UC San Diego

Laplacian

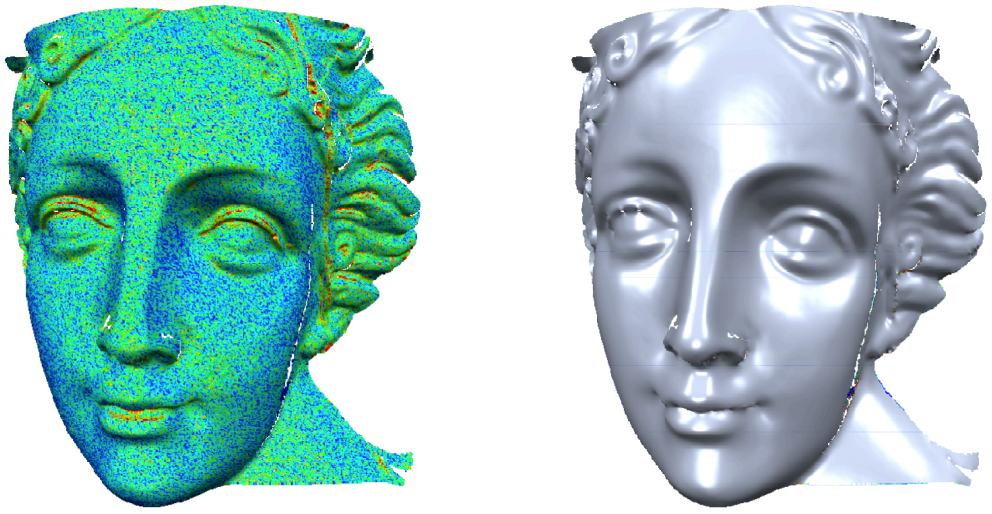
Instructor: Hao Su

MESH SMOOTHING (AKA DENOISING, FILTERING, FAIRING)

UC San Diego

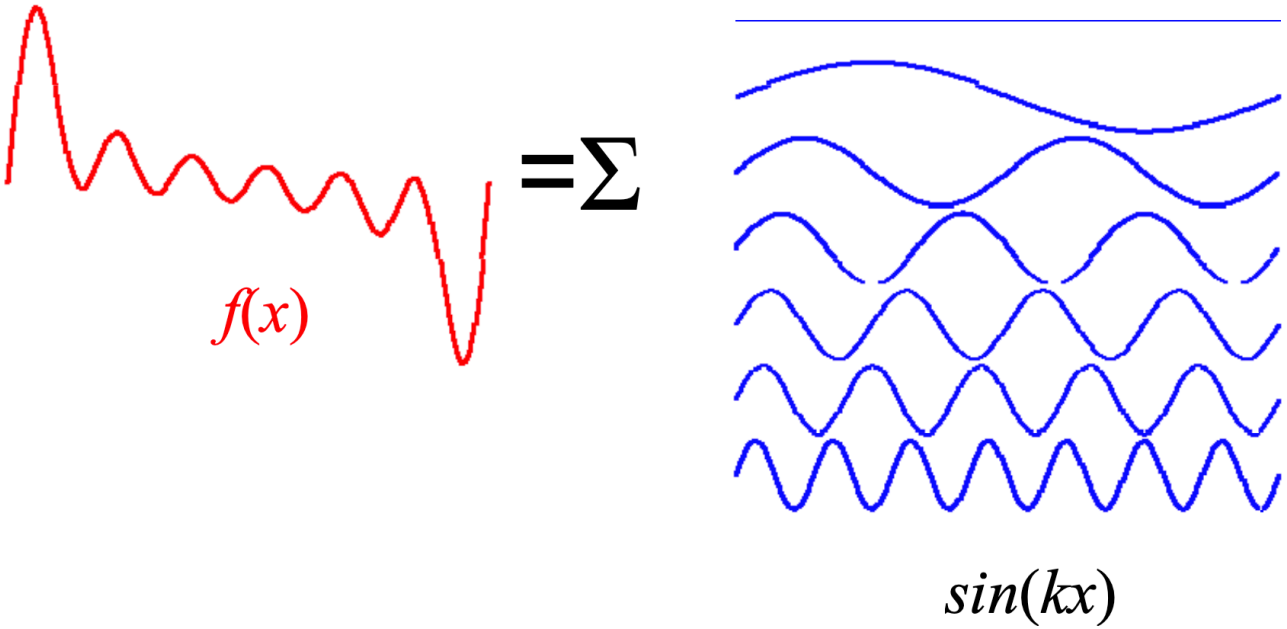
Mesh Smoothing

- Input: Noisy mesh (scanned or other)
- Output: Smooth mesh
- How: Filter out high frequency noise



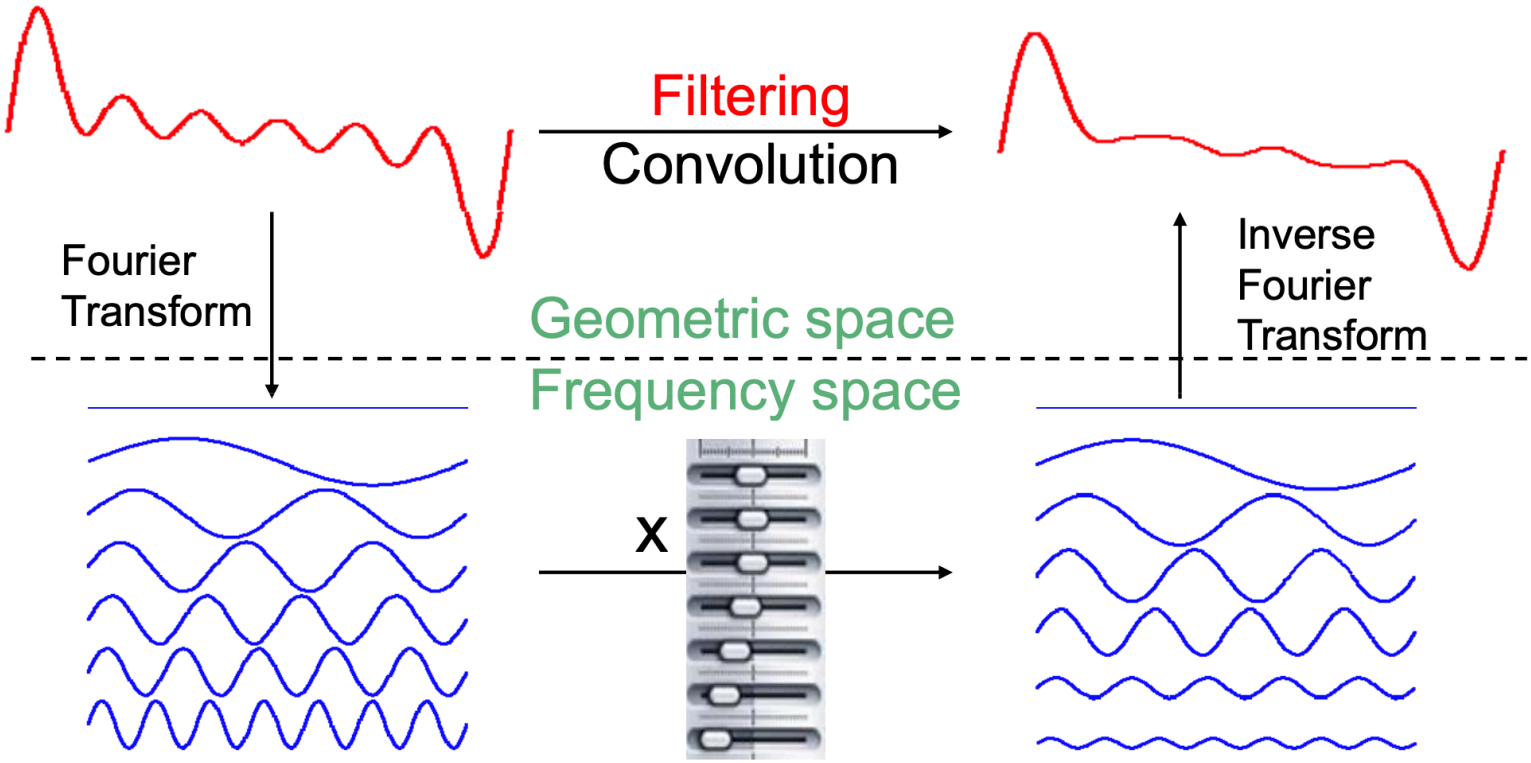
Smoothing by Filtering

Fourier Transform



Smoothing by Filtering

Fourier Transform



Filtering on a Mesh

Filtering

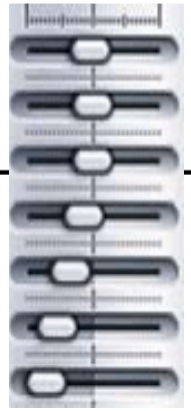
[Taubin 95]

Geometric space

Frequency space

?

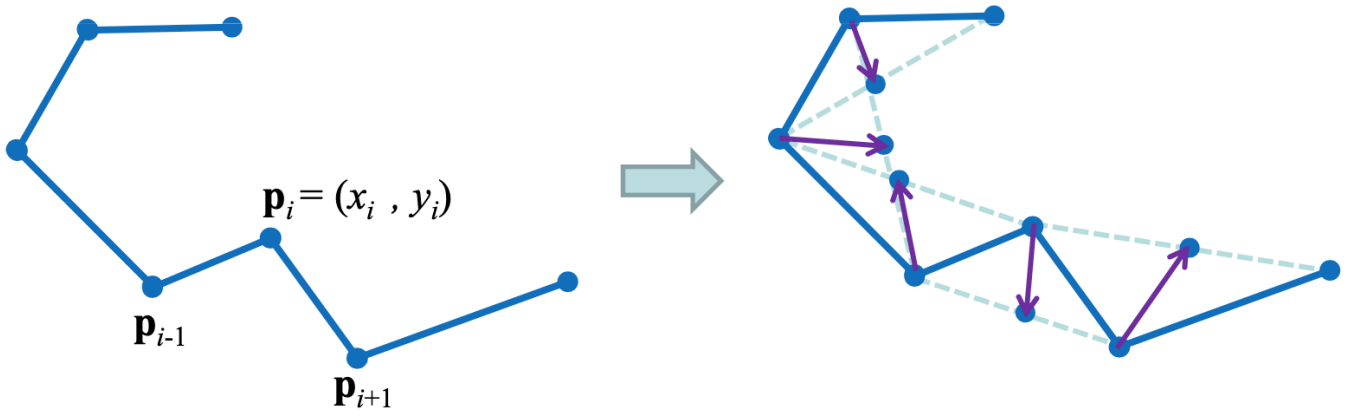
x



?

Laplacian Smoothing

- An easier problem: How to smooth a curve?

