Επεξεργασία Σημάτων σε Γράφους
Θεωρία και Εφαρμογές
Outline

- The Emerging field of Signal Processing on graphs
  - Modeling Signals on Graphs
  - GFT and the notion of Frequency
  - Signal Smoothness with respect to the structure of the Graph
- Applications
  - Cooperative localization and tracking in CAVs
  - Low level processing of Static and Dynamic Meshes
Types of signals on graphs

Social networks

Environmental monitoring

Electrical networks
Modeling signals on graphs

Edge connectivity/weight <-> similarity between vertices.

**Known connectivity**
- Social media
- Sensor networks
- Traffic networks
- 3D point clouds

**Unknown connectivity**
- Neuronal networks
- Internet/WWW
- Gene regulatory networks

The height of each blue bar represents the signal value at the vertex.
Analyze signals on graphs

- Epidemiological data describing the spread of disease
- Census data describing human migration patterns
- Logistics data describing inventories of trade goods
- Anatomical connectivity of distinct functional regions of the cerebral cortex
- Cluster different genes based on their phenotype/participation in metabolism
- Classify human activity from depth sensors
In which cases is it useful?

- **Common data processing tasks:**
  - Filtering
  - Denoising
  - Inpainting
  - Compression

- **Challenges**
  - What is translation over a graph?
  - What is downsampling over a graph?
Types of graphs

- Erdos-Renyi graphs,
- ring graphs,
- Random geometric graphs,
- small-world graphs,
- power-law graphs,
- nearest-neighbor graphs,
- scale-free graphs
Regular Graph Structures

1D Time-series
- Nodes <-> time instances
- Edges are unweighted and directed

2D images
- Nodes <-> pixel
- Edges <-> similarity
Signals on Graphs

Graphs: generic data representation forms encoding the geometric structures of data

Applications: social networks, energy distribution networks, transportation network, wireless sensor network, and neuronal networks.

$G = \{V, E, W\}$

weights: distance/similarity/relationship

$W_{i,j} = \begin{cases} \exp\left(-\frac{[\text{dist}(i,j)]^2}{2\theta^2}\right) & \text{if dist}(i,j) \leq \kappa \\ 0 & \text{otherwise} \end{cases}$

Assumptions
1. Undirected graphs without self loops.
2. Scalar sample values
Modeling the graph

The adjacency matrix is a matrix, $A$, such that $A_{ij} = w_{ij}$. If the graph is undirected, $w_{ij} = w_{ji}$, and $A$ is symmetric.

The degree matrix of $G$ is a diagonal matrix, $D$, with entries $(D)_{ii} = \sum_{j=1}^{N} A_{ij}$ and $(D)_{ij} = 0$ for $i \neq j$.

The combinatorial graph Laplacian defined as $L = D - A$, and the symmetric normalized Laplacian $\mathcal{L} = D^{-1/2}LD^{-1/2}$. 
Graph signal \( f \) in \( \mathbb{R}^N \), where \( |V|=N \)

Graph Laplacian \( \mathcal{L} := D - W \), \( D \): diagonal with sums of weights
\( W \): weight matrix

Normalized Graph Laplacian \( \mathcal{L} = D^{-1/2} L D^{-1/2} \)
The Z Transform

Consider $N$ samples of a signal $s_n$, $n = 0, 1, \cdots, N - 1$ of finite number $N$ of samples and to filters with finite impulse response (FIR filters).

The $z$-transform $s(z)$ of the time signal $s = \{s_n : n = 0, 1, \cdots, N - 1\}$ organizes its samples $s_n$ into an ordered set of time samples, where sample $s_n$ at time $n$ precedes $s_{n+1}$ at time $n + 1$ and succeeds $s_{n-1}$ at time $n - 1$.

In other words, the signal is given by the $N$-tuple $s = (s_0, s_1, \cdots, s_{N-1})$.

This representation is achieved by using a formal variable $z^{-1}$ called the shift (or delay), so that the signal of $N$-samples is represented by

$$s(z) = \sum_{n=0}^{N-1} s_n z^{-n}.$$
The DFT transform

The discrete Fourier transform (DFT) of the signal $s$ is $\hat{s} = \{ \hat{s}_k : k = 0, \cdots, N - 1 \}$ given by

$$\hat{s}_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} s_n e^{-j \frac{2\pi}{N} kn}.$$ 

The $\hat{s}_k$ are the Fourier coefficients of the signal.

The discrete frequencies are $\Omega_k = \frac{2\pi k}{N}, k = 0, 1, \cdots, N - 1$, and the $N$ signals $(x_k[n])$

$$\left\{ x_k[n] = \frac{1}{\sqrt{N}} e^{-j \frac{2\pi}{N} kn} : n = 0, 1, \cdots, N - 1 \right\}_{k=0}^{N-1}$$

are the spectral components.

