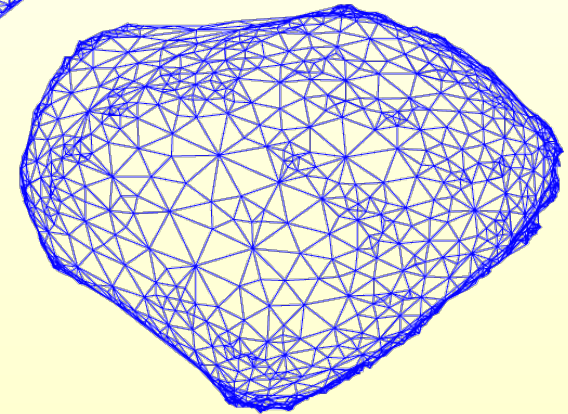
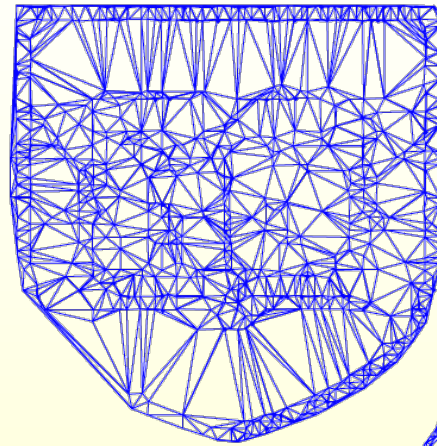


# CS233: Geometric and Topological Data Analysis

---

Graph Laplacians,  
Laplacian Embeddings,  
and Spectral Clustering

16 April 2018



Slides ack: Radu Horaud, Dan Spielman  
<http://perception.inrialpes.fr/>  
<http://www.cs.yale.edu/homes/spielman/561/>

**Last Time: CCA**

# CCA vs PCA

- ◆ For centered variates, the covariance is  $C_{xy} = x^T y$
- ◆ Canonical correlation analysis attempts to answer the question “**which directions account for most of the covariance between the two data sets?**” The goal is to find directions  $w_x, w_y$  so as to maximize

$$w_x^T C_{xy} w_y = (xw_x)^T (yw_y)$$

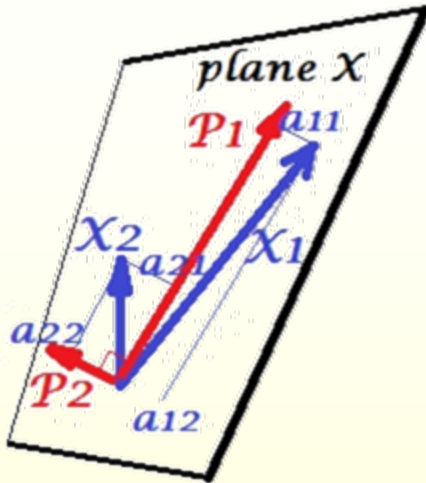
- ◆ subject to  $\|xw_x\| = 1, \|yw_y\| = 1$

- ◆ In PCA we have a single variate and seek the direction that “**maximizes the variance in the data**”

$$w_x^T C_{xx} w_x = (xw_x)^T (xw_x)$$

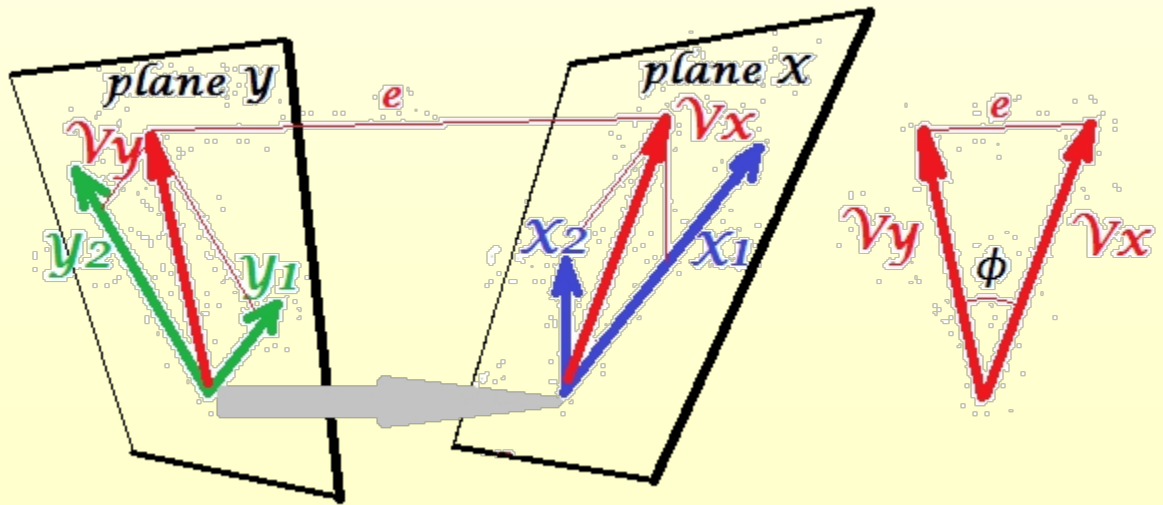
- ◆ subject to  $\|w_x\| = 1$

# CCA vs PCA



PCA

CCA



# CCA (too) Becomes an Eigenvalue Problem

- ◆ A generalized eigenvalue problem

$$C_{xy}C_{yy}^{-1}C_{yx}w_x = \lambda^2 C_{xx}w_x$$
$$Ax = \lambda Bx$$

- ◆ Can symmetrically also get

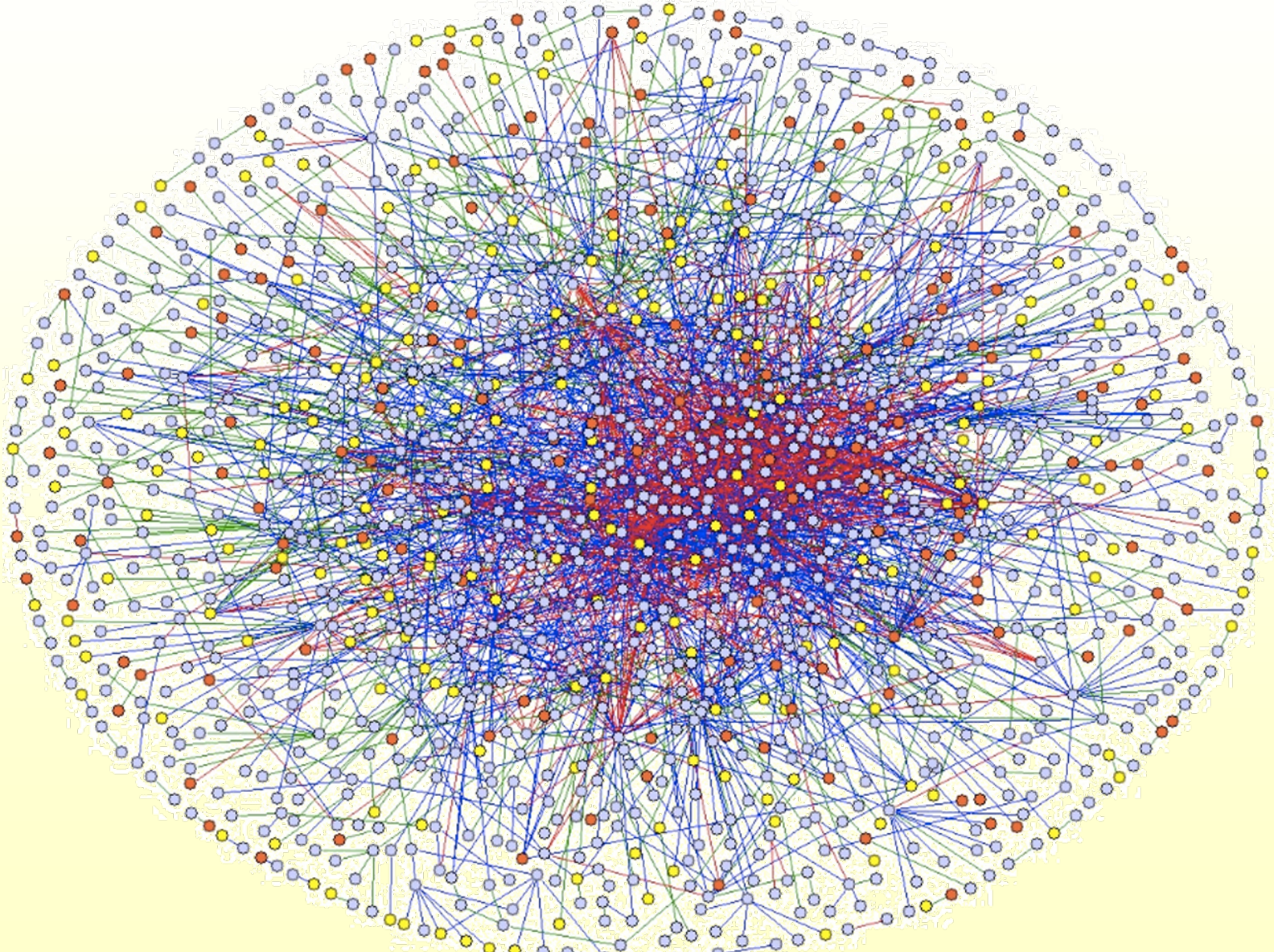
$$C_{yx}C_{xx}^{-1}C_{xy}w_y = \lambda^2 C_{yy}w_y$$

- ◆ One can go back and forth between  $w_x$  and  $w_y$

$$w_y = \frac{C_{yy}^{-1}C_{yx}w_x}{\lambda}$$

# The Graph View of Data

# The Graph View of Data



# Social Networks



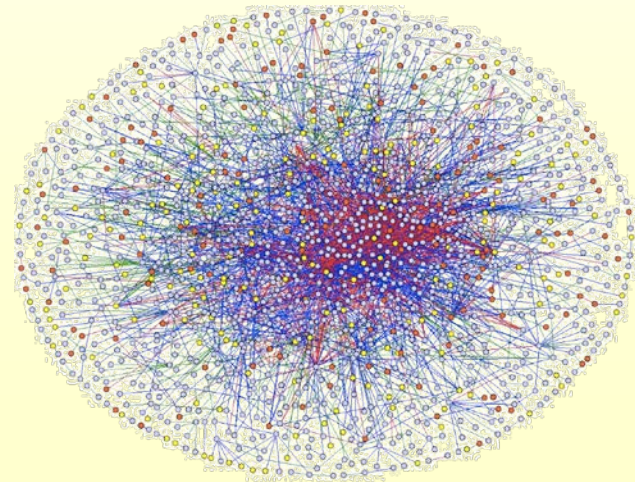
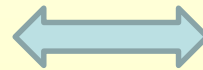
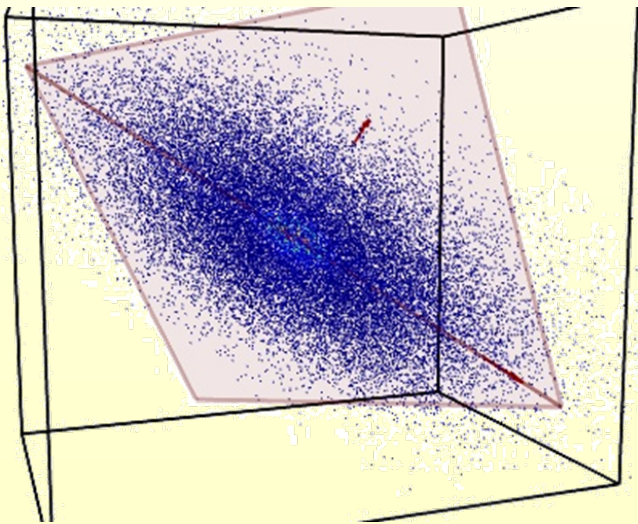
# Connect Points in $R^d$ and Graph Views of Data

- ◆ Points in  $R^d$

- ◆ via near-neighbor graphs

- ◆ Graph

- ◆ via matrix representations of graphs



# Spectral Graph Theory

# Spectral Graph Theory

- The *spectral graph theory* studies the properties of graphs via the eigenvalues and eigenvectors of their associated graph matrices: the *adjacency matrix* and the *graph Laplacian* and its variants.
- Both matrices have been extremely well studied from an algebraic point of view.
- The Laplacian allows a natural link between discrete representations, such as graphs, and continuous representations, such as vector spaces and manifolds.
- The most important application of the Laplacian is *spectral clustering* that corresponds to a computationally tractable solution to the *graph partitioning problem*.
- Another application is *spectral matching* that solves for *graph matching*.

