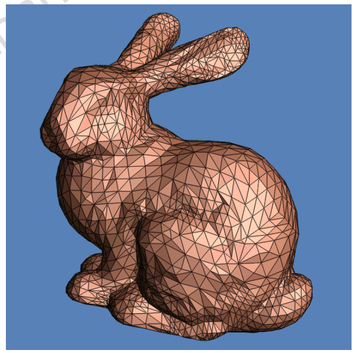
ED5340 - Data Science: Theory and Practise

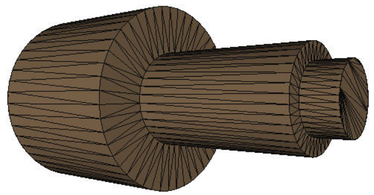
L30 - Manifold Learning

Ramanathan Muthuganapathy (https://ed.iitm.ac.in/~raman)
Course web page: https://ed.iitm.ac.in/~raman/datascience.html
Moodle page: Available at https://courses.iitm.ac.in/

Object representation

- Continuous – Curves / Surfaces represented using equations, typically polynomials. Bezier/B-Spline
- Discrete – Images, Mesh models, Point-set





An overview

- Group together similar pixels
- *Semantic* segmentation
  - –Object recognition
    - Decompose objects to simple tokens (line segments, spots, corners)
  - –Finding buildings in images
    - Fit polygons and determine surface orientations.

Image search and retrieval

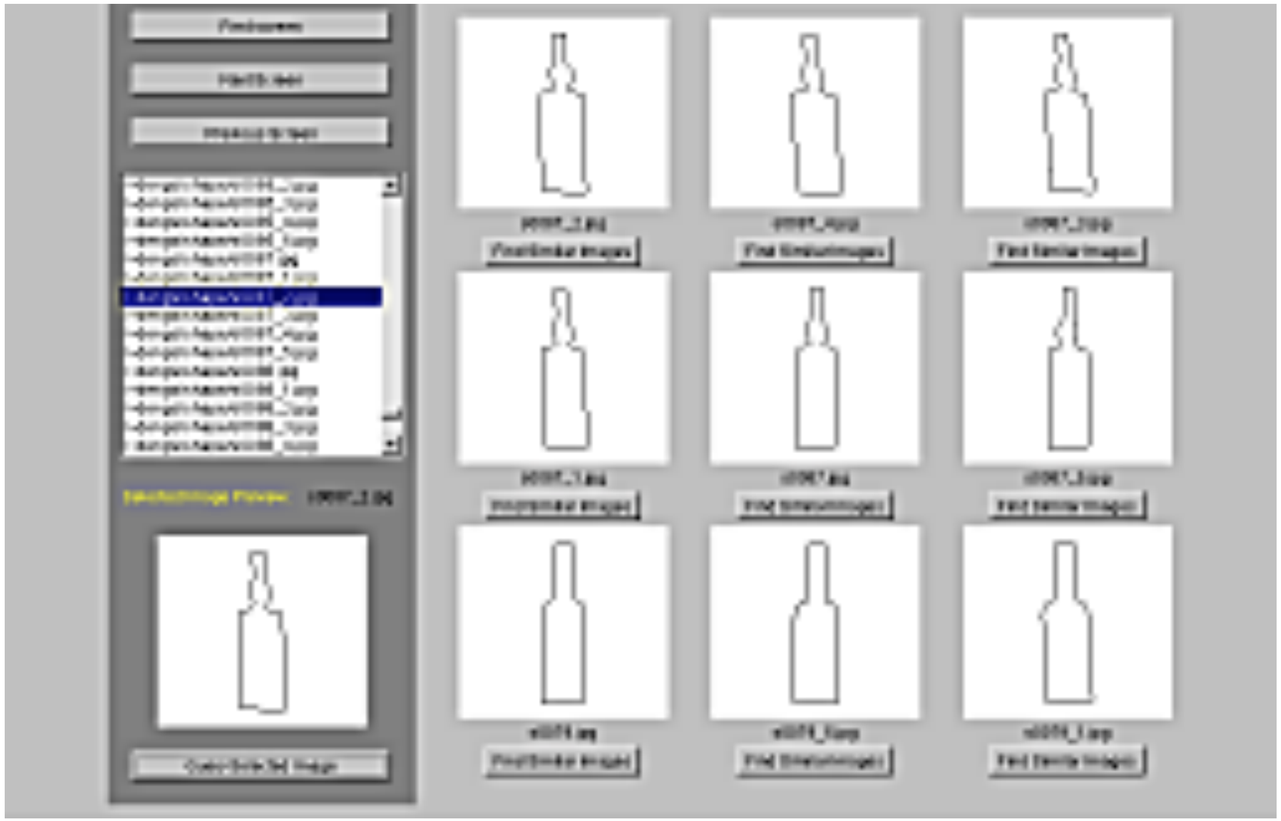


Image correspondence

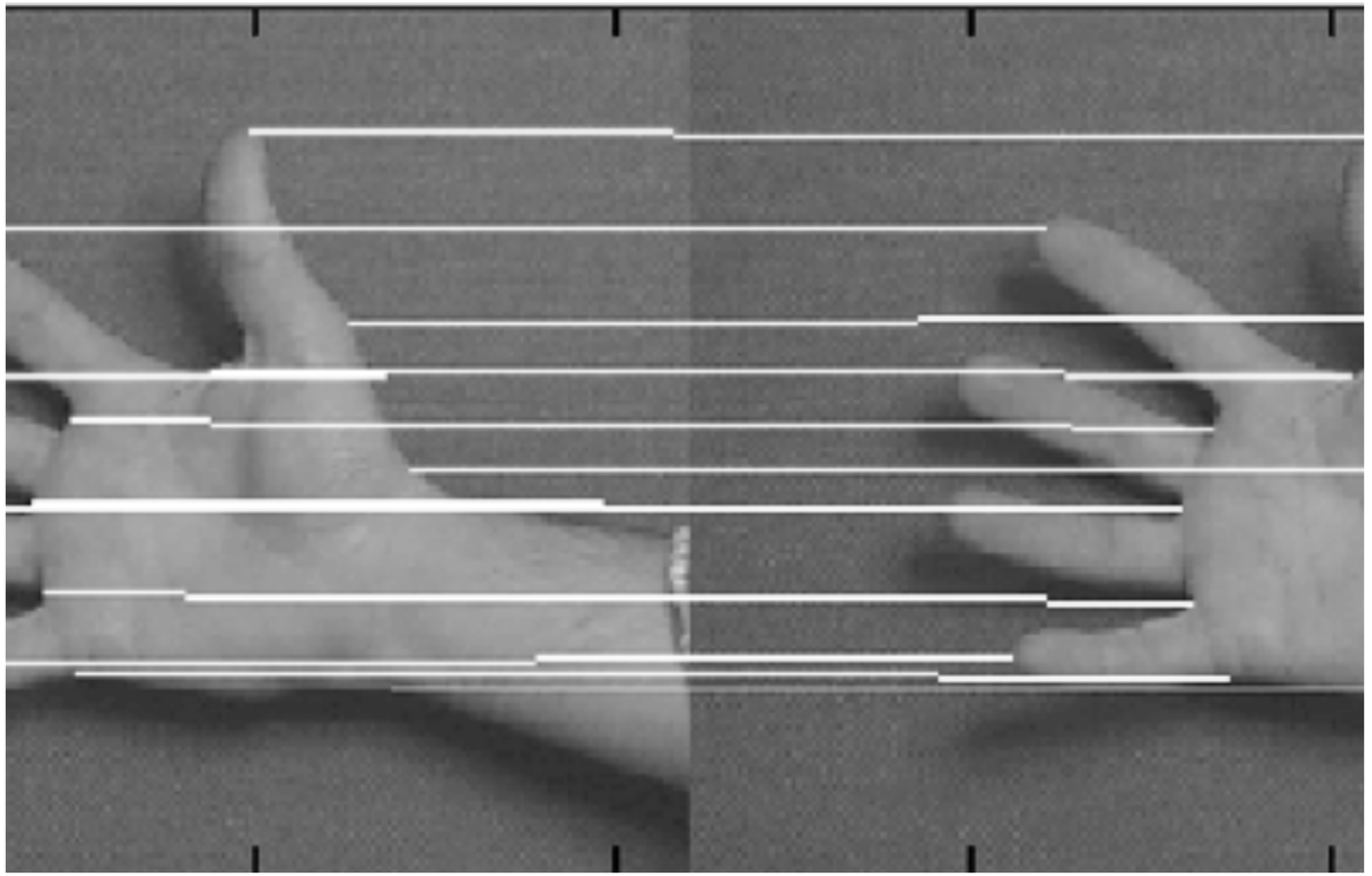
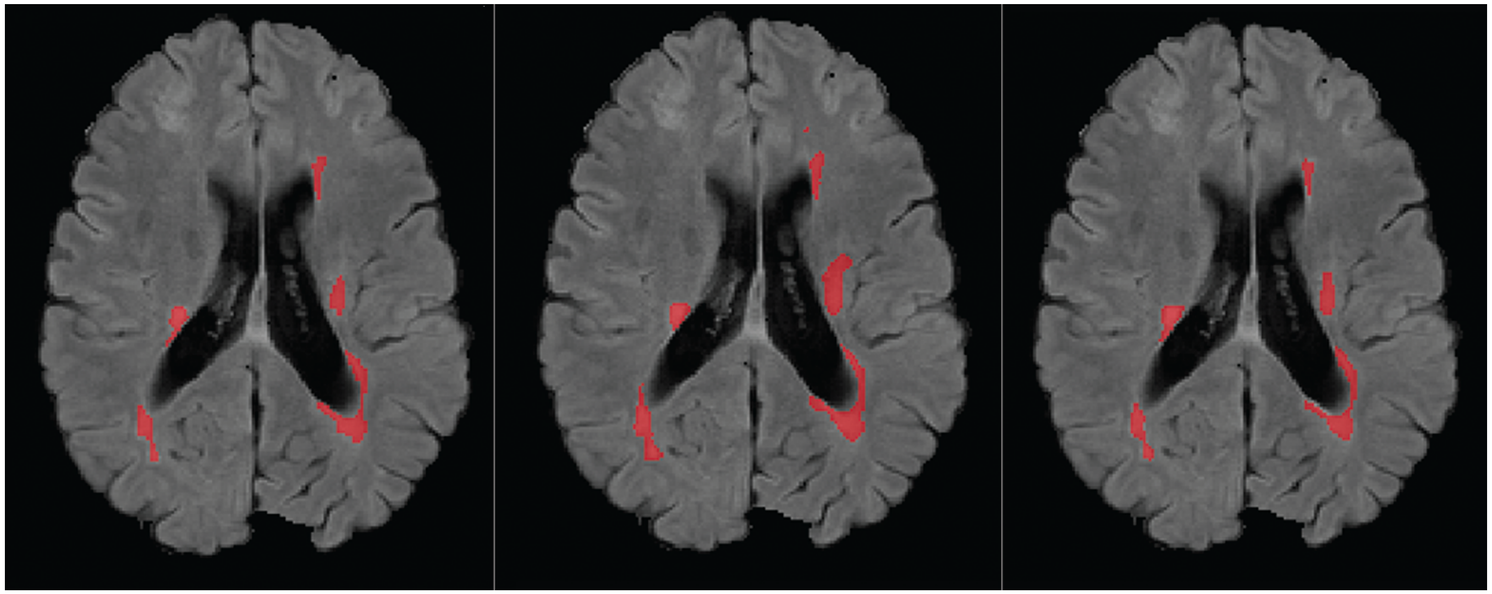


