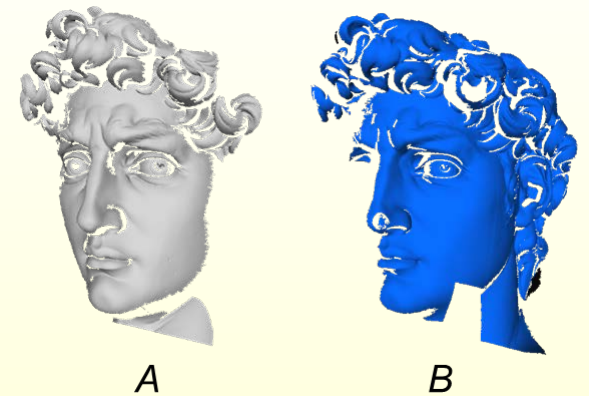


# CS233: Geometric and Topological Data Analysis

---

Alignments and Correspondences, Geometric Features

9 May 2018



Slides ack: Peter Huang, Vova Kim, Niloy Mitra, Maks Ovsjanikov

# Last Time: Differential Geometry, Laplace- Beltrami Operator

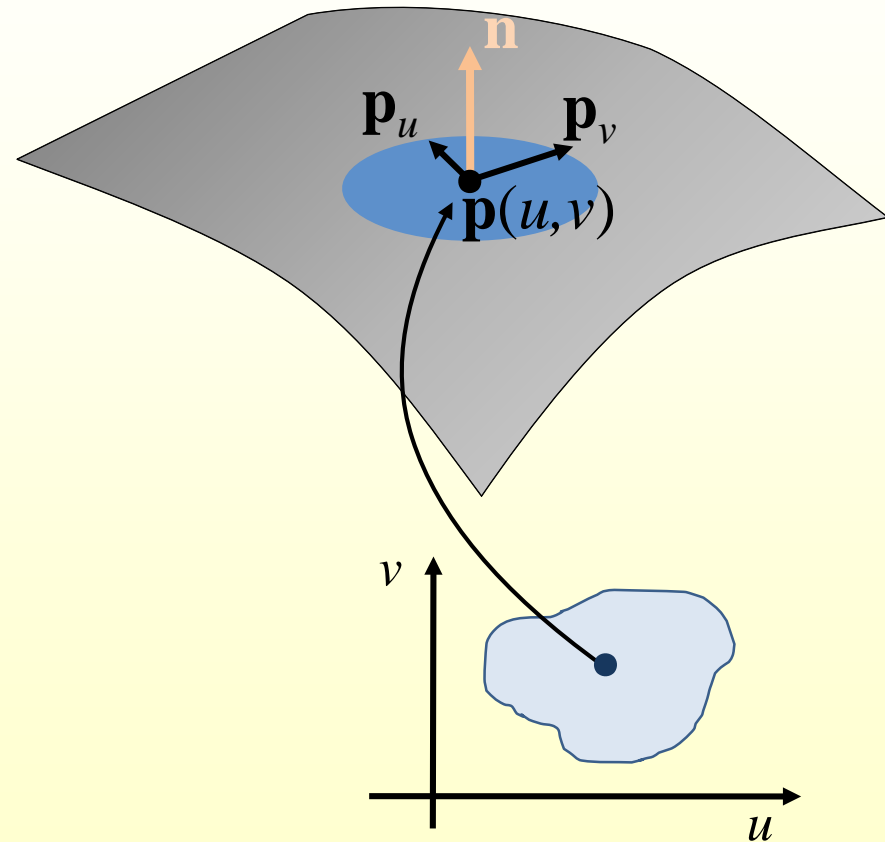
# Surface Normals

- Surface normal:

$$\mathbf{n}(u, v) = \frac{\mathbf{p}_u \times \mathbf{p}_v}{\|\mathbf{p}_u \times \mathbf{p}_v\|}$$

- Assuming *regular* parameterization, i.e.,

$$\mathbf{p}_u \times \mathbf{p}_v \neq 0$$



# Surface Curvatures

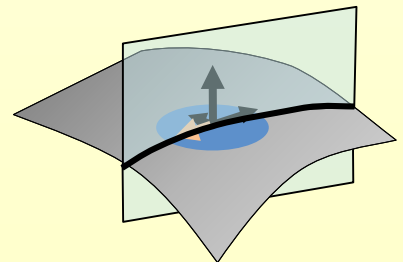
## Principal curvatures

Minimal curvature  $\kappa_1 = \kappa_{\min} = \min_{\varphi} \kappa_n(\varphi)$

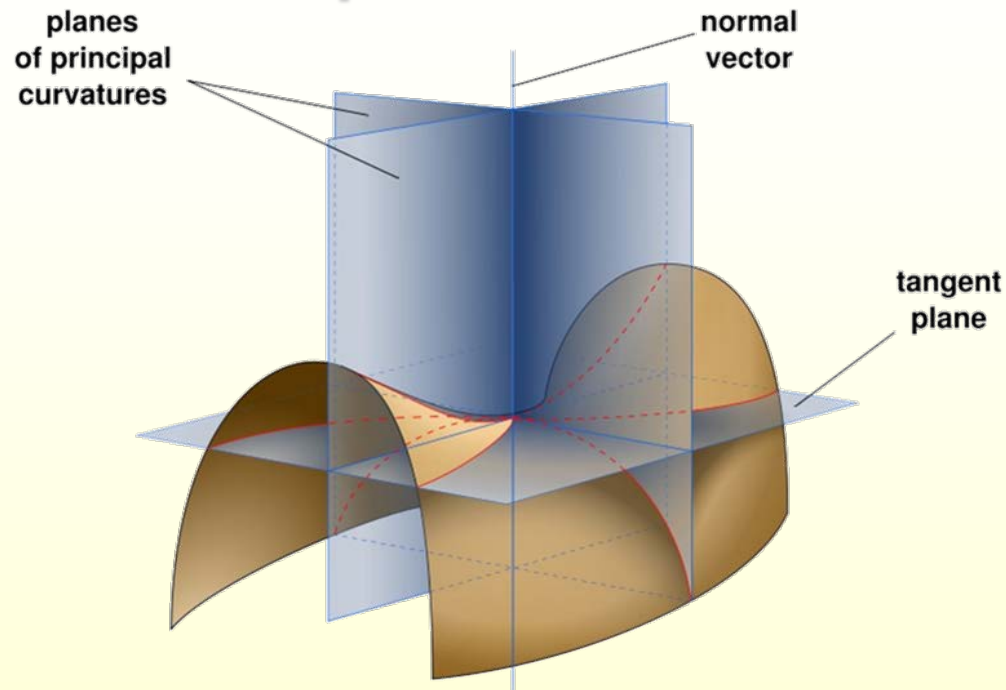
Maximal curvature  $\kappa_2 = \kappa_{\max} = \max_{\varphi} \kappa_n(\varphi)$

Mean curvature  $H = \frac{\kappa_1 + \kappa_2}{2} = \frac{1}{2\pi} \int_0^{2\pi} \kappa_n(\varphi) d\varphi$

Gaussian curvature  $K = \kappa_1 \cdot \kappa_2$



# Principal Directions



**Euler's Theorem:** Planes of principal curvature are **orthogonal** and independent of parameterization.

$$\kappa_n(\varphi) = \kappa_1 \cos^2 \varphi + \kappa_2 \sin^2 \varphi, \quad \varphi = \text{angle with } \mathbf{t}_1$$

# Continuous Laplace-Beltrami Operator

- Extension of Laplace operator to functions on manifolds

$$f : \mathcal{M} \rightarrow \mathbb{R}$$

$$\Delta f : \mathcal{M} \rightarrow \mathbb{R}$$

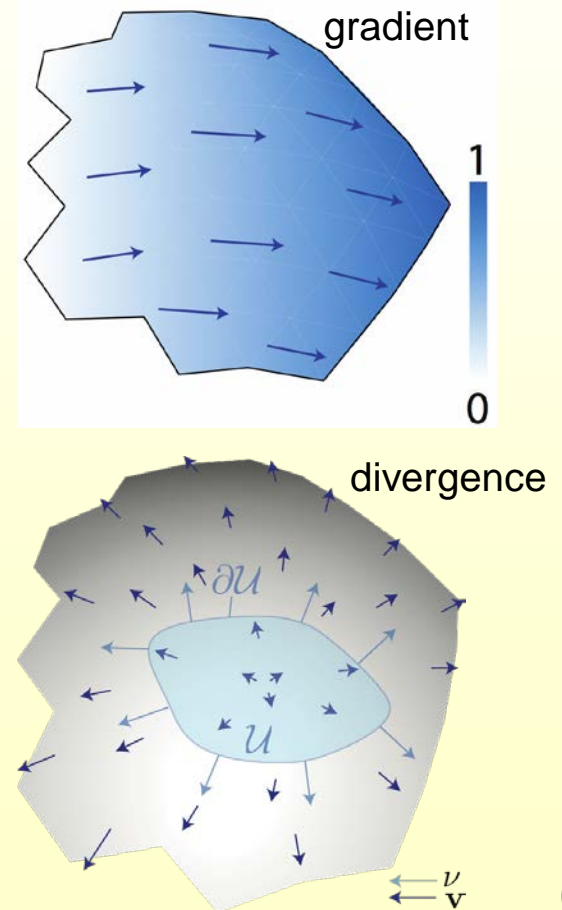
Laplace-Beltrami

gradient operator

$$\Delta_{\mathcal{M}} f = \operatorname{div}_{\mathcal{M}} \nabla_{\mathcal{M}} f$$

function on surface  $M$

divergence operator



# Laplace-Beltrami and Curvature

- ◆ Apply operator to coordinate functions

Laplace-Beltrami

gradient operator

mean curvature

coordinate functions on surface  $M$   
 $\mathbf{p} = (x, y, z)$

divergence operator

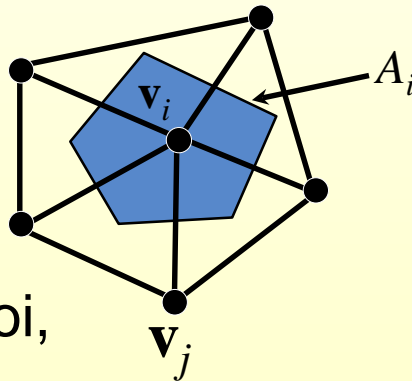
unit surface normal

$$\Delta_{\mathcal{M}} \mathbf{p} = \operatorname{div}_{\mathcal{M}} \nabla_{\mathcal{M}} \mathbf{p} = -2H \mathbf{n} \in \mathbb{R}^3$$

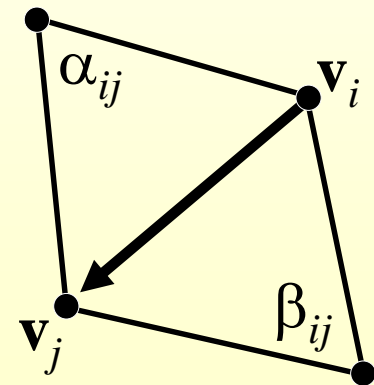
# Discrete LB - Cotangent Formula

- ◆ Better: cotangent formula

$$\Delta_S f(v_i) := \frac{1}{2A_i} \sum_{v_j \in \mathcal{N}_1(v_i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (f(v_j) - f(v_i))$$



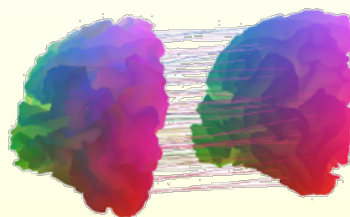
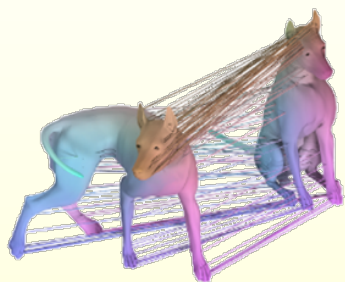
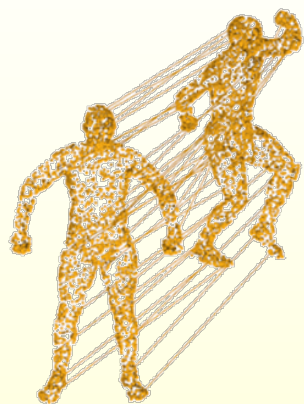
$A_i$ : vertex area (Voronoi, barycentric..)



***Can be derived by discretizing continuous L-B via linear Finite Elements!***

# Shape Features, Alignment, and Correspondences

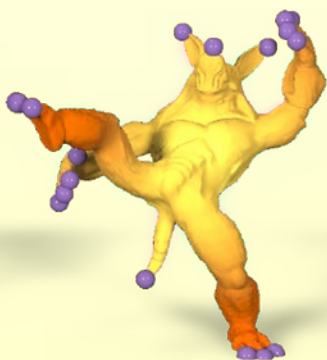
# Alignment and Correspondences, Shape Features



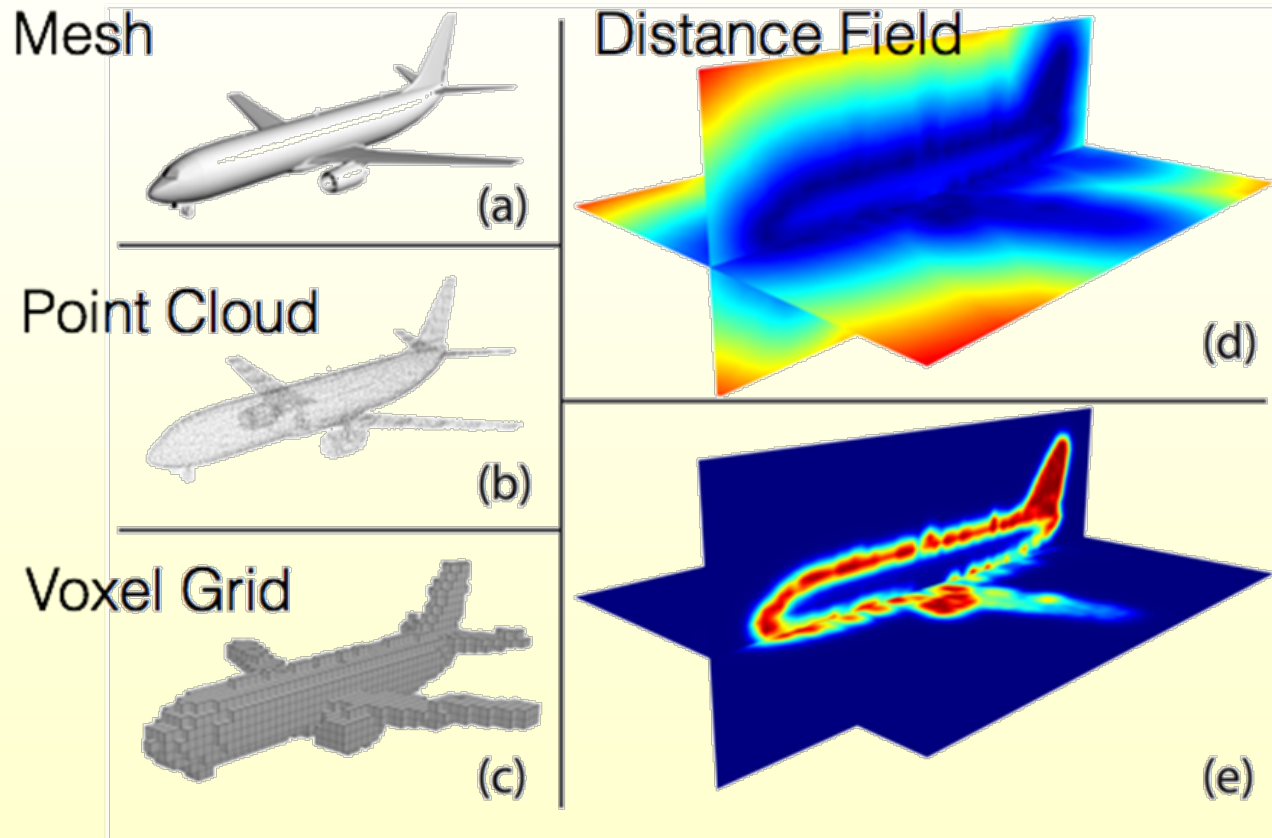
A



B

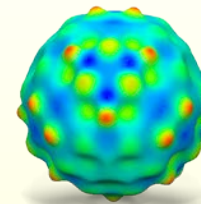


# 3D Representations



# Functions on 3D Geometry

- ◆ Functions on data
  - ◆ Properties, attributes, descriptors, part indicators, etc.
  - ◆ But also beliefs, opinions, etc
- ◆ Operators on functions
  - ◆ Maps of functions to functions
    - ◆ Laplace-Beltrami operator on a manifold  $M$

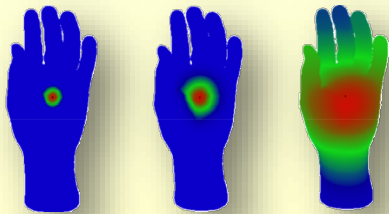


Curvature



Parts

$$\Delta : C^\infty(M) \rightarrow C^\infty(M), \Delta f = \operatorname{div} \nabla f$$



$$\frac{\partial u}{\partial t} = -\Delta u$$

heat diffusion

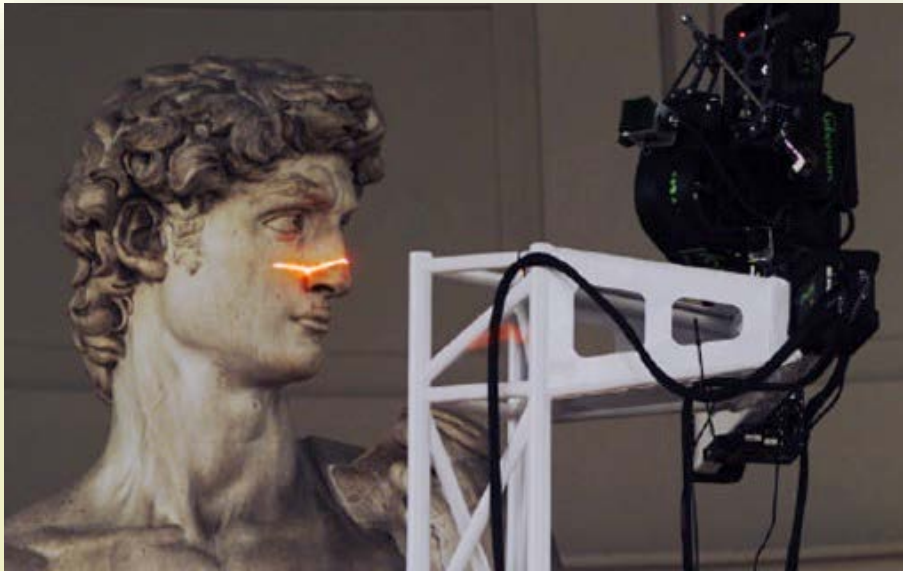


Laplace Beltrami eigenfunctions

# Registration and Alignment

# Shape Acquisition – See CS348a

- ◆ Many 3D scanning techniques



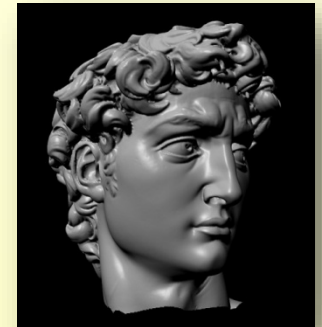
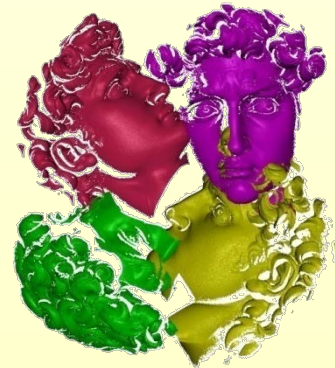
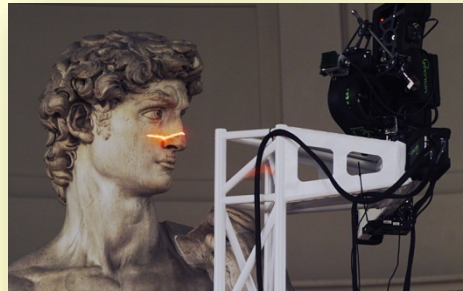
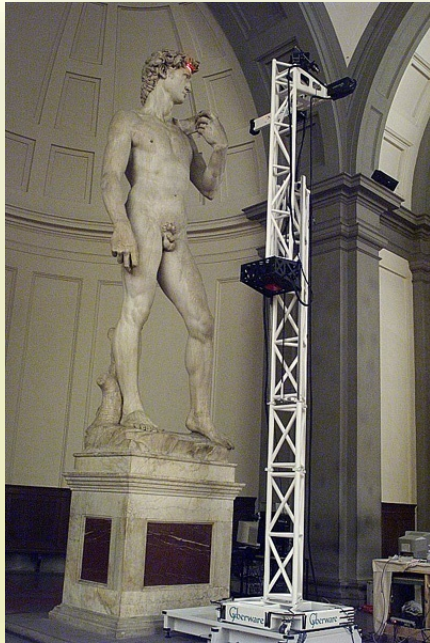
Triangulation scanner



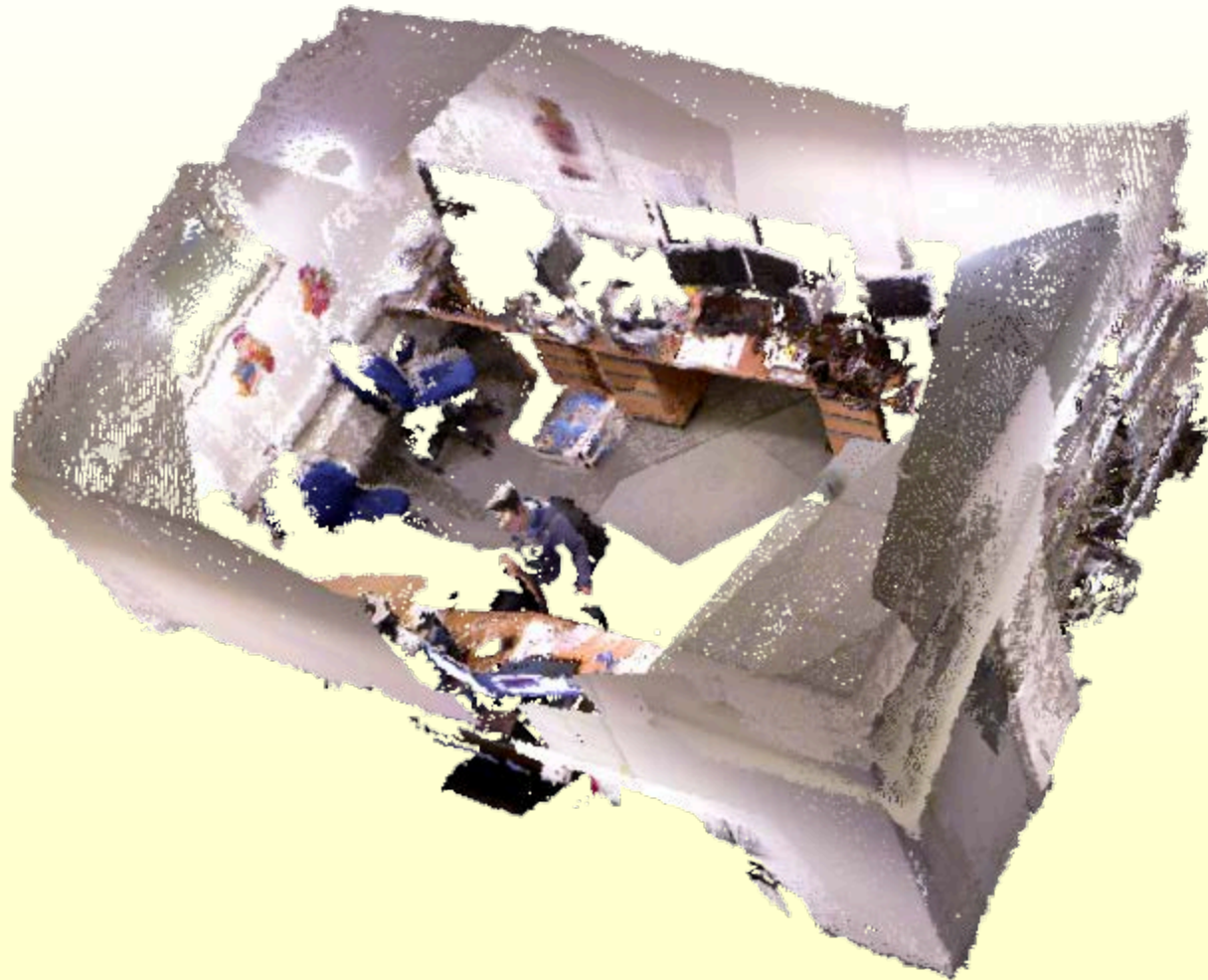
Time of flight scanner

# Aligning and Registering (Partial or Entire) Shapes

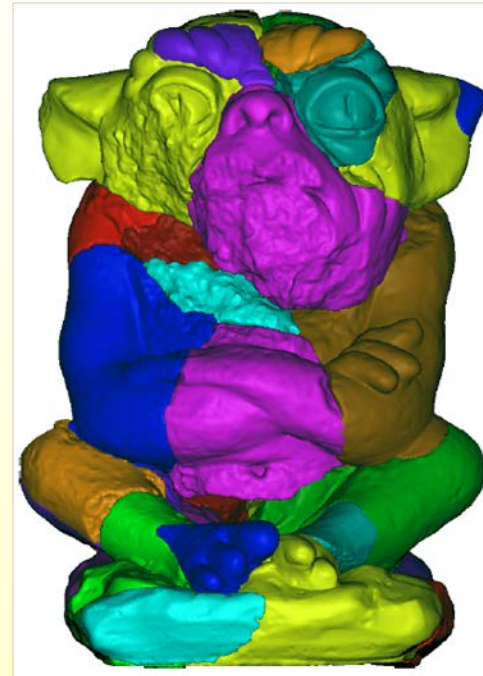
- ◆ The need to align and match shapes arises in many domains
- ◆ A classic example is shape acquisition through a 3D scanner



# Simultaneous Localization and Mapping (SLAM)



# Cultural Artifact Reconstruction



# Protein Structure Alignment



Human (red) and fly (yellow) thioredoxins, compared

# The (3D) Registration Problem

## Given:

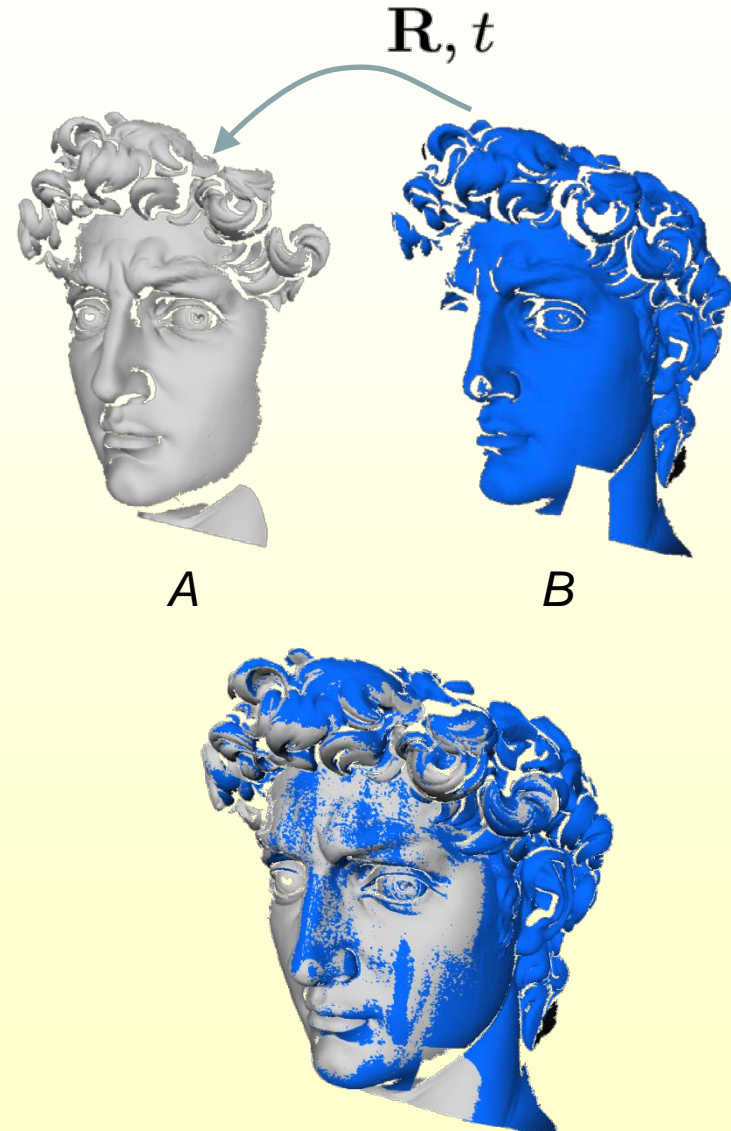
Two shapes  $A$  and  $B$  which partially overlap

## Goal:

Using only **rigid transforms**, register  $B$  against  $A$  by minimizing a measure of **distance** between  $A$  and  $B$

## Assume:

$A$  and  $B$  are positioned close to each other



# Measuring Success: Shape Distances

Given two shapes  $A$  and  $B$ , we are interested in defining a distance or (dis-)similarity measure

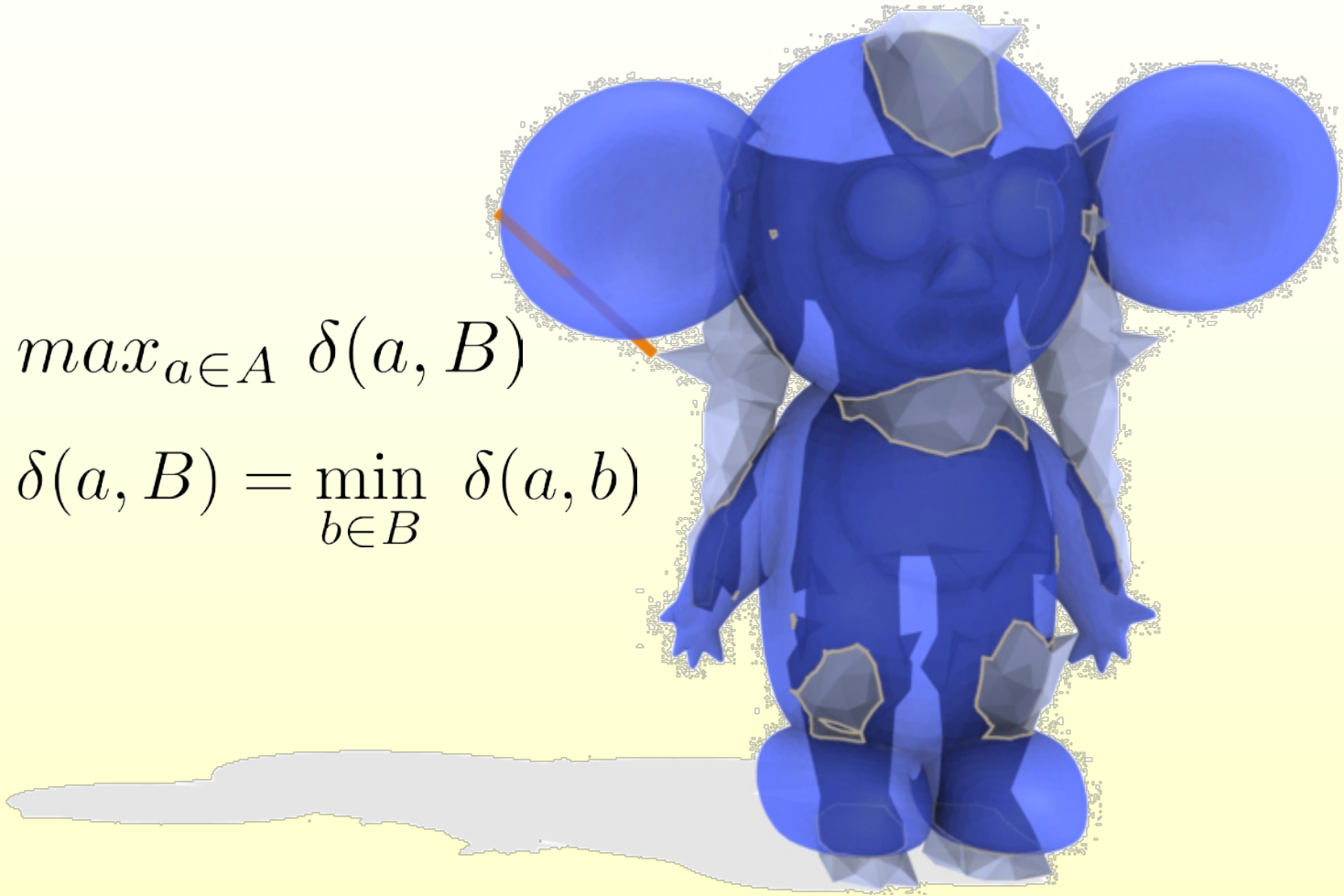
$$\min_T \delta(A, T(B)) \quad \text{[extrinsic]}$$

Such measures are crucial in shape similarity search, shape classification, and in general for defining ML loss functions.