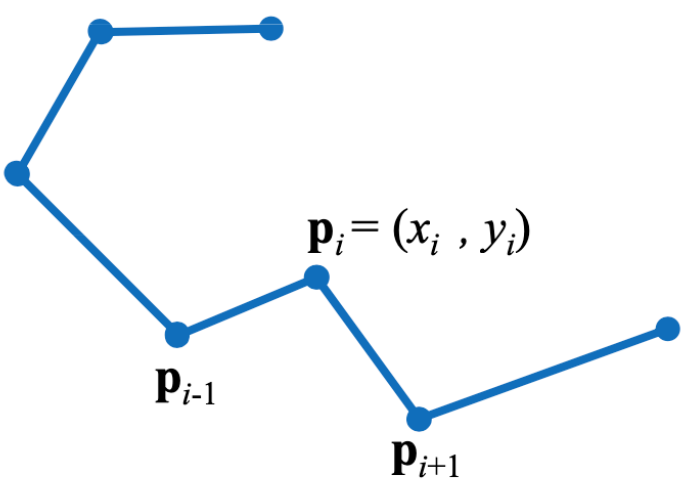


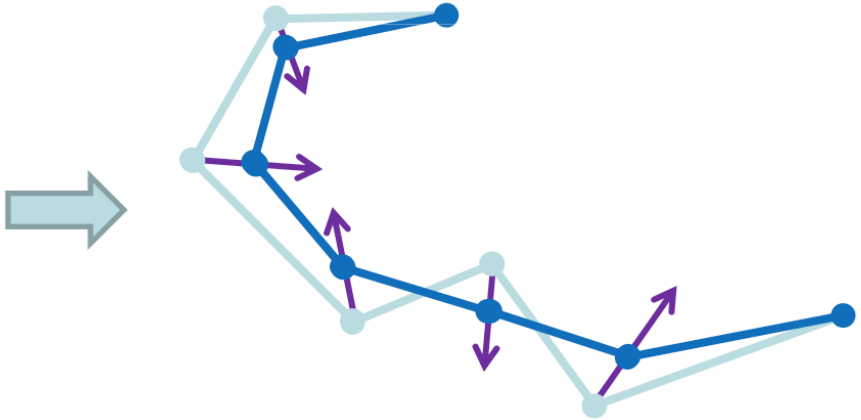
$$(\mathbf{p}_{i-1} + \mathbf{p}_{i+1})/2 - \mathbf{p}_i$$

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

Laplacian Smoothing

- An easier problem: How to smooth a curve?





$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{1}{2} L(\mathbf{p}_i)$$

Finite difference discretization of second derivative
= Laplace operator in one dimension

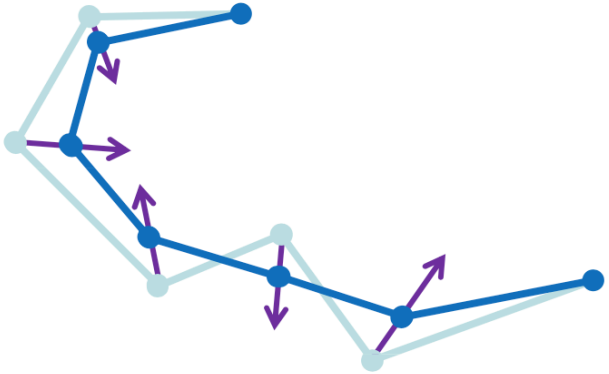
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

Laplacian Smoothing

Algorithm:

Repeat for *m* iterations (for non boundary points):

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda L(\mathbf{p}_i)$$



For which λ?

$0 < \lambda < 1$

Closed curve converges to?

Single point

Spectral Analysis

- Closed curve

Re-write $\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda L\left(\mathbf{p}_i^{(t)}\right)$

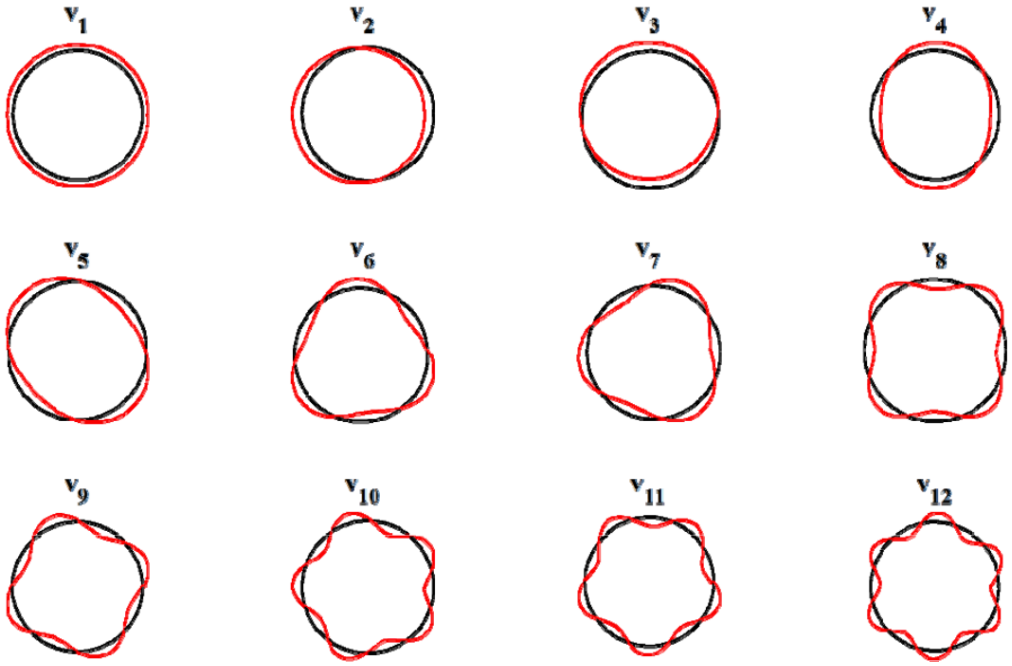
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

in matrix notation: $\mathbf{P}^{(t+1)} = \mathbf{P}^{(t)} - \lambda \mathbf{L}\mathbf{P}^{(t)}$

$$\mathbf{P} = \begin{pmatrix} x_1 & y_2 \\ \dots & \dots \\ x_n & y_n \end{pmatrix} \in \mathbb{R}^{n\times 2} \qquad \mathbf{L} = \frac{1}{2}\begin{pmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & & \dots & & & \\ & & & -1 & 2 & -1 \\ & & & & & \\ -1 & & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n\times n}$$

The Eigenvectors of L

$$\mathbf{L} = \mathbf{V}\mathbf{D}\mathbf{V}^T \qquad \mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & & | \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} k_1 & & & \\ & k_2 & & \\ & & \cdots & \\ & & & k_n \end{pmatrix}$$



$$0 \leq \lambda(L) \leq 2$$

Lemma 5.3.1. *The Laplacian of R_n has eigenvectors*

$$\boldsymbol{x}_k(u) = \cos(2\pi ku/n), \text{ and}$$
$$\boldsymbol{y}_k(u) = \sin(2\pi ku/n),$$

for $0 \leq k \leq n/2$, ignoring $\boldsymbol{y}_0$ which is the all-zero vector, and for even n ignoring $\boldsymbol{y}_{n/2}$ for the same reason. Eigenvectors $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ have eigenvalue $2 - 2\cos(2\pi k/n)$.

http://www.cs.yale.edu/homes/spielman/561/2012/lect05-12.pdf

Spectral Analysis

Then: $\mathbf{P}^{(t+1)} = \mathbf{P}^{(t)} - \lambda \mathbf{L}\mathbf{P}^{(t)} = \left(\mathbf{I} - \lambda \mathbf{L}\right)\mathbf{P}^{(t)}$

After m iterations: $\mathbf{P}^{(m)} = \left(\mathbf{I} - \lambda \mathbf{L}\right)^m \mathbf{P}^{(0)}$

Can be described using eigen-decomposition of $\mathbf{L}$

$$\mathbf{L} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

$$\mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & & | \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} k_1 & & & \\ & k_2 & & \\ & & \cdots & \\ & & & k_n \end{pmatrix}$$

$$\mathbf{P}^{(m)} = \mathbf{V}\left(\mathbf{I} - \lambda \mathbf{D}\right)^m \mathbf{V}^T \mathbf{P}^{(0)}$$

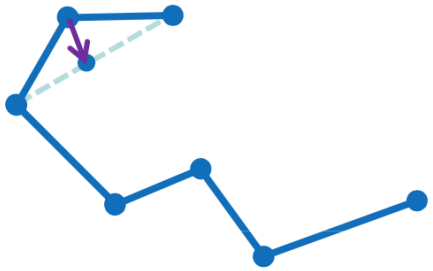
Filtering high frequencies

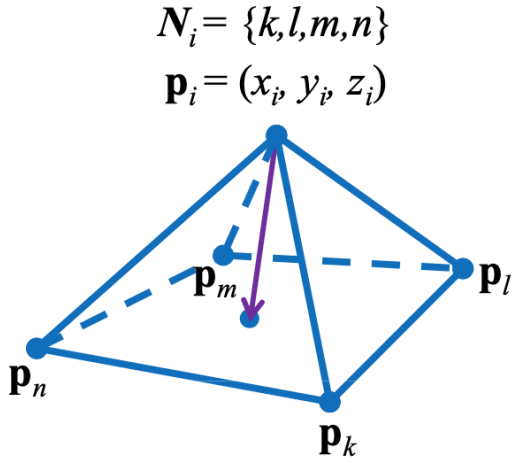
Laplacian Smoothing on Meshes

Same as for curves:

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$

What is $\Delta \mathbf{p}_i$?

