Spectral Meshes

COS 526, Fall 2016

Slides from Olga Sorkine, Bruno Levy, Hao (Richard) Zhang

Motivation

Want frequency domain representation for 3D meshes

- Smoothing
- Compression
- Progressive transmission
- Watermarking
- etc.

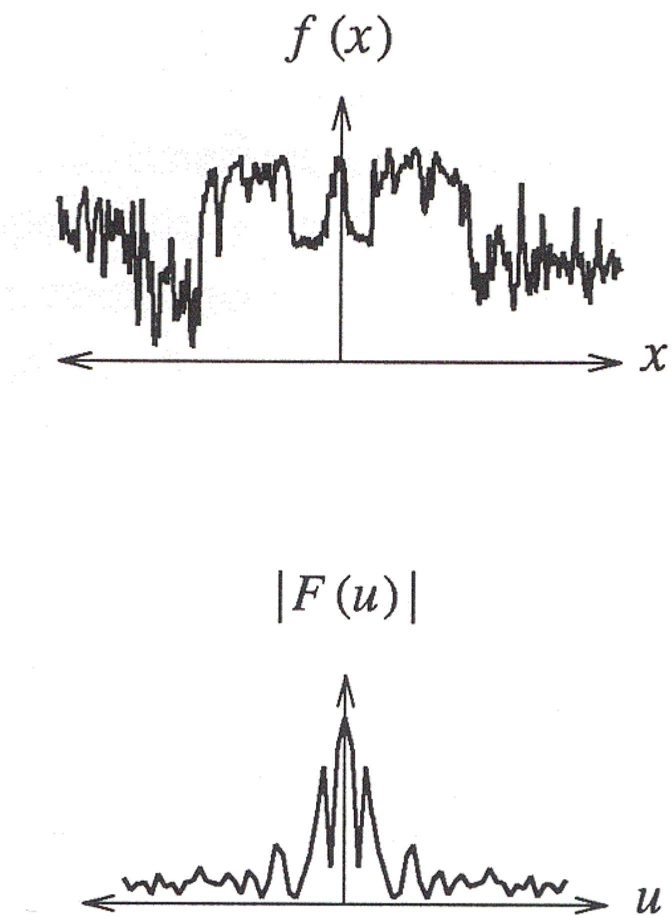
Frequencies in a mesh

This lecture = spectral meshes

- Like Fourier

[Hoppe]

Fourier Transform



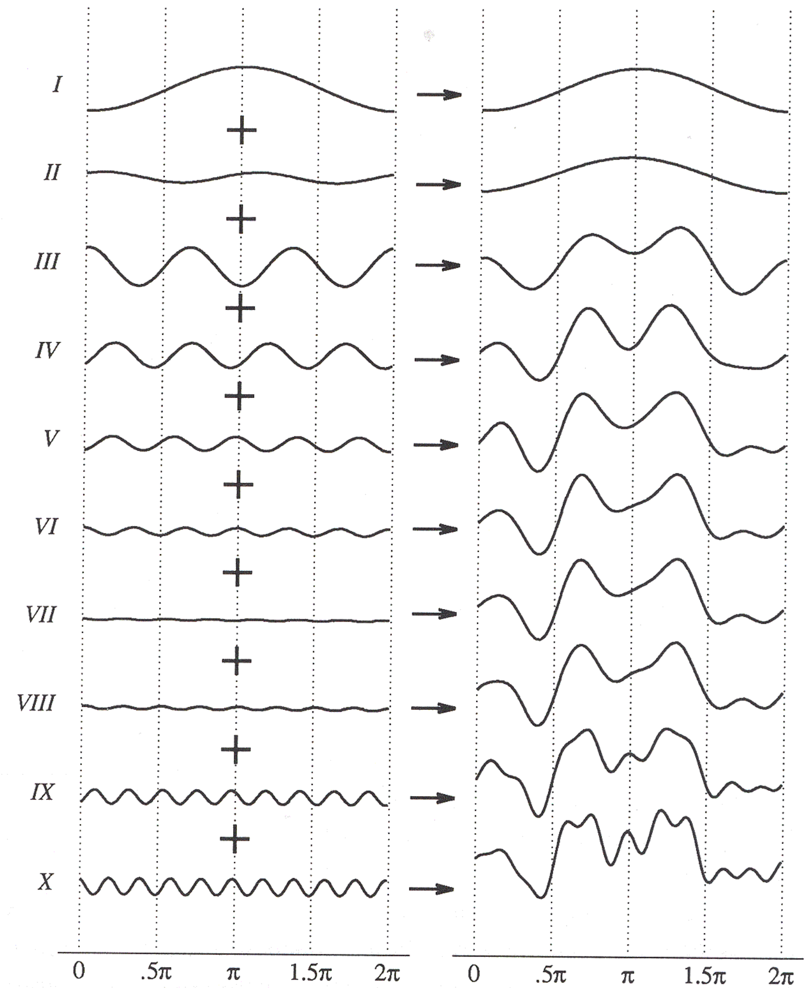
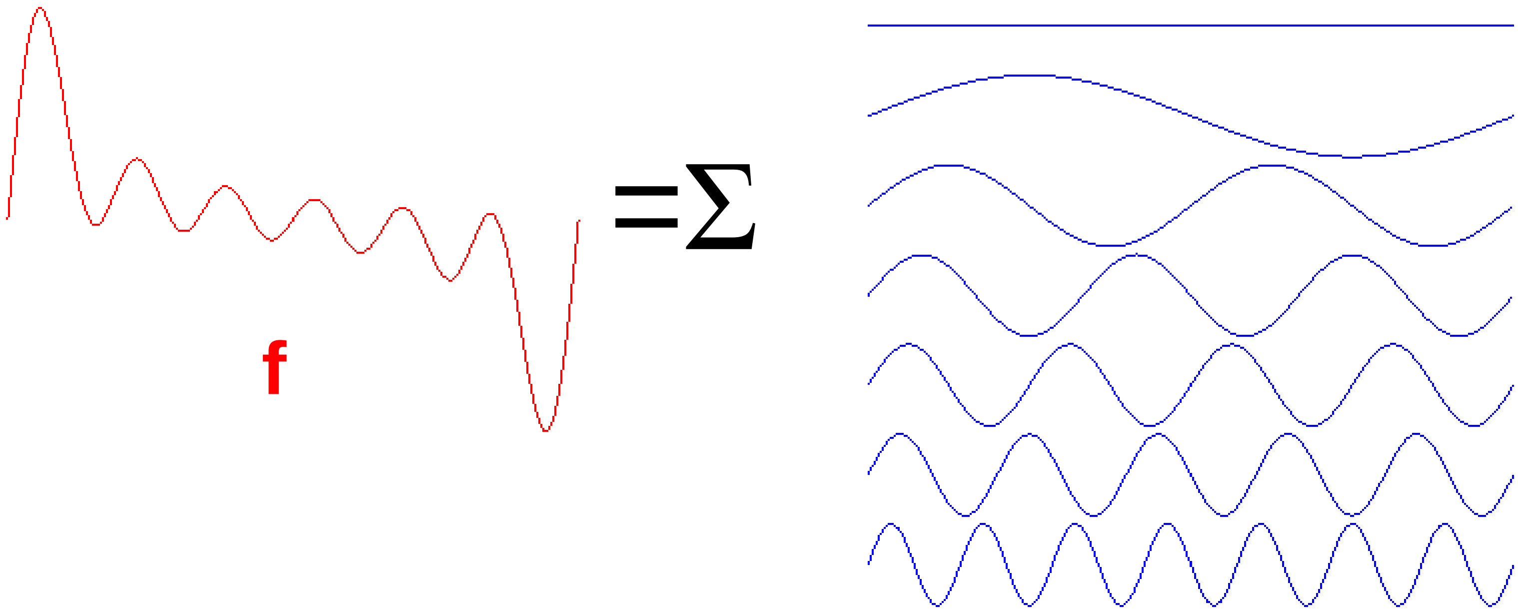


