universitätbonn
Rheinische
Friedrich-Wilhelms-
Universität Bonn

Chapter 4:
Dimensionality Reduction

Jun.-Prof. Dr.-Ing. Thomas Schultz

URL: http://cg.cs.uni-bonn.de
E-Mail: schultz@cs.uni-bonn.de
Office: Friedrich-Ebert-Allee 144, 53113 Bonn

November 22, 2016

Dimensionality Reduction

Goal: Embed high-dimensional data into a low-dimensional space, in a way that important structure is preserved

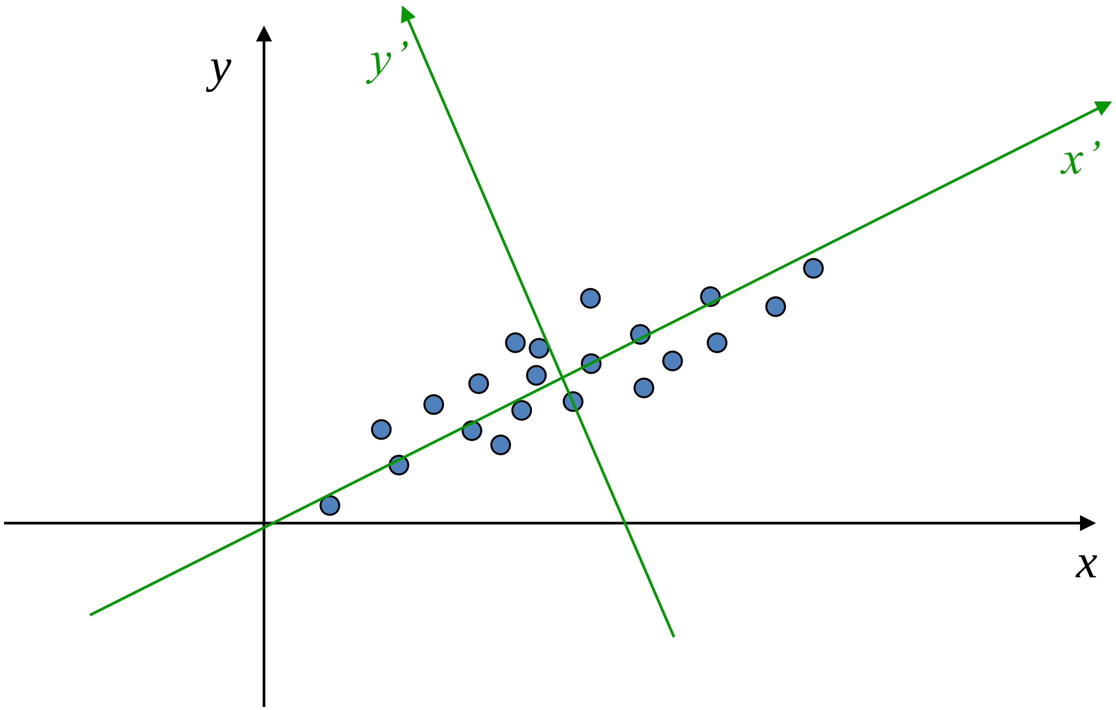
We will cover four methods:

- **PCA:** Brief recap, you should already know this
- **Kernel PCA:** Deal with nonlinear structures
- **MDS:** Finds a low-dimensional embedding based on distances alone
- **ISOMAP:** Allows us to "unroll" nonlinear manifolds

Section 4.1:
(Kernel) Principal Component Analysis

PCA – the general idea

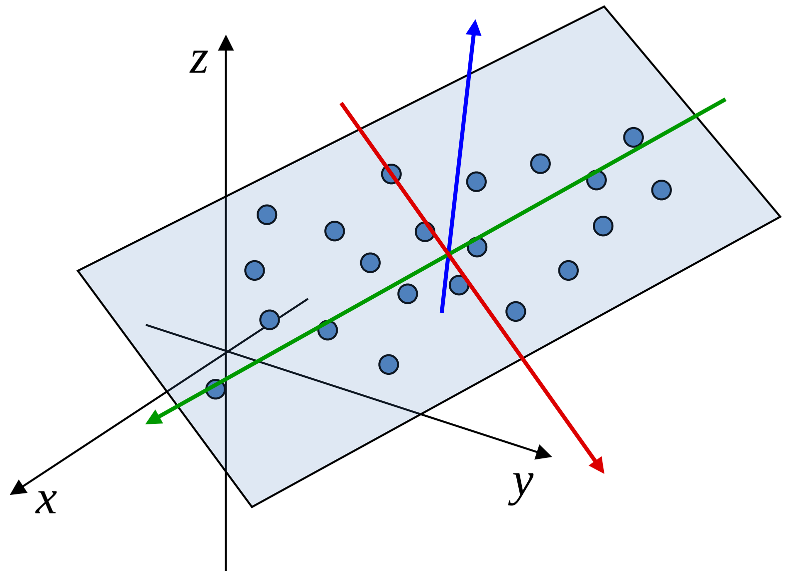
- PCA finds an orthogonal basis that best represents a given data set.



The sum of squared distances from the x' axis is minimized.

PCA – the general idea

- PCA finds an orthogonal basis that best represents a given data set.

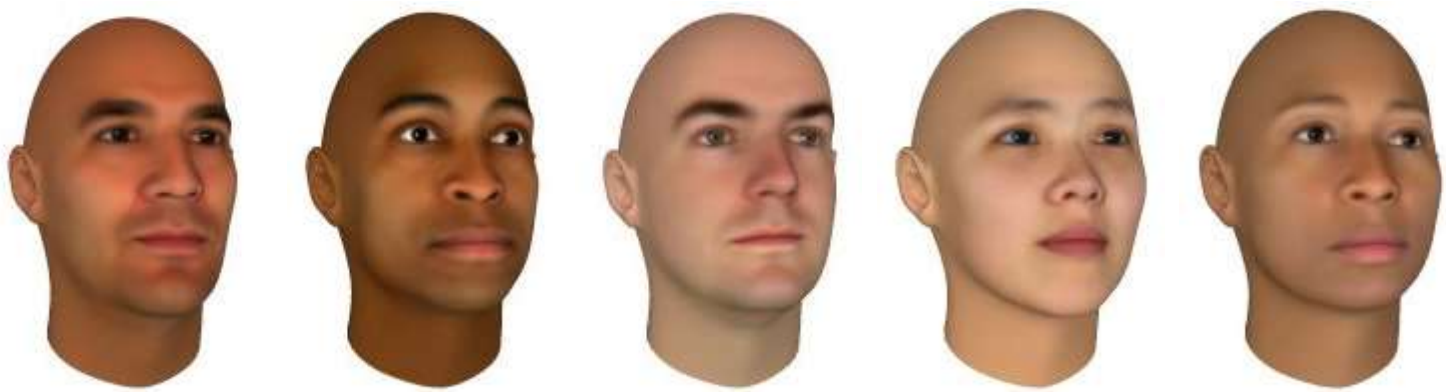


3D point set in standard basis

PCA finds a best approximating plane (again, in terms of Σdistances^2)

Managing high-dimensional data

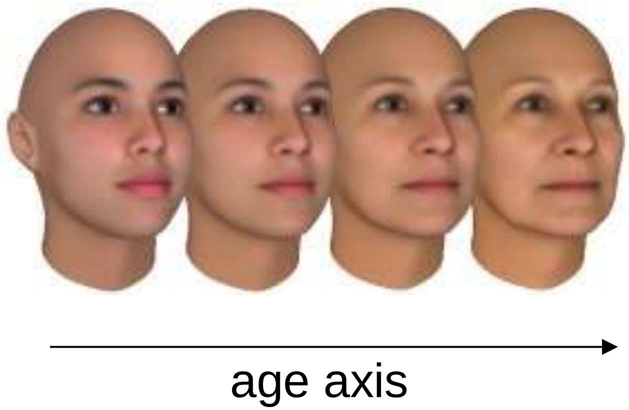
- Example: Data base of face scans (3D geometry + texture)

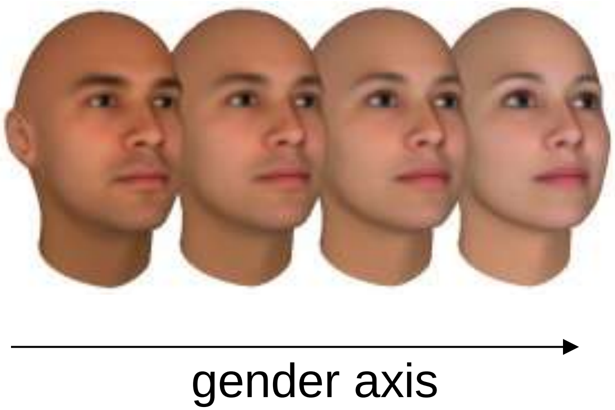


  - 10,000 points in each scan
  - x, y, z, R, G, B – 6 numbers for each point
  - Thus, each scan is a 10,000*6 = **60,000-dimensional vector**!