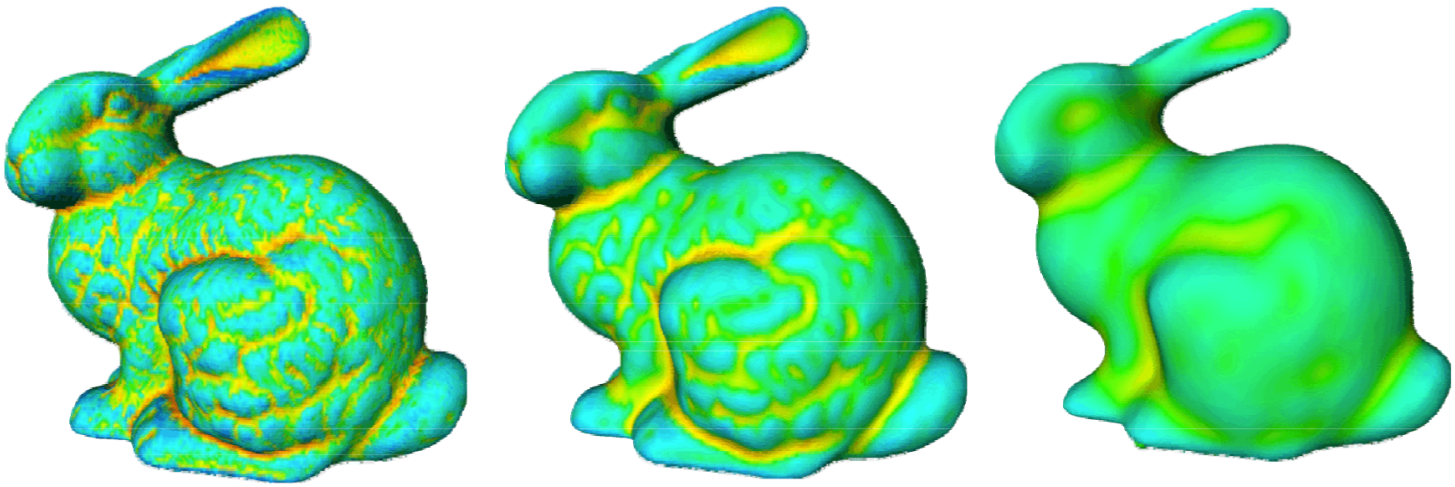




$$\frac{1}{2}\left(\mathbf{p}_{i+1} + \mathbf{p}_{i-1}\right) - \mathbf{p}_i$$

$$\frac{1}{|N_i|}\left(\sum_{j \in N_i} \mathbf{p}_j\right) - \mathbf{p}_i$$

Laplacian Smoothing on Meshes



0 Iterations

5 Iterations

20 Iterations

Laplacian Smoothing

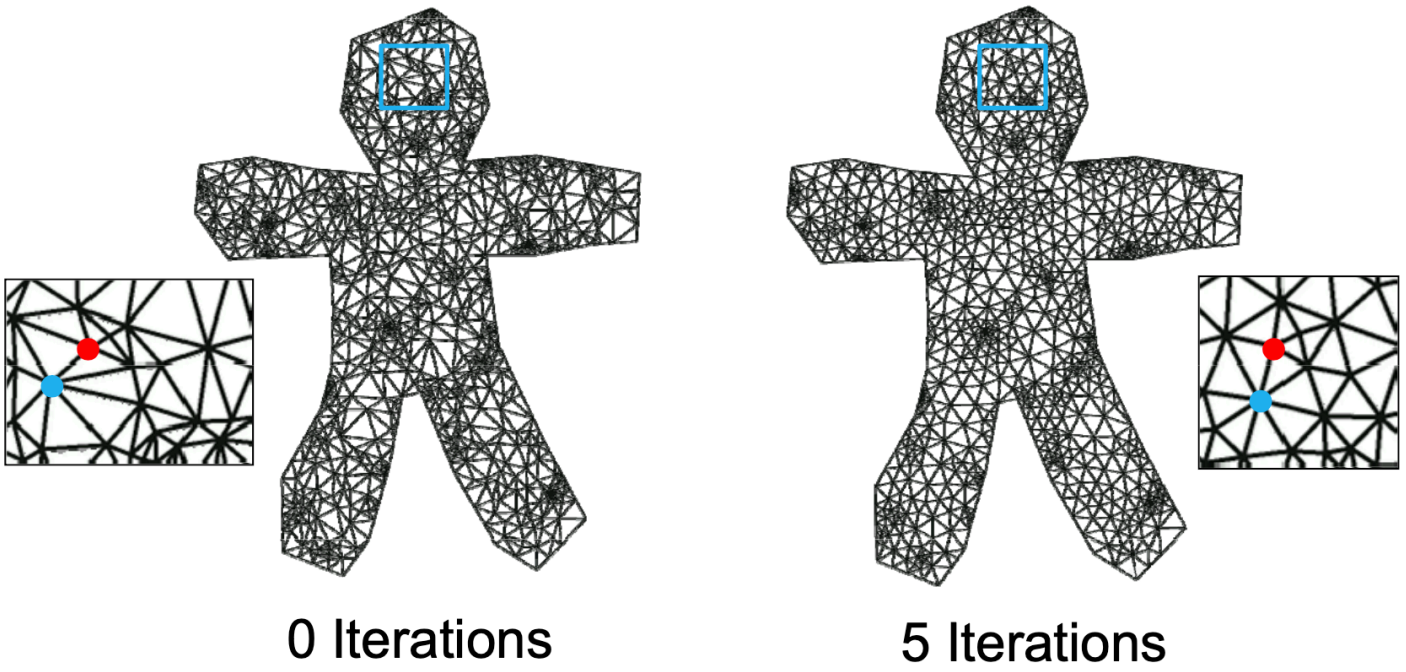
$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$

$\Delta \mathbf{p}_i$ = mean curvature normal

mean curvature flow

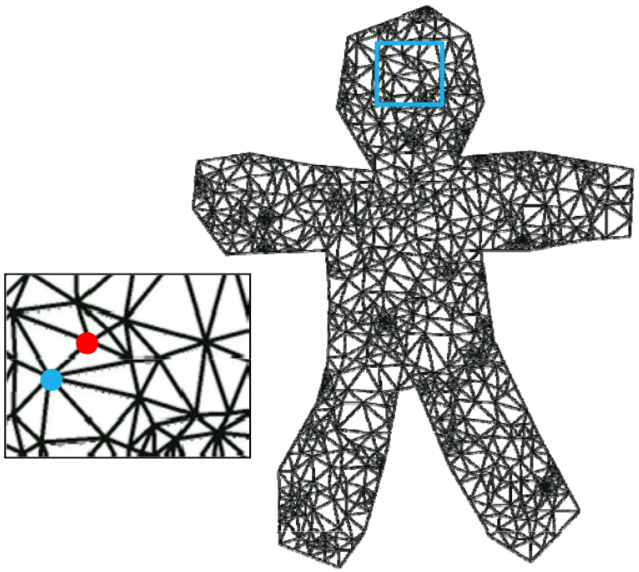
Laplace Operator Discretization

- Sanity check — what should happen if the mesh lies in the plane: $p_i = (x_i, y_i, 0)$?

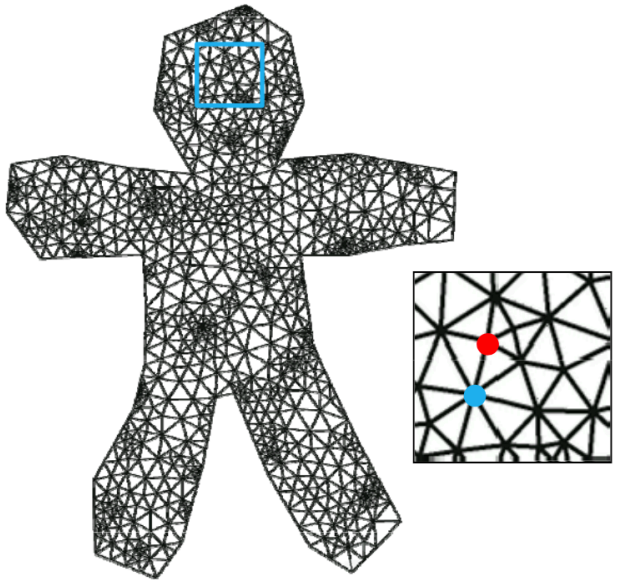


Laplace Operator Discretization

Not good – A flat mesh is smooth, should stay the same after smoothing



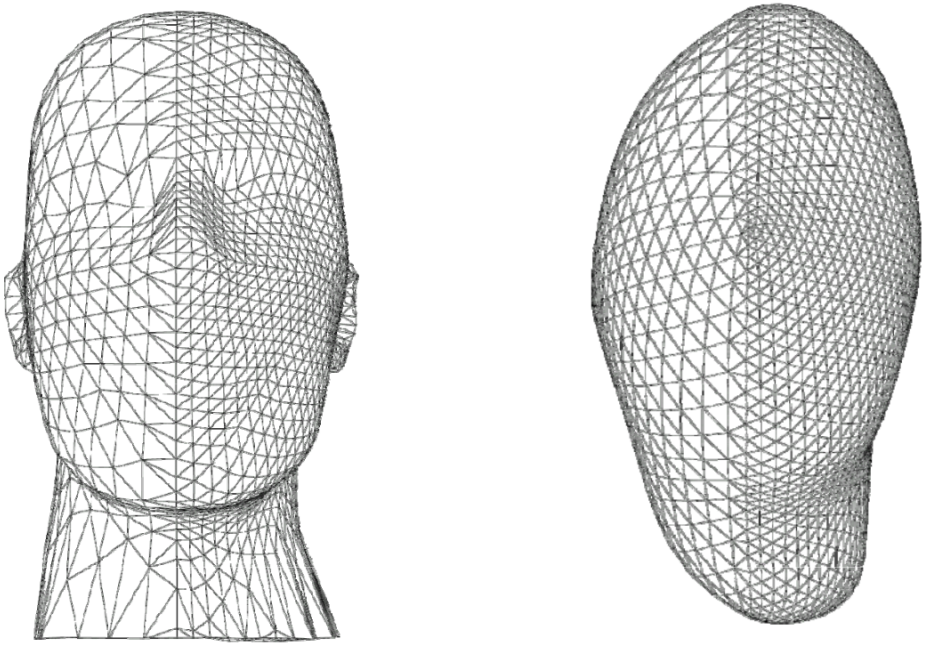
0 Iterations



5 Iterations

Laplace Operator Discretization

Not good – The result should not depend on triangle sizes



From Desbrun et al., Siggraph 1999

What Went Wrong?

Back to curves:

$$\frac{1}{2}\left(\mathbf{p}_{i+1}+\mathbf{p}_{i-1}\right)-\mathbf{p}_i$$

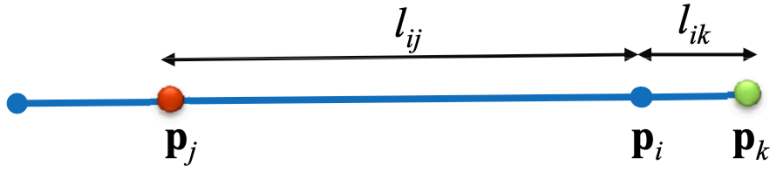
Same weight for both neighbors, although one is closer

The Solution (1D)

Use a weighted average to define Δ

Which weights?

$$w_{ij} = \frac{1}{l_{ij}} \qquad w_{ik} = \frac{1}{l_{ik}}$$



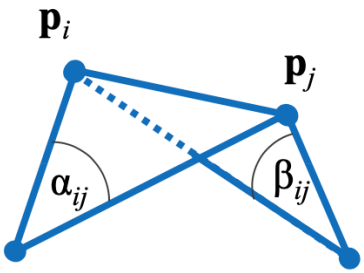
$$L(\mathbf{p}_i) = \frac{w_{ij}\mathbf{p}_j + w_{ik}\mathbf{p}_k}{w_{ij} + w_{ik}} - \mathbf{p}_i$$

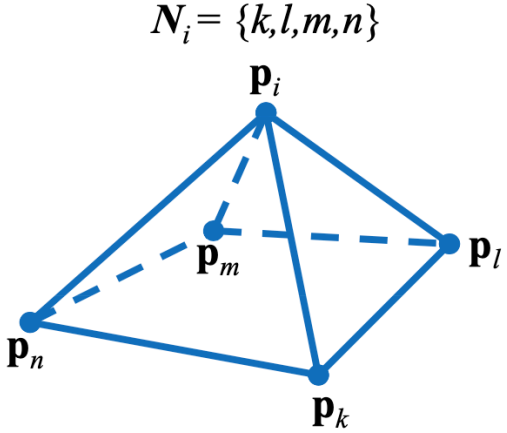
Straight curves will be invariant to smoothing

Solution (2D)

Use a weighted average to define Δ

Which weights?



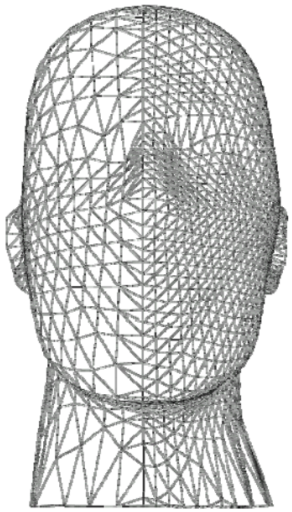


$$w_{ij} = \frac{h_{ij}^1 + h_{ij}^2}{l_{ij}} = \frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right)$$

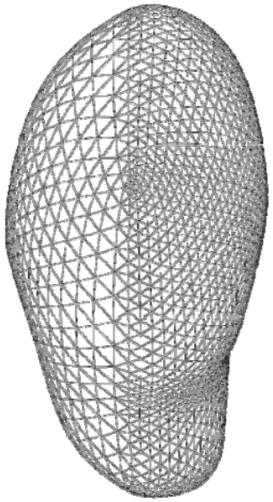
$$L(\mathbf{p}_i) = \frac{1}{\sum_{j\in N_i} w_{ij}}\left(\sum_{j\in N_i} w_{ij}\mathbf{p}_j\right) - \mathbf{p}_i$$

