Spectral Meshes

COS 526: Advanced Computer Graphics

PRINCETON UNIVERSITY

Slide credits: Olga Sorkine, Bruno Lévy, Hao (Richard) Zhang

Motivation

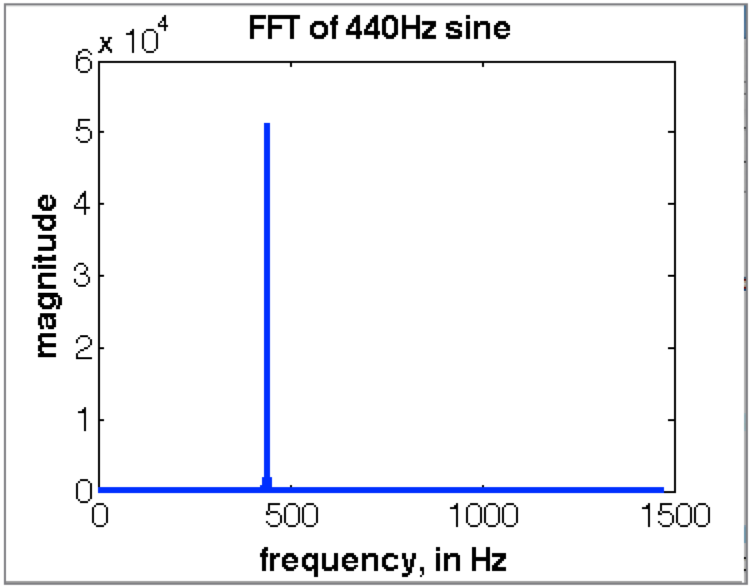
- Want "frequency domain" representation for 3D meshes
  - Smoothing
  - Compression
  - Progressive transmission
  - Watermarking
  - etc.
- Analogous to Fourier transform in 1D / 2D:
  - Express signal as sum of content at different frequencies

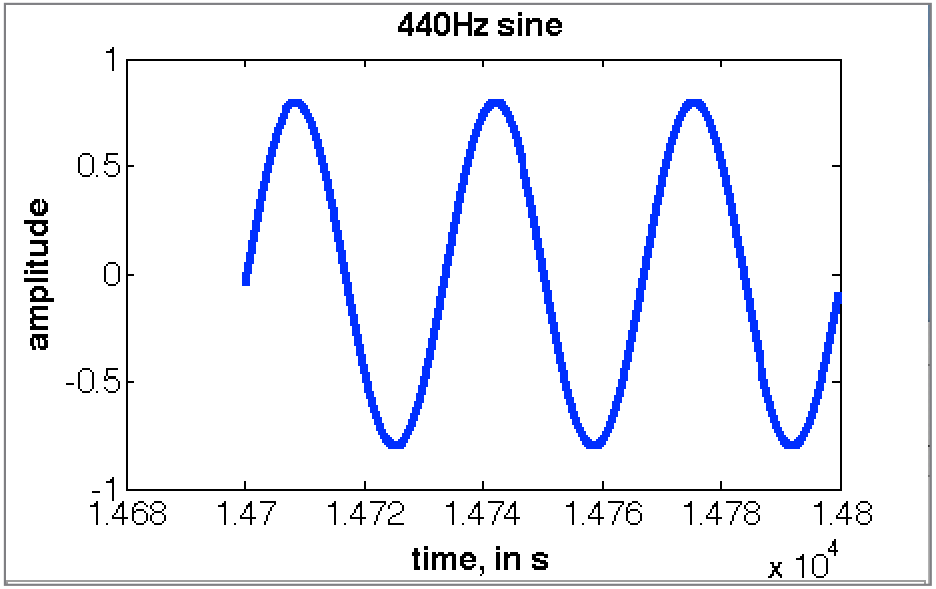
In the Frequency Domain…

Jean Baptiste Joseph Fourier (1768-1830)

Frequency Domain

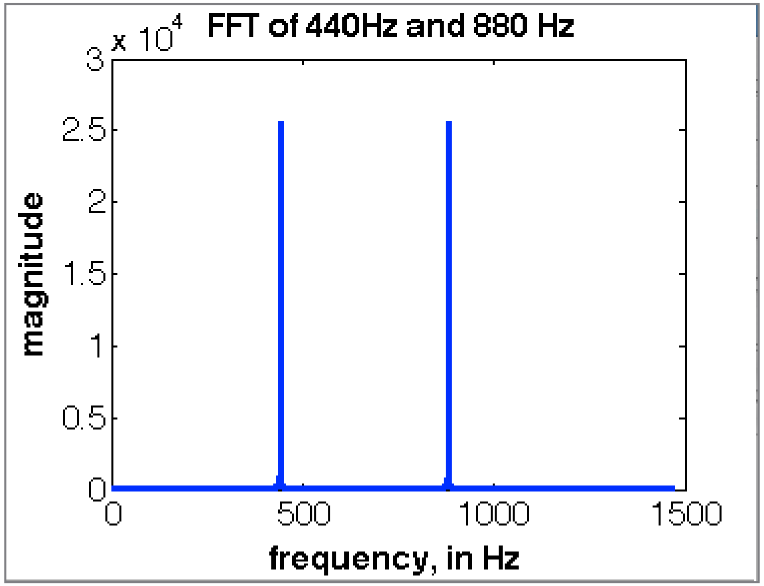
- Any signal can be represented as a sum of sinusoids at discrete **frequencies**, each with a given **magnitude** and **phase**

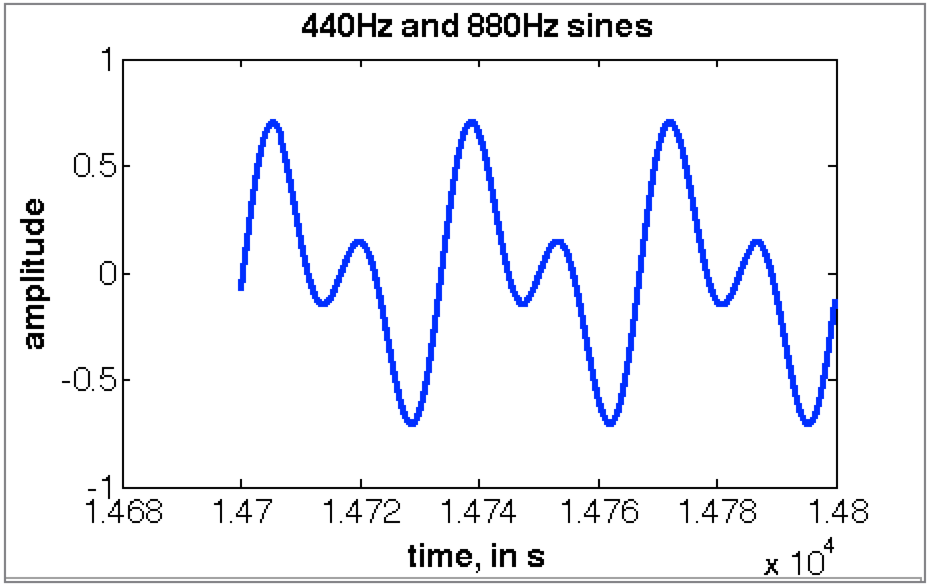




Frequency Domain

- Any signal can be represented as a sum of sinusoids at discrete **frequencies**, each with a given **magnitude** and **phase**