Managing high-dimensional data

- How to find interesting axes is this 60,000-dimensional space?
  - axes that measures age, gender, etc…
  - There is hope: the faces are likely to be governed by a small set of parameters (much fewer than 60,000…)





Goals of Principal Component Analysis

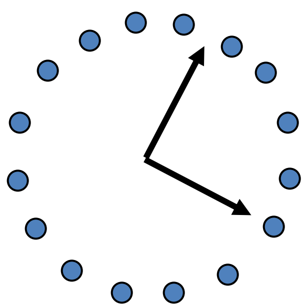
- Given input $\mathbf{x}_i \in \mathbb{R}^p, i = 1,2, \ldots, n$, learn mapping from $\mathbf{x}$ to $\mathbf{y} = \mathrm{U}_k^{\mathrm{T}}(\mathbf{x} - \bar{\mathbf{x}})$
  - **Approximation:** $\mathrm{U}_k \in \mathbb{R}^{p \times k}$ with $k \leq p$
    - Select U_k such that the approximation error $\sum_i \left\| \left(\mathrm{I} - \mathrm{U}_k \mathrm{U}_k^{\mathrm{T}}\right)(\mathbf{x}_i - \bar{\mathbf{x}}) \right\|^2$ is minimal
    - Select U_k so that the variance in the projected data remains maximal
  - **Decorrelation:**
    - Coefficients of **y** are uncorrelated
    - If $k = p$, pure rotation, no approximation

Algorithm: PCA

- **Input:** $\mathbf{x}_i \in \mathbb{R}^p, i = 1,2, \dots, n$
- **Output:** $\mathrm{U}_k \in \mathbb{R}^{p \times k}$ with $k \leq p$ such that $\mathbf{y} = \mathrm{U}_k^{\mathrm{T}}(\mathbf{x} - \bar{\mathbf{x}})$ with $\bar{\mathbf{x}} = \frac{1}{n}\sum_i \mathbf{x}_i$
- **Algorithm:**
  - **Center** the points, $\mathbf{z}_i = \mathbf{x}_i - \bar{\mathbf{x}}$
  - Form $p \times n$ **centered data matrix** $\mathrm{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n]$
  - Form $p \times p$ **scatter matrix** $\mathrm{S} = \mathrm{Z}\mathrm{Z}^{\mathrm{T}}$
  - Compute spectral decomposition $\mathrm{S} = \mathrm{U}\Lambda\mathrm{U}^{\mathrm{T}}$
    - Sort coefficients of Λ in decreasing order
  - Form U_k from *k* leading columns of U
- I assume you have seen the derivation previously

Principal Components

- Eigenvectors that correspond to big eigenvalues are the directions in which the data has strong components (= large variance).
- If the eigenvalues are more or less the same, there is no main direction.

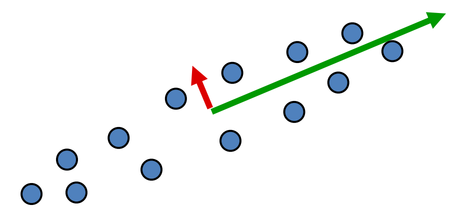


There is no preferred direction.

S looks like this:

$$\begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix}$$

Any vector is an eigenvector



There is a clear preferred direction.

S looks like this:

$$U \begin{pmatrix} \lambda & \\ & \mu \end{pmatrix} U^T$$

μ is close to zero, much smaller than λ.

Example: Biological Shape Analysis

- **Data:** CT images of 48 mouse mandibles
- **Scientific question:** Impact of factors such as evolutionary history, diet, or geography on skeleton shape

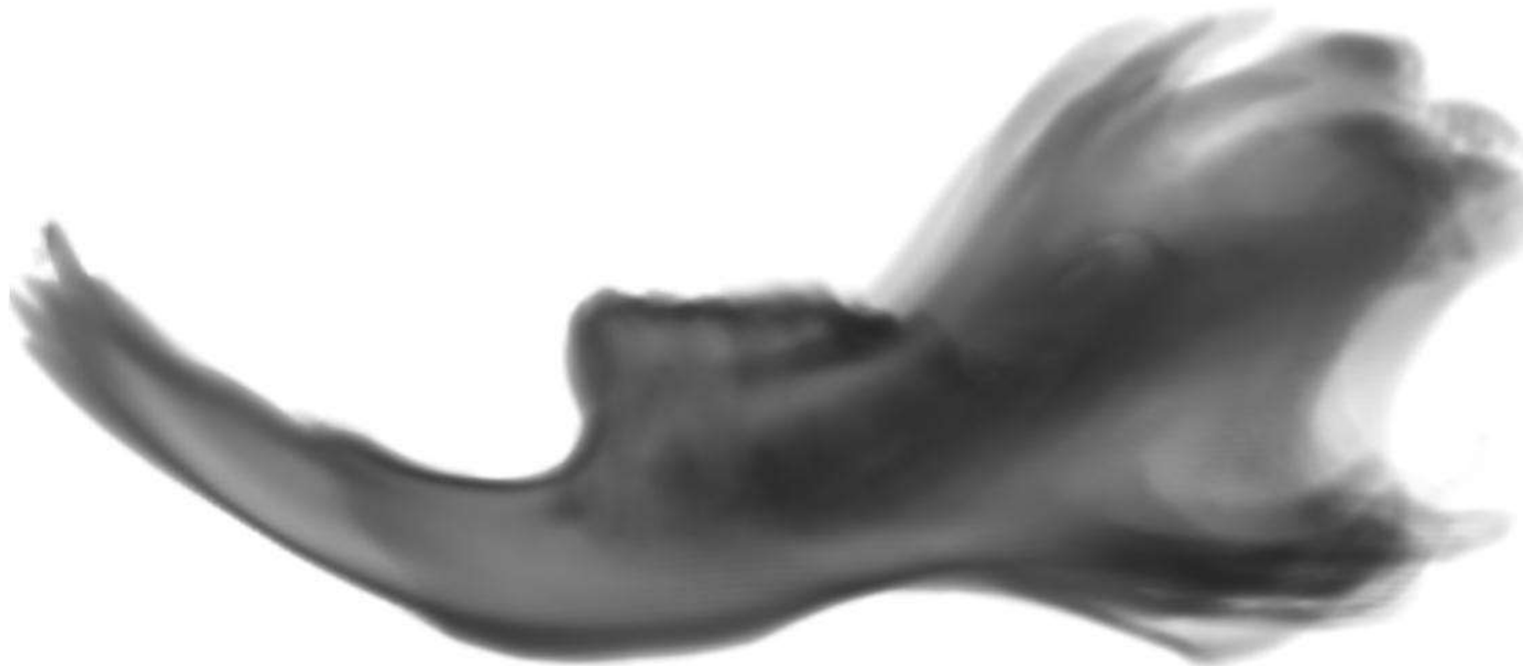
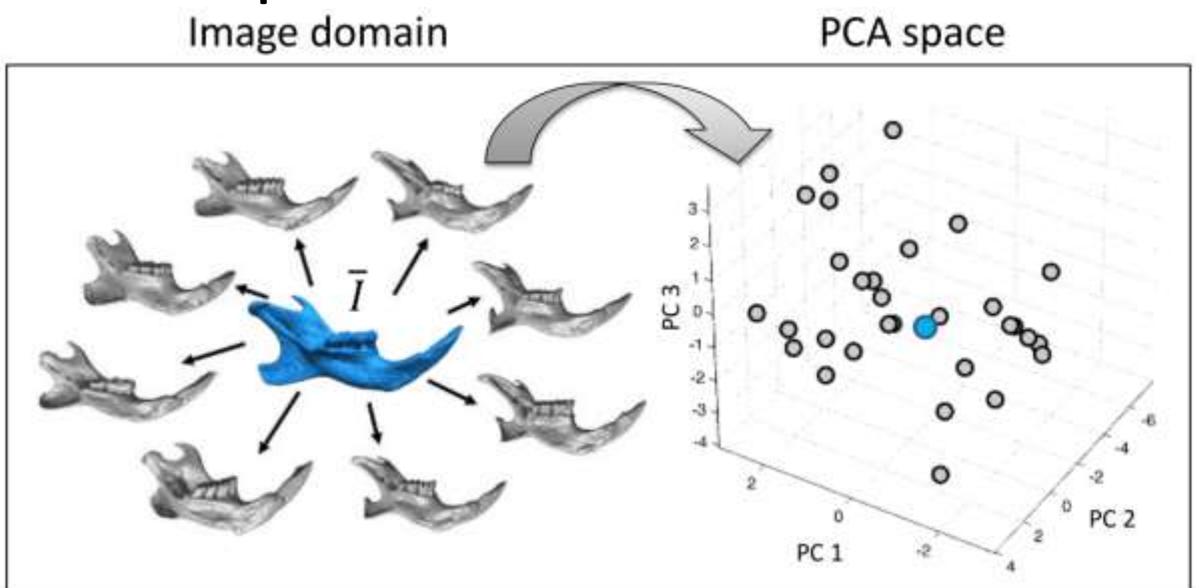


Image by Max Hermann

Linear Shape Spaces

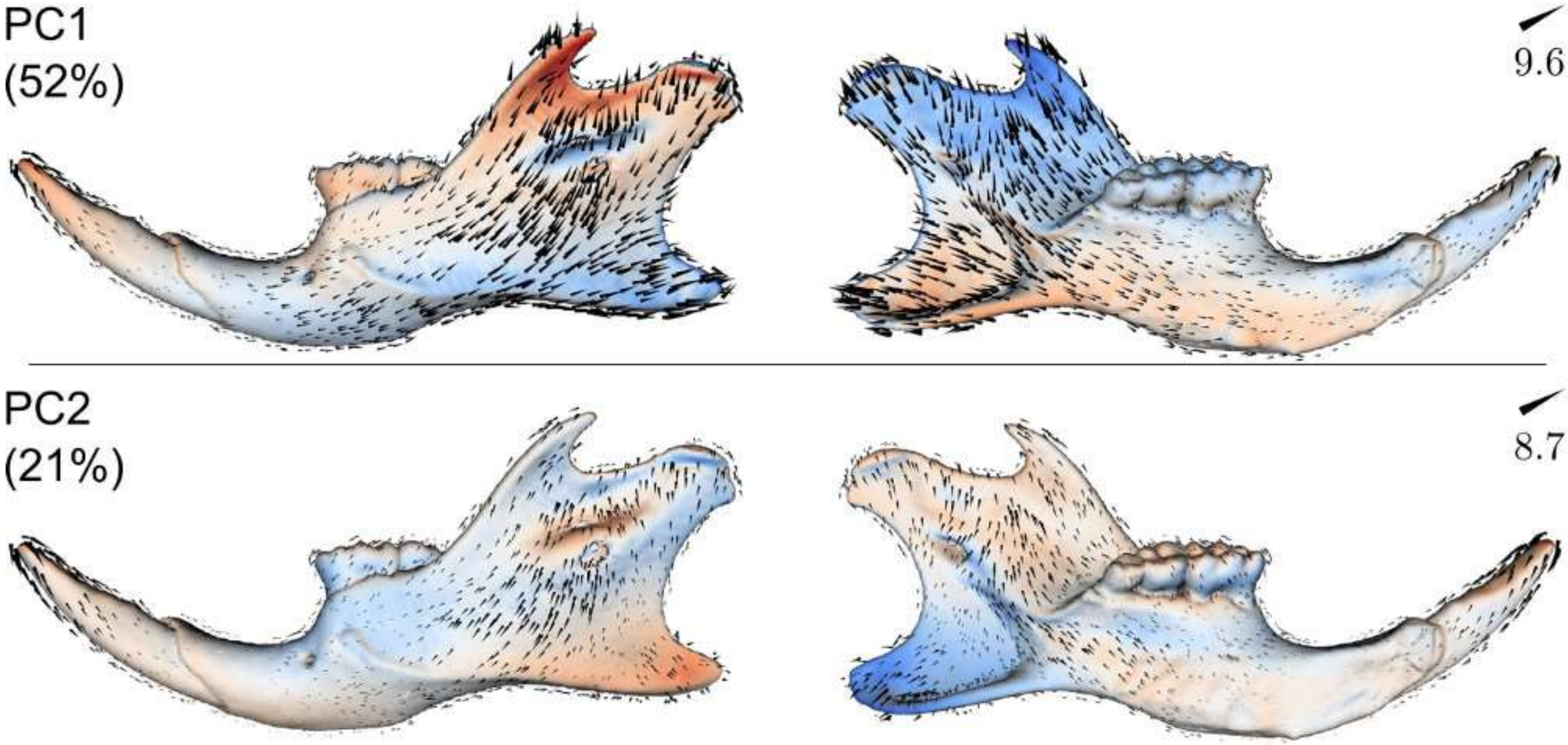
- **Align** all shapes, compute a **mean shape**
- Express each shape as transformation of the mean
  - **Shape vector:** Concatenation of displacement vectors
- Perform PCA on shape vectors to identify **principal modes** of shape variation



