Intro to Digital Geometry Processing

Richard (Hao) Zhang

CMPT 464/764: Geometric Modeling in Computer Graphics
Lecture 12
First example: de-noising an image

How to remove noise from image on the left?
First solution: smoothing via convolution

- Use a box, hat, or Gaussian filter in the frequency domain to perform **low-pass filtering (multiplication)**

- Equivalently, a **convolution** or **local averaging** in the spatial domain
  - box ↔ convolve with sinc:
    \[
    \text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}
    \]
  - hat ↔ sinc²
  - Gaussian ↔ Gaussian

- Easily implemented in hardware
1D continuous convolution

- An integral that computes a “running weighted average”

\[(f \otimes g)(t) = \int f(s)g(t - s)ds\]

- Kernel/weighting function \(g\) is often symmetric about 0
Example: B-splines via convolution

\[ B_0(t) \]

\[ B_1(t) \]

\[ B_2(t) \]

\[ \cdots \]
Convolution over images

- Running average using a **convolution** or **kernel matrix**

- Often leads to **reduced image resolution**

- Typically the **first layer of a convolutional neural network (CNN)**

- Parameters to specify a convolution:
  - Kernel size, weights, **strides** (= how fast kernel shifts)
Strides

- 3 x 3 kernel size with stride = 1

- 3 x 3 kernel size with stride = 2
Effect of convolution

- Convolution can perform blurring, sharpening, embossing, etc.
Effect of convolution

- Convolution can perform blurring, sharpening, embossing, etc.
- Convolution layer produces an **activation/feature map** in CNN
- Convolution kernel gathers filter response
  - When a feature is detected, the response is **amplified/activated**
Spatial vs. frequency: Fourier transform

- Spatial domain is a domain of spatial positions – think of a rectangular grid with grey scale intensities: an image

- The **frequency domain** is a domain of frequency values

- **Fourier transform** (FT): a transformation taking a function from spatial domain to frequency domain

- The **inverse Fourier transform** takes a function from frequency domain back to the spatial domain
Continuous Fourier transform

- Let $f(x)$ be a continuous and integrable function.
- The Fourier transform (FT) of $f$

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi xu} \, dx = \int_{-\infty}^{\infty} f(x)[\cos 2\pi xu - i \sin 2\pi xu] \, dx$$

- The corresponding inverse FT

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{i2\pi xu} \, du$$
Discrete Fourier transform (DFT)

- Let $s(k)$ be a complex-valued signal defined at samples 0, 1, ..., $N - 1$
- The discrete Fourier transform (DFT) of $s$:

$$a(u) = \frac{1}{N} \sum_{k=0}^{N-1} s(k) e^{-i \frac{2\pi u k}{N}} = X(u) + iY(u), u = 0..N - 1$$

- The corresponding inverse DFT

$$s(k) = \sum_{u=0}^{N-1} a(u) e^{i \frac{2\pi u k}{N}}, k = 0..N - 1$$
If $f(x)$ is real then $|F(u)|$ would be symmetric.
Convolution theorem

- Convolution (spatial averaging) in the spatial domain corresponds to multiplication in the frequency domain via the Fourier transform.

<table>
<thead>
<tr>
<th>Spatial filter</th>
<th>Fourier transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>box</td>
<td>sinc</td>
</tr>
<tr>
<td>hat</td>
<td>sinc²</td>
</tr>
<tr>
<td>Gaussian</td>
<td>Gaussian</td>
</tr>
</tbody>
</table>

- All of these filters attenuate high frequencies in the spectral domain.
- They also blur the images, as sharp features are “high-frequency”
Convolution filters (local averaging) are linear

Median filter (special case of rank filters) works best with “spiky” noise

Bilateral filter works to preserve edge features (later)

Both filter are non-linear and attempt to reduce influence of outliers

Nonlinear filters are hard to realize in hardware
Convolution or smoothing of geometry?