The signal is recovered from its Fourier coefficients by the inverse DFT:

$$s_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{s}_k e^{j \frac{2\pi}{N} kn}, s = 0, 1, \cdots, N - 1.$$
An FIR filter is also represented by a polynomial in $z^{-1}$

$$h(z) = \sum_{n=0}^{N-1} h_n z^{-n},$$

so that the output $s_{out}$ of filter $h$ applied to signal $s_{in}$ is:

$$s_{out}(z) = h(z) \cdot s_{in}(z).$$

Defining the shift or delay filter $h_{\text{shift}}(z) = z^{-1}$, and applying it to a signal $s_{in} = (s_0, s_1, \cdots, s_{N-1})$ produces:

$$s_{out} = h_{\text{shift}} \cdot s_{in} = (s_{N-1}, s_0, s_1, \cdots, s_{N-2}).$$
Shift Invariance

The series combination of filters is commutative, a filter commutes with the shift filter—delaying the input signal and then filtering the delayed input signal leads to the same signal as first filtering the input signal $s_{in}$ and then delaying the filtered output.

$$z^{-1} \cdot h(z) = h(z) \cdot z^{-1}.$$  

Writing the signal $s = (s_0, s_1, \cdots, s_{N-1})$ as the vector

$$s = [s_0 \ s_1 \ \cdots \ s_{N-1}]^\top \in \mathbb{C}^N,$$

and a filter $h$ as a matrix $H$, filtering can be written as

$$s_{out} = H \cdot s_{in}$$
Shift Filtering Operation

The shift filtering operation corresponds to multiplication by a circulant matrix $A_c$

$$[s_{N-1} \ s_0 \ \cdots \ s_{N-2}]^\top = A_c \cdot [s_0 \ s_1 \ \cdots \ s_{N-1}]^\top,$$

given by the cyclic shift

$$A_c = \begin{bmatrix}
0 & 0 & 0 & \cdots & 0 & 1 \\
1 & 0 & 0 & \cdots & 0 & 0 \\
0 & 1 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 \\
0 & 0 & \cdots & 0 & 1 & 0 \\
\end{bmatrix}$$
The 0-1 shift matrix $A_c$ as the adjacency matrix of a graph. Labeling the rows and columns of $A_c$ from 0 to $N - 1$, define the graph $G_c = (V, E)$ with node set $V = \{0, 1, \cdots, N - 1\}$. 
Shift Invariance

A filter represented by $H$ will be shift invariant if it commutes with the shift: $AH = HA$

In GSP, filters are defined as matrices and the eigensignals of $h$ are the eigenvectors of the corresponding $H$.

Under certain conditions, every filter commuting with $A$ is a polynomial in $A$, i.e. $H = h(A)$

Then $H = h(A) = Vh(\Lambda)V^{-1}$, where $A = V\Lambda V^{-1}$
Graph Fourier Transform

The graph Fourier transform of graph signal $\mathbf{s}$ is given by the graph Fourier analysis decomposition

$$
\hat{\mathbf{s}} = \mathbf{F}s = \mathbf{V}^{-1}\mathbf{s} = [f_0\mathbf{s} \cdots f_{N-1}\mathbf{s}]^\top
$$

The graph Fourier coefficients or graph spectral coefficients of signal $\mathbf{s}$ are computed using the inner product as

$$
\hat{s}(\lambda_k) = \hat{s}_k = f_k\mathbf{s} = \langle f_k^H, \mathbf{s} \rangle.
$$

The inverse GFT is given by

$$
\mathbf{s} = \mathbf{F}^{-1}\hat{\mathbf{s}} = \mathbf{V}\hat{\mathbf{s}}
$$
Filtering in graph frequency domain

Given the adjacency matrix \( A = \mathbf{V} \Lambda \mathbf{V}^{-1} \)

The graph filter can be expressed as

\[
H = h(A) \\
= h\left( \mathbf{V} \Lambda \mathbf{V}^{-1} \right) \\
= \sum_{m=0}^{M-1} h_m \left( \mathbf{V} \Lambda \mathbf{V}^{-1} \right)^m \\
= \mathbf{V} h(\Lambda) \mathbf{V}^{-1},
\]

where \( h(\Lambda) \) is the diagonal matrix

\[
h(\Lambda) = \text{diag} \left[ h(\lambda_0) \cdots h(\lambda_{N-1}) \right].
\]
The output of $s_{in}$ to filter $h$ is successively

$$s_{out} = H \cdot s_{in}$$

$$= V h (\Lambda) \left( V^{-1} s_{in} \right)$$

**Fourier transf.**

$$= V \text{ diag} \left[ h (\lambda_0) \cdots h (\lambda_{N-1}) \right] \hat{s}_{in}$$

**Filtering in graph Fourier space**

$$= V \left[ h (\lambda_0) \hat{s}_{in_0} \cdots h (\lambda_{N-1}) \hat{s}_{in_{N-1}} \right]^T$$

**Inverse Fourier transf.**
Using the Graph Laplacian

A frequency representation can be similarly built on top of the Laplacian matrix of an undirected graph.

Since this matrix is positive semidefinite, all the eigenvalues are real and non-negative, and a full set of orthogonal eigenvectors can be obtained, so that we can write

\[ L = U \Lambda U^T \]

with \( U \) the GFT matrix, which is real and orthogonal.

