussion over "correct" weights
  - most commonly: $w_{ij} = \frac{1}{|i^*|}$

Laplacian Smoothing

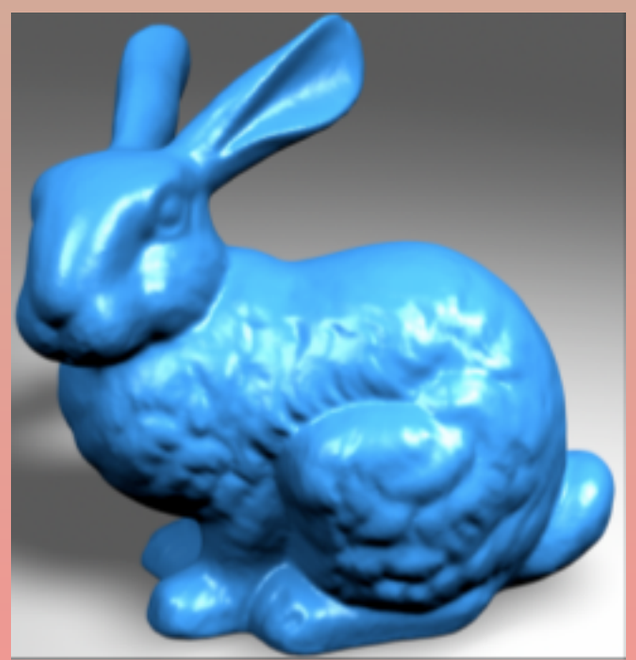
```
Laplacian(G,W,x)
    new Δx = 0;
    for(e = (i, j) ∈ E)
        Δx_i = x_i + w_ij(x_i − x_j);
    end;
return Δx;
```

Figure 1: *Algorithm to evaluate the Laplacian operator. $G = (V,E)$ directed graph, W matrix of weights defined on the edges of G, x input signal on G, Δx output signal.*



```
LaplacianSmoothing(G,W,N,λ,x)
    new Δx
    for(i = 0; i < N; i = i + 1)
        Δx = Laplacian(G,W,x);
        x = x + λΔx;
    end;
return;
```

Figure 2: *The Laplacian Smoothing Algorithm. G graph, W matrix of weights defined on the edges of G, N number of iterations, λ scaling factor, x signal on G to be smoothd.*



Laplacian Matrix

- We can express the discrete Laplacian operator in matrix-vector notation.

$$\Delta x_i = \sum_{j \in i^*} w_{ij} \left(x_j - x_i \right) ,$$

$$\Delta x = -Lx, L = I - W$$

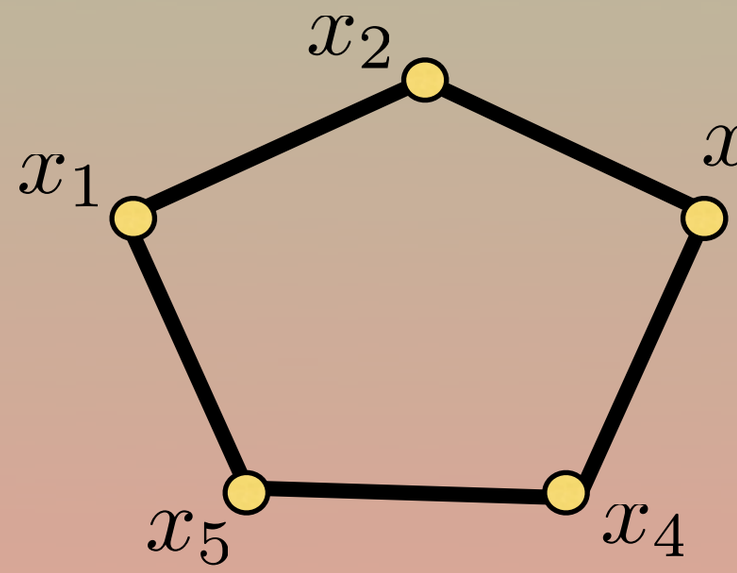
- Laplacian matrix in this paper:

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

Spectral Motivation

- Regular polygon of n vertices.