# Hausdorff Distance

$$\max_{a \in A} \delta(a, B)$$

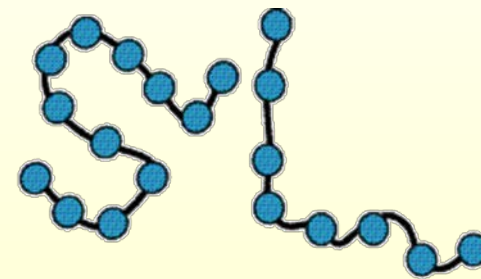
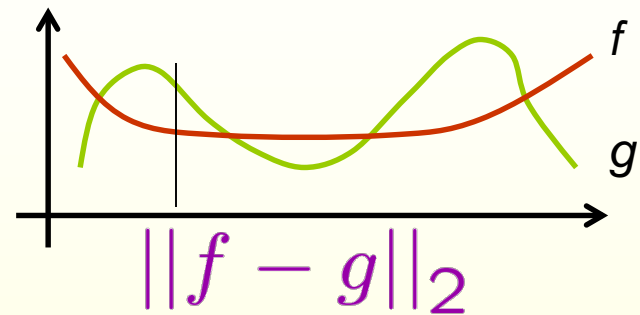
$$\delta(a, B) = \min_{b \in B} \delta(a, b)$$



Hausdorff is a worst-case error – in some cases we prefer MSE

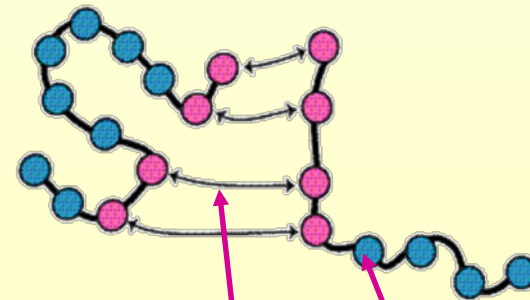
# Issues about Distance Metrics

- ◆ We are all familiar with function norms ( $L_2$ , etc.). The **common parametrization** establishes **correspondences**. We don't have that for structures or shapes.
- ◆ Partial matches need to be considered -- notion of **support**  $\sigma$  for the match.
- ◆ What group of **aligning transforms** is to be considered?
- ◆ Is the resulting distance a **metric**?



Not for partial matches

$$\delta(A, C) \leq \delta(A, B) + \delta(B, C)$$

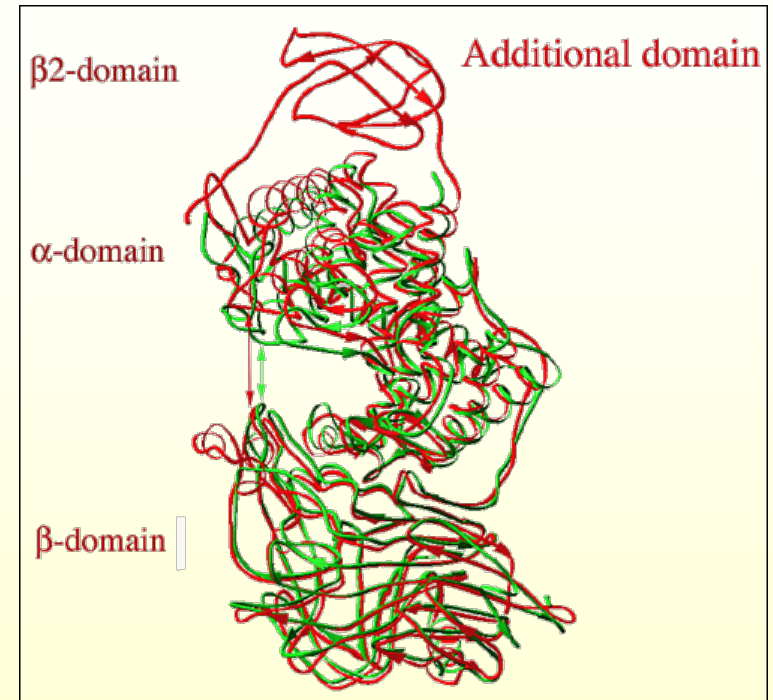
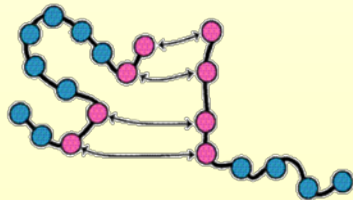


$$\delta(\sigma(A), \sigma(B))$$



# Simultaneous Estimation

- ◆ We are given two shapes  $A$  and  $B$ , each in its own coordinate system
- ◆ We must establish **correspondences** between certain parts (the **alignment supports**) of  $A$  and  $B$
- ◆ We must find an optimal **transform** that best **aligns** the supports of  $A$  and  $B$
- ◆ We must **score** this choice of supports and transform to produce a **distance measure**  $\delta$



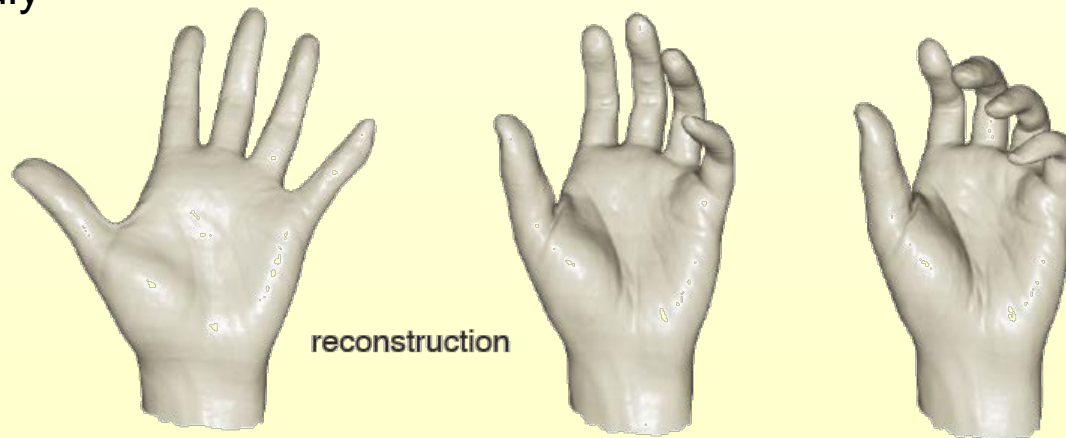
In computing the score,

1. what distance metric do we use?
2. how do we aggregate distances?
3. how do we trade-off larger supports for larger aggregate distance?

# Degrees of Freedom

## ◆ Transform estimation

- ◆ A rigid motion has 6 degrees of freedom (3 for translation and 3 for rotation)
- ◆ We typically estimate the motion using many more pairs of corresponding points, so the problem is **overdetermined** (which is good, given noise, outliers, etc – use least squares approaches)
- ◆ More general transforms require more degrees of freedom. When shape deformations are allowed, the degrees of freedom can grow very rapidly



# A Double Whammy

- ◆ Estimate correspondences
- ◆ Estimate the aligning transform
- ◆ Gives rise to combinatorial searches
- ◆ Transforms can be non-linear

Hard optimization problems!

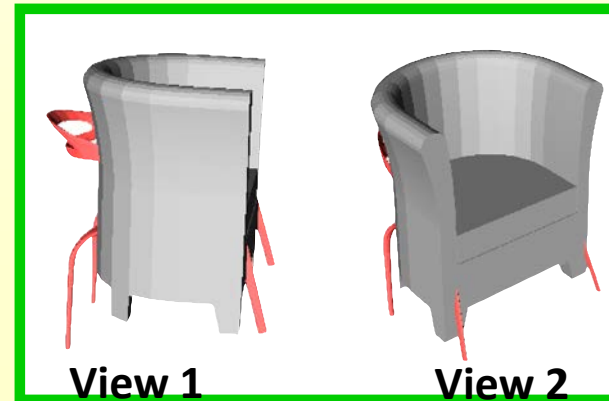
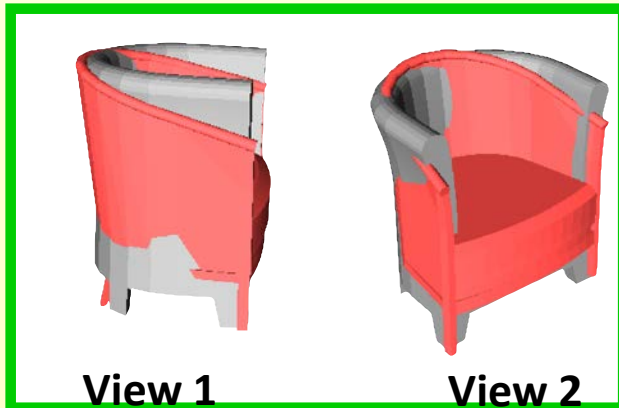
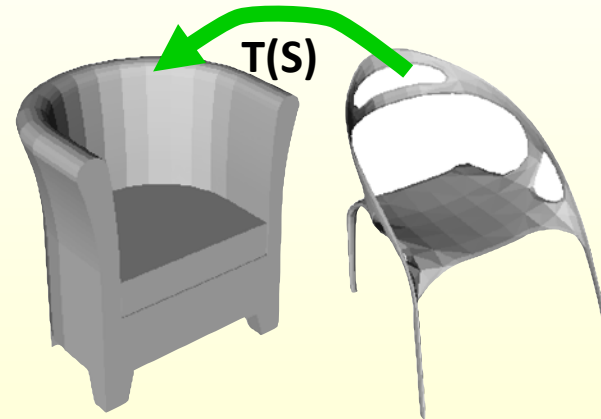
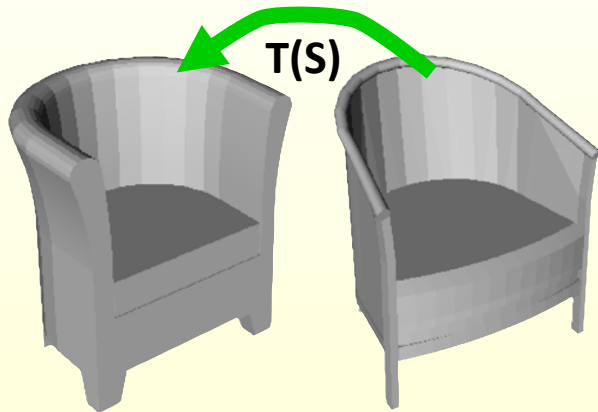
Good features help

Low-dimensionality of some transforms helps

# Rigid Alignment

# General Rigid Alignment

Search for a rigid motion that best aligns the shapes, even if the shapes are different



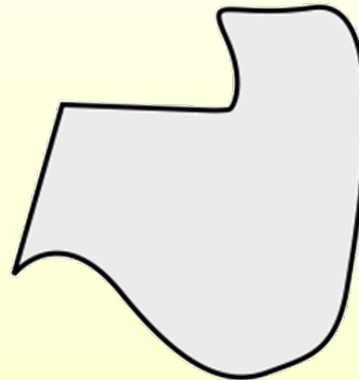
# Rigid Alignment

Search for a rigid motion that best aligns the shapes, even if the shapes are different

$M_1$



$M_2$



$$M_1 \approx T(M_2)$$

$T$  : translation + rotation

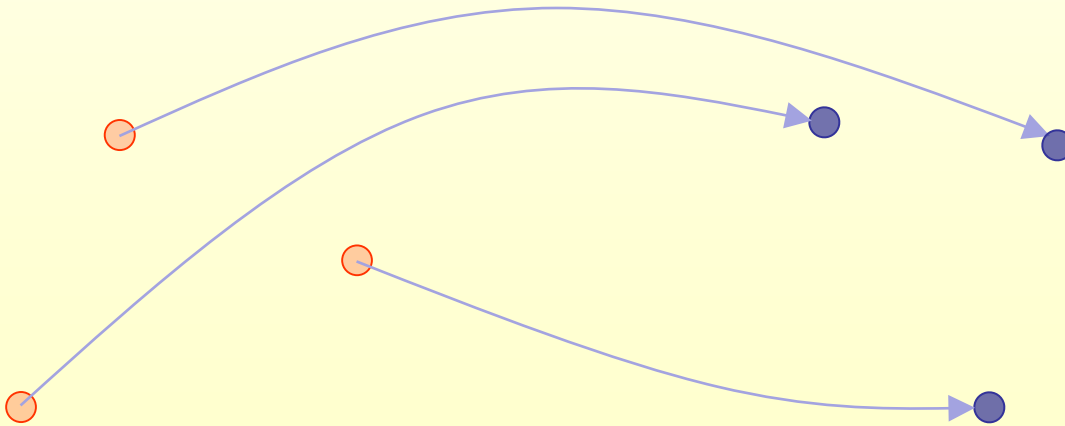
# ICP: Optimal Transformation for Point Clouds

Problem formulation, when given correspondences:

1. Given two sets points:  $\{x_i\}, \{y_i\}, i = 1..n$  in  $\mathbb{R}^3$ . Find the rigid transform:

$\mathbf{R}, t$  that minimizes:

$$\sum_{i=1}^N \|\mathbf{R}x_i + t - y_i\|_2^2$$



# Simplest Case: Rigid Alignment, Given Correspondences

- ◆ We are given two sets of **corresponding** points  $x_1, x_2, \dots, x_n$  and  $y_1, y_2, \dots, y_n$  in  $\mathbb{R}^3$ . We wish to compute the rigid transform  $T$  that best aligns  $x_1$  to  $y_1$ ,  $x_2$  to  $y_2$ , ..., and  $x_n$  to  $y_n$ .
- ◆ We define the error to be minimized by

$$\min_T \sum_{i=1}^n \|T(x_i) - y_i\|^2$$

MSE error, RMS distance, ...

- ◆ Old Problem:
  - ◆ Known and solved as the **orthogonal Procrustes problem** in Factor Analysis (Statistics) [Shönemann, 1966]
  - ◆ Known and solved as the **absolute orientation problem** in Photogrammetry [Horn, 1986]
  - ◆ Also med stat ... etc





# SVD-Based Solution

- ◆ A rigid motion  $T$  is a combination of a translation  $a$  and a rotation  $R$ , so that  $T(x) = R(x) + a$ .
- ◆ If we place the origin of our coordinate system at the mean of the  $x_i$ 's, then the quantity to be minimized simplifies to (up to some constants):

$$\min_{a, R} \left( \sum_{i=1}^n |y_i - a|^2 - 2 \sum_{i=1}^n \langle R x_i, y_i \rangle \right)$$

- ◆ Note that the translational and rotational parts factor. The translational part  $a$  can easily be seen to be optimized by

$$a = \frac{1}{n} \sum_{i=1}^n y_i$$

The centroids of the two point sets have to be aligned!

# The Rotation Part

- ◆ Define

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

$$X = [x_1 - \bar{x}, \dots, x_n - \bar{x}]^T$$

$$Y = [y_1 - \bar{y}, \dots, y_n - \bar{y}]^T$$

- ◆ Here  $X$  and  $Y$  are 3 by  $n$  matrices.

A diagram illustrating matrix multiplication. On the left, a light blue horizontal rectangle contains the letter 'X'. To its right is a purple 'x' symbol. Further right is a light blue vertical rectangle containing 'Y<sup>T</sup>'. To its right is a purple '=' symbol. Finally, on the far right, is a small light blue square.

- ◆ Now compute the SVD\*

$$XY^T = UDV^T \quad (3 \times 3)$$

- ◆  $U$  and  $V$  are 3 by 3 orthogonal matrices, and  $D$  is a diagonal matrix with decreasing non-negative entries along the diagonal (the singular values).
- ◆ Define  $S$  by

$$S = \begin{cases} I, & \text{if } \det U \det V = 1 \\ \text{diag}(1, \dots, 1, -1), & \\ \text{otherwise} \end{cases}$$

- ◆ Then

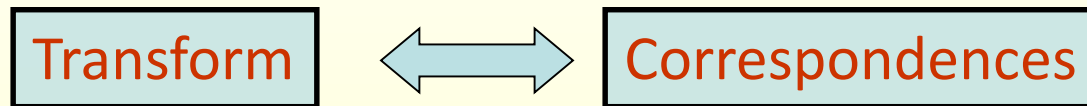
$$R = USV^T$$

\*SVD = singular value decomposition

**$O(n)$  algorithm!**

# How to Get Correspondences?

A *chicken-and-egg problem*: if we knew the optimal aligning transform, then we could get correspondences by **proximity** (possibly with the aid of some global adjustment, e.g., dynamic programming)



Guess one, estimate the other, and *iterate!*

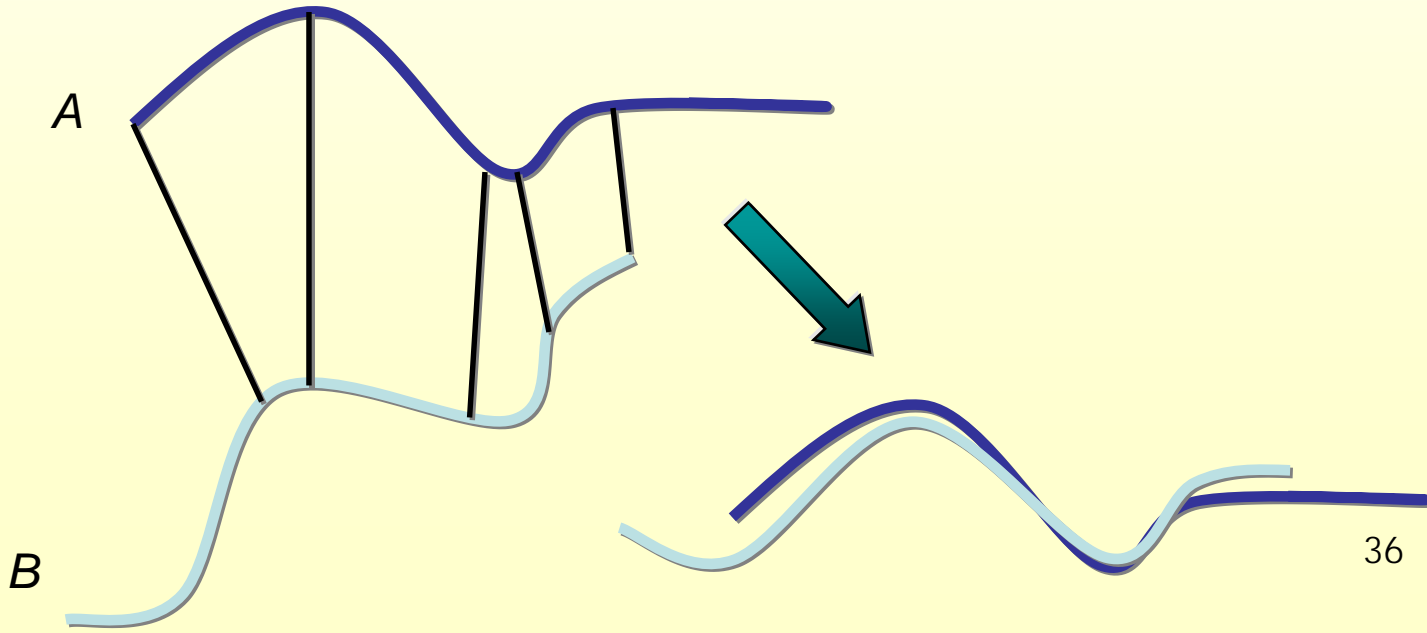
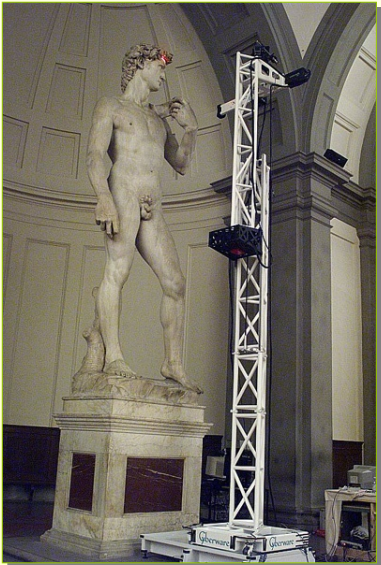
EM like

- Correspondences from proximity (**Iterated Closest Pair**)
- Correspondences from local shape descriptors (**Shape Features**)
- Transform from voting schemes (**RANSAC, Geometric Hashing**)
- Combinations

Local Methods:  
Iterated Closest Pair  
(ICP)  
[Besl, McKay 1992]

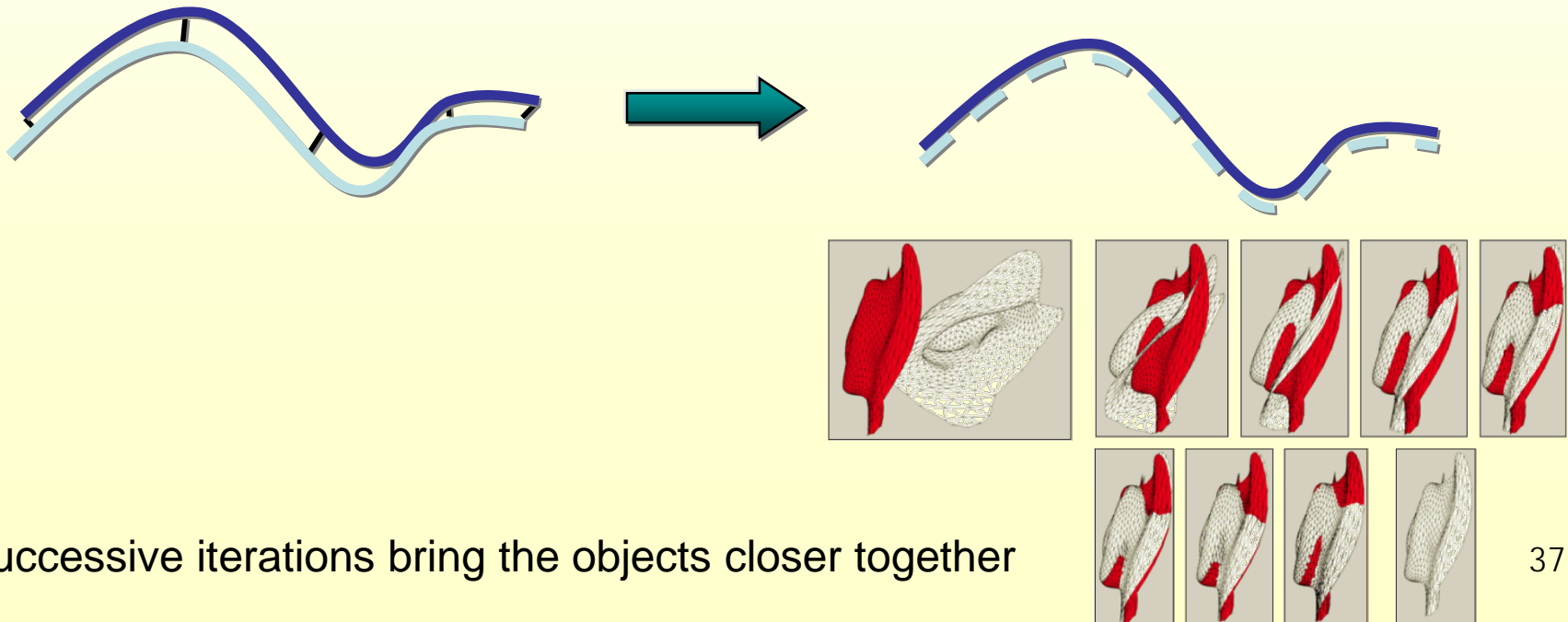
# Aligning 3D Data

- ◆ How to find correspondences: User input? Feature detection? Local shape signatures?
- ◆ When  $A$  and  $B$  are partial scans of the same stationary object captured in a consistent setting, use the simplest alternative: **assume closest points correspond**



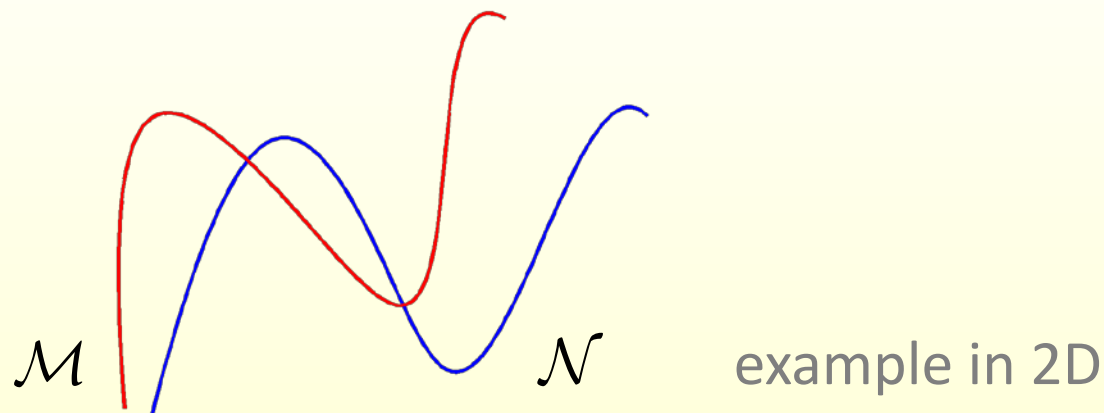
# Aligning 3D Data

- ◆ Align the  $A$  points to their closest  $B$  neighbors, then repeat
- ◆ Converges, if starting positions are “close enough”



# Iterative Closest Point (ICP)

- Classical approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes,  $\mathcal{M}$  and  $\mathcal{N}$ , iterate:

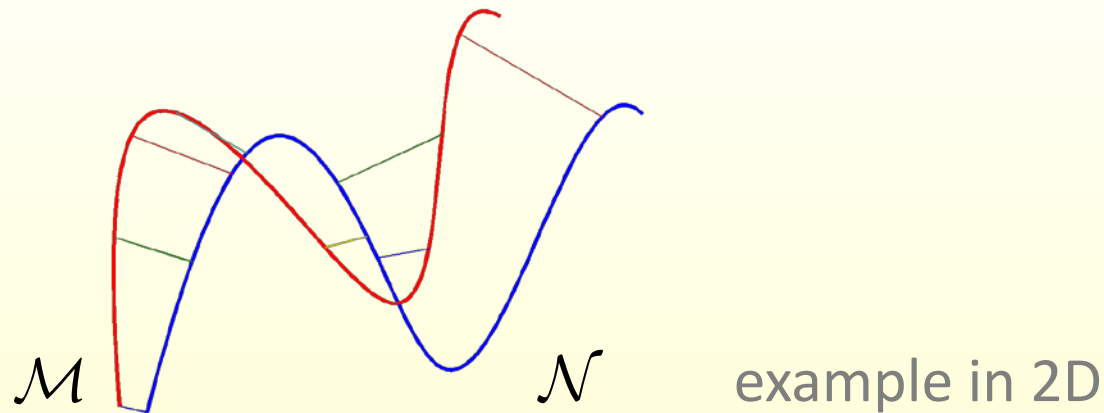
1. For each  $x_i \in \mathcal{M}$  find **nearest** neighbor  $y_i \in \mathcal{N}$
2. Find optimal transformation  $\mathbf{R}, t$  minimizing:

$$\arg \min_{R, t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Classic problem,  
solvable by SVD

# Iterative Closest Point (ICP)

- Classical approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes,  $\mathcal{M}$  and  $\mathcal{N}$ , iterate:

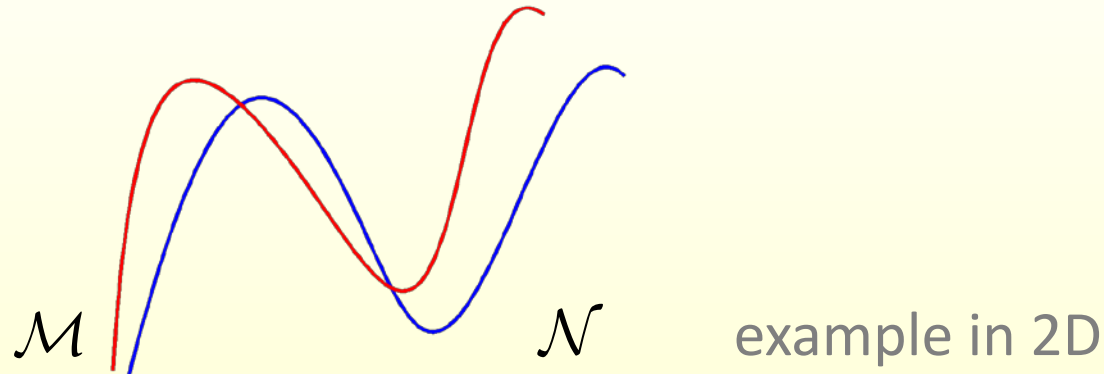
- For each  $x_i \in \mathcal{M}$  find **nearest** neighbor  $y_i \in \mathcal{N}$
- Find optimal transformation  $\mathbf{R}, t$  minimizing:

$$\arg \min_{R, t} \sum_i \|R x_i + t - y_i\|_2^2$$

Classic problem,  
solvable by SVD

# Iterative Closest Point (ICP)

- Classical approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes,  $\mathcal{M}$  and  $\mathcal{N}$ , iterate:

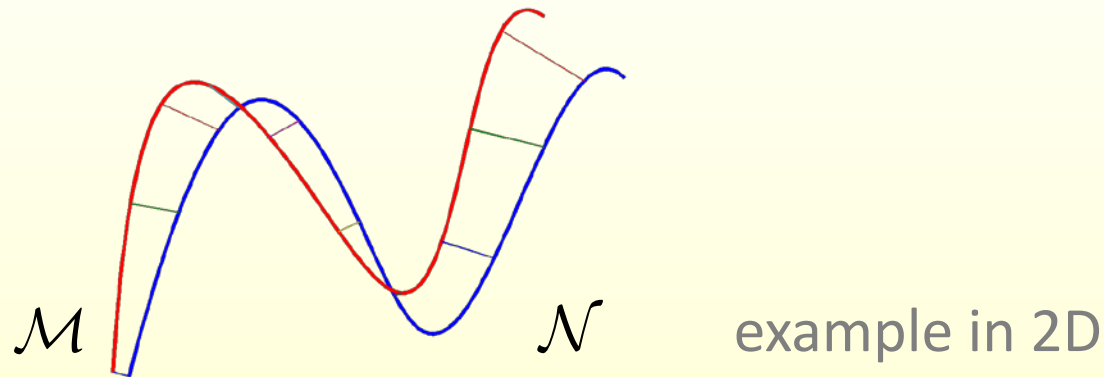
1. For each  $x_i \in \mathcal{M}$  find **nearest** neighbor  $y_i \in \mathcal{N}$
2. Find optimal transformation  $\mathbf{R}, t$  minimizing:

$$\arg \min_{R, t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Classic problem,  
solvable by SVD

# Iterative Closest Point (ICP)

- Classical approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes,  $\mathcal{M}$  and  $\mathcal{N}$ , iterate:

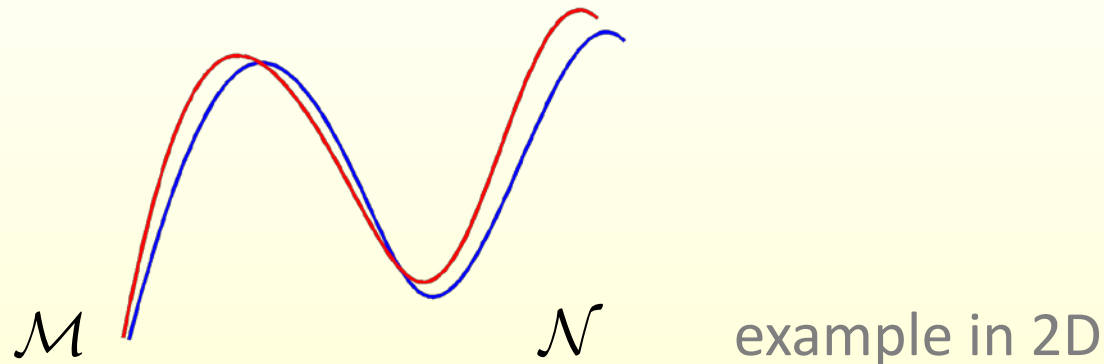
1. For each  $x_i \in \mathcal{M}$  find **nearest** neighbor  $y_i \in \mathcal{N}$
2. Find optimal transformation  $\mathbf{R}, t$  minimizing:

$$\arg \min_{R, t} \sum_i \|R x_i + t - y_i\|_2^2$$

Classic problem,  
solvable by SVD

# Iterative Closest Point (ICP)

- Classical approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes,  $\mathcal{M}$  and  $\mathcal{N}$ , iterate:

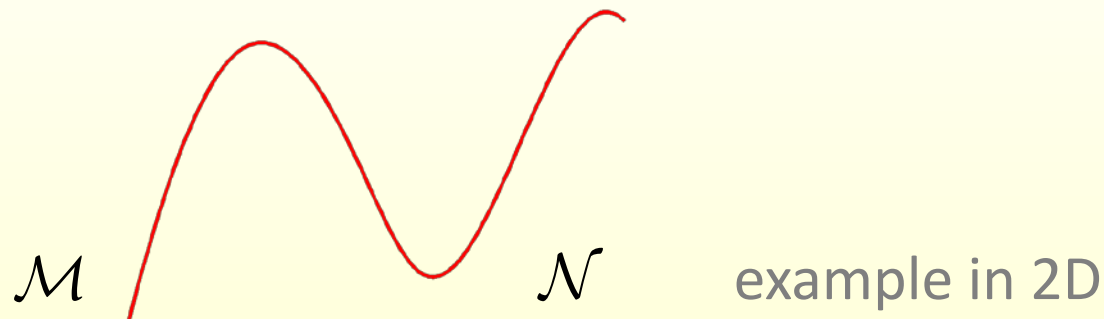
1. For each  $x_i \in \mathcal{M}$  find **nearest** neighbor  $y_i \in \mathcal{N}$
2. Find optimal transformation  $\mathbf{R}, t$  minimizing:

$$\arg \min_{R,t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Classic problem,  
solvable by SVD

# Iterative Closest Point (ICP)

- Classical approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes,  $\mathcal{M}$  and  $\mathcal{N}$ , iterate:

1. For each  $x_i \in \mathcal{M}$  find **nearest** neighbor  $y_i \in \mathcal{N}$
2. Find optimal transformation  $\mathbf{R}, t$  minimizing:

$$\arg \min_{R, t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Classic problem,  
solvable by SVD

# Convergence Theorem

- ◆ The ICP algorithm always converges monotonically to a local minimum, with respect to the MSE distance objective function
  - ◆ Correspondence step improves error bound – because of nearest neighbor computation
  - ◆ Rotation/Translation step improves error bound – because of transform optimization

# Time Analysis

Each iteration includes three main steps

A. Finding the closest points:

$O(N_M)$  per point

$O(N_M * N_S)$  total

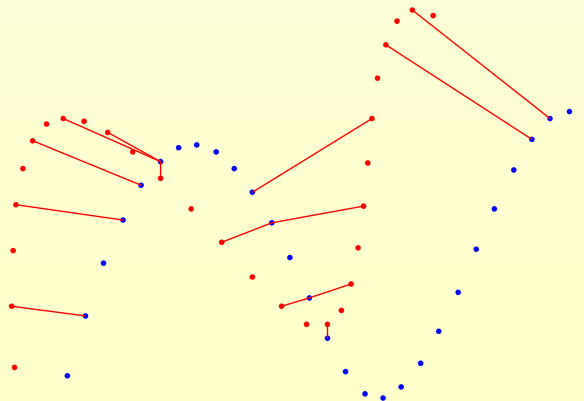
B. Calculating the optimal alignment:  $O(N_S)$

C. Updating the scene:  $O(N_S)$

Fast nearest-neighbor data structures can be very helpful here, e.g., a *k-d* tree

# Variations of ICP

1. **Selecting** source points (from one or both scans): sampling
2. **Matching** to points in the other mesh
3. **Weighting** the correspondences
4. **Rejecting** certain (outlier) point pairs
5. **Assigning** an error metric to the current transform
6. **Minimizing** the error metric w.r.t. the transformation

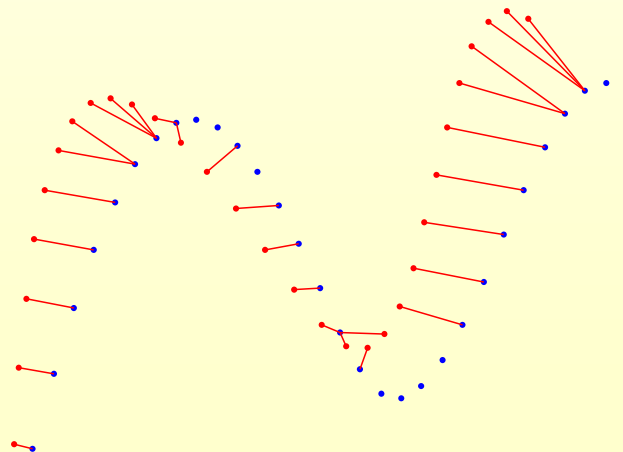


# Iterative Closest Point

Given a pair of shapes,  $X$  and  $Y$ , iterate:

1. For each  $x_i \in X$  find **nearest** neighbor  $y_i \in Y$ .
2. Find deformation  $\mathbf{R}, t$  minimizing:

$$\sum_{i=1}^N \|\mathbf{R}x_i + t - y_i\|_2^2$$



Ideally, most correspondences are 1-1

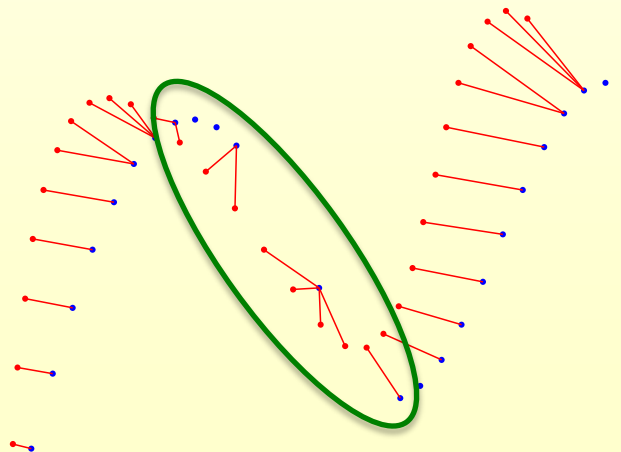
# Iterative Closest Point

Given a pair of shapes,  $X$  and  $Y$ , iterate:

1. For each  $x_i \in X$  find **nearest** neighbor  $y_i \in Y$ .
2. Find deformation  $\mathbf{R}, t$  minimizing:

$$\sum_{i=1}^N \|\mathbf{R}x_i + t - y_i\|_2^2$$

Problem:  
uneven sampling



# Iterative Closest Point

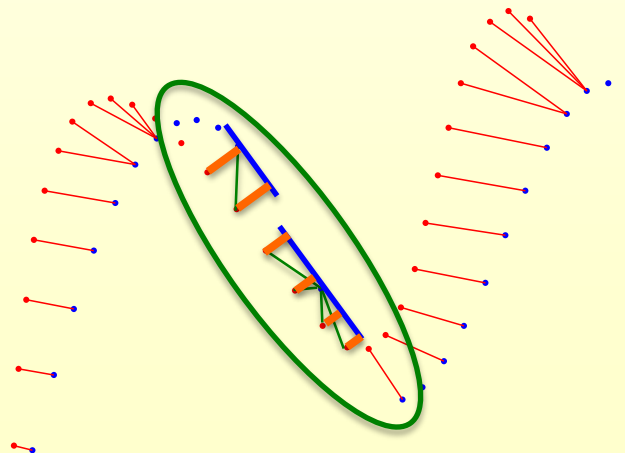
Given a pair of shapes,  $X$  and  $Y$ , iterate:

1. For each  $x_i \in X$  find **nearest** neighbor  $y_i \in Y$ .
2. Find deformation  $\mathbf{R}, t$  minimizing:

$$\sum_{i=1}^N d(\mathbf{R}x_i + t, P(y_i))^2 = \sum_{i=1}^N ((\mathbf{R}x_i + t - y_i)^T \mathbf{n}_{y_i})^2$$

**Solution:**

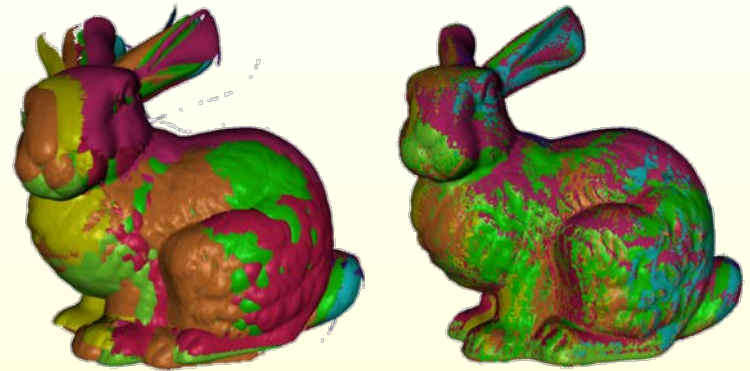
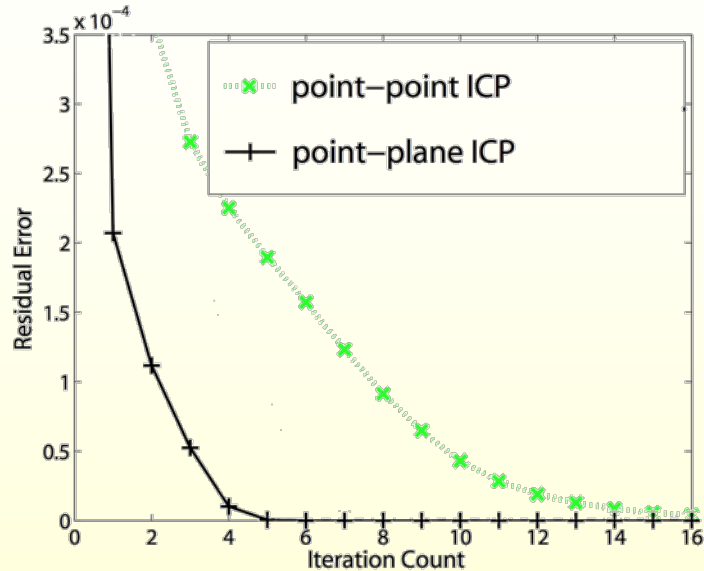
Minimize distance to  
the tangent plane



No longer a closed-form  
solution

Chen, Medioni, '91

# Iterative Closest Point

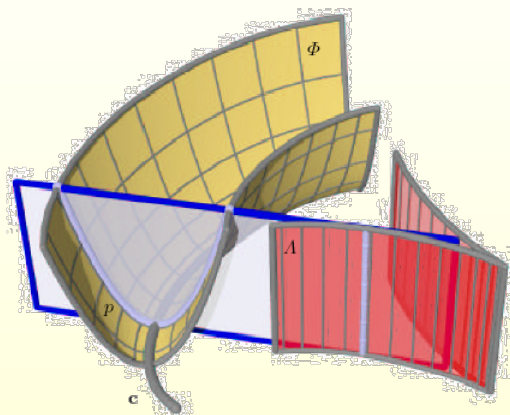


Aligning the bunny to itself:  
Point-to-plane always wins in the end-game.

# ICP Without Correspondences

- ◆ ICP without correspondences
  - ◆ define a **quadratic approximant to the square distance function**

[Pottman & Hofer, 02]

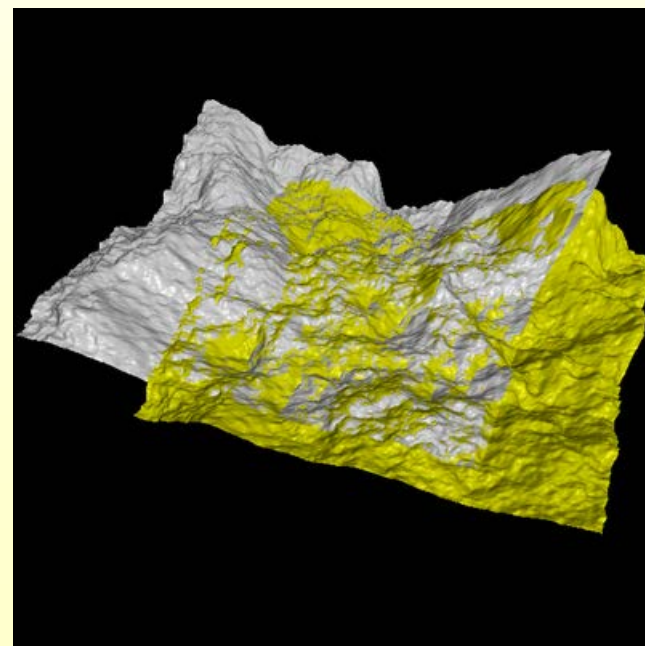


$$F(x_1, x_2, x_3) = \frac{d}{d - \rho_1} x_1^2 + \frac{d}{d - \rho_2} x_2^2 + x_3^2$$

$$F(x_1, x_2) = \frac{d}{d - \rho} x_1^2 + x_2^2$$

← curvature

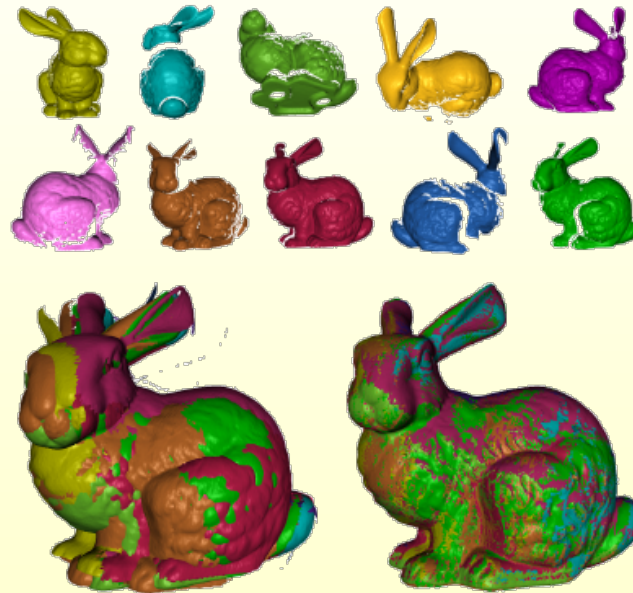
- ◆ perform iterative gradient-descent in this field
- ◆ point to foot-point distance
  - ◆ case  $d$  is large: classical ICP
  - ◆ case  $d$  is small: point-to-plane ICP



# Global Matching Methods

# Global Matching

Given shapes in *arbitrary* positions, find their alignment:



*Robust Global  
Registration*  
Gelfand et al. SGP 2005

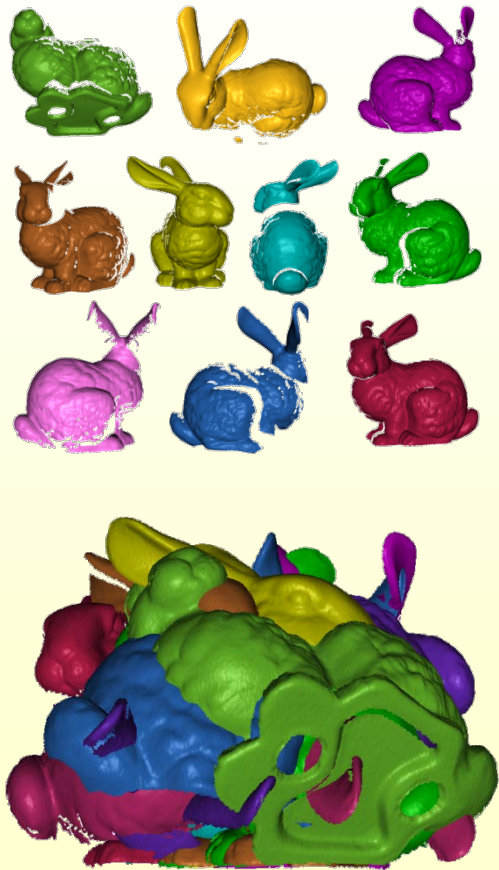
Can be approximate, since will refine later using e.g. ICP

# Partial Alignment

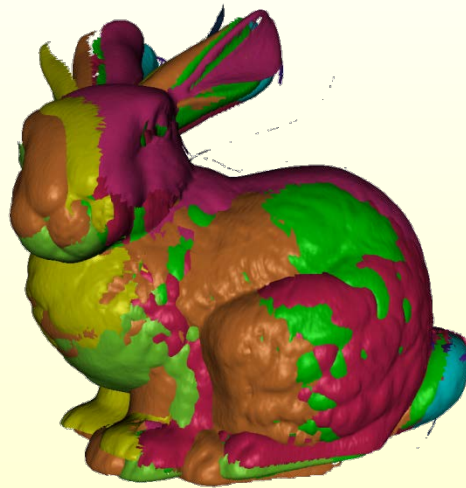


After 6 iterations

# Multiple Alignment Results



Input: 10 scans



Approximate alignment



Refined by ICP

Bundle adjustment

# Global Matching – Approaches

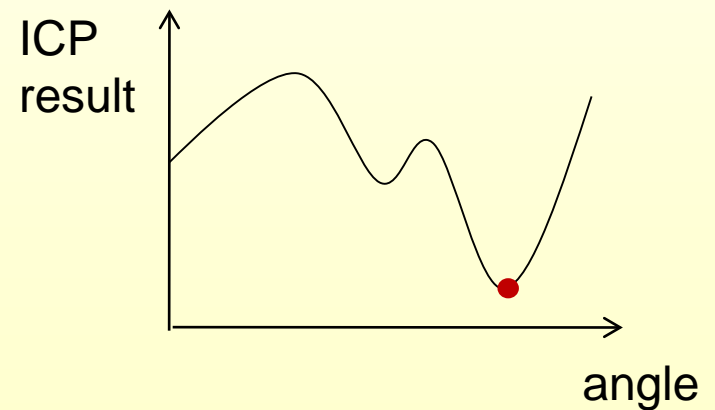
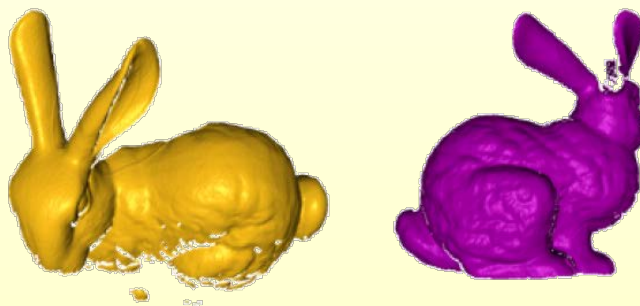
Several classes of approaches:

1. Exhaustive Search
2. Normalization
3. Random Sampling
4. Invariance

# Exhaustive Search:

Compare (ideally) all alignments

- ◆ Sample the space of possible initial alignments.
- ◆ Correspondence is determined by the alignment at which models are closest.



Very common in biology: e.g., protein docking

# Exhaustive Search:

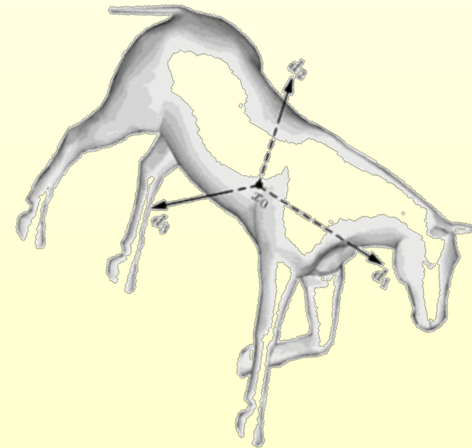
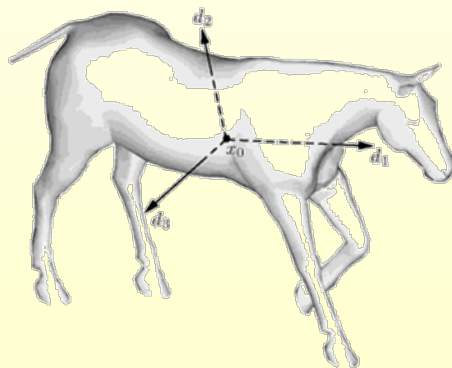
## Compare at all alignments

- ◆ Sample the space of possible initial alignments
- ◆ Correspondence is determined by the alignment at which models are closest
- ◆ Provides optimal result
- ◆ Can be unnecessarily slow
- ◆ Does not generalize well to non-rigid deformations

# Normalization – Canonical Poses

There are only a handful of initial configurations that are important.

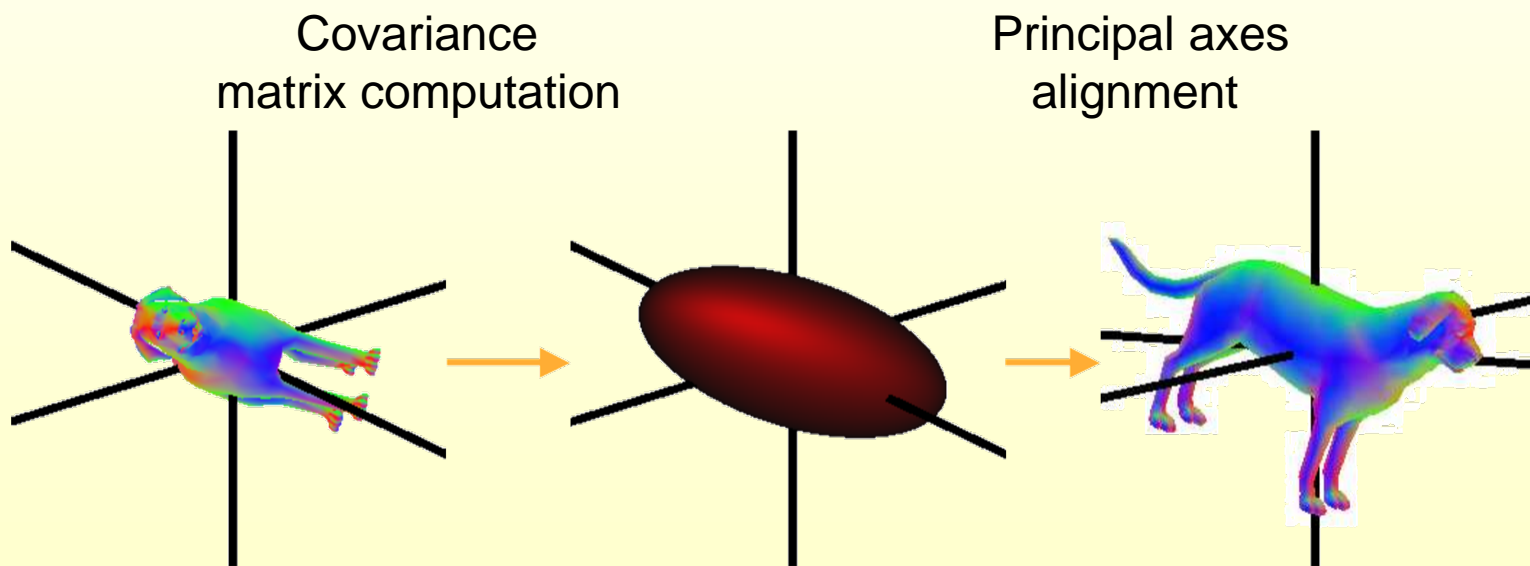
Can center all shapes at the origin and use PCA to find the principal directions of the shape.



In addition sometimes try all permutations of x-y-z.

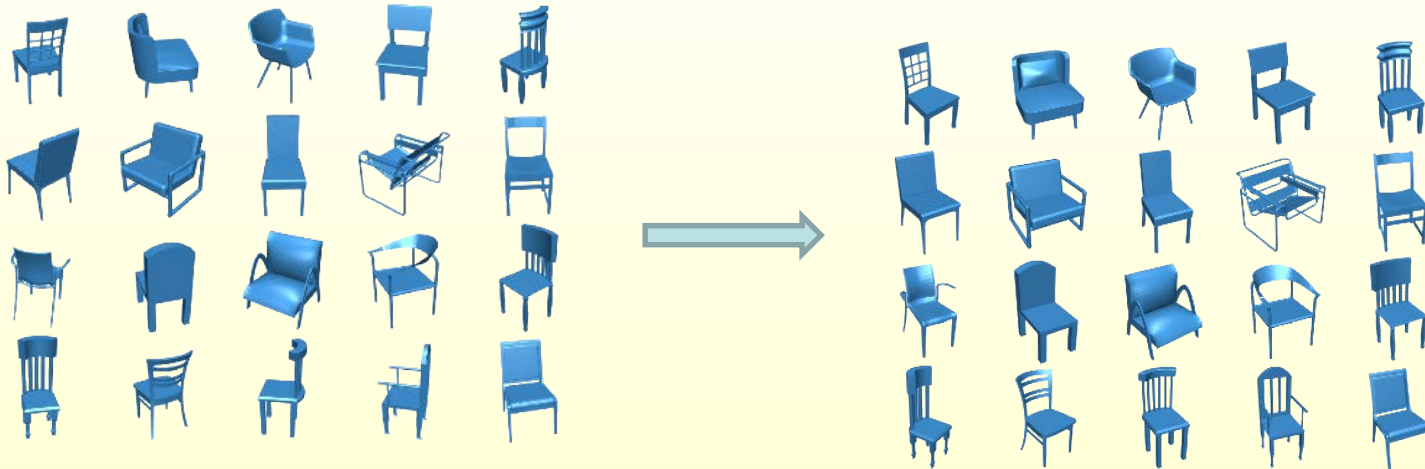
# PCA-Based Alignment

- ◆ Use PCA to place models into canonical coordinate frames
- ◆ Then align those frames



# Normalization – Canonical Poses

There are only a handful of initial configurations that are important.

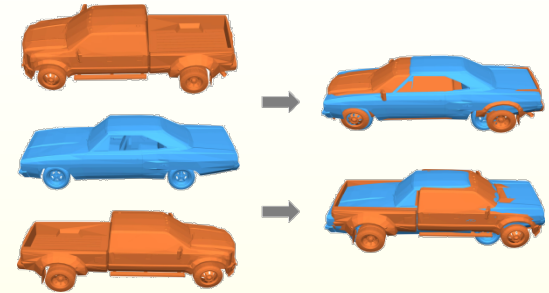


Works well if we have complete shapes and no noise.

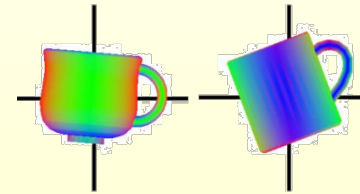
Fails for partial scans, outliers, high noise, etc.

# Problems with PCA

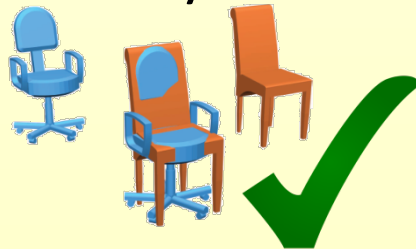
- ◆ Principal axes are not consistently oriented
  - ◆ Symmetries cause problems



- ◆ Axes are unstable when principal values are similar



- ◆ Partial similarity

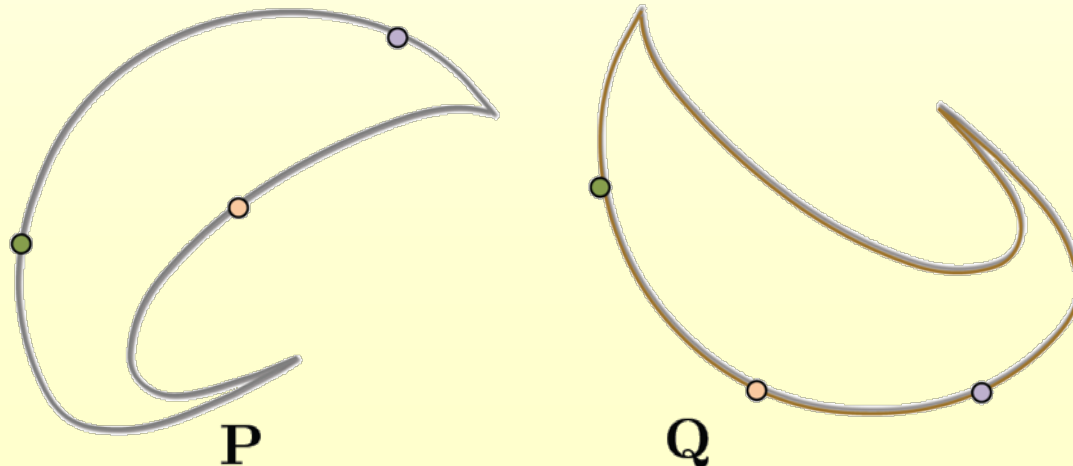


# Random Sampling (RANSAC)

ICP only needs 3 point pairs! – Rigid motion space is 6-dimensional.

Robust and Simple approach. Iterate between:

1. Pick a random pair of 3 points on model & scan
2. Estimate alignment, and check for error.



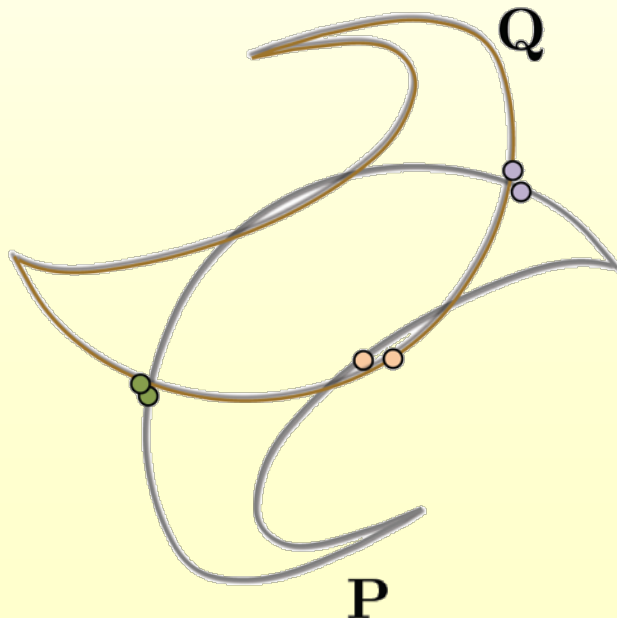
Guess and  
verify

# Random Sampling (RANSAC)

ICP only needs 3 point pairs! – Rigid motion space is 6-dimensional.

Robust and simple approach. Iterate between:

1. Pick a random pair of 3 points on model & scan
2. Estimate alignment, and check for error.



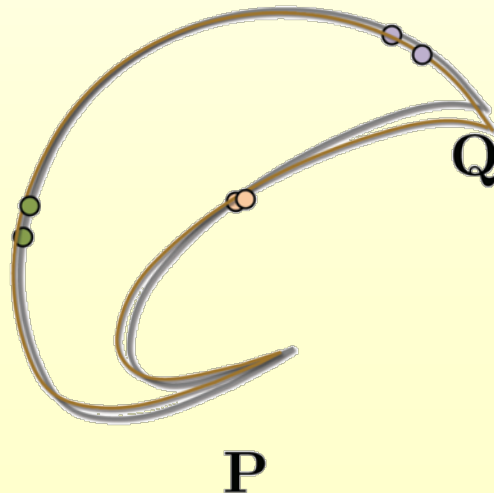
Guess and  
verify

# Random Sampling (RANSAC)

ICP only needs 3 point pairs! – Rigid motion space is 6-dimensional.