Because the eigenvalues are real, they provide a natural way to order the GFT basis vectors in terms of frequency.
GFT cont’d

• Graph Fourier Transform

\[ L = \mathbf{U} \Lambda \mathbf{U}^T = \sum_{i=1}^{N} \lambda_i \mathbf{u}_i \mathbf{u}_i^T, \]

• GFT: projection onto the eigenvectors of the graph Laplacian

\[ \hat{\mathbf{x}} = \mathbf{U}^T \mathbf{x} \]

• Inverse GFT:

\[ \mathbf{x} = \mathbf{U} \hat{\mathbf{x}} \]

• The graph Laplacian eigenvectors associated with low frequencies vary slowly across the graph

• The eigenvectors associated with larger eigenvalues oscillate more rapidly
Graph Laplacian

Spectral properties \[ \mathcal{L} u_\ell = \lambda_\ell u_\ell, \]

- Laplacian is Positive Semi-definite matrix
- Eigenvalues: \( 0 = \lambda_1(L) \leq \lambda_2(L) \leq \ldots \leq \lambda_{N-1}(L) \)
- Eigen-pair system \( \{\lambda_k, u_k\} \) provides Fourier-like interpretation (GFT)
Eigenvectors of Graph Laplacian
Important observations

For connected graphs, the Laplacian eigenvector $u_0$ associated with the eigenvalue 0 is constant and equal to $\frac{1}{\sqrt{N}}$ at each vertex.

The graph Laplacian eigenvectors associated with low frequencies vary slowly across the graph.

If two vertices are connected by an edge with a large weight, the values of the eigenvector at those locations are similar. The eigenvectors associated with larger eigenvalues oscillate more rapidly and are more likely to have dissimilar values on vertices connected by an edge with high weight.
Fig. 2. Three graph Laplacian eigenvectors of a random sensor network graph. The signals’ component values are represented by the blue (positive) and black (negative) bars coming out of the vertices. Note that $u_{50}$ contains many more zero crossings than the constant eigenvector $u_0$ and the smooth *Fiedler vector* $u_1$. 
Estimating the underlying graph

- One signal <-> many different graphs
- Only 1 leads to a smooth graph signal.
  - Only $G_1$ favors smoothness of the resulting graph signal.
Graph Smoothness

Graph based approximation

\[
\hat{f}(\lambda_0) := \langle f, u_\lambda \rangle = \sum_{i=1}^{N} f(i) u_\lambda^*(i).
\]

Smoothness w.r.t. graph

\[
\| f \|_L := \| \mathcal{L}^{\frac{1}{2}} f \|_2 = \sqrt{f^T \mathcal{L} f} = \sqrt{S_2(f)}.
\]

Graph spectral filtering (regularization)

\[
\min_{f} \left\{ \| f - y \|_2^2 + \gamma S_p(f) \right\},
\]

\[
\argmin_{f} \left\{ \| f - y \|_2^2 + \gamma f^T \mathcal{L} f \right\}.
\]

Connectivity of the graph -> encoded in graph Laplacian

Define both a graph Fourier transform (graph Laplacian eigenvectors)

Different notions of smoothness
Graph Smoothness Examples

\[ f^T \mathcal{L}_1 f = 0.14, \quad f^T \mathcal{L}_2 f = 1.31, \quad f^T \mathcal{L}_3 f = 1.81. \]
Cooperative Localization and tracking in connected and automated vehicles

Preliminaries

Centralized Laplacian Localization

• Acquire the Laplacian matrix of V2V cluster graph $L(t)$.

• Differential coordinates per vehicle: $\delta_i^{(t,x)} = \frac{1}{|N_i^{(t)}|-1} \sum_{j \in N_i^{(t)}} \left(-\hat{z}_i^{(t)} \sin \hat{z}_i^{(t)} \right)$ and $\delta_i^{(t,y)} = \frac{1}{|N_i^{(t)}|-1} \sum_{j \in N_i^{(t)}} \left(-\hat{z}_d^{(t)} \cos \hat{z}_d^{(t)} \right)$.

• Extended Laplacian matrix $\tilde{L}(t) = \begin{bmatrix} L(t) & I_{|e_i^{(t)}|} \end{bmatrix}^T$ and vector $b^{(t,x)} = \begin{bmatrix} D(t) \delta^{(t,x)} \hat{z}_p^{(t,x)} \end{bmatrix}^T$, with identity matrix $I_{|e_i^{(t)}|}$, degree matrix $D(t)$, $x$-differential vector $\delta^{(t,x)}$ and $x$-GPS measurements vector $\hat{z}_p^{(t,x)}$ of cluster.

• Estimate $x$-positions vector $x^{(t)}$ in a centralized manner: $\tilde{L}(t) x^{(t)} = b^{(t,x)}$
Distributed Laplacian Localization

- Red \((i = 1)\), yellow \((i = 2)\), green \((i = 3)\).