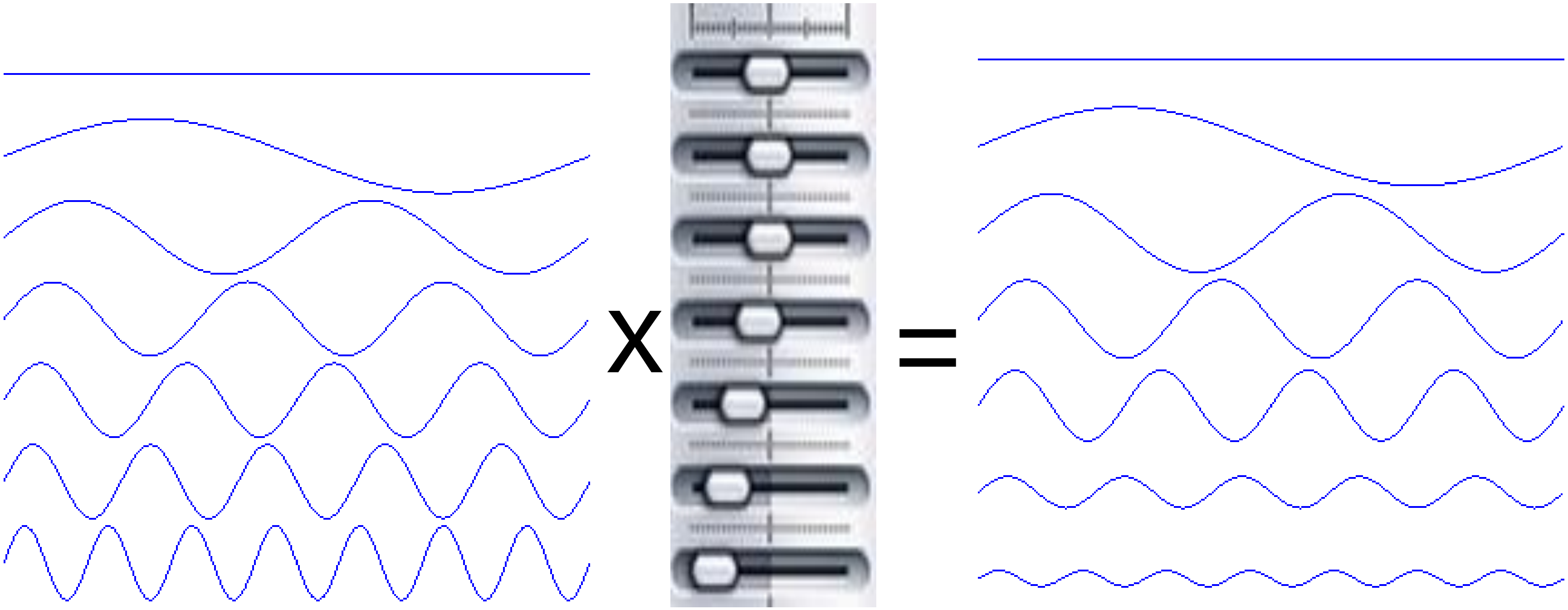
Figure 2.6 Wolberg

Frequency domain

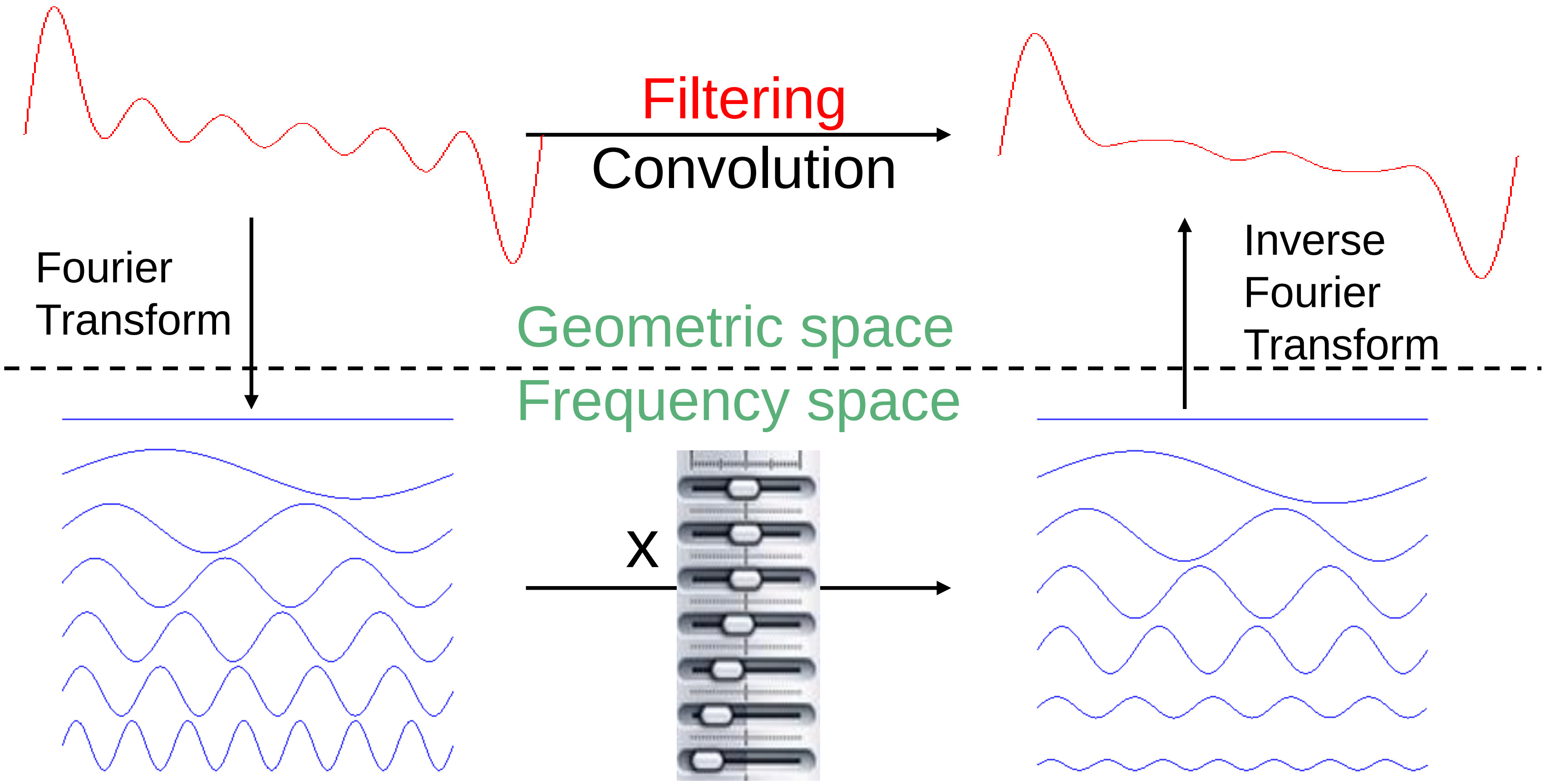


sin(kx)