Robust and simple approach. Iterate between:

1. Pick a random pair of 3 points on model & scan
2. Estimate alignment, and check for error.



Can be expensive!

Can also refine the final result. Picks don't have to be exact.

Guess and verify

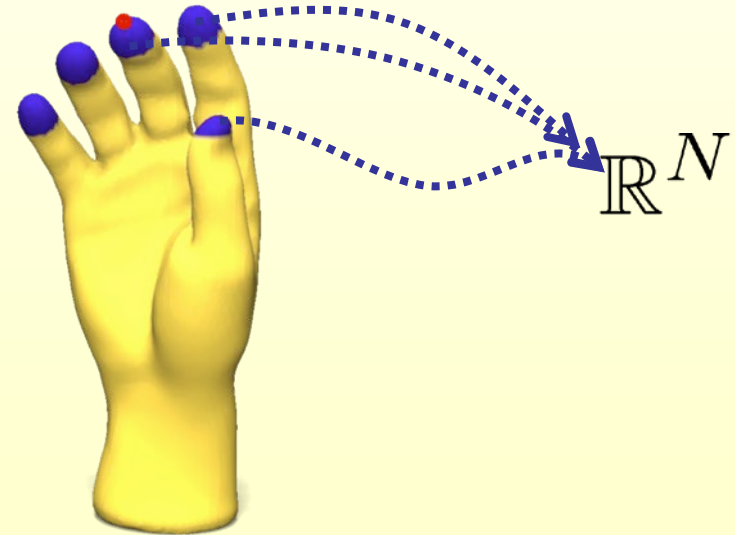
# Global Matching – Invariant Features

Try to characterize the shape using properties that are invariant under the desired set of transformations.

Conflicting interests – invariance vs. informativeness.

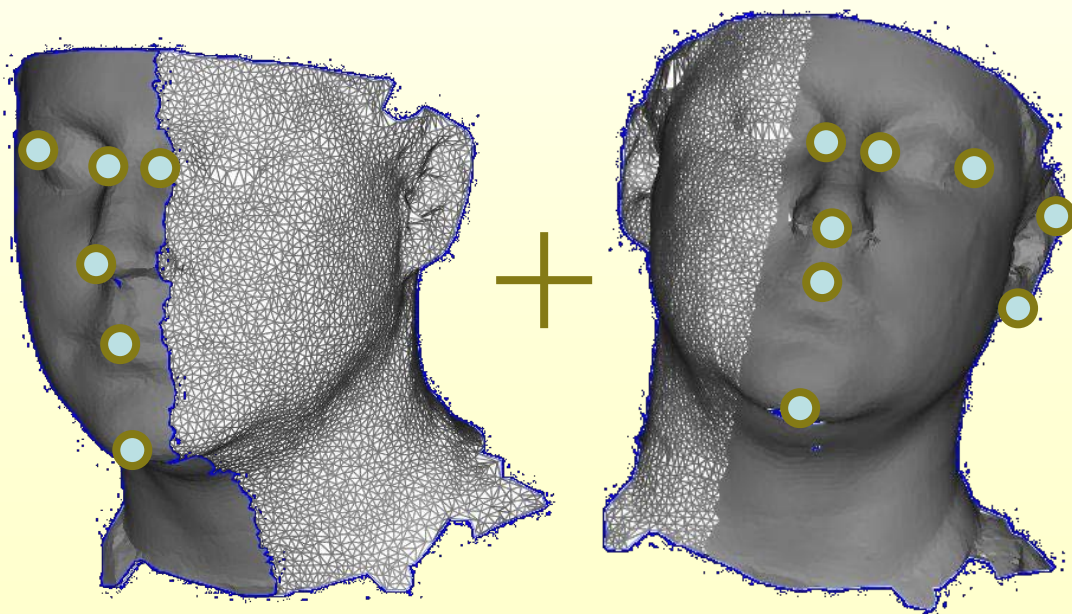
The most common pipeline:

1. identify salient feature points
2. compute informative and commensurable descriptors.



# Matching Using Feature Points

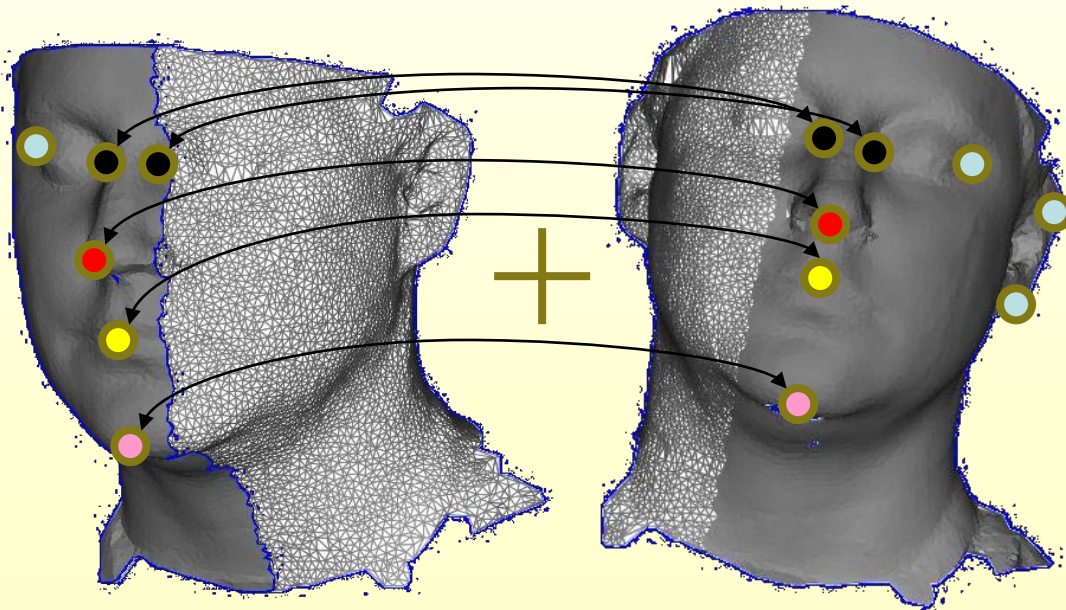
1. Find **feature points** on the two scans (we'll come back to that issue)



Partially Overlapping Scans

# Approach

1. (Find feature points on the two scans)
2. Establish **correspondences**

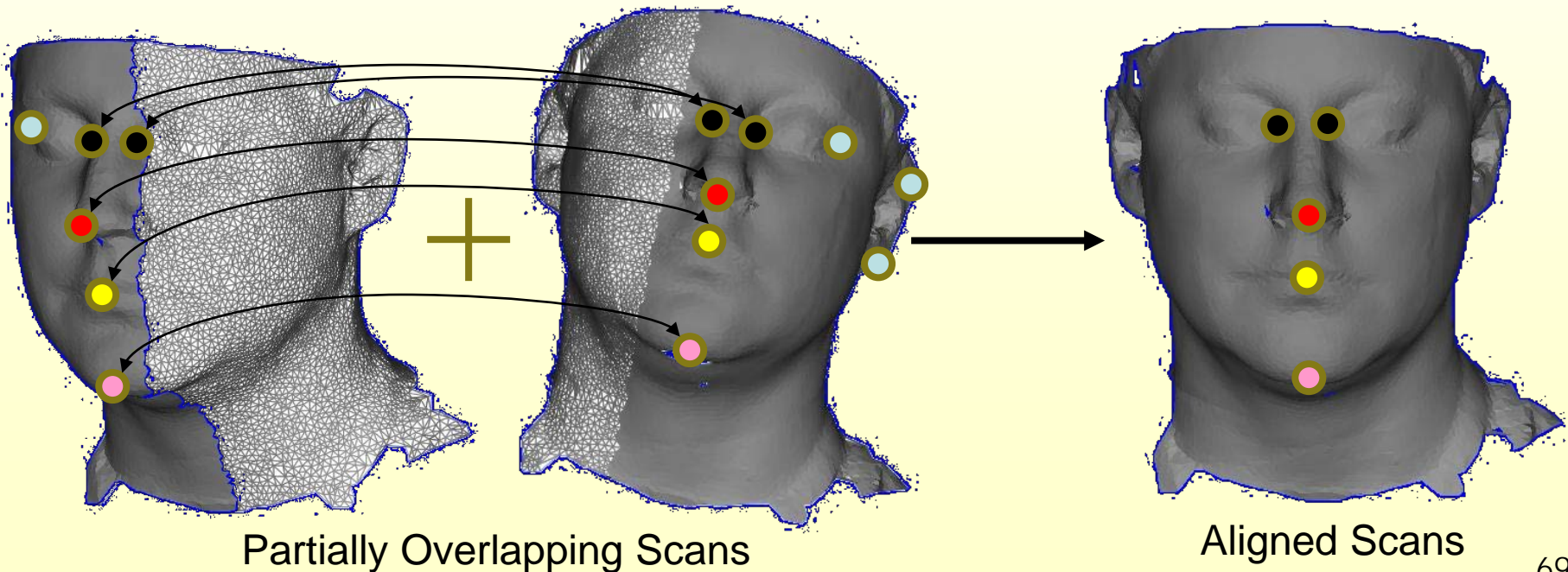


Partially Overlapping Scans

# Approach

1. (Find feature points on the two scans)
2. Establish correspondences
3. Compute the aligning transformation

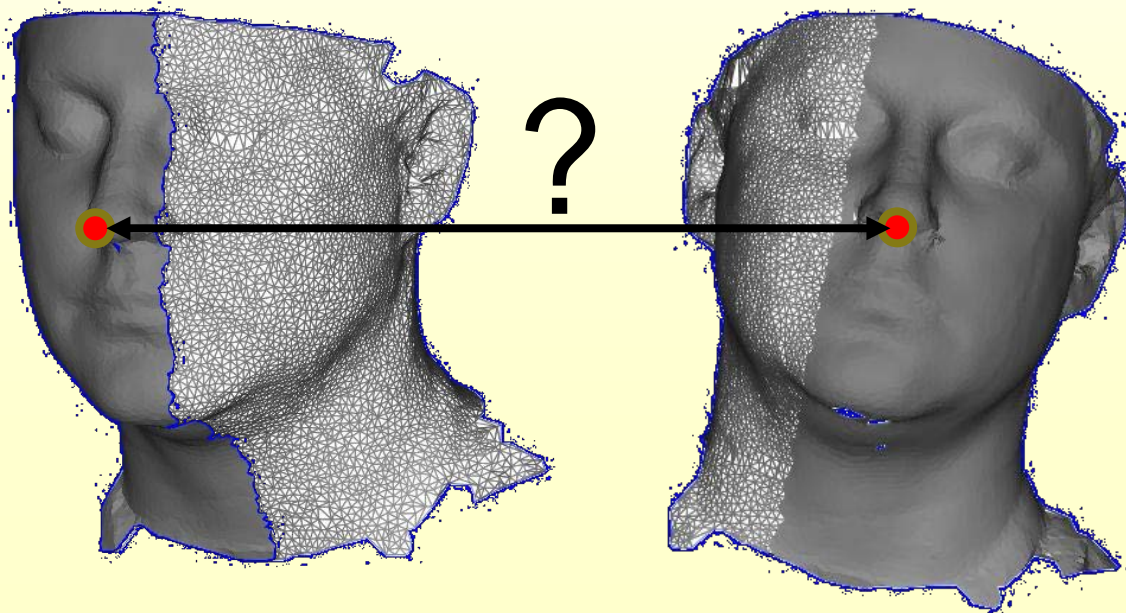
Preserve features  
Various regularizers



# Correspondence

## Goal:

Identify when two points on different scans represent the same feature

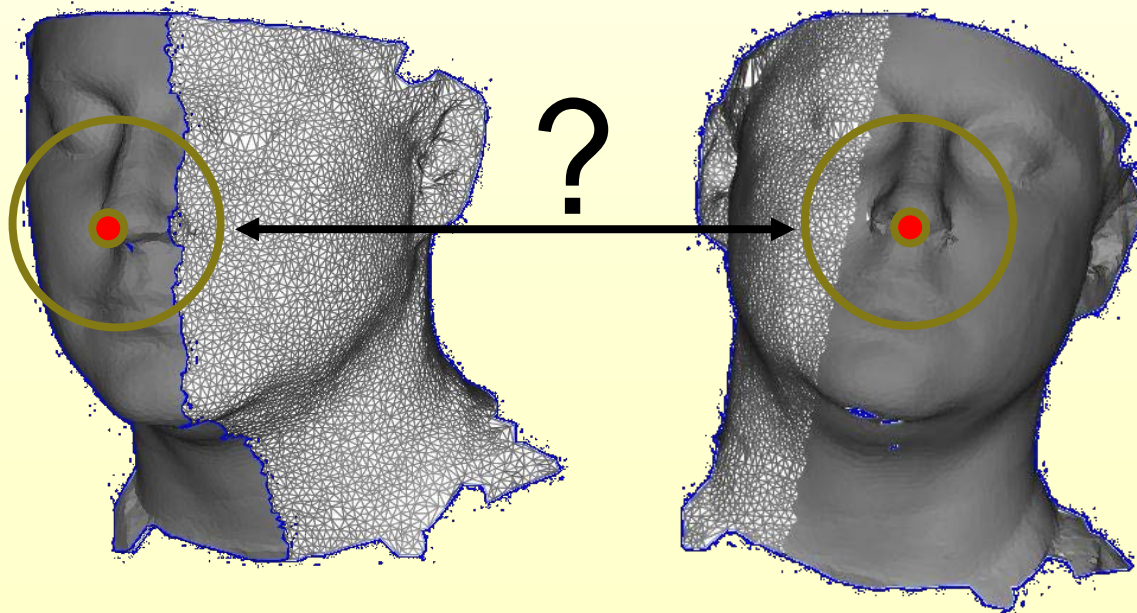


# Correspondence

## Goal:

Identify when two points on different scans represent the same feature:

Are the surrounding regions similar?

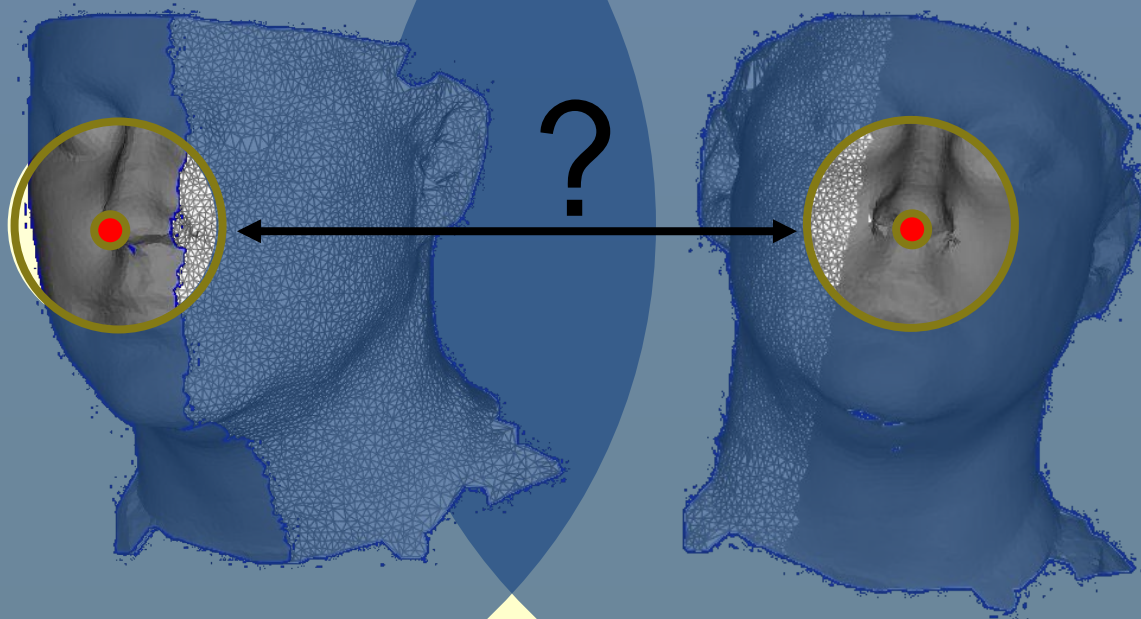


# Correspondence

## Goal:

Identify when two points on different scans represent the same feature:

Are the surrounding regions similar?



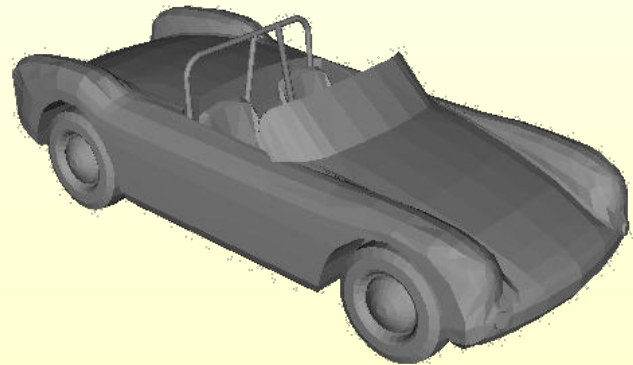
# Shape Descriptors

# Global Shape Similarity

# Global Similarity

## More Generally:

Given two 3D models, determine if they represent the same/similar shapes



# Global Similarity

## More Generally:

Given two models, determine if they represent the same/similar shapes.



Models can have different:  
representations, tessellations, topologies, etc.

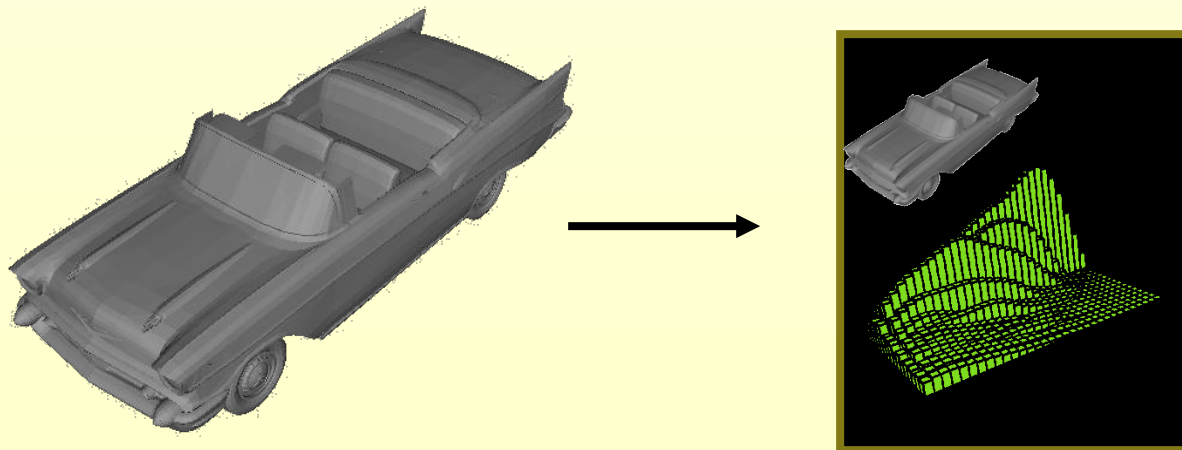
# Global Similarity

## Approach:

1. Represent each model by a **shape descriptor**:

- A structured abstraction of a 3D model
- That captures salient shape information

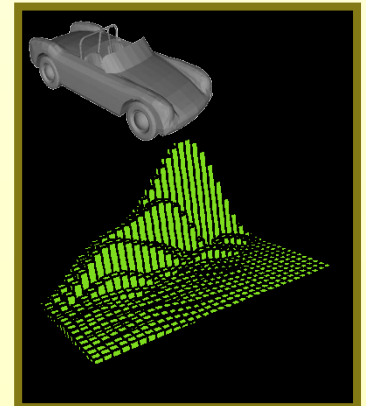
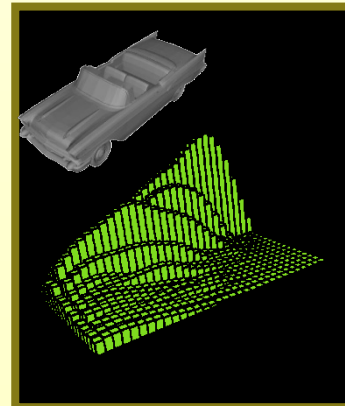
Typically a  
high-dimensional vector



# Global Similarity

## Approach:

1. Represent each model by a **shape descriptor**
2. Compare shapes by comparing their shape descriptors
3. Is the descriptor designed, or learned?



# Shape Descriptors: Examples

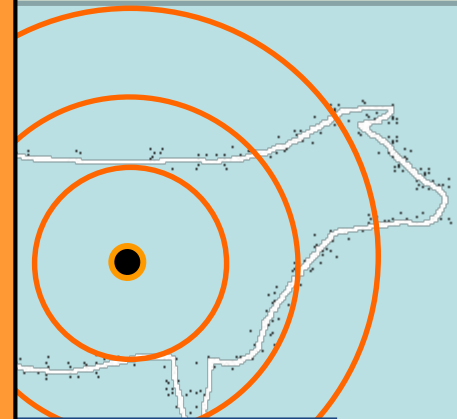
[Ankerst *et al.* 1999]

## Shape Histograms

Shape descriptor stores a histogram of how much surface area resides within different concentric shells in space

### ◆ Feature desiderata

- ◆ Informativeness
- ◆ Stability
- ◆ Invariance
- ◆ Efficiency
- ◆ Adaptability
- ◆ ...



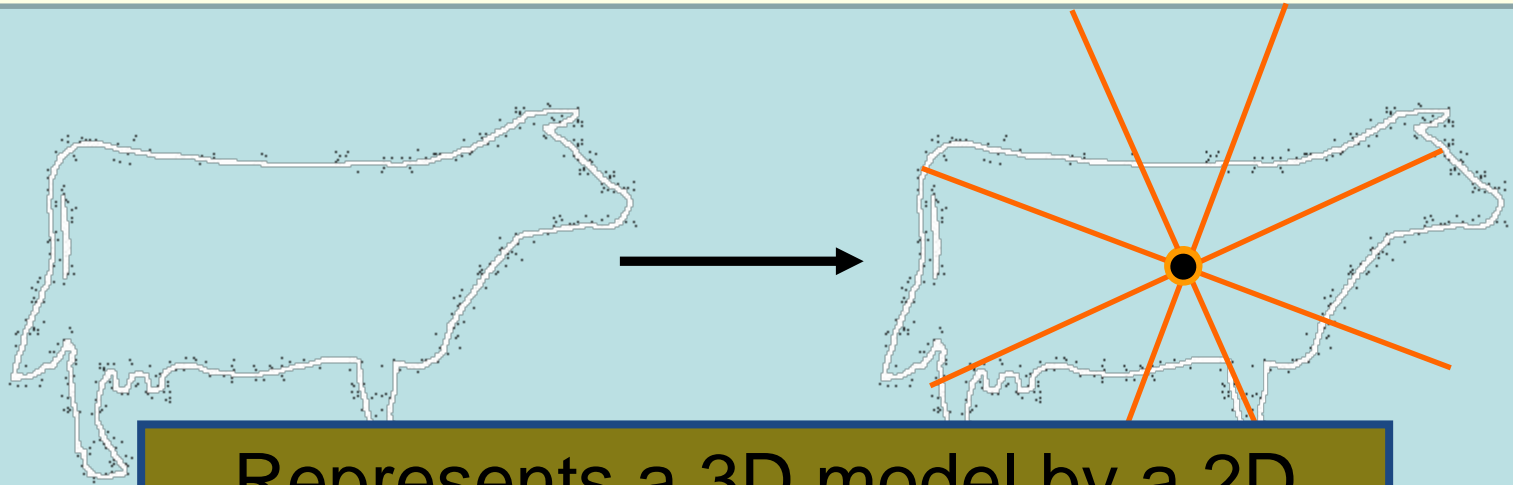
Represents a 2D model by a 1D  
(radial) array of values

# Shape Descriptors: Examples

[Ankerst *et al.* 1999]

## Shape Histograms

Shape descriptor stores a histogram of how much surface area resides within different sectors in space



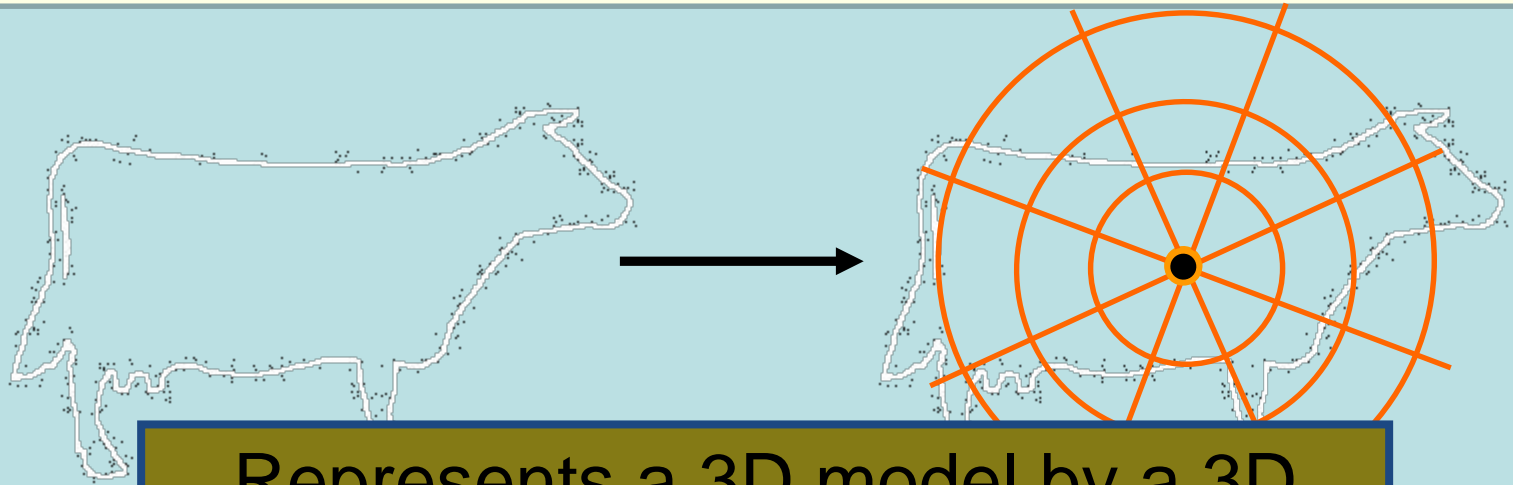
Represents a 3D model by a 2D  
(spherical) array of values

# Shape Descriptors: Examples

[Ankerst *et al.* 1999]

## Shape Histograms

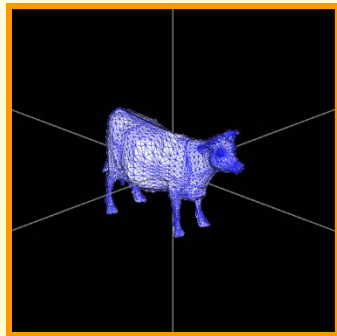
Shape descriptor stores a histogram of how much surface area resides within different shells and sectors in space



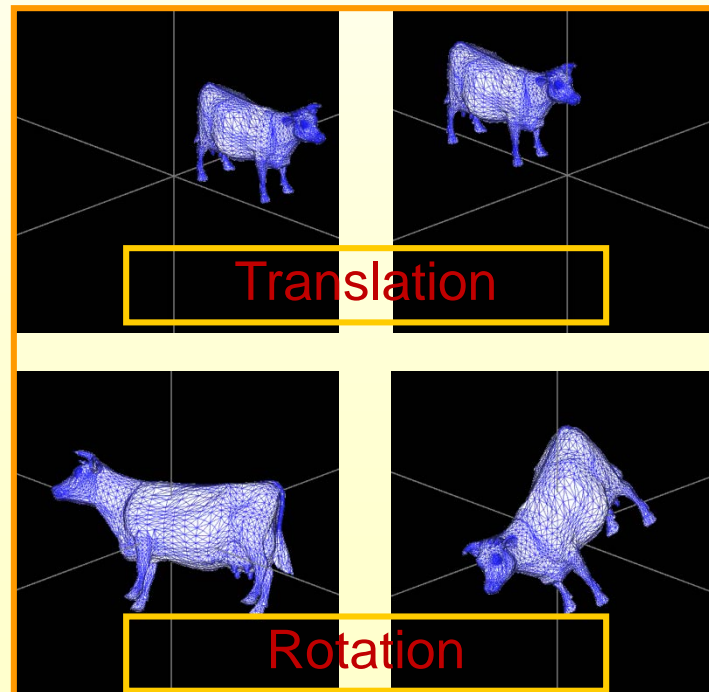
Represents a 3D model by a 3D  
(spherical x radial) array of values

# Shape Descriptors: Challenge

The descriptor must not change when a rigid body transformation (e.g., translation, rotation) is applied to the model

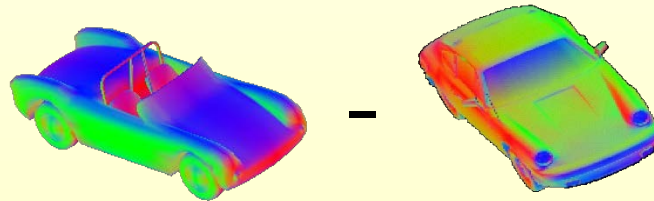


≈



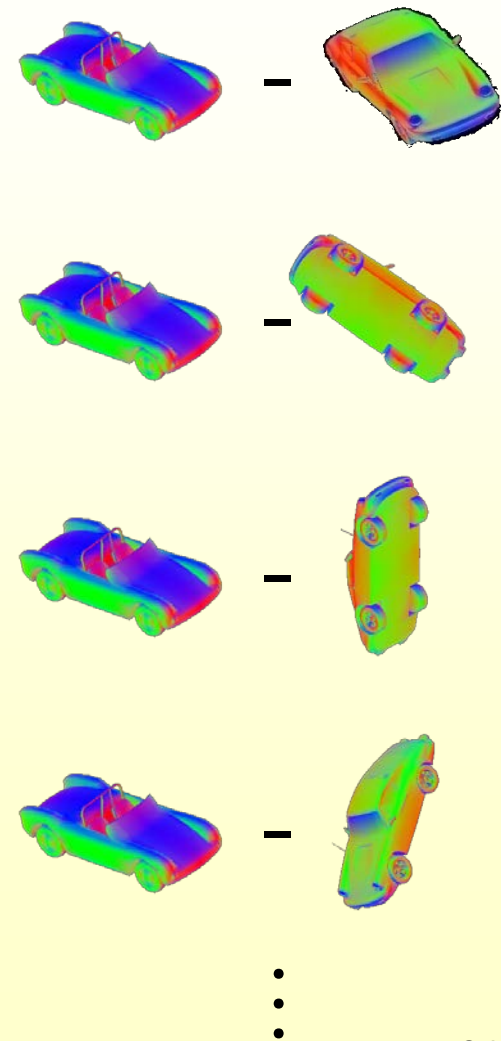
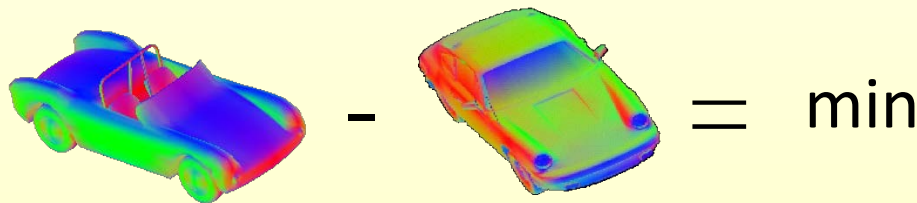
# Shape Descriptors: Challenge

In order to compare two 3D models, we must either make descriptors invariant to rigid motions, or we need to compare the shapes at their optimal alignment



# Shape Descriptors: Challenge

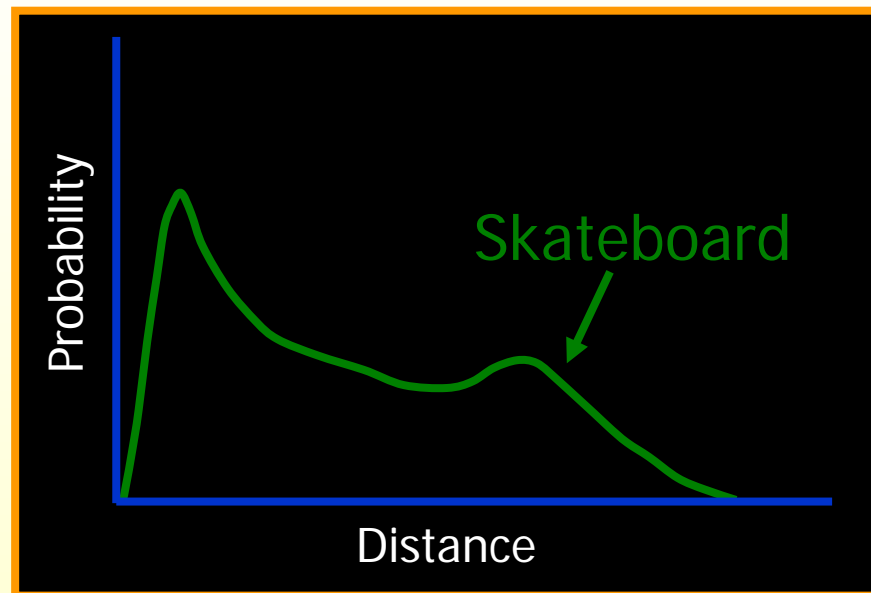
In order to compare two models,  
we need to compare them  
at their optimal alignment



# D2 Shape Distributions

## ◆ Properties

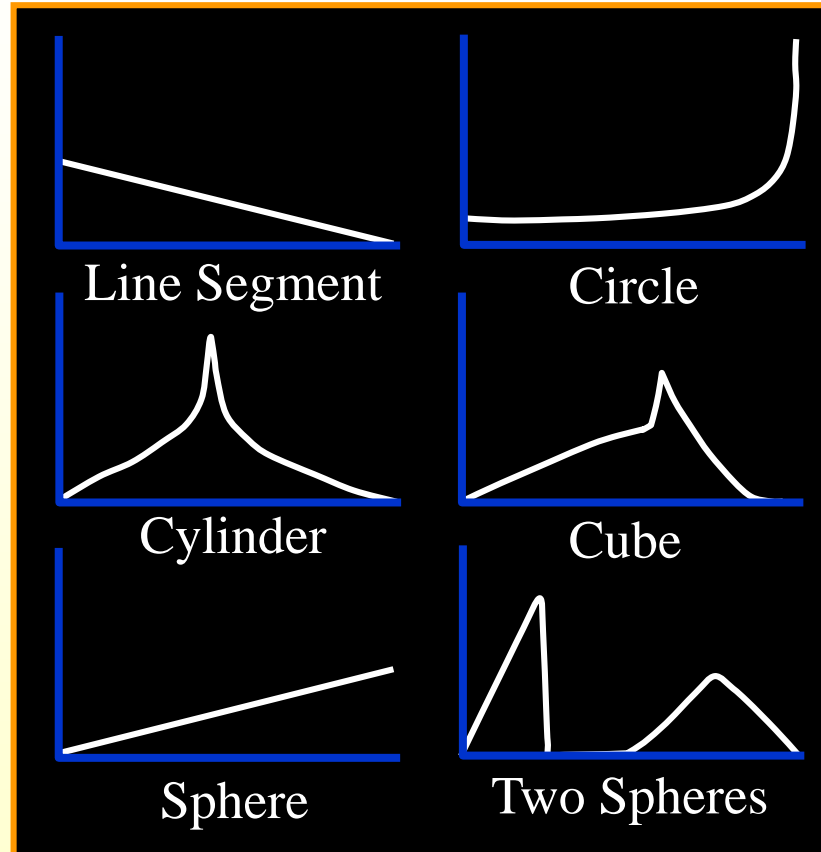
- ◆ Concise to store?
- ◆ Quick to compute?
- ◆ Invariant to transforms?
- ◆ Efficient to match?
- ◆ Insensitive to noise?
- ◆ Insensitive to topology?
- ◆ Robust to degeneracies?
- ◆ Invariant to deformations?
- ◆ Discriminating?