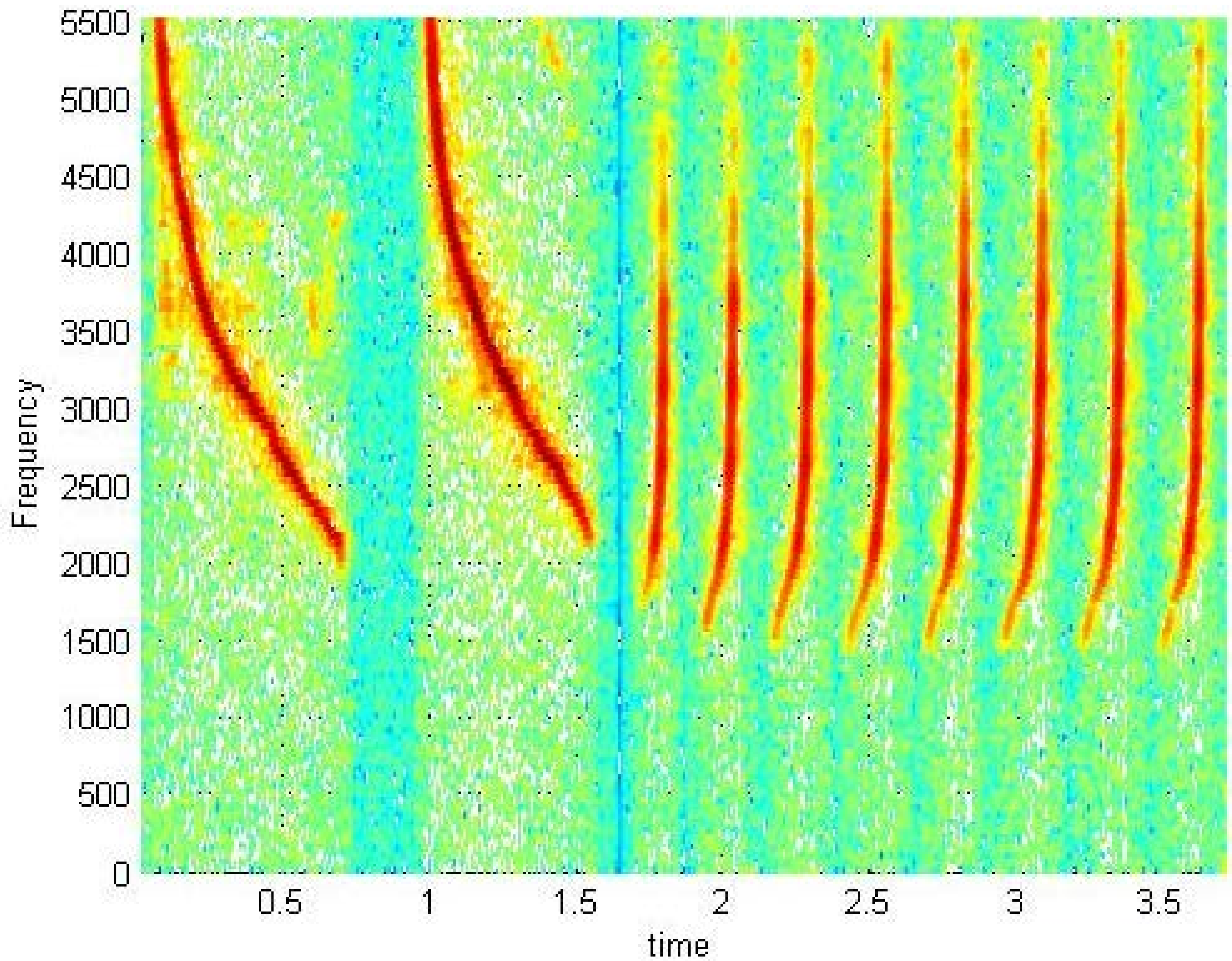




Frequency Content in Audio

- Frequency related to pitch:
  - "A 440": 440 Hz
  - 880Hz: one octave above
- Frequency also related to timbre:
  - Real sounds contain many frequencies
  - Higher frequency content can make sounds "brighter"
- In speech, higher frequencies are related to vowels, consonants (independent of spoken/sung pitch)

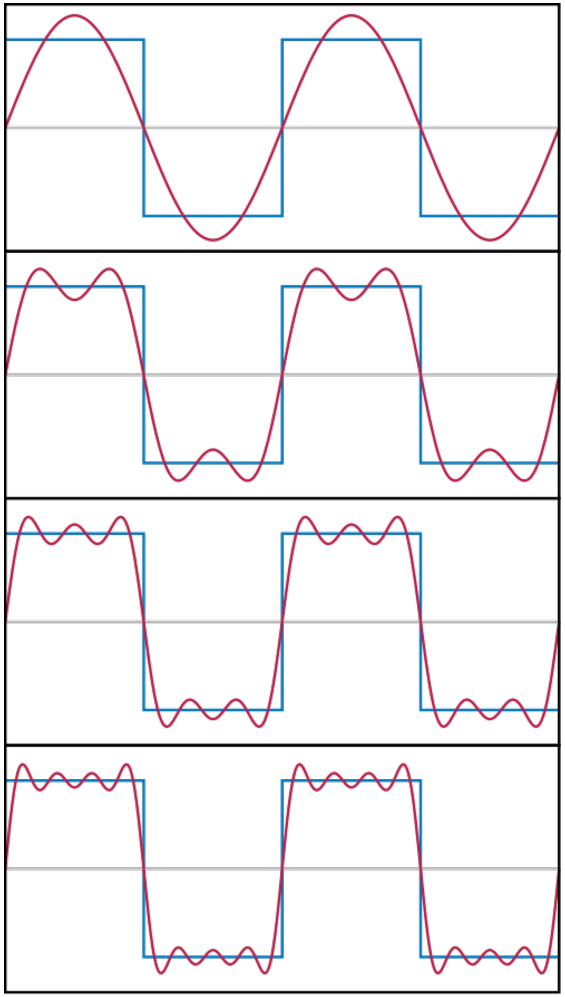
Spectrogram, Northern Cardinal



Fourier Series: Building Up a Function

- Periodic function f(x) defined over $[-\pi \,..\, \pi]$

$$f(x) = \tfrac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx)$$



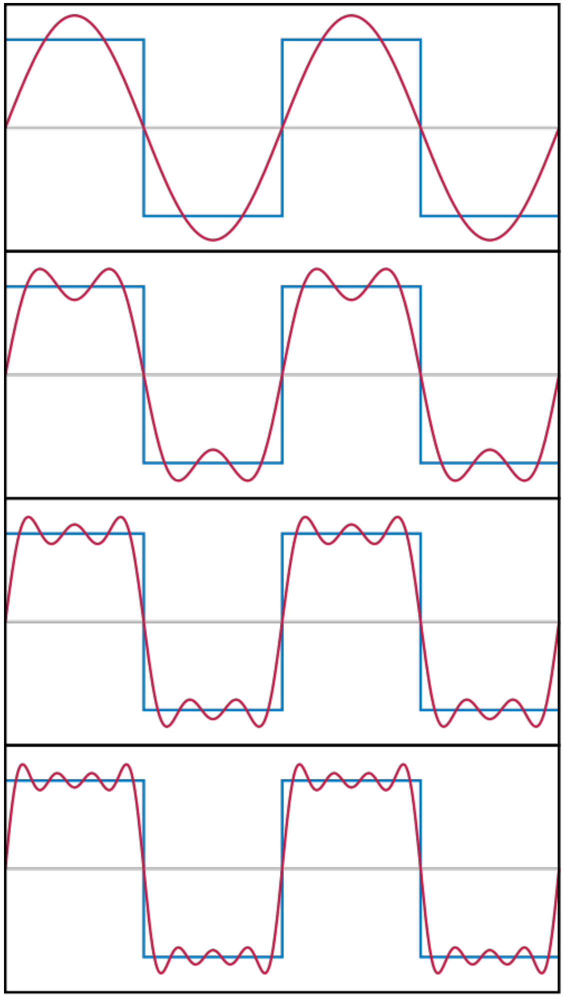
Fourier Series: Finding Coefficients

- Periodic function f(x) defined over $[-\pi \,..\, \pi]$

$$f(x) = \tfrac{1}{2} a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx)$$

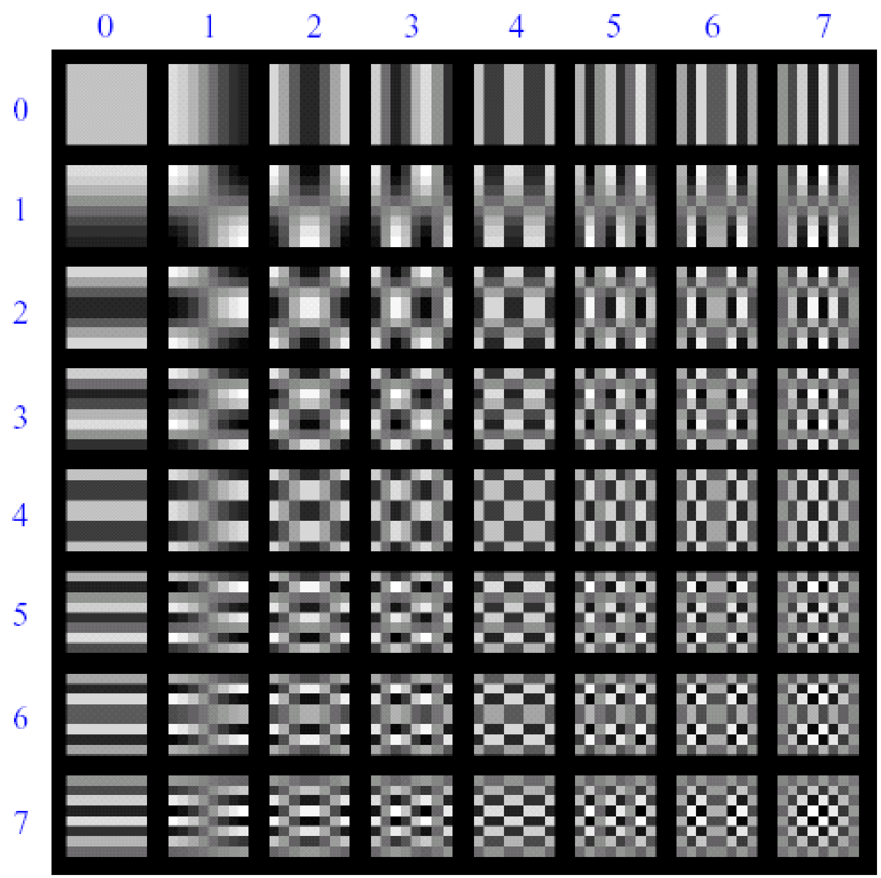
$$a_n = \tfrac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\, dx$$

$$b_n = \tfrac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\, dx$$



Fourier Series in 2D

- 2D bases for 2D signals (images)



$$a\cos(n_x x)\cos(n_y y)$$

Fourier Transform

- Transform applied to function to analyze a signal's frequency content
- Several versions:

	Continuous Time	**Discrete Time**
Aperiodic / unbounded time, continuous frequency	Fourier Transform	Discrete-time Fourier Transform (DTFT)
Periodic or bounded time, discrete frequency	Fourier Series	Discrete Fourier Transform (DFT) (can use FFT for this)