Image Registration

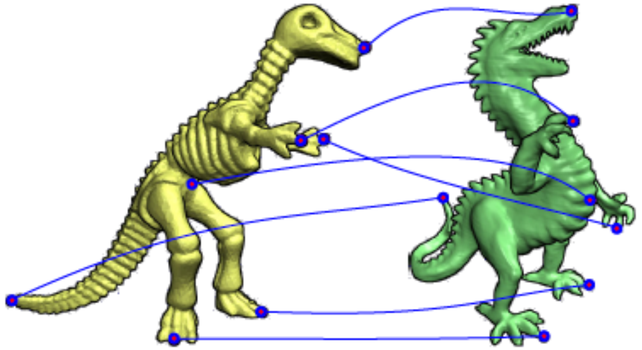
- Image registration is the process of overlaying images (two or more) of the same scene taken at different times, from different viewpoints, and/or by different sensors.
- The registration geometrically align two images (the reference and sensed images).

Biomedical image processing



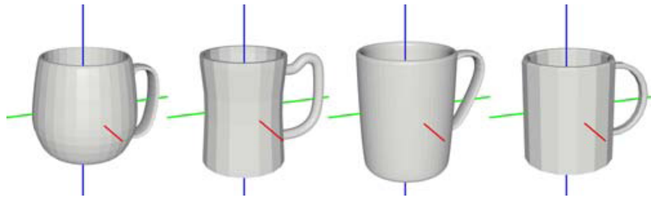
Similar problems in Mesh models

Segmentation



Correspondence

Matching and Retrieval



Alignment

Commomalities

- Large data
- Analysing large data
- Computing features – Shape signatures (fast and simple, but also robust and meaningful)
- Local vs Global

Dimensionality Reduction and Data representation

- Central Problem in Machine Learning and Pattern recognition is to develop appropriate representation for complex data.
- Representation for data lying on a low dimensional manifold embedded in a high dimensional space.
- How can we detect low dimensional structure in high dimensional data?

Spectral Methods

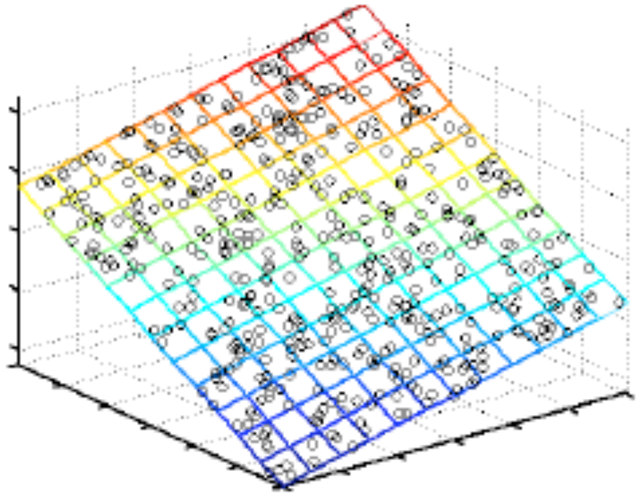
- Matrix analysis - Low dimensional structure is revealed by eigenvalues and eigenvectors.
- Links to spectral graph theory - Matrices are derived from sparse weighted graphs - Graph Laplacian, Laplace Beltrami operator and heat equation.
- Usefulness - Tractable methods can reveal nonlinear structure.

Dimensionality Reduction

- Inputs (high dime $\vec{x}_i \in \Re^D$ with $i = 1, 2, ..., n$
- Outputs (Low dim $\vec{y}_i \in \Re^d$ where $d \ll D$
- Typical Goal – Nearby points remain as such and so the distant points. Estimate *d*

Linear vs. Non-linear

Inputs are real valued vectors in a high dimensional space



Linear

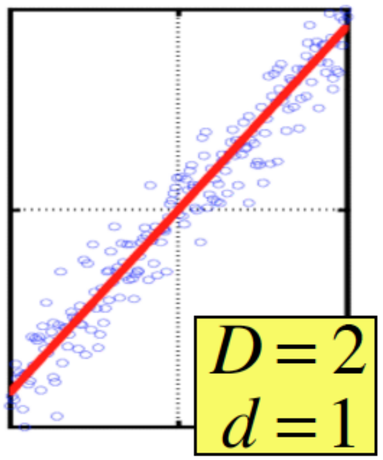
Does the data live in a low dimensional subspace?

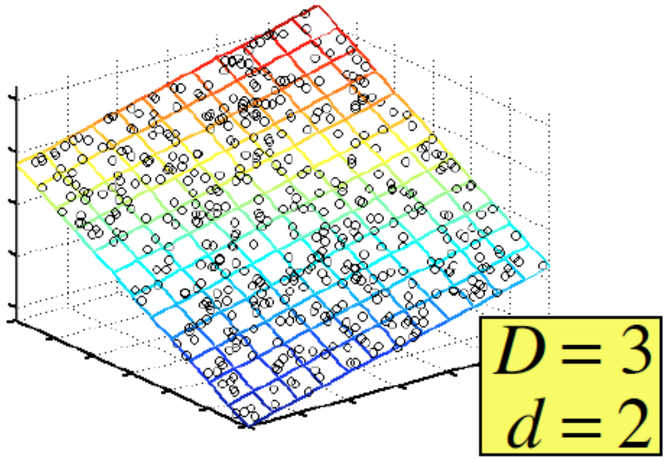
Non-linear

Does the data live on a low dimensional submanifold?

Linear methods (PCA)

- **Does the data mostly lie in a subspace? If so, what is its dimensionality?**
- Covariance matrix and spectral decomposition (up to rotation) – finds maximum variance subspace





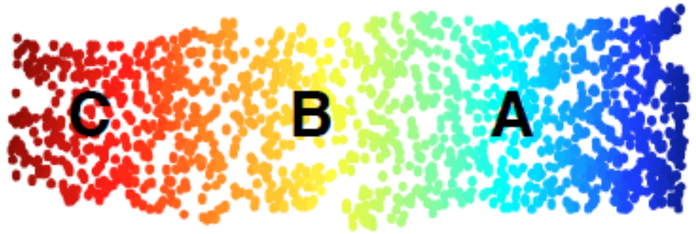
Manifold Learning

- Given high dimensional data from a low dimensional submanifold, how to compute a faithful embedding.