Planar meshes will be invariant to smoothing

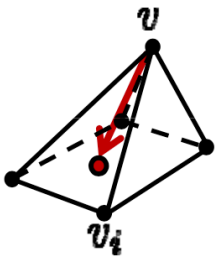
Smoothing with the Cotangent Laplacian



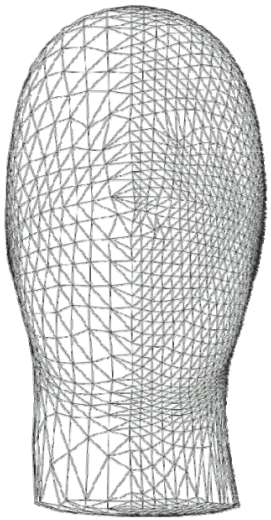
original



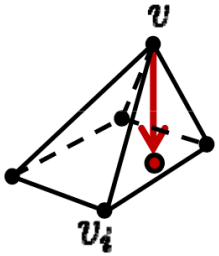
Uniform weights



normal
and
tangential
movement



Cotangent weights

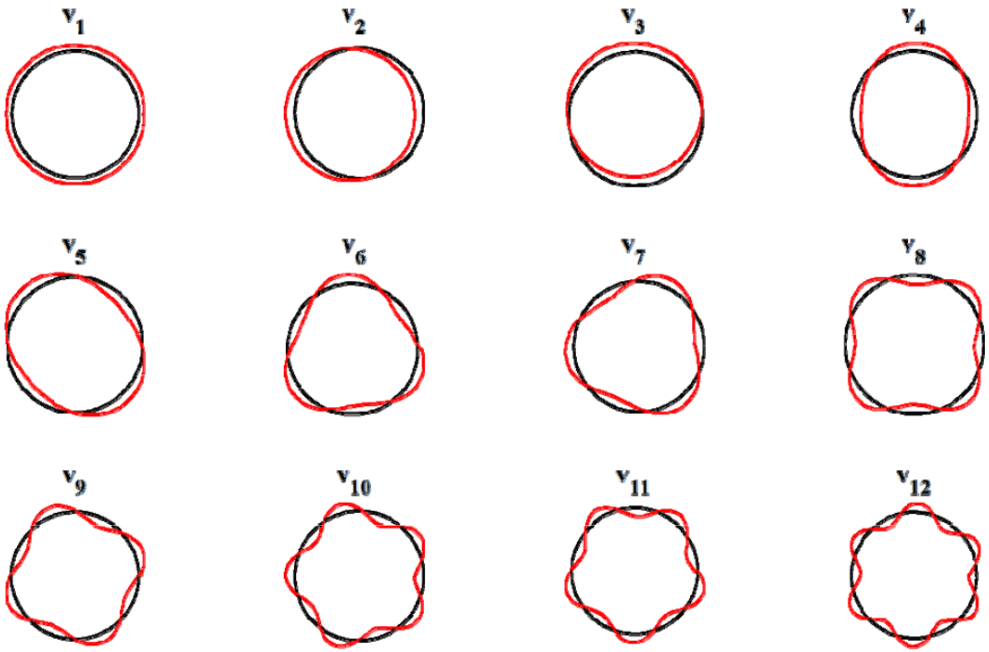


normal
movement

From Desbrun et al., Siggraph 1999

The Eigenvectors of L

$$\mathbf{L} = \mathbf{V}\mathbf{D}\mathbf{V}^T \qquad \mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & & | \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} k_1 & & & \\ & k_2 & & \\ & & \cdots & \\ & & & k_n \end{pmatrix}$$



Spectral Analysis

- Cotangent Laplacian

$$\mathbf{L} = \mathbf{V}\mathbf{D}\mathbf{V}^T \qquad \mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \\ | & | & & | \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} k_1 & & & \\ & k_2 & & \\ & & \dots & \\ & & & k_n \end{pmatrix}$$

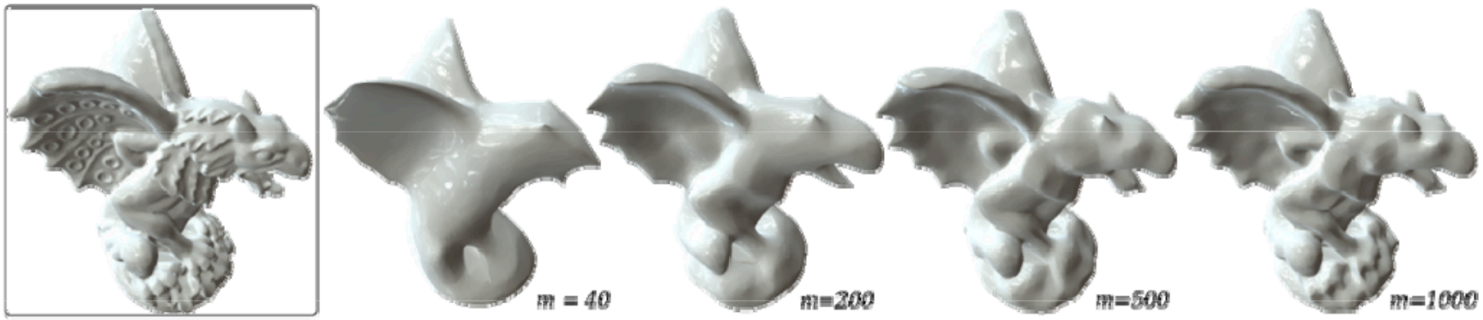
$\mathbf{v}_2$ $\mathbf{v}_{50}$

Demo

From Vallet et al., Eurographics 2008

Smoothing using the Laplacian Eigen-decomposition

$$\mathbf{P}^{smooth} = \mathbf{V}(\mathbf{D}_m)\mathbf{V}^T\mathbf{P} \quad , \quad \mathbf{D}_m = \begin{pmatrix} k_1 & & & \\ & \cdots & & \\ & & k_m & \\ & & & 0 \end{pmatrix}$$

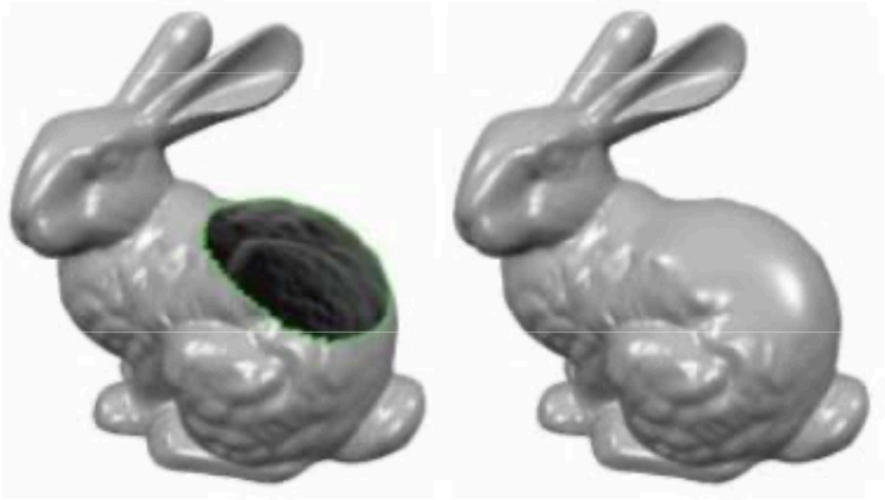


From Vallet et al., Eurographics 2008

SURFACE FAIRING

UC San Diego

- Find surfaces which are “as smooth as possible”
- Applications
  - Smooth blends
  - Hole filling



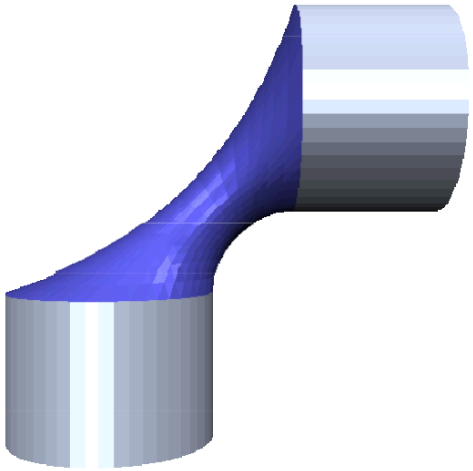
Fairness

- Idea: Penalize “unaesthetic behavior”

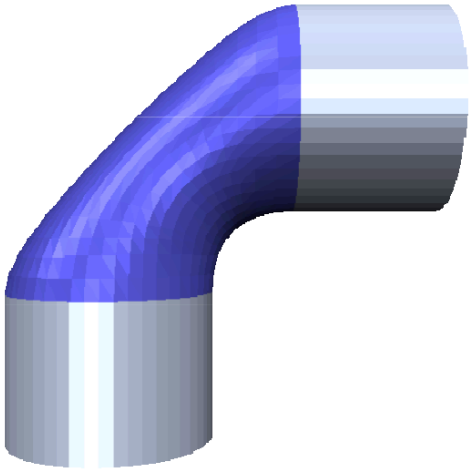
- Measure “fairness”
  - Principle of the simplest shape
  - Physical interpretation

- Minimize some fairness functional
  - Surface area, curvature
  - Membrane energy, thin plate energy

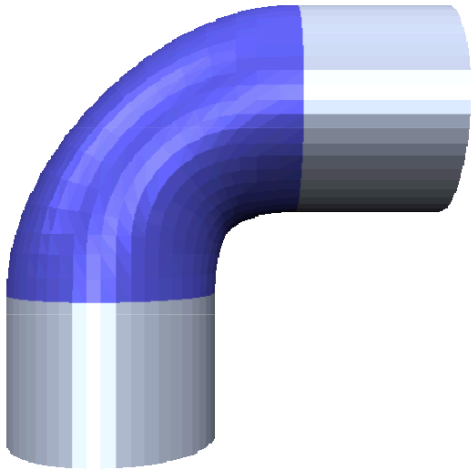
Energy Functionals



Membrane Surface



Thin Plate Surface



Minimum Variation Surface

- Membrane energy (surface area)

$$\int_S \mathrm{d}A \rightarrow \min$$

- Thin-plate energy (curvature)