- Acquire the local Laplacian matrix \(\tilde{L}_1(t)\) and vector \(b_1^{(t,x)}\) for the neighborhood of \(i = 1 \to \text{star topology}\).

\[
\tilde{L}_1(t) = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
-1 & -1 & 2 \\
0 & 0 & 1
\end{bmatrix},
\]

\[
b_1^{(t,x)} = \begin{bmatrix}
\tilde{z}_{p,2}^{(t,x)} \\
\tilde{z}_{p,3}^{(t,x)} \\
(N_1(t) - 1) \delta_1^{(t,x)} \\
\tilde{z}_{p,1}^{(t,x)}
\end{bmatrix}
\]

- Estimate \(x\)-positions vector \(x_1^{(t)}\) of neighborhood (last element corresponds to \(i = 1\)):

\[
\tilde{L}_1(t) x_1^{(t)} = b_1^{(t,x)}
\]
Distributed Localization and Tracking (1/5)

- Star topology:
Distributed Localization and Tracking (2/5)

1. Node $i$ creates Laplacian matrix of star topology

2. Computes differential coordinates: $\delta_i^x = \frac{1}{d_i} \sum_{j \in N_i} (-z_{d,ij} \sin(z_{az,ij}))$

3. Receives control inputs (linear and ang. velocity) and GPS measurements $(z_{p,j}^x, z_{p,j}^y)$ from neighbors

4. Node $j$ sends its own vector of range measurements with respect to its neighborhood

5. Node $i$ must find measurement $(z_{d,ji}, z_{az,ji}) \rightarrow$ data association
Distributed Localization and Tracking (3/5)

- Association:
  1. Node $i$ creates “synthetic” distance $z^s_d$ and angle $z^s_{az}$ using $(z^x_{p,i}, z^y_{p,i})$ and $(z^x_{p,j}, z^y_{p,j})$
  2. Creates ego vector: $vec_i = \begin{bmatrix} -z^s_d \sin(z^s_{az}) \\ -z^s_d \cos(z^s_{az}) \end{bmatrix}$
  3. Creates matrix for range measurements of $j$: $mat_j = \begin{bmatrix} -z_{d,j1} \sin(z_{az,j1}) & -z_{d,j2} \sin(z_{az,j2}) & \ldots & -z_{d,jN_j} \sin(z_{az,jN_j}) \\ -z_{d,j1} \cos(z_{az,j1}) & -z_{d,j2} \cos(z_{az,j2}) & \ldots & -z_{d,jN_j} \cos(z_{az,jN_j}) \end{bmatrix}$
  4. Find the Euclidean norms of $vec_i$ and each column of $mat_j$
  5. The minimum of those norms correspond to: $z_{d,ji}$ and $z_{az,ji}$
Distributed Localization and Tracking (4/5)

• Least squares minimization:

$$\arg\min_{x_i} \| \tilde{L}_i x_i - b_i^x \|_2^2$$

• State vector $$x_i \in \mathbb{R}^{N_i + 1}$$: $$x$$ positions of ego and neighbors

• Measurement vector $$b_i^x \in \mathbb{R}^{2(N_i + 1)}$$:

$$b_i^x = \begin{bmatrix}
\delta_i^x \\
-\z_{d,j_i} \sin \z_{az,j_i} \\
... \\
\z_{p,i}^x \\
\z_{p,j}^x \\
... \\
\end{bmatrix}$$

→ Range measurements of ego and neighbors

→ GPS measurements of ego and neighbors
Distributed Localization and Tracking (5/5)

• Extended Kalman Filter:

\[
\begin{align*}
x^t &= f(x^{t-1}, u^t) + \mathcal{N}(0, R) \\
z^t &= g(x^t) + \mathcal{N}(0, Q)
\end{align*}
\]

• State vector \( x^t \in \mathbb{R}^{3(N_i+1)} \): contains \( x, y, \theta \) of ego and neighbors

• Measurement vector \( z^t \) and \( g(x^t) \) according to Laplacian measurement model

\[
g(x^t) = Hx^t, \quad H = \begin{bmatrix} \tilde{L}_i & 0 & 0 \\
0 & \tilde{L}_i & 0 \\
0 & 0 & 1 \end{bmatrix}
\]
Extended Kalman Filter as benchmark (1/2)

- Extended Kalman Filter:
  \[
  x^t = f(x^{t-1}, u^t) + N(0, R)
  \]
  \[
  z^t = g(x^t) + N(0, Q)
  \]

- State vector \( x^t \in \mathbb{R}^{3(N_t+1)} \): contains \( x, y, \theta \) of ego and neighbors

- Measurement vector \( z^t \):

\[
  z^t = [Z_{d,i} \ldots Z_{d,i} Z_{d,l} \ldots Z_{a,i} Z_{a,l} \ldots Z_{p,i} Z_{p,j} \ldots]
\]

- Distance measurements for ego and neighbors
- Angle measurements for ego and neighbors
- GPS measurements for ego and neighbors

\[\text{Distance measurements for ego and neighbors} \]
\[\text{Angle measurements for ego and neighbors} \]
\[\text{GPS measurements for ego and neighbors} \]
Extended Kalman Filter as benchmark (2/2)

- Extended Kalman Filter:
  \[ x^t = f(x^{t-1}, u^t) + \mathcal{N}(0, R) \]
  \[ z^t = g(x^t) + \mathcal{N}(0, Q) \]

- Nonlinear function \( g(x^t) \):
  \[ g(x^t) = \begin{bmatrix} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\ atan\left(\frac{y_j - y_i}{x_j - x_i}\right) \\ \vdots \\ x_i \\ \vdots \\ y_i \\ \vdots \\ \theta_i \end{bmatrix} \]