- Signal in question: meshes modeling 3D objects
- **Local averaging** is easy, but let us develop these ideas gradually
Geometry representation

- Focus on surfaces, not volume
- Triangle meshes
  - Set of triangles pasted along their edges
  - Piece-wise linear approximations
- Irregular connectivity
  - Regularity require genus 1
  - Semi-regularity needs remeshing
  - Directly available and flexible
Digital geometry processing (DGP)

- Signal processing for (mesh) geometry [Praun 01]
- Work on **discrete** representation of geometry directly
  
  e.g., Compute vertex normals by area-weighted face normals instead of fitting a polynomial patch
- Many DGP methods still motivated by continuous counterparts
- A very active areas of research in computer graphics today
  
  - Subdivision, decimation, segmentation, correspondence, etc.
Why DGP?

- Many geometry problems on meshes are well-studied problems in signal/image processing
  - Smoothing and feature extraction and analysis, e.g., [Taubin 95 & 96, Kobbelt et al. 98, Desbrun et al. 99, Roessl et al. 00, Peng et al. 01, Sun et al. 02, Alexa 02, Zhang & Fiume 02 & 03, Fleishman et al. 03, Jones et al. 03]
  - Segmentation, e.g., [Mangan & Whitaker 99, Katz et al. 03, Page et al. 03, Lee et al. 04, Liu & Zhang 04 & 05, Clements & Zhang 06]
  - Geometry compression, e.g., [Khodakovsky et al. 00, Karni & Gotsman 00 & 01, Sorkine et al. 03, Ben-Chen & Gotsman 03, Zhang & Blok 04]
  - Matching and similarity of 3D shapes, e.g., [Osada et al. 01, Zaharia & Preteux 01, Kazhdan & Funkhouser 03 & 04, Jain & Zhang 05 & 06]
  - LOD analysis, e.g., [Lee et al. 98, Guskov et al. 99], etc.
DGP problems

- Some of the important DGP problems include
  - Smoothing (denoising), restoration, enhancement
  - Edge, in general, feature, detection
  - Segmentation
  - Parameterization
  - Quantization and compression
  - Simplification and multiresolution modeling
  - Sampling and reconstruction issues
  - Correspondence, matching, and recognition
Mesh signal via parameterization

Given a mesh model, how should we define the signal?

- Convert to functions we know how to handle
  - Partition mesh into patches
  - Define each patch as a **height field over a planar domain**
Mesh signal via parameterization

- **Geometry image** [Gu, Gortler & Hoppe 02]
  - Cut, flatten, and map surface into a square
  - \((x, y, z)\) coordinates become RGB values
- For genus-0 shapes, construct **spherical function**

[Gu, Gortler & Hoppe 02]  
[Gostman et al. 03]
Pros and cons of mesh parameterization

- **Pros**
  - Reduce operations on geometry to those on familiar functions
  - More or less overlooks mesh connectivity

- **Cons**
  - Cuts and patch boundaries are artificially introduced
  - Parameterization is generally a hard problem
Mesh geometry as signal over surface

- Define 3D mesh signal over the mesh vertices

\[ M = (A, \mathbf{x}) \]

adjacency matrix \text{coordinate vector}

\[ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \text{ where } x_i \in \mathbb{R}^3 \text{ is the coordinate of the } i\text{-th vertex} \]

- Signal is really defined on (the vertices of) a graph
- Other types of signal possible, e.g., normal, color, displacement
Mesh signal processing

- Many concepts and techniques from image processing to “copy”
  - E.g., smoothing/blurring via local averaging
- DGP originally motivated by smoothing large meshes [Taubin 95]
Mesh smoothing

- Post-processing step for surface reconstruction out of noisy data,
- Frequently used in geometry coding (wavelets) and design
- For very large models, the traditional CAGD approach, i.e., optimization with respect to \textbf{total curvature}, is often too expensive

\[ \int_S \kappa_1^2 + \kappa_2^2 \]
A signal processing approach

- Mesh as a 3D signal defined over the vertices of mesh graph
- Smoothing/denoising via **low-pass filtering** of the mesh signal
- Edge lengths and face angles are ignored for now
- Results in fast smoothing with only local updates
Example: Laplacian smoothing

- Local averaging like a hat (Gaussian over higher-order neighbors) – move vertex towards neighbors’ centroid – **Laplacian smoothing**