Distance on manifolds

- Rank ordering of Euclidean distances is NOT preserved in “manifold learning”.

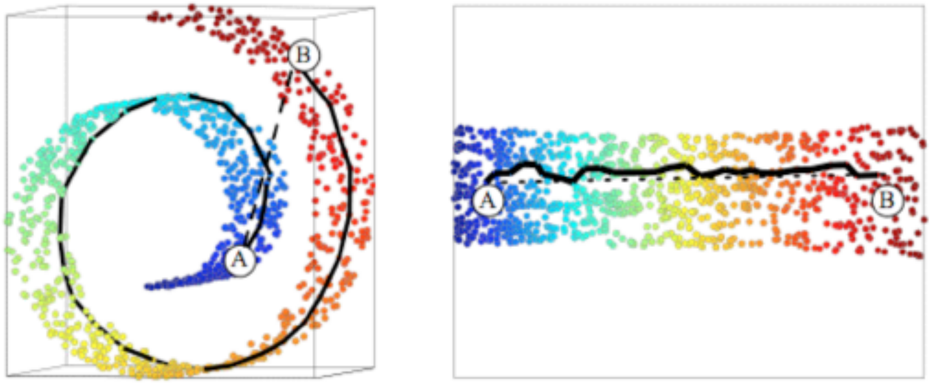
$d(A,C) < d(A,B)$



$d(A,C) > d(A,B)$

Isometric mapping (ISOMAP)

- **Key idea: Preserve geodesic distances as measured along submanifold.**
- Algorithm in a nutshell: Use geodesic instead of (transformed) Euclidean distances in MDS.



ISOMAP Computation

- Build Adjacency graph (kNN is one option), Vertices as inputs and Undirected edges connect neighbours.
- Weight edges by local distances, compute shortest paths.
- Top `d' eigenvectors of Gram matrix yield embedding.
- Number of significant Eigen values yield dimensionality.

Properties of ISOMAP

- Strengths
  - Polynomial-time optimizations
  - No local minima
  - Non-iterative (one pass thru data)
  - Non-parametric
  - Only heuristic is neighborhood size.
- Weaknesses
  - Sensitive to “shortcuts”
  - No out-of-sample extension
  - Too expensive to compute all shortest paths and diagonalize full Gram matrix (use `Landmarks’ for sampling).

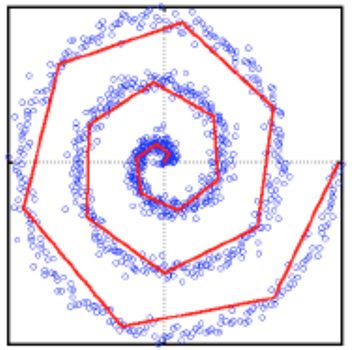
Local approaches

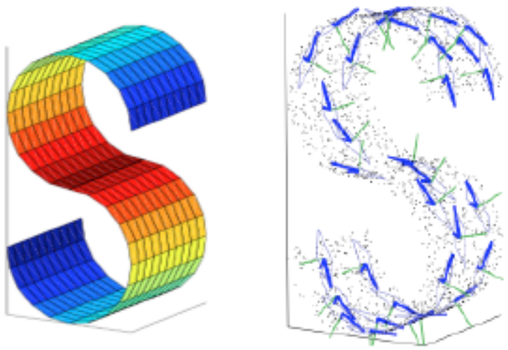
- preserve local geometric relationships
- construct large, sparse matrices
- compute bottom eigenvectors

Local Linear

Based on the premise

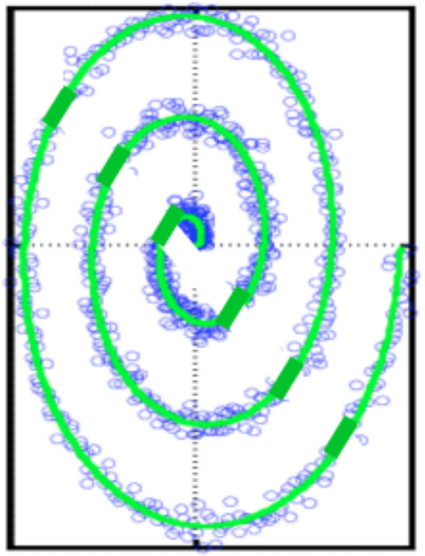
- **Manifolds are globally nonlinear, but locally**

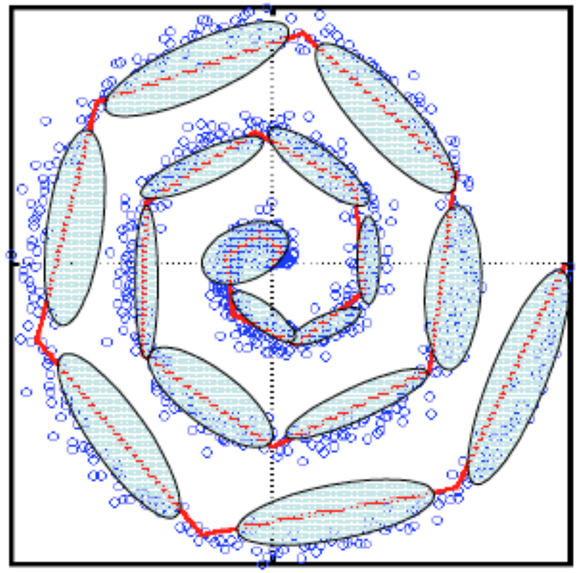




Local vs Global

- Clustering algorithms do not map their inputs into a single continuous global coordinate system of lower dimensionality.





Locally Linear Embedding (LLE)

- Steps
  - Nearest neighbor search (kNN).
  - Least squares fits (Compute Weights W).
  - Sparse eigenvalue problem (compute outputs).

Example

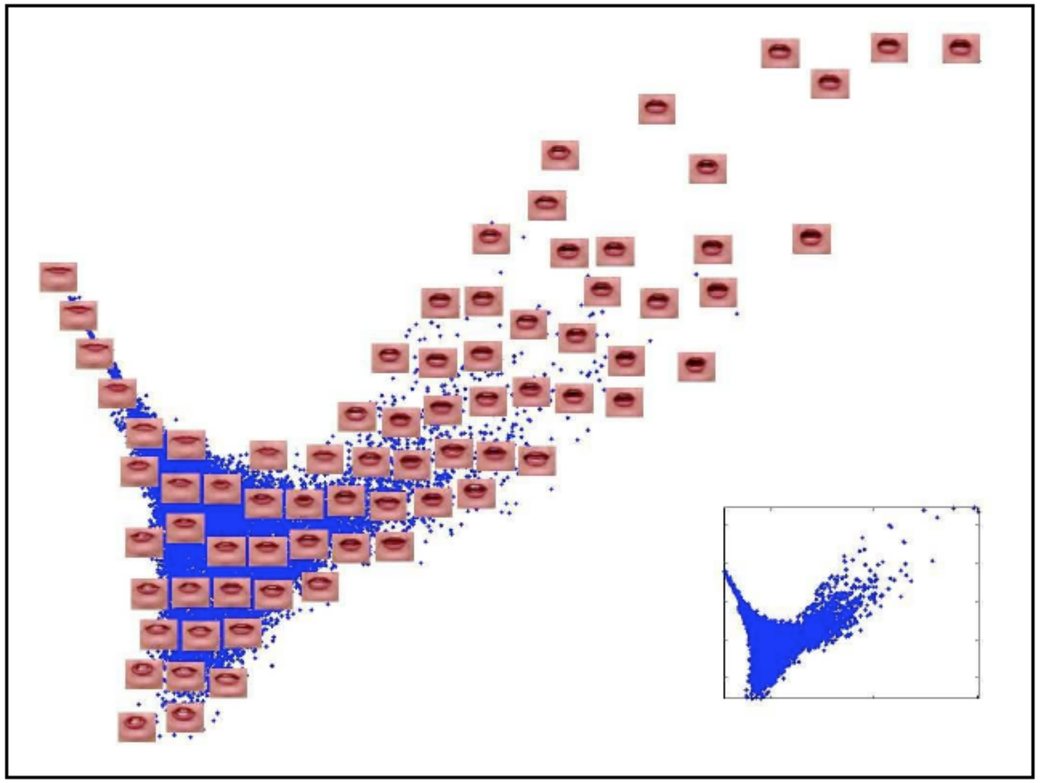
Lips

N=15960
images

K=24
neighbors

D=65664
pixels

d=2
(shown)



Properties of LLE

- Strengths
  - Polynomial-time optimizations
  - No local minima
  - Non-iterative (one pass thru data)
  - Non-parametric
  - Only heuristic is neighborhood size.
- Weaknesses
  - Sensitive to “shortcuts”
  - No out-of-sample extension
  - No estimate of dimensionality