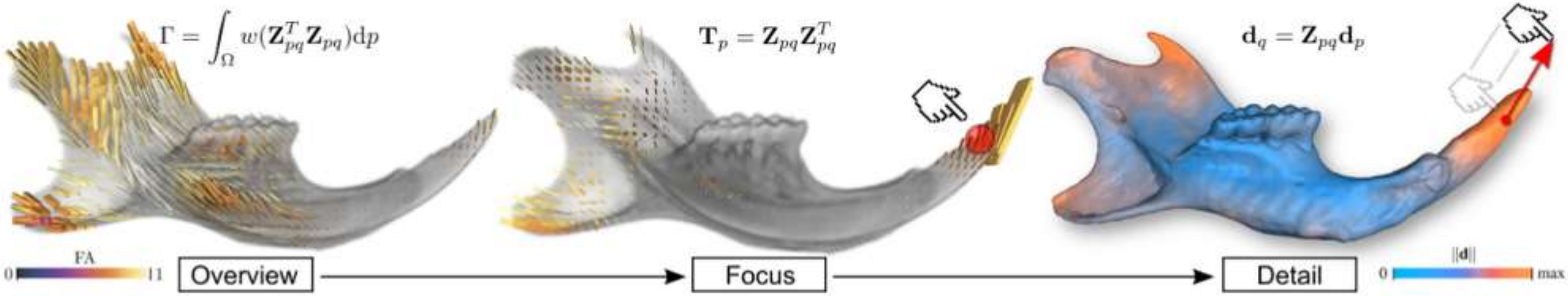
Images by Max Hermann

Linear Shape Space: Example

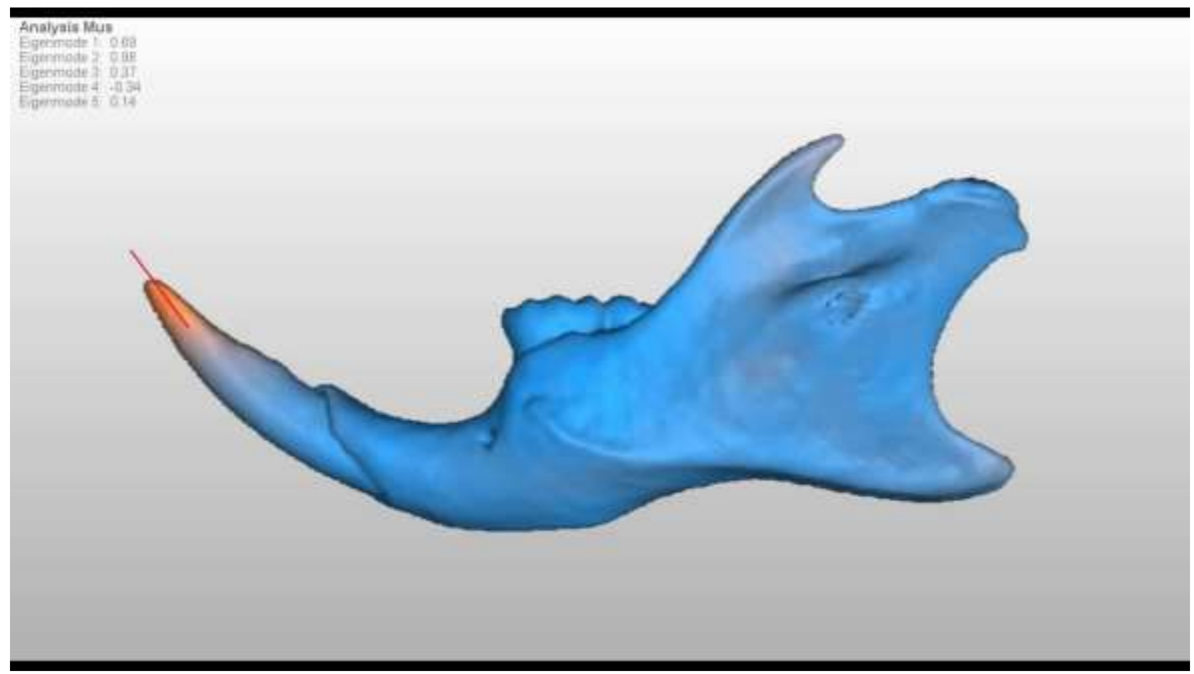
- Normal component: Color
- Tangential component: Arrow glyph



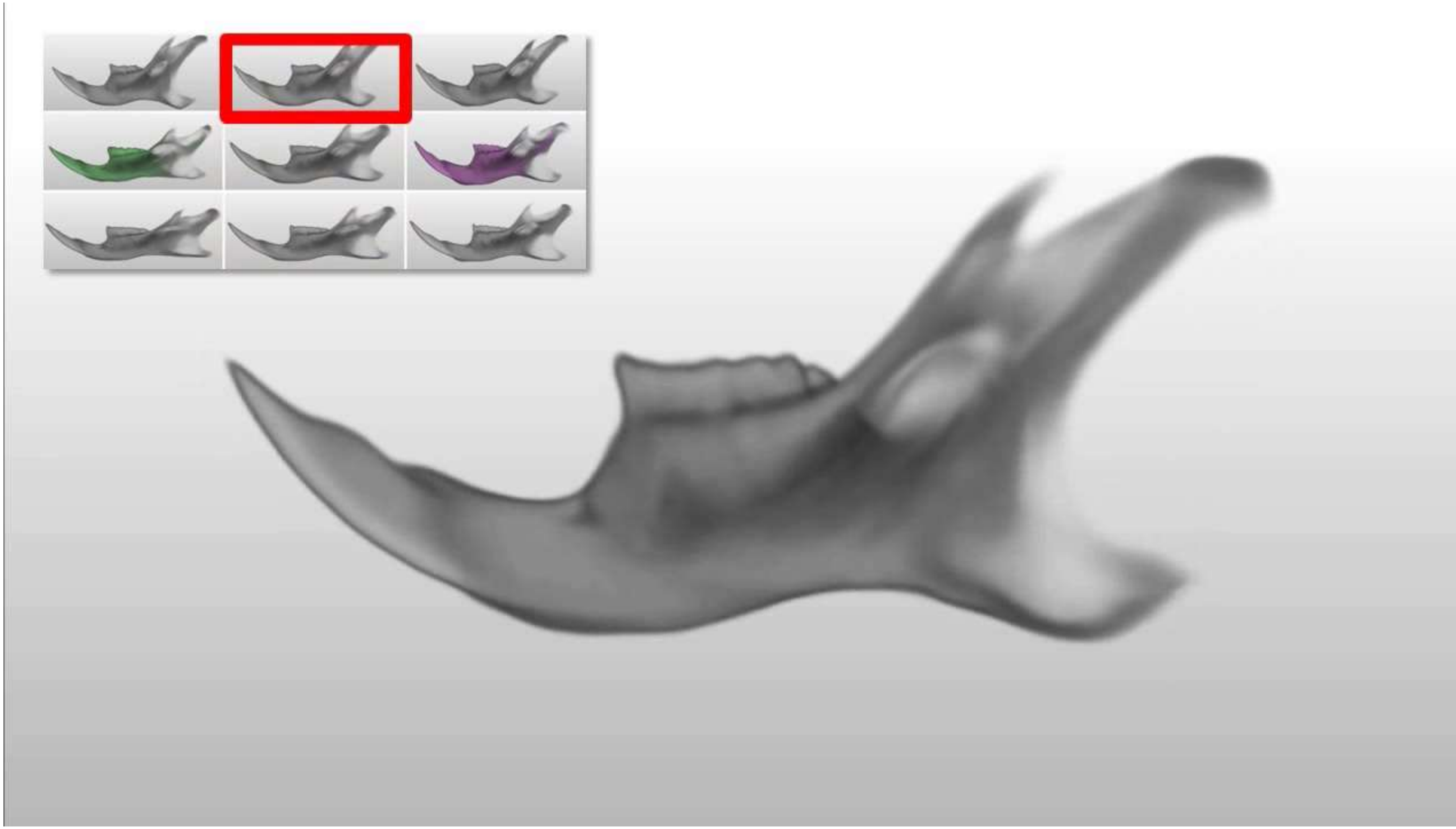
Exploring Shape Spaces



Hermann et al.:
„A Visual Analytics Approach to Study Anatomic Covariation.“
PacificVis 2014



