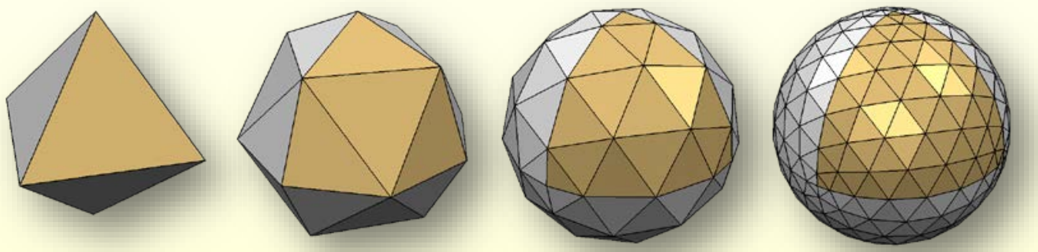
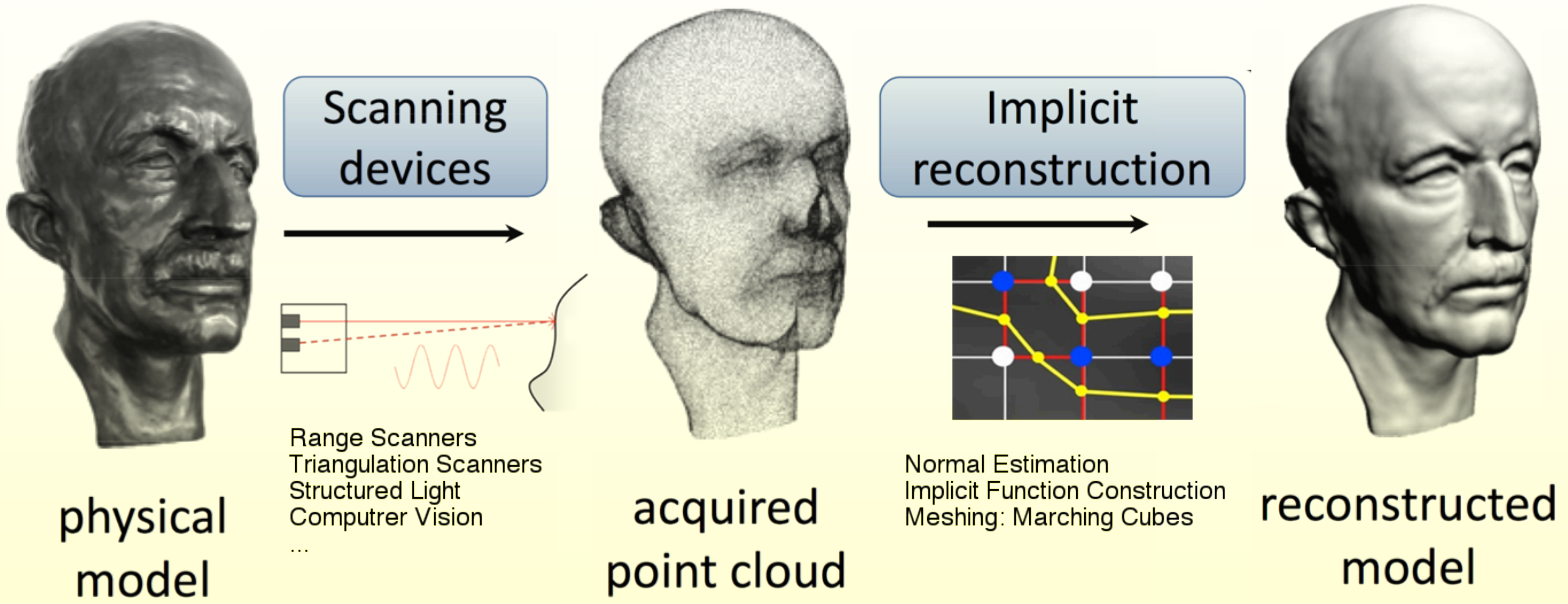
CS348a:
Geometry Processing



Registration and Matching

3D Point Cloud Processing



3D Acquisition Pipeline

3D Point Cloud Processing

This lecture

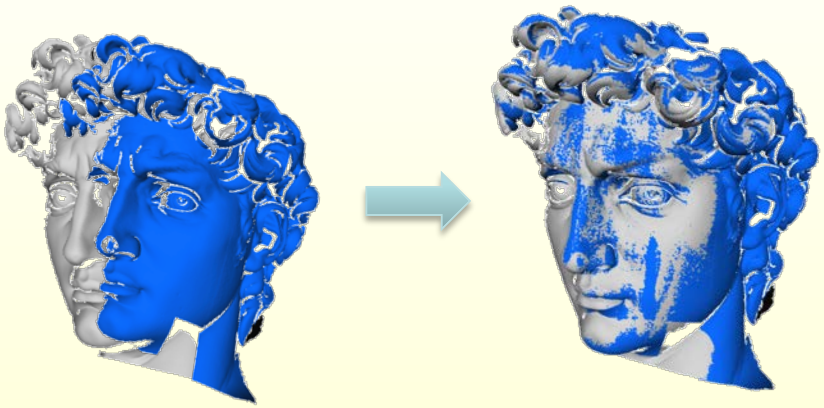
Registration Pipeline

Source: Rusinkiewicz et al.

Steps:

1. Initial registration
2. Pairwise refinement
3. Global relaxation to distribute error
4. Generation of surface

Registration Pipeline

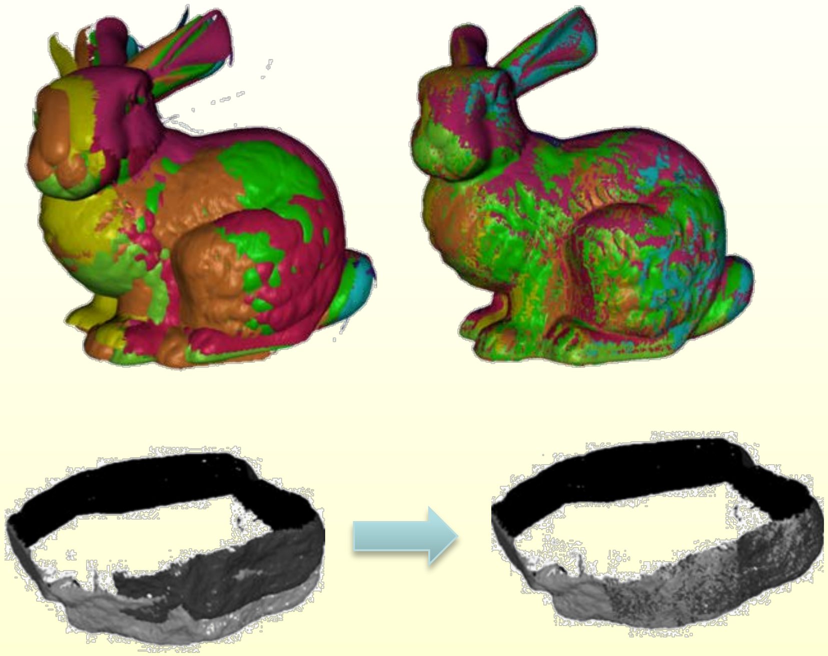


Source: Rusinkiewicz et al.

Steps:

1. Initial registration
2. Pairwise registration
3. Global relaxation to distribute error
4. Generation of surface

Registration Pipeline

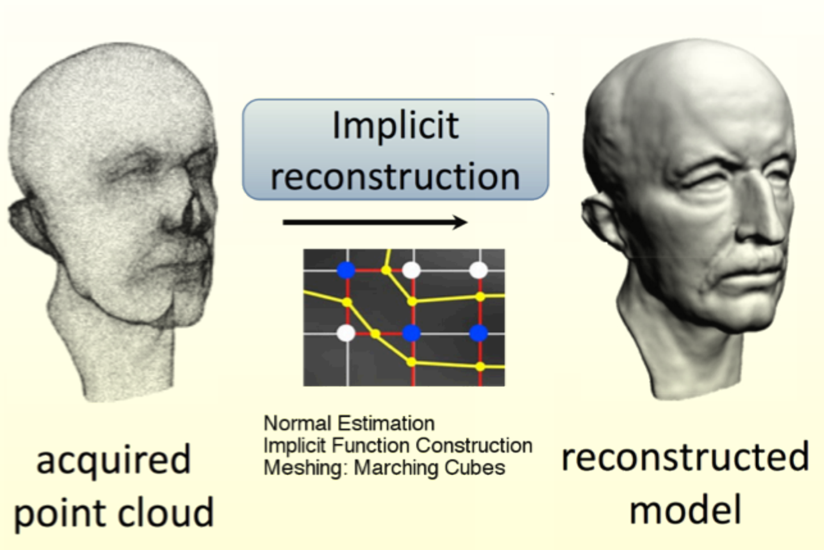


Source: Rusinkiewicz et al.