LLE vs Isomap

- Many similarities
  - Graph-based, spectral method
  - No local minima
- Essential differences
  - Does not estimate dimensionality
  - No theoretical guarantees

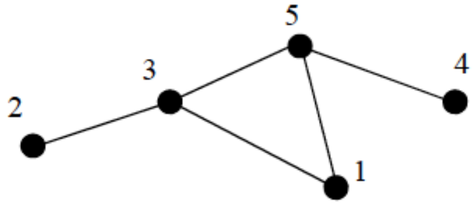
+ Constructs sparse vs dense matrix

? Preserves weights vs distances

Laplacian Eigenmaps

- **Key idea: Map nearby inputs to nearby outputs, where nearness is encoded by graph.**

Networks

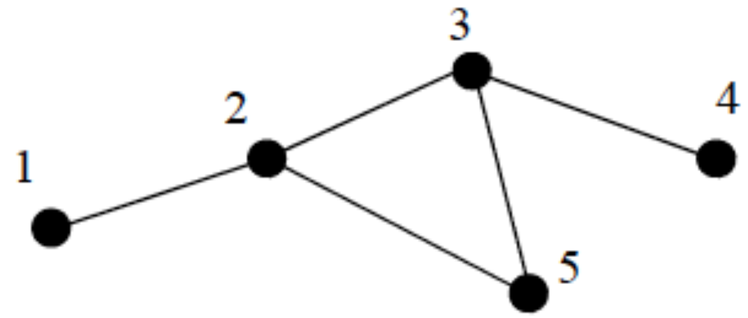


- Networks are structural – it is the arrangement of edges that matters
- In order to compare the edges, we need to know which is which
- We can do this by labelling the vertices
- In a ‘pure’ network, there is no intrinsic difference between the vertices
- We do not know which labelling is the best and there are *n*! labellings

Laplacian – Matrix representation

- A Matrix Representation **X** of a network is matrix with entries representing the vertices and edges
  - – First we label the vertices
  - – Then an element of the matrix *Xuv* represents the edge between vertices *u* and *v*
  - – *Xuu* represents the vertex *u*
  - – The most basic example is the adjacency matrix

Adjacency matrix



$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0
\end{pmatrix}
\end{array}
$$

For an undirected graph, the matrix is symmetric

Degree

- The adjacency contains no vertex information;
- The degree matrix **D** contains the degrees of the vertices

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Laplacian L

- L = D – A

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 \\ 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 2 \end{pmatrix}$$

- L_s = D + A (Signless Laplacian)

Normalised Laplacian

- Normalized Laplacian

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

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

- Entries are

$$L_{uv} = \begin{cases} 1 & u = v \\ -\dfrac{1}{\sqrt{d_u d_v}} & (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$

Eigen Decomposition

- At the heart of spectral graph theory are matrix eigenvalues and eigenvectors

$$\mathbf{X}\mathbf{u} = \lambda\mathbf{u}$$

  - $\mathbf{X}$ is the square matrix we are interested in
  - λ is an eigenvalue of the matrix
  - $\mathbf{u}$ is an (right) eigenvector of the matrix
- Left eigenvector

$$\mathbf{u}^T\mathbf{X} = \lambda\mathbf{u}^T$$

- For a symmetric matrix
  - Always *n* orthogonal eigenvectors
  - Eigenvalues real
  - Left & right eigenvectors the same

Eigen Decomposition (contd.)

- Any square matrix has an eigendecomposition (into eigenvectors and eigenvalues)
- When dealing with undirected graphs – these have a square and symmetric matrix representation
- The eigendecomposition is then

$$\mathbf{X} = \mathbf{U}\Lambda\mathbf{U}^T$$

$$\begin{pmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_{n-1} \end{pmatrix} \qquad \begin{pmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & & \\ \vdots & & \ddots & \\ 0 & & & \lambda_{n-1} \end{pmatrix}$$

All real numbers

Spectrum of the graph

- The graph has a ordered set of eigenvalues ($\lambda_0, \lambda_1, \ldots \lambda_{n-1}$)

  (smallest first)

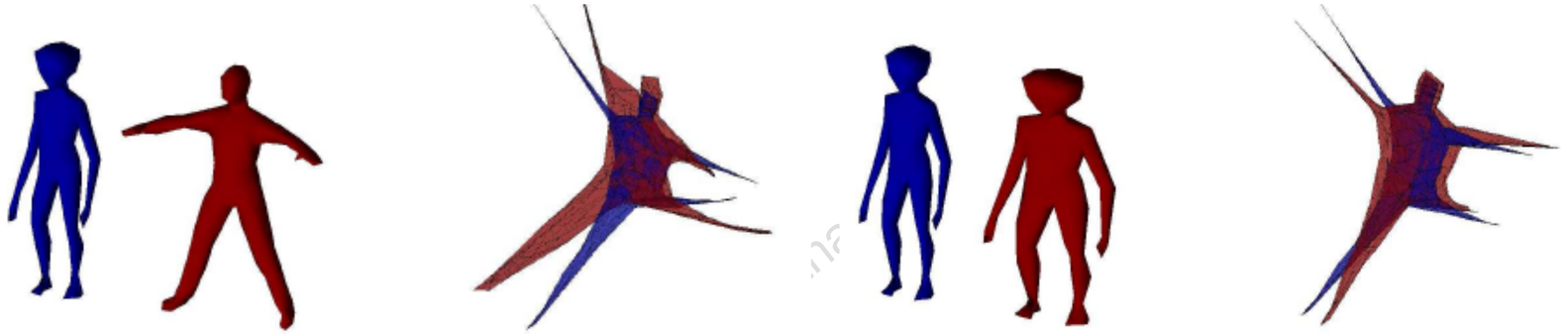
- The (ordered) set of eigenvalues is called the *spectrum* of the graph

Shape Correspondence

- Given two 3D shapes represented as 2-manifold triangle meshes, the correspondence problem seeks to establish a meaningful mapping between them.
- The mapping can be between the two sets of mesh vertices, between two coarse sets of feature points selected on the meshes, or a continuous one between all points on the two manifolds.

Robust 3D Shape Correspondence in the Spectral Domain

Shape Correspondence Using Spectral Embedding



- Using Geodesic distance as affinity matrix and Gaussian kernel.
- Normalized Eigenvectors of the autocorrelation matrix $R = AA^T$
- A is symmetric, $R = A^2$
- Simply the Eigenvalues of the Affinity matrix as `k-components' as spectrum.
- For correspondence, use the square root of the eigenvalues.

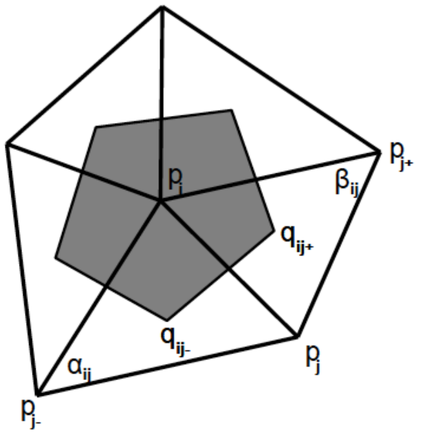
Robust 3D Shape Correspondence in the Spectral Domain

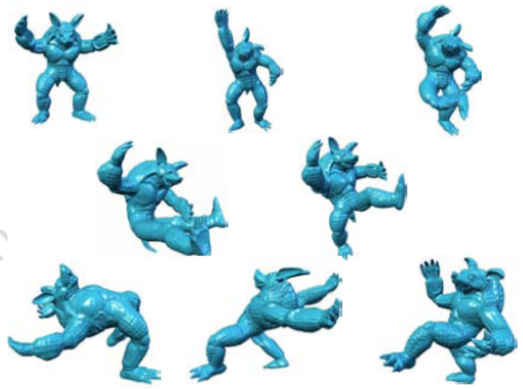
Articulated 3D Models

- Using Affinitiy Martrix A,
- A is Eigen-Decomposed as A = V B V^T B is a diagonal matrix with Eigen Values.
- Geodesic Distance is used as affinity.
- Nystrom Approximation.