Group Analysis

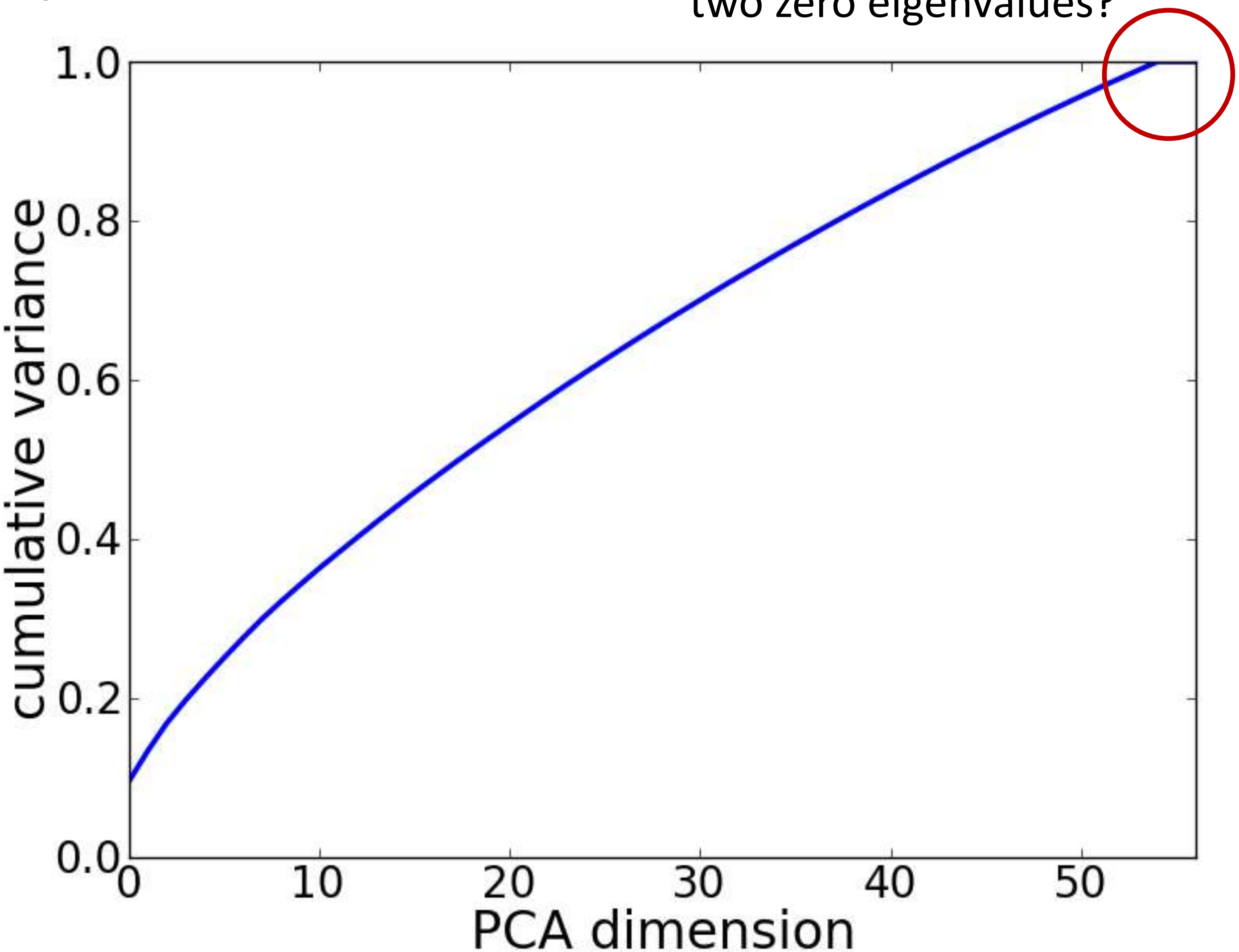


Hermann et al.: „Accurate Interactive Visualization of Large Deformations and Variability in Biomedical Image Ensembles.“ IEEE SciVis 2015

Quiz: Interpreting PCA Result

n=56, p>100,000

two zero eigenvalues?



What if *n*<*p*?

- **Maximum rank** of Z and S is $\min(n-1, p)$
  - Linear dependence introduced by centering
  - It does not make sense to use $k > n-1$
- All eigenvectors of S are in the span of $\mathbf{z}_i$:

$$\lambda_a \mathbf{u}_a = \mathrm{S}\mathbf{u}_a = \sum_i \mathbf{z}_i \mathbf{z}_i^T \mathbf{u}_a = \sum_i \left(\mathbf{z}_i^T \mathbf{u}_a\right) \mathbf{z}_i$$

$$\Rightarrow u_a = \sum_i \frac{\mathbf{z}_i^T \mathbf{u}_a}{\lambda_a} \mathbf{z}_i =: \sum_i \alpha_i^a \mathbf{z}_i$$

Efficient computation for *n*<*p*

- We can replace the $p \times p$ with an $n \times n$ eigenvalue problem
  - Based on $\mathrm{S}\mathbf{u} = \lambda\mathbf{u} \Leftrightarrow \mathbf{z}_i^{\mathrm{T}}\mathrm{S}\mathbf{u} = \lambda\mathbf{z}_i^{\mathrm{T}}\mathbf{u}$ for all *i*

$$\mathbf{z}_i^{\mathrm{T}}\mathrm{S}\mathbf{u}_{\mathrm{a}} = \lambda_a\mathbf{z}_i^{\mathrm{T}}\mathbf{u}_{\mathrm{a}}$$

$$\mathbf{z}_i^{\mathrm{T}}\sum_l \mathbf{z}_l\mathbf{z}_l^T\sum_j \alpha_j^a\mathbf{z}_j = \lambda_a\mathbf{z}_i^{\mathrm{T}}\sum_j \alpha_j^a\mathbf{z}_j$$

$$\sum_{j,l}\alpha_j^a\left[\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_l\right]\left[\mathbf{z}_l^T\mathbf{z}_j\right] = \lambda_a\sum_j \alpha_j^a\left[\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j\right]$$

  - With $n \times n$ centered inner product („Gram“) matrix $K_{ij}^c := \left[\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j\right]$:

$$(\mathrm{K}^{\mathrm{c}})^2\boldsymbol{\alpha}^a = \lambda_a\mathrm{K}^c\boldsymbol{\alpha}^a \Rightarrow \mathrm{K}^c\boldsymbol{\alpha}^a = \lambda_a\boldsymbol{\alpha}^a$$

Proper Normalization and Projecting New Data

- **Normalization:** $\mathbf{u}_a^{\mathrm{T}}\mathbf{u}_a = 1$ translates into

$$\sum_{i,j} \alpha_i^a \alpha_j^a \left[\mathbf{z}_i^T \mathbf{z}_j\right] = (\boldsymbol{\alpha}^a)^T \mathrm{K}^c \boldsymbol{\alpha}^a$$

$$= \lambda_a (\boldsymbol{\alpha}^a)^T \boldsymbol{\alpha}^a = 1$$

  – Therefore:

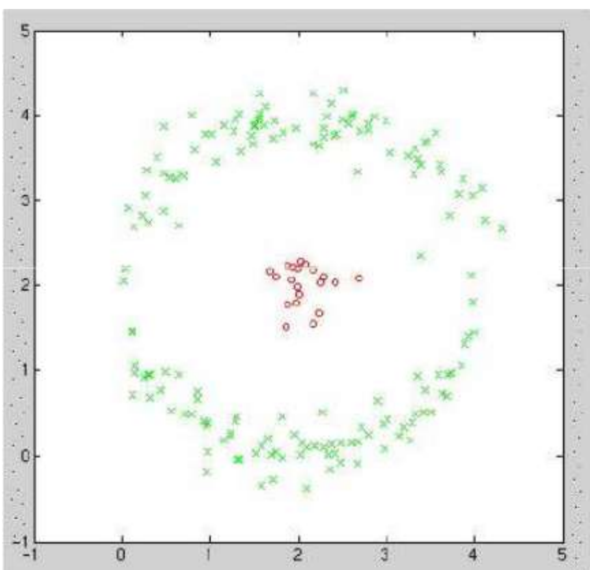
$$\|\boldsymbol{\alpha}^a\| = \frac{1}{\sqrt{\lambda_a}}$$

- **Projection** of a new point **x**:

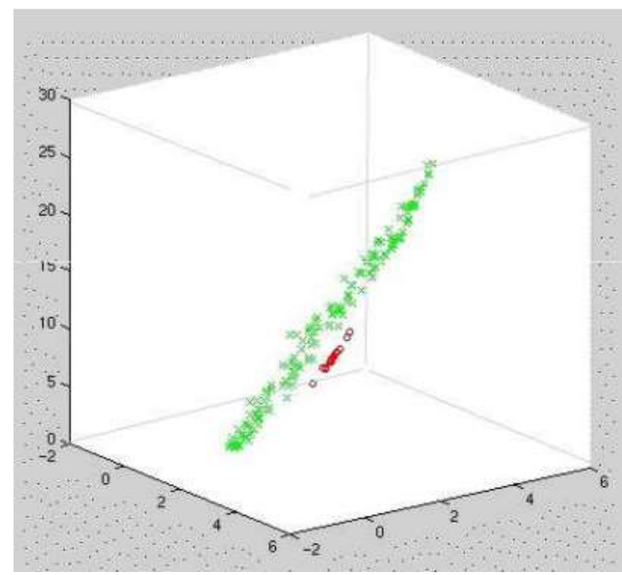
$$\mathbf{u}_a^T(\mathbf{x} - \bar{\mathbf{x}}) = \sum_i \alpha_i^a \, \mathbf{z}_i^T (\mathbf{x} - \bar{\mathbf{x}})$$

Preserving Nonlinear Structures

- PCA works well for **linear structures**
  - Straight lines, planes, etc.
- Can we preserve **nonlinear / curved structure?**
- **Idea:** Nonlinearly map data to a higher-dimensional feature space, apply PCA there



$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$



Kernel Trick

- We reformulated PCA so that it only makes use of the data $\mathbf{x}_i$ within scalar products
- To apply PCA in feature space, we do not explicitly need the feature map $\mathbf{x} \to \phi(\mathbf{x})$, only a nonlinear kernel function $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$
- K should produce positive definite matrices
- Widely used examples (here without proofs):
  - Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^T\mathbf{x}_j$
  - Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$
  - Gaussian (RBF): $K\left(\mathbf{x}_i, \mathbf{x}_j\right) = e^{-\gamma\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}$