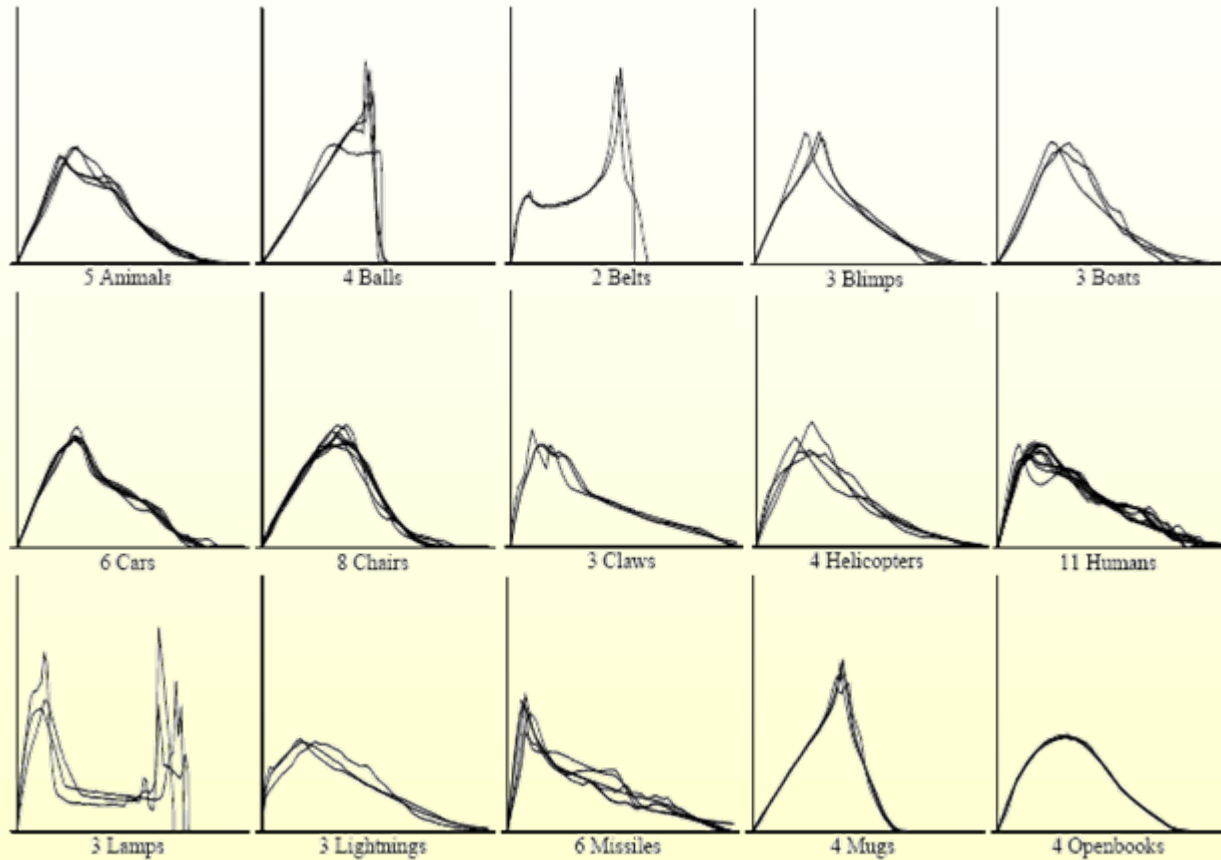
512 bytes (64 values)

0.5 seconds ( $10^6$  samples)

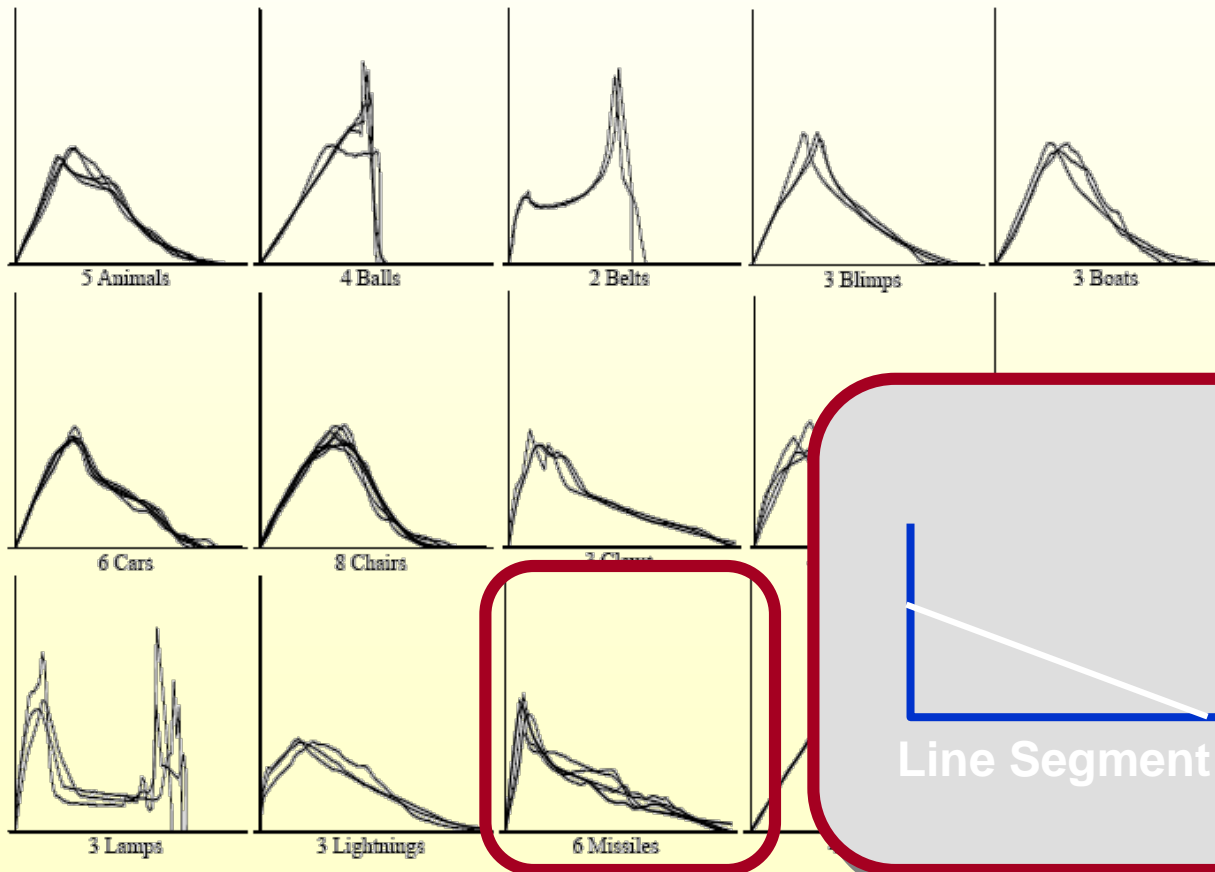
# Discriminating?



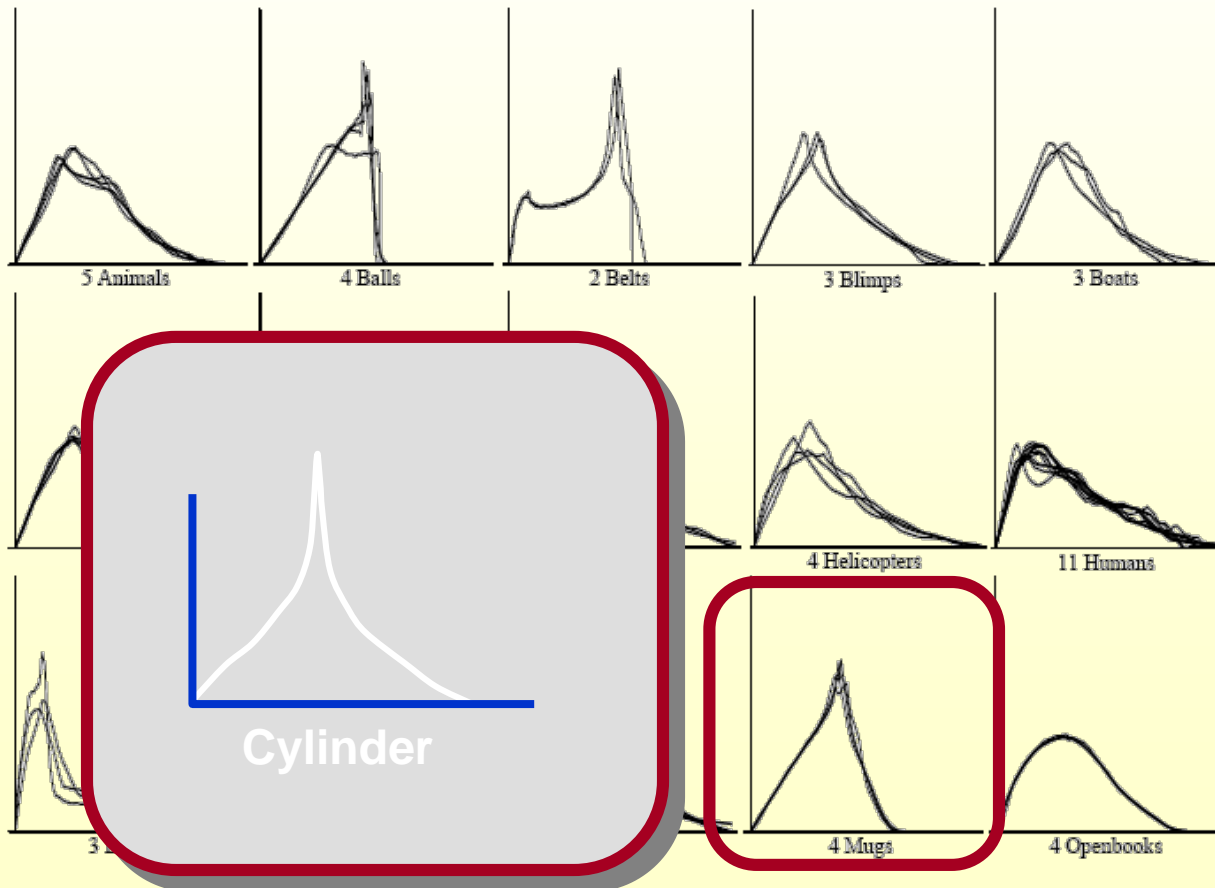
# Stable?



# Stable and Discriminating?



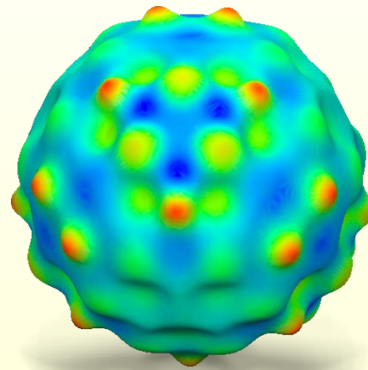
# Stable and Discriminating



# Local Features and Local Shape Similarity

# Classical Curvature

- ◆ Differential features can be noisy on meshes



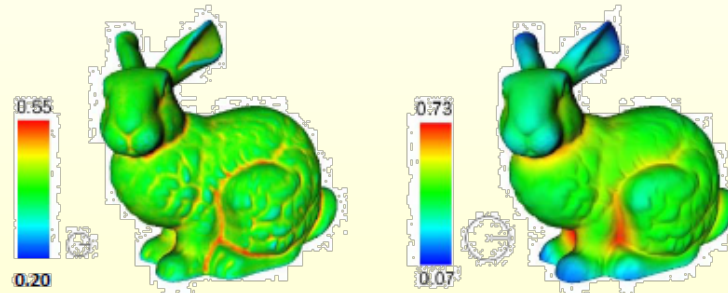
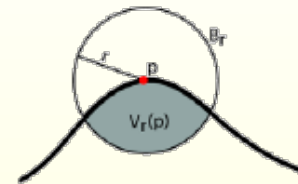
Curvature

# Integral Volume Descriptor

*Integral invariant signatures*, Manay et al. ECCV 2004

*Integral Invariants for Robust Geometry Processing*, Pottmann et al. 2007-2009

$$V_r(p) = \int_{B_r(p) \cap S} dx$$



## Relation to mean curvature

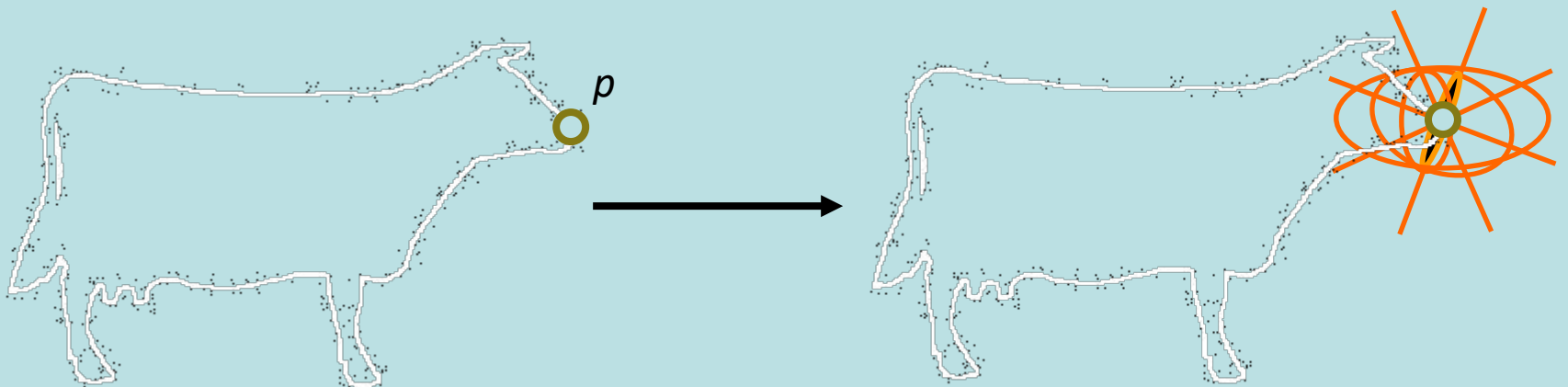
$$V_r(p) = \frac{2\pi}{3}r^3 - \frac{\pi H}{4}r^4 + O(r^5)$$

*Robust Global Registration*,  
Gelfand et al. 2005

# From Global to Local

To characterize the surface about a point  $p$ , take a global descriptor and:

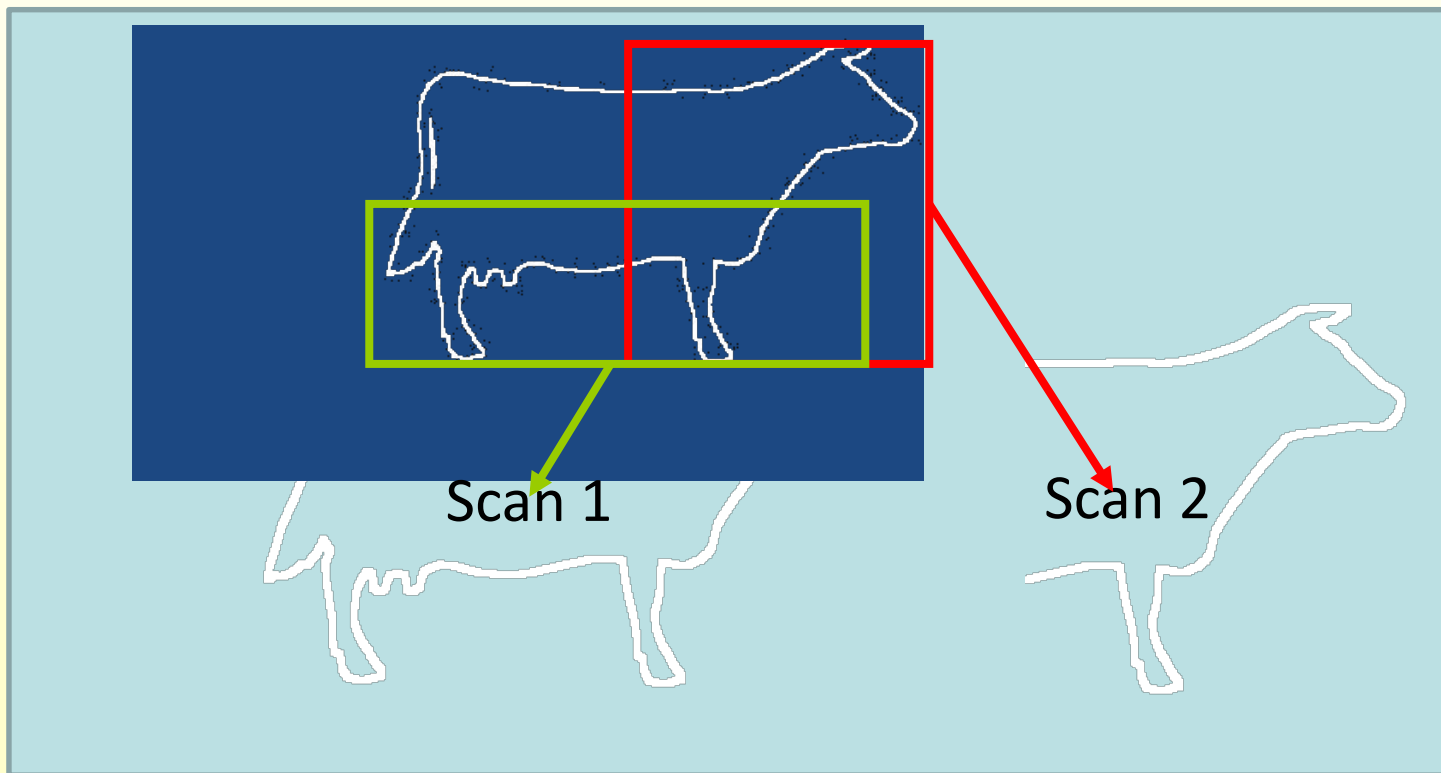
- ◆ center it about  $p$  (instead of the COM), and
- ◆ restrict the extent to a small region about  $p$ .



Shape histograms as local shape descriptors

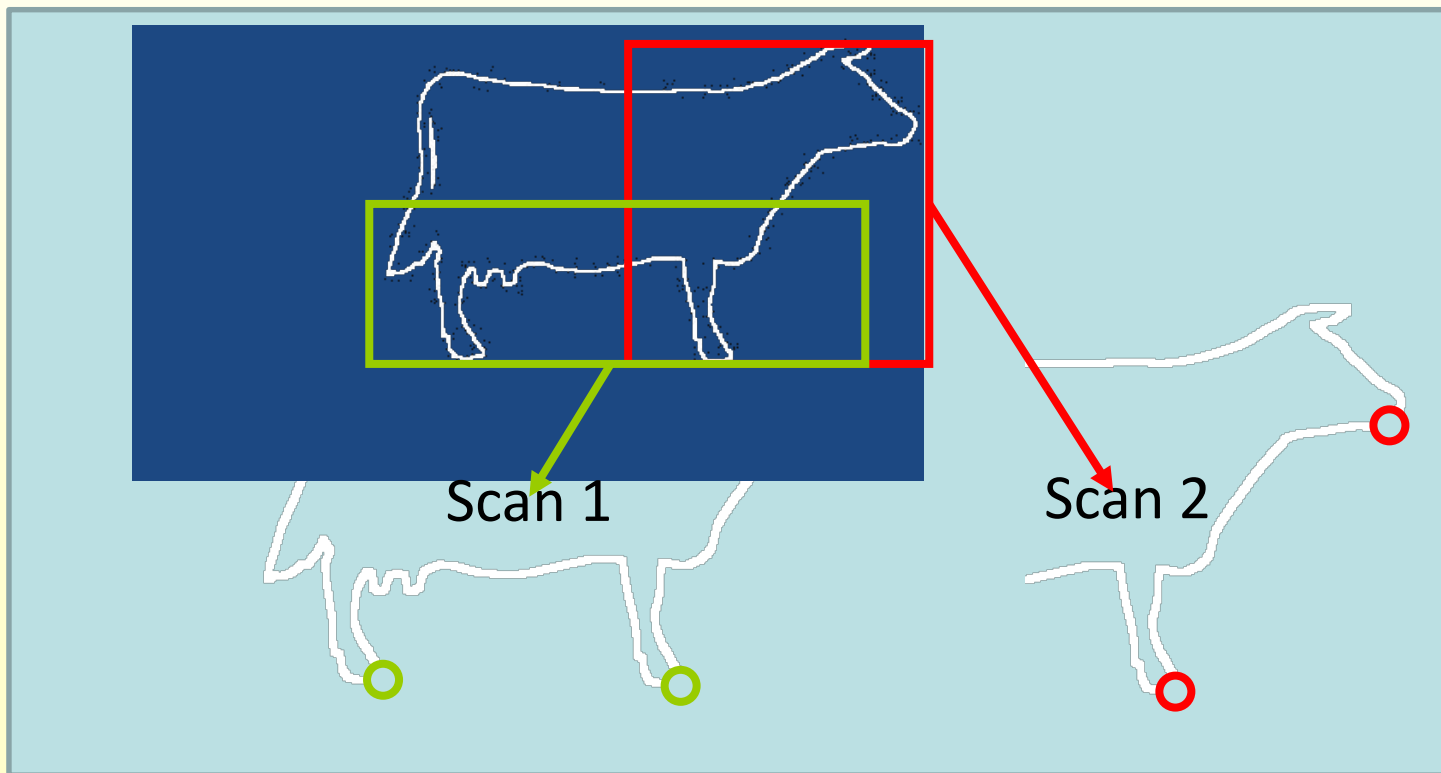
# From Global to Local

Given scans of a model:



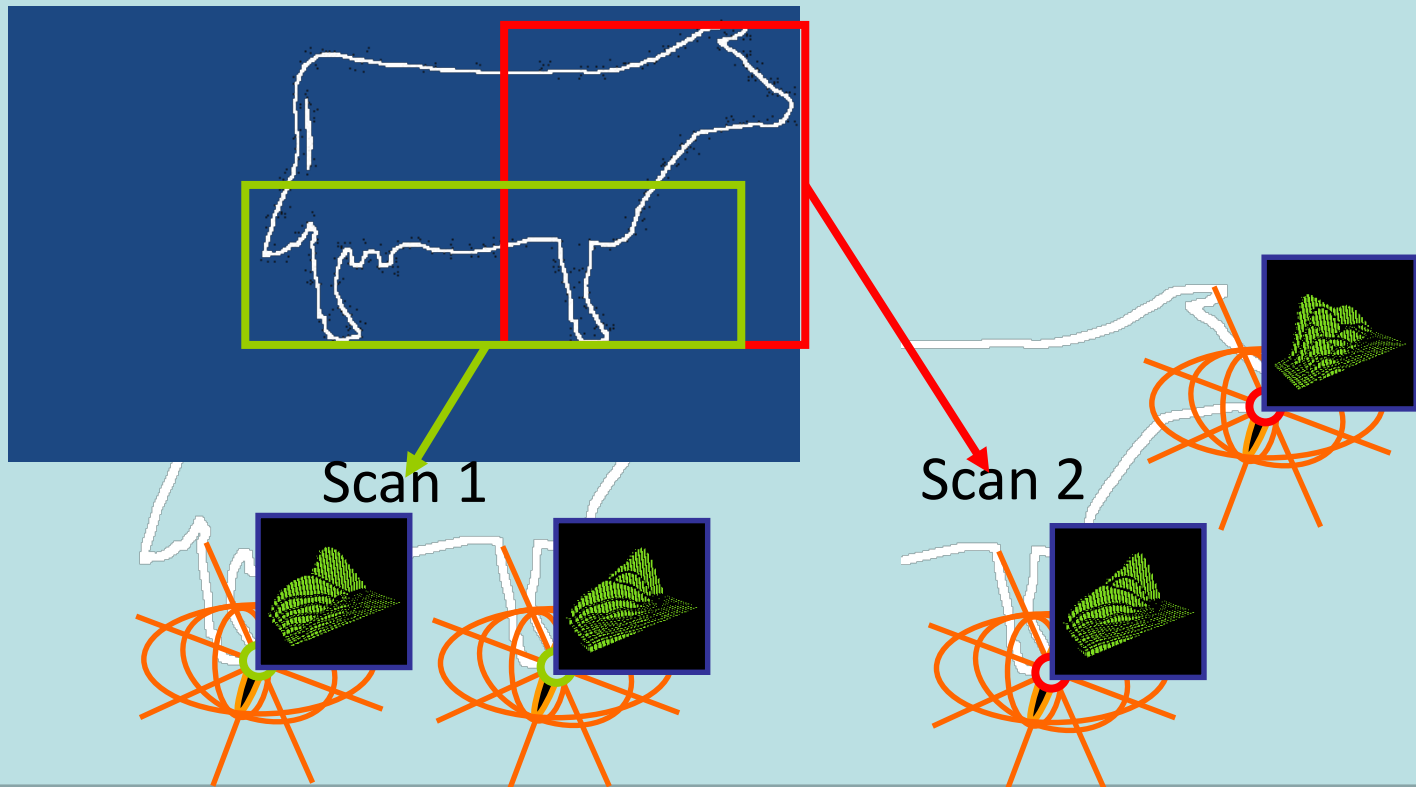
# From Global to Local

- ◆ Identify the feature points



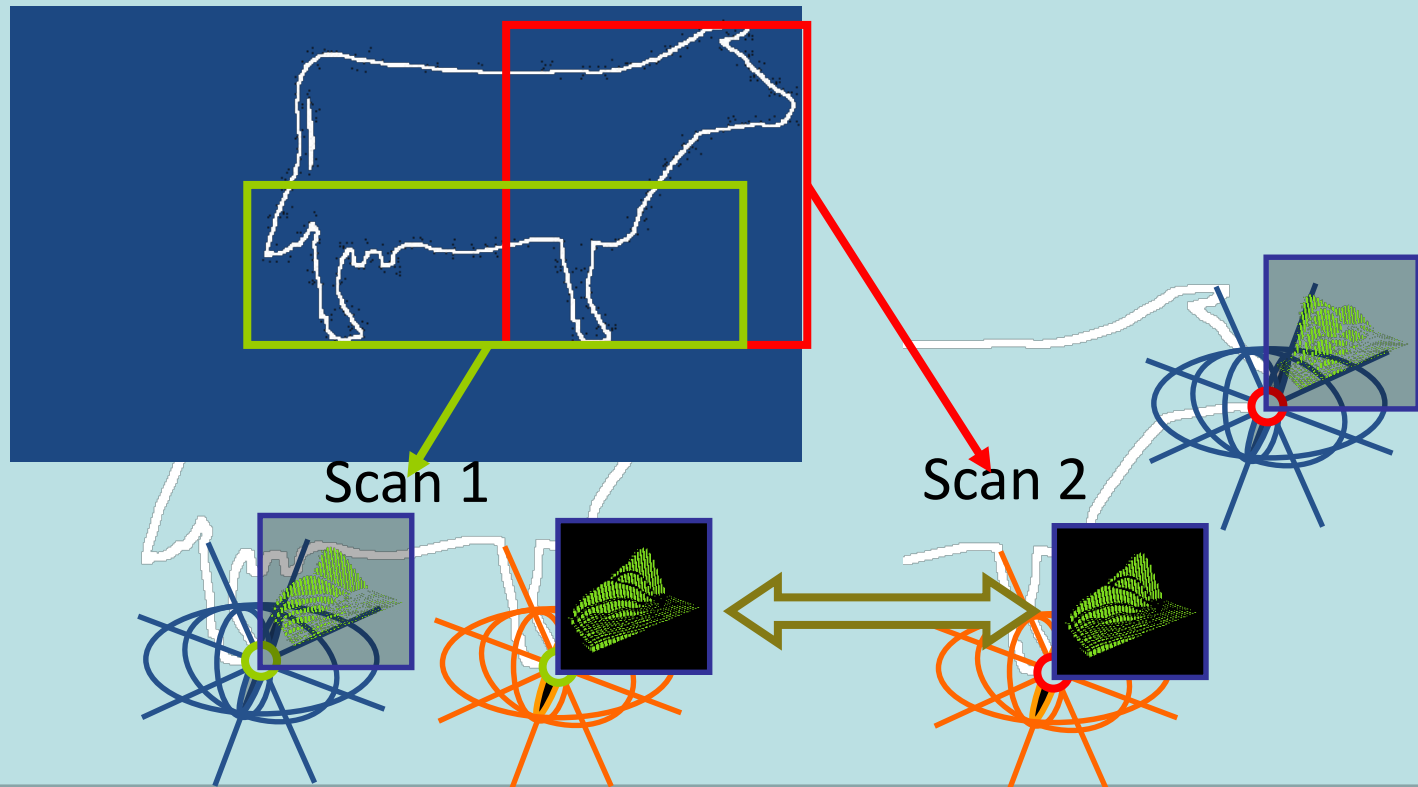
# From Global to Local

- ◆ Identify the features points
- ◆ Compute a local descriptor for each feature



# From Global to Local

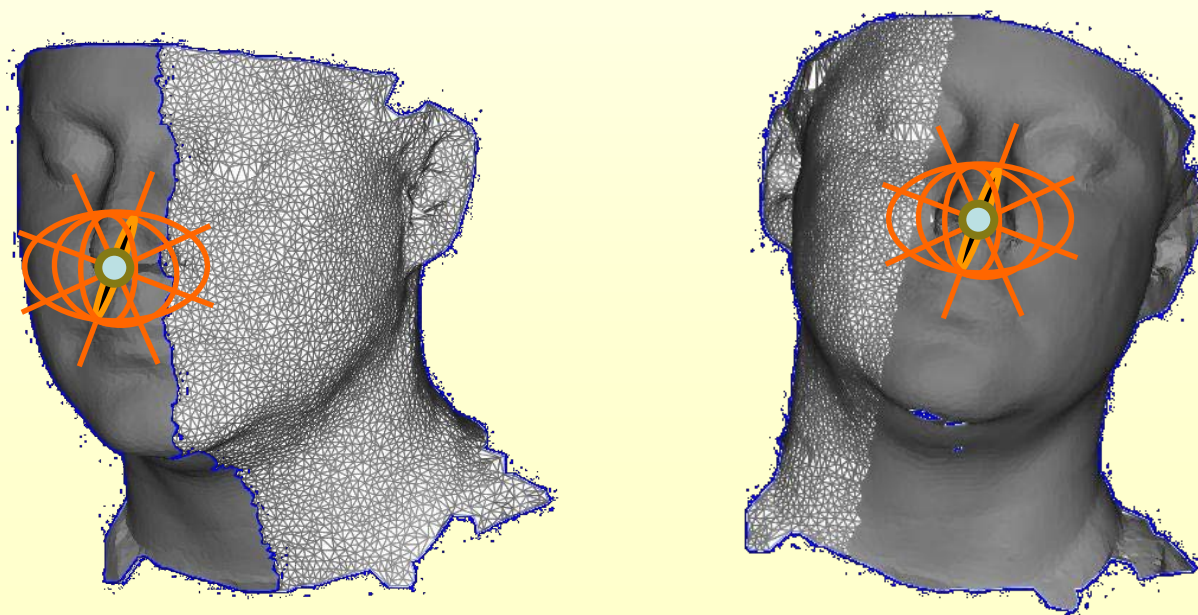
- ◆ Identify the features
- ◆ Compute a local descriptor for each feature
- ◆ For features correspond  $\rightarrow$  descriptors are similar