$n = 5$:



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_5 \end{bmatrix}$$

- Discrete Laplacian:

$$\Delta x_i = \frac{1}{2}(x_{i-1} - x_i) + \frac{1}{2}(x_{i+1} - x_i) \ ,$$

Spectral Motivation

- Discrete Laplacian written in matrix form:

$$\Delta x = -K\,x\ ,$$

$$K = \frac{1}{2}\begin{pmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{pmatrix} .$$

Spectral Motivation

- n real eigenvalues of K in increasing order:

$$k_j = 1 - \cos(2\pi \lfloor j/2 \rfloor / n) \ ,$$

- The n real eigenvectors of K are of the following form:

$$(u_j)_h = \begin{cases} \sqrt{1/n} & \text{if } j = 1 \\ \sqrt{2/n}\, \sin(2\pi h \lfloor j/2 \rfloor / n) & \text{if } j \text{ is even} \\ \sqrt{2/n}\, \cos(2\pi h \lfloor j/2 \rfloor / n) & \text{if } j \text{ is odd} \ . \end{cases}$$

Encoding

- Partition the mesh into submeshes.
- Compute the topological Laplacian matrix for each little submesh.
- Represent each submesh as a linear combination of orthogonal basis functions derived from the eigenvectors of the Laplacian.

Decoding

- Topology encoded/decoded by your method of choice.
- Geometric data sent as coefficient vectors.
- Mesh partitioned and eigenvectors computed based on topology, which are then used to decode the geometry.

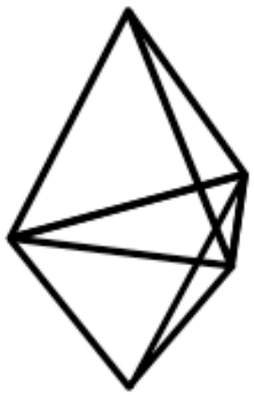
Mesh Signal Processing

- Laplacian matrix:

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

- Eigenvectors form an orthogonal basis of R^n.
- Associated eigenvalues are the squared frequencies.

Mesh Signal Processing



(a)

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

(b)

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

(c)

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

(d)

$$[0 \ 1 \ 1.25 \ 1.25 \ 1.51]$$

(e)

Mesh Laplacian Eigen Matrix Eigenvalues

Geometry Vectors

- We are going to view the geometry as three n-dimensional column vectors (x, y, z), where n is the number of vertices.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