# More Applications

- *Spectral partitioning*: automatic circuit placement for VLSI (Alpert et al 1999), image segmentation (Shi & Malik 2000),
- *Text mining and web applications*: document classification based on semantic association of words (Lafon & Lee 2006), collaborative recommendation (Fouss et al. 2007), text categorization based on reader similarity (Kamvar et al. 2003).
- *Manifold analysis*: Manifold embedding, manifold learning, mesh segmentation, etc.

# Graph Notations and Definitions

We consider *simple graphs* (no multiple edges or loops),  
 $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ :

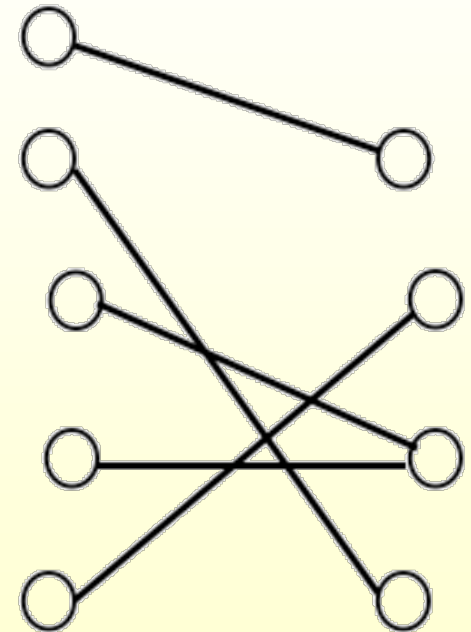
- $\mathcal{V}(\mathcal{G}) = \{v_1, \dots, v_n\}$  is called the *vertex set* with  $n = |\mathcal{V}|$ ;
- $\mathcal{E}(\mathcal{G}) = \{e_{ij}\}$  is called the *edge set* with  $m = |\mathcal{E}|$ ;
- An edge  $e_{ij}$  connects vertices  $v_i$  and  $v_j$  if they are adjacent or neighbors. One possible notation for adjacency is  $v_i \sim v_j$ ;
- The number of neighbors of a node  $v$  is called the *degree* of  $v$  and is denoted by  $d(v)$ ,  $d(v_i) = \sum_{v_i \sim v_j} e_{ij}$ . If all the nodes of a graph have the same degree, the graph is *regular*; The nodes of an *Eulerian* graph have even degree.
- A graph is *complete* if there is an edge between every pair of vertices.

# Subgraphs

- $\mathcal{H}$  is a *subgraph* of  $\mathcal{G}$  if  $\mathcal{V}(\mathcal{H}) \subseteq \mathcal{V}(\mathcal{G})$  and  $\mathcal{E}(\mathcal{H}) \subseteq \mathcal{E}(\mathcal{G})$ ;
- a subgraph  $\mathcal{H}$  is an *induced subgraph* of  $\mathcal{G}$  if two vertices of  $\mathcal{V}(\mathcal{H})$  are adjacent if and only if they are adjacent in  $\mathcal{G}$ .
- A *clique* is a complete subgraph of a graph.
- A *path* of  $k$  vertices is a sequence of  $k$  distinct vertices such that consecutive vertices are adjacent.
- A *cycle* is a connected subgraph where every vertex has exactly two neighbors.
- A graph containing no cycles is a *forest*. A connected forest is a *tree*.

# $k$ -Partite Graphs

- A graph is called  *$k$ -partite* if its set of vertices admits a partition into  $k$  classes such that the vertices of the same class are not adjacent.
- An example of a *bipartite* graph.

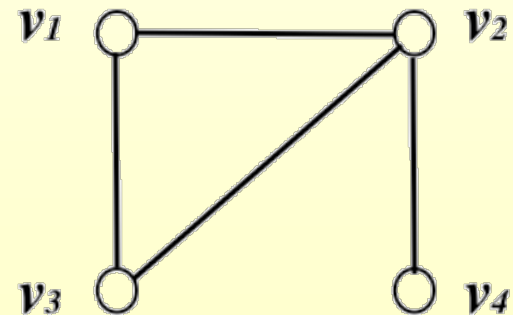


# Adjacency Matrices

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

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

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

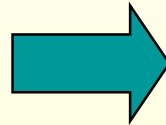
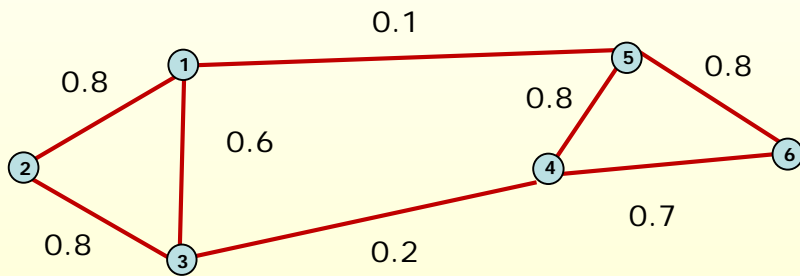


# Weighted Matrices

- ◆ Adjacency matrix ( $A$ )

- ◆  $n \times n$  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 real

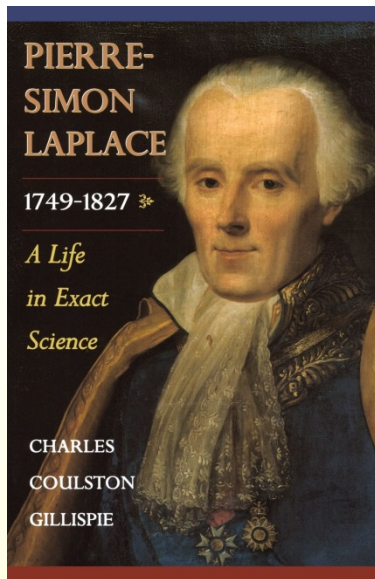
- ⇒ Eigenvectors span orthogonal basis

# Eigenvalues and Eigenvectors

- $\mathbf{A}$  is a real-symmetric matrix: it has  $n$  real eigenvalues and its  $n$  real eigenvectors form an orthonormal basis.
- Let  $\{\lambda_1, \dots, \lambda_i, \dots, \lambda_r\}$  be the set of *distinct* eigenvalues.
- The eigenspace  $S_i$  contains the eigenvectors associated with  $\lambda_i$ :

$$S_i = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \lambda_i\mathbf{x}\}$$

- For real-symmetric matrices, the algebraic multiplicity is equal to the geometric multiplicity, for all the eigenvalues.
- The dimension of  $S_i$  (geometric multiplicity) is equal to the multiplicity of  $\lambda_i$ .
- If  $\lambda_i \neq \lambda_j$  then  $S_i$  and  $S_j$  are mutually orthogonal.



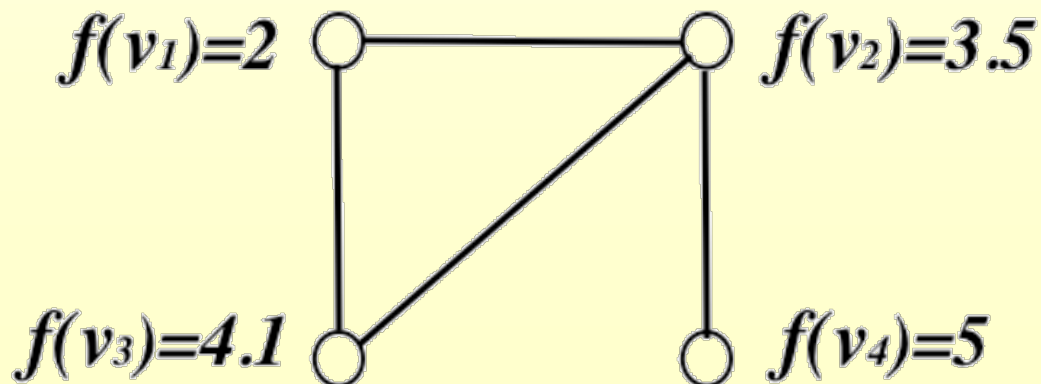
# The Laplacian



$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

# Functions on Graphs

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



# Operators and Quadratic Forms

- The adjacency matrix can be viewed as an operator

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

- It can also be viewed as a quadratic form:

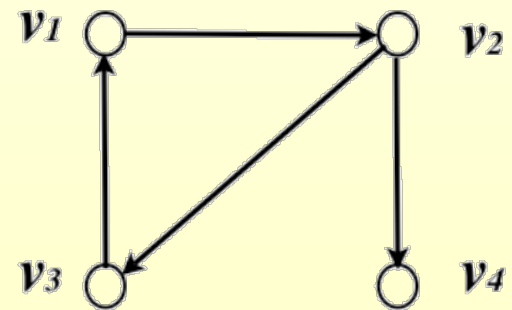
$$\mathbf{f}^\top \mathbf{A} \mathbf{f} = \sum_{e_{ij}} f(i)f(j)$$

# Incidence Matrix

- Let each edge in the graph have an arbitrary but fixed orientation;
- The incidence matrix of a graph is a  $|\mathcal{E}| \times |\mathcal{V}|$  ( $m \times n$ ) matrix defined as follows:

$$\nabla := \begin{cases} \nabla_{ev} = -1 & \text{if } v \text{ is the initial vertex of edge } e \\ \nabla_{ev} = 1 & \text{if } v \text{ is the terminal vertex of edge } e \\ \nabla_{ev} = 0 & \text{if } v \text{ is not in } e \end{cases}$$

$$\nabla = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & +1 \end{bmatrix}$$



# Discrete Differential Operator

- The mapping  $f \longrightarrow \nabla f$  is known as the *co-boundary mapping* of the graph.
- $(\nabla f)(e_{ij}) = f(v_j) - f(v_i)$

$$\begin{pmatrix} f(2) - f(1) \\ f(1) - f(3) \\ f(3) - f(2) \\ f(4) - f(2) \end{pmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & +1 \end{bmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \end{pmatrix}$$

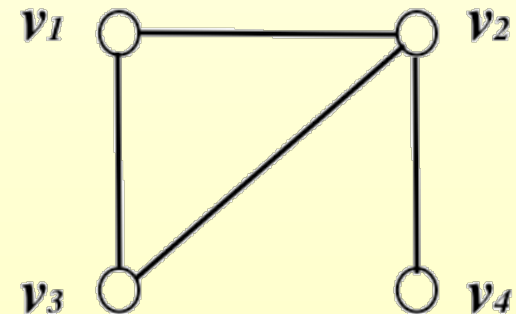
# Graph (Unnormalized) Laplacian

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

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

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

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

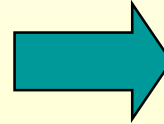
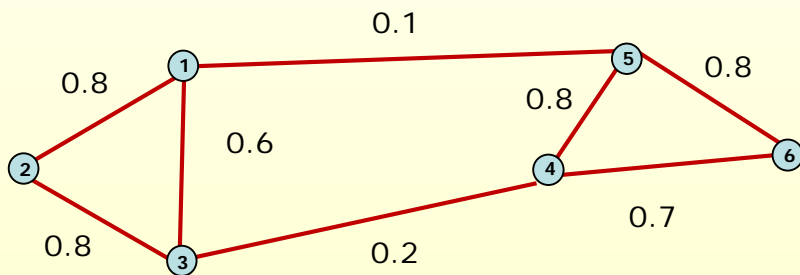