Steps:

1. Initial registration
2. Pairwise registration
3. Global relaxation to distribute error
4. Generation of surface

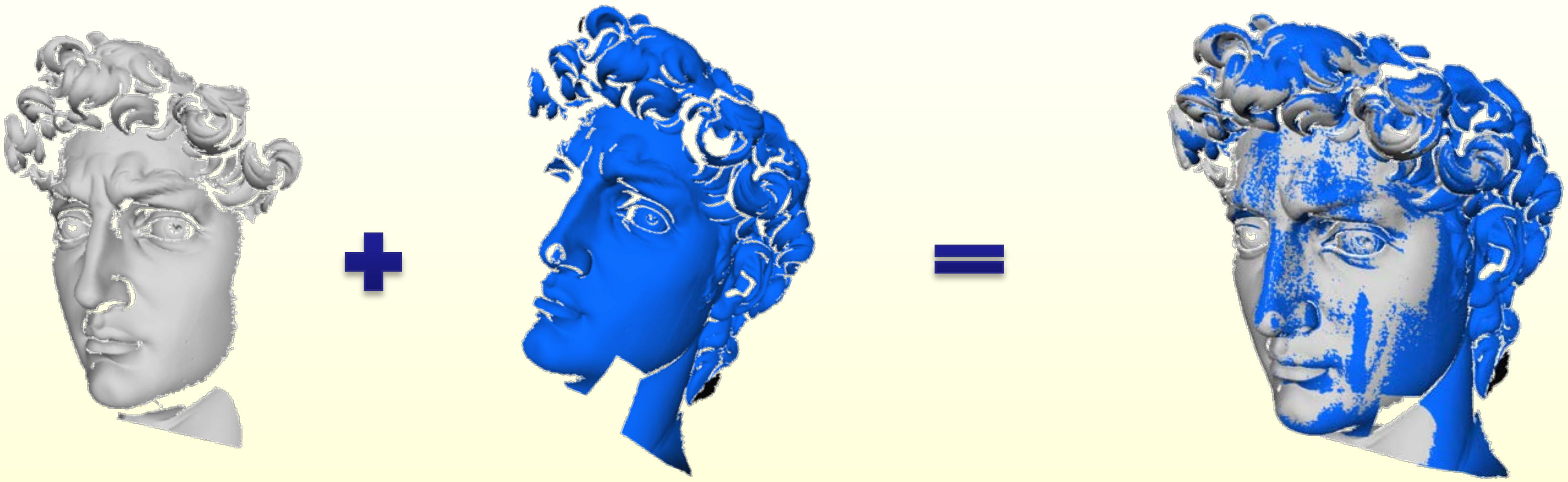
Registration Pipeline



Steps:

1. Initial registration
2. Pairwise registration
3. Global relaxation to distribute error
4. Generation of surface

Fundamental Registration Problem



Given two shapes with partially overlapping geometry, find an alignment between them

Measuring Success: Shape Distances

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

$$\min_T \delta(A, T(B))$$ [extrinsic]

Such measures are crucial in shape similarity search, shape classification, etc.

As another example, shape registration and matching is very important in modern structural biology

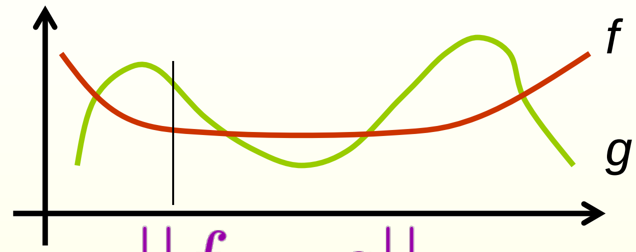
Human (red) and fly (yellow) thierodoxins, compared

Issues about Distance Metrics

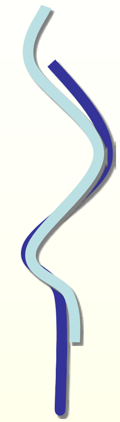
- We are all familiar with function norms (L_2, etc.). The common parametrization establishes correspondences. We don't have that for structures or shapes.
- Partial matches need to be considered -- notion of support σ for the match.
- What group of aligning transforms is to be considered?
- Is the resulting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

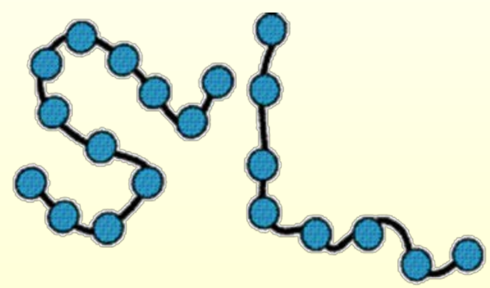
Not for partial matches

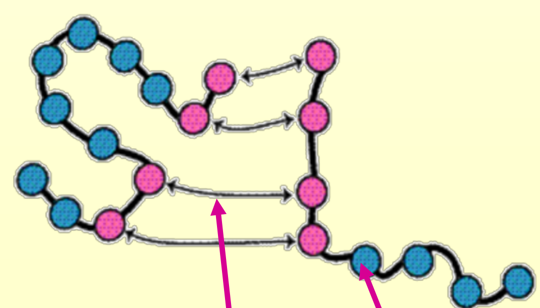
$$\delta(A,C) \leq \delta(A,B) + \delta(B,C)$$



$$||f - g||_2$$



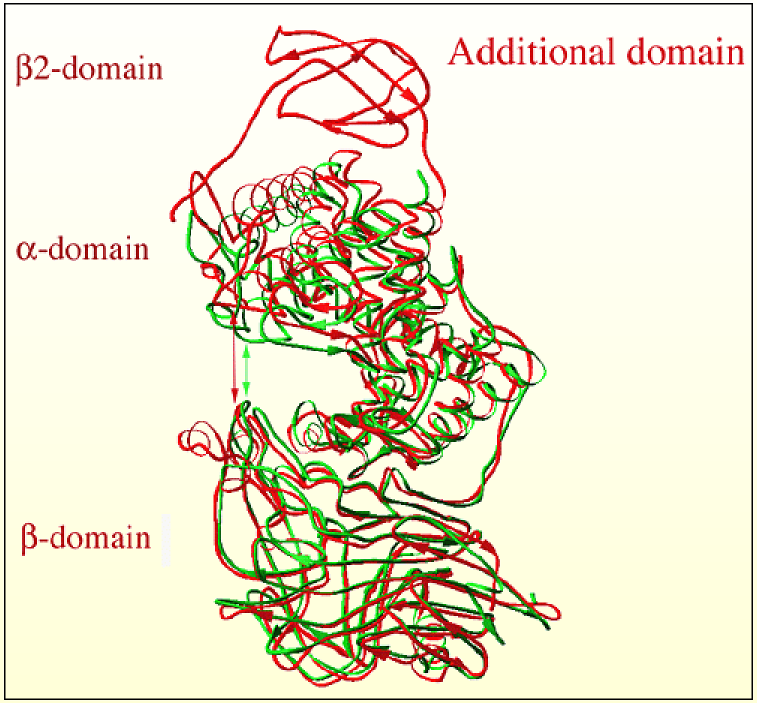


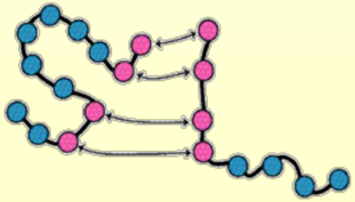


$$\delta(\sigma(A), \sigma(B))$$

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 δ

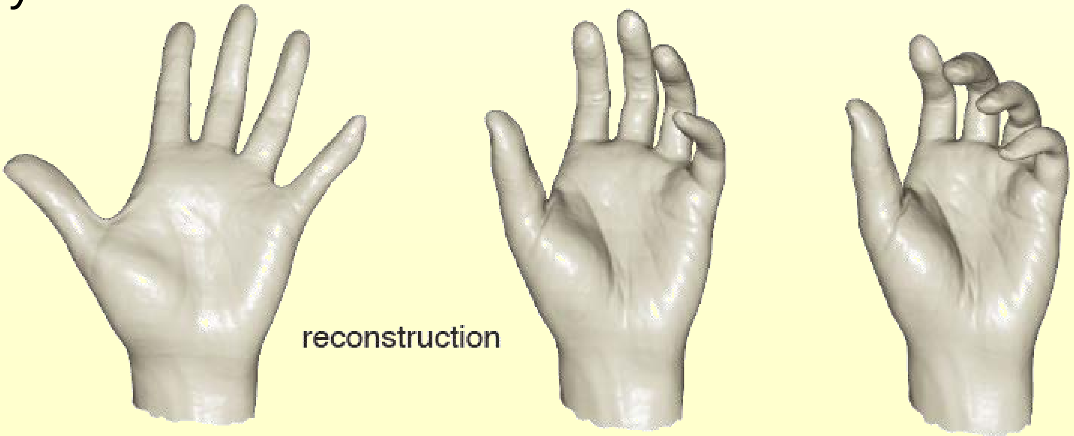




In computing the score, how do we
1. aggregate distances?
2. trade-off larger supports for larger aggregate distance?

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



Other Applications of Alignments

- Manufacturing / Quality Control:

  One shape is a **model** and the other is a **scan** of a product. Useful for finding defects.

- Medicine:

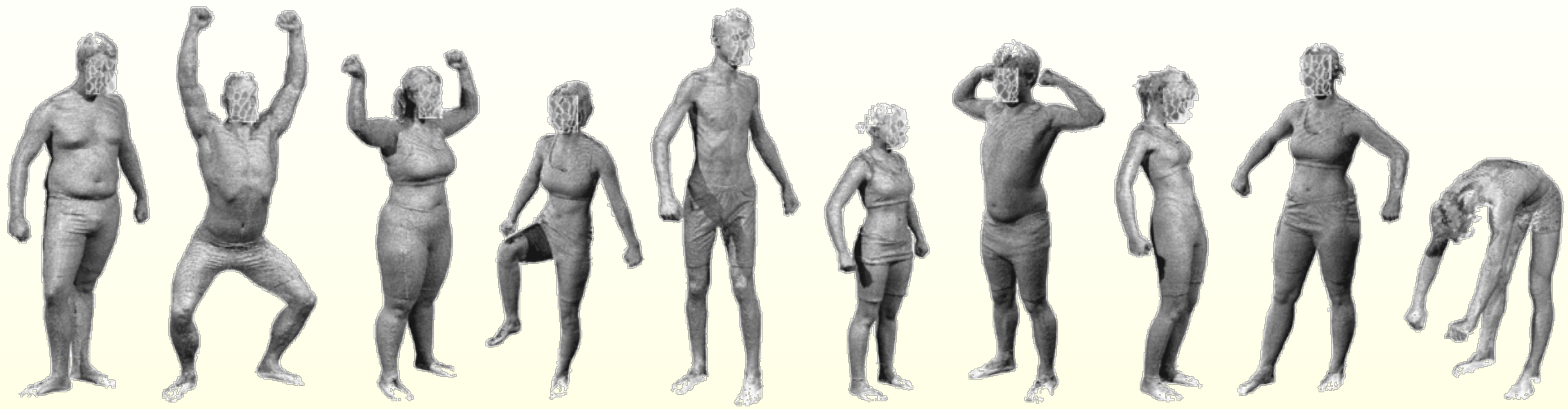
  Finding correspondences between 3D MRI scans of the same person to diagnose or monitor disease.