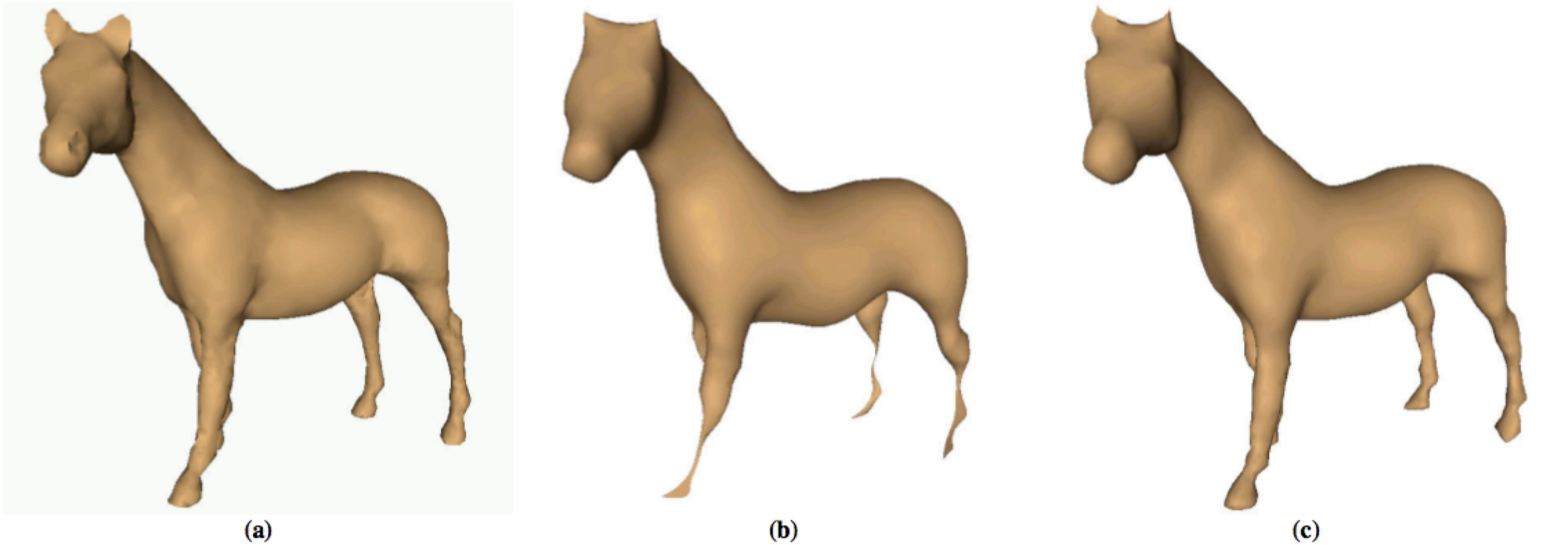
Mesh Signal Processing

- Since $e^1, \ldots, e^n$ form a basis of n-dimensional space, every n-dimensional vector can be written as a linear combination:

$$x = \sum_{j=1}^{n} \hat{x}_j e^j = E\hat{x},$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, E = \begin{bmatrix} | & | & & | \\ e^1 & e^2 & \ldots & e^n \\ | & | & & | \end{bmatrix}, \hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_n \end{bmatrix}$$

Mesh Signal Processing



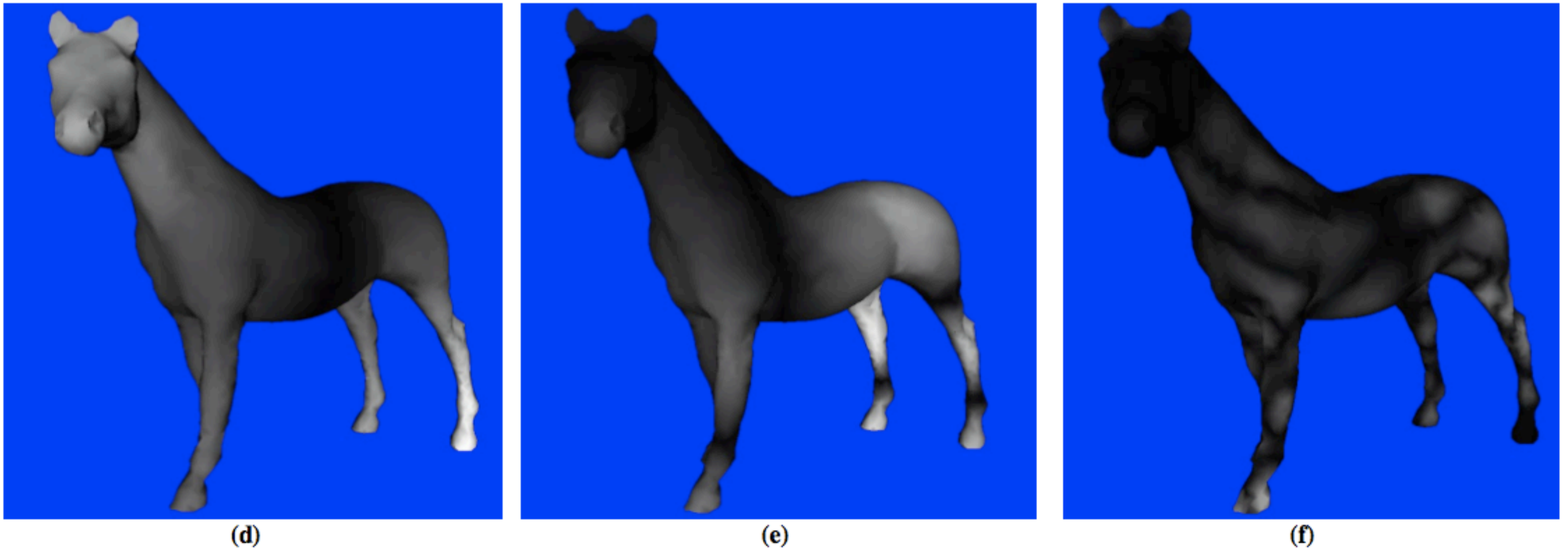
Original model containing 2,978 vertices.

Reconstruction using 100 of the 2,978 basis functions.

Reconstruction using 200 basis functions.

Mesh Signal Processing

The grayscale intensity of a vertex is proportional to the scalar value of the basis function at that coordinate.



