Spectral bases for 3D shapes representation

Rationale

- Sparse matrix representations.
- Efficient computations.
- Integrating software components.
- Real world application

Towards eigendecomposition

We want to learn how the matrix $A$ works:

$$A = U \Lambda U^T$$

If $A$ is symmetric, the eigenvectors are orthogonal (and there's always an eigenbasis).

Spectra and diagonalization

If $A$ is symmetric, the eigenvectors are orthogonal (and there's always an eigenbasis).

Towards eigendecomposition

If we look at arbitrary vectors, it doesn't tell us much.
PCA – the general idea

- PCA finds an orthogonal basis that best represents given data set.
- PCA finds a best approximating plane (again, in terms of $\sum$ distances$^2$)

The sum of distances$^2$ from the $x'$ axis is minimized.

Application: finding tight bounding box

- Oriented bounding box: we find better axes!

Oriented bounding box: we find better axes!

Application: finding tight bounding box

- This is not the optimal bounding box
Spectral bases (eigendecomposition)

- $L$ is a symmetric $n \times n$ matrix
- Spectral analysis of $L$ ($L = b \lambda b^T$)

$L = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}$

Basis vectors
Frequencies, sorted in ascending order

Usage of bounding boxes (bounding volumes)

- Serve as very simple “approximation” of the object
- Fast collision detection, visibility queries
- Whenever we need to know the dimensions (size) of the object

Principal components

- Eigenvectors that correspond to big eigenvalues are the directions in which the data has strong components (= large variance).
- If the eigenvalues are more or less the same – there is no preferable direction.
- Note: the eigenvalues are always non-negative.

How to use what we got

- For finding oriented bounding box – we simply compute the bounding box with respect to the axes defined by the eigenvectors. The origin is at the mean point $m$.

Scatter matrix

- Denote $y_i = x_i - m$, $i = 1, 2, \ldots, n$
- $S = YY^T$
  where $Y$ is $d \times n$ matrix with $y_i$ as columns ($i = 1, 2, \ldots, n$)
For approximation

- In general dimension $d$, the eigenvalues are sorted in descending order:
  \[ \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \]
- The eigenvectors are sorted accordingly.
- To get an approximation of dimension $d' < d$, we take the $d'$ first eigenvectors and look at the subspace they span ($d' = 1$ is a line, $d' = 2$ is a plane...)

Spectral bases for 3D shapes representation

To get an approximating set, we project the original data points onto the chosen subspace:

\[
x_i = m + \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_d v_d
\]

Projection:

\[
x_i' = m + \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_{d'} v_{d'} + 0 \cdot v_{d'+1} + \ldots + 0 \cdot v_d
\]

Motivation – Image compression

What linear combination of 8x8 basis signals produces an 8x8 block in the image?

Spectral bases

- Fourier analysis
  - 1D sine/cosine bases for signals:
How about 3D shapes?

Problem: general 3D shapes cannot be described as height functions

Irregular meshes

- In graphics, shapes are mostly represented by triangle meshes
  - 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

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
    - \((i_1, j_1, k_1)\)
    - \((i_2, j_2, k_2)\)
    - \(\ldots\)
    - \((i_m, j_m, k_m)\)

Irregular meshes

- In graphics, shapes are mostly represented by triangle meshes

Basis shapes

- 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!
  - We assume that the connectivity is known to the decoder side (it is transferred first)

How to define efficient bases?

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

DCT
The Mesh Laplacian operator

\[ L(v_i) = d_i v_i - \sum_{j \in N(i)} \frac{1}{d_j} v_j \]

- Measures the local smoothness at each mesh vertex.

Spectral bases

- \( L \) is a symmetric \( n \times n \) matrix.
- Perform spectral analysis of \( L \).

\[ L = \begin{bmatrix} h & b & \cdots & b \end{bmatrix} \begin{bmatrix} h & b & \cdots & b \end{bmatrix}^T \]

Basis vectors

Frequencies, sorted in ascending order

Laplacian matrix

- \( L \) is constructed based on the connectivity alone.
- So, if the decoder got the connectivity first, it can construct \( L \).

How to represent our mesh geometry in the spectral basis?

- Decompose the mesh geometry in the spectral basis:

\[ \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \\ \vdots \end{bmatrix} b_1 + \begin{bmatrix} \alpha_2 \\ \beta_2 \\ \gamma_2 \\ \vdots \end{bmatrix} b_2 + \cdots + \begin{bmatrix} \alpha_n \\ \beta_n \\ \gamma_n \\ \vdots \end{bmatrix} b_n \]

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

How to represent our mesh geometry in the spectral basis?

- \( X, Y, Z \in \mathbb{R}^n \). Decompose in the spectral basis:

\[ X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \]

\[ Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = b_1 + b_2 + \cdots + b_n \]

\[ Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \gamma_1 b_1 + \gamma_2 b_2 + \cdots + \gamma_n b_n \]
Why this works?

- Low-frequency basis vectors are smooth and slowly-varying:
  - The first basis vector:
    \[ LB_1 = \lambda_1 b_1 = 0 \]
    \[ b_1 = (1, 1, \ldots, 1)^T \]
  - It is the smoothest possible function (Laplacian is constant zero on it)

Why this works?

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

Geometry compression

\[
\begin{bmatrix}
\alpha_1 \\
\beta_1 \\
\gamma_1 \\
\vdots \\
\alpha_n \\
\beta_n \\
\gamma_n
\end{bmatrix}
\begin{bmatrix}
\beta_1 \\
\gamma_1 \\
\vdots \\
\beta_n \\
\gamma_n
\end{bmatrix}
\begin{bmatrix}
b_1 \\
b_2 \\
\vdots \\
b_n
\end{bmatrix}
= \lambda_1
\]

As in JPEG compression, drop the high-frequency components!

Why this works?

- High-frequency basis vectors are non-smooth, oscillating:
  \[ LB_i = \lambda_i b_i \]
  The frequency is large → the Laplacian operator gives large values → the shape of the function is less smooth!

The spectral basis

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

\[ L = \begin{pmatrix}
\text{chain connectivity} \\
\text{horse connectivity}
\end{pmatrix} \]

\[ \text{spectral basis of } L = \begin{pmatrix}
\text{20th basis function} \\
\text{10th basis function} \\
\text{100th basis function}
\end{pmatrix} \]
Geometry compression

The less coefficients are dropped, the less shape information is lost.

Technical algorithm

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

Challenges and trade-offs

As in JPEG, blocking artifacts appear for low bit rates, because the compression is lossy (too much information is lost).

Solution: like in JPEG, work on small blocks:
- 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!

Further info


Good Luck