Filtering

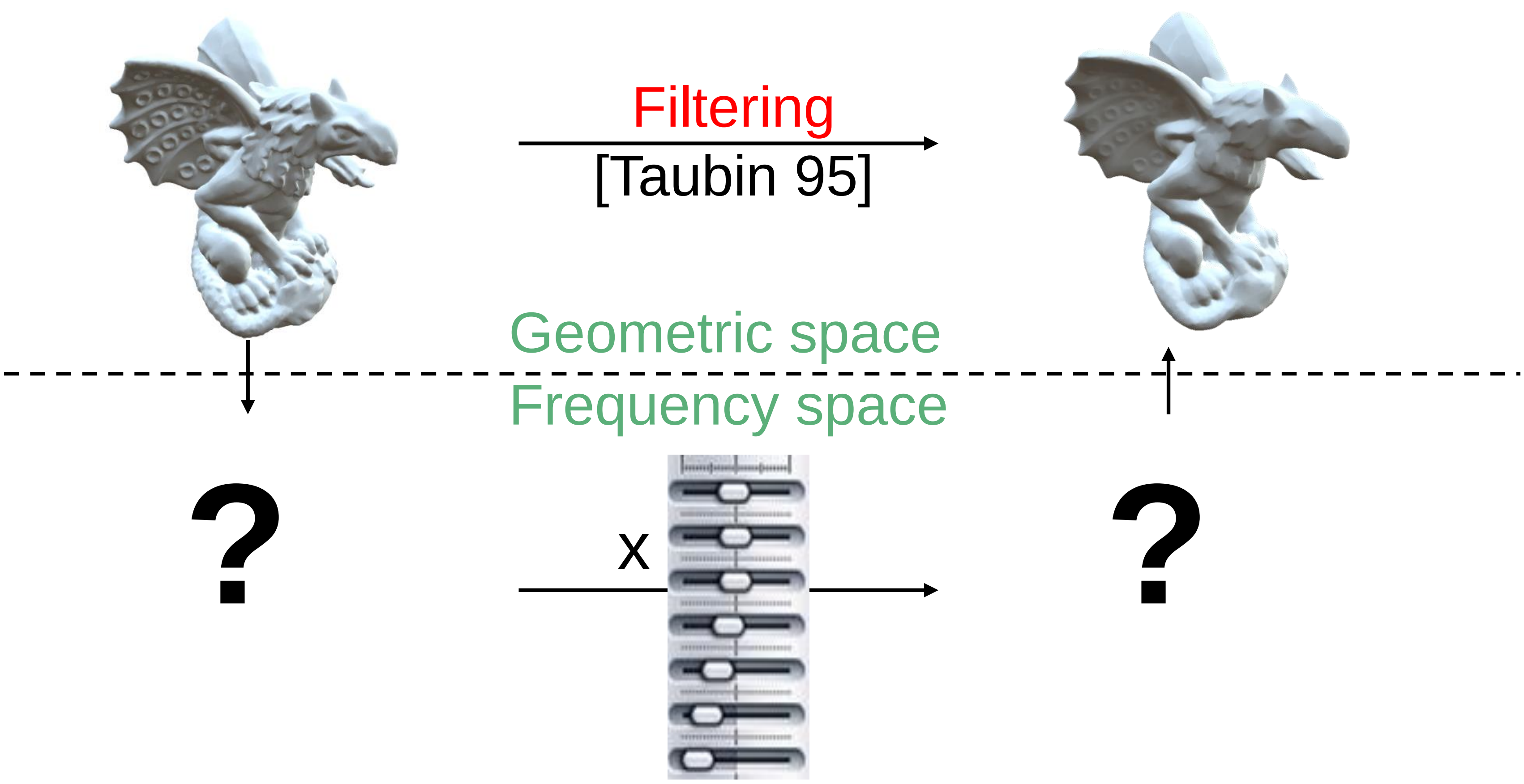


Filtering



Filtering
Convolution
Fourier
Transform
Inverse
Fourier
Transform
Geometric space
Frequency space
X

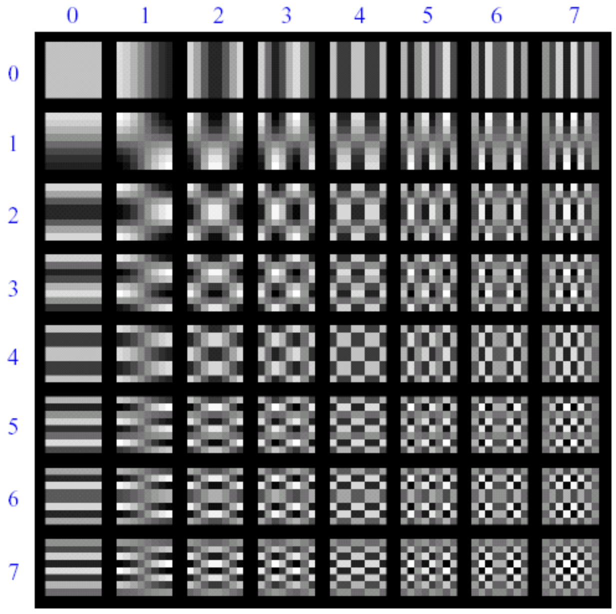
Filtering on a mesh



Frequencies in a function

Fourier analysis

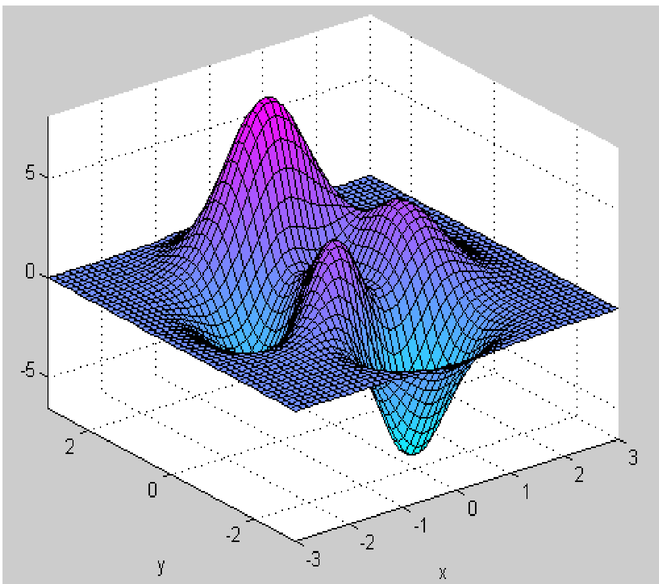
- 2D bases for 2D signals (images)



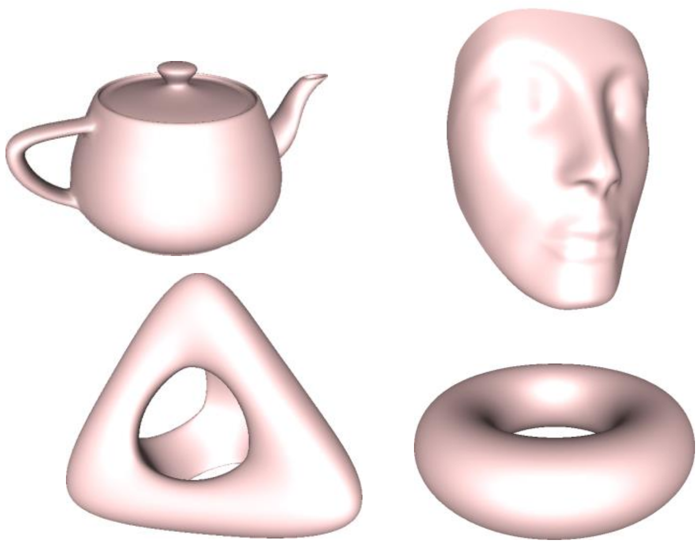
$$\cos\left(\frac{\pi u}{16}(2x+1)\right)\cos\left(\frac{\pi v}{16}(2y+1)\right)$$

How about 3D shapes?