(d) (e) (f)

Second basis function.
Eigenvalue = 4.9x10^-4

Tenth basis function.
Eigenvalue = 6.5x10^-2

Hundredth basis function.
Eigenvalue = 1.2x10^-1

Spectral Coefficient Coding

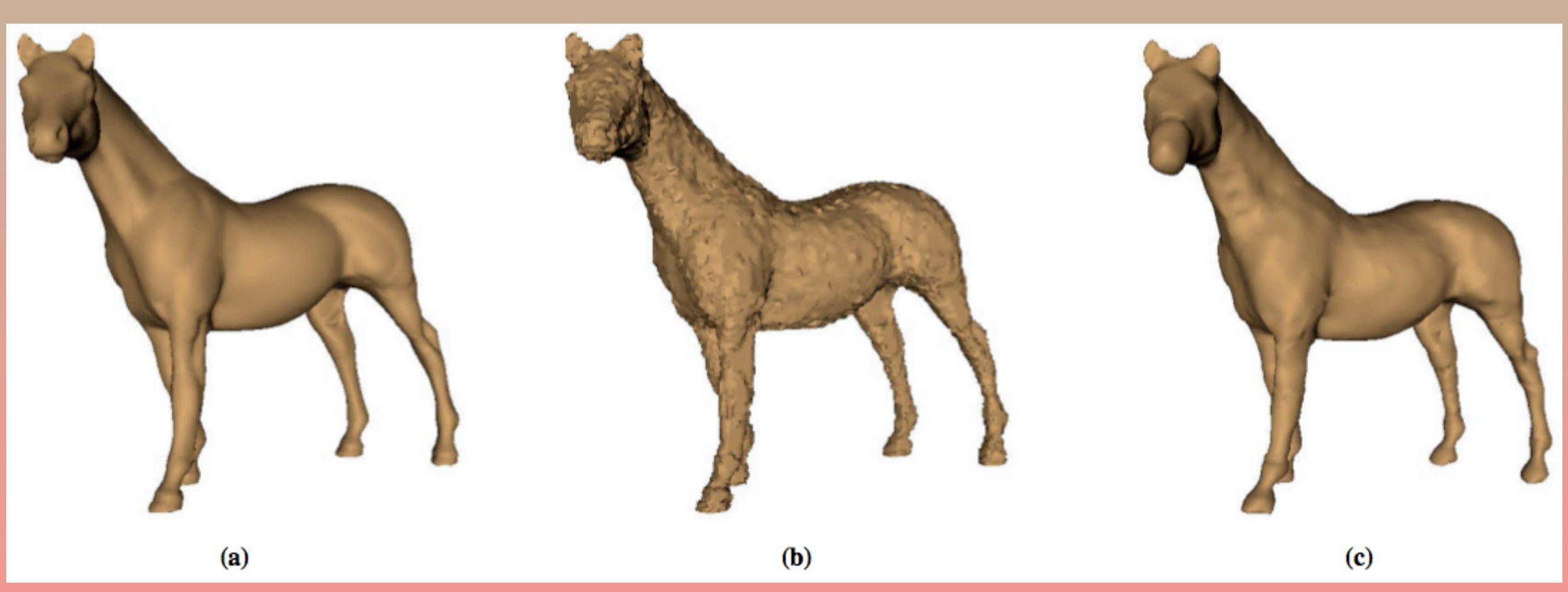
- Uniformly quantize $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$ coefficient vectors to finite precision (10-16 bits).
- Truncate the coefficient vectors.
- Encode using Huffman or arithmetic coder.

Spectral Coefficient Coding

- Trade-off:
  - Small number of high-precision coefficients
  - Large number of low-precision coefficients
- Optimize based on visual metrics.
  - Number of retained coefficients per coordinate per submesh chosen such that a visual quality is met.

A Visual Metric

- RMS geometric distance between corresponding vertices in both models does not capture properties like smoothness.



A Visual Metric

- Geometric Laplacian:

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

- Average of the norm of the geometric distance and the norm of the Laplacian distance.

$$\| M_1 - M_2 \| = \frac{1}{2n} \left(\| v^1 - v^2 \| + \| GL(v^1) - GL(v^2) \| \right)$$

Mesh Partitioning

- Eigenvectors can be calculated in O(n) time since Laplacian is sparse. (n is the number of vertices.)
- When n is large, eigenvalues become too close, leading to numerical instability.
- Necessary to partition mesh.