- Smoothing operator: \( S = (1 - k)I + kC \)

- Uniform Tutte Laplacian operator (\( TL \)) \( T = I - C \)

- What is the **frequency-domain** interpretation?

\[
C_{ij} = \begin{cases} 
1/d_i, & \text{if } (i, j) \text{ is an edge and } d_i \text{ is the valence of } i \\
0, & \text{otherwise}
\end{cases}
\]
DFT for mesh signals

- How to mimic classical Fourier transform (FT) on mesh signals
- It is NOT the classical 2D Discrete FT (DFT)
- Key fact: Basis functions of 1D DFT are precisely the *eigenvectors* of the 1D discrete Laplacian
- So let us find a Laplacian operator for our mesh signal and perform an *eigenvalue decomposition* [Taubin 95]
- The uniform Tutte Laplacian $T$ is an appropriate choice
Eigenvector plots (the 1D case)

Figure 6: Plots of first 8 eigenvectors of the 1D discrete Laplacian operator ($n = 401$)

Not the DFT basis, but the real ones; oscillatory behaviours similar
Eigen-stuff, just in case (aside)

- For a square matrix $A$, if $Ax = \lambda x$ with $x \neq 0$, then $x$ is an eigenvector of $A$ with the corresponding eigenvalue or characteristic value $\lambda$.

- The quadratic form: $x^T Ax = \lambda ||x||^2$

- If $A$ is a real matrix, then it is symmetric if and only if its eigenvectors can be made orthonormal.

- Two vectors $u$ and $v$ are orthogonal if $u^T v = 0$.

- If $E$ is a matrix whose columns are the orthonormal eigenvector of $A$, then $E^T E = I$ and $AE = \Lambda E$. So

$$A = E \Lambda E^T \text{ or } E^T AE = \Lambda$$

The latter is referred to as the diagonalization of $A$. 
Spectral characterization of meshes

- **Eigenvalue decomposition** of a mesh signal $x$ with respect to $T$:

$$x = e_1X_1 + e_2X_2 + \ldots + e_nX_n = EX$$

where $e_1, e_2, \ldots, e_n$ are the eigenvectors of $T$. Call $X$ the **ED-transform** of $x$ with respect to $T$.

- Eigenvectors of the $T$ appear to represent the **vibration modes** of the mesh graph, while eigenvalues represent **frequencies** [Taubin 95]
Plot of “high-frequency” eigenvectors
How about convolution?

Linear mesh filtering:

\[ f(T)x = \sum_{j=1}^{n} f(T)e_jX_j = \sum_{j=1}^{n} e_j f(\lambda_j)X_j \]

where \( f \) is a polynomial or rational polynomial, \( \lambda \)'s are eigenvalues.

- One can design low (e.g., for smoothing), high, band-pass filters.
  - But computing eigenvectors is prohibitively expensive – \( O(n^3) \)
  - So use **polynomial or rational polynomial filters**
    - Polynomial filtering \( \rightarrow \) matrix-vector multiplications
    - Rational polynomial filters \( \rightarrow \) solution of linear system
Low-pass filtering and Laplacian smoothing

The Laplacian smoothing filter \((1 - k\lambda)^N\) attenuates all but the zero frequency. It causes shrinkage and distortion.

A more desirable filter shape is shown.
Laplacian smoothing (i.e., the “shrinking” phase) is followed by an “unshrinking” operation – filtering by \((I + \mu U)\) – at each step of the procedure [Taubin 95]

- **Pass-band frequency**: \(1/\lambda - 1/\mu\)
- First-order derivative at 0 is non-zero
- **Amplification** within the pass-band – can lead to large (unnatural) ripples
- Slow smoothing due to really gentle “roll-off”
Effect of low-frequency amplification

[Zhang & Fiume 02]
Butterworth filtering

- Laplacian smoothing attenuates all but the zero frequency
- Poly-filters need high degrees to get relatively sharp “roll-off ”
- So use Butterworth filter with a moderate order [Zhang & Fiume 03]

\[
 f_{\text{BUT}}(\delta) = \frac{1}{1 + (\delta / \hat{g})^{2N}} = \frac{1}{1 + (\lambda \delta)^{2N}}
\]