Applying Euler's Formula

- Euler's formula:

$$e^{ix} = \cos(x) + i\sin(x)$$

- Apply:

$$f(x) = \tfrac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx)$$

becomes

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}$$

where

$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-inx}\,dx$$

Fourier Transform

- [Continuous] Fourier transform:

$$F(k) = \mathcal{F}\big(f(x)\big) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i k x} dx$$

- Discrete Fourier transform:

$$F_k = \sum_{x=0}^{n-1} f_x\, e^{-2\pi i \frac{k}{n} x}$$

- *F* is a function of frequency – describes "how much" *f* contains of sinusoids at frequency *k*
- Fourier transform is invertible

DFT and Inverse DFT (IDFT)

$$F_k = \sum_{x=0}^{n-1} f_x \, e^{-2\pi i \frac{k}{n} x}$$

$$\updownarrow$$

$$f_x = \frac{1}{n} \sum_{k=0}^{n-1} F_k e^{2\pi i \frac{k}{n} x}$$

Computing Discrete Fourier Transform

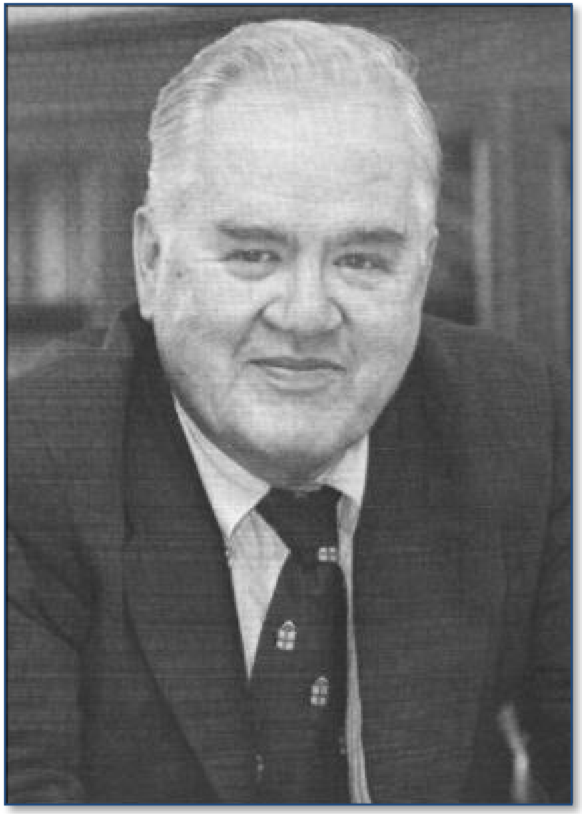
$$F_k = \sum_{x=0}^{n-1} f_x \, e^{-2\pi i \frac{k}{n} x}$$

- Straightforward computation: for each of *n* DFT values, loop over *n* input samples. Total: $O(n^2)$
- Fast Fourier Transform (FFT): $O(n \log_2 n)$ time

The FFT

Discovered by Johann Carl Friedrich Gauss (1777-1855)

The FFT



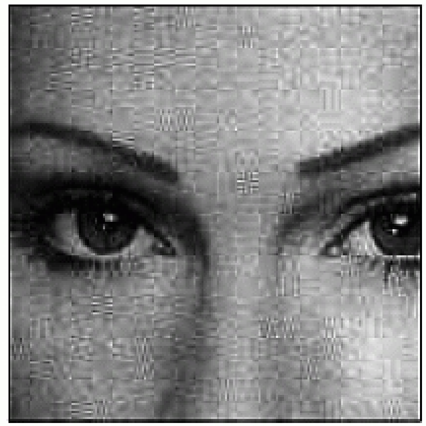
Rediscovered and popularized in 1965 by
J. W. Cooley and John Tukey (Princeton alum and faculty)

Computing Discrete Fourier Transform

- Fast Fourier Transform (FFT): $O(n \log_2 n)$ time
  - Revolutionized signal processing, filtering, compression, etc.
  - Also turns out to have less roundoff error

JPEG Image Compression

a. Original image



b. With 10:1 compression

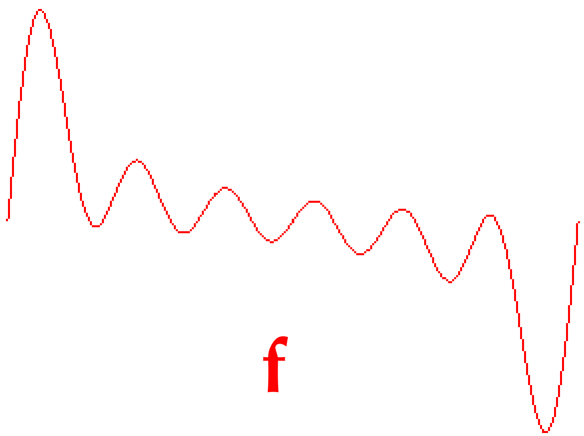
FIGURE 27-15
Example of JPEG distortion. Figure (a) shows the original image, while (b) and (c) shows restored images using compression ratios of 10:1 and 45:1, respectively. The high compression ratio used in (c) results in each 8×8 pixel group being represented by less than 12 bits.

c. With 45:1 compression

Discrete Cosine Transform (DCT)

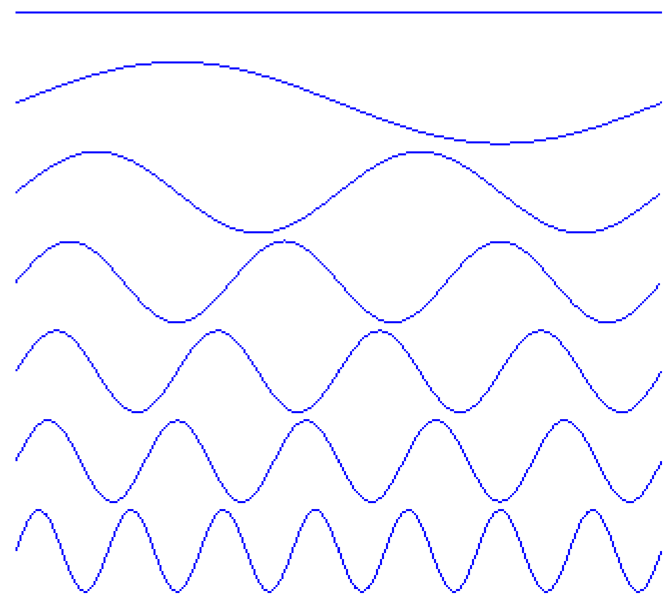
[Steven W. Smith 1997]