# Pose Normalization

## From Global to Local

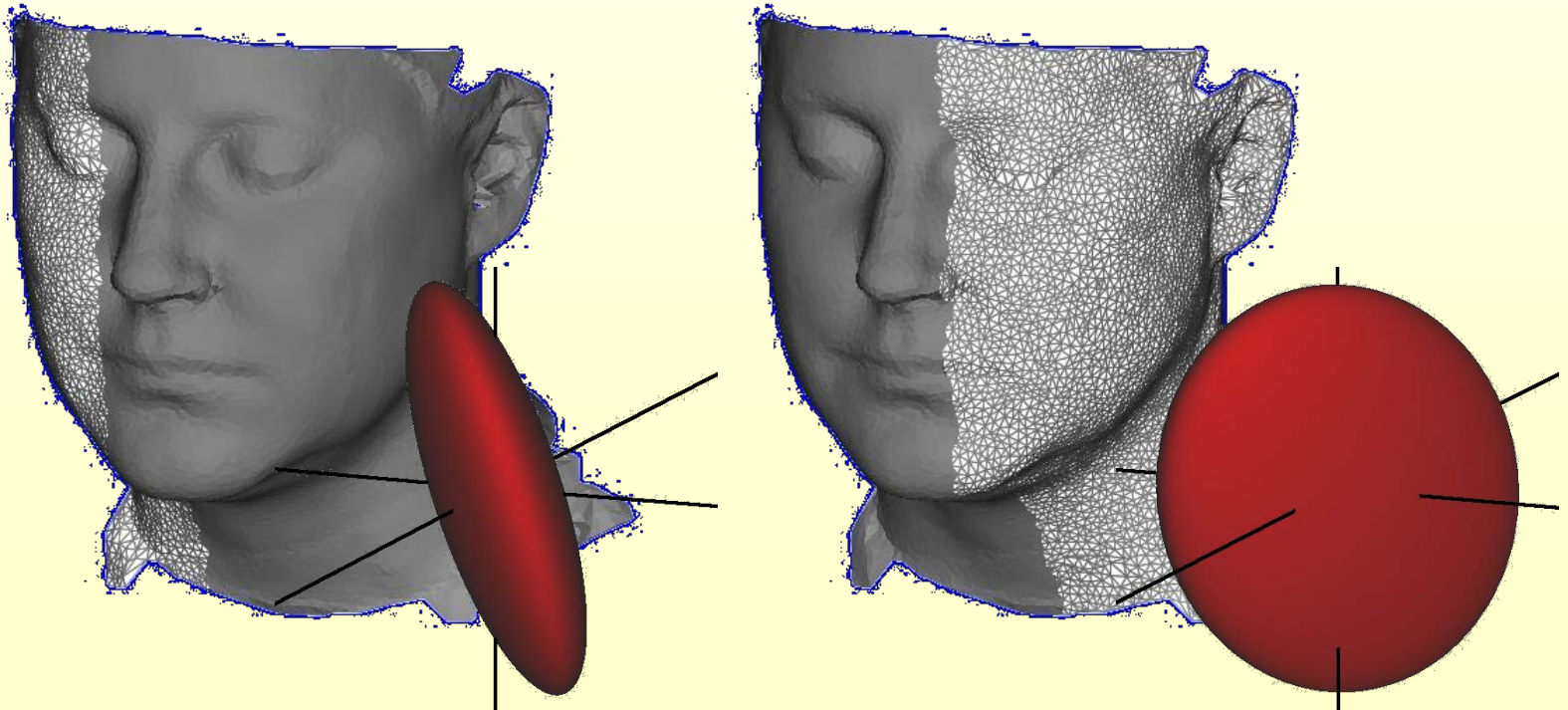
- ✓ Translation: Accounted for by centering the descriptor at the point of interest.
- ✗ Rotation: We still need to be able to match descriptors across different rotations.



# Pose Normalization

## Challenge

- ◆ Since only parts of the models are given, we cannot use global normalization to align the local descriptors



# Pose Normalization

## Challenge

- ◆ Since only parts of the models are given, we cannot use global normalization to align the local descriptors

## Solutions

- ◆ Normalize using local information

# Spin Images

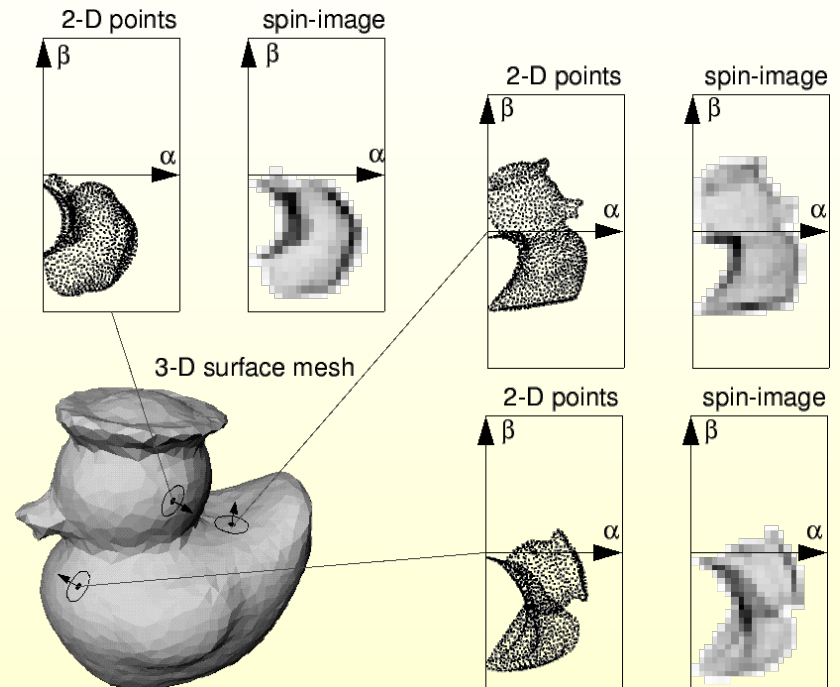
Creates an image associated with a neighborhood of a point.

Compare points by comparing their *spin images* (2D).

Given a point and a normal, every other point is indexed by two parameters:

$\beta$  distance to tangent plane

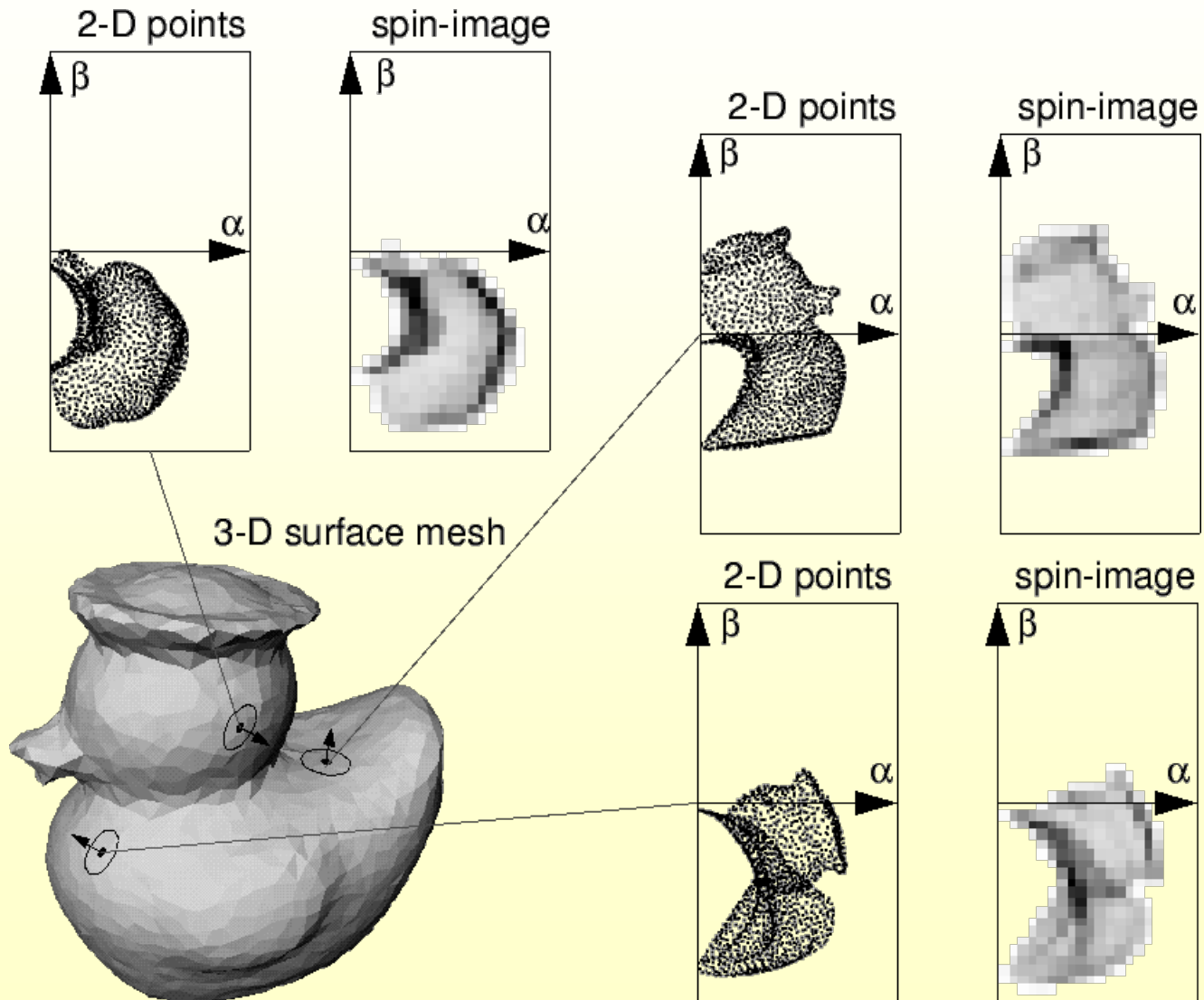
$\alpha$  distance to normal line



*Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes*<sub>101</sub>  
Johnson et al, PAMI 99

$\alpha$  - radial dist.  
 $\beta$  - elevation

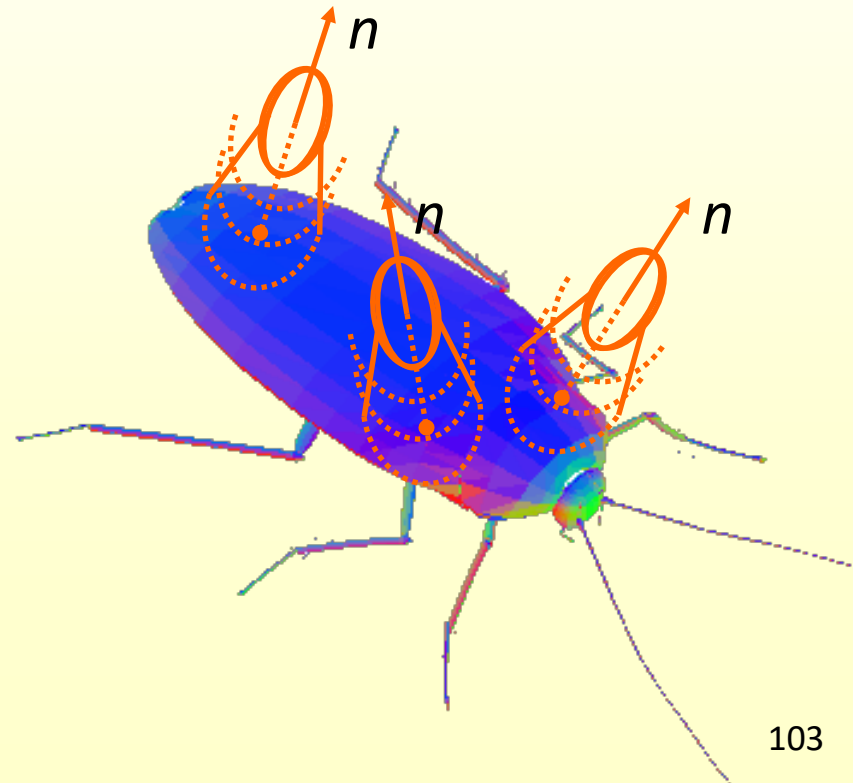
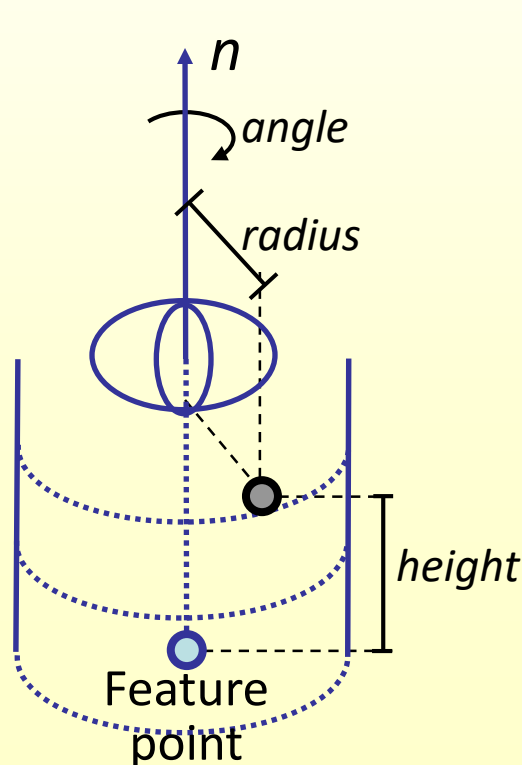
# Spin Images



# Local Descriptors: Examples

## Variations of Shape Histograms:

For each feature, represent its local geometry in cylindrical coordinates about the normal

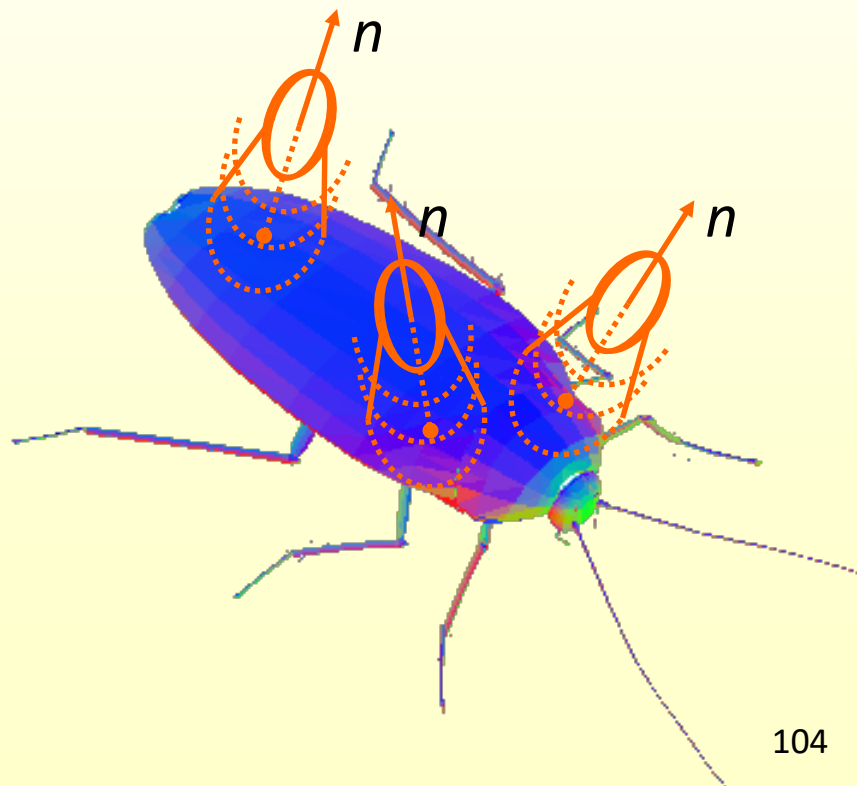


# Local Descriptors: Examples

## Variations of Shape Histograms:

For each feature, represent its local geometry in cylindrical coordinates about the normal

- ◆ **Spin Images:** Store energy in each normal ring

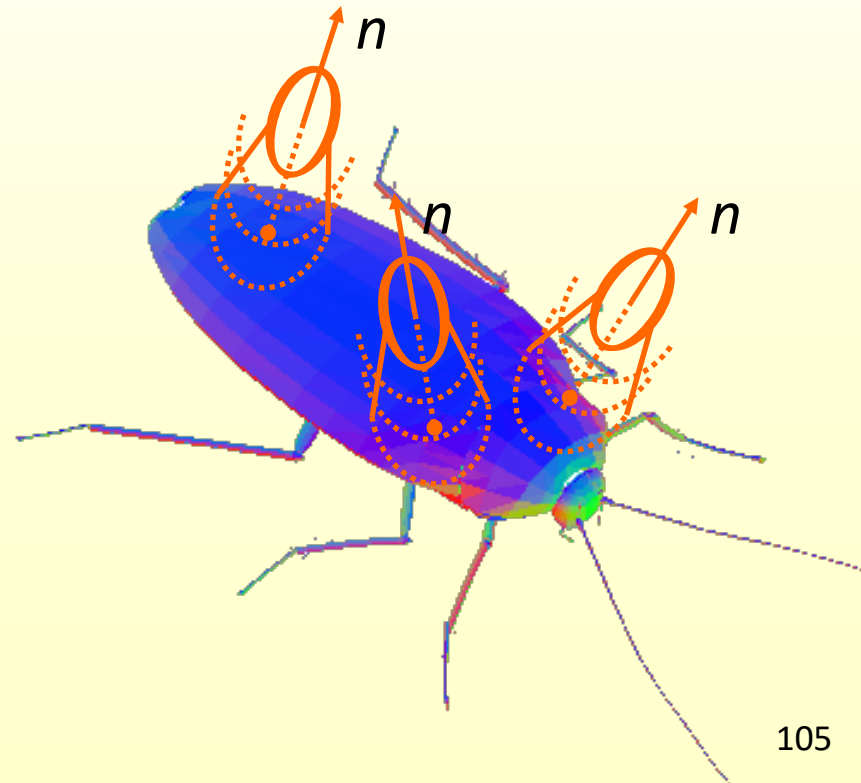


# Local Descriptors: Examples

## Variations of Shape Histograms:

For each feature, represent its local geometry in cylindrical coordinates about the normal

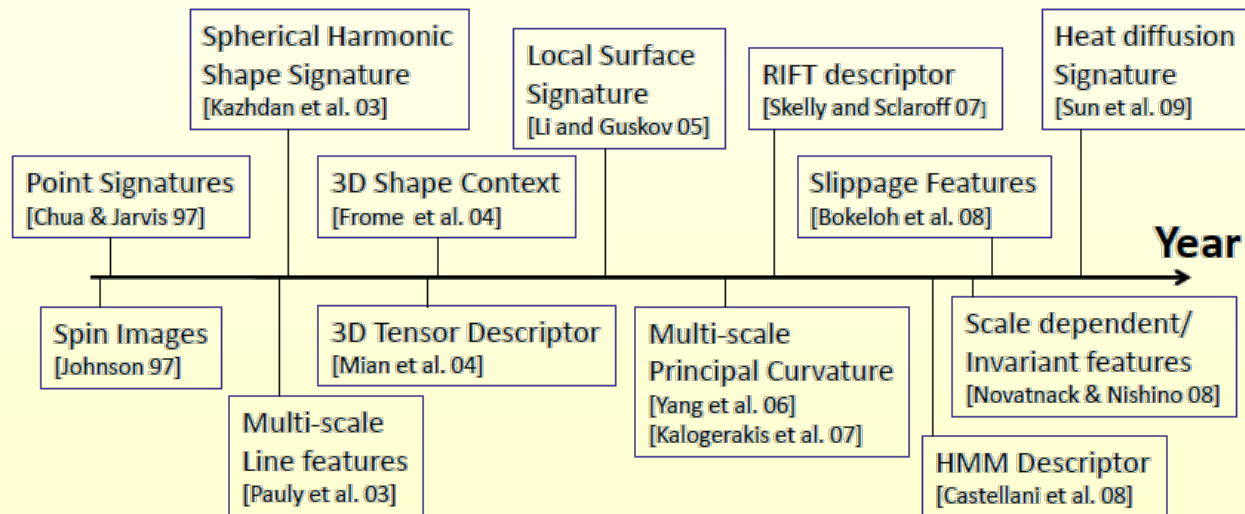
- ◆ **Spin Images:** Store energy in each normal ring
- ◆ **Harmonic Shape Contexts:** Store power spectrum of each normal ring



# Main Question

How to compare regions on the shape in an invariant manner?

A large variety of *descriptors* have been suggested.



# Intrinsic Descriptors Deformable Matching

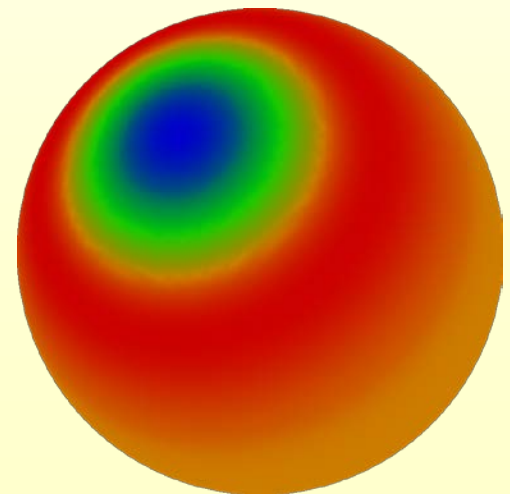
# Laplace-Beltrami Operator

- Analog of Fourier transform on the sphere, but now on a general 2D manifold
- LB is an operators that can be applied to functions on manifolds to yield other functions

$$\Delta : C^\infty(M) \rightarrow C^\infty(M), \Delta f = \operatorname{div} \nabla f$$

$$\operatorname{div} \nabla f$$

$$\frac{\partial f}{\partial t} = \Delta f = \operatorname{div} \nabla f.$$



# LB Eigen-decomposition

- ◆ The Laplace-Beltrami operator  $\Delta$  has an eigendecomposition

$$\Delta\phi_i = \lambda_i\phi_i$$

Invariant to isometric deformations!



$$\lambda_0 = 0$$

$$\lambda_1 = 2.6$$

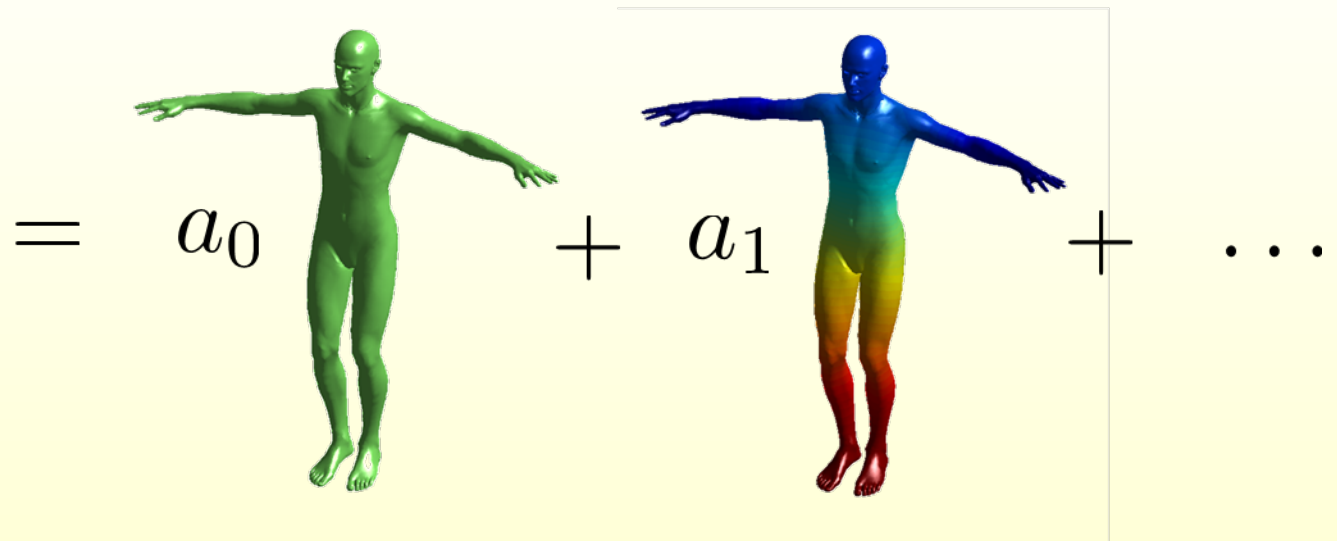
$$\lambda_2 = 3.4$$

$$\lambda_3 = 5.1$$

$$\lambda_4 = 7.6$$

# Multiscale Basis for a Function Space

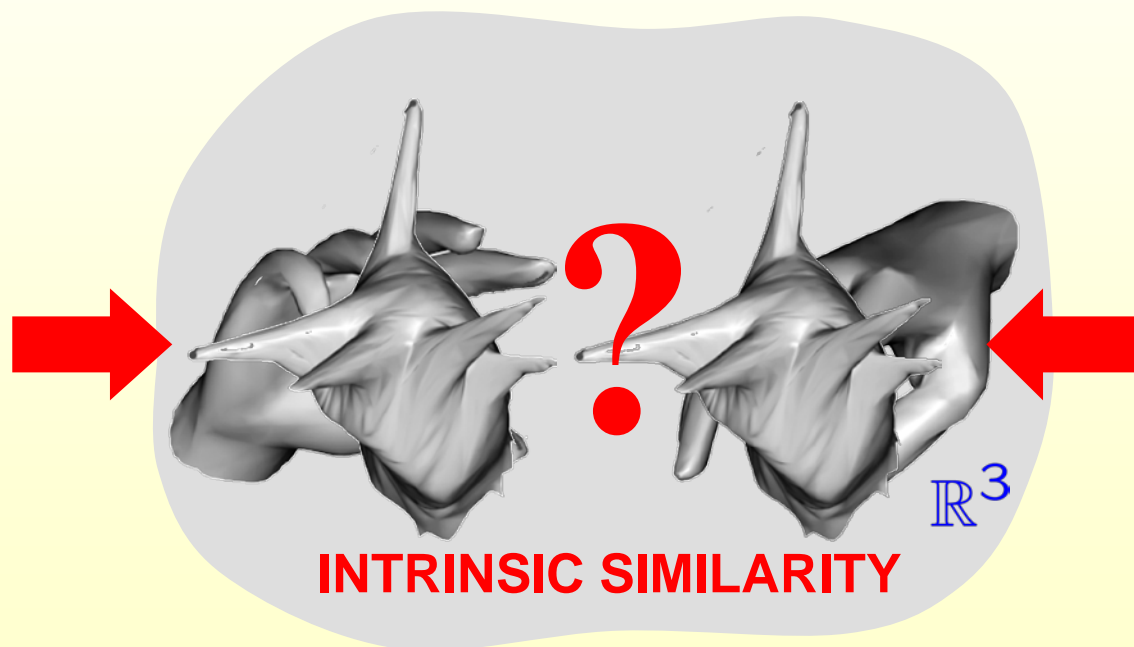
$$f : M \rightarrow \mathbb{R}$$



$$f = \sum_{i=0}^{\infty} a_i \phi_i \quad a_i = \int_M f(x) \phi_i(x) d\mu$$

# Global Point Signature

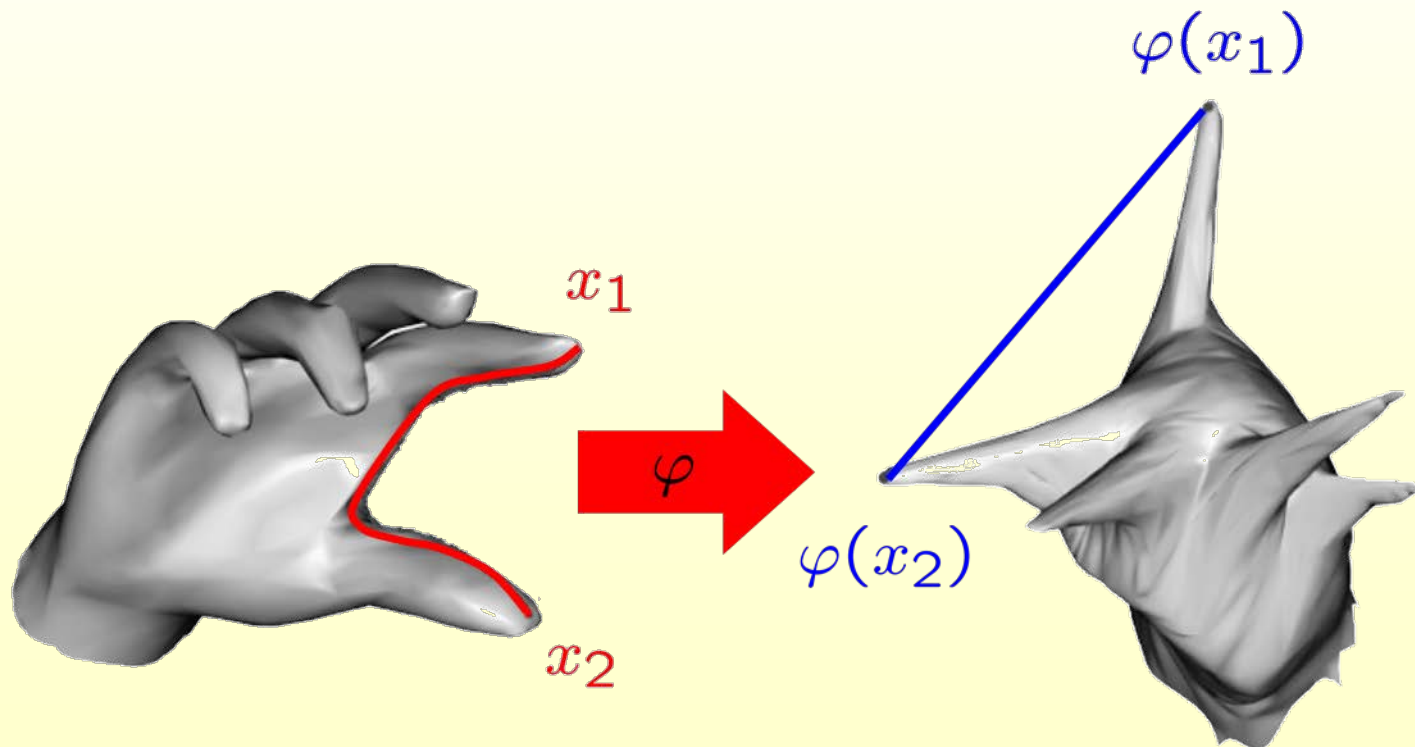
$$GPS(p) = \left( \frac{1}{\sqrt{\lambda_1}} \phi_1(p), \frac{1}{\sqrt{\lambda_2}} \phi_2(p), \frac{1}{\sqrt{\lambda_3}} \phi_3(p), \dots \right)$$