- Animation Reconstruction & 3D Video.

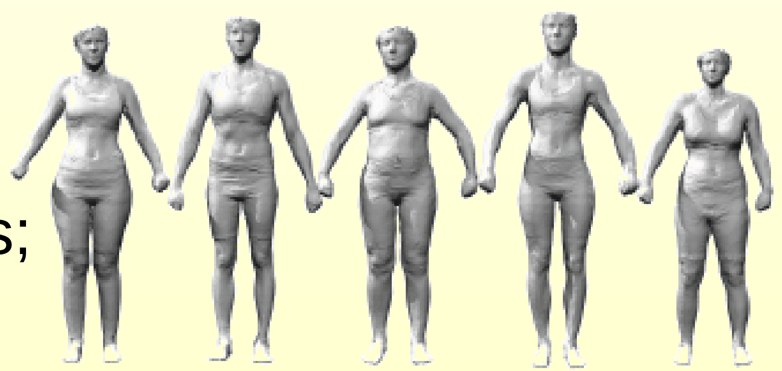
- Statistical Shape Analysis:

  Building models for a collection of shapes.

Applications – Statistical Analysis of Shape Variations



- Scan many people. Learn a deformation model (e.g. PCA).
- Find the principal variation modes; create new random instances.
- Requires alignment.



female, 1.6 m, 65kg

A Statistical Model of Human Pose and Body Shape ,
Hasler et al. Eurographics 09

Method Taxonomy

Local vs. Global

refinement (e.g. ICP) | alignment (search)

Rigid vs. Deformable

rotation, translation | general deformation

Pair vs. Collection

two shapes | multiple shapes

Method Taxonomy

Local vs. Global

refinement (e.g. ICP) | alignment (search)

Today

Rigid vs. Deformable

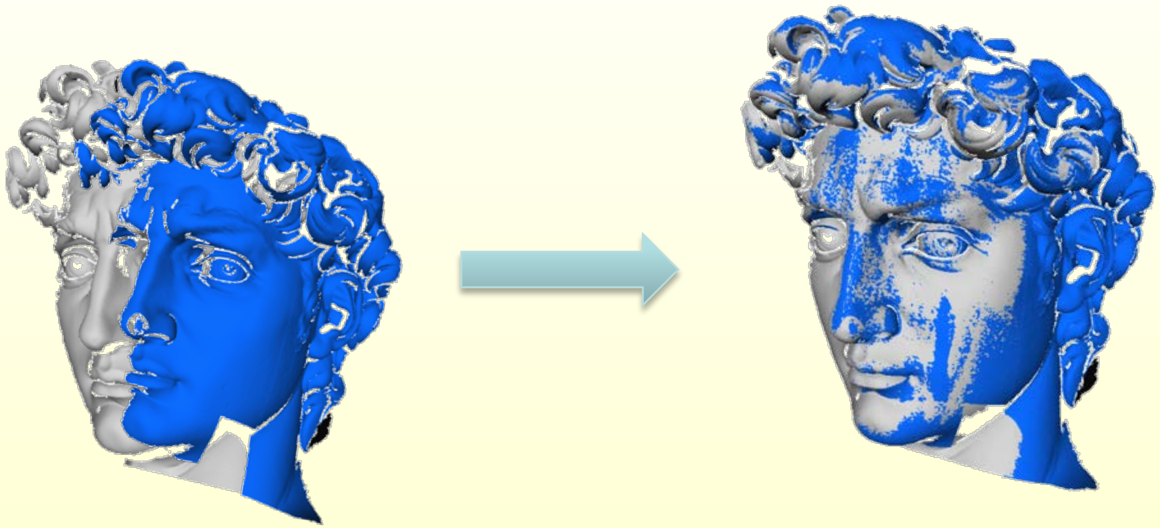
rotation, translation | general deformation

Pair vs. Collection

two shapes | multiple shapes

Local Alignment

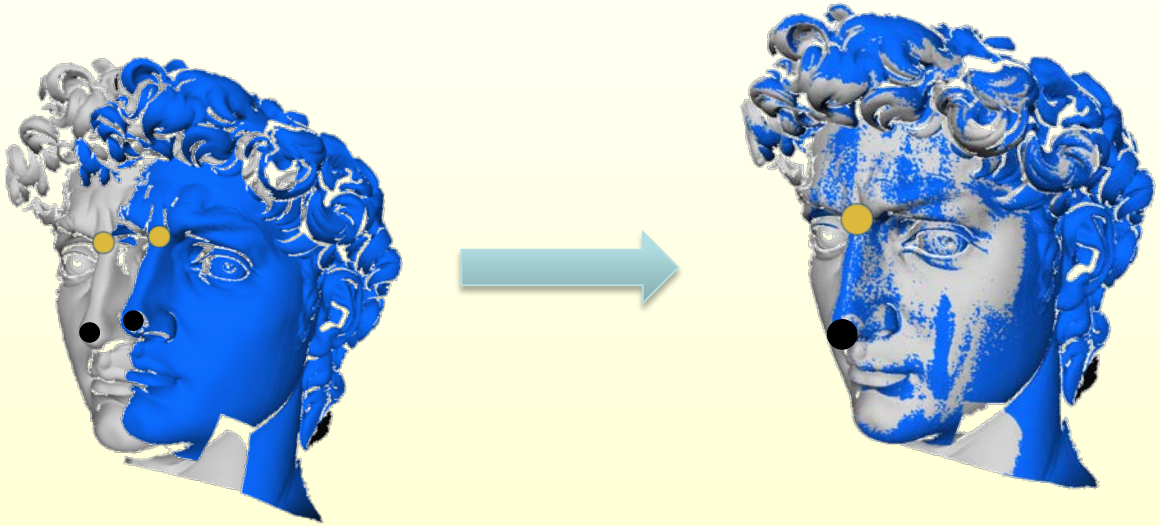
- Simplest instance of the registration problem



Given two shapes that are **approximately aligned** (e.g. by a human, or via prior knowledge) we want to find the optimal rigid transformation that brings them into correspondence.

Local Alignment

- What does it mean for an alignment to be **good**?



Intuition: we want “corresponding points” to be close after transformation.

Problems

1. We don’t know what points correspond.
2. We don’t know the optimal alignment.

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)

Transform ⟷ Correspondences

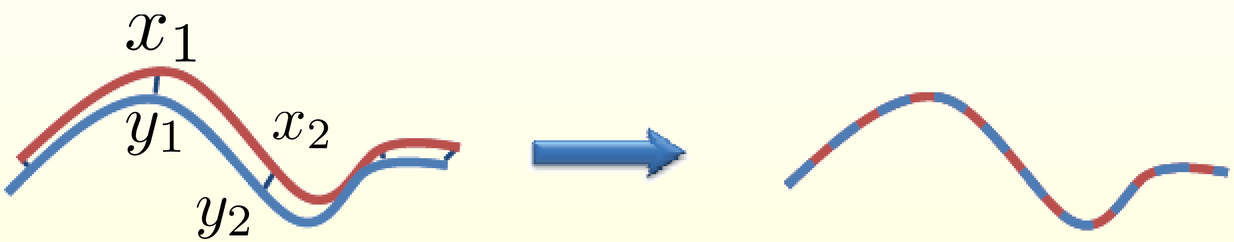
Guess one, estimate the other, and *iterate!*

EM like

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

Iterative Closest Point (ICP)

- Approach: iterate between finding correspondences and finding the transformation:



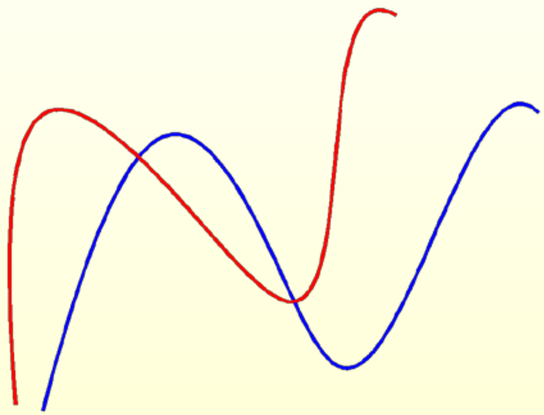
Given a pair of shapes, X and Y, iterate:

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

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

Iterative Closest Point (ICP)

- Approach: iterate between finding correspondences and finding the transformation:

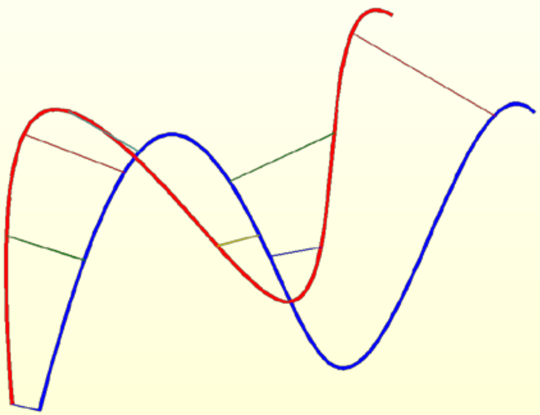


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find rigid motion $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Approach: iterate between finding correspondences and finding the transformation:

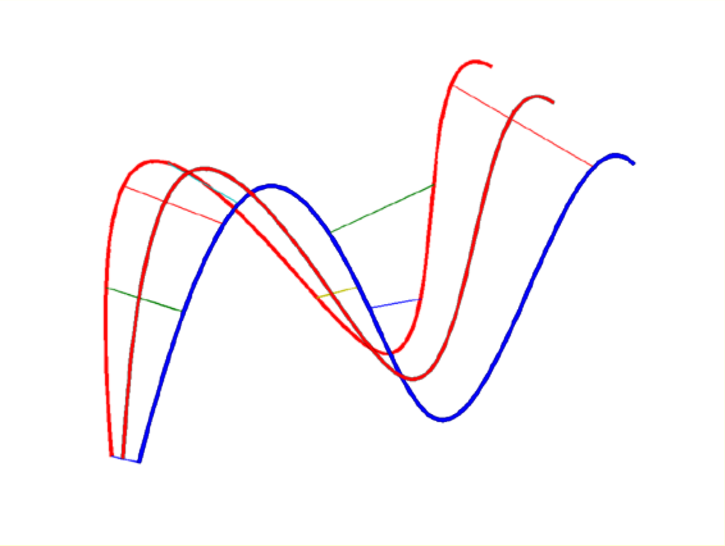


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Approach: iterate between finding correspondences and finding the transformation:

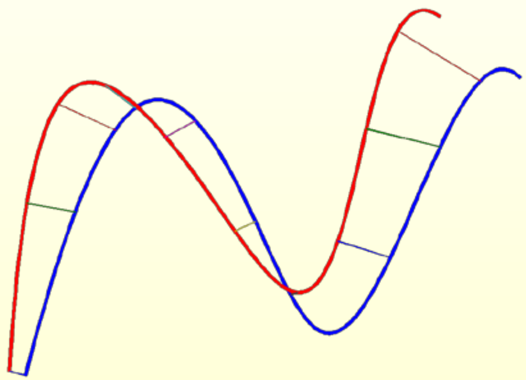


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Approach: iterate between finding correspondences and finding the transformation:

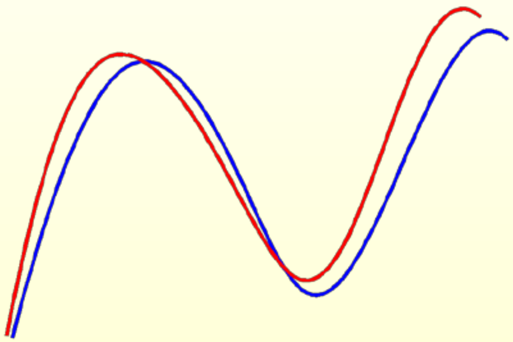


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Approach: iterate between finding correspondences and finding the transformation:

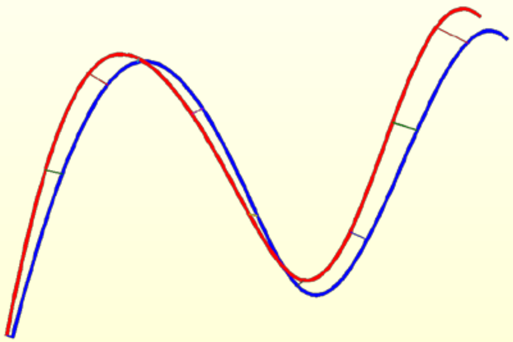


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Approach: iterate between finding correspondences and finding the transformation:

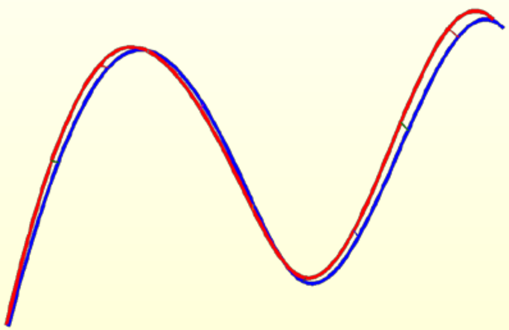


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Approach: iterate between finding correspondences and finding the transformation:

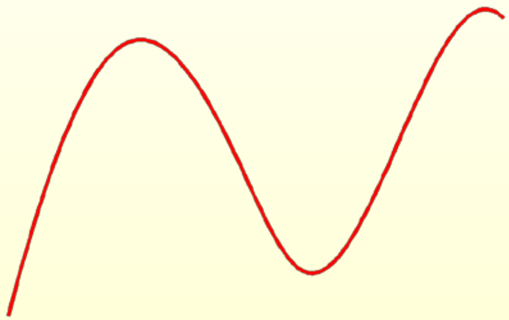


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Approach: iterate between finding correspondences and finding the transformation:



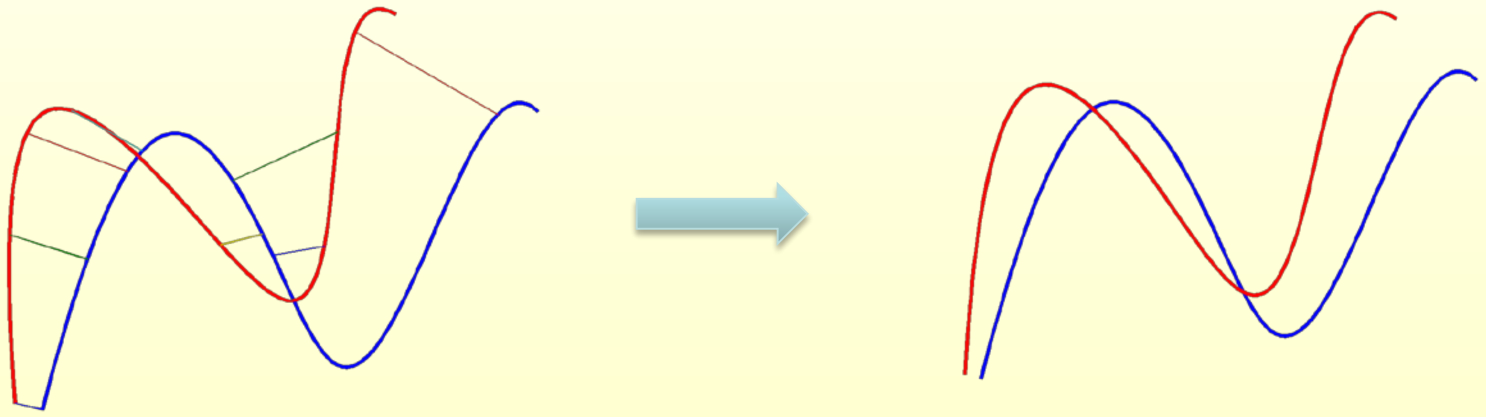
Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

Iterative Closest Point

- Requires two main computations:

1. Computing nearest neighbors
2. Computing the optimal transformation



ICP: Nearest Neighbor Computation

Closest points

$$y_i = \arg\min_{y \in Y} \|y - x_i\|$$

- How to find closest points **efficiently**?
- Straightforward complexity: $\mathcal{O}(MN)$

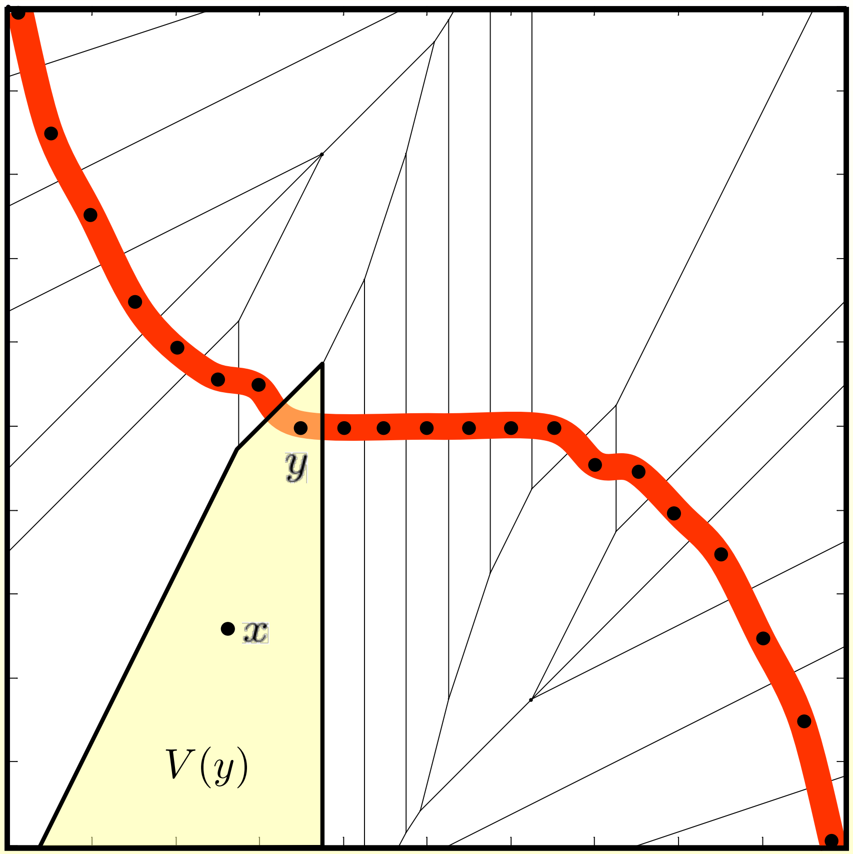
  M number of points on X, N number of points on Y.