- Need to **solve large linear system**
Butterworth filters (aside)

- Frequently used in DSP
- Pass-band frequency: \( \hat{\gamma} = \frac{1}{\lambda} \)
- First-order derivative at 0 is 0 as long as \( N \geq 1 \) – no pass-band amplification
- In fact, Butterworth filters are maximally flat in the pass band, that is, the first \( 2N - 1 \) derivatives are 0 at 0
- Monotone behavior in both pass and stop bands
- Sharpness of “roll-off ” increases as \( N \) increase
A numerical problem (aside)

- Need to solve the linear system \( [I + (\lambda U)^2]^N x = b \)
- Choosing the appropriate iterative solver is crucial
- For \( N = \frac{1}{2} \), we have **implicit fairing** [Desbrun et al. 99]
  - **successive overrelaxation** (SOR) appears to work the best
  - Beats Laplacian smoothing in smoothing speed for aggressive smoothing
- For larger \( N \) (i.e., better shape preservation), factorize the system in the complex domain and use biconjugate gradient stabilized method
Results

“Ringing”: rippling phenomenon caused by convolving with the sinc
Feature-preserving mesh smoothing

- Feature preservation without explicit feature detection, e.g., as in [Clarenz et al. 00]

[Clarenz et al. 00] — A geometric (local) optimization approach
Bilateral mesh smoothing

- Goal: denoising while still preserving features
- Aim for efficiency compared with (local) geometric optimization
- Inspired by bilateral filtering from image processing
Bilateral image smoothing

Pixel updates at each step [Smith & Brady 97]:

\[
p' = \frac{1}{k_p} \sum_{q \in \text{neighborhood of } p} \frac{q \cdot f(\text{distance}) \cdot g(\text{similarity})}{k_p}
\]

where

- \( k_p = \sum_{q \in \text{neighborhood of } p} f(\text{distance}) \cdot g(\text{similarity}) \), \( f \) and \( g \) are both Gaussians

- \( \text{distance} = \) spatial distance between two pixels
- \( \text{similarity} = \) intensity difference between two pixels

- Contribution of a neighboring pixel \( q \) to new value of pixel \( p \) depends also on the difference between pixel values – e.g., discount influence of outliers
Bilateral mesh filtering as robust estimation

- How to measure difference in “intensities” at nearby mesh vertices?
- **New vertex location determined by nearby triangles**
  - Triangles form the surface – they are first-order approximation of the surface (i.e., tangent planes)
  - Triangles do not have to form a manifold, but need to know which triangles are near a particular vertex
- “Intensity difference” is distance between vertex and a robust predictor given by a nearby triangle
- Spatial distance between vertex and triangle is Euclidean distance between vertex and triangle centroid
- Triangle area is also weighed in

[Jones et al. 03]
Robust vertex prediction

\[ p' = \frac{1}{k_p} \sum_{\text{triangle } q \text{ nearby}} \Pi_q(p) A_q f(\|c_q - p\|) g(\|\Pi_q(p) - p\|) \]

- \( A_q \): area of triangle \( q \)
- \( \Pi_q(p) \): prediction for \( p \) given by triangle \( q \)

Prediction given by projection to tangent plane

Noise in normal is problematic

Normals need to be smoothed beforehand
A summary on mesh smoothing

- The prototypical digital geometry processing problem
- Lots of past and current work, but is still on-going, e.g.,
  - Geometric Laplacian to reduce tangential shifts [Desbrun et al. 99]
  - Mean and median filtering [Yagou et al. 02]
  - Bilateral filtering applied to smoothing of point sets [Velho 04]
- Strive for efficiency and quality, esp. in feature preservation
- Tough questions: when is your smoothing result good enough?
- Latest works on data-driven and deep learning based mesh denoising
  - Learn how to denoise from pairs of noisy and ground-truth models
Data-driven mesh denoising [2017]