A spectral approach to shape-based retrieval of articulated 3D models

Deformation invariant shape representation





discrete Laplace-Beltrami differential operator

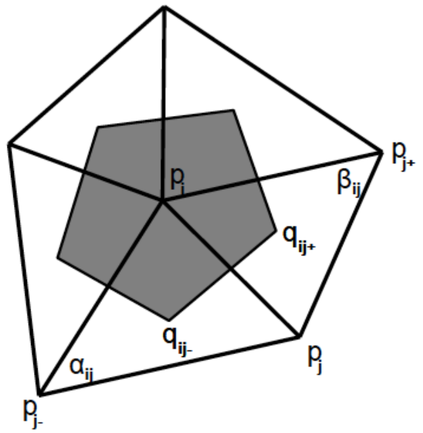
$$\Delta f(p_i) \approx \frac{1}{s_i} \sum_{j \in N(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} [f(p_j) - f(p_i)].$$

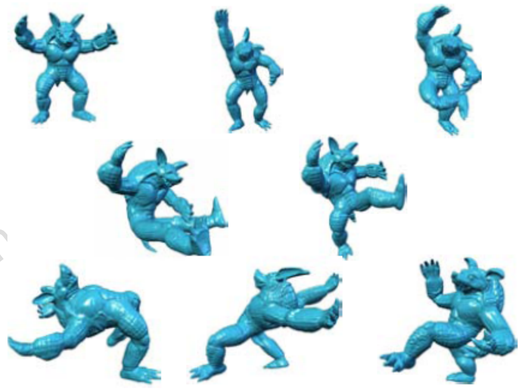
$$m_{ij} = \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2}$$

$$L_{ij} = \begin{cases} \sum_k m_{ik}/s_i & \text{if } i = j, \\ -m_{ij}/s_i & \text{if } i \text{ and } j \text{ adjacent}, \\ 0 & \text{otherwise.} \end{cases}$$

Non-symmetric

Deformation invariant shape representation





$S_{ii} = s_i$, $M_{ij} = m_{ij}$, $L = S^{-1}M$ $\quad M\vec{v} = \lambda S\vec{v}.$ (generalize EVP)

Given a point $\mathbf{p}$ on the surface, we define its *Global Point Signature*, $GPS(\mathbf{p})$, as the infinite-dimensional vector

$$GPS(\mathbf{p}) = \left(\frac{1}{\sqrt{\lambda_1}}\phi_1(\mathbf{p}), \frac{1}{\sqrt{\lambda_2}}\phi_2(\mathbf{p}), \frac{1}{\sqrt{\lambda_3}}\phi_3(\mathbf{p}), \ldots \right),$$

where $\phi_i(\mathbf{p})$ is the value of the eigenfunction ϕ_i at the point

Computation

- Many efficient computational routines available for Eigen decomposition (Lapack)
- Can handle only low number of vertices
- 10000+ vertices, not very suitable
- Require only some of the larges Eigenvalues
- For sparse network, small set of eigenvalues, use the Lanczos method

Spectrum

- If two graphs are isomorphic, they have the same spectrum
- This does not solve the isomorphism problem, as two different graphs may have same spectrum.

Spectrum of A

Spectrum of **A**:

- Positive and negative eigenvalues

$$\sum \lambda_i = 0$$

$$-d_{\max} \le \lambda_0 \le \lambda_1 \ldots \le 0 \le \ldots \lambda_{n-1} \le d_{\max}$$

$$\lambda_{n-1} \ge -\lambda_0$$

Eigenvectors:

- Perron-Frobenius Theorem (**A** non-negative matrix)
  - λ_{n-1} is largest magnitude eigenvalue

Corresponding Eigen vector is non-negative)

Spectrum of L

Spectrum of **L**

- **L** positive semi-definite

$$\sum \lambda_i = 2|E|$$

$$0 = \lambda_0 \leq \lambda_1 \ldots \leq \lambda_{n-1} < n$$

- There always exists an eigenvector **1** with eigenvalue 0
  - Because of zero row-sums
- The number zeros in the spectrum is the number of disconnected components of the graph.

Spectrum of Normalized L

Spectrum of $\hat{\mathbf{L}}$

- $\hat{\mathbf{L}}$ positive semi-definite

$$\sum \lambda_i = |V|$$

$$0 = \lambda_0 \leq \lambda_1 \ldots \leq \lambda_{n-1} \leq 2$$

- As with Laplacian, the number zeros in the spectrum is the number of disconnected components of the graph.
- Eigenvector exists with eigenvalue 0 and entries

$$\left(\sqrt{d_1} \quad \sqrt{d_2} \quad \ldots \quad \sqrt{d_n}\right)^T$$

- 'scale invariance'

Info from Spectrum

- The Laplacians are positive semidefinite with smallest eigenvalue 0
  - Normalized Laplacian has max eigenvalue 2
- Sp(**L**) for a graph of disconnected components is the union of the spectra of all the components
  - Hence the number of zero eigenvalues counts the number of components
- For regular graphs, the spectrum of **A** and **L** directly related

$$\mathbf{D} = k\mathbf{I}$$

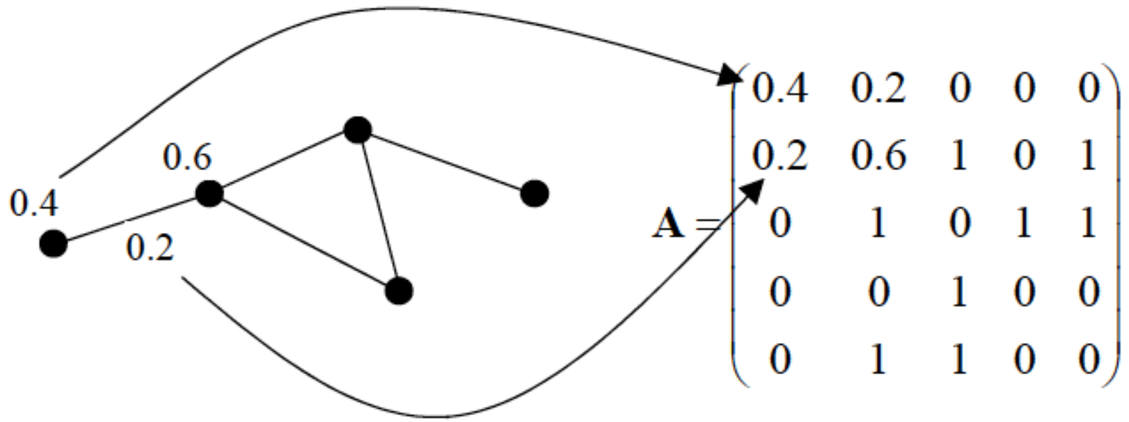
$$\mathbf{L} = k\mathbf{I} - \mathbf{A}$$

$$\lambda_L = k - \lambda_A$$

- Smallest eigenpair of **A** becomes largest of **L**

Coding attributes

- Currently considered edges as present or absent {0,1}
- We can add more edge info – weights
- Diagonal entries can be used to encode vertices.



Spectral features

All graphs which have simple spectra can be distinguished from each other in polynomial time

- Simple spectrum means than there are no repeated eigenvalues in the spectrum
- Hence the eigendecomposition is unique
- Then we can order the components of the eigenvectors in polynomial time
  - For example by sorting
- Comparison then determines if they are isomorphic

Laplacian vs LLE

- More similar than different
  - Graph-based, spectral method
  - Sparse eigenvalue problem
  - Similar results in practice
- Essential differences
  - Preserves locality vs local linearity
  - Uses graph Laplacian

$$L = D - W \quad \text{(unnormalized)}$$
$$\mathcal{L} = I - D^{-1/2} W D^{-1/2} \quad \text{(normalized)}$$

Applications – Normalized cut

Normalized cut

$$\text{Ncut}(P,Q) = \frac{\text{cut}(P,Q)}{\text{assoc}(P,V)} + \frac{\text{cut}(P,Q)}{\text{assoc}(Q,V)}$$

$$\text{assoc}(P,V) = \sum_{u \in P, v \in V} A_{uv}$$

Adjacency - NCut

$$a_{ij} = \exp\left(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2\sigma_1^2}\right) \times \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_2^2}\right) & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < R, \\ 0 & \text{otherwise.} \end{cases}$$

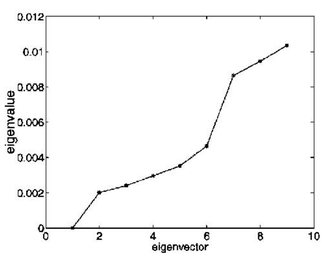
F's – feature vectors such as intensity, color, texture
X's – position of the pixel

Ncut (Contd)

(1) Construct the graph G starting from the data set X calculating the adjacency between patterns using Eq. (74).
(2) Compute the degree matrix D.
(3) Construct the matrix $L_{\mathrm{N}} = D^{-1/2} L D^{-1/2}$.
(4) Compute the eigenvector $\mathbf{e}_2$ associated to the second smallest eigenvalue λ_2.
(5) Use $D^{-1/2}\mathbf{e}_2$ to segment G.

The Eigenvector corresponding to the second smallest Eigenvalue is Called Fiedler Vector.

Normalized cut and image segment

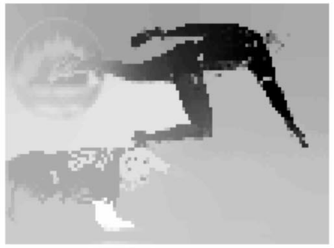


(a)

(b)

(c)

(d)



(e)

(f)

(g)

(h)

(i)

Subplot (a) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplots (b)-(i) show the eigenvectors corresponding the second smallest to the ninth smallest eigenvalues of the system.

NJW Algorithm

(1) Compute the affinity matrix $A \in \mathbb{R}^{n \times n}$:

$$a_{ij} = \begin{cases} \exp\left(-\dfrac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

(2) Construct the matrix D.