Problem: 2D surfaces embedded in 3D are not (height) functions



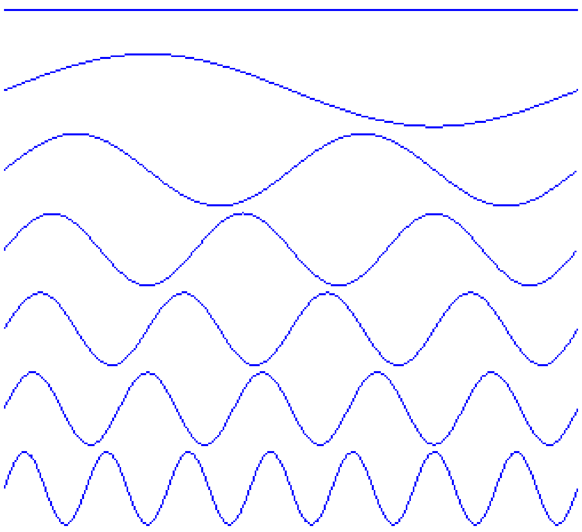
Height function, regularly sampled above a 2D domain



General 3D shapes

Basis functions for 3D meshes

Need extension of the Fourier basis to a general (irregular) mesh



sin(kx)

on

?

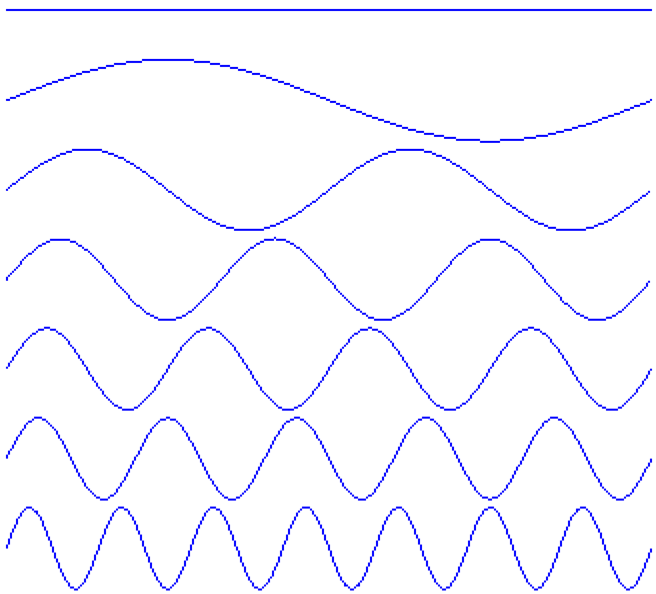
Basis functions for 3D meshes

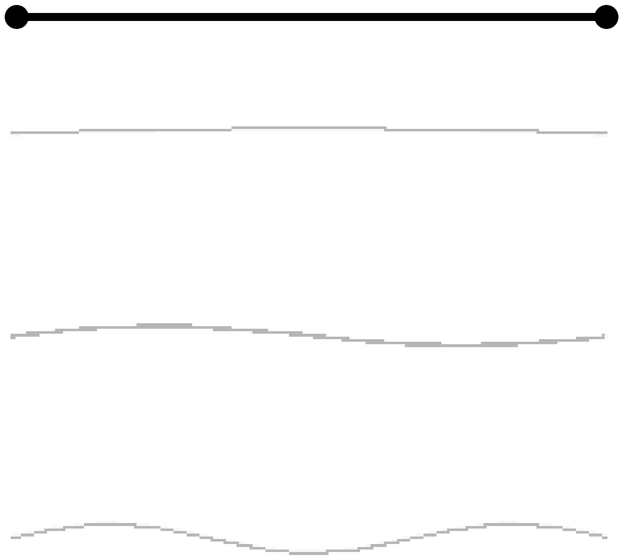
We need a collection of **basis functions**

- First basis functions will be very smooth, slowly-varying
- Last basis functions will be high-frequency, oscillating

We will represent our shape (mesh geometry) as a **linear combination** of the basis functions

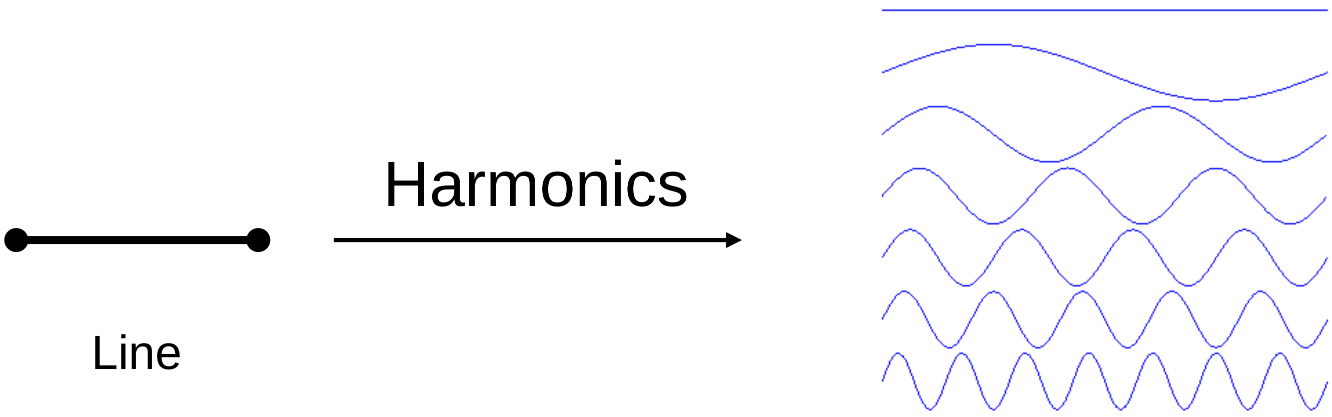
Harmonics





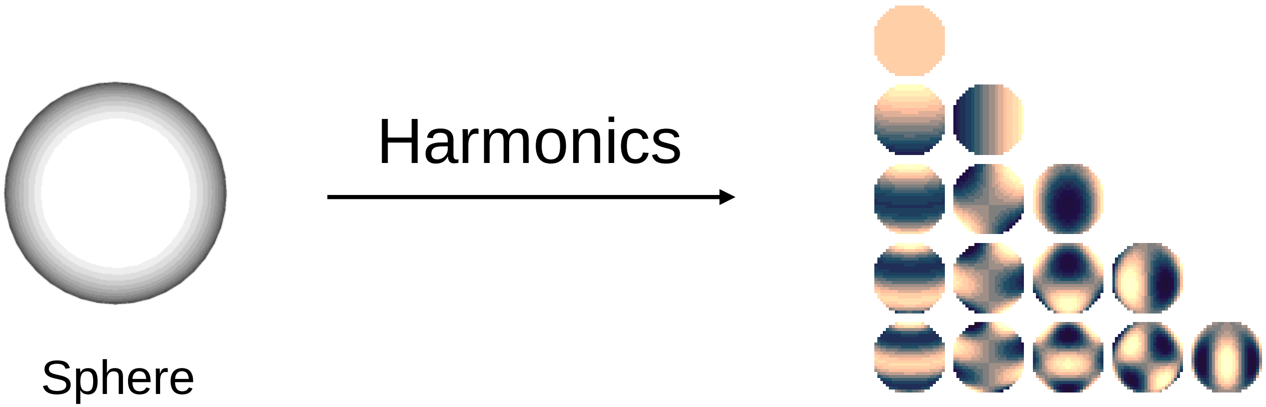
sin(kx) are the stationary vibrating modes = harmonics of a string