$$\int_S \kappa_1^2 + \kappa_2^2 \, \mathrm{d}A \rightarrow \min$$

- Variational Calculus

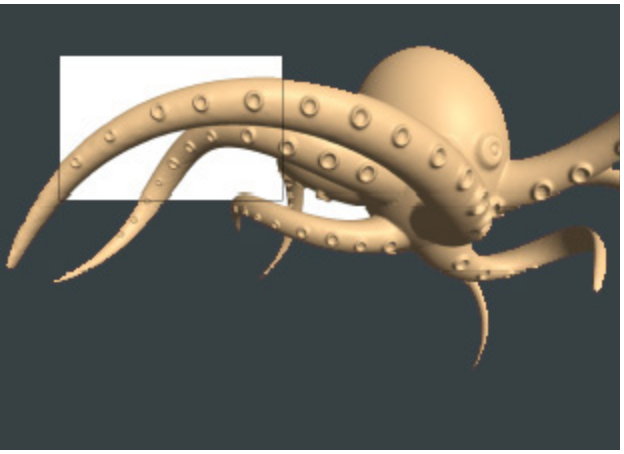
$$\Delta p = 0$$

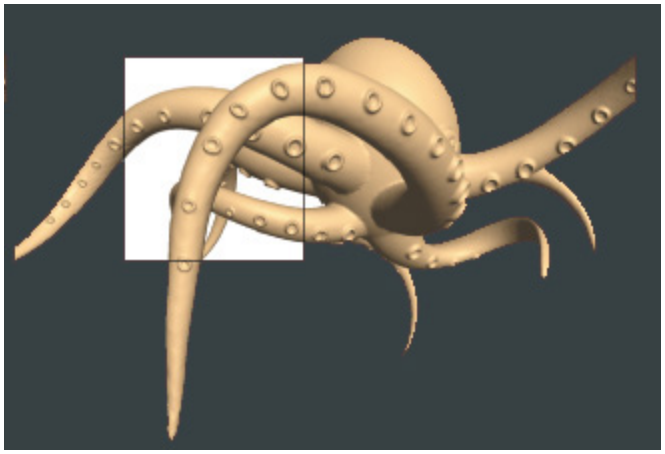
LAPLACIAN MESH EDITING

UC San Diego

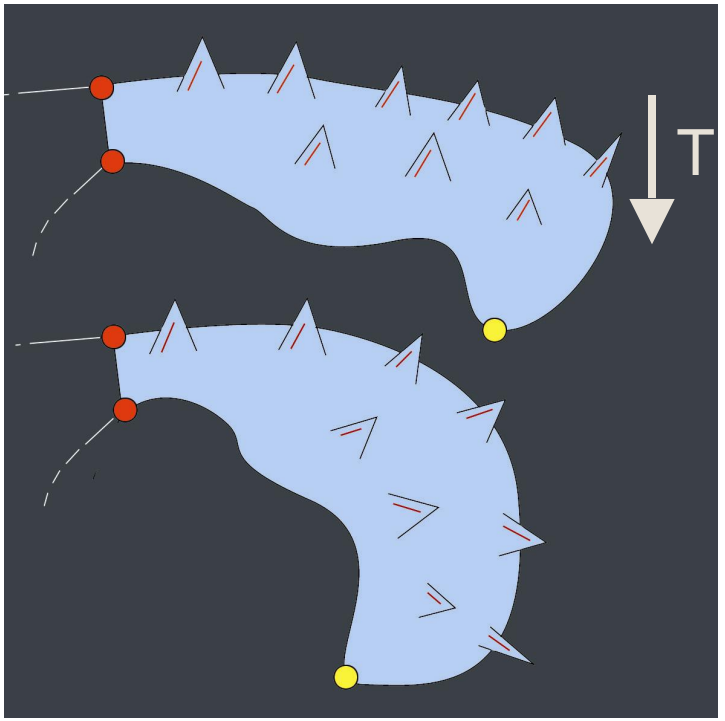
Our Goal

Edit a surface while retaining its visual appearance





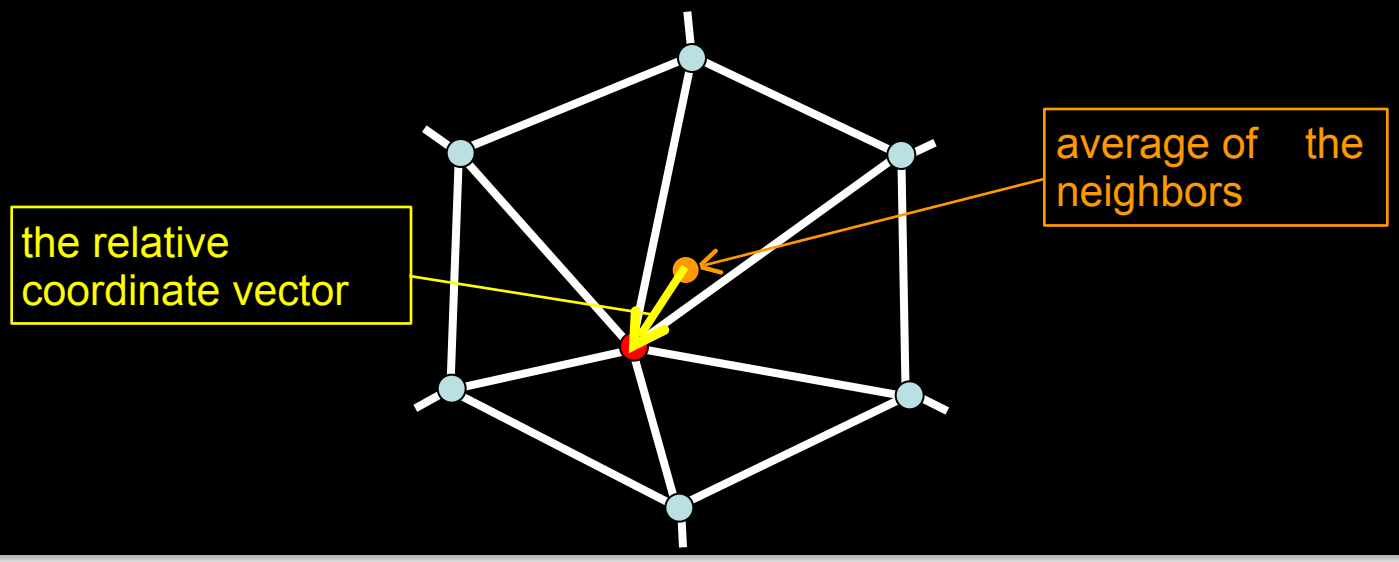
- Editing a surface while retaining its visual appearance
  - Smooth deformation
  - Smooth transition
  - Preserve relative local directions
  - of the details
  - Minimal user interaction
  - Interactive time response



Differential Coordinates

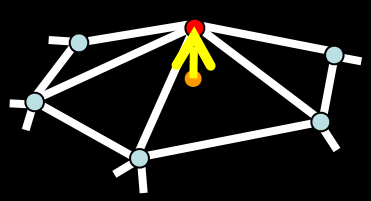
- Differential coordinates are defined for triangular mesh vertices

$$\delta_i = L(\mathbf{v}_i) = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in N(i)} \mathbf{v}_j$$

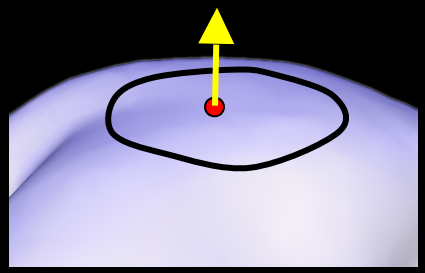


Why differential coordinates?

- They represent the local detail / local shape description
  - The direction approximates the normal
  - The size approximates the mean curvature



$$\delta_i = \frac{1}{d_i} \sum_{\mathbf{v} \in N(i)} (\mathbf{v}_i - \mathbf{v})$$



$$\frac{1}{len(\gamma)} \int_{\mathbf{v} \in \gamma} (\mathbf{v}_i - \mathbf{v}) ds$$

$$\lim_{len(\gamma) \to 0} \frac{1}{len(\gamma)} \int_{\mathbf{v} \in \gamma} (\mathbf{v}_i - \mathbf{v}) ds = H(\mathbf{v}_i)\mathbf{n}_i$$

Laplacian reconstruction

- Transforming the mesh to the differential representation:

$$\left(\delta^{(x)}, \delta^{(y)}, \delta^{(z)}\right) = M\left(P^{(x)}, P^{(y)}, P^{(z)}\right)$$

$$\left(P^{(x)}, P^{(y)}, P^{(z)}\right) = M^{-1}\left(\delta^{(x)}, \delta^{(y)}, \delta^{(z)}\right)$$

- Note that $\text{rank}(M) = n - 1$, where $n = \#V$

$$M_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & j \in \{j : (j,i) \in E\} \\ 0 & otherwise \end{cases}$$

Laplacian reconstruction

- Thus for reconstructing the mesh from the Laplacian representation:

  add constraints to get full rank system and therefore unique solution, i.e. unique minimizer to the functional

$$\left\| M \cdot P^{(x)} - \delta^{(x)} \right\|^2 + \sum_{i \in I} w_i \left(p_i^{(x)} - c_i^{(x)} \right)^2$$

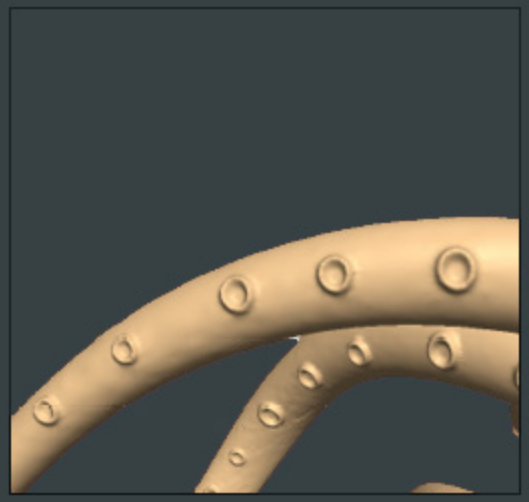
where I is the index set of constrained vertices, $w_i > 0$ are weights and c_i are the spatial constraints.

Laplacian reconstruction

The use of Laplacian (differential) representation and least squares solution forces local detail preserving

Edit a Surface While Retaining its Visual Appearance

Original surface

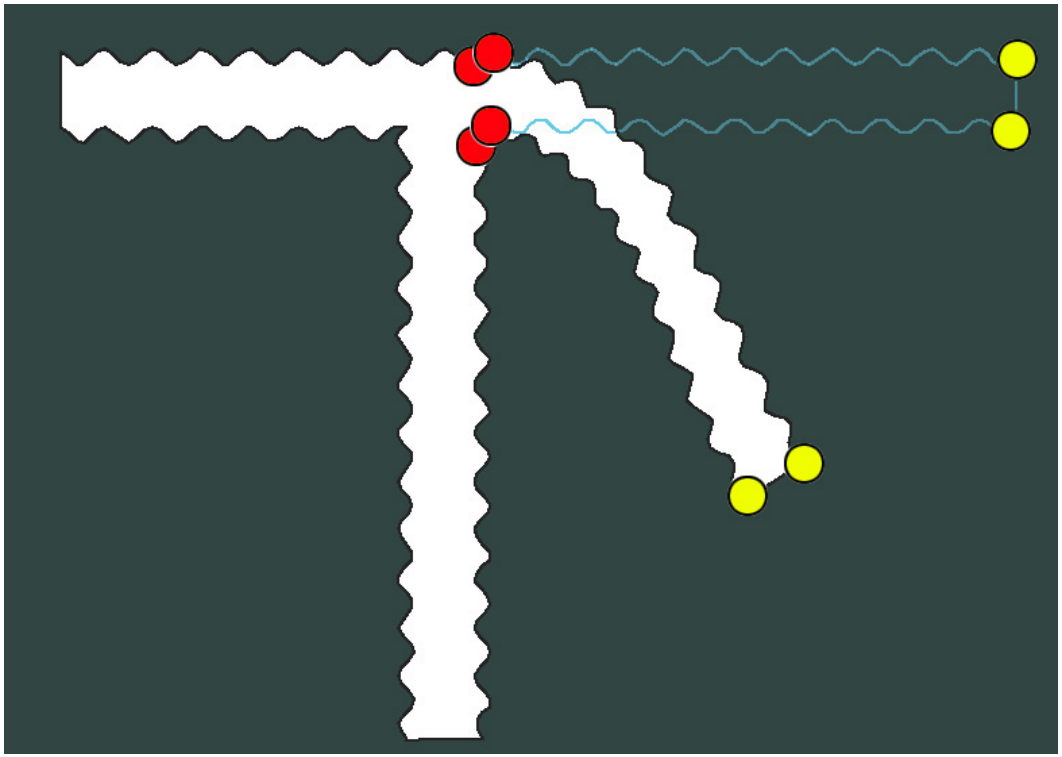


The details are deformed

The details shape is preserved

Rotated Laplacian reconstruction

- We'd like to perform deformation which preserves the detail orientation and shape:

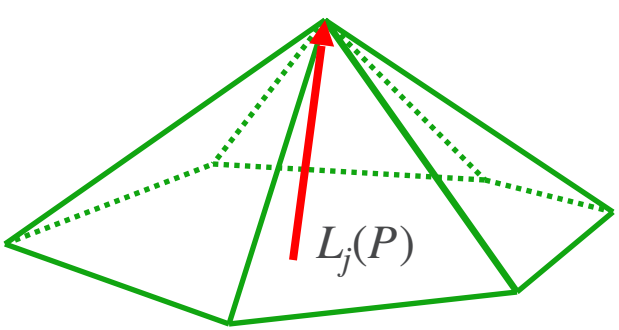


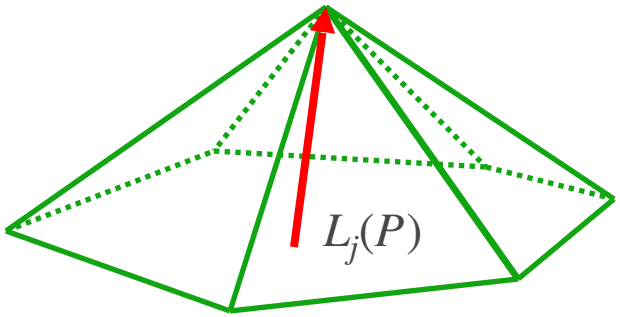
- We'd like to estimate the target shape Laplacians

Rotated Laplacian reconstruction

- The Laplacians are translation invariant:

$$L_m\left(T(P)\right) = L_m(P)$$

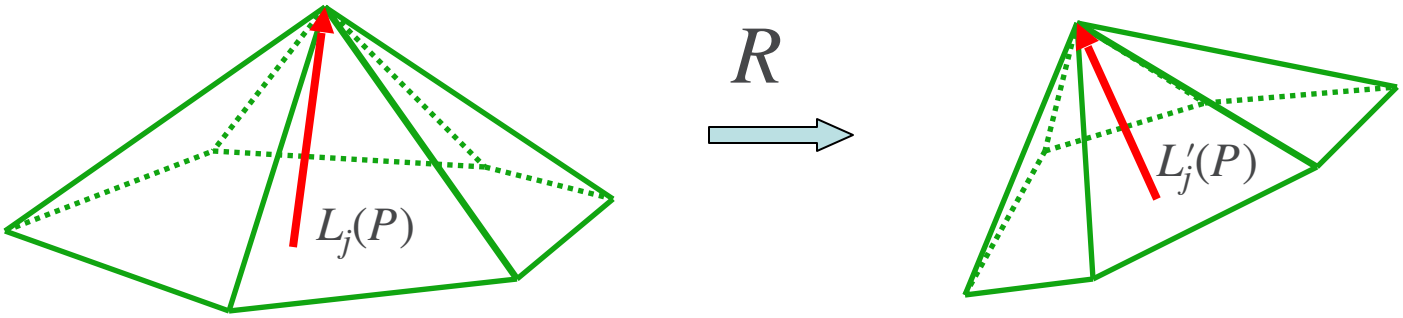




Rotated Laplacian reconstruction

- Laplacians are not rotational invariant (they represent detail with orientation)
- Note that the Laplacian operator commute with linear rotations :

$$L_m(R(P)) = R(L_m(P))$$



Rotated Laplacian reconstruction

- Therefore we get:

$$L_j(P') = L_j(A_j(P)) =$$
$$= L_j(R_j(P)) = R_j(L_j(P))$$

- So all we need is to estimate the local rotations.