# Degree Matrix

- ◆ Degree matrix ( $D$ )

- ◆  $n \times n$  diagonal matrix

- ◆  $D(i,i) = \sum_j w_{ij}$  : total weight of edges incident to vertex  $x_i$



	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
$x_1$	1.5	0	0	0	0	0
$x_2$	0	1.6	0	0	0	0
$x_3$	0	0	1.6	0	0	0
$x_4$	0	0	0	1.7	0	0
$x_5$	0	0	0	0	1.7	0
$x_6$	0	0	0	0	0	1.5

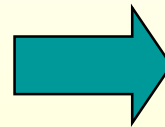
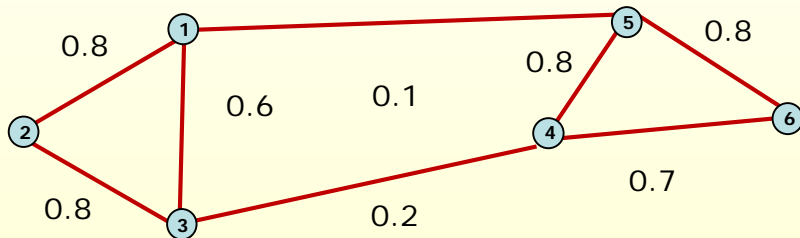
- Important application:
  - Normalize adjacency matrix

# Laplacian Matrix

$$L = D - A$$

## ◆ Laplacian matrix ( $L$ )

◆  $n \times n$  symmetric matrix



	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
$x_1$	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
- Eigenvectors are real and orthogonal
- Eigenvalues and eigenvectors provide insight into the connectivity of the graph...

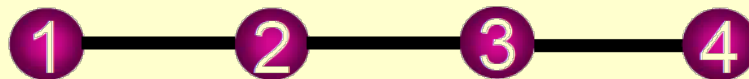
# Laplacian Defines Natural Quadratic Form of Graphs

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

A measure of continuity of  $x$  over the graph

$L = D - A$  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:

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

- As a quadratic form:

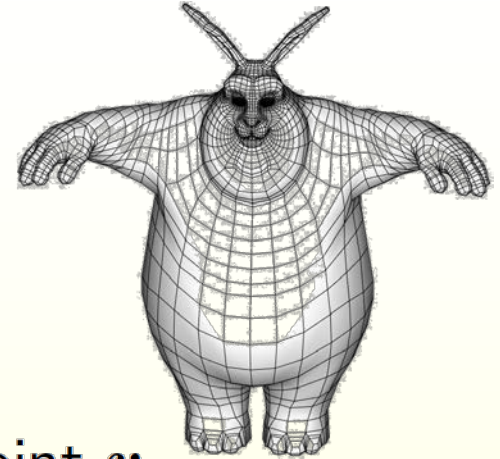
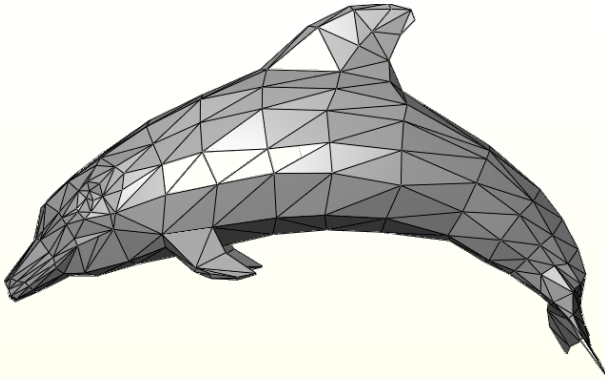
$$\mathbf{f}^\top \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{e_{ij}} w_{ij} (f(v_i) - f(v_j))^2$$

- $\mathbf{L}$  is symmetric and positive semi-definite.
- $\mathbf{L}$  has  $n$  non-negative, real-valued eigenvalues:

$$0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n.$$

← 0 always an eigenvalue

# Discrete Surface Laplacians: 3D Meshes



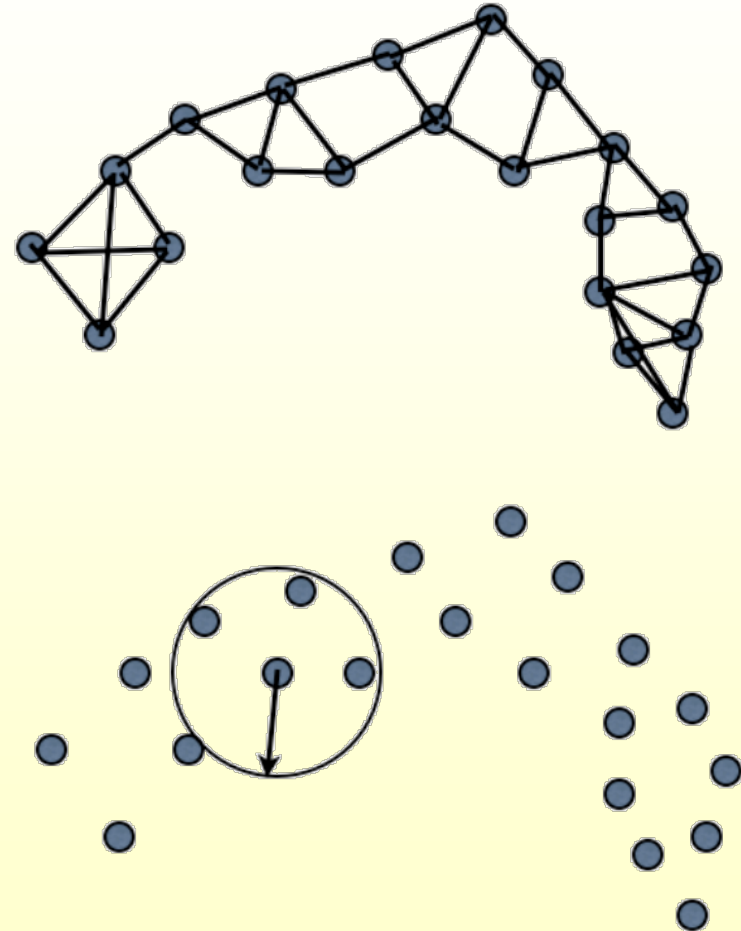
- A graph vertex  $v_i$  is associated with a 3D point  $\mathbf{v}_i$ .
- The weight of an edge  $e_{ij}$  is defined by the Gaussian kernel:

$$w_{ij} = \exp\left(-\|\mathbf{v}_i - \mathbf{v}_j\|^2 / \sigma^2\right)$$

- $0 \leq w_{\min} \leq w_{ij} \leq w_{\max} \leq 1$
- Hence, the geometric structure of the mesh is encoded in the weights.
- Other weighting functions were proposed in the literature.

# Point Cloud Laplacians

- 3-nearest neighbor graph
- $\epsilon$ -radius graph
- KNN may guarantee that the graph is connected (depends on the implementation)
- $\epsilon$ -radius does not guarantee that the graph has one connected component



# Connected Graph Laplacians

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

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

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{1}_{L_1} \\ &\dots \\ \mathbf{u}_k &= \mathbf{1}_{L_k} \end{aligned}$$

- 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  $\mathbf{1}_{L_1} + \dots + \mathbf{1}_{L_k} = \mathbf{1}_n$

# Courant/Fischer Definition of Eigenvectors / Eigenvalues

$$\lambda_1 = \min_{x \neq 0} \frac{x^T L x}{x^T x} \quad v_1 = \arg \min_{x \neq 0} \frac{x^T L x}{x^T x}$$

$$\lambda_2 = \min_{x \perp v_1} \frac{x^T L x}{x^T x} \quad v_2 = \arg \min_{x \perp v_1} \frac{x^T L x}{x^T x}$$

(here  $v_1 = \mathbf{1}$ )

$$\lambda_k = \min_{S \text{ of dim } k} \max_{x \in S} \frac{x^T L x}{x^T x}$$

$$v_k = \arg \min_{x \perp v_1, \dots, v_{k-1}} \frac{x^T L x}{x^T x}$$

# The Fiedler Vector

- The first non-null eigenvalue  $\lambda_{k+1}$  is called the Fiedler value.
- The corresponding eigenvector  $\mathbf{u}_{k+1}$  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 Fiedler vector has been extensively used for *spectral bi-partitioning*
- Theoretical results are summarized in Spielman & Teng 2007:  
<http://cs-www.cs.yale.edu/homes/spielman/>

# Laplacian Eigenvectors for Connected Graphs

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

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

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

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

- Each component is bounded by:

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

$\lambda_2$  = algebraic connectivity,  
monotone under graph  
inclusion

The larger  $\lambda_2$ , the more  
connected the graph.

# 1-d Laplacian Embedding

- Map a weighted graph onto a line such that connected nodes stay as close as possible, i.e., minimize

$$\sum_{i,j=1}^n w_{ij} (f(v_i) - f(v_j))^2, \text{ or:}$$

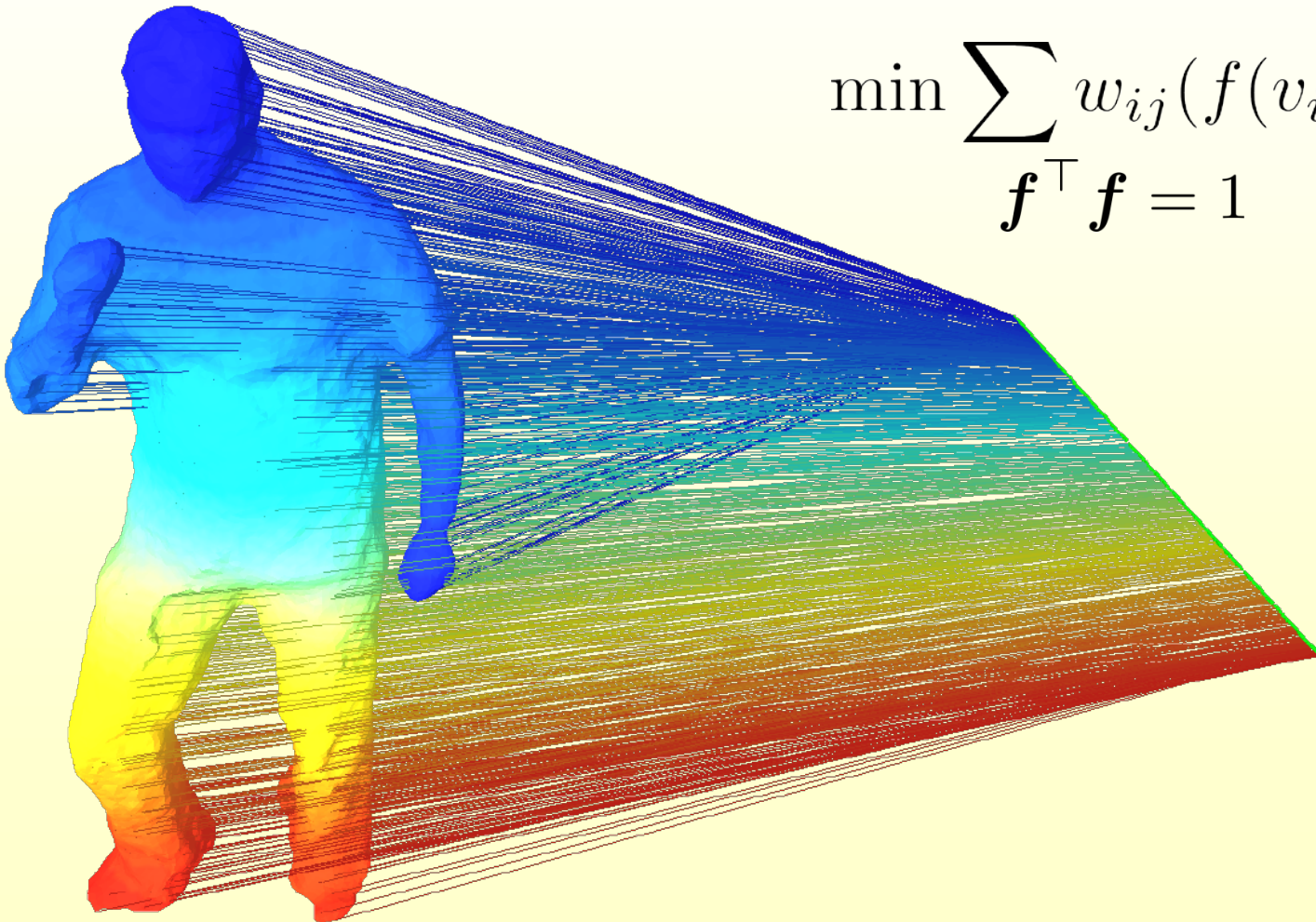
$$\arg \min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L} \mathbf{f} \text{ with: } \mathbf{f}^\top \mathbf{f} = 1 \text{ and } \mathbf{f}^\top \mathbf{1} = 0$$