Filtering in the Frequency Domain



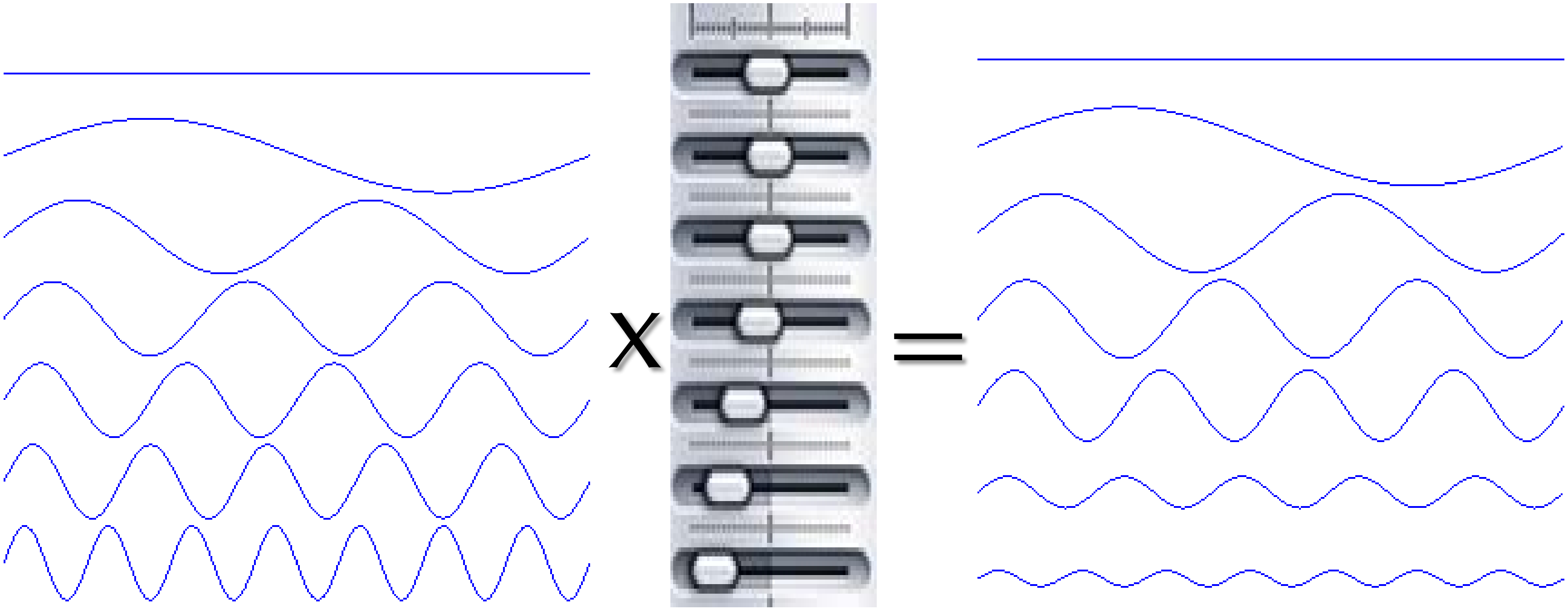
f

$$=\Sigma$$

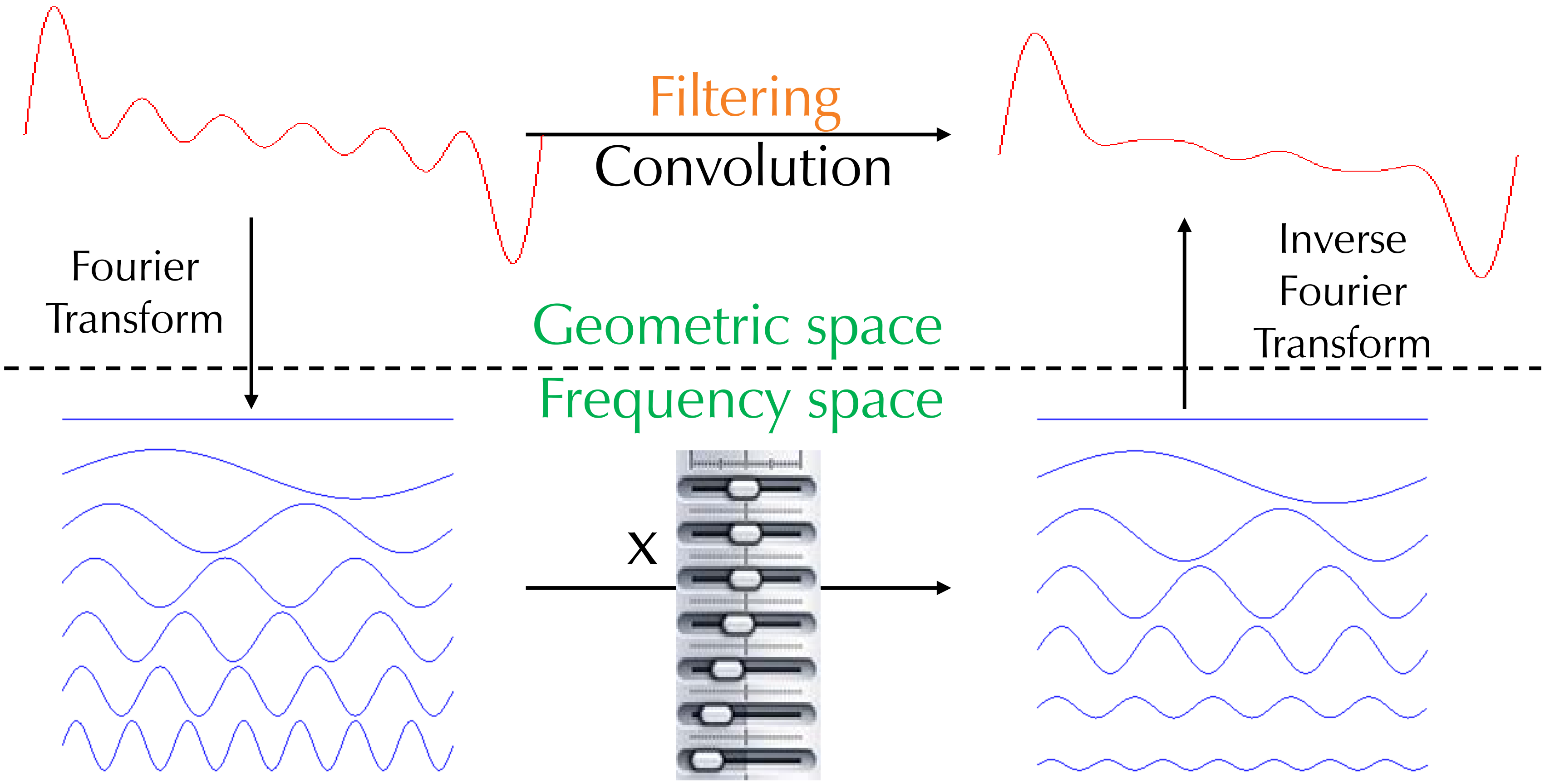


sin(kx)

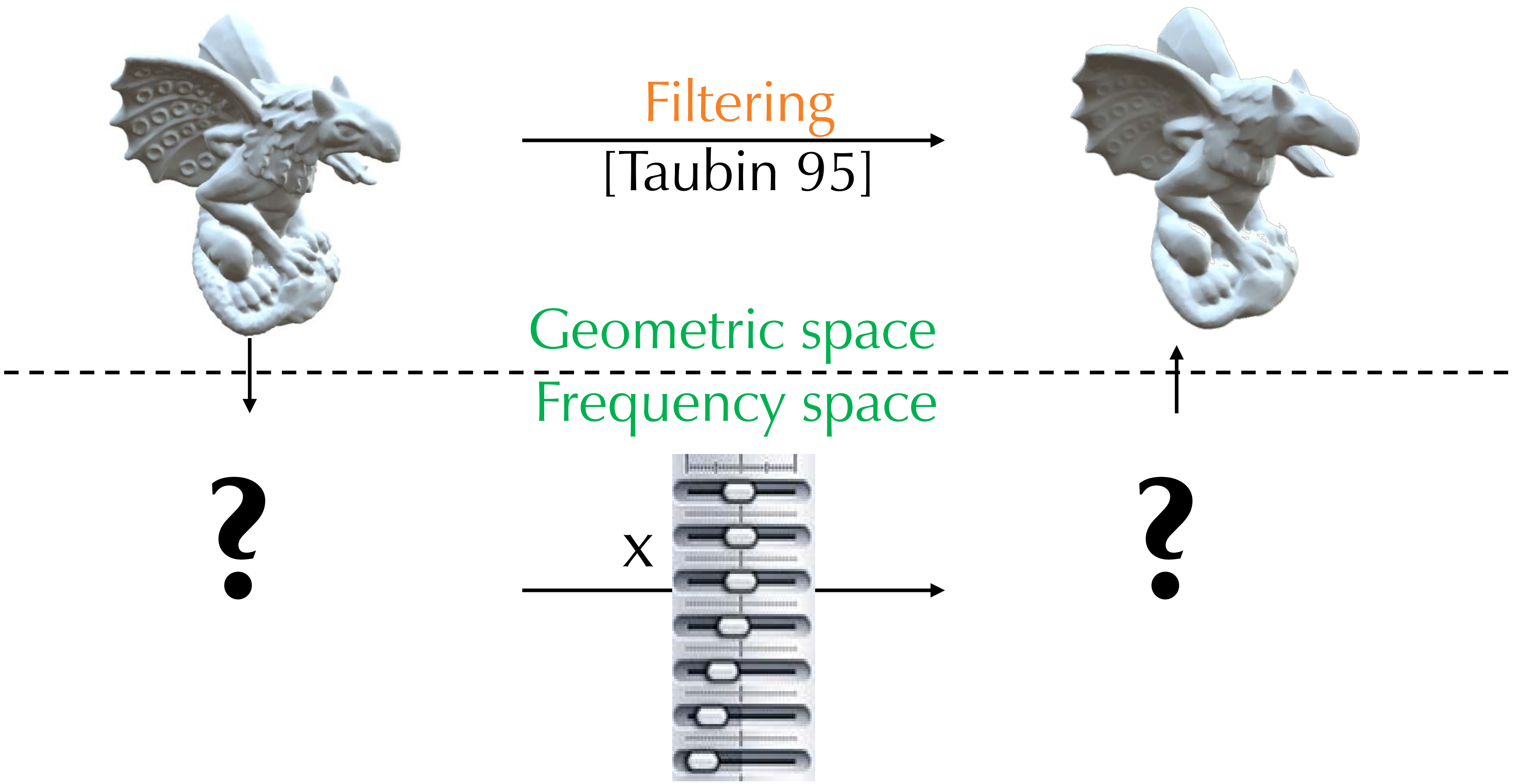
Filtering in the Frequency Domain



Filtering

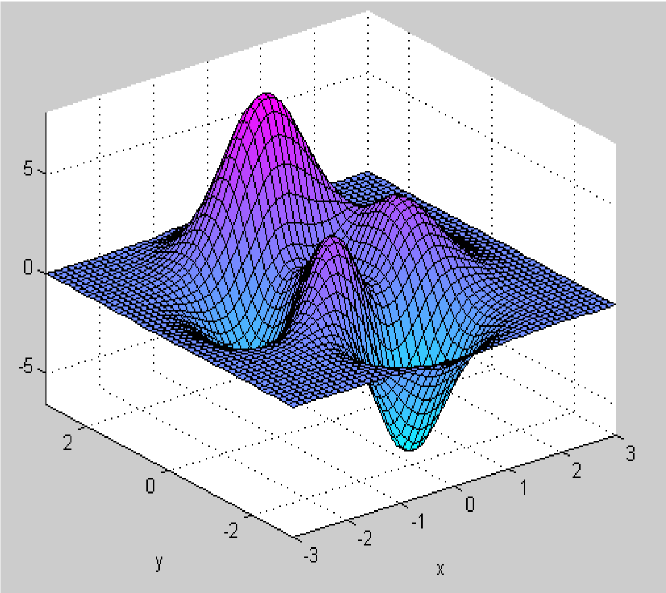

Filtering
Convolution
Fourier
Transform
Inverse
Fourier
Transform
Geometric space
Frequency space
X

Filtering on a Mesh?