Rotated Laplacian reconstruction

- In summary we have the following steps:
-

1. Reconstruct the surface with original Laplacians:

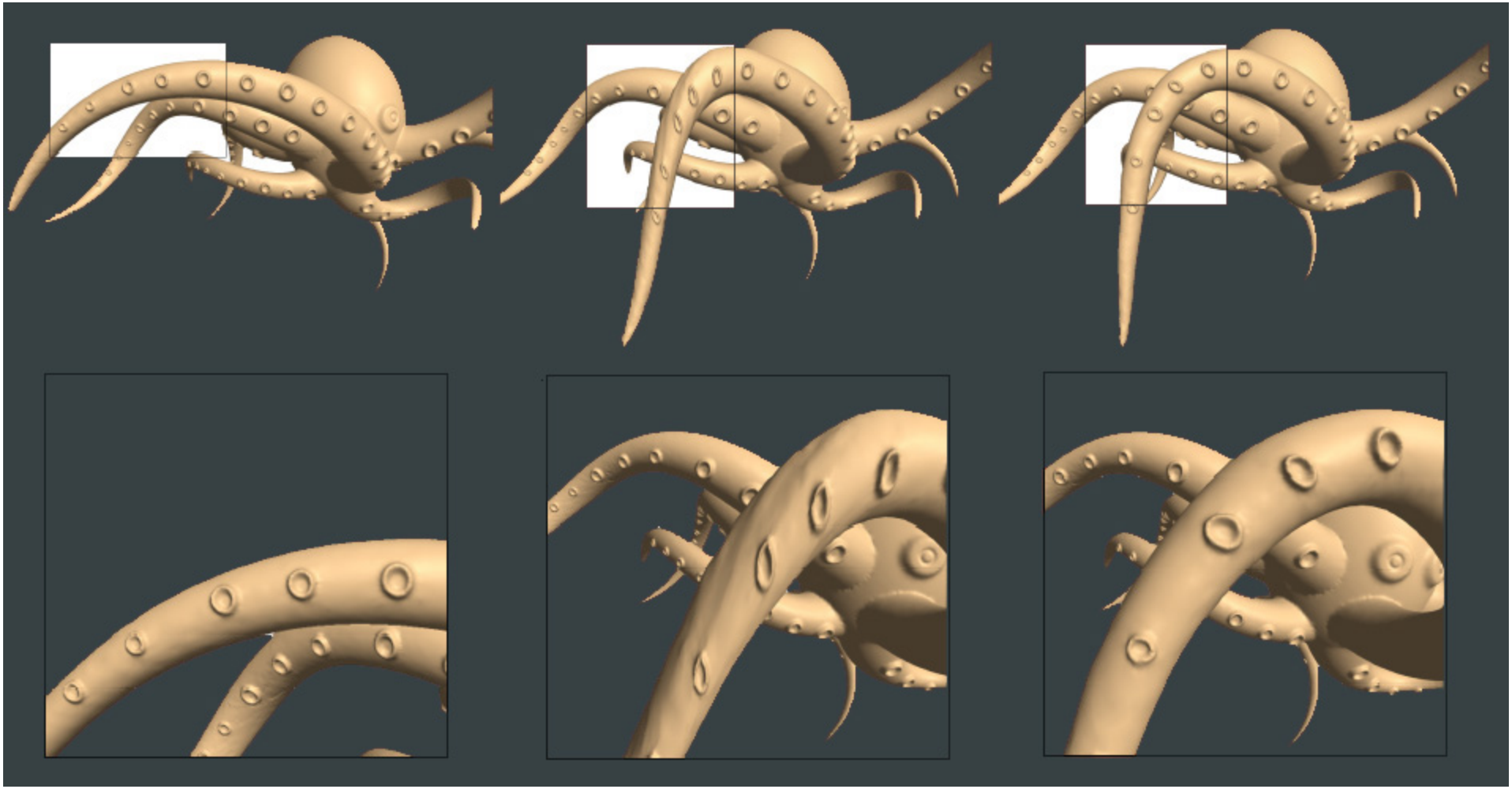
$$M^{-1}(\delta, C)$$

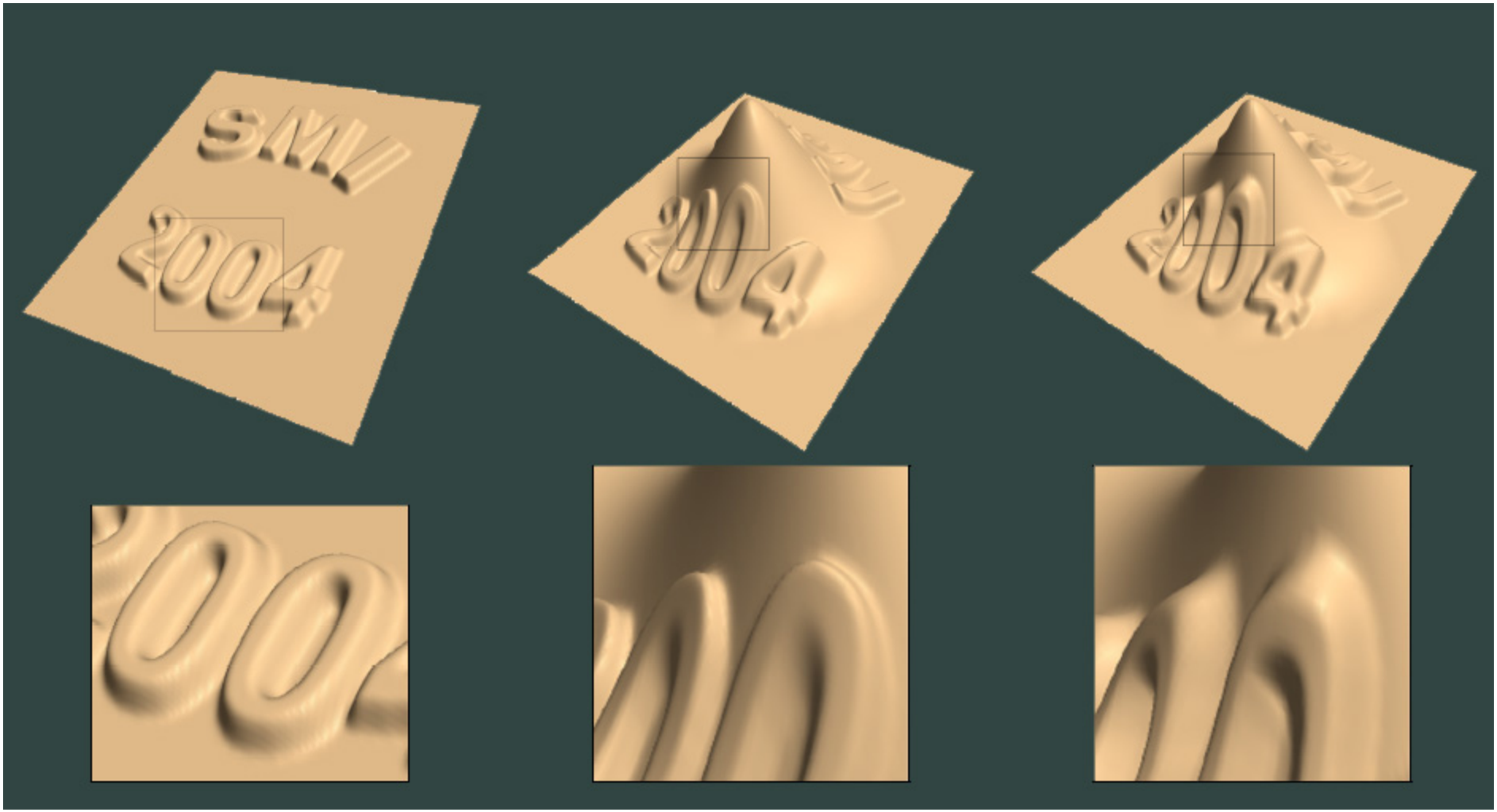
2. Approximate local rotations R_j

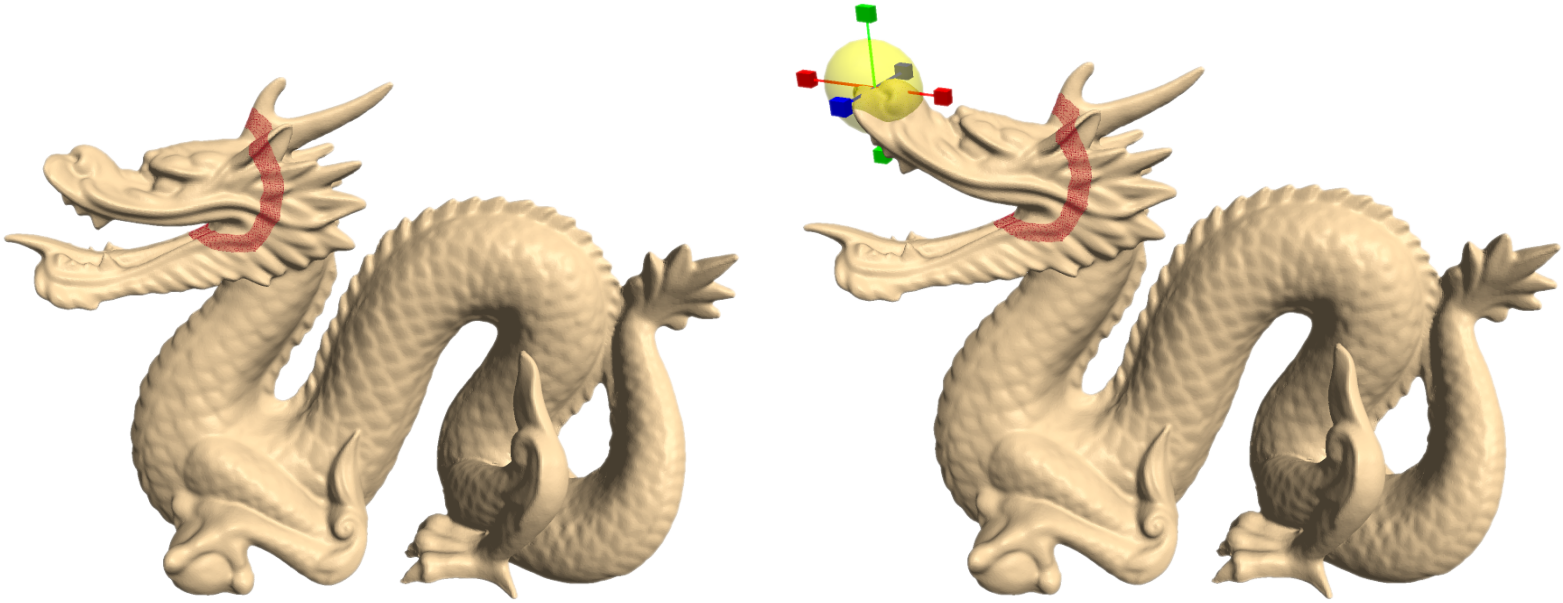
3. Rotate each Laplacian coordinate $L_j(P)$ by R_j