  - Distance model
  - Angle model
  - GPS and heading model

- Jacobian matrix: \( H = \frac{\partial g(x^t)}{\partial x^t} \bigg|_{x_0} \) (linearization point \( x_0 \): GPS measurements)

- Data association in two directions, e.g., find \( z_{d,ij} \) and \( z_{d,ji} \) which best fits GPS of i and j
**Results (1/3)**

**Perfect association**

![CDF graph for perfect association](image1)

Reduction of GPS Localization Mean Square Error:
1) CCEKF: 80%
2) Local tracker: 85%
3) Jacob tracker: 77%

**Not perfect association**

![CDF graph for not perfect association](image2)

Reduction of GPS Localization Mean Square Error:
1) CCEKF: 80%
2) Local tracker: 85%
3) Jacob tracker: 56%
Results (2/3) – Individual vehicle (idx = 9)

Reduction of GPS Localization Error:
1) CCEKF: 75%
2) Local tracker: 82%
3) Jacob tracker: 74%

Reduction of Average GPS Localization Error of neighborhood:
1) Local tracker: 68%
2) Jacob tracker: 62%
Results (3/3) – Individual vehicle (idx = 9)

Not perfect

Reduction of GPS Localization Error:
1) CCEKF: 75%
2) Local tracker: 80%
3) Jacob tracker: 71%

Reduction of Average GPS Localization Error of neighborhood:
1) Local tracker: 58%
2) Jacob tracker: 57%
Feature Preserving Mesh Denoising Based on Graph Spectral Processing

Can a method handle all these situations?
Related Work

Denoising Methods

Anisotropic

- **Anisotropic Geometric Diffusion**
  - Anisotropic Filtering of Non-Linear Surface Features [HP04]
  - Anisotropic diffusion of surfaces and functions on surfaces [BX03]
  - Smoothing by example: Mesh denoising by averaging with similarity-based weights [YBS06]
  - Rolling guidance normal filter for geometric processing [WFLTG15]

- **Bilateral Filtering of Vertices & Normals**
  - Guided mesh normal filtering [ZDZL16]
  - Bi-normal filtering for mesh denoising [WYPWQLH15]
  - Bilateral mesh denoising [FD11]
  - Non-iterative, feature preserving mesh smoothing [JDD03]
  - A cascaded approach for feature preserving surface mesh denoising [WZY12]
  - Fast and effective feature-preserving mesh denoising [SRML07]
  - Bilateral normal filtering for mesh denoising [ZFC11]
  - Fuzzy vector median-based surface smoothing [SB04]
  - Coarse-to-fine normal filtering for feature-preserving mesh denoising based on isotropic subneighborhoods [ZWW13]

Sparse Modeling and Recovery

- A robust scheme for feature preserving mesh denoising [LDC16]
- Variational mesh denoising using total variation and piecewise constant function space [ZWZ15]
- Mesh denoising via l0 minimization [H13]
- Efficiently combining positions and normals for precise 3D geometry [NRD05]
- A bayesian method for probable surface reconstruction and decimation [DT06]
- Tensor voting guided mesh denoising [WLP17]
- Robust feature-preserving mesh denoising based on consistent subneighborhoods [FP10]
- Decoupling noises and features via weighted l1-analysis compressed sensing [WYLD14]

Data Driven Approaches

- Mesh denoising via cascaded normal regression [WLT16]

Isotropic

- Laplacian smoothing and Delaunay triangulations [F88]
- A signal processing approach to fair surface design [T95]
- Implicit fairing of irregular meshes using diffusion and curvature flow [DMS99]
Schema of Our Approach

Two Stage Graph Spectral Processing (TSGSP)

Coarse Denoising
- Noisy Mesh
- Partitioning
- Smoothing Factor per Part
- Low Pass Graph Filter Using OI
- Evaluate Cut off Frequency
- Smoothed Vertices
- Smoothed Normals

Feature-aware Fine Denoising
- Features Extraction
- Ideal Patches Selection
- Graph Spectral Processing
Two Stage Graph Spectral Processing

Fast Coarse Denoising

- Sequential Scanning in Parts
- Mesh Partitioning

Initialization of Parameters per Part and Coordinate

Estimation of Noise and Ideal Subspace Size Using Model based Bayesian Learning

Fast GSP using Orthogonal Iterations

Smoothed Vertices

Feature Aware Guided Normal Filtering

- Identification of Features
- Iterative Vertex Updating Scheme Using Graph Spectral Processed Normals

Smoothed Normals

Ideal Patches Selection for Guided Normal Estimation

Noisy Mesh

Mesh Partitioning

Smoothed Features
Polygon Models

Face \( f_i = \{v_{i1}, v_{i2}, v_{i3}\} \)

First ring area \( F = \{f_{i1}, f_{i2}, \cdots, f_{ik}\} \)

Normal of centroid \( n_{ci} = \frac{(v_{i2} - v_{i1}) \times (v_{i3} - v_{i1})}{\| (v_{i2} - v_{i1}) \times (v_{i3} - v_{i1}) \|} \)