Centering in Feature Space

- We require the centered kernel matrix $K_{ij}^{c} = (\Phi_i - \overline{\Phi})^T \left(\Phi_j - \overline{\Phi}\right)$, but evaluating the kernel function gives us $K_{ij} = K\left(\mathbf{x}_i, \mathbf{x}_j\right) = {\Phi_i}^{\mathrm{T}} \Phi_j$
  - Applying *K* to centered data $\mathbf{z}_i$ is *not* equivalent to centering feature vectors Φ_i
- Centering can be performed as follows:

$$
\begin{aligned}
K_{ij}^{c} &= (\Phi_i - \overline{\Phi})^{\mathrm{T}} \left(\Phi_j - \overline{\Phi}\right) \\
&= \Phi_i^{\mathrm{T}} \Phi_j - \overline{\Phi}^{\mathrm{T}} \Phi_j - \Phi_i^{\mathrm{T}} \overline{\Phi} + \overline{\Phi}^{\mathrm{T}} \overline{\Phi}
\end{aligned}
$$

  - $\overline{\Phi}^{\mathrm{T}} \Phi_j = \frac{1}{n} \sum_i {\Phi_i}^{\mathrm{T}} \Phi_j = \frac{1}{n} \sum_i K_{ij}$
  - $\Phi_i^{\mathrm{T}} \overline{\Phi} = \frac{1}{n} \sum_j {\Phi_i}^{\mathrm{T}} \Phi_j = \frac{1}{n} \sum_j K_{ij}$
  - $\overline{\Phi}^{\mathrm{T}} \overline{\Phi} = \frac{1}{n^2} \sum_{i,j} {\Phi_i}^{\mathrm{T}} \Phi_j = \frac{1}{n^2} \sum_{i,j} K_{ij}$

Notation: Centering in Feature Space

- Let $\mathbf{1} \in \mathbb{R}^n$ denote the column vector in which each coefficient equals 1.
  - Then, $\mathrm{H} = \mathrm{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathrm{T}$ is a matrix with $\frac{n-1}{n}$ on the diagonal and $-\frac{1}{n}$ everywhere else
- Easy to verify that

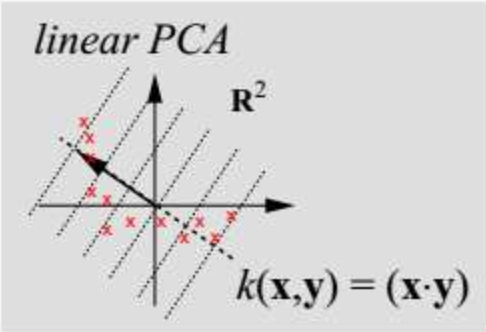
$$\mathrm{K}^c = \mathrm{HKH}$$

Algorithm: Kernel PCA

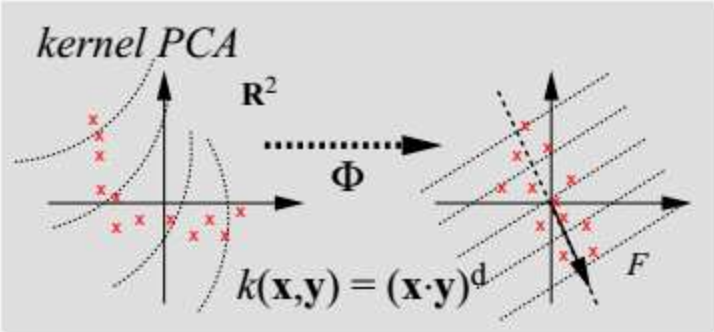
- **Input:** $\mathbf{x}_i \in \mathbb{R}^p, i = 1,2, \ldots, n$ and kernel function $K: \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$
- **Output:** $\mathrm{U}_k \in \mathbb{R}^{p \times k}$ with $k \leq p$ and $\mathrm{D}_k \in \mathbb{R}^{k \times k}$ such that $\mathbf{y} = \mathrm{D}_k^{-1}\mathrm{U}_k^{\mathrm{T}}\mathrm{H}\left(\mathbf{k} - \frac{1}{n}\mathrm{K}\mathbf{1}\right)$ with $k_i = K(\mathbf{x}_i, \mathbf{x})$
- **Algorithm:**
  - Compute **kernel matrix** $K_{ij} = K\left(\mathbf{x}_i, \mathbf{x}_j\right)$
  - **Center kernel matrix** $K^c = \mathrm{HKH}$
  - Compute **spectral decomposition** $\mathrm{K}^{\mathrm{c}} = \mathrm{U}\Lambda\mathrm{U}^{\mathrm{T}}$
    - Sort coefficients of Λ in decreasing order
  - Form U_k from *k* leading columns of U
  - Form D_k from top-left *k* × k block of $\sqrt{\Lambda}$

Kernel PCA: Advantages and Disadvantages

- **Advantages** of Kernel PCA:
  - Better reflects nonlinear structures
  - Given suitable kernels, can be applied to more abstract objects or to unfold manifolds (see next sections)



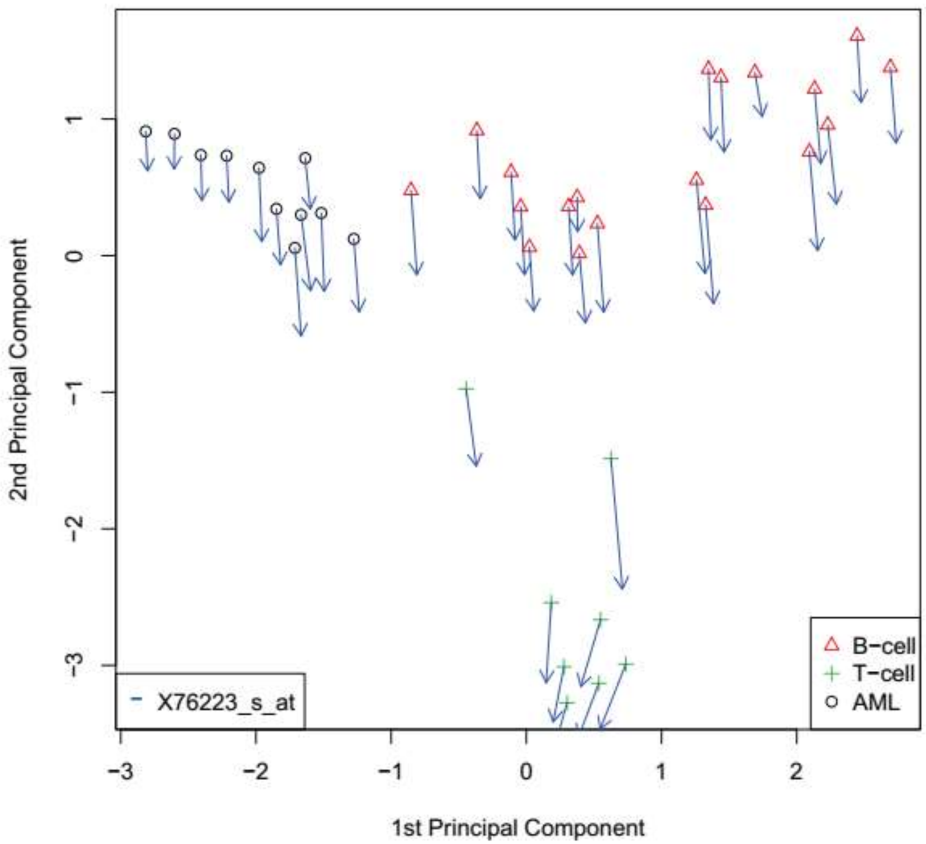
- **Disadvantages** of Kernel PCA:
  - Less interpretable: Principal modes are no longer a fixed combination of input variables
  - Unlike with linear PCA, it is not easy to find a vector that corresponds to given Kernel PCA coordinates („pre-image problem“)

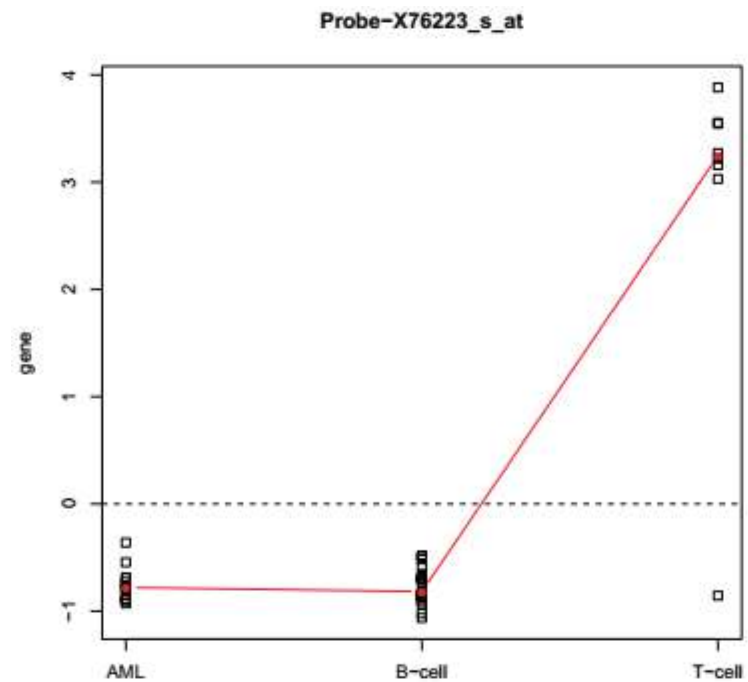


- **Advantage / Disadvantage:**
  - Complexity grows with number of samples
  - If needed, cluster as pre-process (e.g., k-means)

Example: Kernel PCA of Gene Expression Data

- Leukemia dataset with *n*=19, *p*=3051
- Kernel PCA with RBF kernel





Example from [Reverter et al., 2012]

Section 4.2: Multidimensional Scaling

Multidimensional scaling (MDS)

Multidimensional scaling (MDS)

- A dimensionality reduction technique
- Maps pairwise distances to coordinates
- Provides model of non-geometric data
- Useful for
  - visualizing high-dimensional data
  - preprocessing data before clustering

Example: Multidimensional scaling (MDS)