4. Reconstruct the edited surface:

$$M^{-1}\left[R_j(L_j(P)), C\right]$$

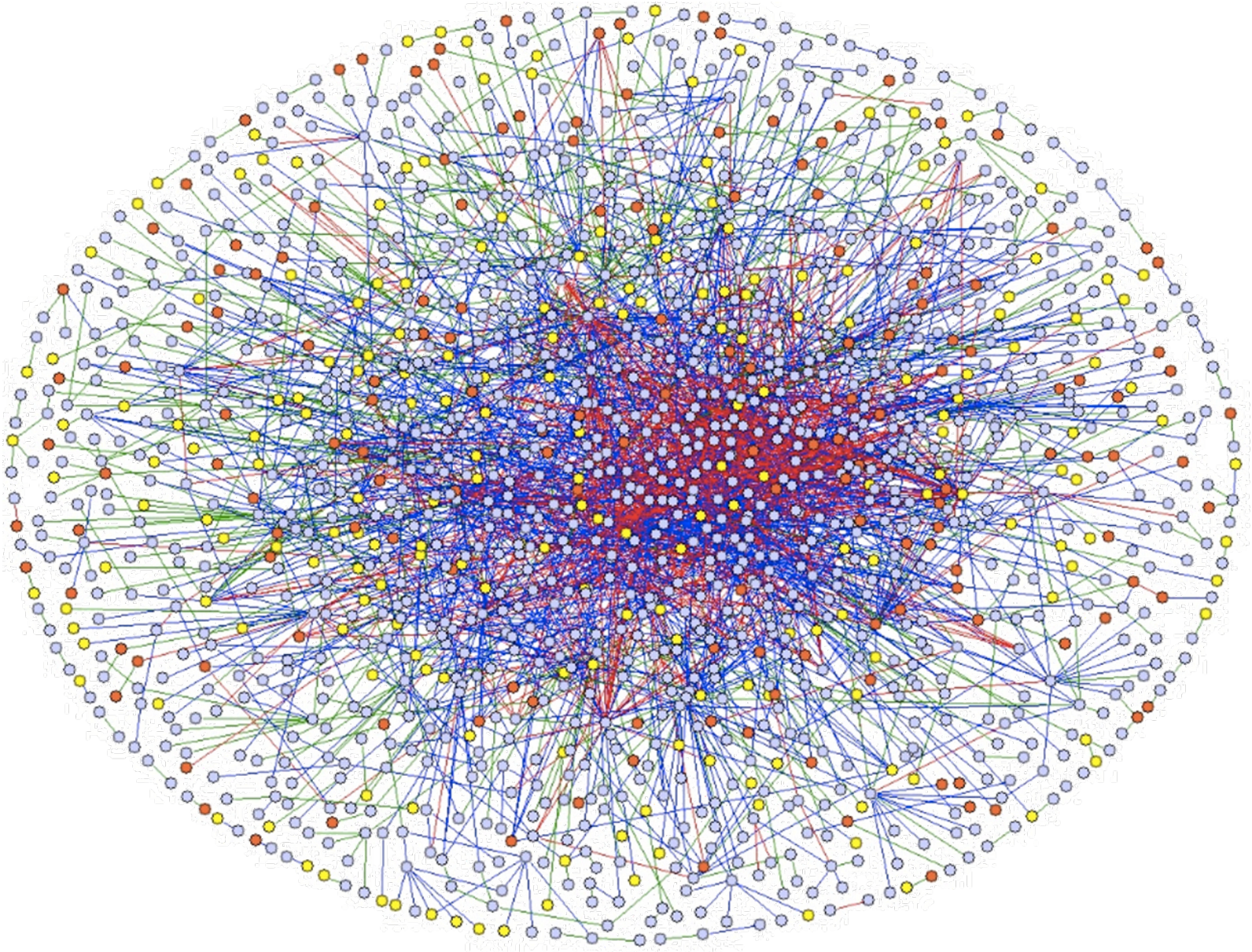



SMV
2004



GENERAL DISCRETE GRAPH LAPLACIAN (NOTATIONS)

The Graph View of Data



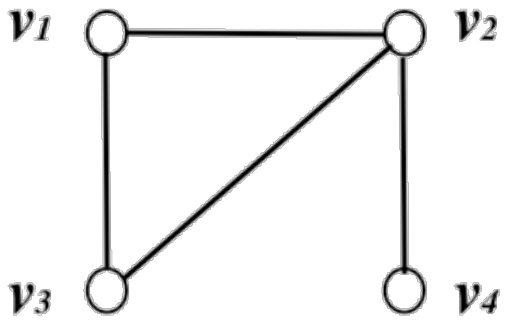
Social Networks

Adjacency Matrices

- For a graph with n vertices, the entries of the $n \times n$ adjacency matrix are defined by:

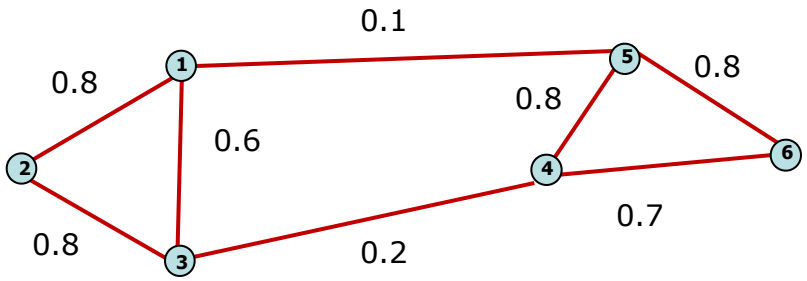
$$\mathbf{A} := \begin{cases} A_{ij} = 1 & \text{if there is an edge } e_{ij} \\ A_{ij} = 0 & \text{if there is no edge} \\ A_{ii} = 0 & \end{cases}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



Weighted Matrices

- Adjacency matrix (A)
  - n x n matrix
  - $A = [w_{ij}]$ edge weight between vertex x_i and x_j



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

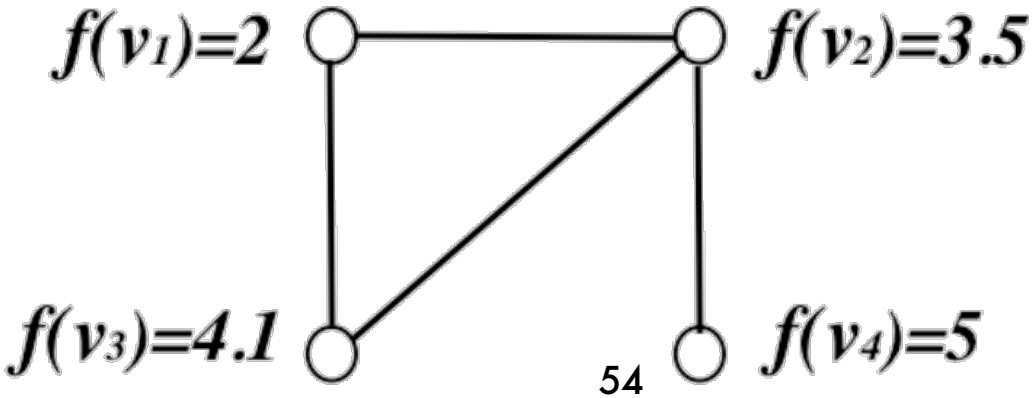
- Important properties:
  - Symmetric matrix

  ⇒ Eigenvalues are <u>real</u>

  ⇒ Eigenvector could span <u>orthogonal base</u>

Functions on Graphs

- We consider real-valued functions on the set of the graph's vertices, $\boldsymbol{f} : \mathcal{V} \longrightarrow \mathbb{R}$. Such a function assigns a real number to each graph node.
- $\boldsymbol{f}$ is a vector indexed by the graph's vertices, hence $\boldsymbol{f} \in \mathbb{R}^n$.
- Notation: $\boldsymbol{f} = (f(v_1), \ldots, f(v_n)) = (f(1), \ldots, f(n))$.
- The eigenvectors of the adjacency matrix, $\mathbf{A}\boldsymbol{x} = \lambda \boldsymbol{x}$, can be viewed as *eigenfunctions*.



Operators and Quadratic Forms

- The adjacency matrix can be viewed as an operator

$$\boldsymbol{g} = \mathbf{A}\boldsymbol{f}; g(i) = \sum_{i \sim j} f(j)$$

- It can also be viewed as a quadratic form:

$$\boldsymbol{f}^\top \mathbf{A} \boldsymbol{f} = \sum_{e_{ij}} f(i) f(j)$$

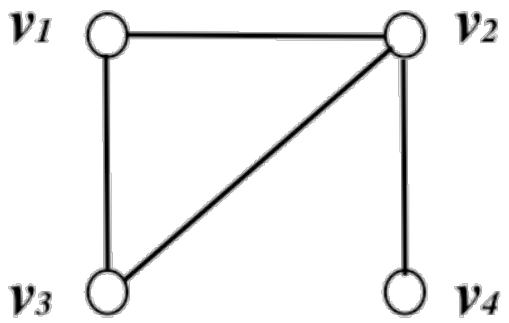
Graph (Unnormalized) Laplacian

- $\mathbf{L} = \nabla^\top \nabla$
- $(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} (f(v_i) - f(v_j))$
- Connection between the Laplacian and the adjacency matrices:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

- The degree matrix: $\mathbf{D} := D_{ii} = d(v_i)$.

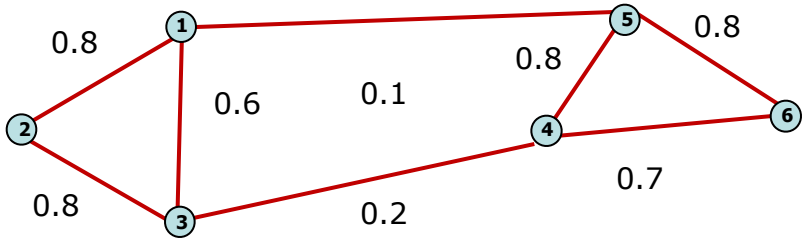
$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$



Laplacian Matrix

- **Laplacian matrix (L)**
  - n x n symmetric matrix

$$L = D - A$$



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	0.8-	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

- Important properties:
  - Eigenvalues are non-negative real numbers (Gershgorin circle theorem)
  - Eigenvectors are real and orthogonal
  - Eigenvalues and eigenvectors provide an insight into the connectivity of the graph…

Laplacian Defines Natural Quadratic Form of Graphs

$$x^T L x = \sum_{(i,j) \in E} (x(i) - x(j))^2$$

$L = D - A$ where D is diagonal matrix of degrees

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

Undirected Weighted Graphs

- We consider *undirected weighted graphs*: Each edge e_{ij} is weighted by $w_{ij} > 0$.
- The Laplacian as an operator:

$$(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} w_{ij}(f(v_i) - f(v_j))$$

- As a quadratic form:

$$\boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} = \frac{1}{2} \sum_{e_{ij}} w_{ij}(f(v_i) - f(v_j))^2$$

- $\mathbf{L}$ is symmetric and positive semi-definite.
- $\mathbf{L}$ has n non-negative, real-valued eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

GENERAL DISCRETE GRAPH LAPLACIAN (SOME PROPERTIES)

Connected Graph Laplacians

- $\mathbf{L}\boldsymbol{u} = \lambda \boldsymbol{u}$.
- $\mathbf{L}\mathbf{1}_n = \mathbf{0}$, $\lambda_1 = 0$ is the smallest eigenvalue.
- The *one* vector: $\mathbf{1}_n = (1 \dots 1)^\top$.
- $0 = \boldsymbol{u}^\top \mathbf{L} \boldsymbol{u} = \sum_{i,j=1}^{n} w_{ij}(u(i) - u(j))^2$.
- If any two vertices are connected by a path, then $\boldsymbol{u} = (u(1), \dots, u(n))$ needs to be constant at all vertices such that the quadratic form vanishes. Therefore, a graph with one connected component has the constant vector $\boldsymbol{u}_1 = \mathbf{1}_n$ as the only eigenvector with eigenvalue 0.