Centroid \( c_i = \frac{(v_{i1} + v_{i2} + v_{i3})}{3} \)

Vertex \( v_i = (x_i, y_i, z_i) \)
Cells connectivity

- Each mesh consists of points that are connected with each other creating cells

Random part of a point cloud
Cells connectivity

- Each mesh consists of points that are connected with each other creating cells

\[ R_{ij} = \begin{cases} 
1 & \text{if } i, j \in E \\
0 & \text{otherwise}
\end{cases} \]
## Adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
**Laplacian matrix**

- **Laplacian matrix** $L = D - R$

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>4</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>5</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>
Graph Fourier Transform

\[ \mathcal{L} = \begin{bmatrix} \mathbf{U}_{k-m} & \mathbf{U}_m \end{bmatrix} \begin{bmatrix} \Lambda_{k-m} & 0 \\ 0 & \Lambda_m \end{bmatrix} \left( \begin{bmatrix} \mathbf{U}_{k-m} \\ \mathbf{U}_m \end{bmatrix} \right)^T \]

\[ \hat{\mathbf{v}} = \mathbf{U}_m \mathbf{U}_m^T \mathbf{v} \]

Which is the ideal value of \( m \)?

It needs to be removed.

Smoothed vertices

Laplacian Matrix

Eigenvectors

Eigenvalues

Noisy subspace
Finding the Ideal Subspace Size $m$

- Eigenvalues
- Projected values of $x$ coordinate into $U^T$

Features & Noise
Smoothed areas
Choosing Different Number of m

- Keeping the 10%
- Keeping the 20%
- Keeping the 30%
- Keeping the 40%
Limitations of Spectral Analysis in Large Meshes

Spectral analysis is impossible to be applied in large meshes (time consuming)
Moreover, we no longer deal exclusively with individual shapes, but with entire scenes, resulting in a sequence of 3D surfaces that are affected by noise with different characteristics. e.g.,

- Aerial scanning [1], [2]
- Slam scanning [3]
- Underwater scanning [4]
- Scalable multiobject scanning [5]
- Large-scale terrestrial scanning [6]
- Large statues scanning [7]

Spectral Analysis per Submesh

Using SVD in each submesh

\[
[U_1, \Lambda_1, V_1] = SVD(R_1)
\]

\[
[U_2, \Lambda_2, V_2] = SVD(R_2)
\]

\[
[U_s, \Lambda_s, V_s] = SVD(R_s)
\]

\[ R = (L + \delta I)^{-1} \]

Laplacian Matrix

Small positive scalar value ensuring positive definiteness of R
Spectral Analysis per Submesh

\[ R = (L + \delta I)^{-1} \]

\[
[U_1, \Lambda_1, V_1] = SVD(R_1) \\
[U_2, \Lambda_2, V_2] = SVD(R_2) \\
[U_s, \Lambda_s, V_s] = SVD(R_s)
\]

It is also a time consuming process
Our Approach (Orthogonal Iterations)

Initialization

\[ [U_1, \Lambda_1, V_1] = SVD(R_1) \]

If \( i = 1 \)

\[ U_i = \text{Orthonorm}\{R_i^\beta U_{i-1}\} \]

If \( i > 1 \)

Exploit the coherence between the spectral components of the different submeshes using orthogonal iterations (OI)

\[ R = (L + \delta I)^{-\beta} \]

For \( \beta > 1 \), the required iterations for convergence are reduced
Orthogonal Iterations Results

Similar results in comparison with SVD but much faster!!

<table>
<thead>
<tr>
<th>Twelve</th>
<th>Sphere</th>
<th>Fandisk</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>$\theta$</td>
<td>$t$</td>
</tr>
<tr>
<td>$R^1$</td>
<td>0.031</td>
<td>11.57</td>
</tr>
<tr>
<td>$R^3$</td>
<td>0.099</td>
<td>10.04</td>
</tr>
<tr>
<td>$R^5$</td>
<td>0.136</td>
<td>9.871</td>
</tr>
<tr>
<td>$R^7$</td>
<td>0.157</td>
<td>9.82</td>
</tr>
<tr>
<td>$R^9$</td>
<td>0.201</td>
<td>9.77</td>
</tr>
<tr>
<td>SVD</td>
<td>0.901</td>
<td>9.83</td>
</tr>
</tbody>
</table>

Execution time (in seconds) and angle difference $\theta$ for different cases of $R^\beta$ and $SVD$. 
Orthogonal Iterations Results

Coarse denoising results using different value of $R^\beta$

(a) $\beta = 1$, (b) $\beta = 2$, (c) $\beta = 3$, (d) $\beta = 4$, (e) $\beta = 5$, (f) $\beta = 6$, (g) $\beta = 7$, (h) SVD

Visual results of OI and SVD are identical for $\beta > 4$
Segmentation Examples Using Metis
Example of Using Overlapped Submeshes

Small-scale features may be cut by the submesh separation

However, they appear uncut in another submesh

We guarantee that any point of the mesh appears with full connectivity degree in at least one submesh.
How we overcome the problems created by the mesh separation

- Overlapped patches
- Weighted average for the reconstruction

Example

- The red point appears in three different overlapped submeshes (a), (b) and (c)
- In each submesh, it has a different degree related to the connected points contained in the same submesh.
- The corresponding weights are: (a) 5, (b) 4, (c) 3
Fandisk model using different partitioning

Edge effect problem is apparent in areas where two or more segments are connected

Using overlapping approach, edge effect problem is mitigated. However, it is not eliminated

Our proposed approach seems to be unaffected by different partitioning cases
In lower levels of noise, more eigenvalues can be kept. The subspace size of noise is shorter.