# Global Point Signature

almost invariant under isometries – but not completely canonical

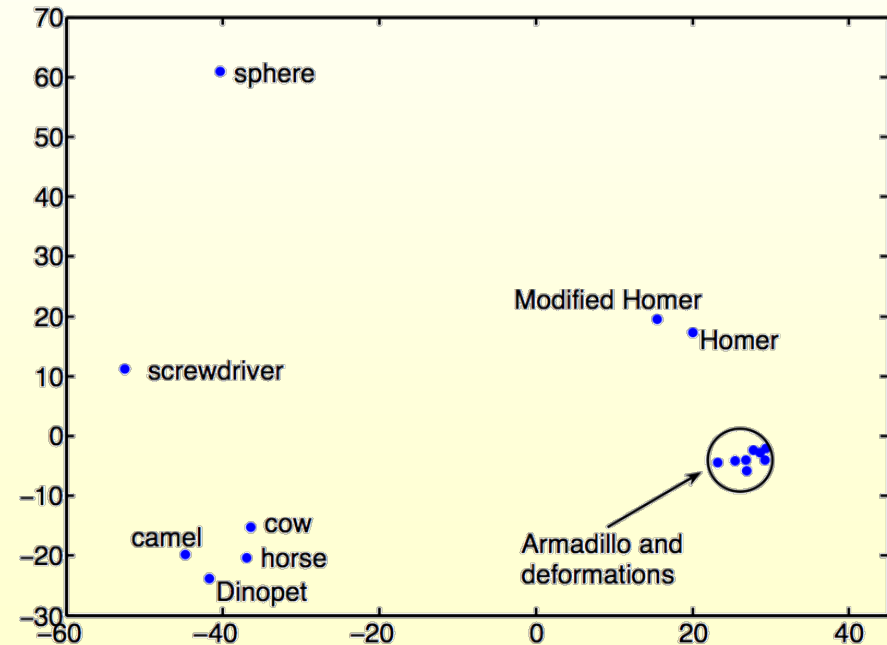
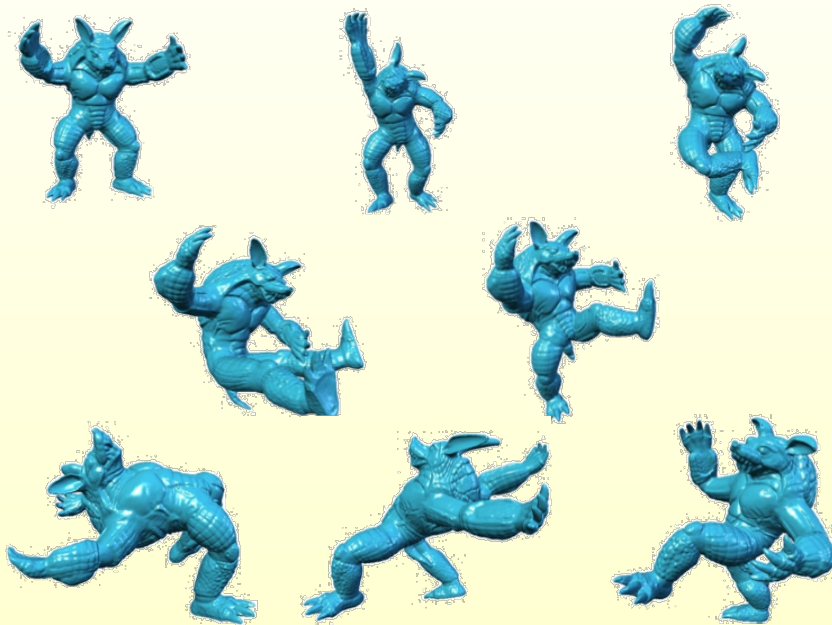
$$GPS(p) = \left( \frac{1}{\sqrt{\lambda_1}} \phi_1(p), \frac{1}{\sqrt{\lambda_2}} \phi_2(p), \frac{1}{\sqrt{\lambda_3}} \phi_3(p), \dots \right)$$



Diffusion distances are also intrinsic  
and also canonical

# Global Point Signature

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



**Figure 4:** *Armadillo and its deformations.*

Similar to D2, but use histograms in embedded space  
(rather than Euclidean)

# Global Point Signature

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

- ◆ Pros

- ◆ Isometry-invariant

- ◆ Global (each point feature depends on entire shape)

- ◆ Cons

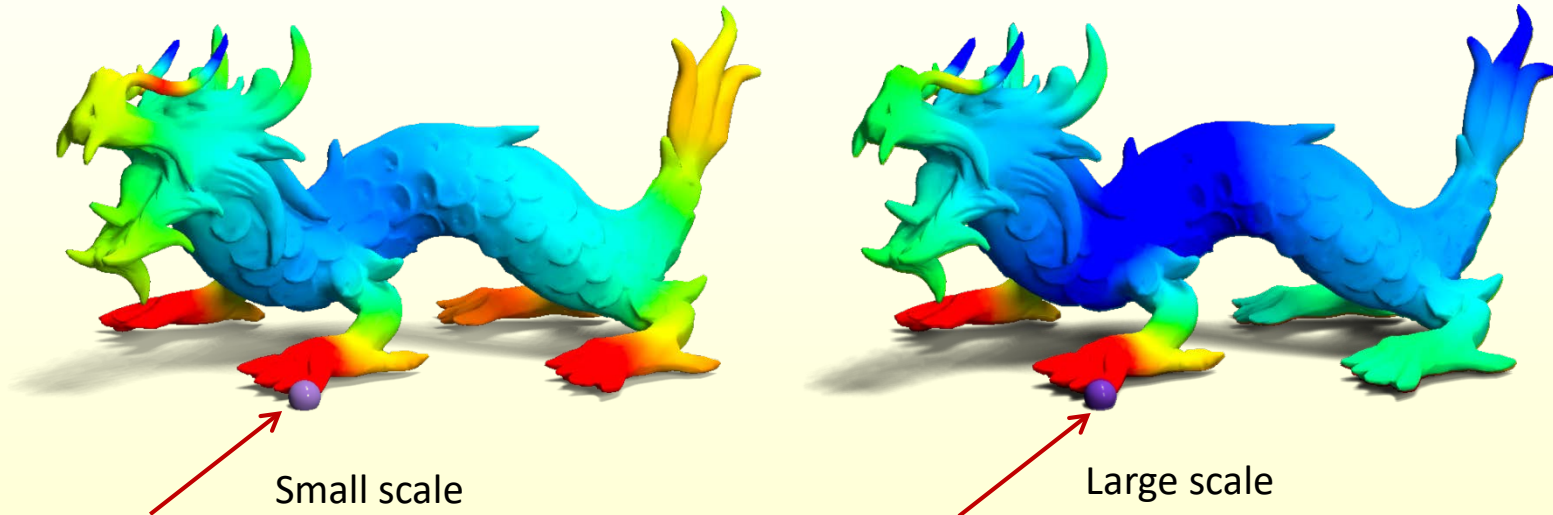
- ◆ Eigenfunctions may flip sign

- ◆ Eigenfunctions might change positions due to deformations

- ◆ Only global

# The Issue of Scale

- Given a point (●) on a shape, find other points with “similar” neighborhoods



- Inherently multiscale question: on a manifold, locally all points are the same. Need a meaningful way to compare point neighborhoods at different scales
- At what scale do neighborhoods become unique?

# Back to Heat Diffusion

- Heat diffusion on a Riemannian manifold:

If  $u(x, t)$  is the amount of heat at point  $x$  at time  $t$ ,  
then

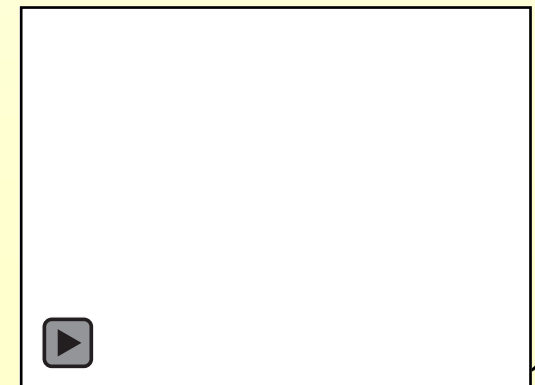
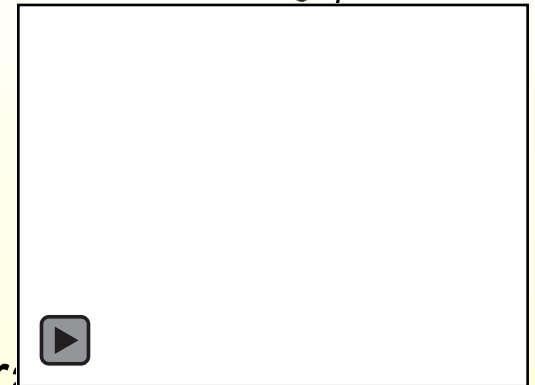
$$\frac{\partial u}{\partial t} = \Delta u$$

$\Delta$  : Laplace-Beltrami Operator (div grad)

- Given an initial distribution  $f(x)$ . After time  $t$ :

$$f(x, t) = e^{-t\Delta} f$$

$H_t$  heat operator

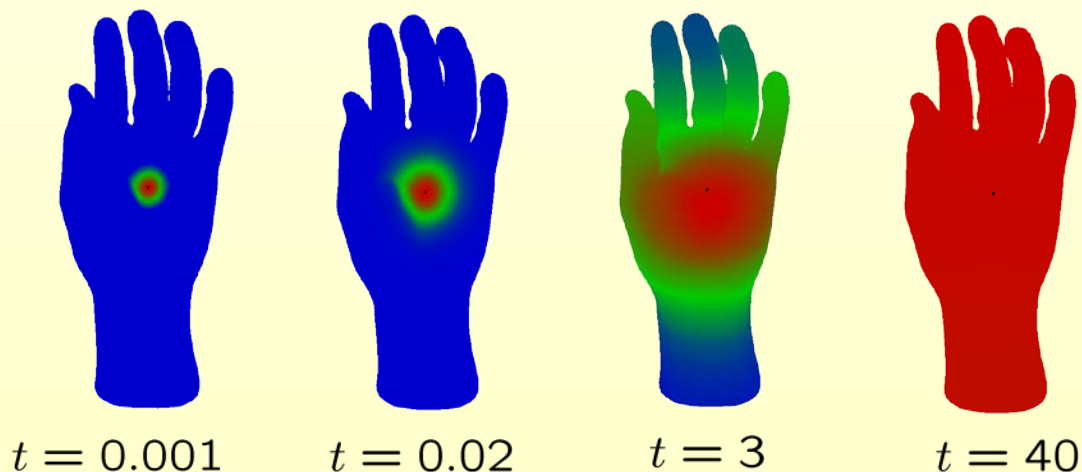


# The Heat Kernel

- Heat kernel  $k_t(x, y)$

$$f(x, t) = \int_{\mathcal{M}} k_t(x, y) f(y) dy$$

$k_t(x, y)$ : amount of heat transferred from  $x$  to  $y$  in time  $t$ .  
How well  $x$  and  $y$  are connected at scale  $t$



# Background

- Heat Kernel  $k_t(x, y)$ . Also the probability density function of Brownian motion on  $\mathcal{M}$ :

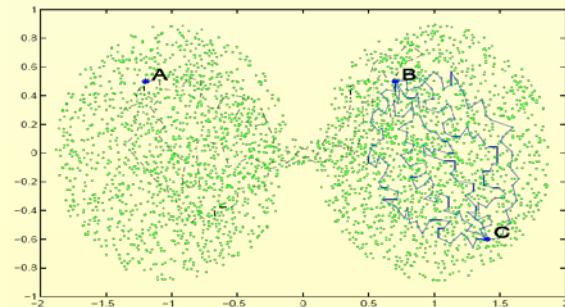
$$\mathbb{P} \left( W_x^t \in C \right) = \int_C k_t(x, y) dy$$

- Intuitively: weighted average over all paths possible between  $x$  and  $y$  in time  $t$

- Related to **Diffusion Distance**:

$$D_t(x, y) = k_t(x, x) - 2k_t(x, y) + k_t(y, y)$$

a robust multi-scale measure  
of proximity



# Heat Kernel Properties

## Basic Properties

- $k_t(x, y) = k_t(y, x)$
- $k_{t+s}(x, y) = \int_M k_t(x, z)k_s(z, y)dz$
- $k_t(x, y) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i(x)\phi_i(y)$   
Eigenfunctions of LB

# Heat Kernel Properties

- Invariant under isometric deformations

If  $T : X \rightarrow Y$  is an isometry, then:

$$k_t(X, Y) = k_t(T(x), T(y))$$

- Conversely: it characterizes the shape up to isometry.

If  $k_t(X, Y) = k_t(T(x), T(y)) \quad \forall x, y, t$  then:

$T$  is an isometry.

This is because:

$$\lim_{t \downarrow 0} (t \log k_t(x, y)) = -\frac{1}{4} d_{\mathcal{M}}^2(x, y) \quad \forall x, y$$

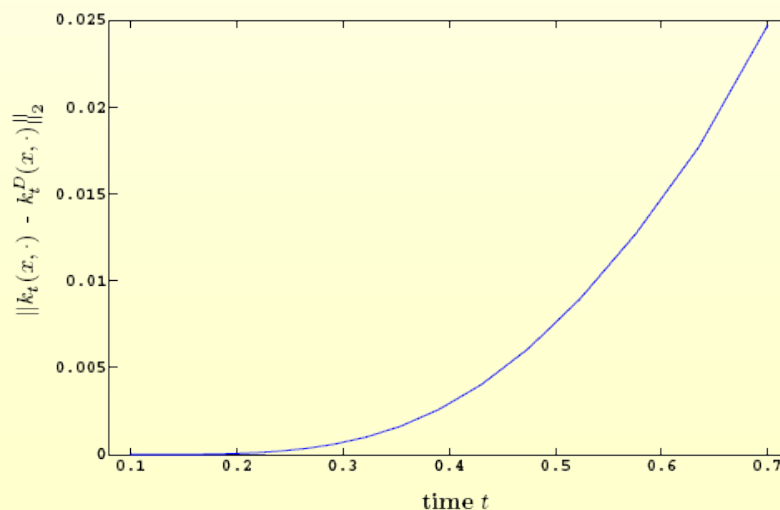
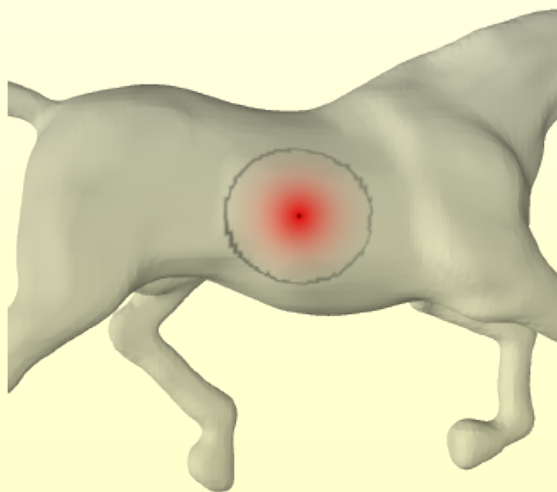
where  $d_{\mathcal{M}}(\cdot, \cdot)$  is the geodesic distance

# Heat Kernel Properties

## • Multiscale:

For a fixed  $x$ , as  $t$  increases, heat diffuses to larger and larger neighborhoods

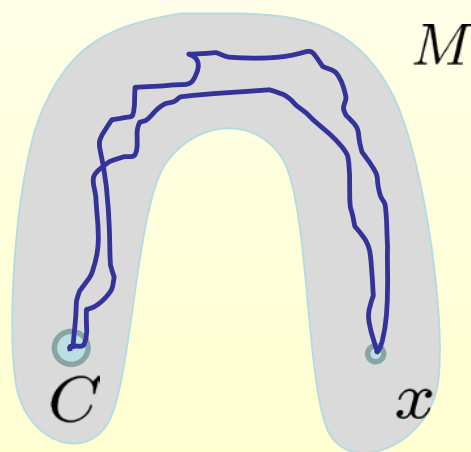
Therefore,  $k_t(x, \cdot)$  is determined by (reflects the properties of) a neighborhood that grows with  $t$



# Heat Kernel Properties

## ● Robustness:

$k_t(x, \cdot)$  is the probability density function of BM, a weighted average over all paths, which is generally not very sensitive to local perturbations

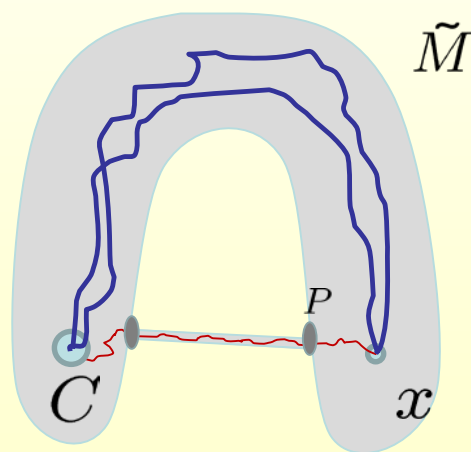


$$k_t^M(x, C) = \mathbb{P}(W_x^t \in C)$$

# Heat Kernel Properties

## Robustness:

$k_t(x, \cdot)$  is the probability density function of BM, a weighted average over all paths, which is generally not very sensitive to local perturbations

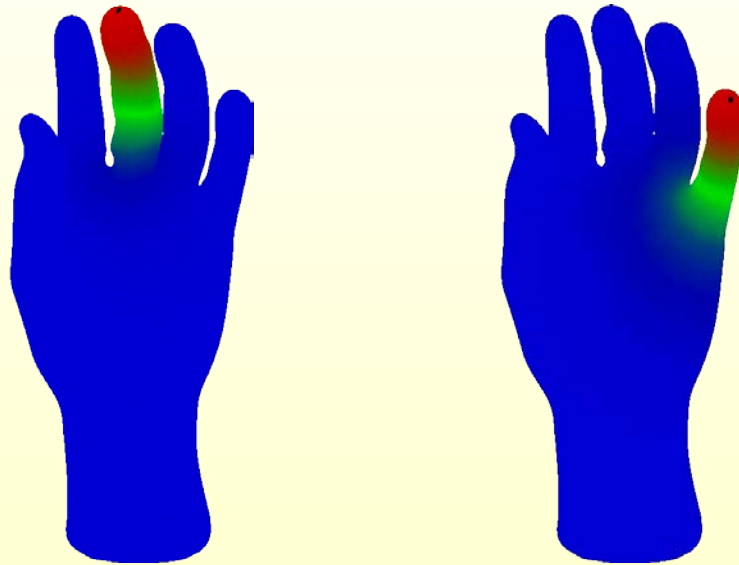


$$k_t^{\tilde{M}}(x, C) = \mathbb{P}(\tilde{W}_x^t \in C)$$

Only paths through the modified area  $P$  will change

# Defining a Signature

- Let  $k_t(x, \cdot)$  be the signature of  $x$  at scale  $t$   
The heat kernel has all the properties we want  
Except easy comparison ...



- $k_t(x, \cdot)$  is a function on the entire manifold
- Nontrivial to align the domains of such functions across different shapes, or even for different points of the same shape

# The Heat Kernel Signature

- Let  $k_t(x, \cdot)$  be the signature of  $x$  at scale  $t$   
The heat kernel has all the properties we want.  
Except easy comparison ...

- We define the **Heat Kernel Signature** (HKS), by restricting to the diagonal:

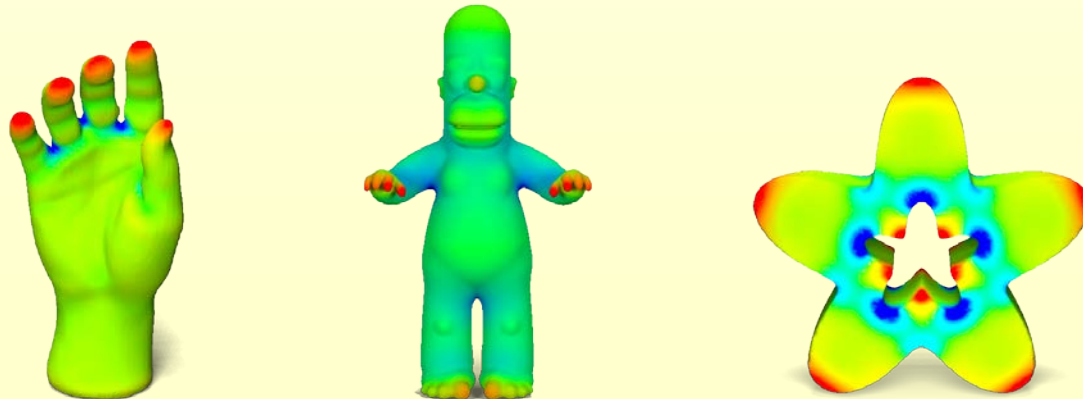
$$\text{HKS}(x) = \{k_t(x, x), t \in \mathbb{R}^+\}$$

- Now HKSs of two points can be easily compared since they are defined on a common domain (time)

# Defining a Signature

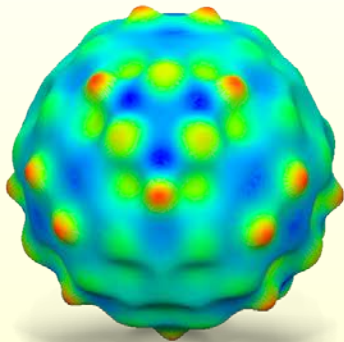
- Since HKS is a restriction of the heat kernel, it is:
  - Robust
  - Multiscale
- Question: How informative is it?
  - Related to Gaussian curvature for small  $t$  :

$$k_t(x, x) = \frac{1}{4\pi t} \sum_{i=0}^{\infty} a_i t^i \quad a_0 = 1, a_1 = \frac{1}{6}K$$

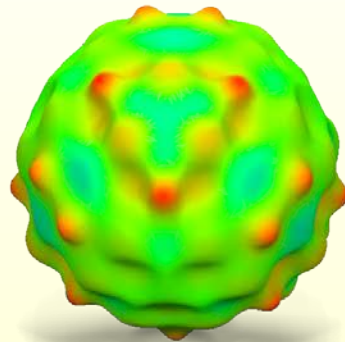


# Defining a Signature

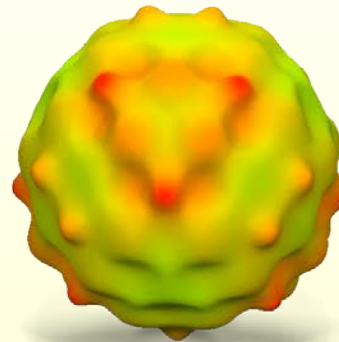
- HKS can be interpreted as a multiscale, robust, intrinsic curvature:



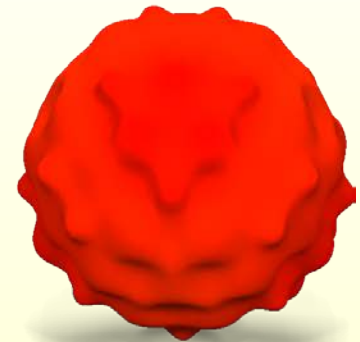
$t = 0.004$



$t = 0.008$



$t = 0.02$



$t = 2$

# Informative Theorem

- The set of all HKSs on a shape almost always defines it up to isometry!
- **Theorem:** If  $X$  and  $Y$  are two compact manifolds, such that  $\Delta_X$  and  $\Delta_Y$  have only non-repeating eigenvalues, then a homeomorphism  $T : X \rightarrow Y$  is an isometry **if and only if**, for all  $x$

$$\text{HKS}(x) = \text{HKS}(T(x))$$

- The set of all HKSs characterizes the intrinsic structure of the manifold

# Informative Theorem

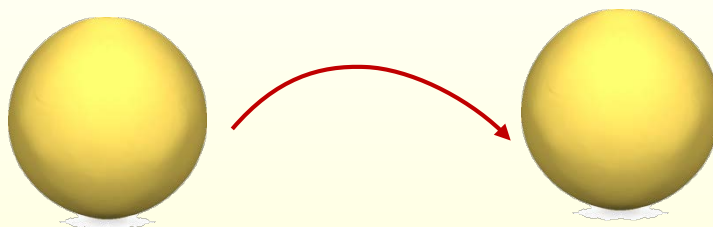
- Heat kernel is related to the eigenvalues and eigenfunctions of the LB-operator:

$$\text{HKS}(x, t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i^2(x)$$

- Invariant to rotations within the eigenspace

# Informative Theorem

- How general is the theorem?
  - If there are repeated eigenvalues, it does not hold:



On the sphere,  $\text{HKS}(x) = \text{HKS}(y) \forall x, y$  but there are non-isometric maps between spheres.

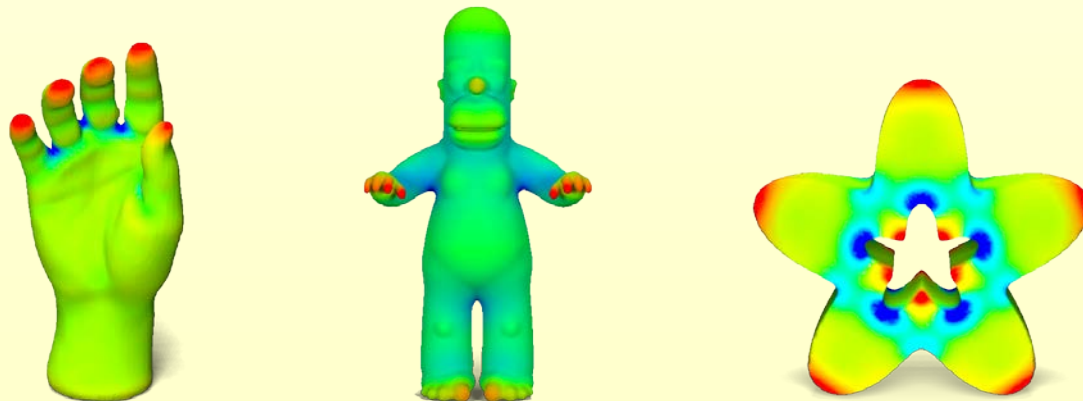
- Uhlenbeck's Theorem (1976): for “almost any” metric on a 2-manifold  $X$ , the eigenvalues of  $\Delta_X$  are non-repeating

# Informative Property

- Conclusion:
  - HKS is informative for individual points
  - And, as a set, for the entire shape

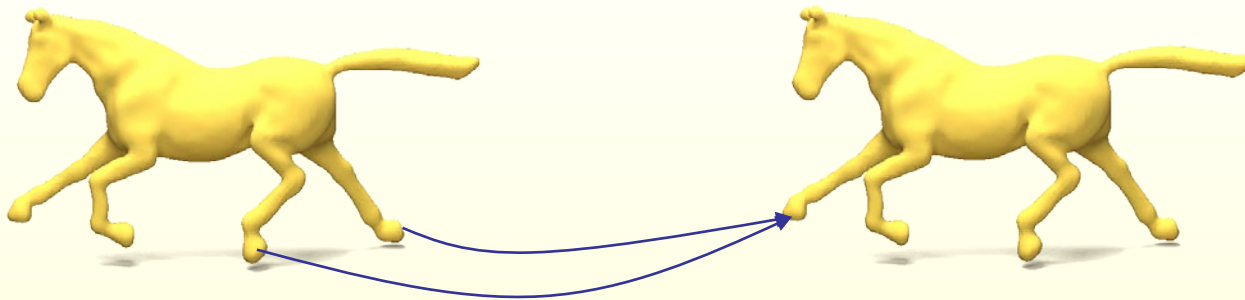
Can be used both for multiscale point matching  
and for shape comparison

$$\text{HKS}(x) = \{k_t(x, x), t \in \mathbb{R}^+\}$$



# Applications

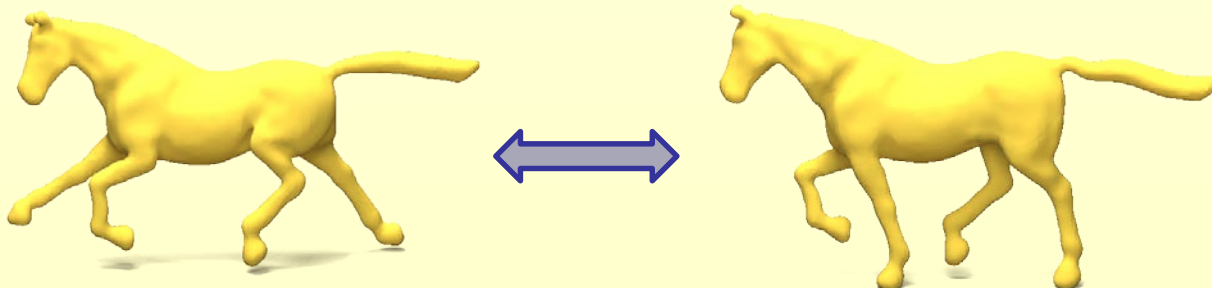
- Multi-scale matching with HKS, structure discovery



- Shape comparison

- Shape retrieval using HKS

- Spectral version of Gromov-Hausdorff



# Multiscale Matching

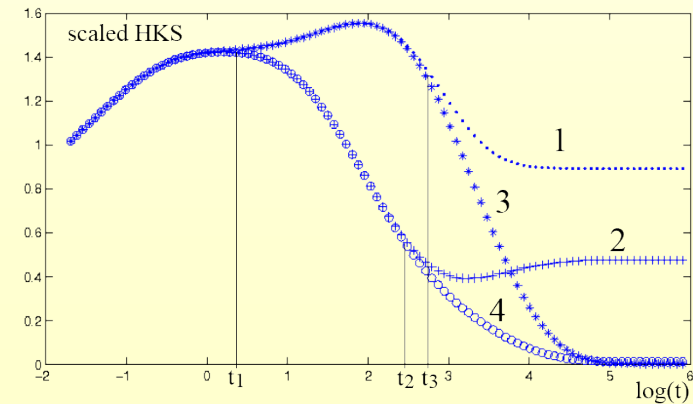
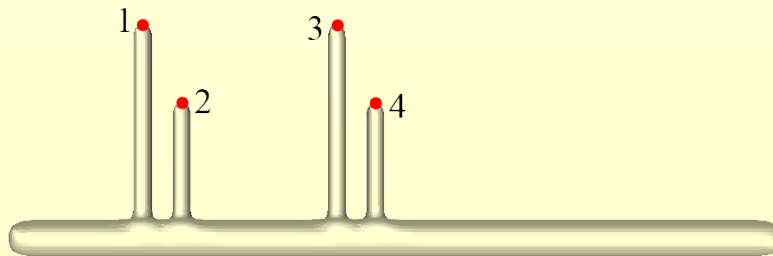
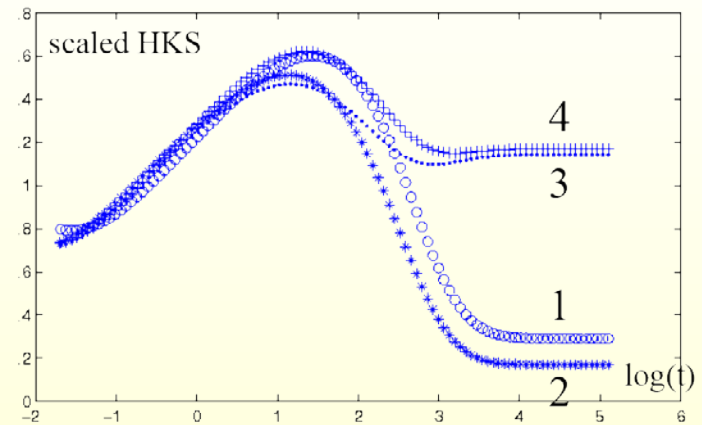
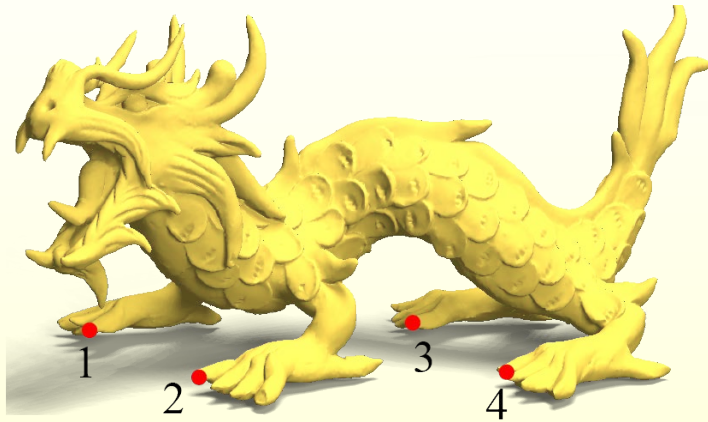
- Two heuristics for making HKSs commensurable:
  - For a fixed point  $x$ , sample HKS on a logarithmic scale at times  $t_i$
  - For a fixed time  $t$  scale each HKS, by the sum over all points of  $M$

$$\text{HKS}(x) = \left\{ \frac{k_{t_i}(x, x)}{\sum_j e^{-t_i \lambda_j}}, i \in 1, 2, \dots, 100 \right\}$$
$$t_i = \alpha^i t_0$$

- Compare using L2 norm of the HKS vectors.