- More sophisticated: X divides the space into **Voronoi cells**

$$V(y \in Y) = \{z \in \mathbb{R}^3 : \|y - z\| < \|y' - z\| \ \forall\ y' \in Y \neq y\}$$

- Given a query point y, **determine to which cell it belongs**.

Closest Points: Voronoi Cells

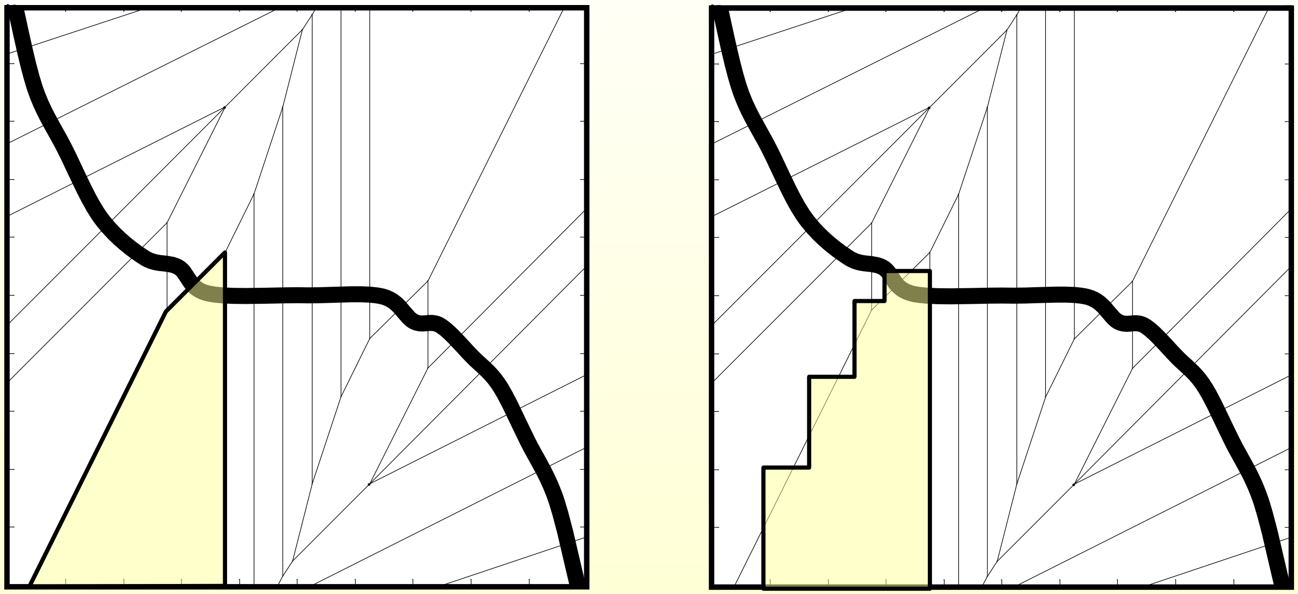


$$V(y \in Y) = \{z \in \mathbb{R}^3 : \|y - z\| < \|y' - z\| \ \forall\ y' \in Y \neq y\}$$

Source:
M. Bronstein

Closest Points: Voronoi Cells

Approximate nearest neighbors

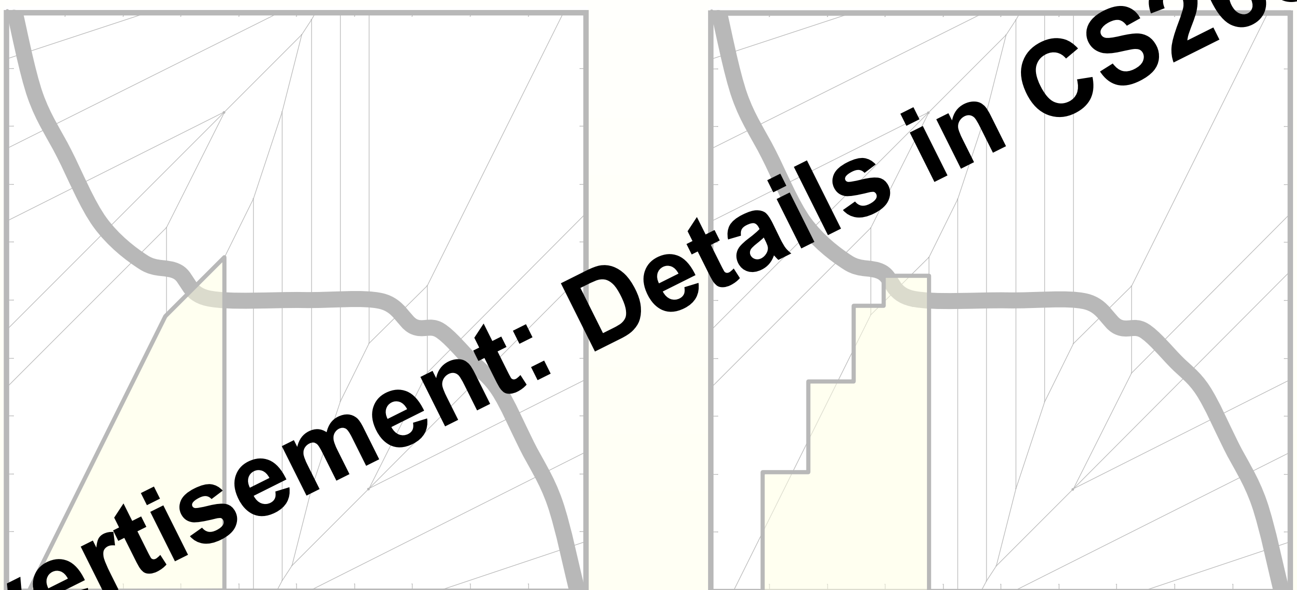


M. Bronstein

- To reduce search complexity, **approximate** Voronoi cells.
- Use **binary space partition trees** (e.g. **kd-trees** or **octrees**).
- Approximate nearest neighbor search complexity: $\mathcal{O}(N \log M)$.

Closest Points: Voronoi Cells

Approximate nearest neighbors



Advertisement: Details in CS268

M. Bronstein

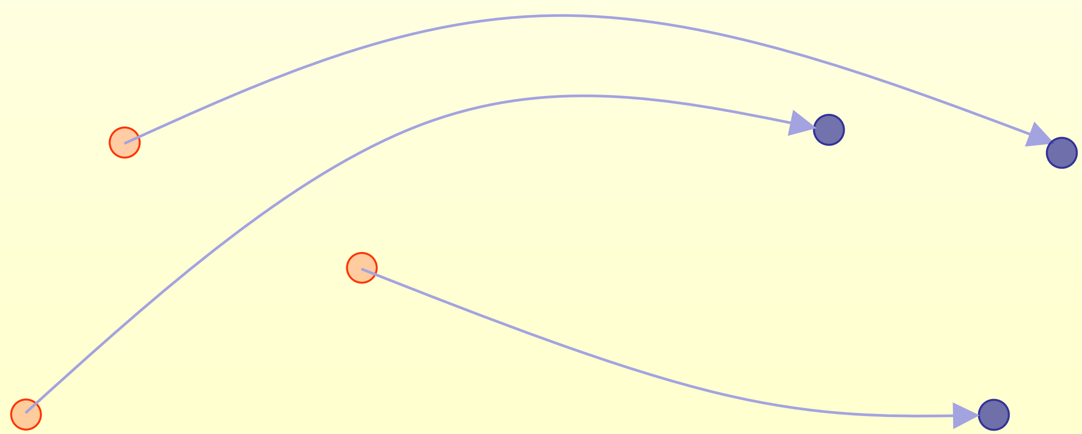
- To reduce search complexity, **approximate** Voronoi cells.
- Use **binary space partition trees** (e.g. **kd-trees** or **octrees**).
- Approximate nearest neighbor search complexity: $\mathcal{O}(N \log M)$.

ICP: Optimal Transformation

Problem Formulation:

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 $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ in $\Re^3$. 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]
  - Know … *absc … m* in Ph … 198
  - Also … med … stati … , 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*.
- The quantity to be minimized is:

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

The unknowns

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):

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

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

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

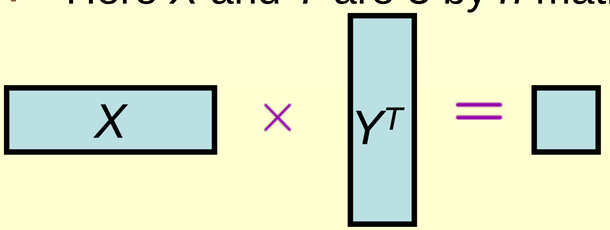
The centroids of the two point sets have to be aligned!

The Rotation Part

- Define

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$
$$X = [x_1 - \bar{x}, \ldots x_n - \bar{x}]^T$$
$$Y = [y_1 - \bar{y}, \ldots y_n - \bar{y}]^T$$

- Here *X* and *Y* are 3 by *n* matrices.



- 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

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

- Then $\boxed{R = USV^T}$

O(*n*) algorithm!