NJW Algorithm

(3) Compute a normalized version of A, defining this Laplacian:

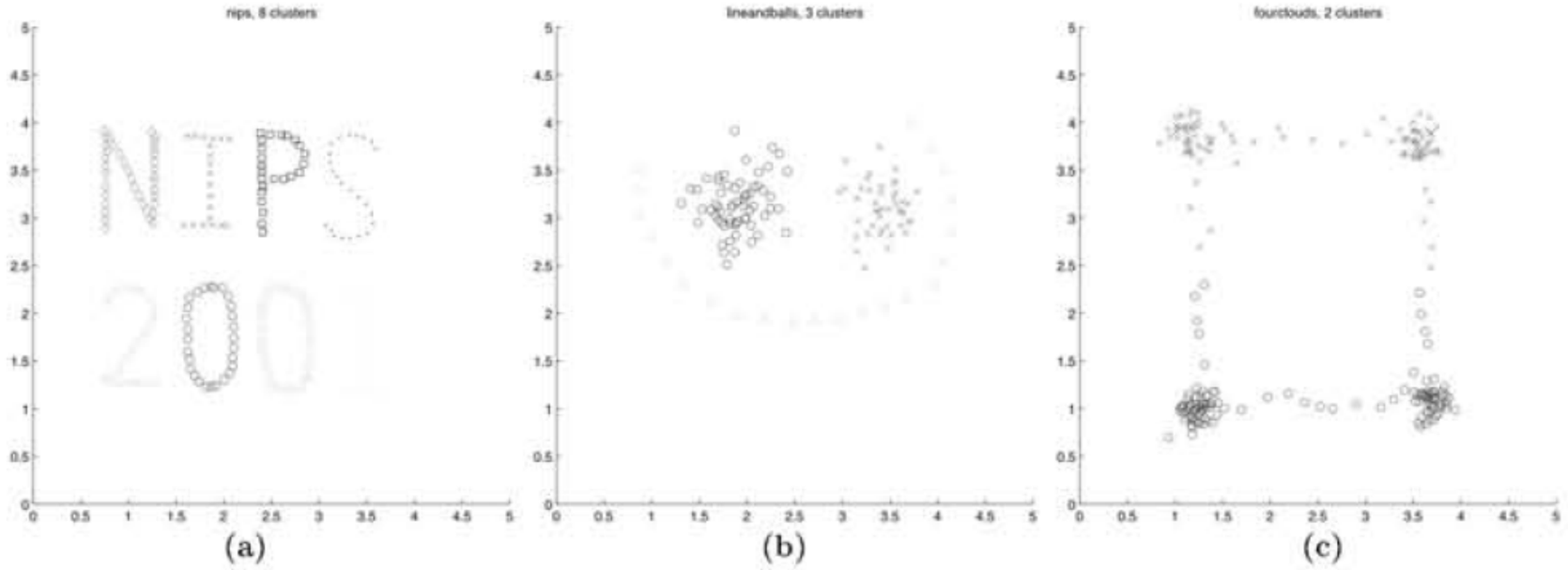
$$L = D^{-1/2} A D^{-1/2}. \tag{76}$$

(4) Find the k eigenvectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ of L associated to the largest eigenvalues $\{\lambda_1, \ldots, \lambda_k\}$.
(5) Form the matrix Z by stacking the k eigenvectors in columns.
(6) Compute the matrix Y by normalizing each of the Z's rows to have unit length:

$$y_{ij} = \frac{z_{ij}}{\sum_{r=1}^{k} z_{ir}^2}. \tag{77}$$

In this way all the original points are mapped into a unit hypersphere.
(7) In this new representation of the original n points apply a clustering algorithm that attempts to minimize distortion such as K-means.

NJW Algorithm (contd)



Markov Random Walks

- Using Stochastic matrix $P = D^{-1}A$
- P_{ij} represent the probability of moving from node i to node j
- Solve the eigenvalue problem $Px = \lambda x$
- *L* and *P* have the same solution with eigenvalues of *P* equal to $1 - \lambda_i$ where λ_i are the eigenvalues of *L*.

The eigenvalues of P are $\lambda_1 = 1 \geq \lambda_2 \geq \ldots \lambda_n \geq -1$;

Differential equations

- Diffusion, or heat flow

$$\frac{\partial p}{\partial t} = \nabla^2 p$$

- Wave propagation

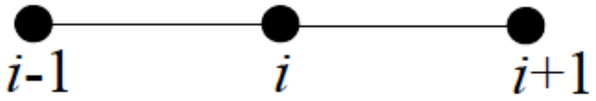
$$\frac{\partial^2 p}{\partial t^2} = -\nabla^2 p$$

- Schrödinger Equation

$$i\hbar \frac{\partial p}{\partial t} = -\frac{\hbar^2}{2m} \nabla^2 p + Vp$$

Laplacian

- A graph which encodes the neighbourhood structure



- The Lapacian of this graph is $\mathbf{L} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$
- Apply **L** to a vector (a 'function' taking values on the vertices) $(\mathbf{Lp})_i = -p_{i+1} - p_{i-1} + 2p_i$

$$\approx -\nabla^2 p$$

- So the graph Laplacian is a discrete representation of the calculus Laplacian
  - Vertices are points in space
  - Edges represent neighbourhood structure of space

Diffusion

- On a network, we identify the Laplacian operator ∇^2 with the Laplacian of the network **L**

$$\nabla^2 \longrightarrow -\mathbf{L}$$

- Discrete space, continuous time diffusion process

$$\frac{\partial p}{\partial t} = \nabla^2 p$$

$$\downarrow$$

$$\frac{\partial \mathbf{p}}{\partial t} = -\mathbf{L}\mathbf{p}$$

Non-linear methods

- Underlying framework
  - Derive sparse graph
  - Derive matrix from graph weights
  - Embedding from Eigenvectors.
- Key difference
  - Algorithm differs in formulation of graph weights
  - Shortest paths, geodesic path, least square fits, semidefinite programming, etc.

Extra slides

Dimensionality reduction

- Given a set X1, X2, …. , Xk in Rl, find y1, .., yk in Rm (m <<< l) s.t. yi represents xi.
- Graph embedding is done by computing Eigenvectors of Graph Laplacian.
  - Construct Adjacency Graph
  - Choosing Weights
  - Eigenmaps constructed using generalized Eigenvector problem.

- Principal component analysis (PCA), Multidimensional Scaling (MDS)

Spectral graph theory is a branch
in mathematics which aims to characterise the properties of
unweighted graphs using the eigenvalues and eigenvectors
of the adjacency matrix or the closely related Laplacian matrix
[16]

Spectral Embedding of Graphs

Algorithm

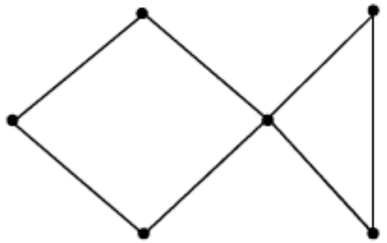
- Three steps
  - Identify k-nearest neighbors
  - Assign weights $W_{ij} = 1 \text{ or } W_{ij} = \exp\left(-\beta \left\| \vec{x}_i - \vec{x}_j \right\|^2\right)$
  - Compute outputs by minimizing: $\Psi(y) = \sum_{ij} \frac{W_{ij} \left\| \vec{y}_i - \vec{y}_j \right\|^2}{\sqrt{D_{ii} D_{jj}}} \text{ where } D_{ii} = \sum_j W_{ij}$

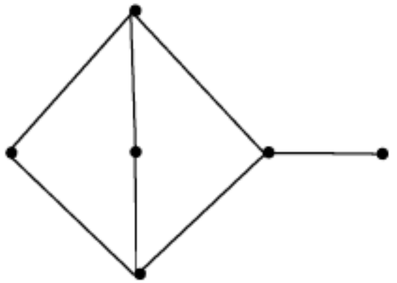
(sparse eigenvalue problem as in LLE)

Spectrum

- These two graphs have the same spectrum using the Laplacian representation

[5.24]
$[3]^2$
2
0.76





[5.24]
$[3]^2$
2
0.76

- This is a *cospectral* pair
- Necessary but not sufficient...
- The matrix representation we use has a big effect on how many of these cospectral graphs there are

Define partition vector $\mathbf{x}$ such that

$$x_u = \begin{cases} +1 \text{ if } u \in P \\ -1 \text{ if } u \in Q \end{cases}$$

Then $$\text{Ncut}(\mathbf{x}) = \frac{\sum_{x_u>0, x_v<0} -A_{uv} x_u x_v}{\sum_{x_u>0, v \in V} A_{uv}} + \frac{\sum_{x_u<0, x_v>0} -A_{uv} x_u x_v}{\sum_{x_u<0, v \in V} A_{uv}}$$