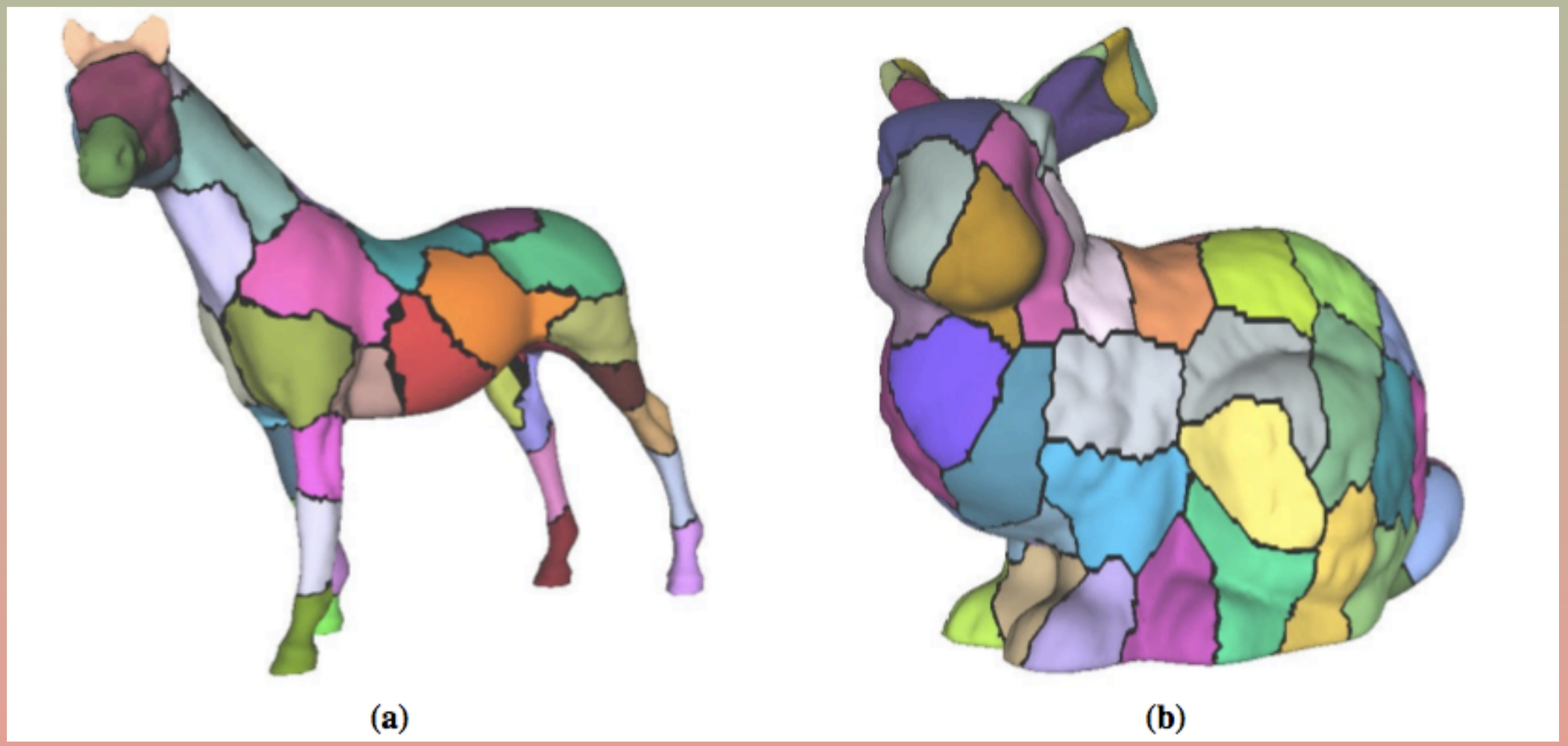
Mesh Partitioning

- Capture local properties better.
- Minimize Damage:
  - roughly same number of vertices in each submesh.
  - minimize number of edges straddling different submeshes (*edge-cut*).

Mesh Partitioning

- Optimal solution is NP-Complete.
- MeTIS
  - Optimized linear-time implementation.
  - Meshes of up to 100,000 vertices.
  - preference to minimizing the *edge-cut* over balancing partition.

