Spectral Mesh Representation

Slides from Olga Sorkine

Motivation

- Want frequency domain representation for 3D meshes
  - Smoothing
  - Compression
  - Progressive transmission
  - Watermarking
  - etc.

Frequencies in a mesh

- One possibility = multires meshes

Frequencies in a function

- Fourier analysis (DCT)
  - cosine bases for signals:
    - \( \cos \left( \frac{2\pi d}{N} x \right) \)
    - \( \cos \left( \frac{2\pi d}{N} x \right) \)

Frequencies in a function

- Fourier analysis (DCT)
  - 2D bases for 2D signals (images)

How about 3D shapes?

- Problem: general 3D shapes are not (height) functions
  - Height function, regularly sampled above a 2D domain
  - General 3D shapes
Irregular meshes

In graphics, shapes are mostly represented by triangle meshes.

Geometry: Vertex coordinates
- \((x_1, y_1, z_1)
- \((x_2, y_2, z_2)
- \ldots
- \((x_n, y_n, z_n)\)

Connectivity: List of triangles
- \((i_1, j_1, k_1)
- \((i_2, j_2, k_2)
- \ldots
- \((i_m, j_m, k_m)\)

How to define efficient bases?

- Extension of the 2D DCT basis to a general (irregular) mesh

DCT

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
- We will induce the basis functions from the mesh connectivity!

The Mesh Laplacian operator

\[ L(v_i) = d_i v_i - \sum_{j \neq i} v_j = d_i \left( v_i - \frac{1}{d_i} \sum_{j \neq i} v_j \right) \]

- Measures the local smoothness at each mesh vertex

Laplacian operator in matrix form

\[
\begin{pmatrix}
    d_1 & -1 & 0 & \cdots & 0 \\
    0 & d_2 & -1 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \ddots & \vdots \\
    0 & \cdots & -1 & -1 & -1 \\
-1 & \cdots & -1 & -1 & \vdots \\
    -1 & \cdots & -1 & -1 & d_{n+1}
\end{pmatrix}
\begin{pmatrix}
    v_1 \\
    v_2 \\
    \vdots \\
    v_i \\
    v_{n+1}
\end{pmatrix}
= \begin{pmatrix}
    \delta_1 \\
    \delta_2 \\
    \vdots \\
    \delta_i \\
    \delta_{n+1}
\end{pmatrix}
\]

\( L \) matrix
How to represent our mesh geometry in the spectral basis?

Decompose the mesh geometry in the spectral basis:

\[ \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \alpha_1^T \\ \beta_1^T \\ \gamma_1^T \\ \vdots \\ \alpha_n^T \\ \beta_n^T \\ \gamma_n^T \end{bmatrix} = \alpha_1 \mathbf{b}_1 + \beta_1 \mathbf{b}_2 + \cdots + \gamma_n \mathbf{b}_n \]

The first components are low-frequency
The last components are high-frequency

How to represent our mesh geometry in the spectral basis?

\[ \mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathbb{R}^n. \] Decompose in the spectral basis:

\[ \begin{align*}
\mathbf{X} &= \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = a_1 \mathbf{b}_1 + a_2 \mathbf{b}_2 + \cdots + a_n \mathbf{b}_n \\
\mathbf{Y} &= \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} = \beta_1 \mathbf{b}_1 + \beta_2 \mathbf{b}_2 + \cdots + \beta_n \mathbf{b}_n \\
\mathbf{Z} &= \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_n \end{bmatrix} = \gamma_1 \mathbf{b}_1 + \gamma_2 \mathbf{b}_2 + \cdots + \gamma_n \mathbf{b}_n
\end{align*} \]

The spectral basis

First functions are smooth and slow, last oscillate a lot

The spectral basis

Low-frequency basis vectors are smooth and slowly-varying:
- The first basis vector:
  \[ L \mathbf{b}_1 = 0 \]
  \[ \mathbf{b}_1 = (1, 1, \ldots, 1)^T \]
- It is the smoothest possible function (Laplacian is constant zero on it)

The spectral basis

Low-frequency basis vectors are smooth and slowly-varying:
- The first basis vector:
  \[ L \mathbf{b}_1 = 0 \]
  \[ \mathbf{b}_1 = (1, 1, \ldots, 1)^T \]
- \[ L \mathbf{b}_1 = 0 \] because all the rows of \( L \) sum to zero
The spectral basis

- Low-frequency basis vectors are smooth and slowly-varying:
  - The following basis vectors:
    \[ L b_i = \lambda_i b_i \]
    
    - The frequency is small \( \rightarrow \) the Laplacian operator gives small values \( \rightarrow \) the shape of the function is smooth.

- High-frequency basis vectors are non-smooth, oscillating:
  \[ L b_i = \lambda_i b_i \]
  
  - The frequency is large \( \rightarrow \) the Laplacian operator gives large values \( \rightarrow \) the shape of the function is less smooth.

Most shape information is in low-frequency components

Applications

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

Mesh smoothing

- Aim to remove high frequency details

Mesh smoothing

- Drop the high-frequency components

\[
\begin{bmatrix}
\alpha_1 \\
\beta_1 \\
\gamma_1 \\
\alpha_2 \\
\beta_2 \\
\gamma_2 \\
\vdots \\
\alpha_n \\
\beta_n \\
\gamma_n \\
\end{bmatrix} \rightarrow 
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\alpha_3 \\
\vdots \\
\alpha_n \\
\end{bmatrix} \cdot b_1 + 
\begin{bmatrix}
\beta_1 \\
\beta_2 \\
\beta_3 \\
\vdots \\
\beta_n \\
\end{bmatrix} \cdot b_2 + \ldots + 
\begin{bmatrix}
\gamma_1 \\
\gamma_2 \\
\gamma_3 \\
\vdots \\
\gamma_n \\
\end{bmatrix} \cdot b_n
\]

High-frequency components!
Mesh compression

- Aim to represent surface with fewer bits
  - 1.4 bits/vertex vs 36 bits/vertex

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

What happens if quantize xyz coordinates?

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.

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 coefficients 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
Why it works

- Write $x$ as: $x = a_1e_1 + a_2e_2 + \ldots + a_ne_n$

- Therefore, $\delta = Lx = \lambda_1a_1e_1 + \lambda_2a_2e_2 + \ldots + \lambda_na_ne_n$

- Quantization error for $\delta$ ($\delta \rightarrow \delta + q\delta$) is: $q\delta = c_1e_1 + c_2e_2 + \ldots + c_ne_n$

- Resulting error in $x$: $q_L = L^{-1}q\delta = (1/\lambda_1)c_1e_1 + (1/\lambda_2)c_2e_2 + \ldots + (1/\lambda_n)c_ne_n$

Thus, the error in $x$ will contain strong low-frequency components but weak high-frequency components.

Spectral mesh compression

- Low-frequency errors are hard to see.
- High-frequency errors – here $c_i$ are large $(1/\lambda_i)$ is small – attenuates high-frequency errors.
- Small $\lambda_i$ – low frequencies – small $c_i$ $(1/\lambda_i)$ is large – amplifies low-frequency errors.

Progressive transmission

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

Mesh watermarking

- Embed a bitstring in the low-frequency coefficients
- Low-frequency changes are hard to notice.

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)