A Graph with k Connected Components

- Each connected component has an associated Laplacian. Therefore, we can write matrix $\mathbf{L}$ as a *block diagonal matrix*:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & \\ & \ddots & \\ & & \mathbf{L}_k \end{bmatrix}$$

- The spectrum of $\mathbf{L}$ is given by the union of the spectra of $\mathbf{L}_i$.
- Each block corresponds to a connected component, hence each matrix $\mathbf{L}_i$ has an eigenvalue 0 with multiplicity 1.
- The spectrum of $\mathbf{L}$ is given by the union of the spectra of $\mathbf{L}_i$.
- The eigenvalue $\lambda_1 = 0$ has multiplicity k.

The Eigenspace of λ1 = 0

- The eigenspace corresponding to $\lambda_1 = \ldots = \lambda_k = 0$ is spanned by the k mutually orthogonal vectors:

$$\boldsymbol{u}_1 = \mathbf{1}_{L_1}$$
$$\ldots$$
$$\boldsymbol{u}_k = \mathbf{1}_{L_k}$$

- with $\mathbf{1}_{L_i} = (0000111110000)^\top \in \mathbb{R}^n$
- These vectors are the *indicator vectors* of the graph's connected components.
- Notice that $\mathbf{1}_{L_1} + \ldots + \mathbf{1}_{L_k} = \mathbf{1}_n$

The Fiedler Vector

- The first non-null eigenvalue λ_{k+1} is called the Fiedler value.
- The corresponding eigenvector $\boldsymbol{u}_{k+1}$ is called the Fiedler vector.
- The multiplicity of the Fiedler eigenvalue is always equal to 1.
- The Fiedler value is the *algebraic connectivity of a graph*, the further from 0, the more connected.
- The Fidler vector has been extensively used for *spectral bi-partioning*
- Theoretical results are summarized in Spielman & Teng 2007: `http://cs-www.cs.yale.edu/homes/spielman/`

Laplacian Eigenvectors for Connected Graphs

- $\boldsymbol{u}_1 = \mathbf{1}_n, \mathbf{L}\mathbf{1}_n = \mathbf{0}$.
- $\boldsymbol{u}_2$ is the *the Fiedler vector* with multiplicity 1.
- The eigenvectors form an orthonormal basis: $\boldsymbol{u}_i^\top \boldsymbol{u}_j = \delta_{ij}$.
- For any eigenvector $\boldsymbol{u}_i = (\boldsymbol{u}_i(v_1) \dots \boldsymbol{u}_i(v_n))^\top$, $2 \leq i \leq n$:

$$\boldsymbol{u}_i^\top \mathbf{1}_n = 0$$

- Hence the components of $\boldsymbol{u}_i$, $2 \leq i \leq n$ satisfy:

$$\sum_{j=1}^{n} \boldsymbol{u}_i(v_j) = 0$$

- Each component is bounded by:

$$-1 < \boldsymbol{u}_i(v_j) < 1$$

λ_2 = algebraic connectivity, monotone under graph inclusion

Some Special Graphs

- The complete graph on n vertices, K_n, which has edge set $\{(u, v) : u \neq v\}$.
- The star graph on n vertices, S_n, which has edge set $\{(1, u) : 2 \leq u \leq n\}$.
- The hypercube

The **hypercube** on 2^k vertices. The vertices are elements of $\{0,1\}^k$. Edges exist between vertices that differ in only one coordinate.

Complete Graph

Lemma 2.5.1. *The Laplacian of K_n has eigenvalue 0 with multiplicity 1 and n with multiplicity $n-1$.*

Proof. To compute the non-zero eigenvalues, let $\boldsymbol{\psi}$ be any non-zero vector orthogonal to the all-1s vector, so

$$\sum_u \boldsymbol{\psi}(u) = 0. \tag{2.6}$$

We now compute the first coordinate of $\boldsymbol{L}_{K_n}\boldsymbol{\psi}$. Using (2.3), we find

$$\left(\boldsymbol{L}_{K_n}\boldsymbol{\psi}\right)(1) = \sum_{v\geq 2}(\boldsymbol{\psi}(1) - \boldsymbol{\psi}(v)) = (n-1)\boldsymbol{\psi}(1) - \sum_{v=2}^{n}\boldsymbol{\psi}(v) = n\boldsymbol{\psi}(1), \quad \text{by (2.6)}.$$

As the choice of coordinate was arbitrary, we have $\boldsymbol{L}\boldsymbol{\psi} = n\boldsymbol{\psi}$. So, every vector orthogonal to the all-1s vector is an eigenvector of eigenvalue n. □

Alternative approach. Observe that $\boldsymbol{L}_{K_n} = n\boldsymbol{I} - \mathbf{1}\mathbf{1}^T$. □

Star Graph

Lemma 2.5.2. *Let $G = (V, E)$ be a graph, and let v and w be vertices of degree one that are both connected to another vertex z. Then, the vector $\boldsymbol{\psi} = \boldsymbol{\delta}_v - \boldsymbol{\delta}_w$ is an eigenvector of $\boldsymbol{L}_G$ of eigenvalue 1.*

Proof. Just multiply $\boldsymbol{L}_G$ by $\boldsymbol{\psi}$, and check vertex-by-vertex that it equals $\boldsymbol{\psi}$. □

As eigenvectors of different eigenvalues are orthogonal, this implies that $\boldsymbol{\psi}(u) = \boldsymbol{\psi}(v)$ for every eigenvector with eigenvalue different from 1.

Lemma 2.5.3. *The graph S_n has eigenvalue 0 with multiplicity 1, eigenvalue 1 with multiplicity $n - 2$, and eigenvalue n with multiplicity 1.*

Hypercube Graph

- Exercise

LAPLACIAN GRAPH EMBEDDING

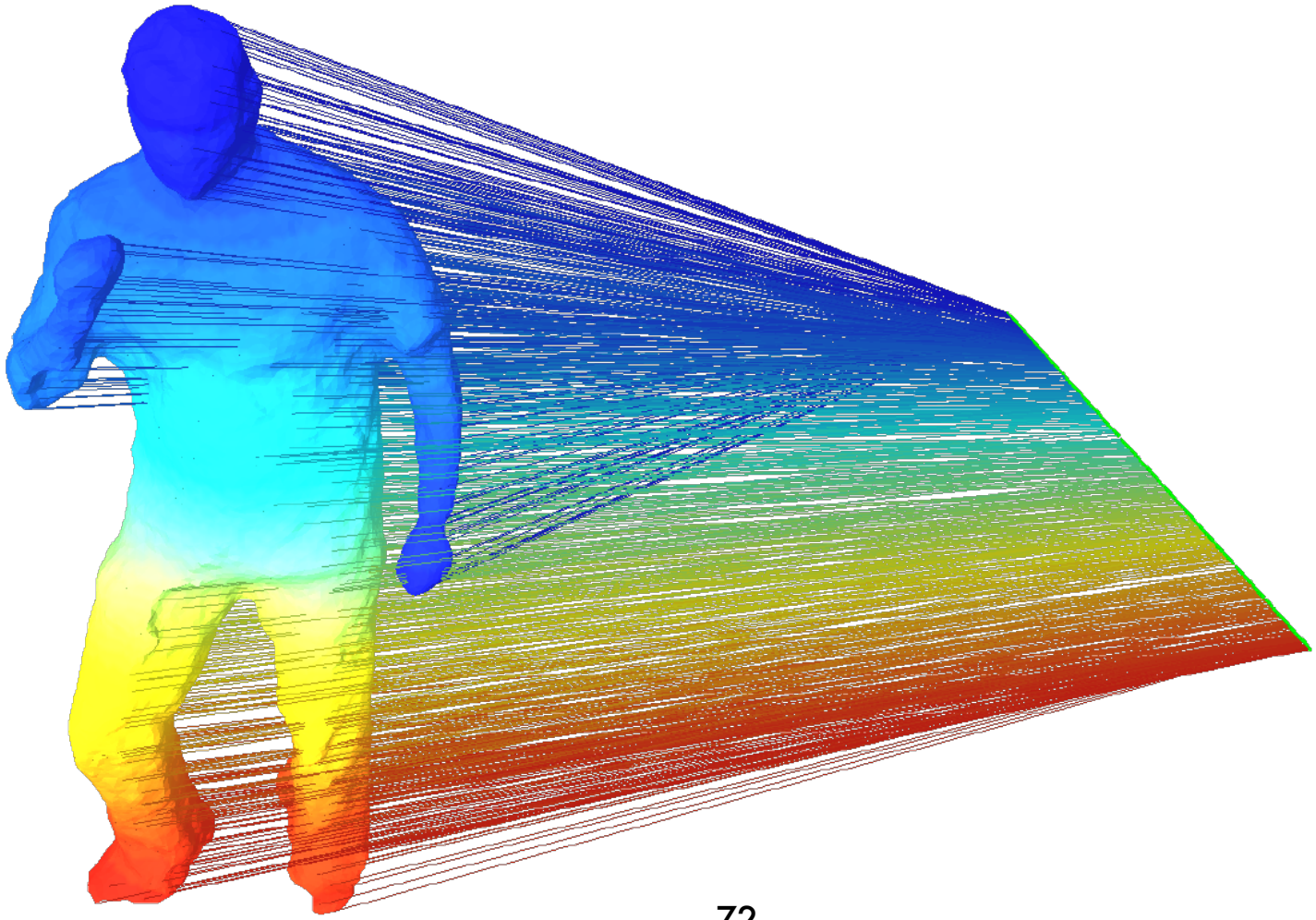
1-d Laplacian Embedding

- Map a weighted graph onto a line such that connected nodes stay as close as possible, i.e., minimize $\sum_{i,j=1}^{n} w_{ij}(f(v_i) - f(v_j))^2$, or:

$$\arg\min_{\boldsymbol{f}} \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} \text{ with: } \boldsymbol{f}^\top \boldsymbol{f} = 1 \text{ and } \boldsymbol{f}^\top \mathbf{1} = 0$$

- The solution is the eigenvector associated with the smallest nonzero eigenvalue of the eigenvalue problem: $\mathbf{L}\boldsymbol{f} = \lambda \boldsymbol{f}$, namely the Fiedler vector $\boldsymbol{u}_2$.
- For more details on this minimization see Golub & Van Loan *Matrix Computations*, chapter 8 (The symmetric eigenvalue problem).

1-d Embedding Example



Higher-d Embeddings

- Embed the graph in a k-dimensional Euclidean space. The embedding is given by the $n \times k$ matrix $\mathbf{F} = [\boldsymbol{f}_1 \boldsymbol{f}_2 \ldots \boldsymbol{f}_k]$ where the i-th row of this matrix – $\boldsymbol{f}^{(i)}$ – corresponds to the Euclidean coordinates of the i-th graph node v_i.
- We need to minimize (Belkin & Niyogi '03):

$$\arg \min_{\boldsymbol{f}_1 \cdots \boldsymbol{f}_k} \sum_{i,j=1}^{n} w_{ij} \|\boldsymbol{f}^{(i)} - \boldsymbol{f}^{(j)}\|^2 \text{ with: } \mathbf{F}^\top \mathbf{F} = \mathbf{I}.$$

- The solution is provided by the matrix of eigenvectors corresponding to the k lowest nonzero eigenvalues of the eigenvalue problem $\mathbf{L}\boldsymbol{f} = \lambda \boldsymbol{f}$.

2-d Embeddings