Example: map of the US

Given a list of cities…

Chicago

Raleigh

Boston

Seattle

San Francisco

Austin

Orlando

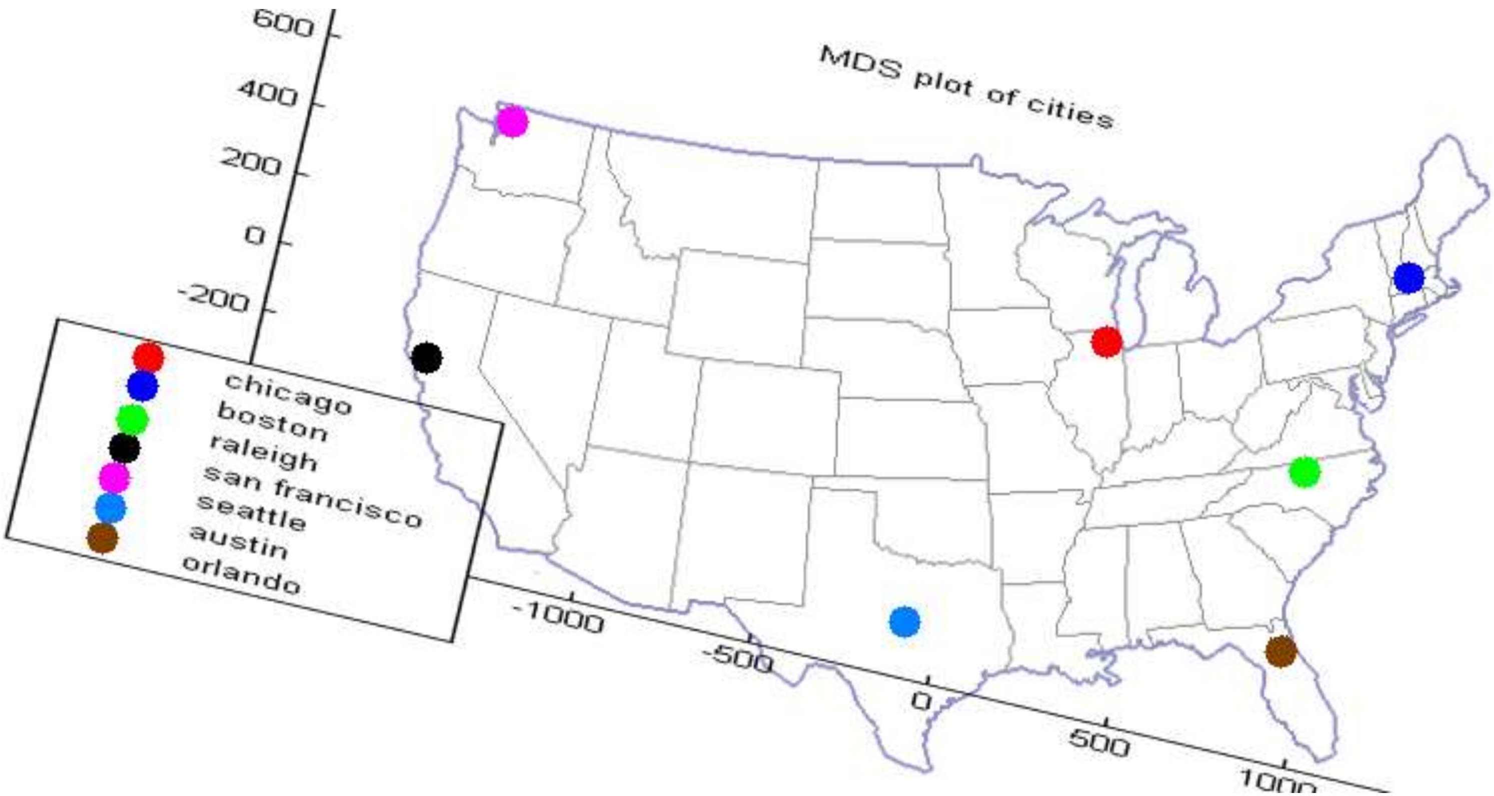
Example: Multidimensional scaling (MDS)

Knowing only the distances between them…

	Chicago	Raleigh	Boston	Seattle	San Francisco	Austin	Orlando
Chicago	0						
Raleigh	641	0					
Boston	851	608	0				
Seattle	1733	2363	2488	0			
San Francisco	1855	2406	2696	684	0		
Austin	972	1167	1691	1764	1495	0	
Orlando	994	520	1105	2565	2458	1015	0

Example: Multidimensional scaling (MDS)

Result: Given only the distances between points, MDS finds suitable coordinates for each point, of the specified dimension



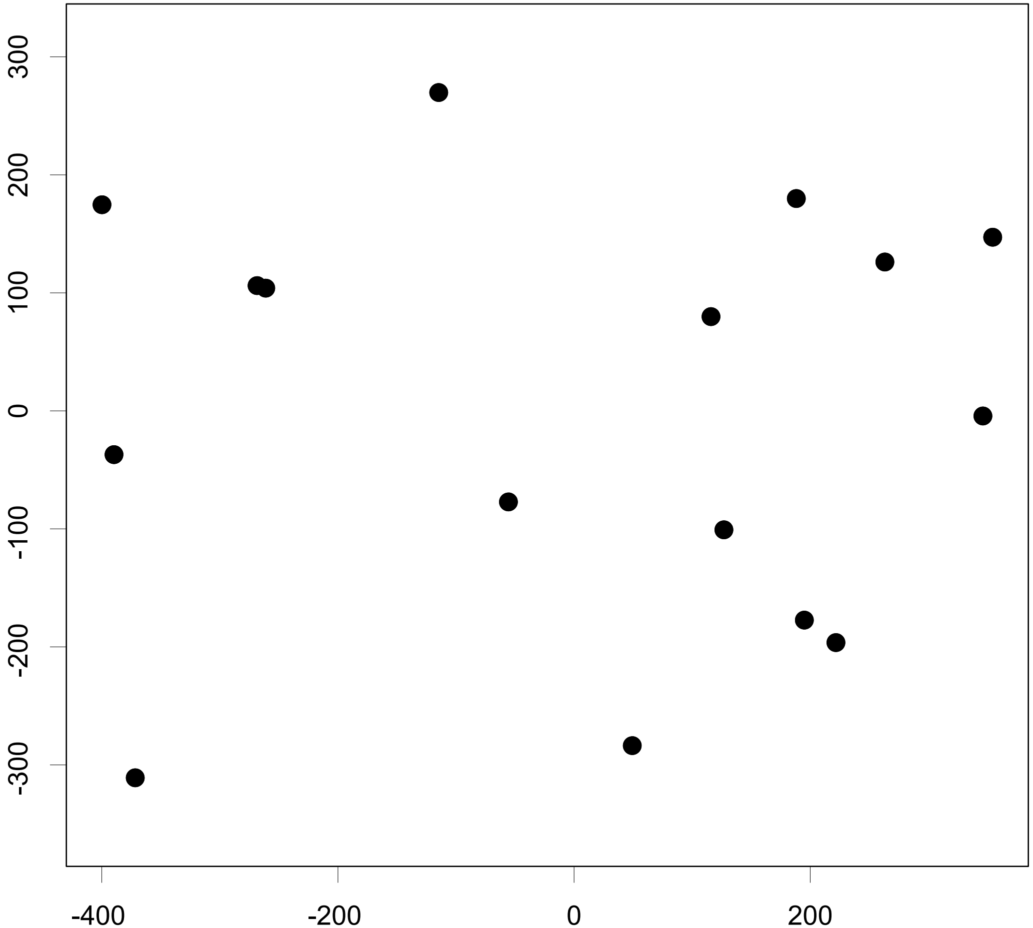
Example with non-Euclidean Dissimilarity Matrix

- Given: *street* distances between 16 German cities (according to a route planner)

	B	HB	DD	D	EF	HH	H	KI	MD	MZ	M	P	SB	SN	S	WI
Berlin (B)	0	392	193	559	303	289	286	353	156	580	585	36	722	211	633	570
Bremen (HB)	392	0	472	287	334	124	131	208	253	476	749	363	575	224	629	466
Dresden (DD)	193	472	0	615	218	500	367	542	232	496	465	211	638	400	510	486
Düsseldorf (D)	559	287	615	0	400	411	277	495	419	216	611	529	298	511	404	201
Erfurt (EF)	303	334	218	400	0	362	219	458	235	289	414	274	431	475	339	279
Hamburg (HH)	289	124	500	411	362	0	157	97	280	528	775	283	670	109	655	518
Hannover (H)	286	131	367	277	219	157	0	248	147	384	632	256	526	252	512	374
Kiel (KI)	353	208	542	495	458	97	248	0	376	624	872	348	766	175	752	614
Magdeburg (MD)	156	253	232	419	235	280	147	376	0	465	519	125	607	309	567	455
Mainz (MZ)	580	476	496	216	289	528	384	624	465	0	430	544	146	621	210	14
München (M)	585	749	465	611	414	775	632	872	519	430	0	555	487	757	232	424
Potsdam (P)	36	363	211	529	274	283	256	348	125	544	555	0	693	207	604	541
Saarbrücken (SB)	722	575	638	298	431	670	526	766	607	146	487	693	0	771	262	159
Schwerin (SN)	211	224	400	511	475	109	252	175	309	621	757	207	771	0	756	619
Stuttgart (S)	633	629	510	404	339	655	512	752	567	210	232	604	262	756	0	220
Wiesbaden (WI)	570	466	486	201	279	518	374	614	455	14	424	541	159	619	220	0

MDS: Non-Euclidean Example

- MDS result



MDS: Non-Euclidean Example

- MDS result rotated and overlayed on a geographic map

Multidimensional scaling: Basic Concepts

- Input
  - Symmetric dissimilarity matrix, containing pair-wise dissimilarities $\delta_{ij} = \delta(x_i, x_j)$ between (p-dimensional) data samples $x_1, \dots, x_n$
  - Desired dimensionality k ($k < p$, often $k = 2{,}3$)
- Output
  - Image of data in a Euclidean frame $y_1, \dots, y_n$ such that pair-wise distances $\mathrm{d_{ij}} = d(y_i, y_j)$ are best approximation to original dissimilarities $\delta(x_i, x_j)$ for k dimensions
- Embedding errors
  - Take into account $n(n-1)/2$ errors between the individual distances
  - Have to be invariant against rigid transformations (i.e. translation, rotation, mirroring)

Find $y_i \in \mathbb{R}^k$ such that $d\left(y_i, y_j\right) \approx \delta_{ij}$

MDS: What is a good approximation?