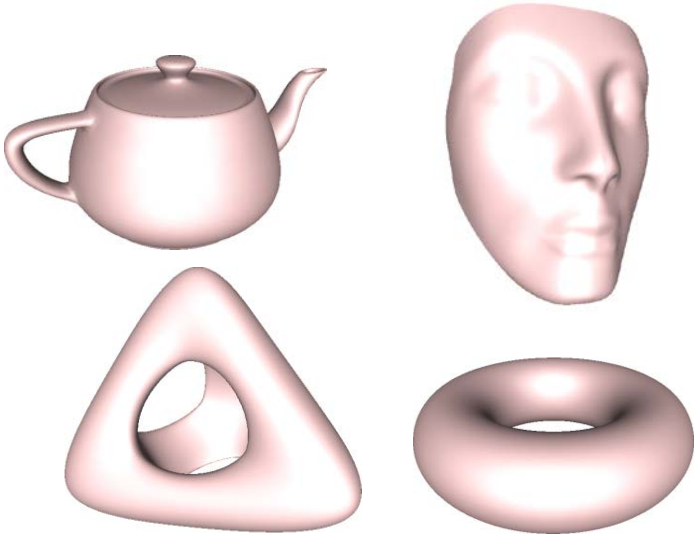


Filtering on a Mesh?

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



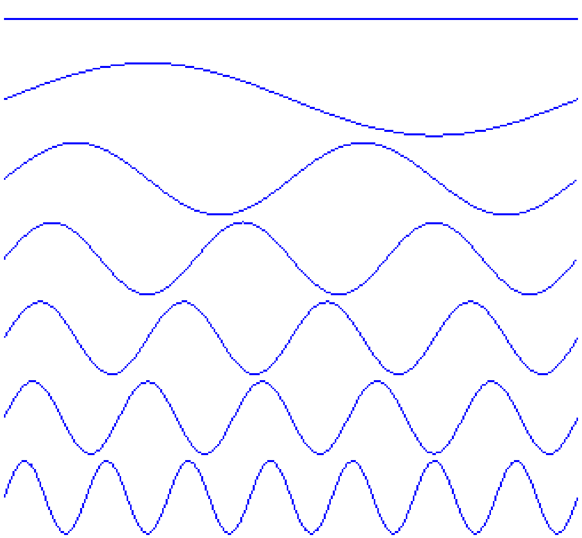
Height function, regularly sampled above a 2D domain



General 3D shapes

Basis Functions for 3D Meshes

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



sin(kx)

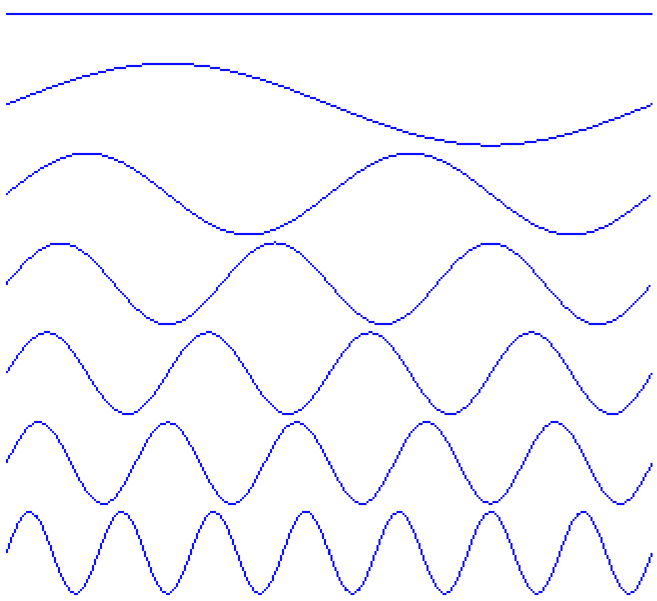
on

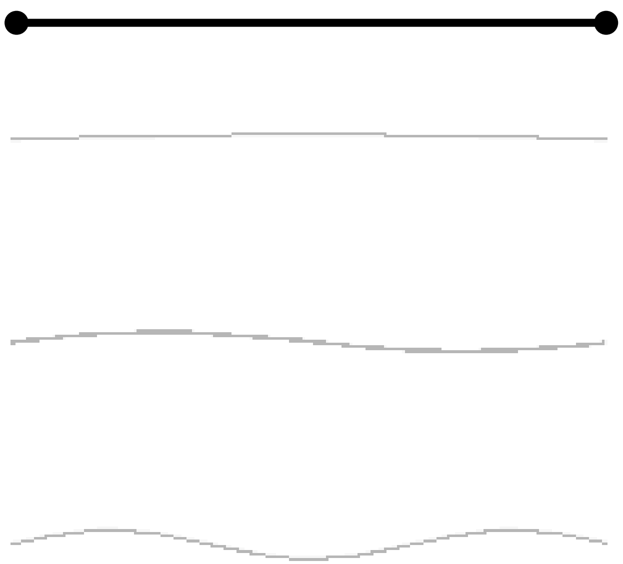
?

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

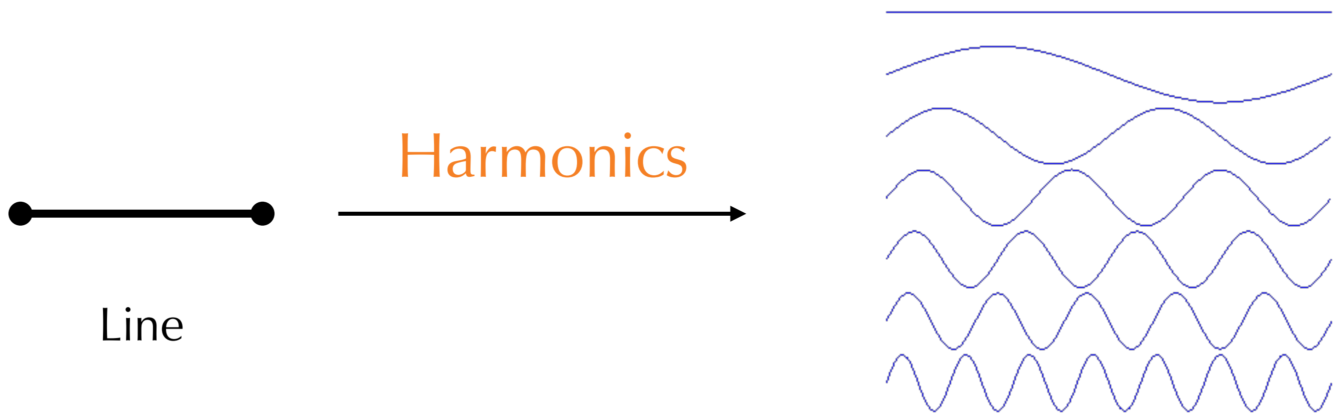
Harmonics





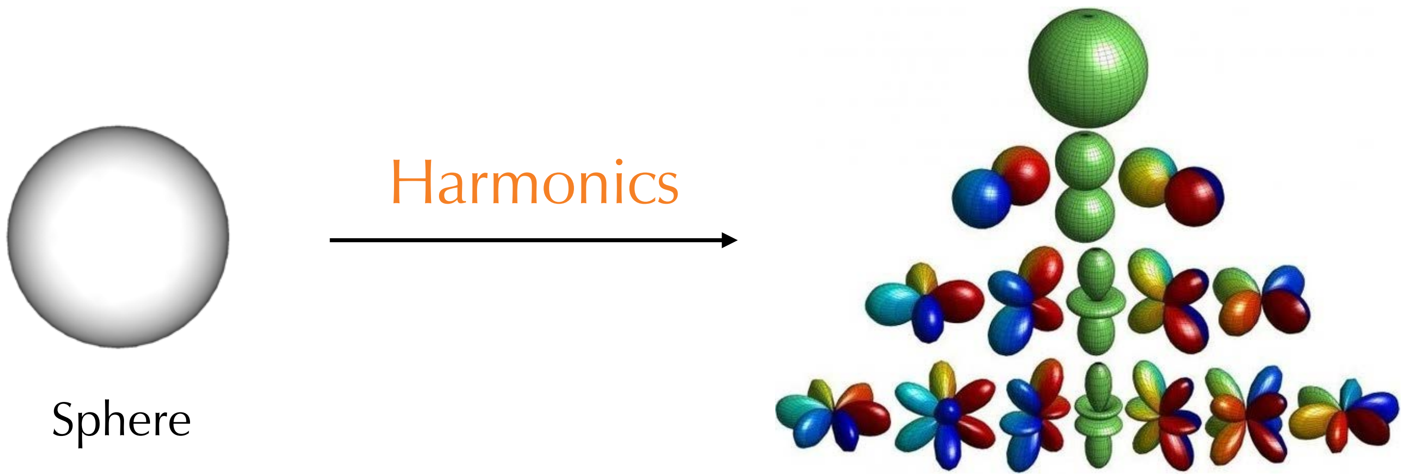
sin(kx) are the stationary vibrating modes = harmonics of a string

Harmonics



Stationary vibrating modes

Spherical Harmonics



Stationary vibrating modes

- You may recognize these from chemistry as "electron orbitals"