METIS

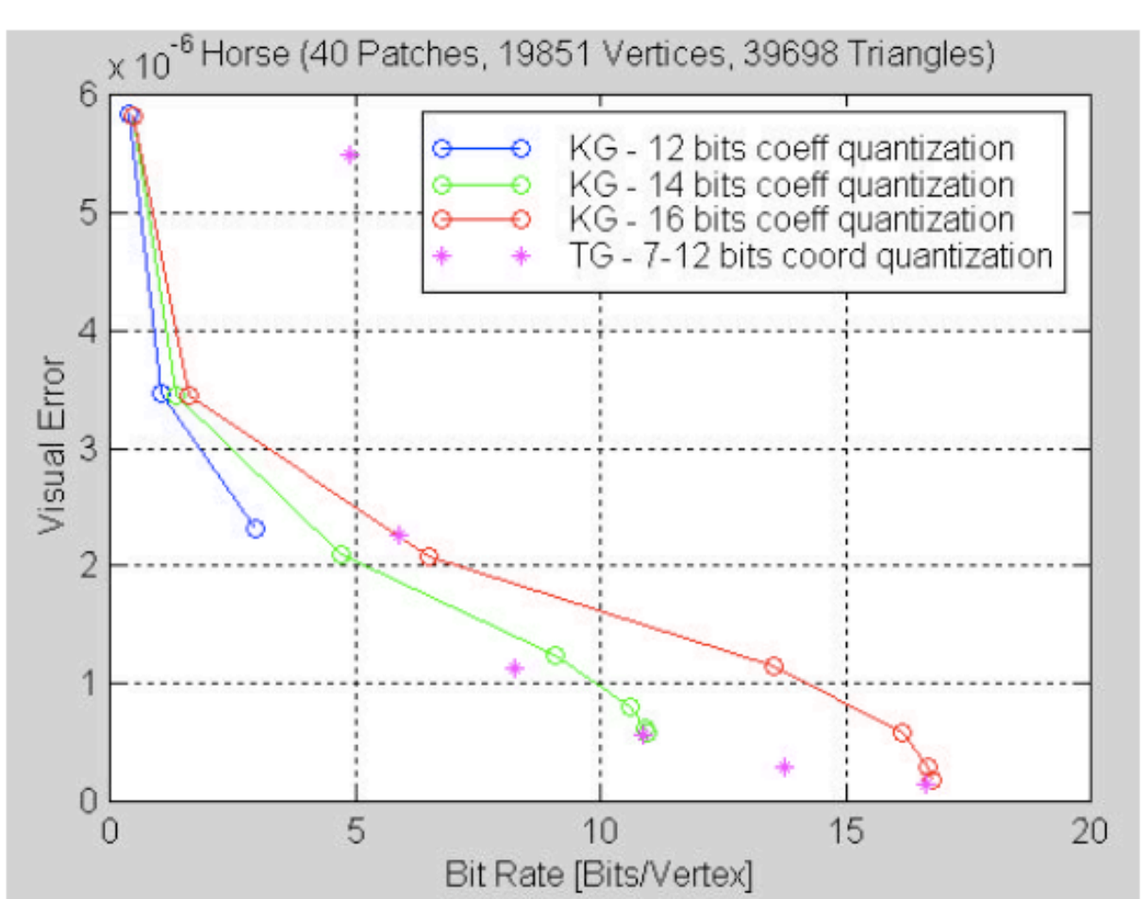


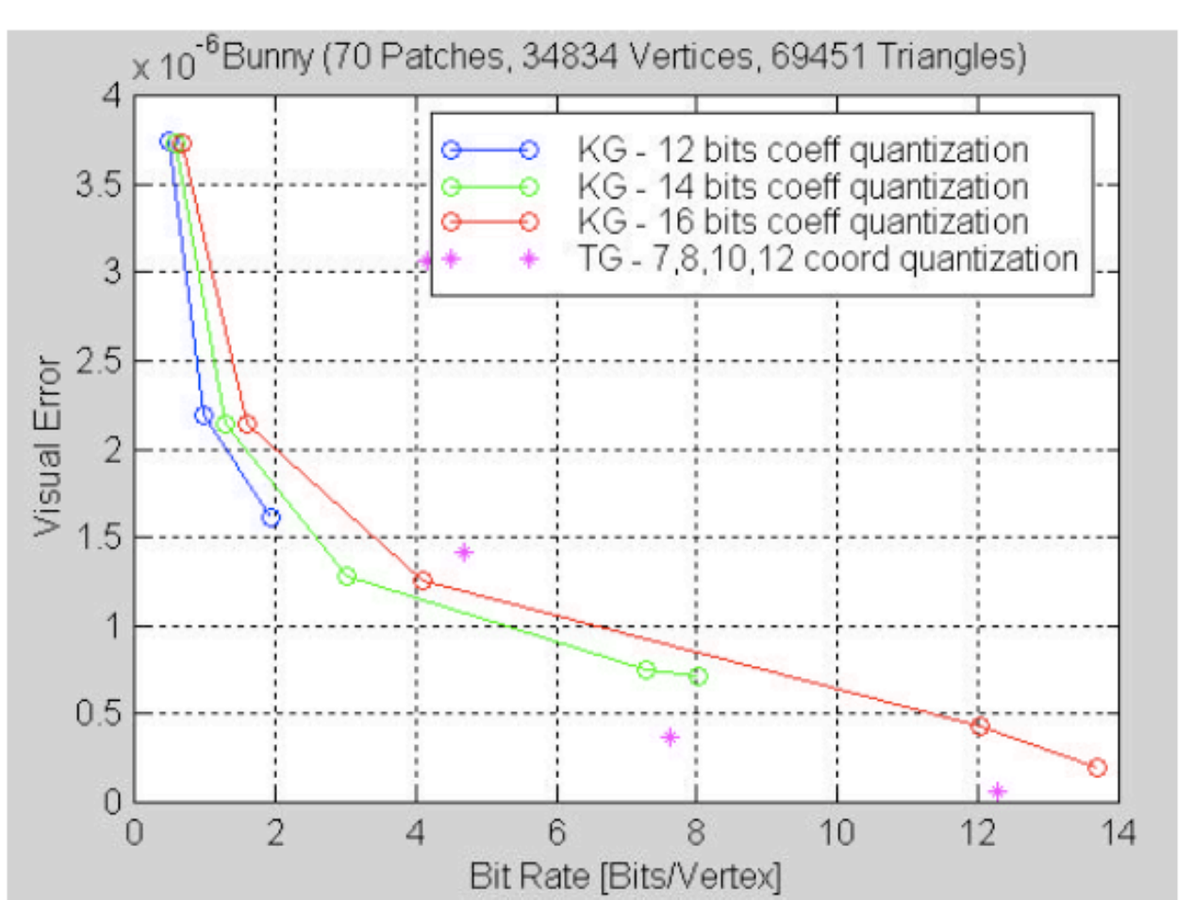
40 submeshes

70 submeshes

Results

- Comparison with Touma-Gotsman (TG) compression.