$\mathbf{f} \perp \mathbf{1}$

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

# 1-d Embedding Example

$$\min \sum w_{ij} (f(v_i) - f(v_j))^2$$
$$f^\top f = 1 \quad f \perp \mathbf{1}$$



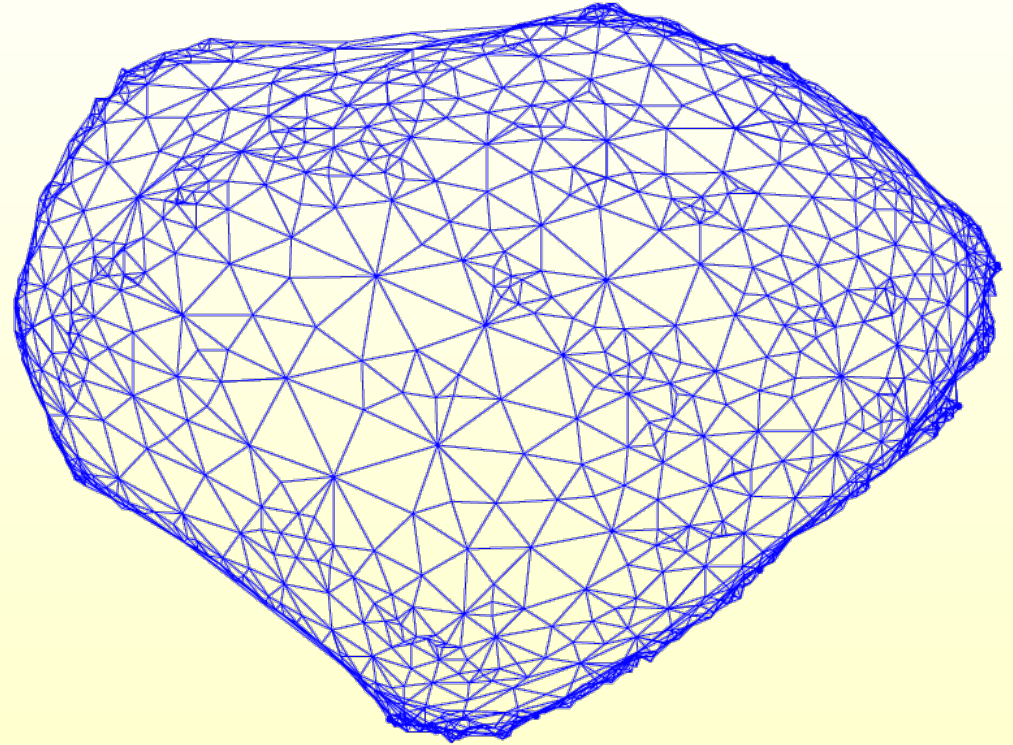
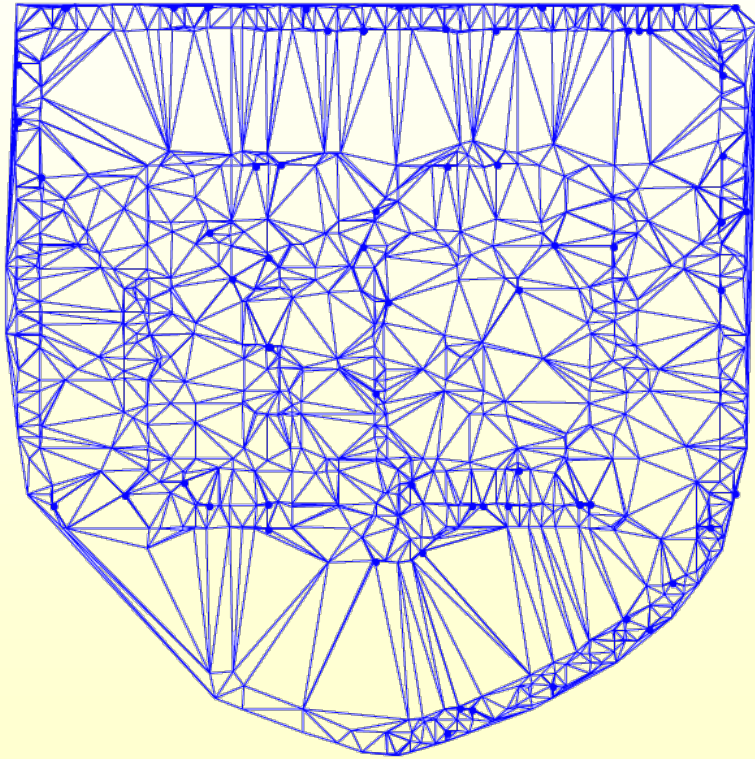
# Higher-d Embeddings

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

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

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

# 2-d Embeddings



# Spectral Graph Drawing

Condition for eigenvector  $Lx = \lambda x$

Gives  $x(i) = \frac{1}{d_i - \lambda} \sum_{j \sim i} x(j)$  for all  $i$

$\lambda$  small says  $x(i)$  near average of neighbors

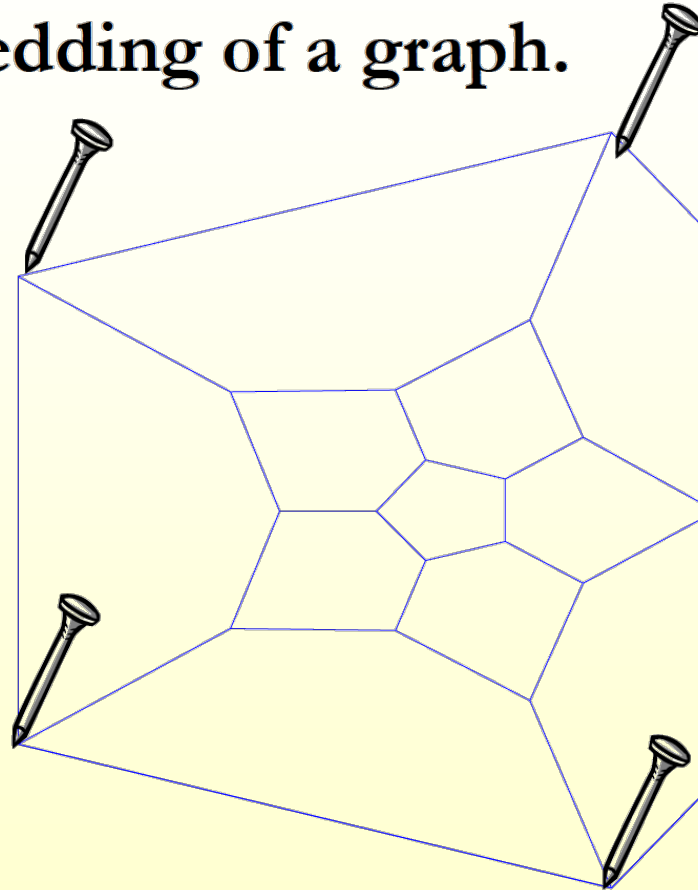
Tutte '63: If fix outside face, and let every other vertex be average of neighbors, get planar embedding of planar graph.

# Tutte Embedding

Tutte '63 embedding of a graph.

Fix outside face.  
Edges  $\rightarrow$  springs.

Vertex at center  
of mass of nbrs.