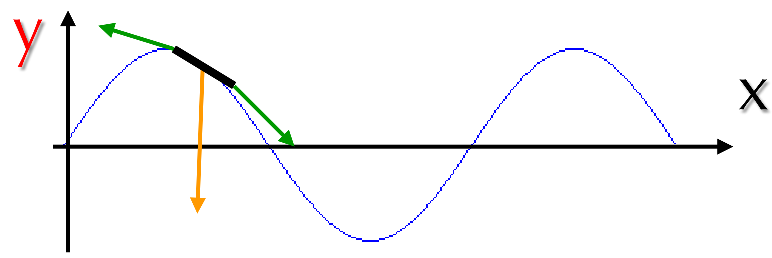
Manifold Harmonics

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

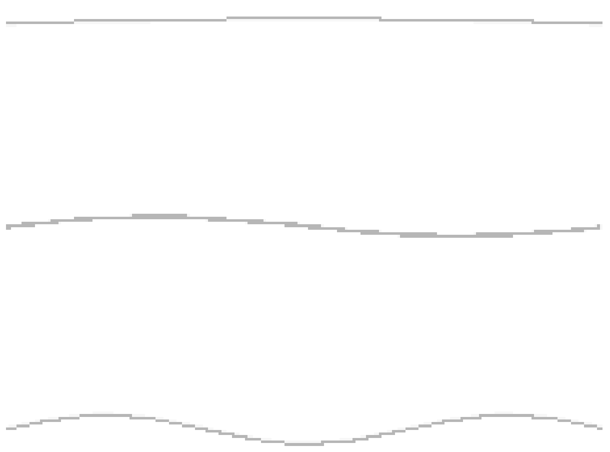
Stationary vibrating modes

Harmonics

- **Wave equation**:
  $T \, \partial^2 y/\partial x^2 = \mu \, \partial^2 y/\partial t^2$
  T: stiffness μ: mass



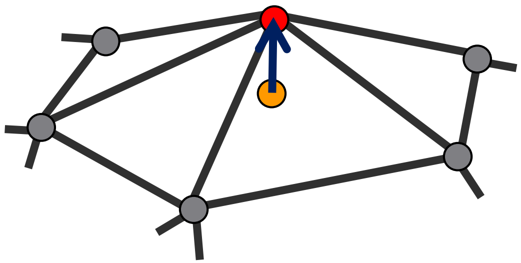
- **Stationary modes**:
  $y(x,t) = y(x)\sin(\omega t)$
  $\partial^2 y/\partial x^2 = -\mu\omega^2/T \, y$
  - **eigenfunctions** of $\partial^2/\partial x^2$



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

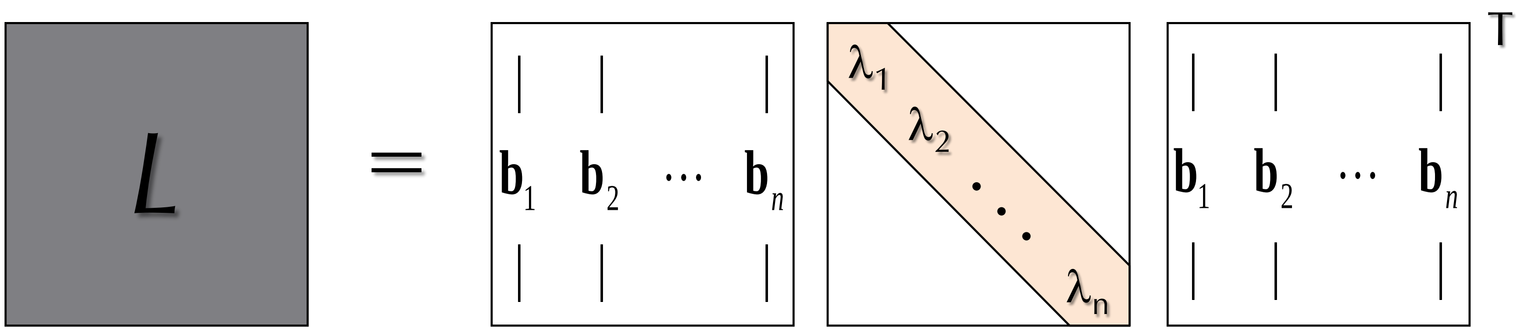
Laplacian Operator in Matrix Form

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

L matrix

Spectral Bases

- L is a symmetric $n \times n$ matrix
- Eigenfunctions of L computed with spectral analysis

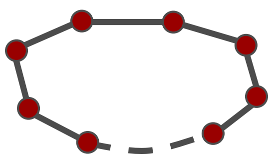


Basis vectors

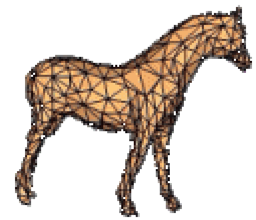
Frequencies, sorted in ascending order

The Spectral Basis

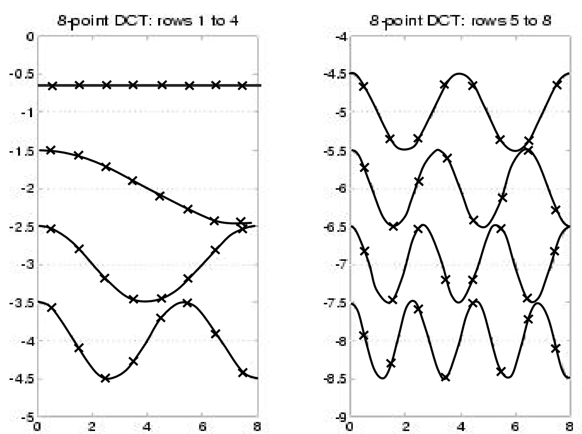
- First functions are smooth and slow, last oscillate a lot



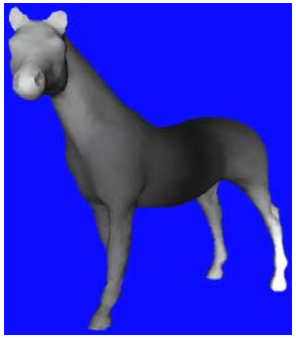
chain connectivity



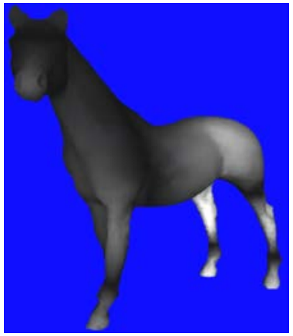
horse connectivity



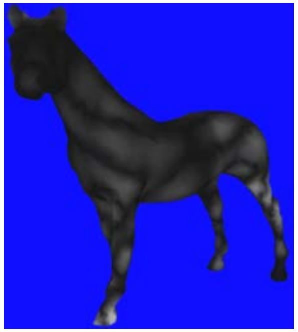
spectral basis of L =
the DCT basis



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

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

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

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

Spectral Mesh Representation

- Coordinates represented in spectral basis:

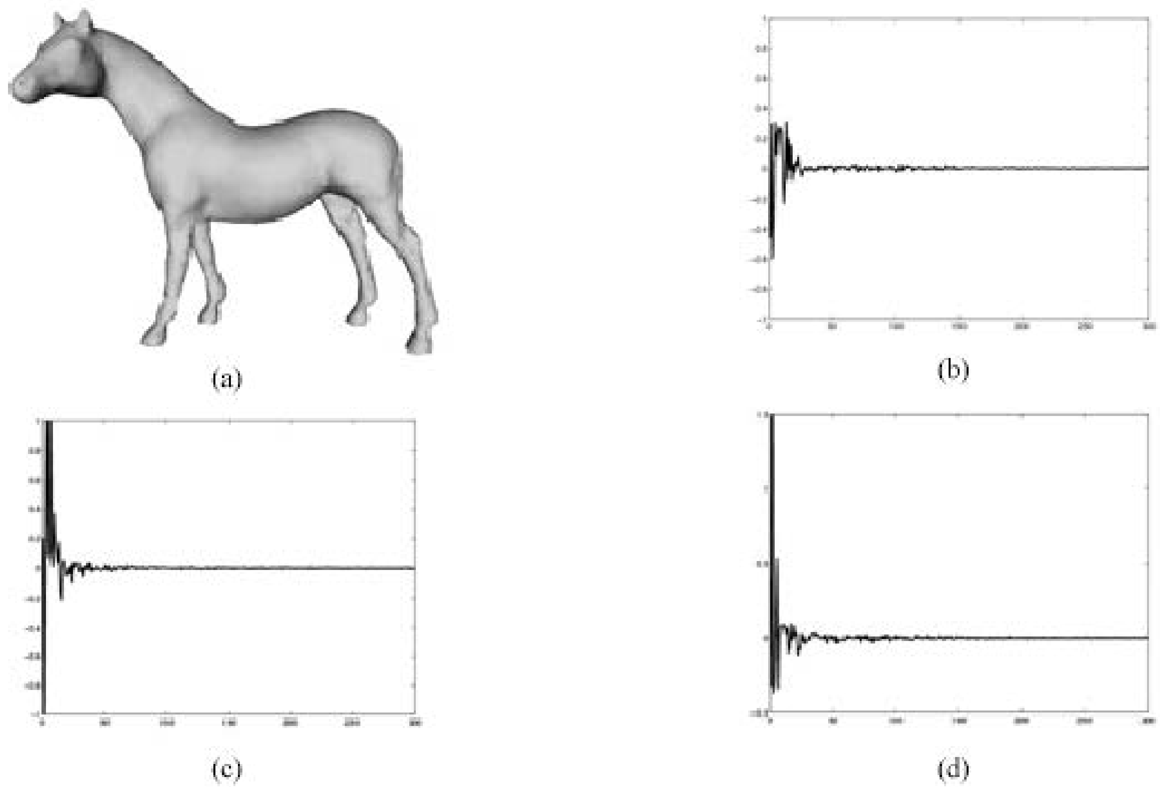
$$\begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \end{pmatrix}^{\mathrm{T}} \mathbf{b}_1 + \begin{pmatrix} \alpha_2 \\ \beta_2 \\ \gamma_2 \end{pmatrix}^{\mathrm{T}} \mathbf{b}_2 + \ldots + \begin{pmatrix} \alpha_n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