Different models follow a common behavior.

Higher level of noise means lower remaining eigenvalues.

Remaining percentage of eigenvalues per different levels of noise.
Initial Estimation of Value $m[0]$

$$m_{ij}[0] = k_i \left(1 - \frac{e^{-|e_{ij}|}}{\sigma_z[0]}\right) \forall i = 1, s, \forall j \in \{x, y, z\}$$

- **Initial estimated value of $m$ per segment and per coordinate**
- **Number of points included in the $i$-th submesh**
- **Normalized value of smoothing factor**
- **Standard deviation of noise vector**
- **Per segment and per coordinate**
- **Number of Submeshes**
Estimation of Smoothness for Each Patch

\[ e_i = \| L^{1/2}[i] v_i \|_2^2 = v_i^T L[i] v_i, \quad \forall i = 1, \ldots, s \]

- Small values of \( \tilde{e}_{ij} \) represent smooth areas
- Large values of \( \tilde{e}_{ij} \) represent rough areas
The smoother an area is, the lower the smoothness factor is.

In areas with more high-frequency features the value of smoothness factor appears to be higher.

Heatmap of Smoothness Factor per Segment

Different views of Julio model
Different Smoothness Factor per Different Areas

Different colors of heatmap define areas with different smoothness factor

Non Smoothed Areas

Areas with corners have higher values (yellow areas) in comparison with smoothed areas (blue areas)

Smoothed Areas
Dynamic Identification of Ideal $m^*$

- Identification of the optimal Subspace Size $m$ per submesh and per coordinate, using Model based Bayesian Learning (MBL).
- We use the initialized values of $m_{ij}[0]$ and $\sigma^2_{z_{ij}}[0]$
- While stopping criterion $\sigma^2_{z_{ij}} - e < \left\| v_{ij} - U m_{ij} v_{p_{ij}} \right\|^2 / k_i < \sigma^2_{z_{ij}} + e$ is not satisfied

\[
if \quad \frac{\left\| v_{ij} - U m_{ij} v_{p_{ij}} \right\|^2}{k_i} < \sigma^2_{z_{ij}} - e \quad \rightarrow \quad m_{ij}[t] = m_{ij}[t - 1] + 1 \\
if \quad \frac{\left\| v_{ij} - U m_{ij} v_{p_{ij}} \right\|^2}{k_i} > \sigma^2_{z_{ij}} + e \quad \rightarrow \quad m_{ij}[t] = m_{ij}[t - 1] - 1
\]
Value of m for Minimizing the $\theta$ Metric

In each model there is a value $m$ (green cycle) which minimizes the metric $\theta$
The process takes place per patch (red part) and per coordinate (x in these Figures)

Twelve Model affected by different levels of noise

 Initialization of m helps for reducing the iterations of MBL

Initial value of m can be lower or higher of the ideal value
Features Identification

- For each face $i$ of the mesh we create a patch $P_i = \{ f_{i1}, f_{i2}, \ldots, f_{ik} \}$

The $k$ closest faces are selected by the $k$-nn algorithm.
Features Classification

We distinguish the following three cases:

(a) Corner ($\lambda_{i1} \cong \lambda_{i2} \cong \lambda_{i3}$)
(b) Edge ($\lambda_{i1} \cong \lambda_{i2} < \lambda_{i3}$)
(c) Flat area ($\lambda_{i1} < \lambda_{i2} \cong \lambda_{i3}$)

Features

Non - Features

k-means clustering to the eigenvalues

Features
Non - Features

2 Classes
Coarse Denoising Results

Normal Vectors of Original Mesh

Normal Vectors of Noisy Mesh

Smoothed Normal Vectors by Coarse Denoising
**K-nn Algorithm for Finding Neighbors**

Creating patches based on k-nn neighbors

\[ P_i = [f_{i1} f_{i2} f_{i3} \ldots f_{ik}] \]

This area is not always the ideal representative of face \( f_i \)

For each centroid we find k nearest centroids, creating a patch of neighbor centroid which can be used as an initial candidate patch.
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

26/2/2021
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΩΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

26/2/2021
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

26/2/2021
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ
Candidate Areas of Blue Face

Which one of them is the Ideal patch?
Rules to Estimate the Ideal Candidate Areas

The best neighborhood of a feature face $i$ is estimated based on:

$$A_i = \begin{cases} P_{ig} | \max \left( \frac{\tau_g \cdot \xi_{ig}}{\omega_{ig}} \right) & \text{if } f_i \text{ has been classified as feature} \\ P_i & \text{if } f_i \text{ has been classified as flat area} \end{cases}$$

$$\tau_g = \frac{\left| \lambda_{g1} - \lambda_{g3} \right|}{\lambda_{g3}}$$

$$\xi_{ig} = < \mathbf{n}_{ci}, \sum_{j \in P_{ig}} \frac{\mathbf{n}_{cj}}{|P_{ig}|} >$$

$$\omega_{ig} = \max \left( |\mathbf{n}_{ci} - \mathbf{n}_{cj}|_2 \right) \forall j \in P_{ig}$$