**William T. Tutte.**

*3-connected  $\rightarrow$  get planar embedding*

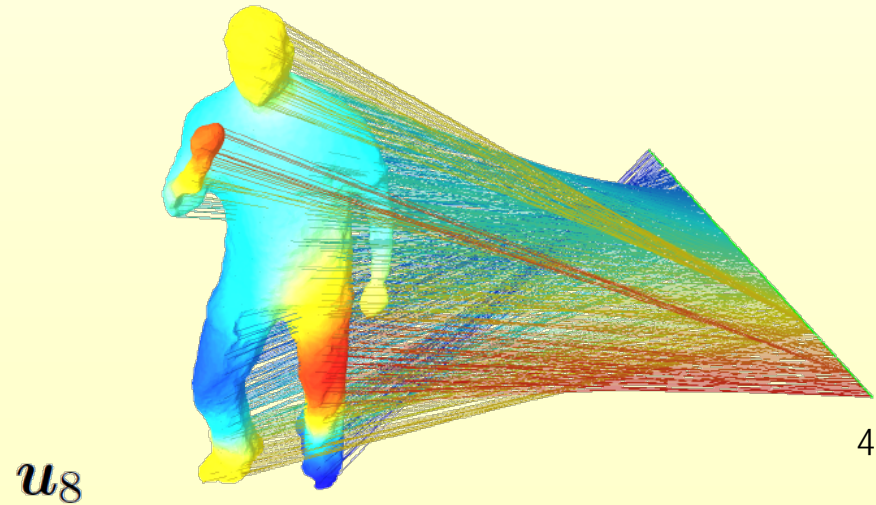
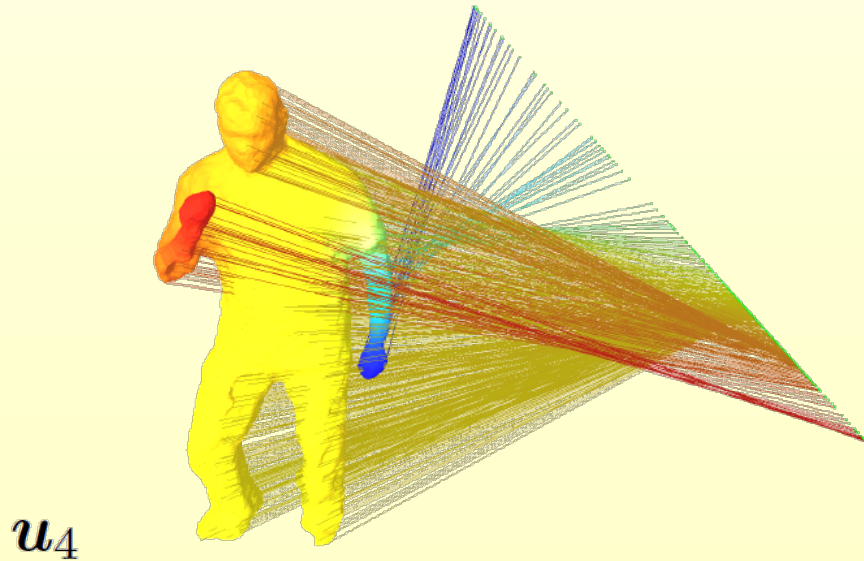
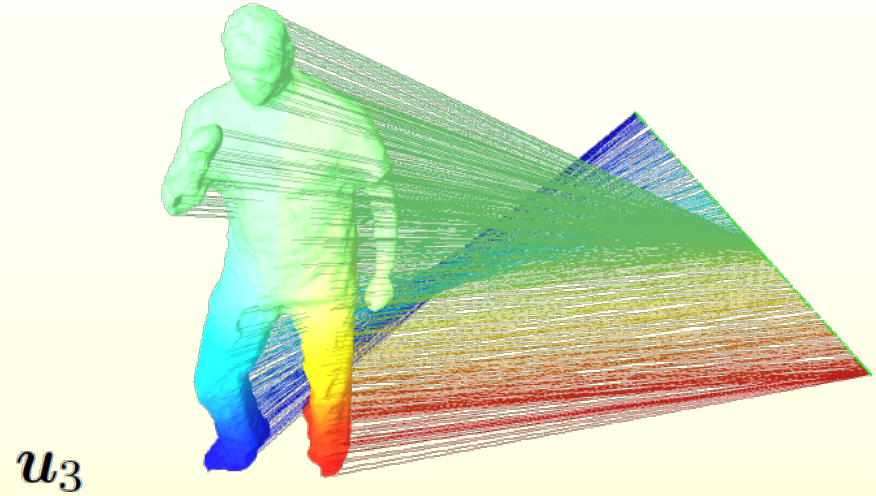
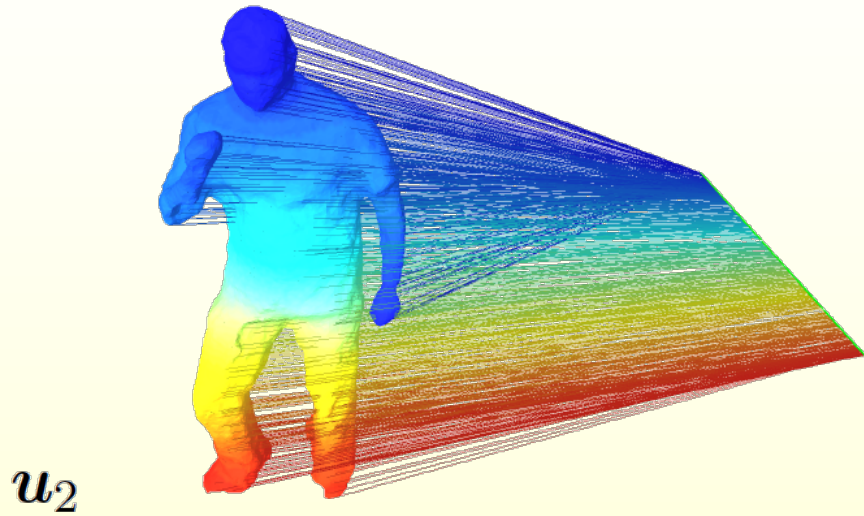
# Spectral Embedding Using Unnormalized Laplacian

- Compute the eigendecomposition  $\mathbf{L} = \mathbf{D} - \mathbf{A}$ .
- Select the  $k$  smallest non-null eigenvalues  $\lambda_2 \leq \dots \leq \lambda_{k+1}$
- $\lambda_{k+2} - \lambda_{k+1} = \mathbf{eigengap}$ .
- We obtain the  $n \times k$  matrix  $\mathbf{U} = [\mathbf{u}_2 \dots \mathbf{u}_{k+1}]$ :

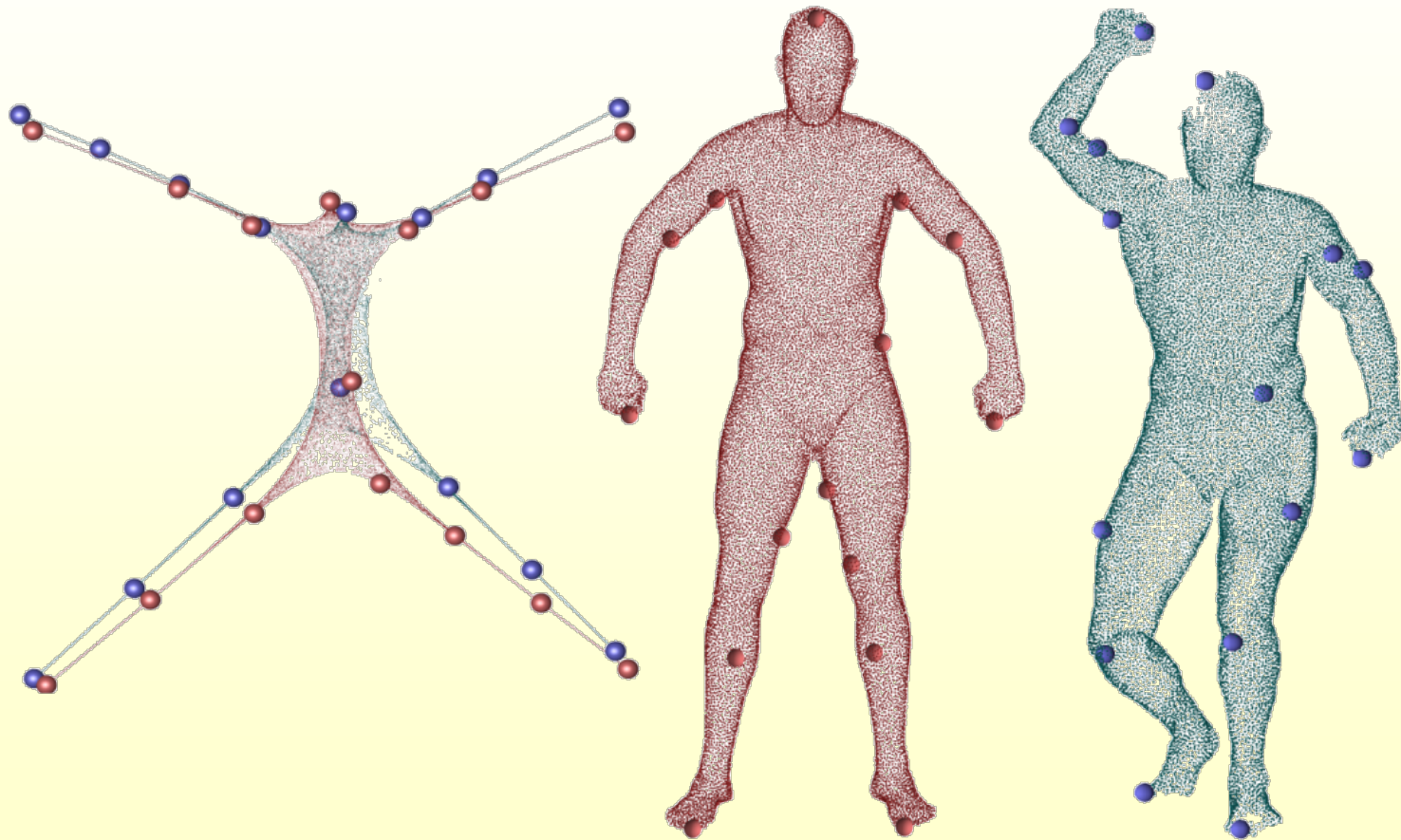
$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_2(v_1) & \dots & \mathbf{u}_{k+1}(v_1) \\ \vdots & & \vdots \\ \mathbf{u}_2(v_n) & \dots & \mathbf{u}_{k+1}(v_n) \end{bmatrix}$$

- $\mathbf{u}_i^\top \mathbf{u}_j = \delta_{ij}$  (orthonormal vectors), hence  $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_k$ .
- Column  $i$  ( $2 \leq i \leq k + 1$ ) of this matrix is a mapping on the eigenvector  $\mathbf{u}_i$ .

# More Eigenvectors, More 1-d Embeddings



# Useful for Isometric Comparisons



Uses random-walk Laplacian (“diffusion distances”)

# Laplacian Variants

# The Normalized Spectral Embedding of a Graph

- (Euclidean)  $\mathbf{L}$ -embedding of a graph:

$$\mathbf{X} = \mathbf{\Lambda}_k^{-\frac{1}{2}} \mathbf{U}^\top = [\mathbf{x}_1 \ \dots \ \mathbf{x}_j \ \dots \ \mathbf{x}_n]$$

The coordinates of a vertex  $v_j$  are:

$$\mathbf{x}_j = \begin{pmatrix} \frac{\mathbf{u}_2(v_j)}{\sqrt{\lambda_2}} \\ \vdots \\ \frac{\mathbf{u}_{k+1}(v_j)}{\sqrt{\lambda_{k+1}}} \end{pmatrix}$$

# Why the Scaling?

Both

- the *commute-time distance* (CTD) and
- the *principal-component analysis* of a graph (graph PCA)

are two important concepts; They allow to reason "statistically" on a graph. They are both associated with the *unnormalized* Laplacian matrix.

# Commute-Time Distance (CTD)

- The CTD is a well known quantity in Markov chains;
- It is the average number of (weighted) edges that it takes, starting at vertex  $v_i$ , to randomly reach vertex  $v_j$  for the first time and go back;
- The CTD decreases as the number of connections between the two nodes increases;
- It captures the connectivity structure of a small graph volume rather than a single path between the two vertices – such as the shortest-path geodesic distance.
- The CTD can be computed in closed form:

$$\text{CTD}^2(v_i, v_j) = \text{vol}(\mathcal{G}) \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

# Graph PCA

- The mean (remember that  $\sum_{j=1}^n \mathbf{u}_i(v_j) = 0$ ):

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_j = \mathbf{\Lambda}_k^{-\frac{1}{2}} \begin{pmatrix} \sum_{j=1}^n \mathbf{u}_2(v_j) \\ \vdots \\ \sum_{j=1}^n \mathbf{u}_{k+1}(v_j) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

- The covariance matrix:

$$\mathbf{S} = \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X} \mathbf{X}^\top = \frac{1}{n} \mathbf{\Lambda}_k^{-\frac{1}{2}} \mathbf{U}^\top \mathbf{U} \mathbf{\Lambda}_k^{-\frac{1}{2}} = \frac{1}{n} \mathbf{\Lambda}_k^{-1}$$

- The vectors  $\mathbf{u}_2, \dots, \mathbf{u}_{k+1}$  are the directions of *maximum variance* of the graph embedding, with  $\lambda_2^{-1} \geq \dots \geq \lambda_{k+1}^{-1}$ .