The first components are low-frequency

The last components are high-frequency

The Spectral Basis

- Most shape information is in low-frequency components



(a) (b) (c) (d)

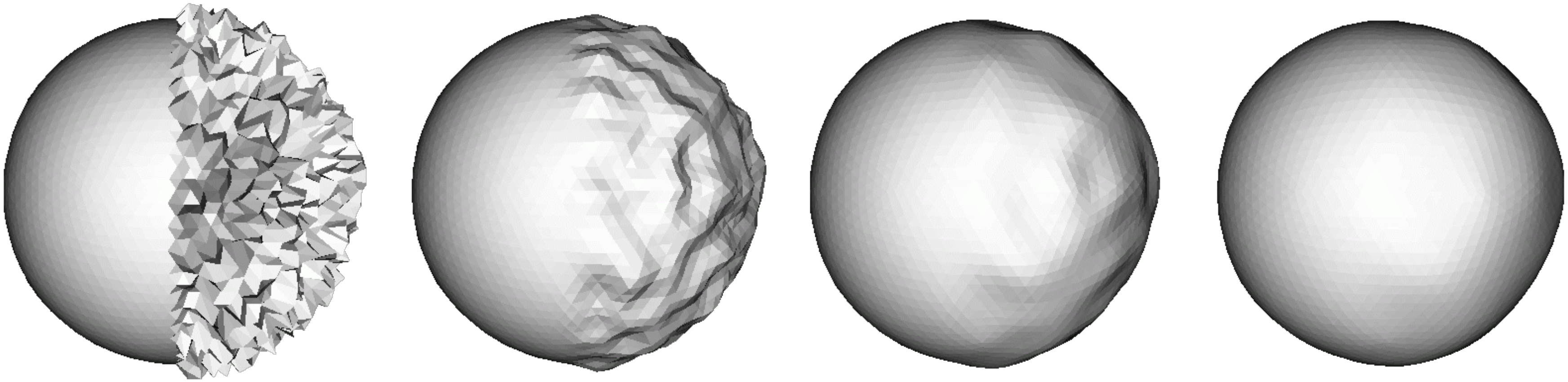
[Karni and Gotsman 00]

Applications

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

Mesh Smoothing

- Aim to remove high frequency details



[Taubin 95]

Spectral Mesh Smoothing

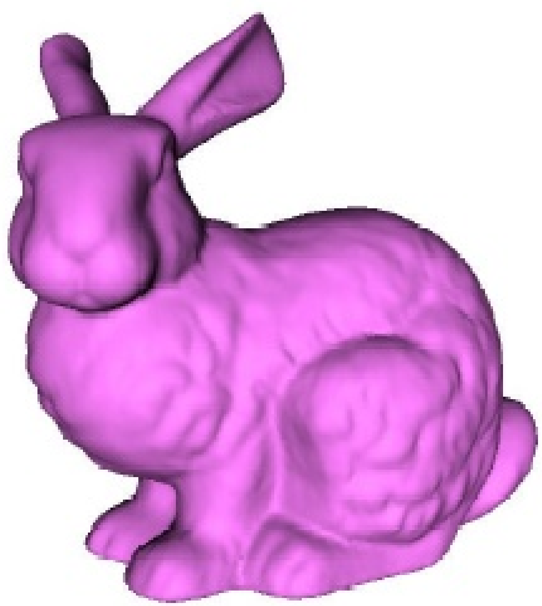
- Drop the high-frequency components

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

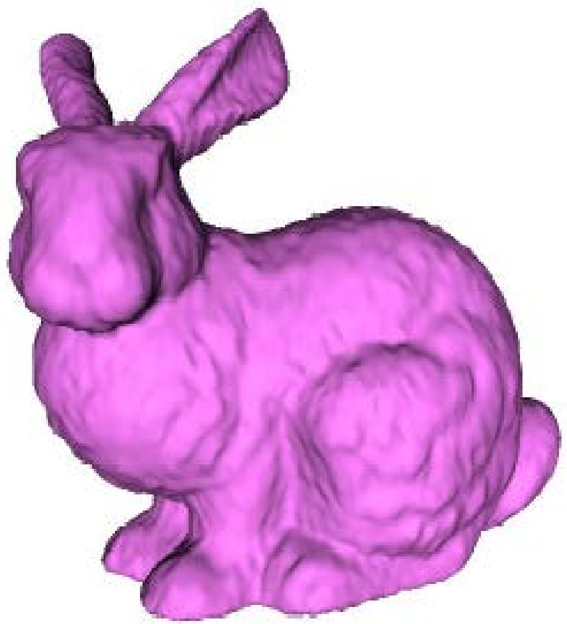
High-frequency components!

Mesh Compression

- Aim to represent surface with fewer bits



36 bits/vertex



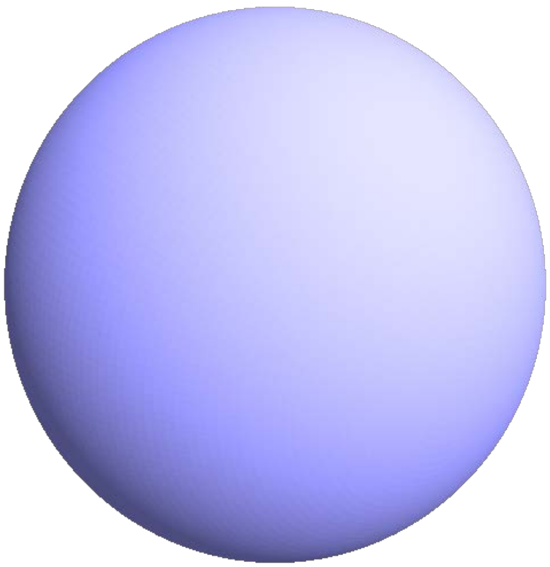
1.4 bits/vertex

Mesh Compression

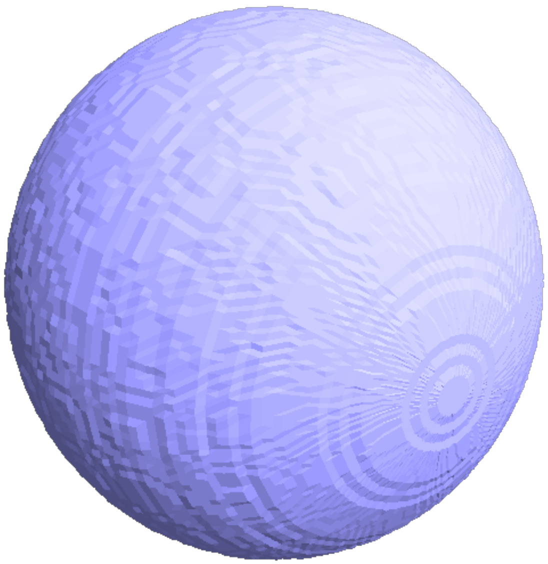
- Most of mesh data is in geometry
  - The connectivity (the graph) can be very efficiently encoded
    - About 2 bits per vertex only
  - The geometry (x,y,z) is heavy!
    - When stored naively, at least 12 bits per coordinate are needed, i.e. 36 bits per vertex