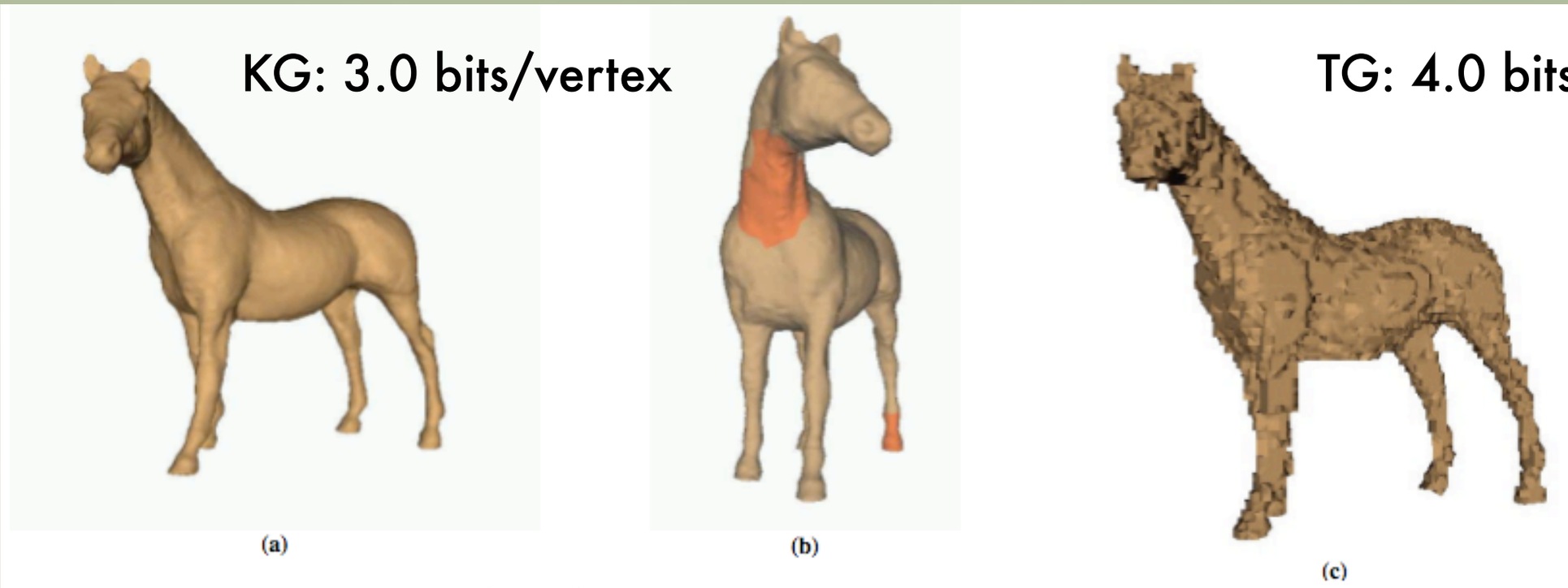




Results

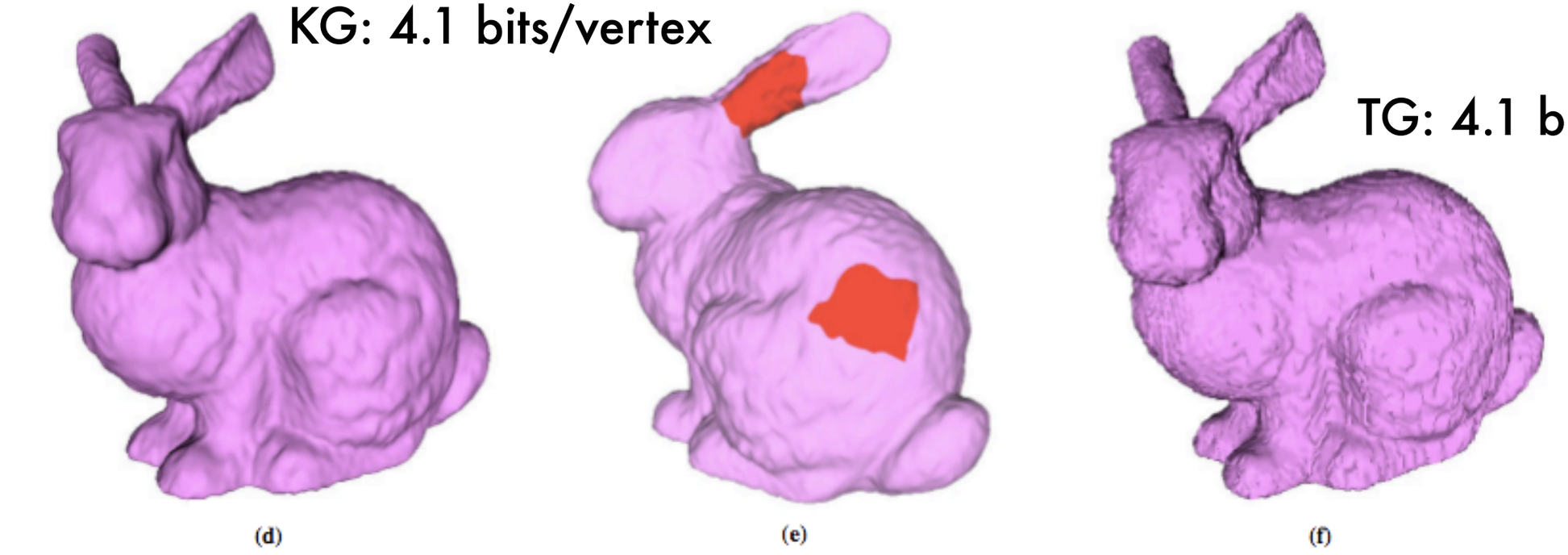
KG: 3.0 bits/vertex

TG: 4.0 bits/vertex



KG: 4.1 bits/vertex

TG: 4.1 bits/vertex



Discussion

- Can’t just throw out high frequency detail; it is not noise!
- Can’t use the algorithm on very large meshes. MeTIS only accommodates meshes of up to 100,000 vertices.