# Laplacian Variants

- The normalized graph Laplacian (symmetric and semi-definite positive):

$$\mathbf{L}_n = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

- The transition matrix (allows an analogy with Markov chains):

$$\mathbf{L}_t = \mathbf{D}^{-1} \mathbf{A}$$

- The random-walk graph Laplacian:

$$\mathbf{L}_r = \mathbf{D}^{-1} \mathbf{L} = \mathbf{I} - \mathbf{L}_t$$

- These matrices are similar:

$$\mathbf{L}_r = \mathbf{D}^{-\frac{1}{2}} \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} \mathbf{D}^{\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L}_n \mathbf{D}^{\frac{1}{2}}$$

# Eigenvectors/Eigenvalues for

$$L_n, L_r$$

- $L_r \mathbf{w} = \lambda \mathbf{w} \iff L \mathbf{w} = \lambda \mathbf{D} \mathbf{w}$ , hence:

$$L_r : \lambda_1 = 0; \mathbf{w}_1 = \mathbf{1}$$

- $L_n \mathbf{v} = \lambda \mathbf{v}$ . By virtue of the similarity transformation between the two matrices:

$$L_n : \lambda_1 = 0 \quad \mathbf{v}_1 = \mathbf{D}^{\frac{1}{2}} \mathbf{1}$$

- More generally, the two matrices have the same eigenvalues:

$$0 = \lambda_1 \leq \dots \leq \lambda_i \dots \leq \lambda_n$$

- Their eigenvectors are related by:

$$\mathbf{v}_i = \mathbf{D}^{\frac{1}{2}} \mathbf{w}_i, \quad \forall i = 1 \dots n$$

# Random Walk Spectral Embedding

- The  $n \times k$  matrix contains the first  $k$  eigenvectors of  $\mathbf{L}_r$ :

$$\mathbf{W} = [ \mathbf{w}_2 \quad \dots \quad \mathbf{w}_{k+1} ]$$

- It is straightforward to obtain the following expressions, where  $\mathbf{d}$  and  $\mathbf{D}$  are the degree-vector and the degree-matrix:

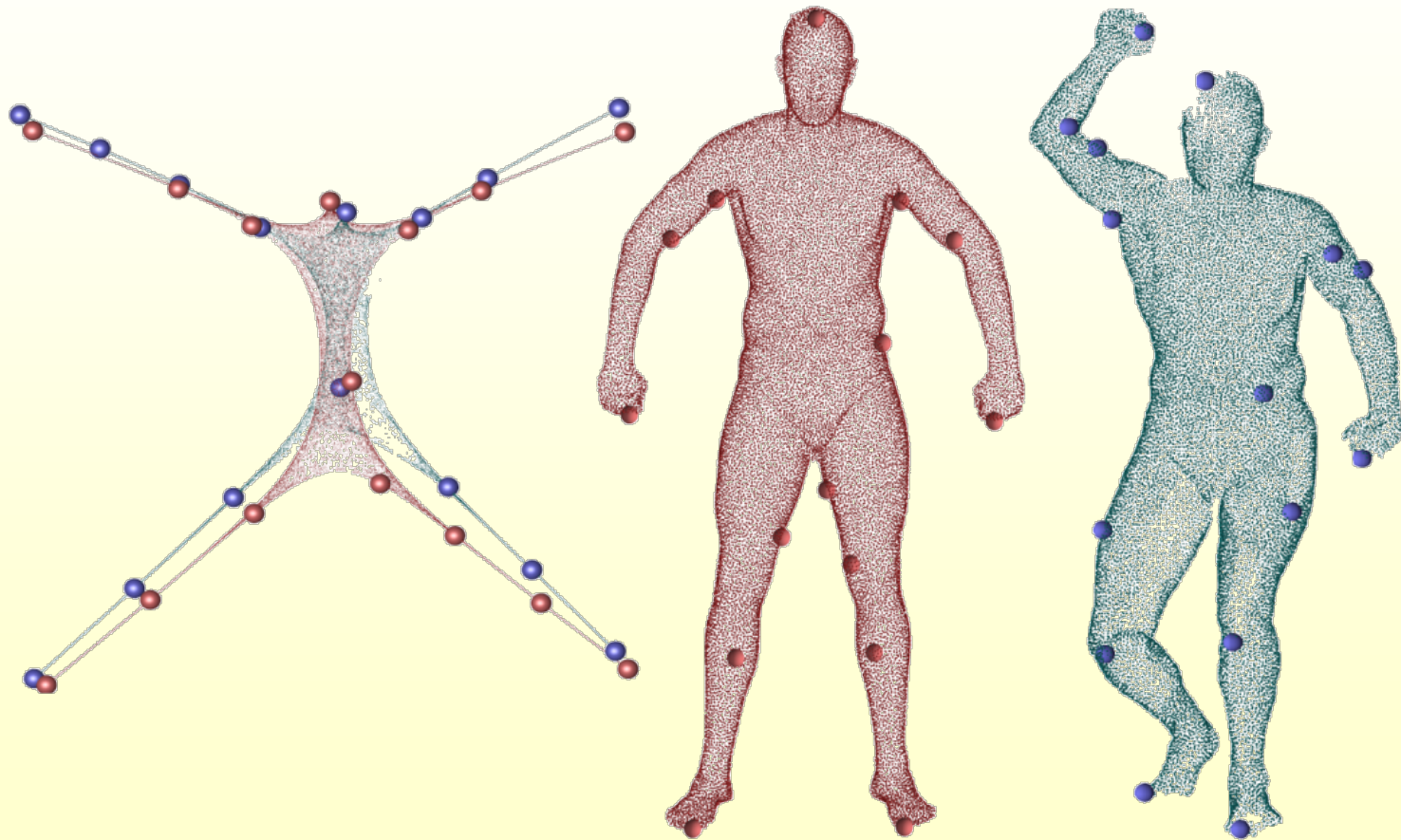
$$\mathbf{w}_i^\top \mathbf{d} = 0, \quad \forall i, 2 \leq i \leq n$$

$$\mathbf{W}^\top \mathbf{D} \mathbf{W} = \mathbf{I}_k$$

- The isometric embedding using the random-walk Laplacian:

$$\mathbf{Y} = \mathbf{W}^\top = [ \mathbf{y}_1 \quad \dots \quad \mathbf{y}_n ]$$

# Useful for Isometric Comparisons



Uses random-walk Laplacian

# Graph Partitioning

- **The graph-cut problem:** Partition the graph such that:
  - ① Edges between groups have very low weight, and
  - ② Edges within a group have high weight.

$$\text{cut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k W(A_i, \bar{A}_i) \text{ with } W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

- **Ratio cut:** (Hagen & Kahng 1992)

$$\text{RatioCut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{|A_i|}$$

- **Normalized cut:** (Shi & Malik 2000)

$$\text{NCut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{\text{vol}(A_i)}$$

vol = sum of  
degrees

# Spectral Clustering

- Both ratio-cut and normalized-cut minimizations are NP-hard problems
- Spectral clustering is a way to solve relaxed versions of these problems:
  - ① The smallest non-null eigenvectors of the *unnormalized Laplacian* approximate the RatioCut minimization criterion, and
  - ② The smallest non-null eigenvectors of the *random-walk Laplacian* approximate the NCut criterion.

# 2-Partition, Fiedler Again

We first rewrite the problem in a more convenient form. Given a subset  $A \subset V$  we define the vector  $f = (f_1, \dots, f_n)' \in \mathbb{R}^n$  with entries

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A}. \end{cases} \quad (2)$$

Now the RatioCut objective function can be conveniently rewritten using the unnormalized graph Laplacian. This is due to the following calculation:

$$\begin{aligned} f'Lf &= \frac{1}{2} \sum_{i,j=1}^n w_{ij}(f_i - f_j)^2 \\ &= \frac{1}{2} \sum_{i \in A, j \in \bar{A}} w_{ij} \left( \sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \frac{1}{2} \sum_{i \in \bar{A}, j \in A} w_{ij} \left( -\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \\ &= \text{cut}(A, \bar{A}) \left( \frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) \\ &= \text{cut}(A, \bar{A}) \left( \frac{|A| + |\bar{A}|}{|A|} + \frac{|A| + |\bar{A}|}{|\bar{A}|} \right) \\ &= |V| \cdot \text{RatioCut}(A, \bar{A}). \end{aligned}$$

Additionally, we have

$$\sum_{i=1}^n f_i = \sum_{i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i \in \bar{A}} \sqrt{\frac{|A|}{|\bar{A}|}} = |A| \sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}| \sqrt{\frac{|A|}{|\bar{A}|}} = 0.$$

In other words, the vector  $f$  as defined in Equation (2) is orthogonal to the constant one vector  $\mathbf{1}$ . Finally, note that  $f$  satisfies

$$\|f\|^2 = \sum_{i=1}^n f_i^2 = |A| \frac{|\bar{A}|}{|A|} + |\bar{A}| \frac{|A|}{|\bar{A}|} = |\bar{A}| + |A| = n.$$

# Spectral Clustering Using the Random-Walk Laplacian

- For details see (von Luxburg '07)
  - Input: Laplacian  $\mathbf{L}_r$  and the number  $k$  of clusters to compute.
  - Output: Cluster  $C_1, \dots, C_k$ .
- 1 Compute  $\mathbf{W}$  formed with the first  $k$  eigenvectors of the random-walk Laplacian.
  - 2 Determine the spectral embedding  $\mathbf{Y} = \mathbf{W}^\top$
  - 3 Cluster the columns  $\mathbf{y}_j, j = 1, \dots, n$  into  $k$  clusters using the K-means algorithm.

# $k$ -Means Clustering

See Bishop'2006 (pages 424–428) for more details.

- What is a cluster: a group of points whose inter-point distance are small compared to distances to points outside the cluster.
- Cluster centers:  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ .
- Goal: find an assignment of points to clusters as well as a set of vectors  $\boldsymbol{\mu}_i$ .
- Notations: For each point  $\mathbf{y}_j$  there is a *binary indicator variable*  $r_{ji} \in \{0, 1\}$ .
- Objective: minimize the following *distorsion measure*:

$$J = \sum_{j=1}^n \sum_{i=1}^k r_{ji} \|\mathbf{y}_j - \boldsymbol{\mu}_i\|^2$$

# $k$ -Means Algorithm

- 1 Initialization: Choose initial values for  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ .
- 2 First step: Assign the  $j$ -th point to the closest cluster center:

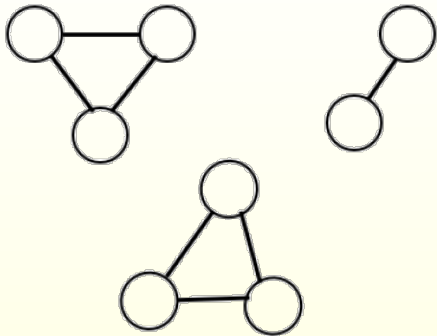
$$r_{ji} = \begin{cases} 1 & \text{if } i = \arg \min_l \|\mathbf{y}_j - \boldsymbol{\mu}_l\|^2 \\ 0 & \text{otherwise} \end{cases}$$

- 3 Second Step: Minimize  $J$  to estimate the cluster centers:

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^n r_{ji} \mathbf{y}_j}{\sum_{j=1}^n r_{ji}}$$

- 4 Convergence: Repeat until no more change in the assignments.

# Spectral Clustering: The Ideal Case

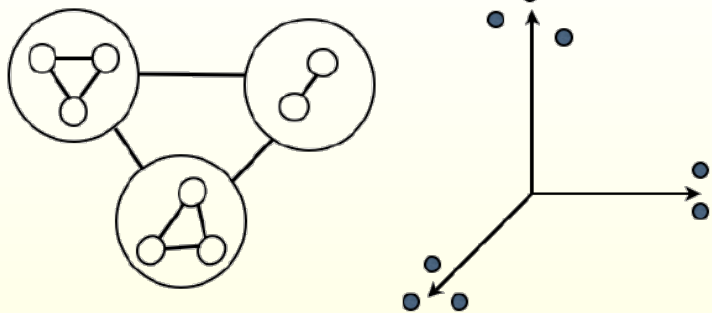


- $\lambda_1 = \lambda_2 = \lambda_3 = 0$
- $w_1, w_2, w_3$  form an orthonormal basis.
- The connected components collapse to  $(100), (010), (001)$ .
- Clustering is trivial in this case.