*SVD = singular value decomposition

ICP: Optimal Transformation

Problem Formulation:

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:
$$\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$$
2. Closed form solution:
   1. Construct: $C = \sum_{i=1}^{N}(y_i - \mu^Y)(x_i - \mu^X)^T$, where $\mu^X = \frac{1}{N}\sum_i x_i,$ $\mu^Y = \frac{1}{N}\sum_i y_i$
   2. Compute the SVD of C: $C = U\Sigma V^T$
      1. If $\det(UV^T) = 1, R_{\text{opt}} = UV^T$
      2. Else $R_{\text{opt}} = U\tilde{\Sigma}V^T, \tilde{\Sigma} = \text{diag}(1, 1, \ldots, -1)$
   3. Set $t_{\text{opt}} = \mu^Y - R_{\text{opt}}\mu^X$

Note that C is a 3x3 matrix. SVD is very fast.

Arun et al., Least-Squares Fitting of Two 3-D Point Sets

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

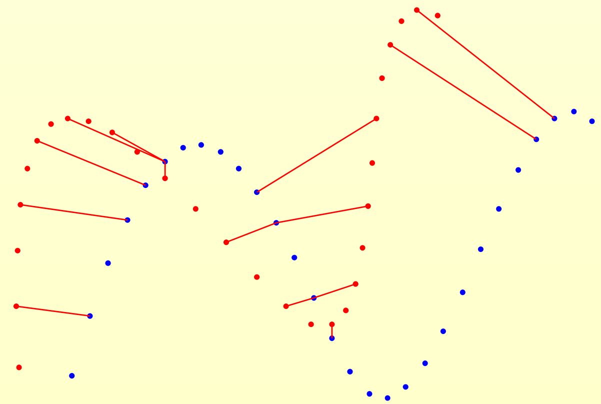
Convergence:

- at each iteration $\sum_{i=1}^{N} d^2(x_i, Y)$ decreases.
- Converges to local minimum
- Good initial guess: global minimum.

[Besl&McKay92]

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

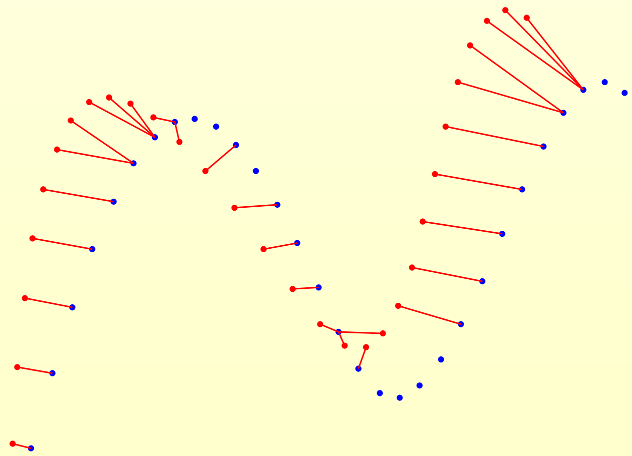


Iterative Closest Point

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:

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



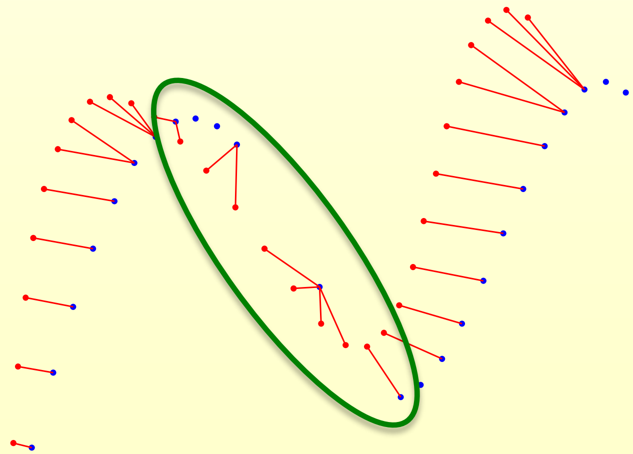
Iterative Closest Point

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



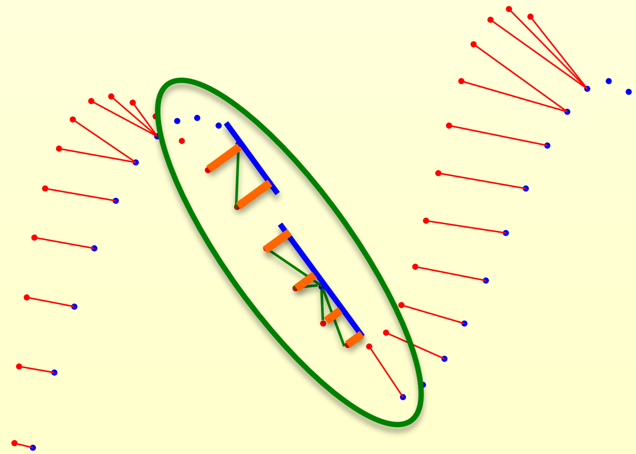
Iterative Closest Point

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:

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

Solution:
Minimize distance to the tangent plane



Chen, Medioni, '91

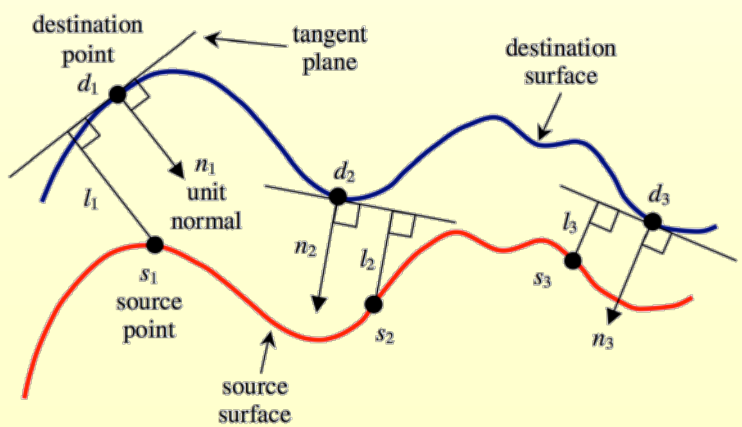
Iterative Closest Point.

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:

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

Solution:
Minimize distance to the tangent plane



Kok-Lim Low, ‘04

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:

$$\mathbf{R}_{\text{opt}}, t_{\text{opt}} = \underset{\boxed{\mathbf{R}^T\mathbf{R}=\text{Id},}\ t}{\arg\min} \sum_{i=1}^{N} \left((\mathbf{R}x_i + t - y_i)^T \mathbf{n}_{y_i} \right)$$

Question:

How to minimize the error?

Challenge:

Although the error is **quadratic** (linear derivative), the space of rotation matrices is **not linear.**

Problem:

No closed form solution.

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:

$$\mathbf{R}_{\text{opt}}, t_{\text{opt}} = \underset{\boxed{\mathbf{R}^T\mathbf{R}=\text{Id},} \, t}{\arg\min} \sum_{i=1}^{N} \left((\mathbf{R}x_i + t - y_i)^T \mathbf{n}_{y_i} \right)$$

Common Approach:

Linearize rotation. Assume rotation angle is small.

$$\mathbf{R}x_i \approx x_i + r \times x_i$$

r : axis,
$\|r\|_2$: angle of rotation.

Note: follows from Rodrigues's formula

$$R(r,\alpha)x_i = x_i \cos(\alpha) + (r \times x_i)\sin(\alpha) + r(r^T x_i)(1 - \cos(\alpha))$$

And first order approximations: $\sin(\alpha) \approx \alpha, \;\; \cos(\alpha) \approx 1$

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 r, t minimizing:

$$E(r,t) = \sum_{i=1}^{N} \left((x_i + r \times x_i + t - y_i)^T \mathbf{n}_{y_i} \right)$$

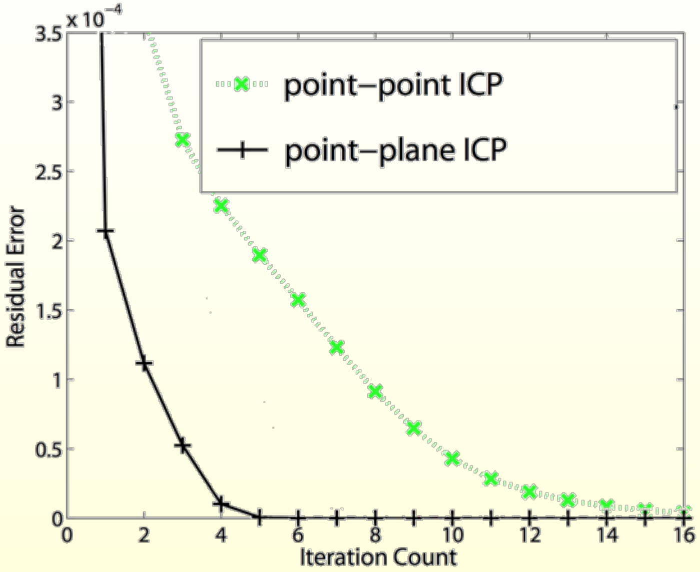
$$= \sum_{i=1}^{N} \left((x_i - y_i)^T \mathbf{n}_{y_i} + r^T (x_i \times \mathbf{n}_{y_i}) + t^T \mathbf{n}_{y_i} \right)^2$$