Harmonics



Stationary vibrating modes

Spherical Harmonics



Stationary vibrating modes

Manifold Harmonics

$\xrightarrow{\text{Harmonics}}$ **?**

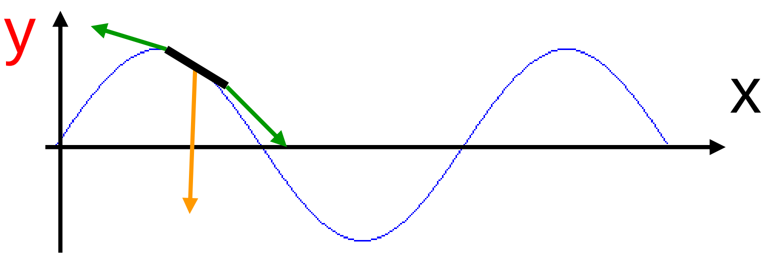
Stationary vibrating modes

Harmonics

Wave equation:

$T\,\partial^2 y/\partial x^2 = \mu\,\partial^2 y/\partial t^2$

T: stiffness μ: mass

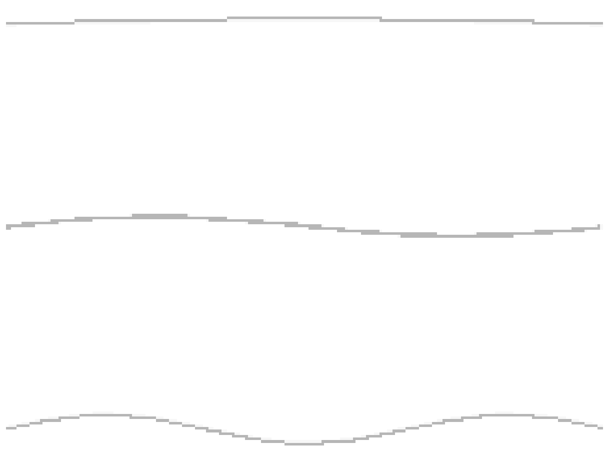


Stationary modes:

$y(x,t) = y(x)\sin(\omega t)$

$\partial^2 y/\partial x^2 = -\mu\omega^2/T\ y$

eigenfunctions of $\partial^2/\partial x^2$



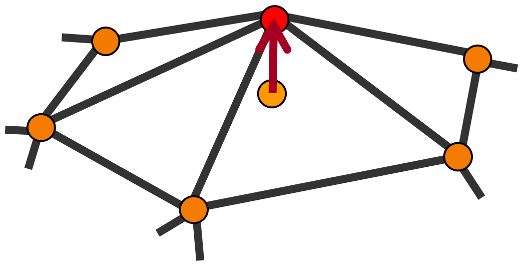
Harmonics

Harmonics are **eigenfunctions** of $\partial^2/\partial x^2$

On a mesh, $\partial^2/\partial x^2$ is the Laplacian Δ

Frequency domain basis functions for 3D meshes are **eigenfunctions** of the Laplacian

The Mesh Laplacian operator



$$L(\mathbf{v}_i) = d_i \mathbf{v}_i - \sum_{j \in N(i)} \mathbf{v}_j = d_i \left(\mathbf{v}_i - \frac{1}{d_i} \sum_{j \in N(i)} \mathbf{v}_j \right)$$

Measures the local smoothness at each mesh vertex

Laplacian operator in matrix form

$$
\begin{pmatrix}
d_1 & -1 & 0 & \cdots & -1 & \cdots & \cdots & 0 \\
0 & d_2 & & -1 & & & -1 & \\
\vdots & & d_3 & & & & & \\
\vdots & & & \ddots & & & & \\
\vdots & & & & \ddots & & & \\
\vdots & & & & & \ddots & & \\
0 & -1 & & -1 & & -1 & d_{n-1} & \\
-1 & & -1 & & -1 & & & d_n
\end{pmatrix}
\begin{pmatrix}
\mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{v}_{n-1} \\ \mathbf{v}_n
\end{pmatrix}
=
\begin{pmatrix}
\delta_1 \\ \delta_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \delta_{n-1} \\ \delta_n
\end{pmatrix}
$$

L matrix

Spectral bases

L is a symmetric $n \times n$ matrix

Eigenfunctions of L computed with spectral analysis

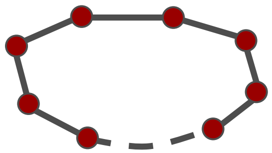
$$L = \begin{bmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \begin{bmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ | & | & & | \end{bmatrix}^T$$

Basis vectors

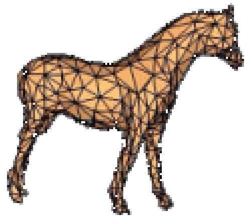
Frequencies, sorted in ascending order

The spectral basis

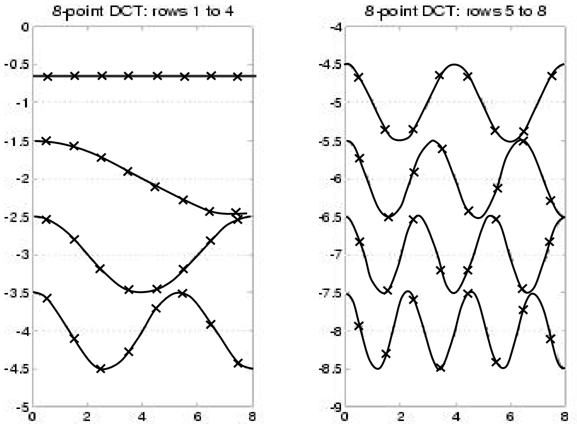
First functions are smooth and slow, last oscillate a lot



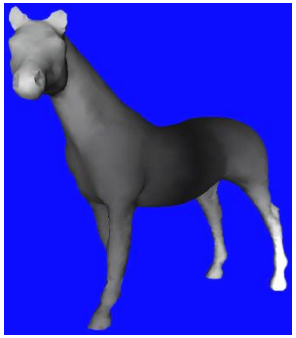
chain connectivity



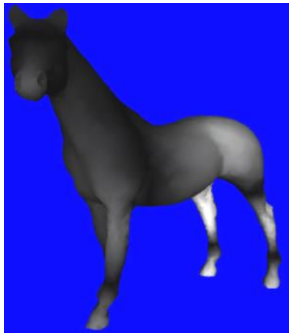
horse connectivity



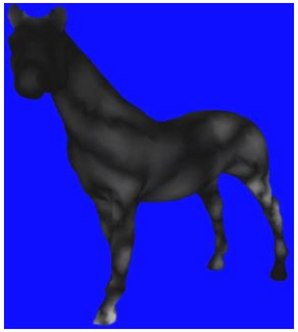
spectral basis of L = the DCT basis



2nd basis function



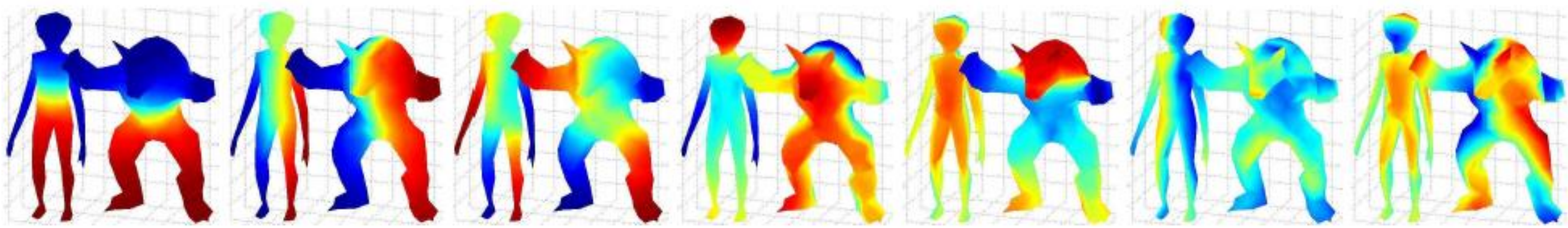
10th basis function



100th basis function

The spectral basis

First functions are smooth and slow, last oscillate a lot



Spectral mesh representation

Coordinates represented in spectral basis:

$\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathbf{R}^{\mathrm{n}}$.