# Multiscale Matching

- Comparing points through their HKS signatures:

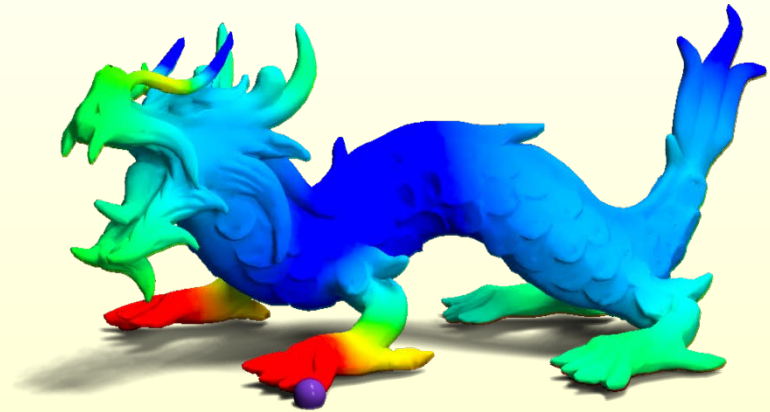


# Multiscale Matching

- Comparing points through their HKS signatures:



Medium scale



Full scale

# Multiscale Matching

- Finding similar points – robustly:



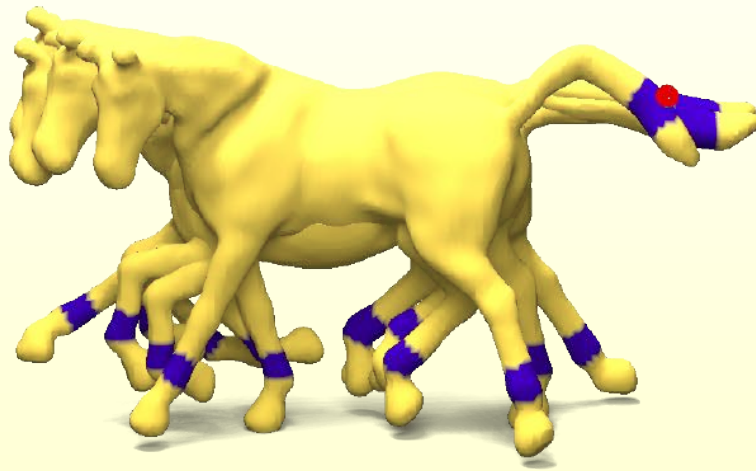
Medium scale



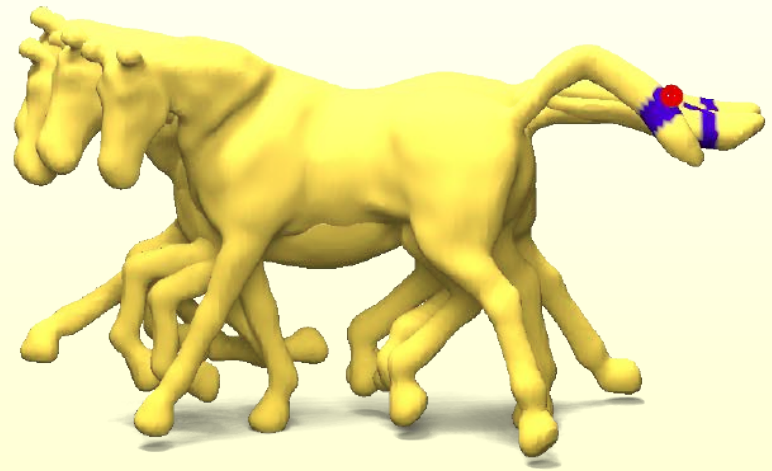
Full scale

# Multiscale Matching

- Finding similar points across multiple shapes:



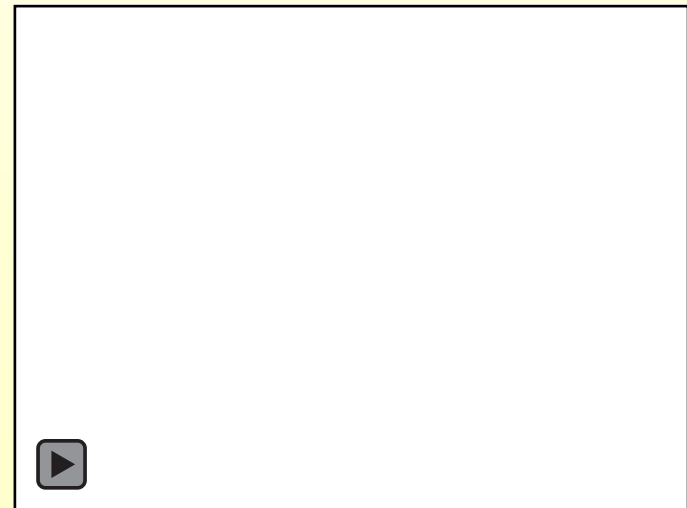
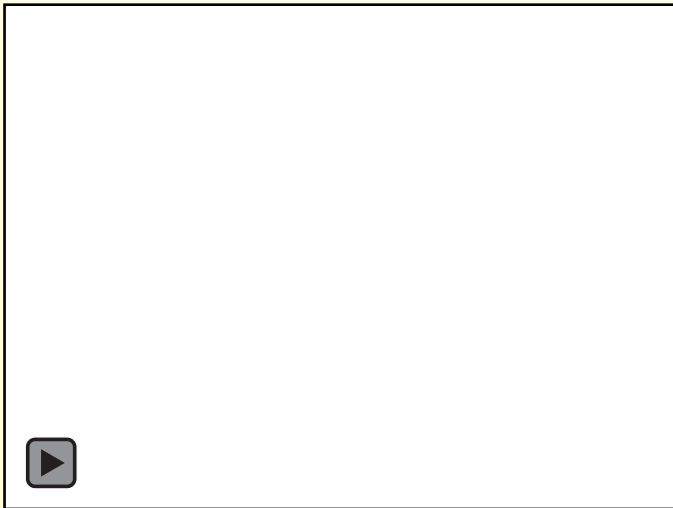
Medium scale



Full scale

# Feature Detection

- **Persistent feature detection:**
  - Intuition: heat diffuses slower at points with high curvature. Heat will tend to concentrate in “hot spots” – extremities of the surface
  - Approach: track the local maximum of the heat kernel for increasing  $t$

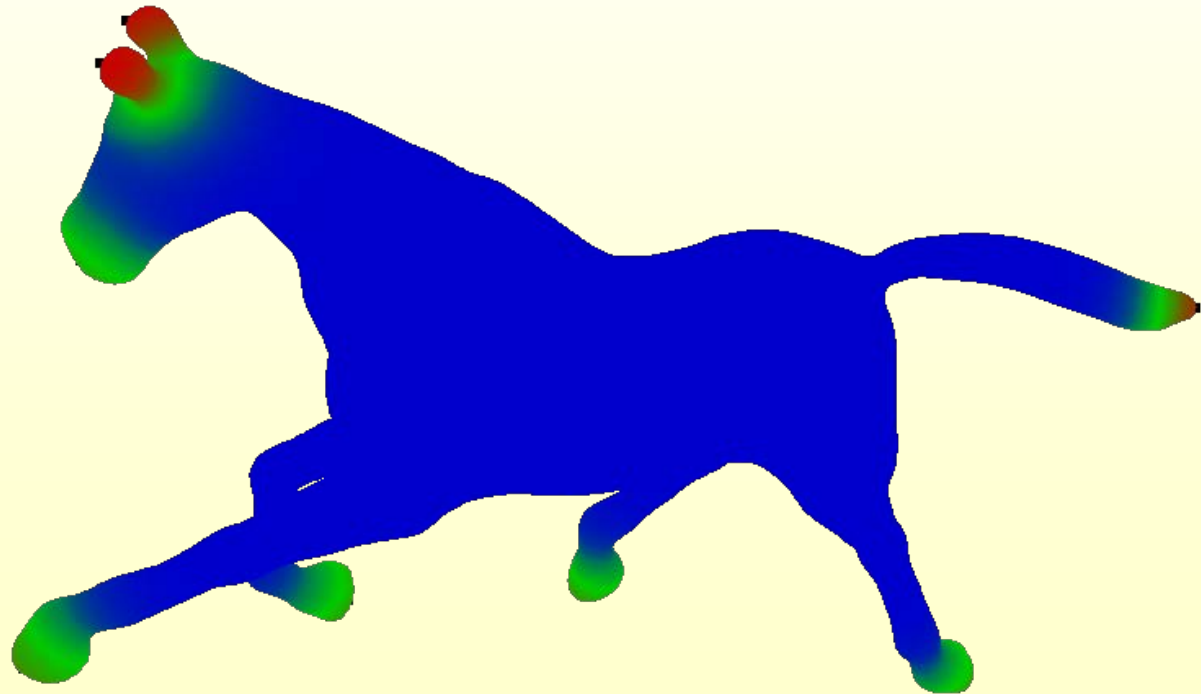


# Feature Detection

- Persistent feature detection:

- Find points that are long term maxima of their heat kernels:

$$k_t(x, \cdot)$$

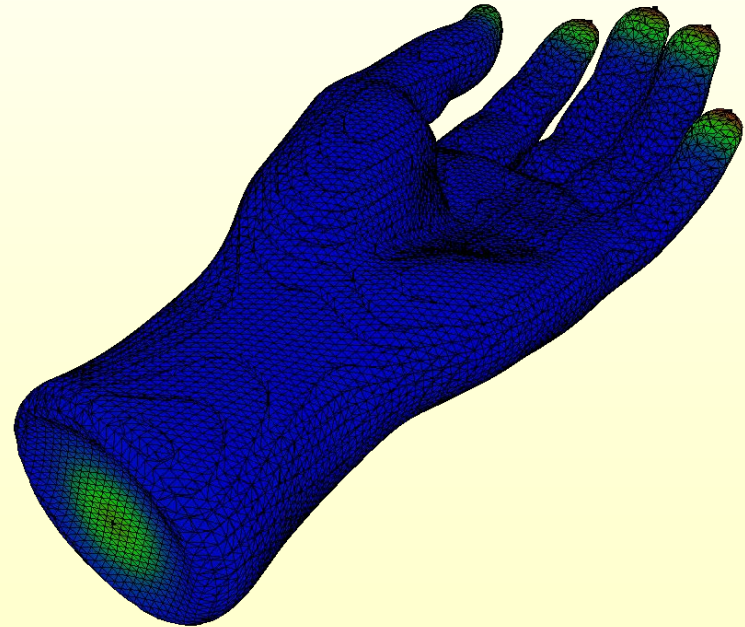
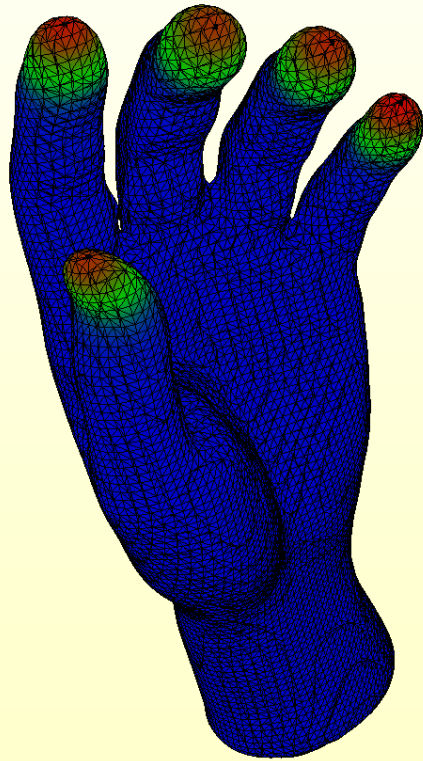


# Feature Detection

- Persistent feature detection:

- Find points that are long term maxima of their heat kernels:

$$k_t(x, \cdot)$$



# Feature Detection

- **Persistent feature detection:**

- Find points that are long term maxima of their heat kernels:

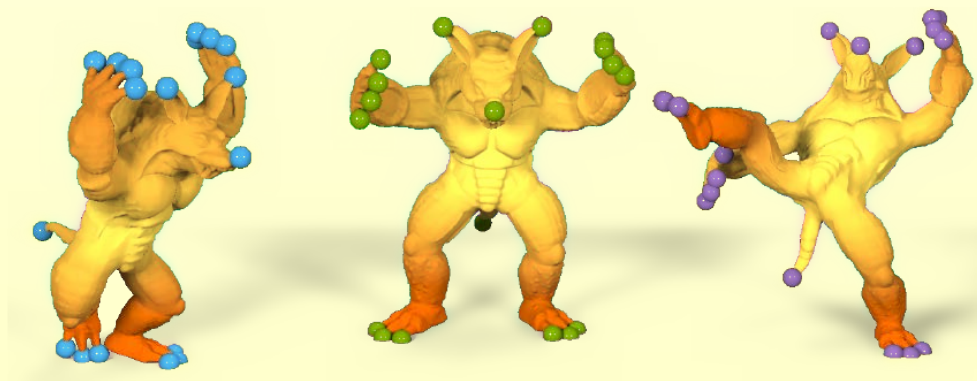
$$k_t(x, \cdot)$$

- This may be expensive since the heat kernel at every point is a function over the whole shape. However, long term behavior at nearby points is similar due to mixing

- Approximation: find points that are local maxima of

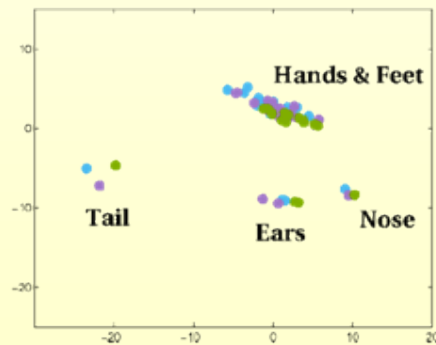
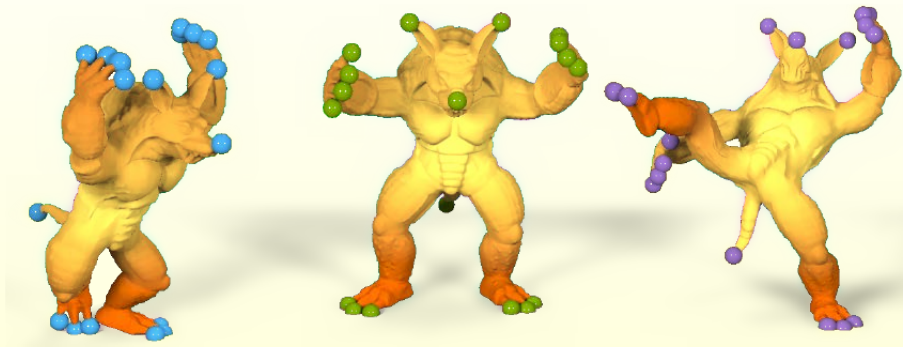
$$k_t(x, x)$$

for large enough  $t$

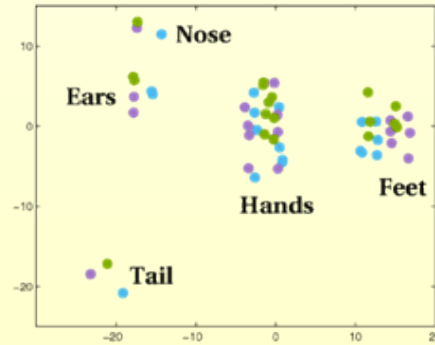


# Shared Structure

- 2D MDS embedding of feature points on three shapes according to distances of their HKS



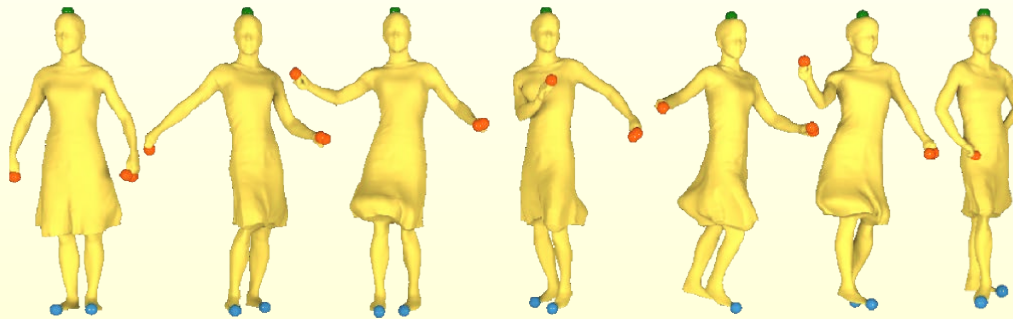
$t = [0.1, 4]$



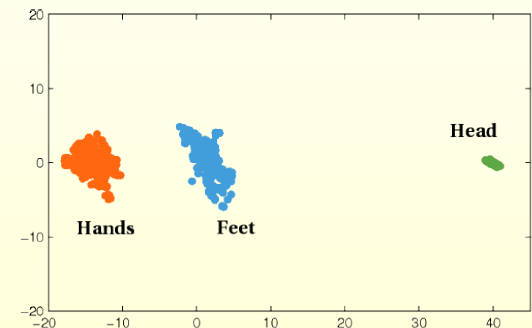
$t = [0.1, 80]$

# Shared Structure

- 2D MDS embedding of feature points on **175 shapes** according to distances of their HKS.

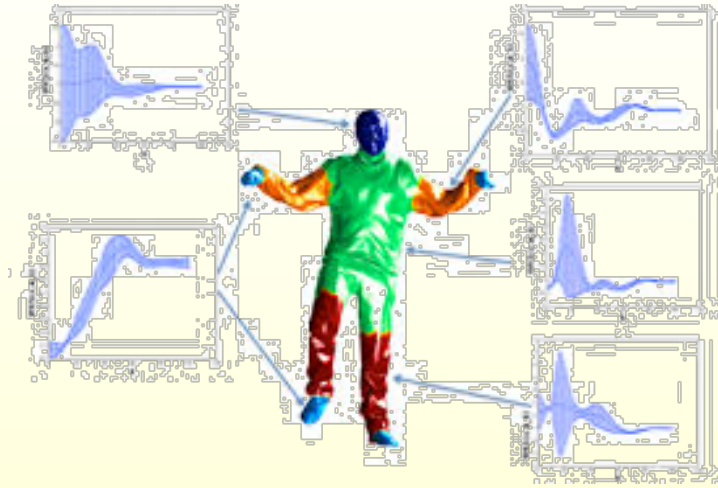
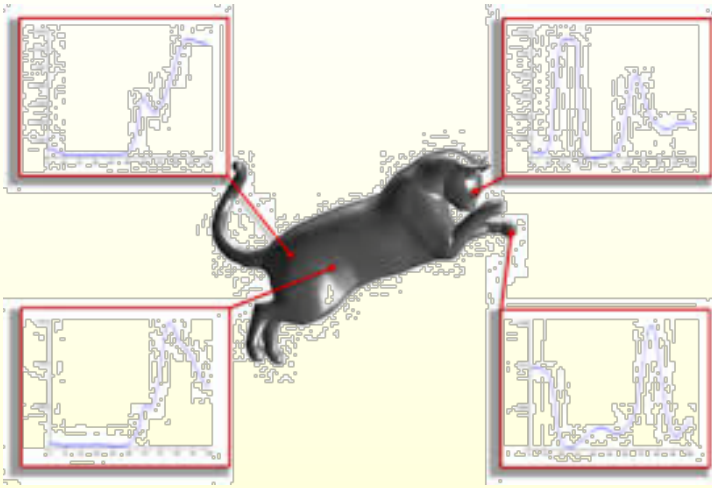


Feature points found on a few poses of the dancer model by Vlasic *et al.*



MDS of features from all 175 poses using a full range of scales

# The Wave Kernel Signature (WKS)



Based on solutions of the quantum Schrödinger equation

$$\frac{\partial \psi}{\partial t}(x, t) = i \Delta \psi(x, t)$$

# Shape Search

# Bag-of-Words Models

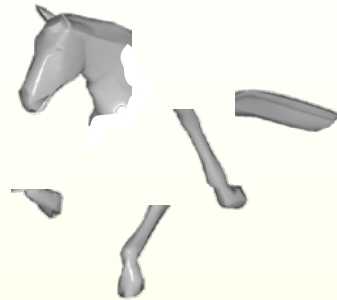
Visual Object



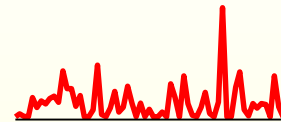
Bag of 'words'



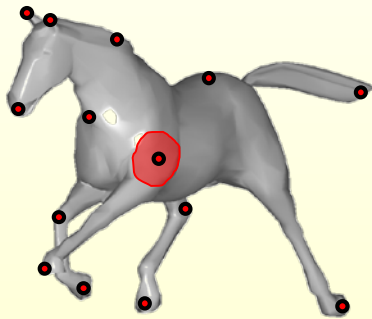
# Spatial Words



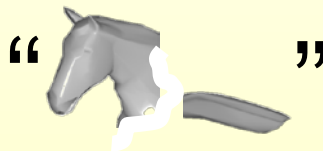
**Geometric words**



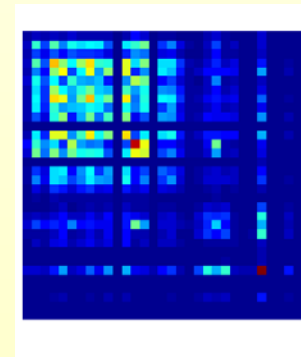
**Bag of geometric words**



**Feature descriptor**

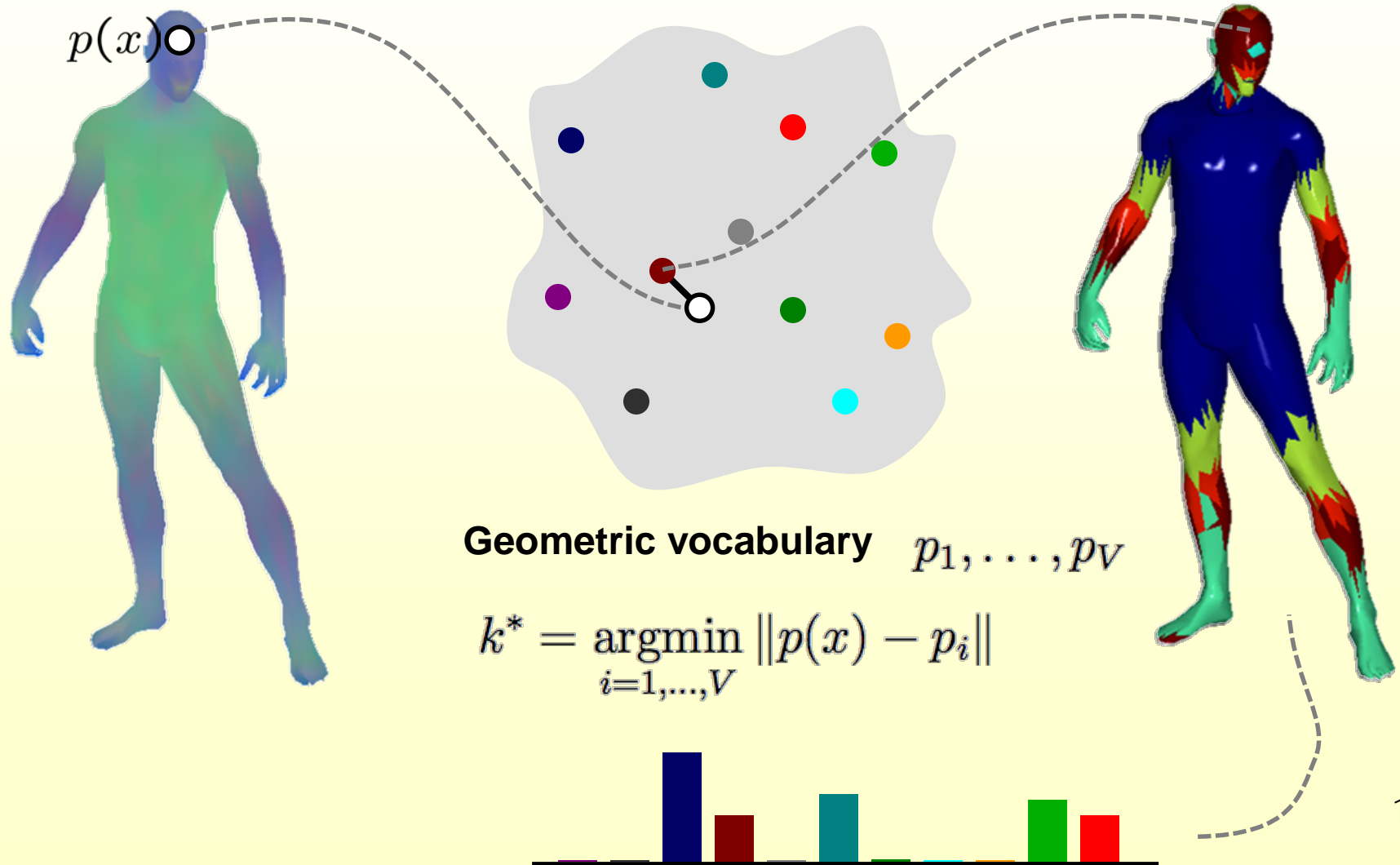


**Geometric expressions**

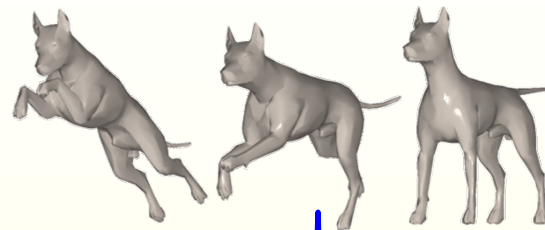


**Spatially-sensitive bag of words**

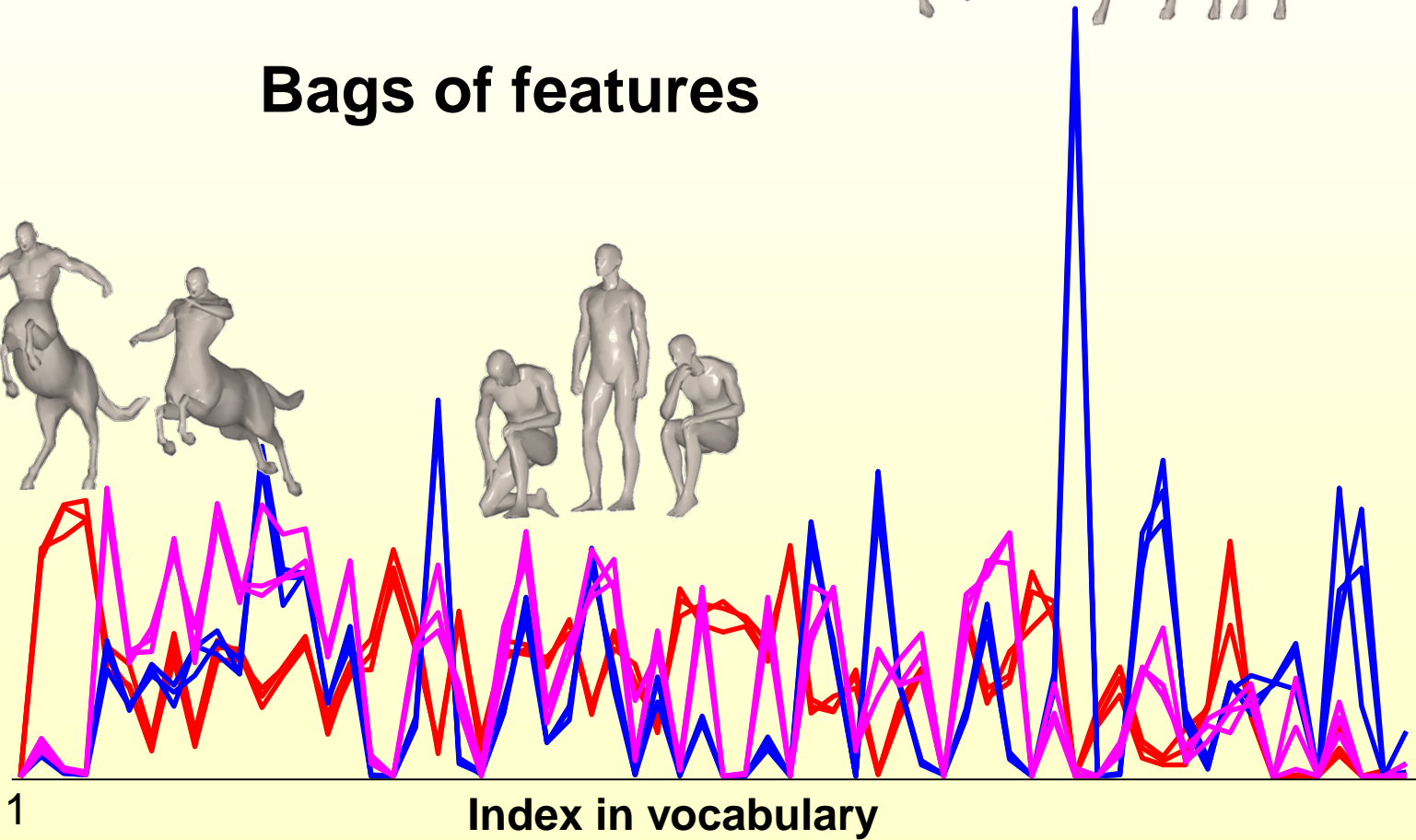
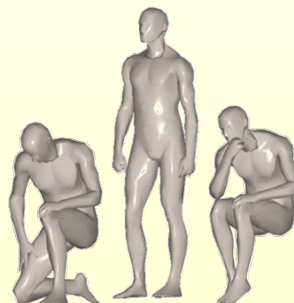
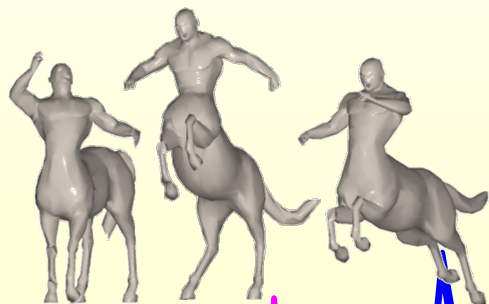
# ShapeGoogle: HKS-Based BoW Shape Search



# Shape Signatures



**Bags of features**



**The End**