- The normalized difference $\tau_g$ between max and min eigenvalues.
- The inner product between the face normal and the average normal of the patch.
- The maximum distance between the face normal and the other face normals belonging to the same patch.
Fine Tune the Normals

The ideal neighborhoods and weights are used to fine tune the normals according to:

\[ \bar{n}_c = (D^{-1}C_w)\zeta \hat{n}_c \]

\[ C_w = W_c \circ W_s \circ C_a \]

- Iteratively updating of the vertices

\[ v_i^{(t+1)} = v_i^{(t)} + \frac{\sum_{j \in \Psi_i} \bar{n}_{cj}^{(t)}}{|\Psi_i|} \left( < \bar{n}_{cj}^{(t)}, (c_j^{(t)} - v_i^{(t)}) > \right) \]

\[ c_j^{(t+1)} = \frac{v_{j1}^{(t+1)} + v_{j2}^{(t+1)} + v_{j3}^{(t+1)}}{3} \quad \forall j \in \Psi_i \]

- Bilateral weights

\[ W_{ci} = \begin{cases} e^{-\frac{\|c_i-c_j\|^2}{2\sigma_c^2}} & \text{if } C_{ai} = 1 \\ 0 & \text{otherwise} \end{cases} \]

\[ W_{si} = \begin{cases} e^{-\frac{\|\bar{n}_{ci}-\bar{n}_{cj}\|^2}{2\sigma_s^2}} & \text{if } C_{ai} = 1 \\ 0 & \text{otherwise} \end{cases} \]
Results after using a wide range of alternative approaches
Gaussian weights

TSGSP

knn patches

After using the k-nn patch without searching for the ideal patches

After using Gaussian weights instead of Bilateral weights

Noisy Model

Original Noisy Mesh

TSGSP

After using our proposed approach TSGSP
Our approach preserves the real geometry and the vertices position
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

L₀ min
Mesh Guided
TSGSP
Original
How Coarse Denoising Step Can Improve the Results of Other Methods When it is Used as a Pre-processing Step
Denoising results of state-of-the-art methods (Block Model)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilateral L</td>
<td>Non Iterative</td>
</tr>
<tr>
<td>Fast &amp; Effective</td>
<td>Bilateral Normal (l)</td>
</tr>
<tr>
<td>Bilateral Normal</td>
<td>Bilateral Normal (g)</td>
</tr>
<tr>
<td>L_0 min</td>
<td>Guided Mesh</td>
</tr>
</tbody>
</table>

Denoising results of the same methods using coarse desoising as pre-processing step

Similar results as previous, however the process is faster because less iterations are required.
Denoising Results of Scanned Object by Kinect Devices
Kinect v1 (cone)

Original Mesh

Noisy Mesh

θ metric in degrees

Bilateral Normal Filtering (12.98°)

Mesh Guided (10.38°)

L₀ min (10:34°)

Cascade (8.91°)

TSGSP (7.98°)
Kinect fusion (Boy)

Original Mesh

Noisy Mesh

θ metric in degrees

Bilateral Normal Filtering (9.39°)

Mesh Guided (7.90°)

$L_0$ min (7.65°)

Cascade (7.79°)

TSGSP (7.59°)
Kinect fusion (Pyramid)

Original Mesh

Noisy Mesh

θ metric in degrees

Bilateral Normal Filtering
(9.13°)

Mesh Guided
(7.84°)

L₀ min
(7.69°)

Cascade
(7.59°)

TSGSP
(7.40°)
Kinect v2 (Cone)

Original Mesh

Noisy Mesh

θ metric in degrees

Bilateral Normal Filtering (8.50°)

Mesh Guided (7.62°)

$L_0$ min (7.74°)

Cascade (7.68°)

TSGSP (7.41°)
Staircase Effect Denoising Results
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

Original scanned model  Bilateral  Fast & Effective  Bilateral normal  TSGSP  Staircase-Aware Smoothing
More Denoising Results
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

Original

Noisy

[ZDZBL15]
theta = 7.94 deg

[WLT16]
theta = 7.67 deg

Our Approach
theta = 6.94 deg

26/2/2021
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

Original
Noisy
[ZDZBL15] theta = 4.75 deg
[WLT16] theta = 3.75 deg
Our Approach theta = 3.29 deg
Original  Noisy  \([\text{ZDZBL15}]\)  \(\text{theta} = 1.15\) deg  \([\text{WLT16}]\)  \(\text{theta} = 1.80\) deg  Our Approach  \(\text{theta} = 0.66\) deg
ΑΛΓΟΡΙΘΜΙΚΕΣ ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΕ ΕΜΦΑΣΗ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

Original

Noisy

[ZDZBL15]
theta = 3.20 deg

[WLT16]
theta = 3.05 deg

Our Approach
theta = 2.75 deg
Ερωτήσεις