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \alpha_1 \mathbf{b}_1 + \alpha_2 \mathbf{b}_2 + \ldots \alpha_n \mathbf{b}_n$$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \beta_1 \mathbf{b}_1 + \beta_2 \mathbf{b}_2 + \ldots \beta_n \mathbf{b}_n$$

$$\mathbf{Z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} = \gamma_1 \mathbf{b}_1 + \gamma_2 \mathbf{b}_2 + \ldots \gamma_n \mathbf{b}_n$$

Spectral mesh representation

Coordinates represented in spectral basis:

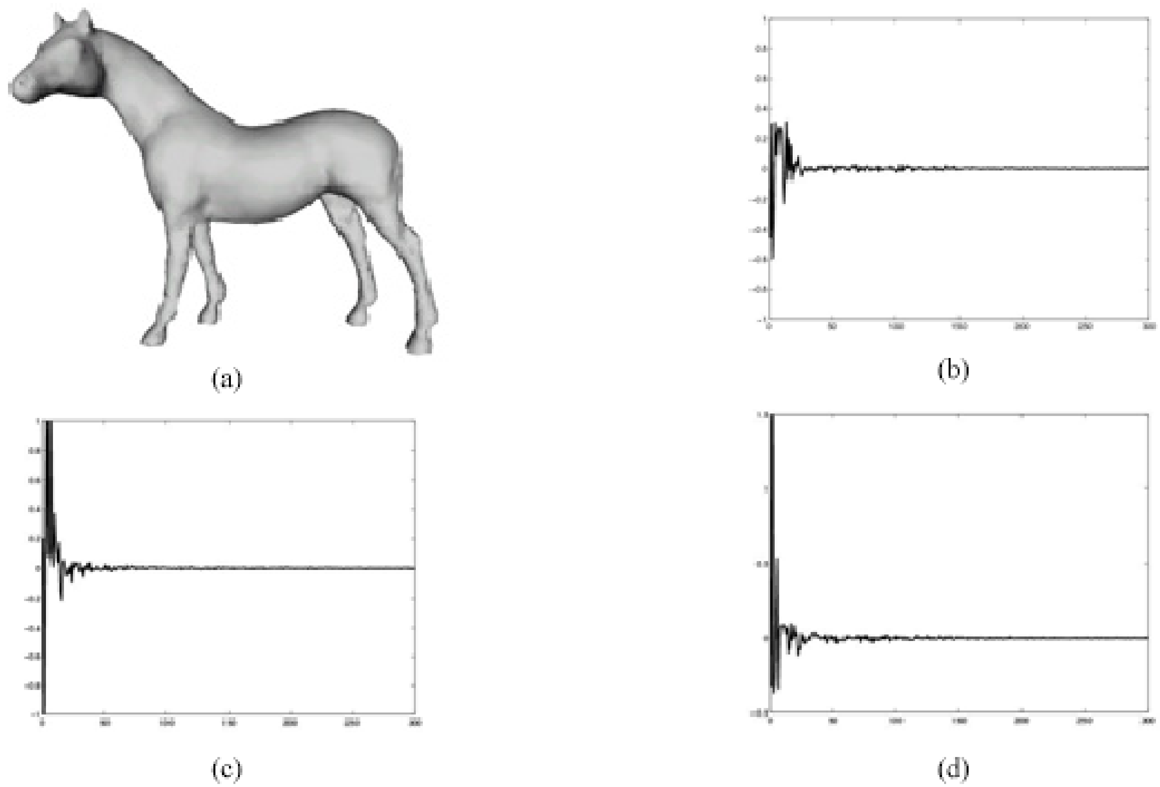
$$\begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \end{pmatrix}^{\mathrm{T}} \mathbf{b}_1 + \begin{pmatrix} \alpha_2 \\ \beta_2 \\ \gamma_2 \end{pmatrix}^{\mathrm{T}} \mathbf{b}_2 + \ldots + \begin{pmatrix} \alpha_n \\ \beta_n \\ \gamma_n \end{pmatrix}^{\mathrm{T}} \mathbf{b}_n$$

The first components are low-frequency

The last components are high-frequency

The spectral basis

Most shape information is in low-frequency components



[Karni and Gotsman 00]

Applications

Smoothing

Compression

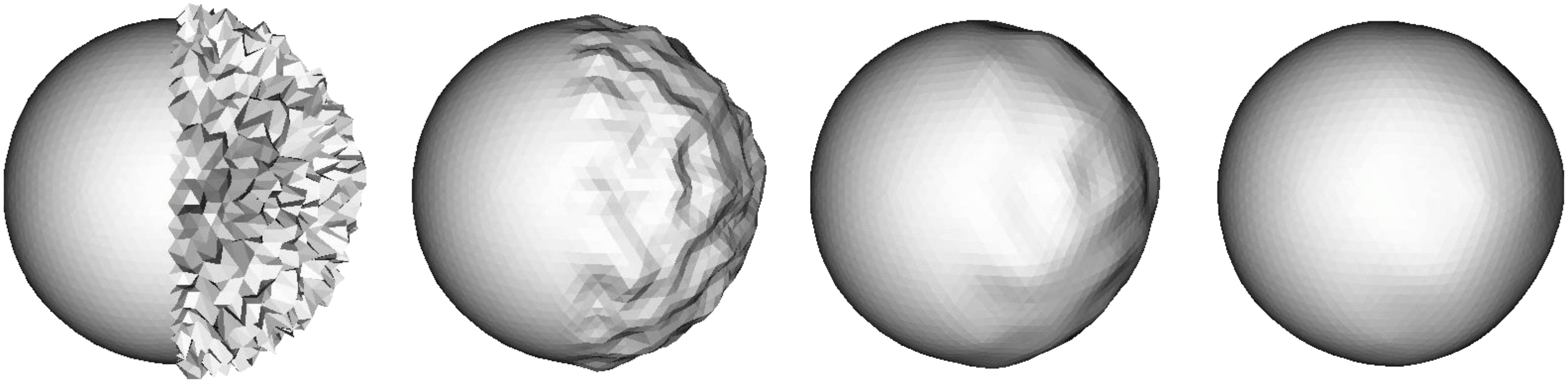
Progressive transmission

Watermarking

etc.

Mesh smoothing

Aim to remove high frequency details



[Taubin 95]

Spectral mesh smoothing

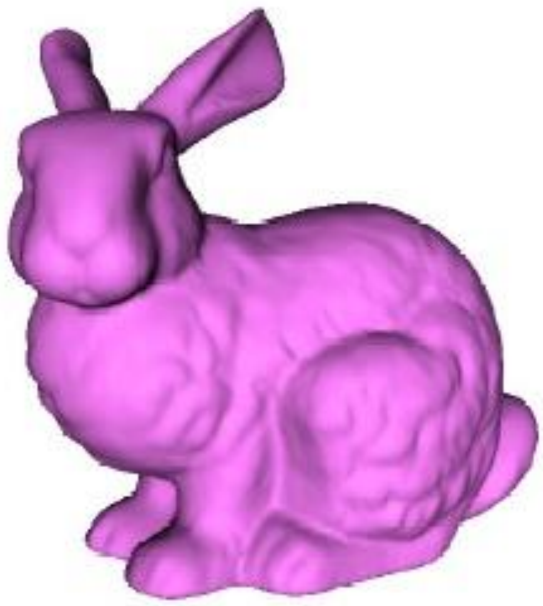
Drop the high-frequency components

$$\begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \end{pmatrix}^{\mathrm{T}} \mathbf{b}_1 + \begin{pmatrix} \alpha_2 \\ \beta_2 \\ \gamma_2 \end{pmatrix}^{\mathrm{T}} \mathbf{b}_2 + \ldots + \begin{pmatrix} \alpha_n \\ \beta_n \\ \gamma_n \end{pmatrix}^{\mathrm{T}} \mathbf{b}_n$$