- Possible error functionals ("Stress"):

$$J_{ee} = \frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$$

(punishes large absolute deviations)

$$J_{ff} = \sum_{i<j}\left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$$

(punishes large relative deviations)

$$J_{ef} = \frac{1}{\sum_{i<j}\delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}$$

Compromise between J_{ee} and J_{ff}

(Commonly L2 distance measure $d_{ij} = \left\|y_i - y_j\right\|_2$ is used here.)

MDS: What is a good approximation?

- Example global vs. local

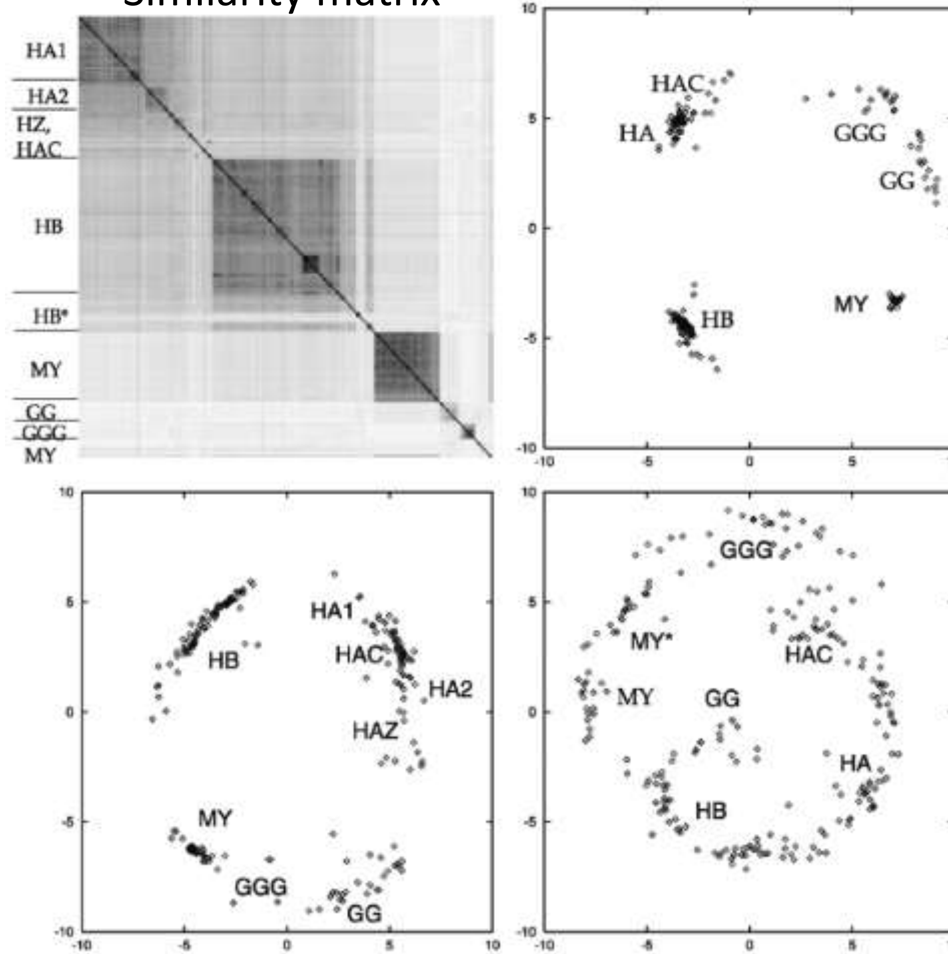
$$J_{ee} = \frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$$

$$J_{ff} = \sum_{i<j}\left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$$

$$J_{ef} = \frac{1}{\sum_{i<j}\delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}$$

Similarity matrix

J_{ee}



J_{ff} J_{ef}

Image from [Klock et al. 2000]

MDS: Solution by Iterative Methods

- Common solution: Iterative optimization (e.g. gradient descent)
- Requires derivatives of objective function with respect to embedded point positions:

$$
\begin{aligned}
\frac{\partial J_{ee}}{\partial y_k} &= \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{kj}) \frac{y_k - y_j}{d_{kj}} \\
\frac{\partial J_{ff}}{\partial y_k} &= 2 \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}^2} \cdot \frac{y_k - y_j}{d_{kj}} \\
\frac{\partial J_{ef}}{\partial y_k} &= \frac{2}{\sum_{i<j} \delta_{ij}} \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}} \cdot \frac{y_k - y_j}{d_{kj}}
\end{aligned}
$$

Metric MDS: Solution via Kernel PCA

- Assume dissimilarity matrix $\Delta = (\delta_{ij})$ results from points in an unknown feature space:
$$\{\Phi_i \in \mathbb{R}^q, i = 1, \dots, n\}$$
- In this case:
$$\delta_{ij}^2 = \left\|\Phi_i - \Phi_j\right\|^2$$
$$= \|\Phi_i\|^2 + \left\|\Phi_j\right\|^2 - 2\langle\Phi_i, \Phi_j\rangle$$
- Idea:
If we can transform Δ into an inner product matrix, we can continue as in kernel PCA

Metric MDS: Reduction to Kernel PCA

- Inner product matrix of centered features Φ_i

$$\mathrm{K}^{\mathrm{c}} = \left[\left\langle \Phi_i, \Phi_j \right\rangle\right]$$

- Then, $\delta_{ij}^2 = K_{ii} + K_{jj} - 2K_{ij}$

  Since features are centered $\sum_i \Phi_i = 0$,
  we have that $\sum_i K_{ij} = 0$ and thus:

$$\frac{1}{N}\sum_i \delta_{ij}^2 = \frac{1}{N}\sum_i K_{ii} + K_{jj} \quad \coloneqq \delta_{\oplus j}^2$$

$$\frac{1}{N}\sum_j \delta_{ij}^2 = K_{ii} + \frac{1}{N}\sum_j K_{jj} \quad \coloneqq \delta_{i\oplus}^2$$

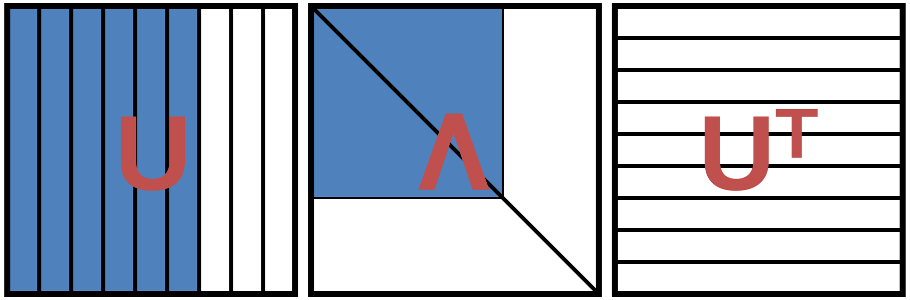
$$\frac{1}{N^2}\sum_i \sum_j \delta_{ij}^2 = \frac{2}{N}\sum_i K_{ii} \quad \coloneqq \delta_{\oplus\oplus}^2$$

- Transform dissimilarities to

$$K_{ij} = -\frac{1}{2}\left(\delta_{ij}^2 - \delta_{i\oplus}^2 - \delta_{\oplus j}^2 + \delta_{\oplus\oplus}^2\right)$$

Metric MDS: Solution via Eigenvectors

1. Compute $\mathrm{K^c} = -\frac{1}{2}\mathrm{HDH}$ with $\mathrm{H} = \mathrm{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$
   – If Δ is metric, this defines $\mathrm{K^c}$ such that $\mathrm{K^c} = \left[\langle \Phi_i, \Phi_j \rangle\right]$
2. Compute spectral decomposition of $\mathrm{K^c}$
   – $\mathrm{K^c} = \mathrm{U}\Lambda\mathrm{U}^T$



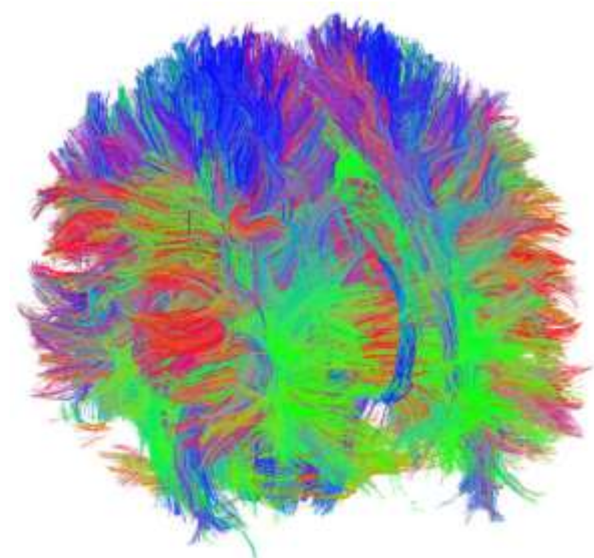
3. Lower dimensional embedding:
   - Want k-dimensional coordinates
   - U_k = first k columns of U
   - Λ_k = k x k submatrix of Λ
   - Desired coordinates = $U_k \Lambda_k^{1/2}$

Properties of MDS

- Effort independent of dimensionality
  - Distance matrix is all that matters
- Solution unique only up to rigid transformation and reflection
- Approaches:
  - Optimization-based: Works with all matrices and different error measures
  - Analogous to kernel PCA: Assumes underlying metric space, permits projection of new points (if you know distances to all existing ones)

Example: Full-Brain Tractography

Problem: Nerve fiber pathways present themselves as an impenetrable knot of curves. How to interact with them (e.g., make a selection) in an easy way?



Idea: It is much simpler to interact with 2D views! Can't we represent each streamline as a point in 2D image space, such that similar streamlines are placed nearby?