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

# Spectral Clustering: The Perturbed Case



- See (von Luxburg '07) for a detailed analysis.
- The connected components are no longer *disconnected*, but they are only connected by few edges with low weight.

- The Laplacian is a perturbed version of the ideal case.
- Choosing the first  $k$  nonzero eigenvalues is easier the larger the eigengap between  $\lambda_{k+1}$  and  $\lambda_{k+2}$ .
- The fact that the first  $k$  eigenvectors of the perturbed case are approximately piecewise constant depends on  $|\lambda_{k+2} - \lambda_{k+1}|$ .
- **Choosing  $k$  is a crucial issue.**

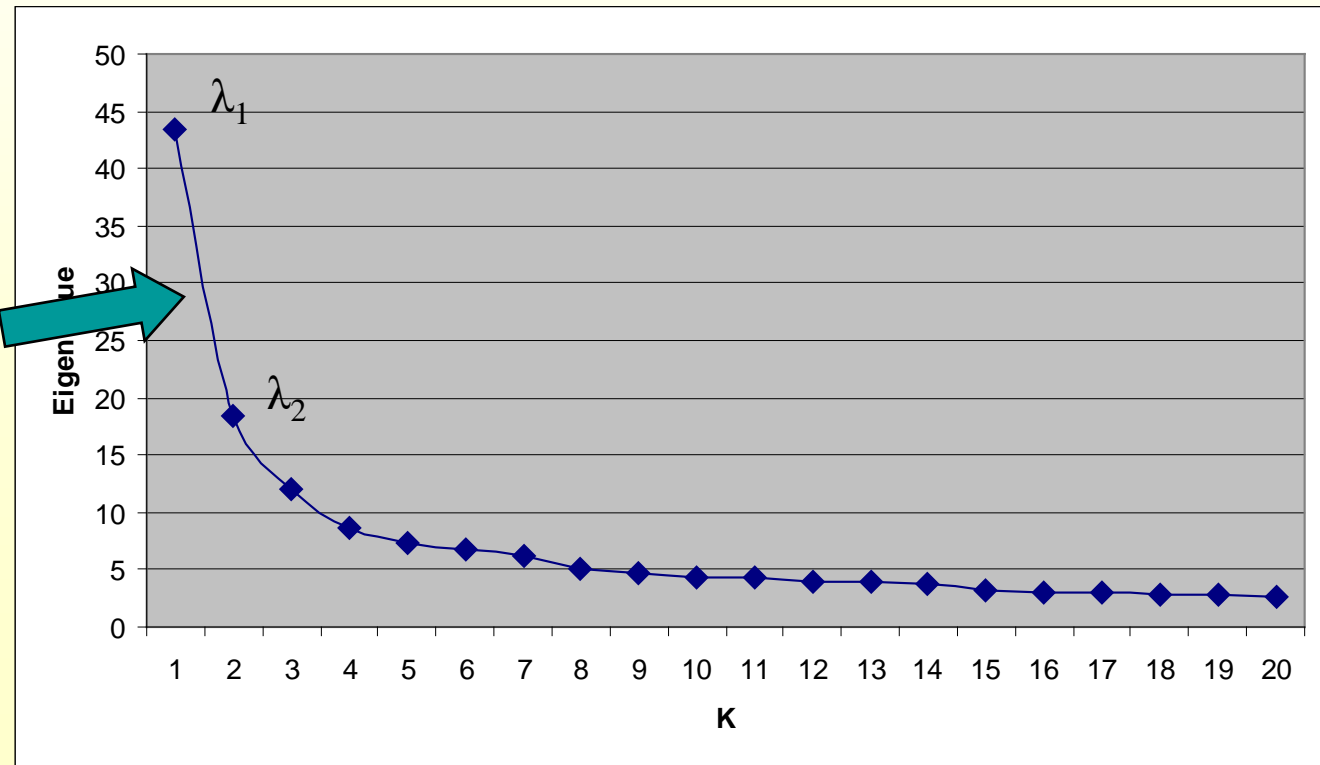
# Spectral Gap: Selecting $k$

- ◆ *Eigengap*: the difference between two consecutive eigenvalues.
- ◆ Most stable clustering is generally given by the value  $k$  that maximizes the expression

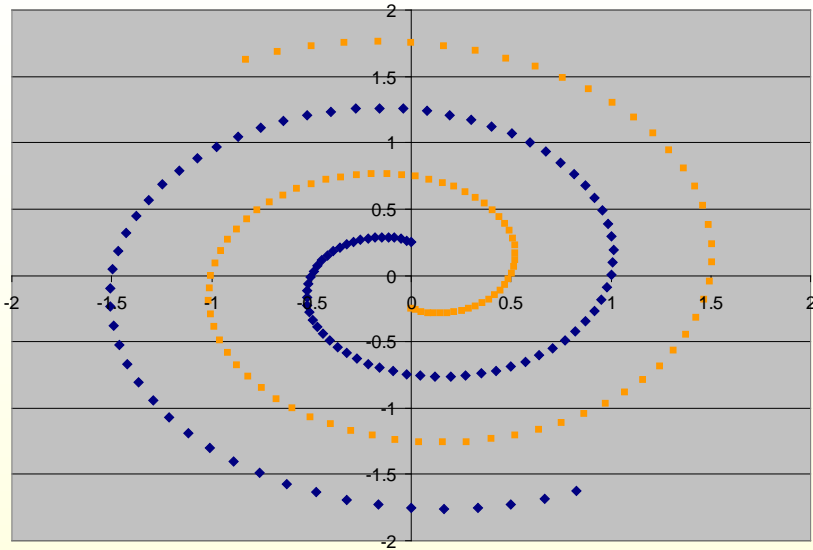
$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$

$$\max \Delta_k = |\lambda_2 - \lambda_1|$$

⇒ Choose  $k=2$

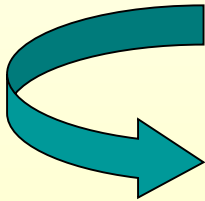


# Spirals Again

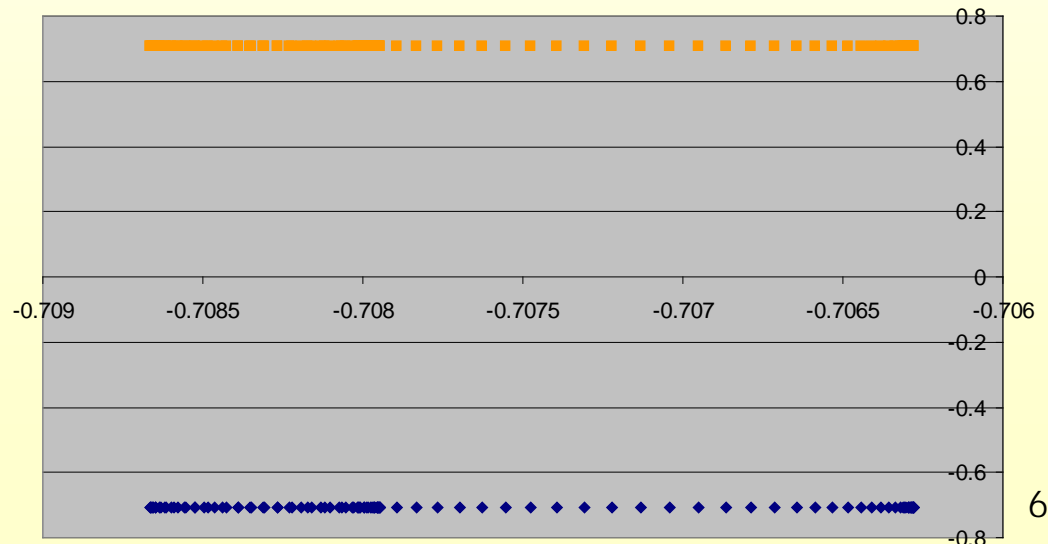


Dataset exhibits complex cluster shapes

⇒ Direct  $k$ -means performs very poorly in this space due to bias toward dense spherical clusters.



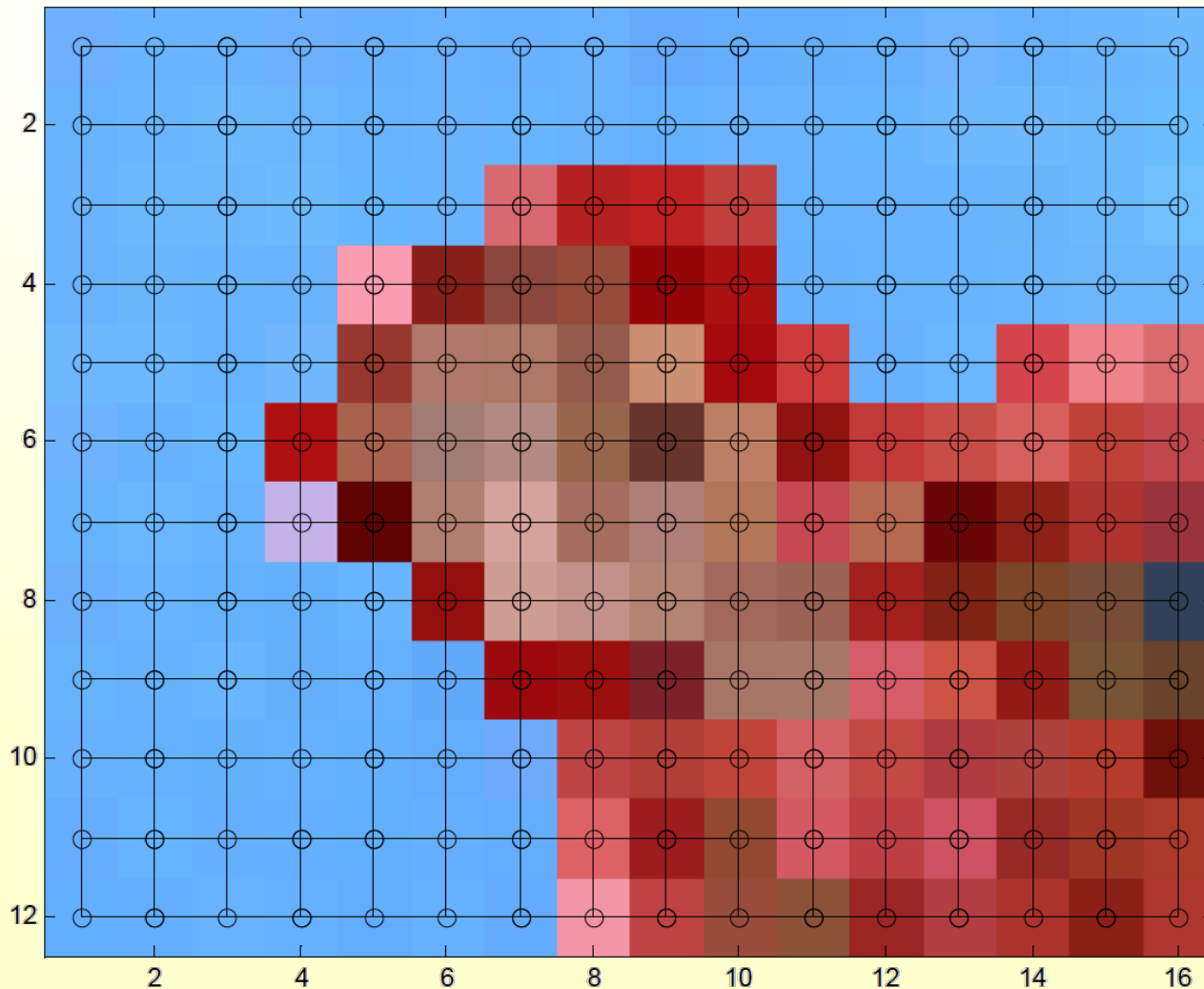
In the embedded space given by two leading eigenvectors, clusters are trivial to separate.