Mesh Compression

- What happens if we just 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

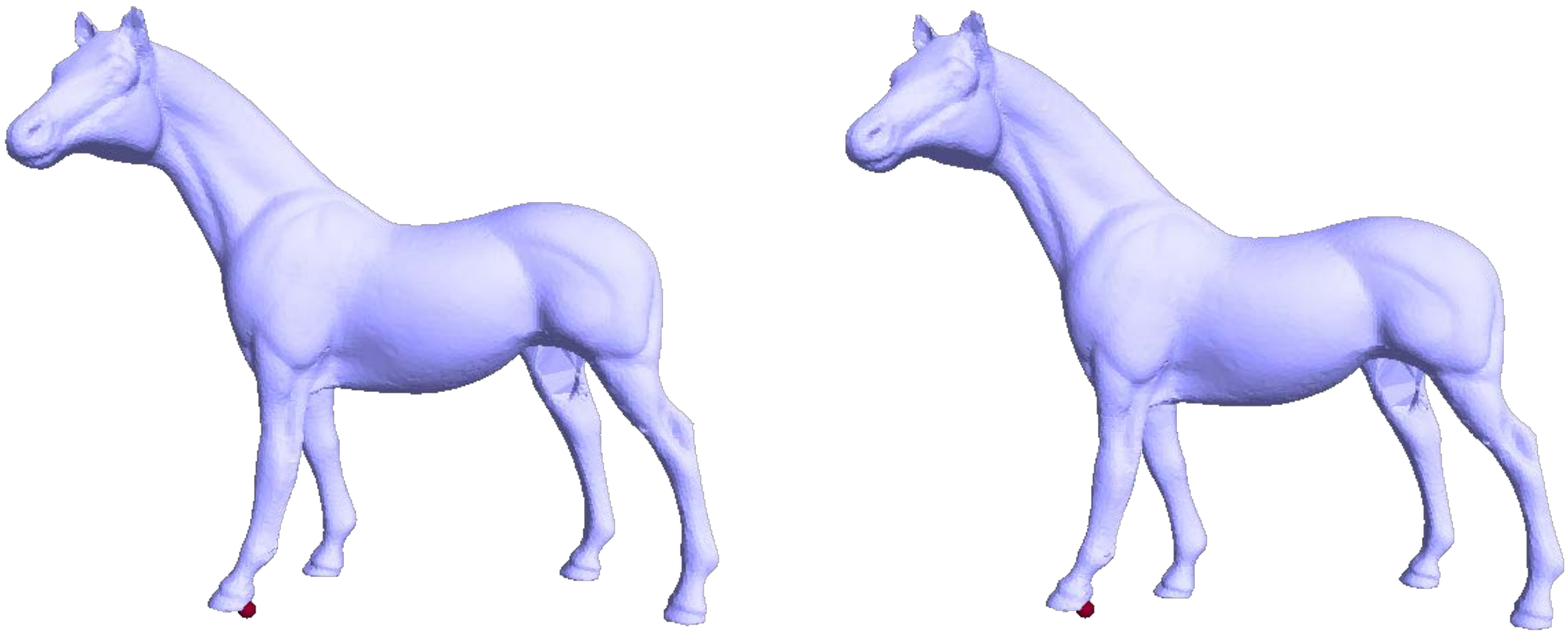
- Transform the Cartesian coordinates to another space where quantization error will have low frequency in the regular Cartesian space
- Quantize the transformed coordinates
- Low-frequency errors are less apparent to a human observer

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

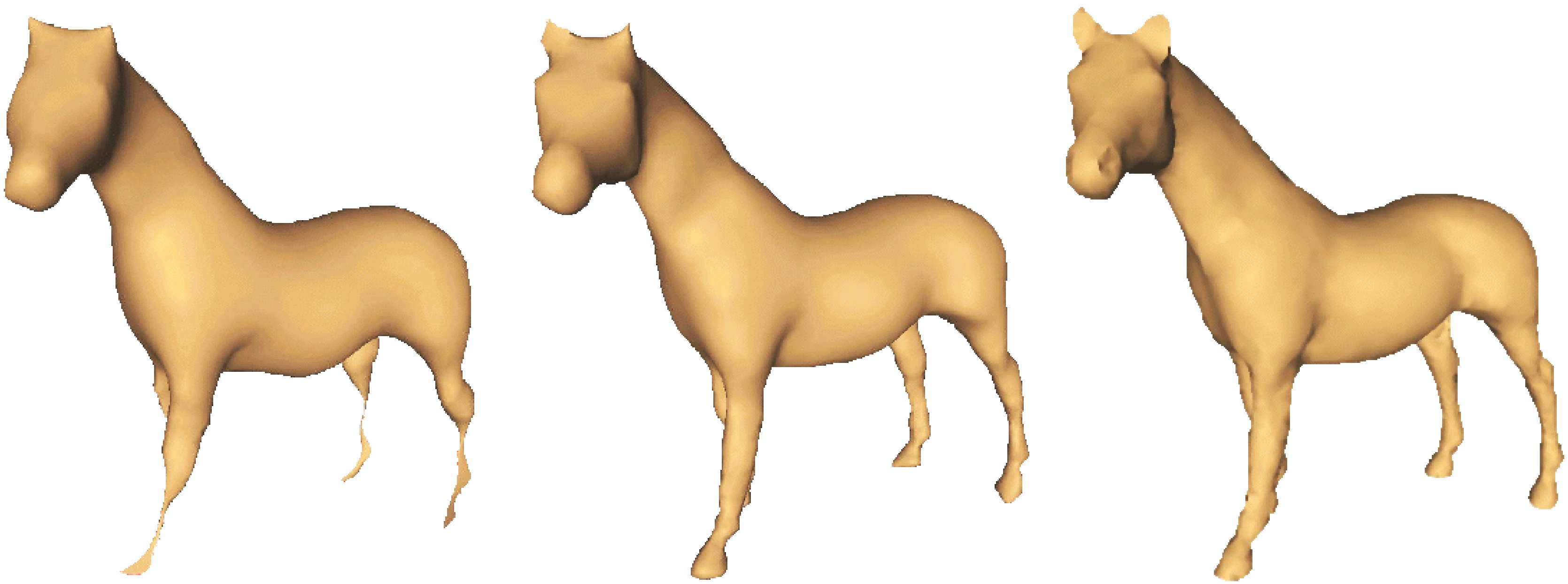
Spectral Mesh Compression

- Low-frequency errors are hard to see



Progressive Transmission

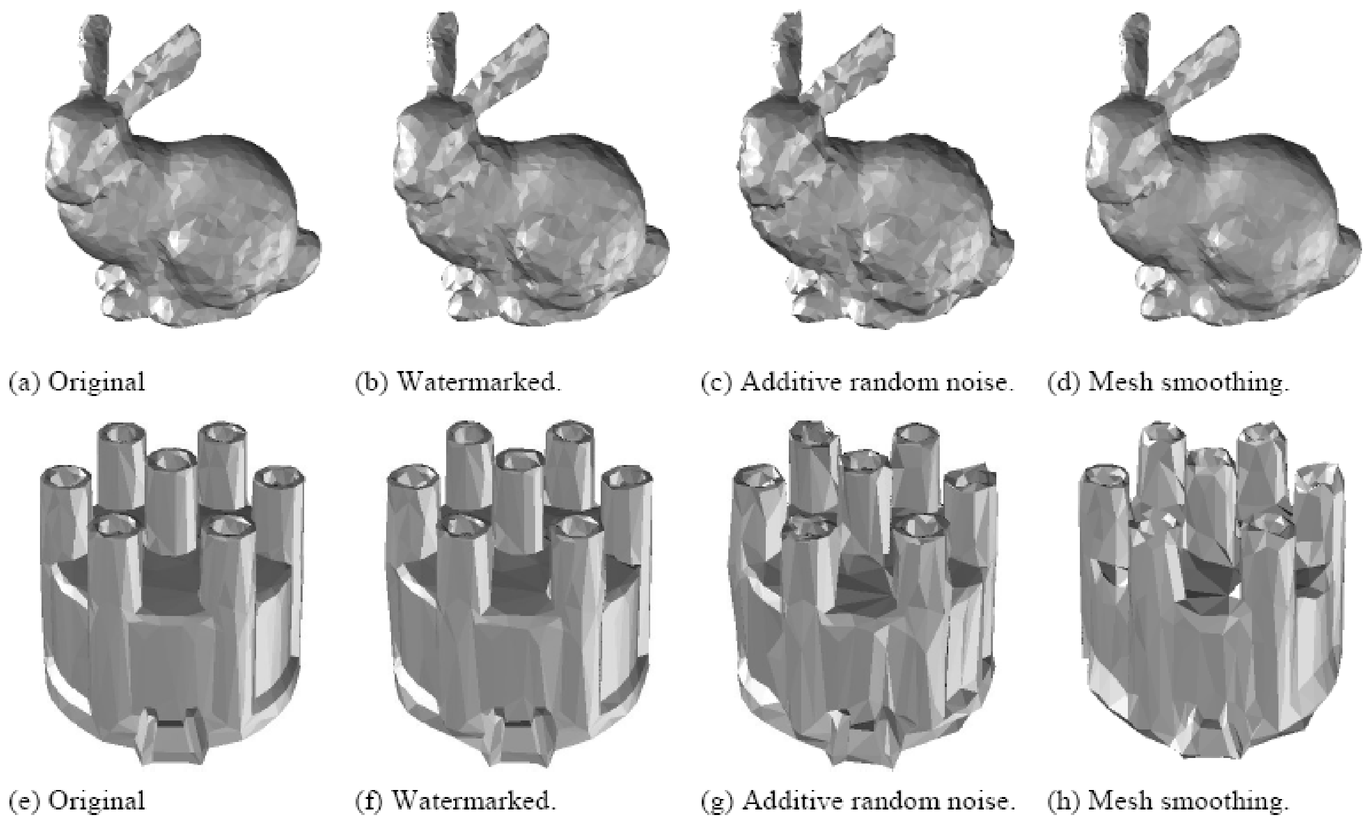
- First transmit the lower-eigenvalue coefficients (low frequency components), then gradually add finer details by transmitting more coefficients



[Karni and Gotsman 00]

Mesh Watermarking / Steganography

- Embed a bitstring in the low-frequency coefficients



[Ohbuchi et al. 2003]

Caveat

- Performing spectral decomposition of a large matrix (n>1000) is expensive – $O(n^3)$
  - No FFT because of lack of regular structure
- Possible solutions:
  - Simplify mesh
  - Work on small blocks (like JPEG)