Mesh Denoising via Cascaded Normal Regression

Peng-Shuai Wang 1
Tsinghua University

Yang Liu 2
Microsoft Research Asia

Xin Tong 2

ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA 2016)
Spectral processing

- Linear mesh filtering so far avoids computing the spectrum explicitly
- Direct access or manipulation of the spectral coefficients
  - Spectral coefficients as shape descriptors, e.g., for shape similarity [Rui et al. 98, Zhang & Fiume 02, Jain & Zhang 06 & 07]
  - Embedding information into spectral coefficients for mesh watermarking [Ohbuchi et al. 01 & 02]
  - JPEG-like geometry compression [Karni & Gotsman 00]
Fourier descriptors (in continuous 2D)

- Defined for **closed (periodic) 2D contours** \((x(p), y(p))\)
- **Continuous formulation** in \(\mathbb{C}\): \(z(p) = x(p) + i y(p)\)
- Expand \(z(p)\) in Fourier series, where \(P\) is the perimeter of the contour

\[
\hat{z}_v = \frac{1}{P} \int_0^P z(p) \exp \left(\frac{-2\pi i v p}{P}\right) \, dp \quad v \in \mathbb{Z}.
\]

- Reconstruction from the Fourier coefficients

\[
z(p) = \sum_{v=-\infty}^{\infty} \hat{z}_v \exp \left(\frac{2\pi i v p}{P}\right). \quad p \in [0, P]
\]
Interpretation of 2D Fourier descriptors

- Coefficients $\hat{z}_v$ for integer $v$'s are the **Fourier Descriptors** (FDs).
- The first FD $\hat{z}_0$ is the **centroid** of the boundary.
- The FD $\hat{z}_1$ describes a circle traced counterclockwise $\hat{z}_1 \exp\left(\frac{i2\pi p}{P}\right)$.
- The FD $\hat{z}_{-1}$ describes a circle traced clockwise $\hat{z}_{-1} \exp\left(-\frac{i2\pi p}{P}\right)$.
- Reconstruction using these three FDs adds the two circles to the centroid, resulting in an ellipse!

$$z_{\{-1,0,1\}}(p) = \hat{z}_0 + \hat{z}_1 \exp\left(\frac{i2\pi p}{P}\right) + \hat{z}_{-1} \exp\left(-\frac{i2\pi p}{P}\right)$$
Continuous 2D FDs, continued

- Each higher-order pair \((\hat{\theta}_d, \hat{\theta}_{-d})\) also characterize two oppositely traced circles, combining into an ellipse.
- However, these circles go around \(d\) times over.
- When incorporated into reconstruction, they add higher frequency details.
- Fourier descriptors can thus be seen as robust against noise.
2D FD reconstruction

Reconstruction of an “L” and a “T” with 2, 3, 4, and 8 FDs
Extensions to 2D and 3D

- Discrete Fourier descriptors for shape recognition
  - 2D closed contours (e.g., for handwritten characters), also to handle symmetry [Zahn & Roskies 72]
  - Normalized FDs for 3D closed contours [Zhang & Fiume 02]
- Generalization to 3D meshes
  - Start with closed 2-manifold meshes
  - Eigenvalue decomposition with respect to a discrete Laplacian
3D mesh FD reconstruction of an “L”

Original (362 vertices) and then using first 5, 10, 13, 15, 20, 30 spectral coefficients for reconstruction
3D mesh FD reconstruction of a snake

Original (530 vertices) and then using first 5, 10, 20, 30, 50 spectral coefficients for reconstruction
The operator to use generalizes the Laplacian or adjacency matrix

Generalize adjacencies to encode **pair-wise distances or affinities**

- Affinity matrix is typically obtained using a Gaussian
- Affinities encode the “context” of each entity, e.g., mesh vertex or face, in its relationship with the others

Spectral $k$-d embedding from $k$ leading eigenvectors

Use of Laplacian matrix $L = D - A$ is also possible

Example:

- **$k$-means clustering in the spectral domain**
- Distance is Euclidean and using a Gaussian
Spectral clustering

Encode information about pair-wise **point affinities**

Input data

Operator $A$

Spectral embedding

Leading eigenvectors

$$A_{ij} = e^{-\frac{|p_i - p_j|^2}{2\sigma^2}}$$
Spectral clustering continued

Key app of (spectral) clustering: shape segmentation