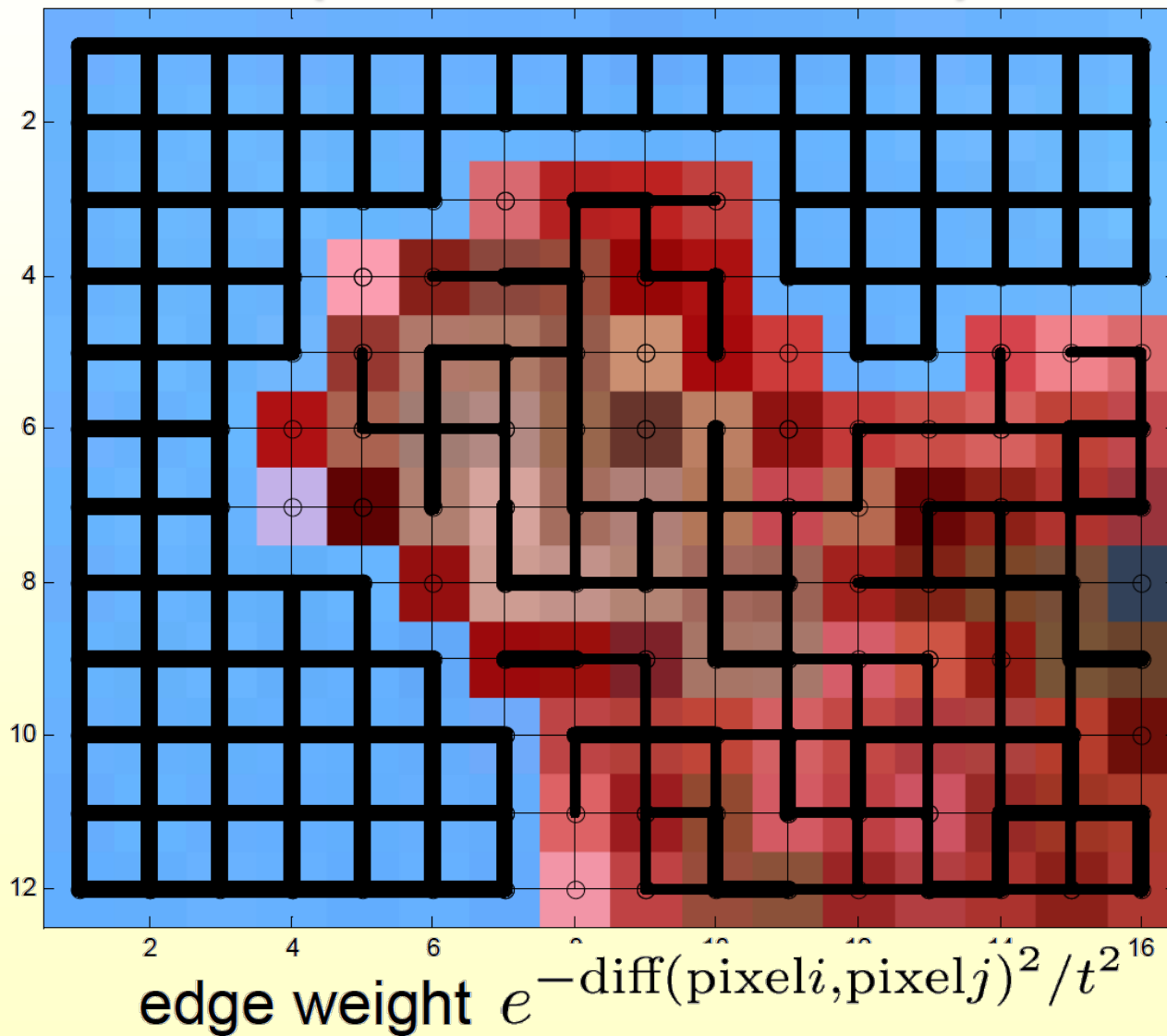
# Spectral Image Segmentation



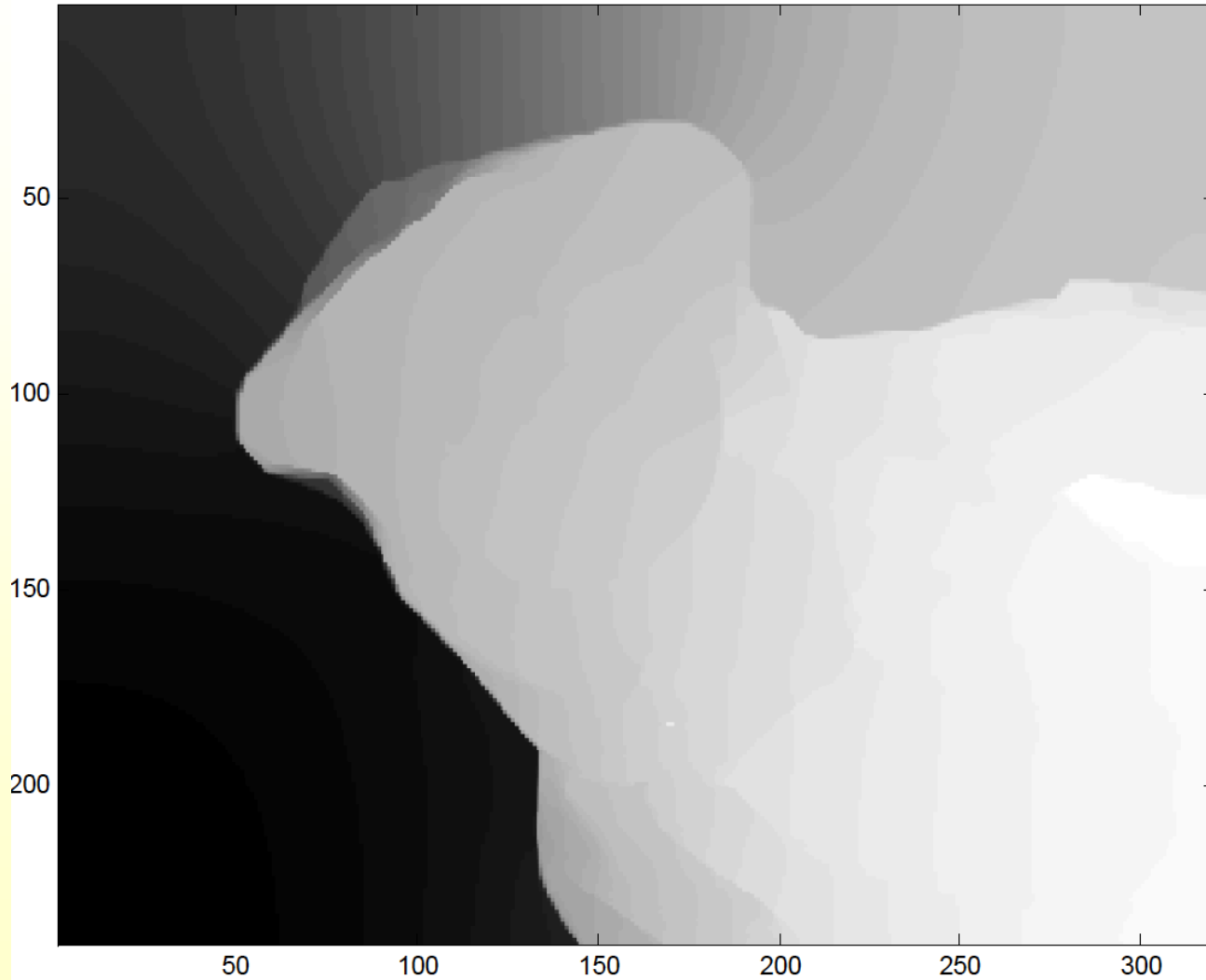
# Spectral Image Segmentation (Shi-Malik '00)



# Spectral Image Segmentation (Shi-Malik '00)



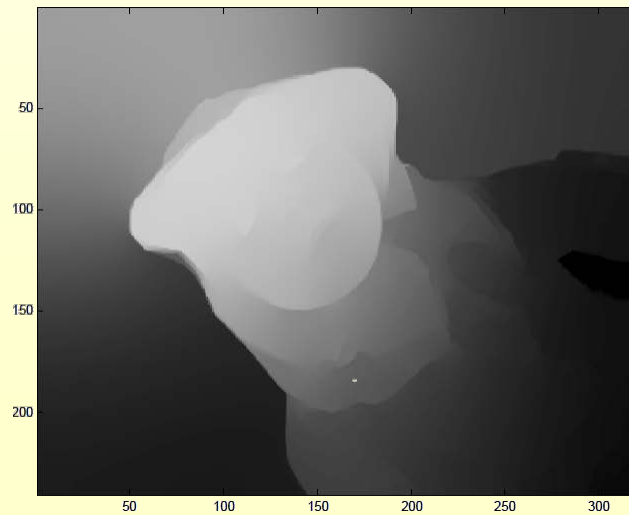
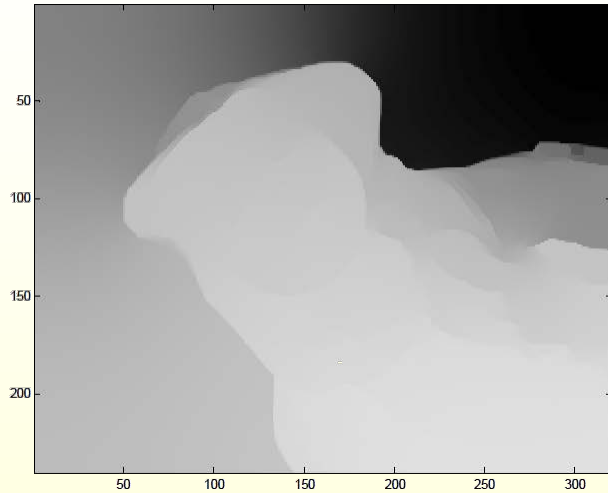
# Second Eigenvector



# Second Eigenvector Sparsest Cut



# 3<sup>rd</sup> and 4<sup>th</sup> Eigenvectors



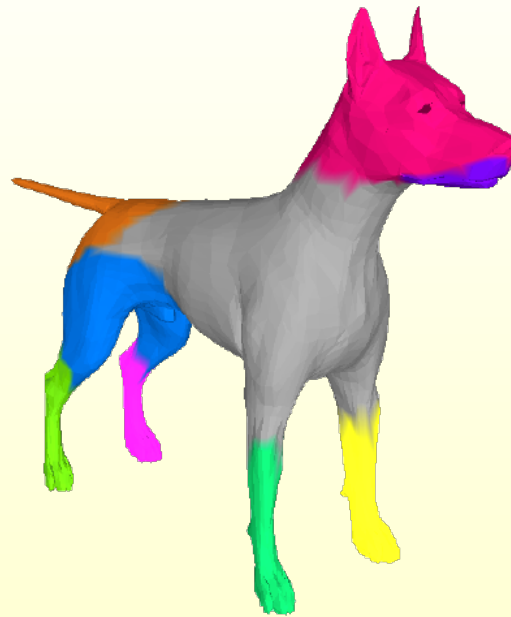
# Mesh Segmentation Using Spectral Clustering



K=6



K=6



K=9



K=6

# Conclusion

- Spectral graph embedding based on the graph Laplacian is a very powerful tool;
- Allows links between graphs and Riemannian manifolds
- There are strong links with Markov chains and random walks
- It allows clustering (or segmentation) under some conditions

**The End**