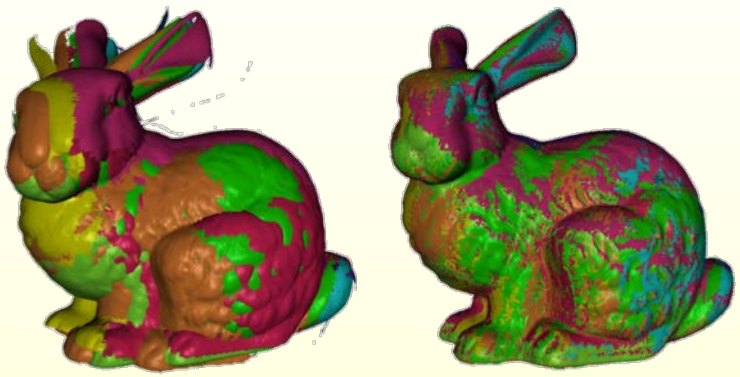
Setting: $\frac{\partial}{\partial r} E(r,t) = 0$ and $\frac{\partial}{\partial t} E(r,t) = 0$ leads to a 6x6 linear system

$$Ax = b$$

$$x = \begin{pmatrix} r \\ t \end{pmatrix} \quad A = \sum \begin{pmatrix} x_i \times \mathbf{n}_{y_i} \\ \mathbf{n}_{y_i} \end{pmatrix} \begin{pmatrix} x_i \times \mathbf{n}_{y_i} \\ \mathbf{n}_{y_i} \end{pmatrix}^T \quad b = \sum (y_i - x_i)^T \mathbf{n}_{y_i} \begin{pmatrix} x_i \times \mathbf{n}_{y_i} \\ \mathbf{n}_{y_i} \end{pmatrix}$$

Iterative Closest Point

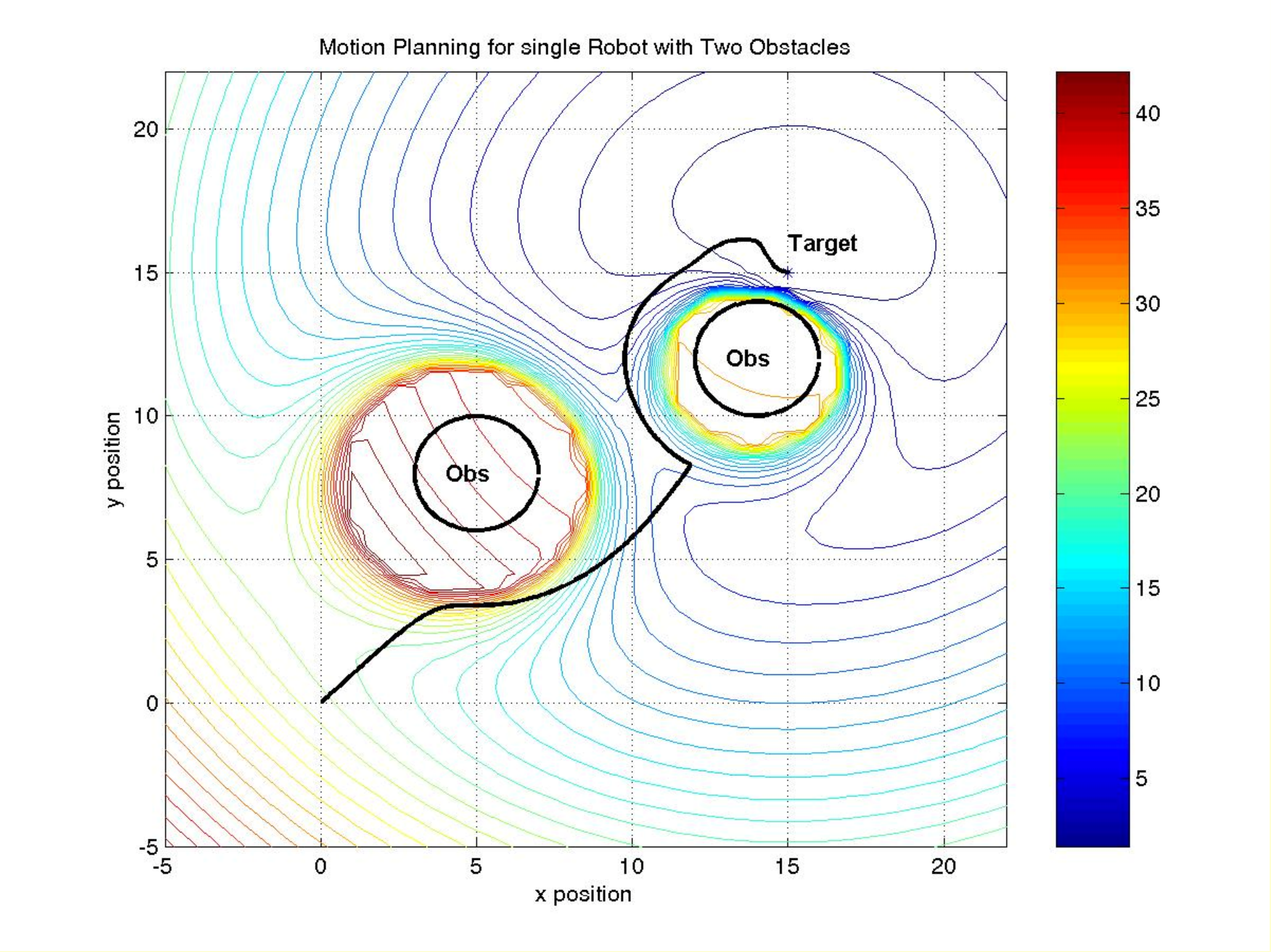




Aligning the bunny to itself:
Point-to-plane always wins in the end-game.

Distance Fields for Registration

“Gravitational” Potential



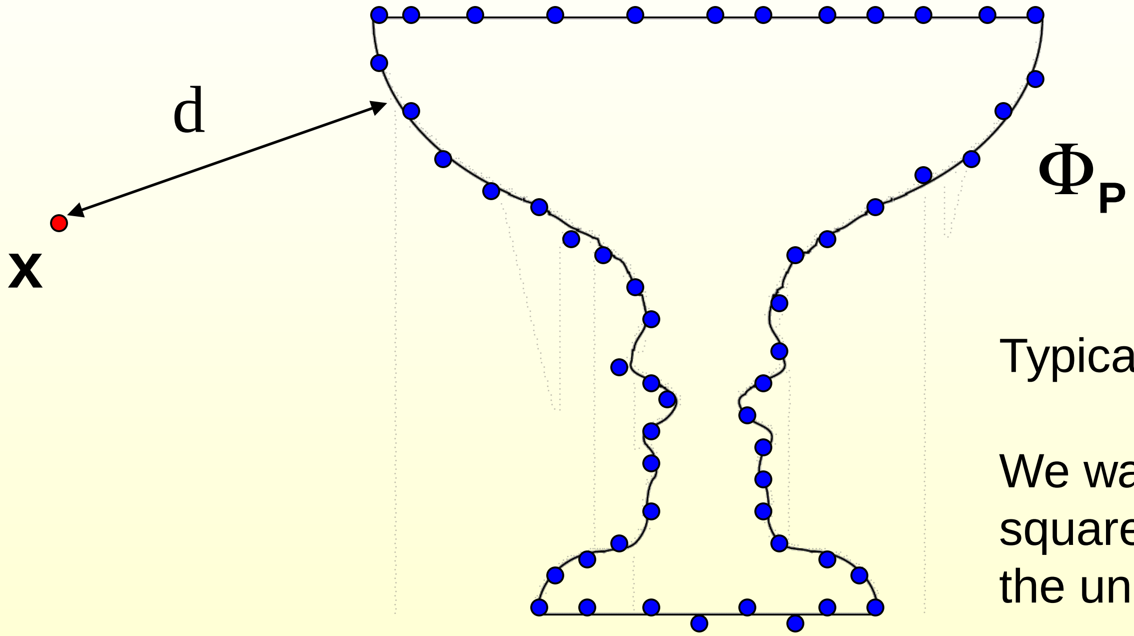
Robot motion planning via potential fields

“Gravitational” Potential

- Given two related shapes, the “data” A and the “model” B, create a potential field that pulls B to the correct alignment with A

- Key tasks
  - Define the potential field
  - Formulate the optimization problem
  - Do gradient descent using approximate

Squared Distance Function (**F**)