Image by Tobias Isenberg

Linked 2D Views

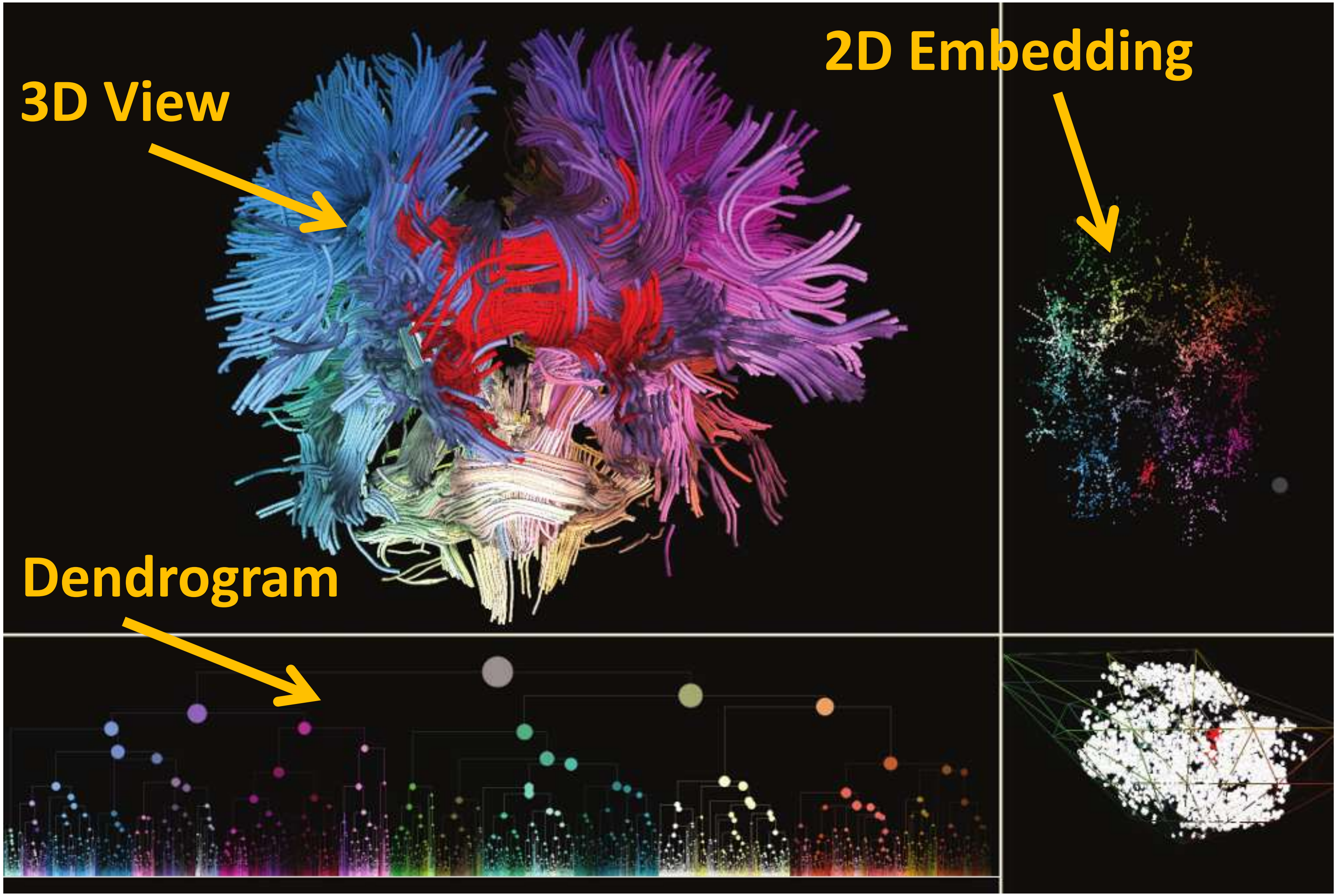


Image taken from [Jianu et al. 2009]

Section 4.3:
ISOMAP

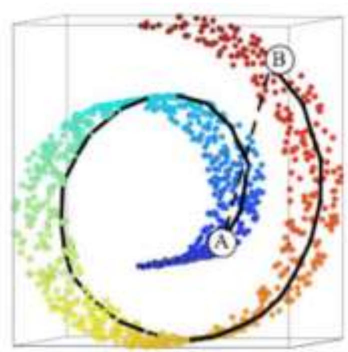
Distances on Nonlinear Manifolds

- MDS allows us to use arbitrary distances
- On **nonlinear manifolds,** we should use **geodesic distances** (length of shortest connection on manifold), not Euclidean:

$\boxed{d(A,C) < d(A,B)}$ $\boxed{d(A,C) > d(A,B)}$

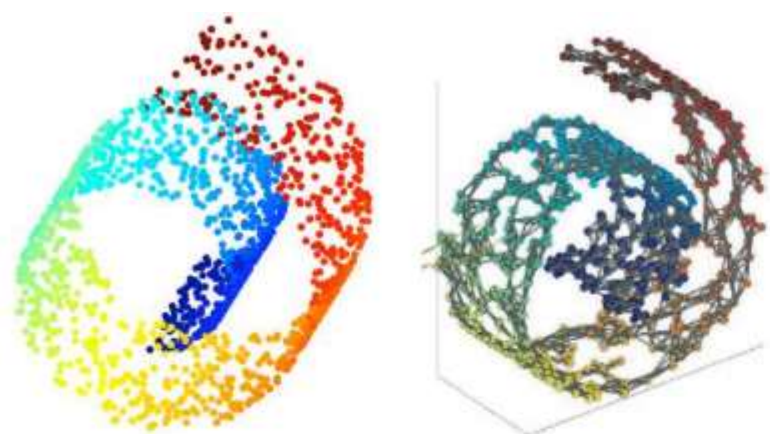
Idea: Estimating Geodesic Distance

- We are usually not given the manifold, only a set of discrete input points
- **Estimate geodesic distances:**
  - Use Euclidean distance for nearby points
  - Connect distant points by short hops between nearby points, measure length of shortest path

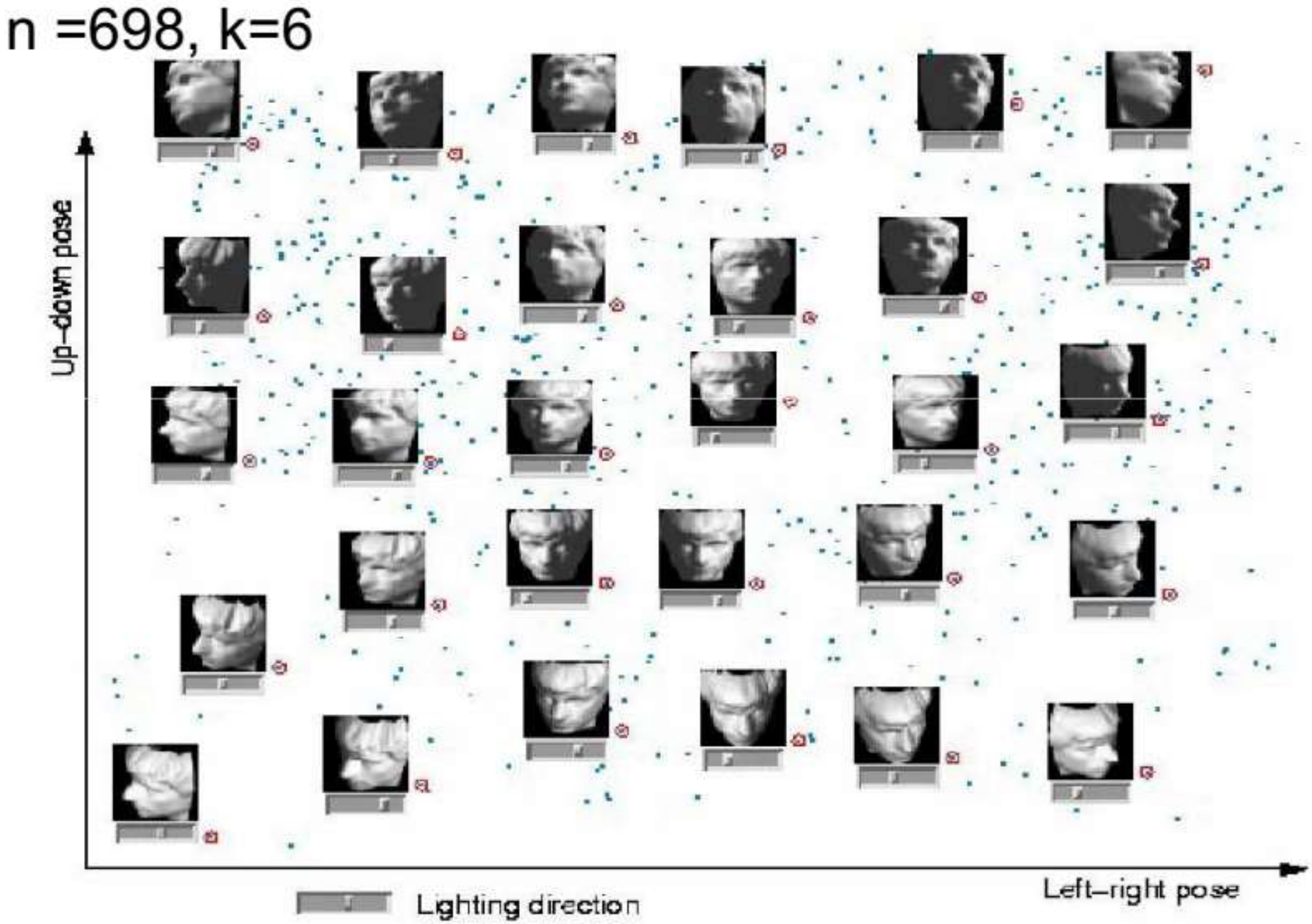


ISOMAP Algorithm

1. Build a neighborhood graph (e.g., k-NN) that approximates the manifold structure
   – Make sure it‘s connected
   – Very similar to spectral clustering
2. For the resulting graph, compute all-pairs shortest paths
3. Perform metric MDS based on resulting distance matrix



ISOMAP: Example result



Summary: Dimensionality Reduction

- Dimensionality reduction projects high-dimensional objects to low-dimensional space
  - **PCA:** Takes Euclidean input and finds linear projection that preserves maximum variance
  - **Kernel PCA:** Generalizes PCA to better preserve nonlinear structures
  - **MDS:** Can be applied even if original data does not have coordinates or if distances are non-Euclidean
  - **ISOMAP:** Combines MDS with distance measures on neighborhood graphs to approximate nonlinear manifolds