With a bit of transformation we can turn this into a matrix form

$$\text{Ncut}(\mathbf{y}) = \frac{\mathbf{y}^T(\mathbf{D}-\mathbf{A})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}$$

$$y_u \in \{1, -b\}$$

$$\mathbf{y}^T\mathbf{D}\mathbf{1} = 0$$

And we should try to minimise Ncut to find the best partition

- As it is, the problem is hard because **y** is discrete
- Take the relaxation of the problem, i.e. **y** allowed to take real values
- Solution is easily given by solving the eigenvalue problem

$$(\mathbf{D}-\mathbf{A})\mathbf{y}=\lambda\mathbf{D}\mathbf{y}$$

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{A})\mathbf{D}^{-\frac{1}{2}}\mathbf{z}=\lambda\mathbf{z} \qquad (\mathbf{z}=\mathbf{D}^{\frac{1}{2}}\mathbf{y})$$

$$\left(\mathbf{I}-\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{z}=\lambda\mathbf{z}$$

$$\hat{\mathbf{L}}\mathbf{z}=\lambda\mathbf{z}$$

- Hence the solution is an eigenvector of the normalized Laplacian

- If we want the smallest Ncut, then we should choose the eigenvector with smallest eigenvalue
- 0 is an eigenvalue of $\hat{\mathbf{L}}$, with corresponding eigenvector

$$\mathbf{u}_0 = \left(\sqrt{d_1} \quad \sqrt{d_2} \quad \ldots \quad \sqrt{d_n}\right)^T$$

  - But $\mathbf{z}=\mathbf{u}_0$ does not satisfy condition $\mathbf{y}^T\mathbf{D1} = 0$

$$\mathbf{y}^T\mathbf{D1} = \mathbf{z}^T\mathbf{D}^{-\frac{1}{2}}\mathbf{D1} = \mathbf{1}^T\mathbf{D}\ \mathbf{1} \neq 0$$

- However the eigenvector with second smallest eigenvalue does satisfy this condition
- This is called the *Fiedler vector* and gives an approximate solution to min normalized cut $\mathbf{y}^T\mathbf{D}\mathbf{1} = 0$

$$\mathbf{z} = \mathbf{x}_1$$

$$\mathbf{y} = \mathbf{D}^{-1/2}\mathbf{z}$$

- The sign of the components of the Fiedler vector gives the partition
  - Eg $\mathbf{x}_1 = \begin{pmatrix} -0.2 & 0.1 & 0.3 & -0.6 \end{pmatrix}$

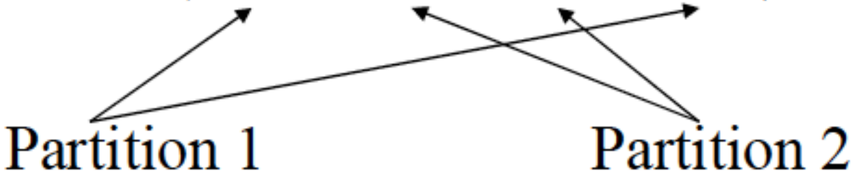
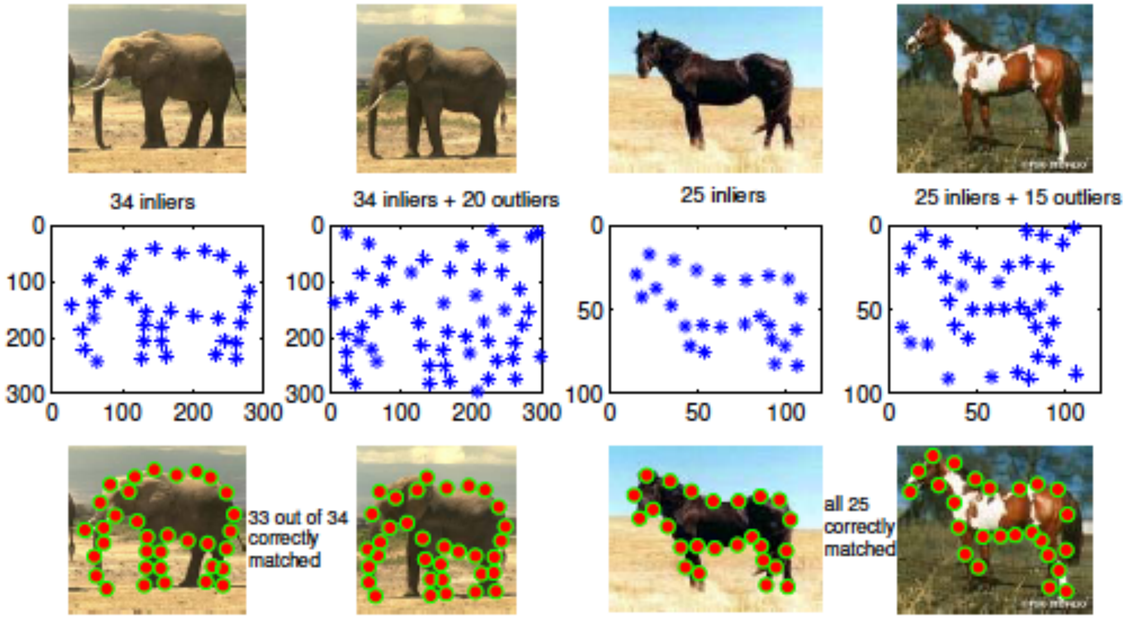


Image correspondence



finding consistent correspondences between two sets of features, such as object recognition,
shape matching, wide baseline stereo, 2D and 3D registration.

A Spectral Technique for Correspondence Problems Using Pairwise Constraints

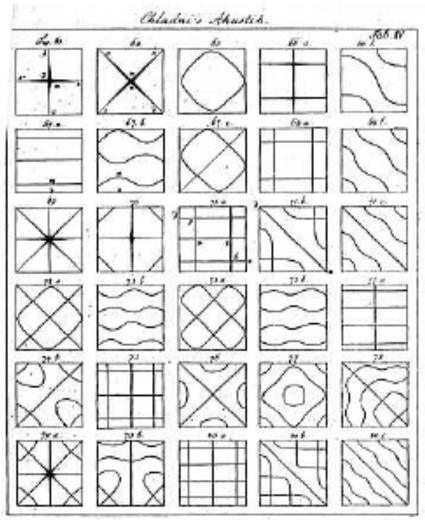
Laplace Spectra

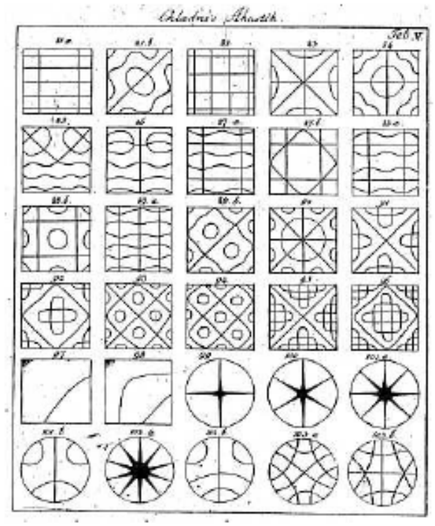
- continuous Laplace–Beltrami operator - interpret images as Riemannian manifolds.