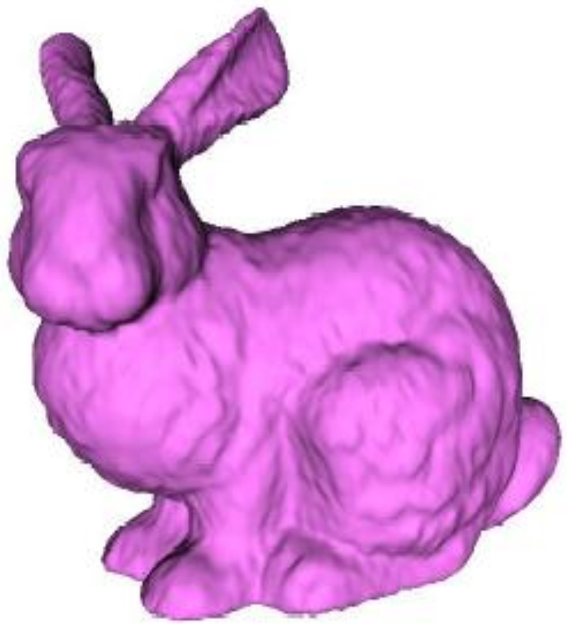
High-frequency components!

Mesh compression

Aim to represent surface with fewer bits



36 bits/vertex



1.4 bits/vertex

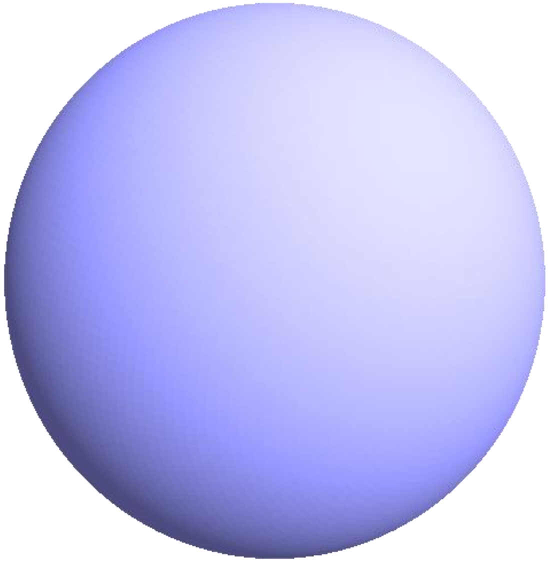
Mesh compression

Most of mesh data is in geometry

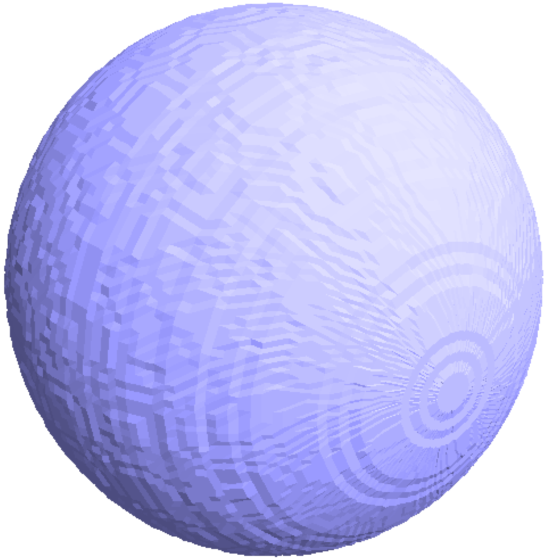
- The connectivity (the graph) can be very efficiently encoded
  - » About 2 bits per vertex only
- The geometry (x,y,z) is heavy!
  - » When stored naively, at least 12 bits per coordinate are needed, i.e. 36 bits per vertex

Mesh compression

What happens if quantize xyz coordinates?



original



8 bits/coordinate

Mesh compression

Quantization of the Cartesian coordinates introduces high-frequency errors to the surface.

High-frequency errors alter the visual appearance of the surface – affect normals and lighting.

Mesh compression

Transform the Cartesian coordinates to another space where quantization error will have low frequency in the regular Cartesian space

Quantize the transformed coordinates.

Low-frequency errors are less apparent to a human observer.

Spectral mesh compression

The encoding side:

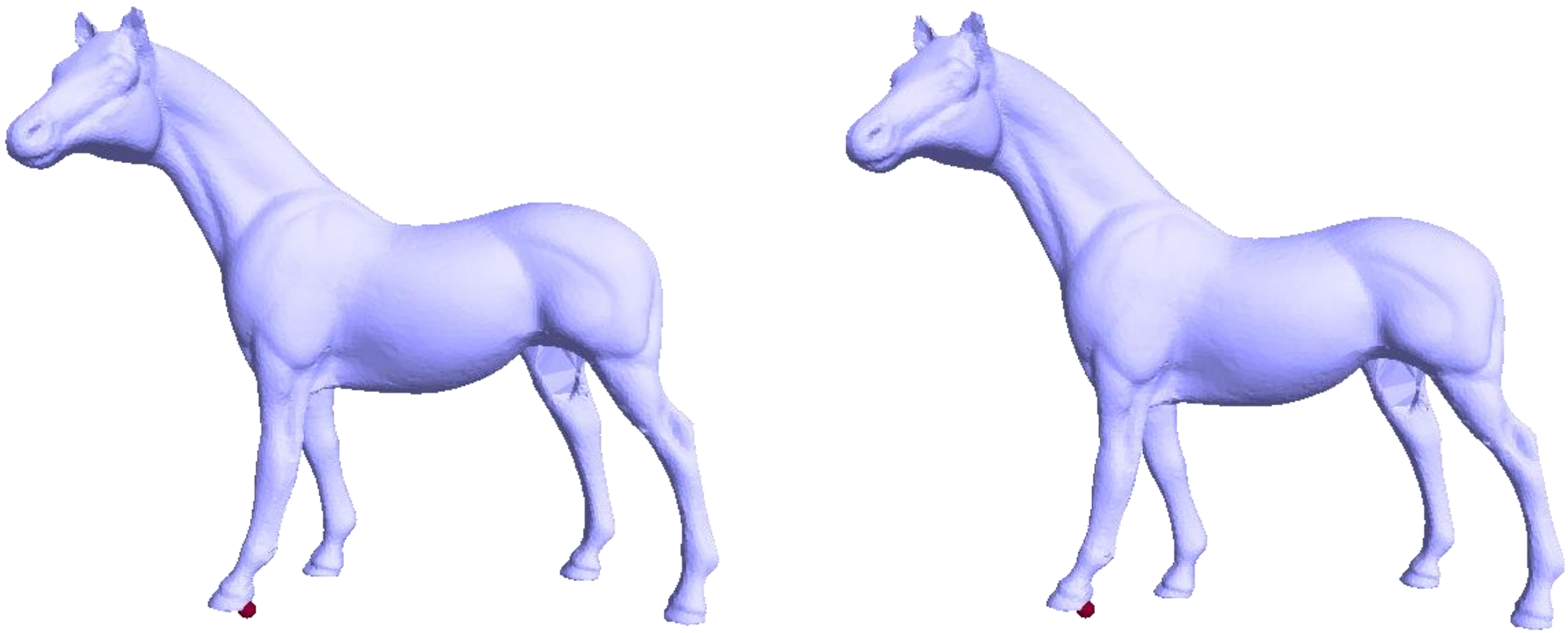
- Compute the spectral bases from mesh connectivity
- Represent the shape geometry in the spectral basis and decide how many coeffs. to leave (**K**)
- Store the connectivity and the **K** non-zero coefficients

The decoding side:

- Compute the first **K** spectral bases from the connectivity
- Combine them using the **K** received coefficients and get the shape

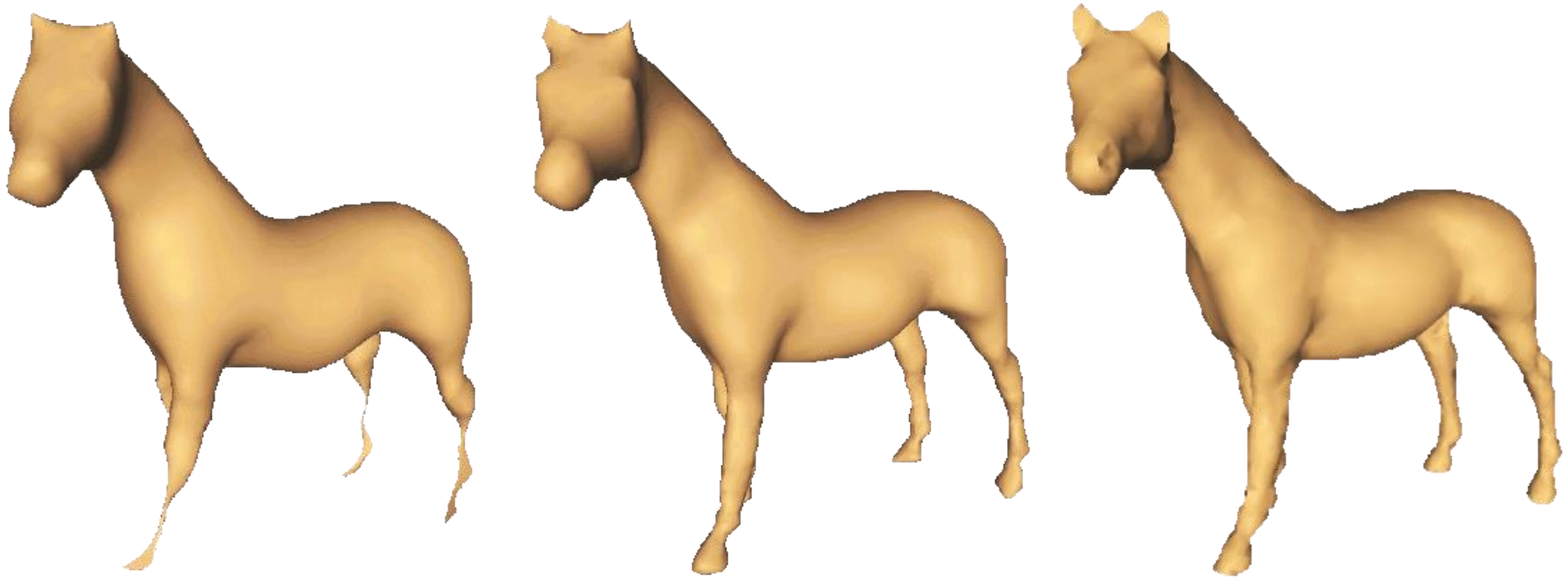
Spectral mesh compression

Low-frequency errors are hard to see



Progressive transmission

First transmit the lower-eigenvalue coefficients (low frequency components), then gradually add finer details by transmitting more coefficients.

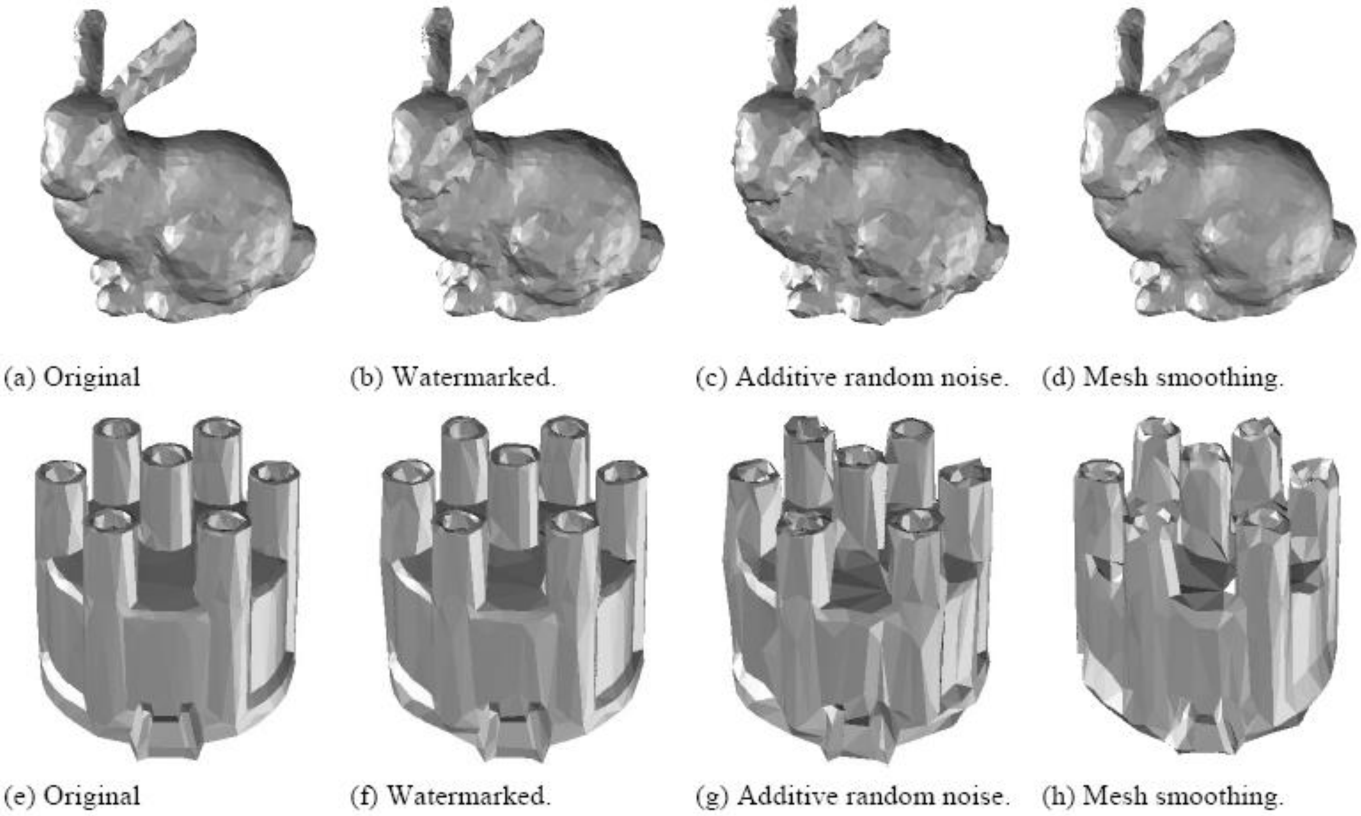


[Karni and Gotsman 00]

Mesh watermarking

Embed a bitstring in the low-frequency coefficients

- Low-frequency changes are hard to notice



(a) Original (b) Watermarked. (c) Additive random noise. (d) Mesh smoothing.

(e) Original (f) Watermarked. (g) Additive random noise. (h) Mesh smoothing.

[Ohbuchi et al. 2003]

Caveat

Performing spectral decomposition of a large matrix ($n > 1000$) is prohibitively expensive ($O(n^3)$)

- Today's meshes come with 50,000 and more vertices
- We don't want the decompressor to work forever!

Possible solutions:

- Simplify mesh
- Work on small blocks (like JPEG)