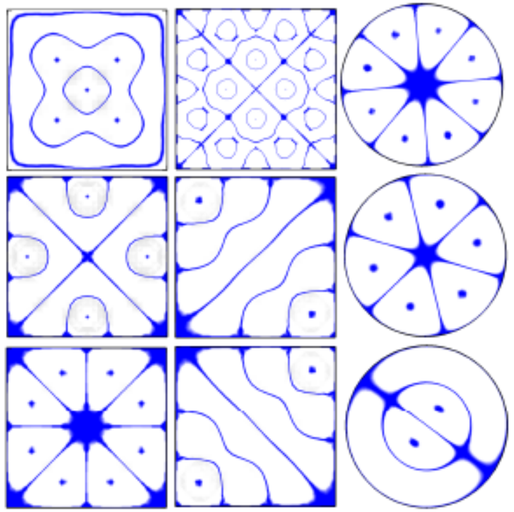
Laplace Beltrami Eigen Functions

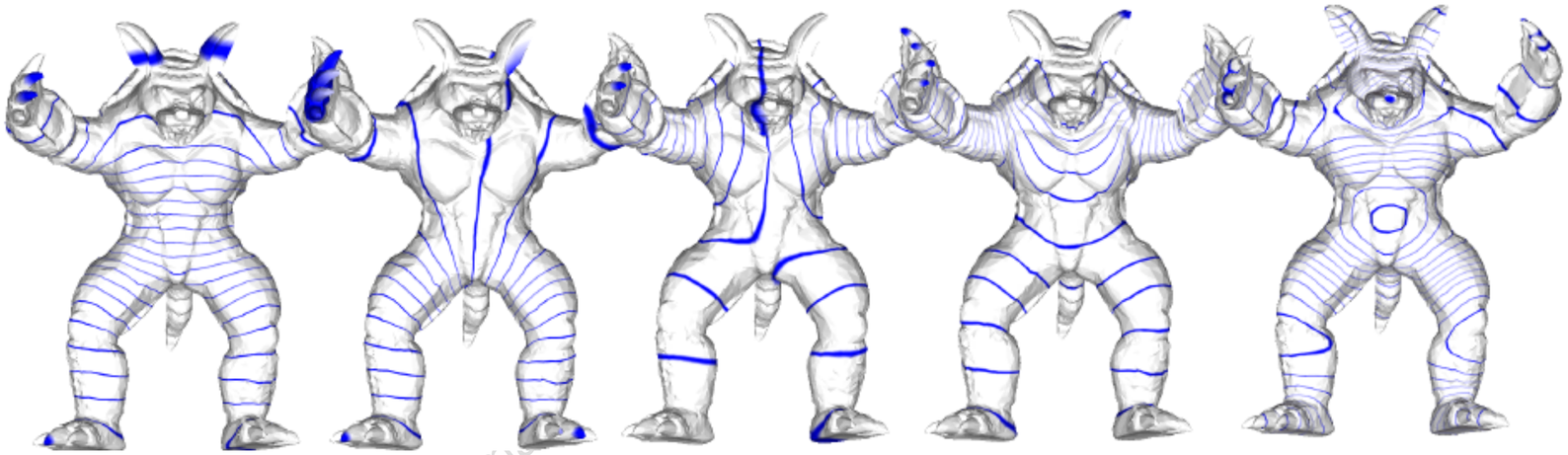
The Graph Laplacian $L = (a_{i,j})$ is a matrix defined by:

$$
\begin{aligned}
a_{i,j} &= w_{i,j} > 0 && \text{if } (i,j) \text{ is an edge} \\
a_{i,i} &= -\sum w_{i,j}) && \\
a_{i,j} &= 0 && \text{otherwise}
\end{aligned}
$$









Heat Kernel

- Solution

$$\boxed{\begin{aligned} p(t) &= \mathbf{H}(t)\,p(0) \\ \mathbf{H}(t) &= \exp(-\mathbf{L}t) \end{aligned}}$$

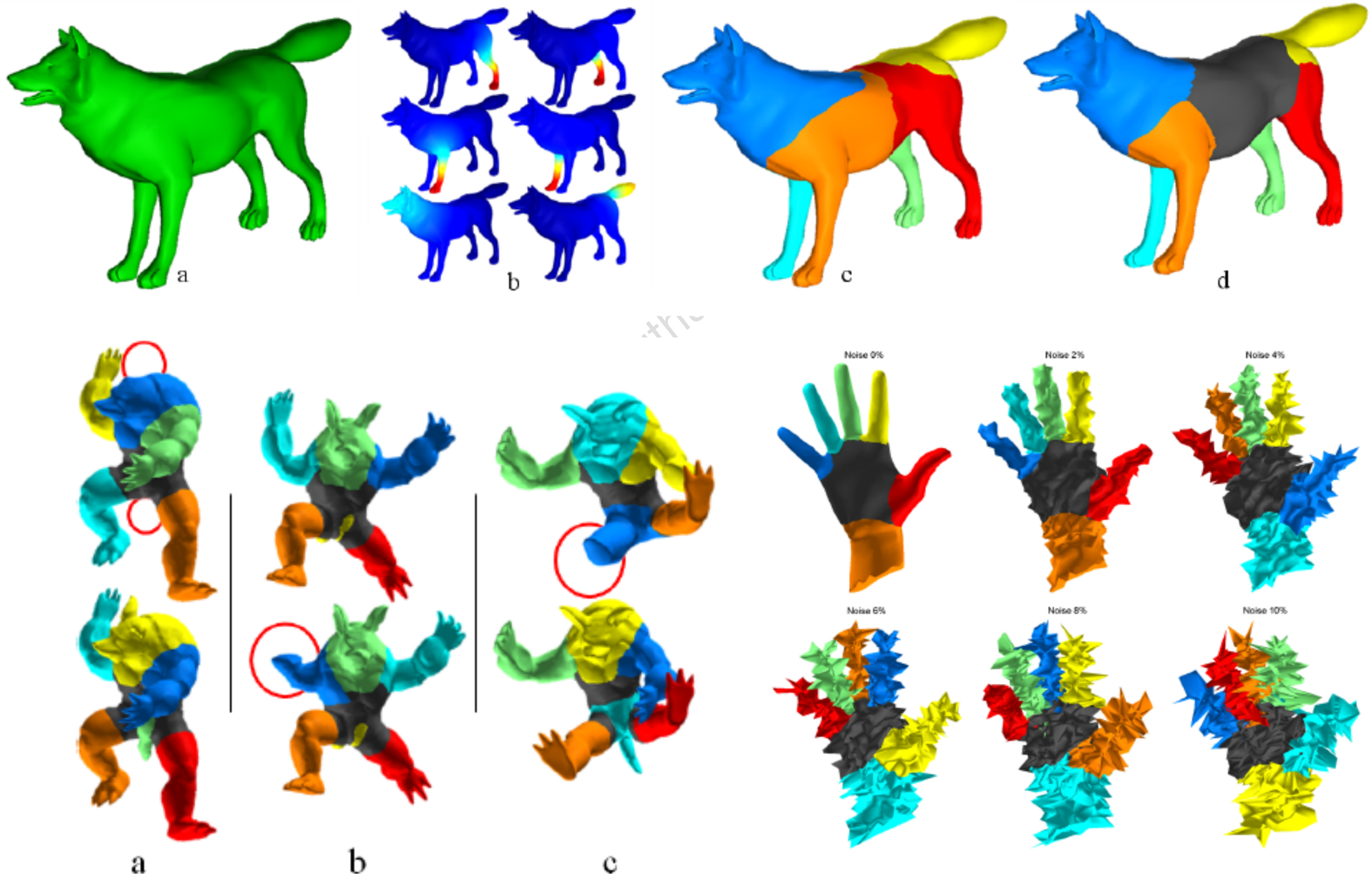
$$\frac{\partial}{\partial t}\mathbf{H}(t)\mathbf{p}(0) = \left(\frac{\partial}{\partial t}\mathbf{H}(t)\right)\mathbf{p}(0)$$

$$= \left(\frac{\partial}{\partial t}\exp(-\mathbf{L}t)\right)\mathbf{p}(0)$$

$$= -\mathbf{L}\exp(-\mathbf{L}t)\mathbf{p}(0) = -\mathbf{L}\mathbf{H}(t)\mathbf{p}(0)$$

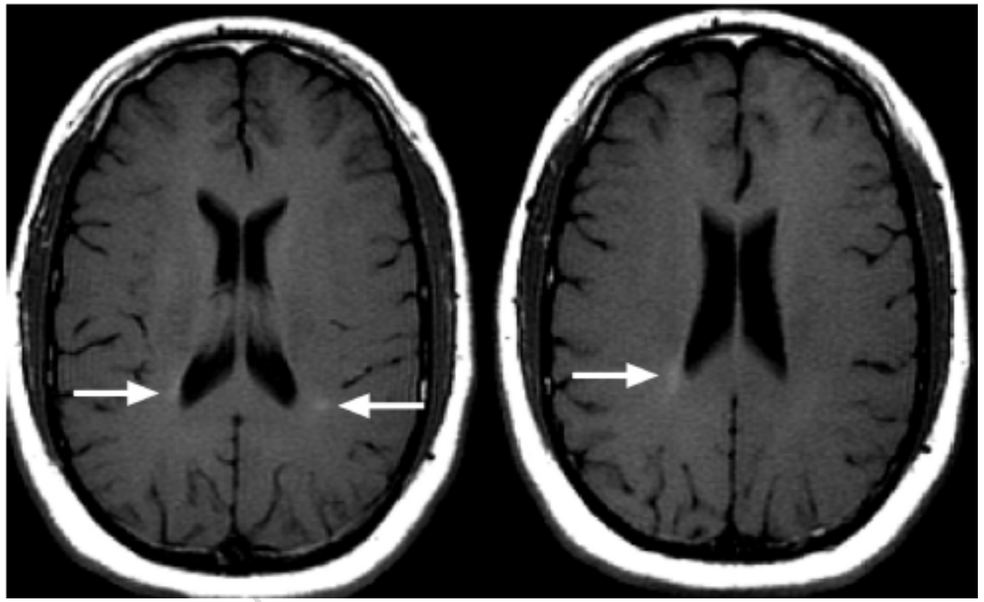
- Heat kernel $\mathbf{H}(t)$
- $H_{ij}(t)$ describes the amount of heat flow from vertex i to j at time t
- Essentially another matrix representation, but can vary time to get different representations

Heat Kernel Signature (HKS)



Heat Walk: Robust Salient Segmentation of Non-rigid Shapes

Deep Learning – MS detection



Multiple Sclerosis is an inflammatory disease in which the insulating covers of nerve cells in the brain and spinal cord are damaged