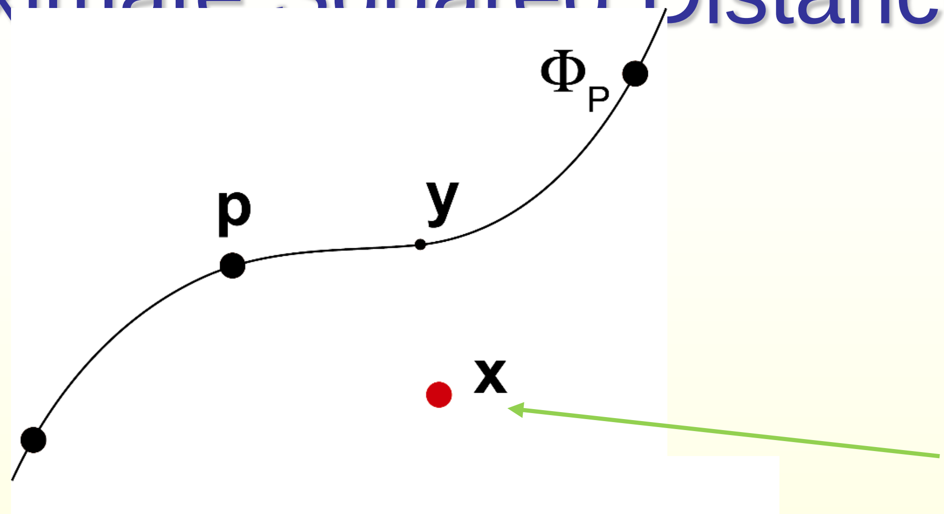


Typically A is a point cloud

We want to approximate the squared distance function to the underlying object

$$\mathbf{F}(\mathbf{x}, \Phi_{\mathbf{P}}) = d^2$$

Approximate Squared Distance



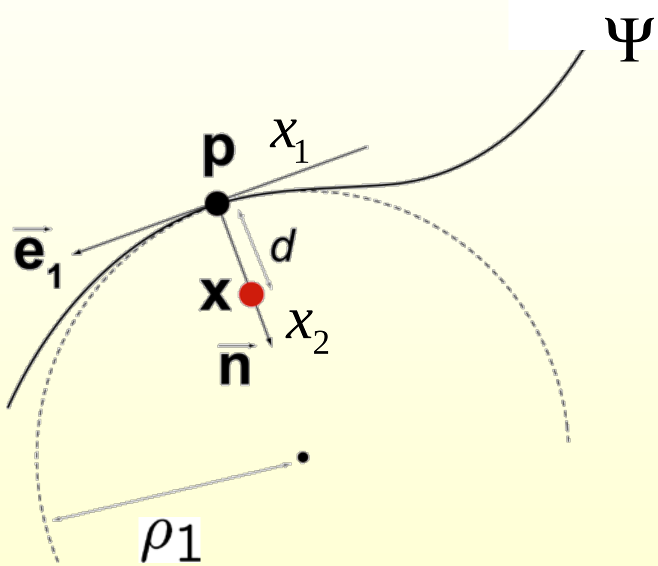
$\mathbf{F(x, \Phi_P)}$ valid in the neighborhood of **x**

Aim for 2nd order approximation, because we want to take derivatives.

Pairwise Rigid Correspondence

Geometry of the square distance function

For a curve Ψ,
around point x.



$y = (x_1, x_2)$

To second order:

$$d^2(y, \Psi) \approx \frac{d}{d - \rho_1} x_1^2 + x_2^2$$ in the Frenet frame at *p*

[Pottmann and Hofer 2003]

Approximate Squared Distance

For a curve Ψ, to second order:

$$d^2(y, \Psi) \approx \frac{d}{d - \rho_1} x_1^2 + x_2^2$$

For a surface Φ, to second order:

$$d^2(y, \Phi) \approx \frac{d}{d - \rho_1} x_1^2 + \frac{d}{d - \rho_2} x_2^2 + x_3^2$$

$\rho_1 = 1/\kappa_1$ and $\rho_2 = 1/\kappa_2$ are inverse principal curvatures

[Pottmann and Hofer 2003]

Approximate Squared Distance

For a surface Φ, to second order:

$$d^2(y,\Phi) \approx \frac{d}{d-\rho_1}x_1^2 + \frac{d}{d-\rho_2}x_2^2 + x_3^2$$

$\rho_1 = 1/\kappa_1$ and $\rho_2 = 1/\kappa_2$ are inverse principal curvatures

Note that as $d \to 0, \quad d^2(y,\Phi) \to x_3^2$ point-to-plane

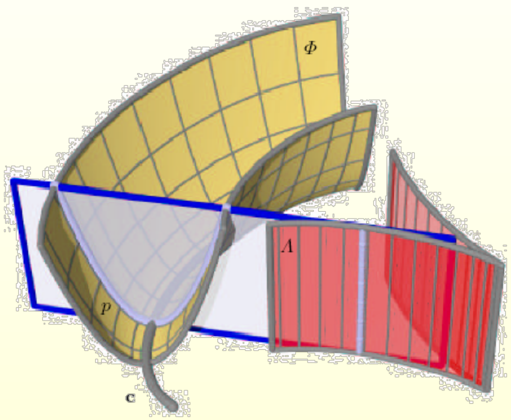
$d \to \infty, \quad d^2(y,\Phi) \to x_1^2 + x_2^2 + x_3^2$ point-to-point

In general neither metric guarantees second order consistency

ICP Without Correspondences

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

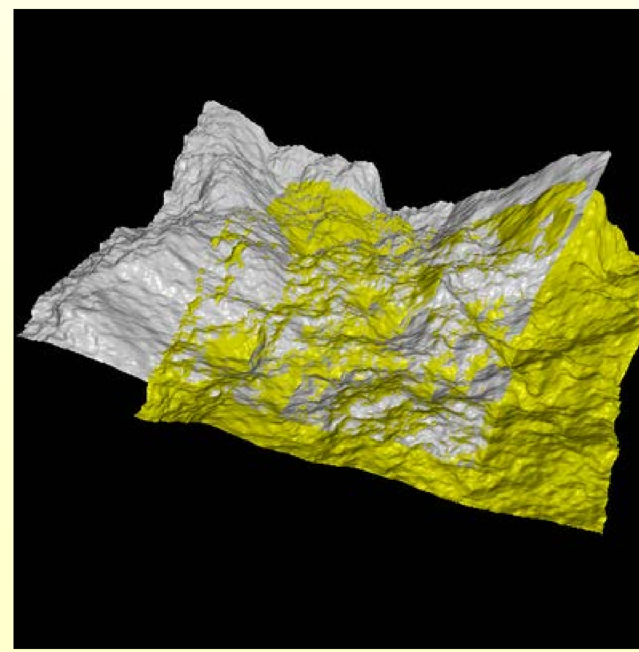
[Pottman & Hofer, 02]



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

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

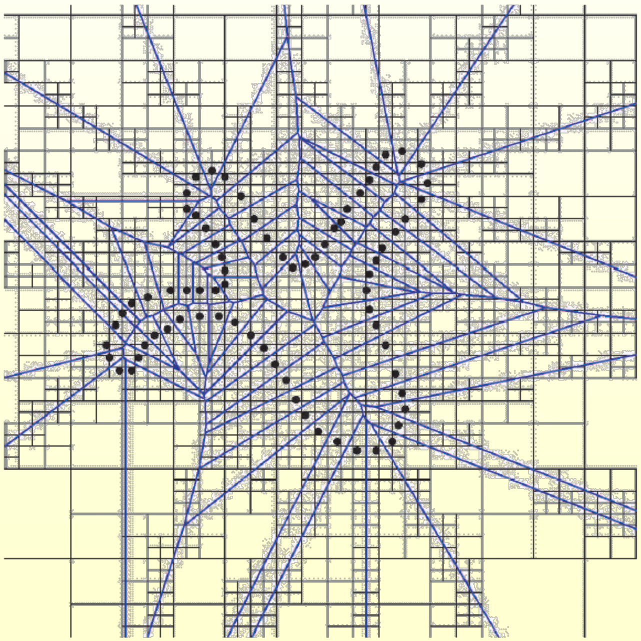
curvature



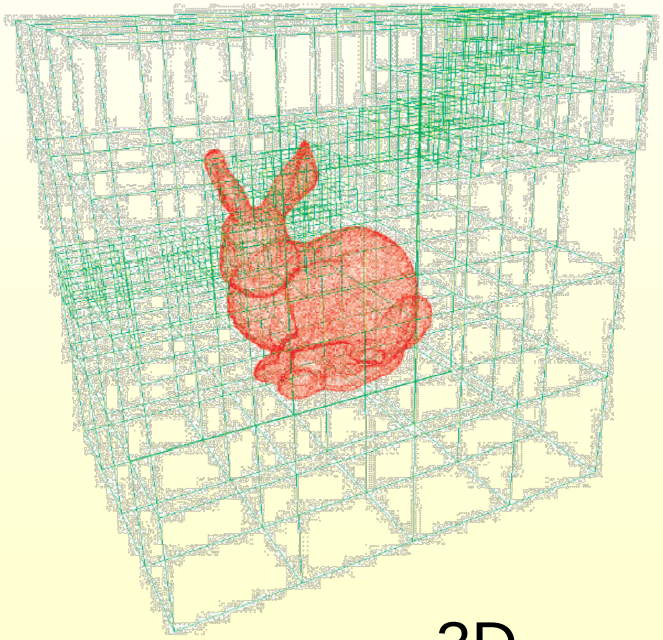
  - perform iterative gradient-descent in this field
  - point to foot-point distance
    - case *d* is large: classical ICP
    - case *d* is small: point-to-plane ICP

$d^2(y, \Phi_P)$ Using d2 Tree

Partition the space into cells where each cell stores a quadratic approximant of the squared distance function.



2D



3D

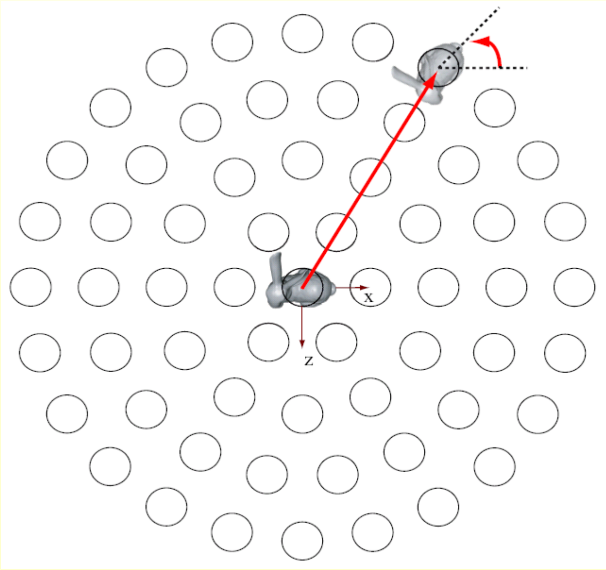
Leopoldseder S et al. *d2-tree: A hierarchical representation of the squared distance function*

Registration Using d2 Tree

Build using bottom-up approach: fit a quadratic approximation to a fine grid.

Merge cells if they have similar approximations.

Funnel of convergence:

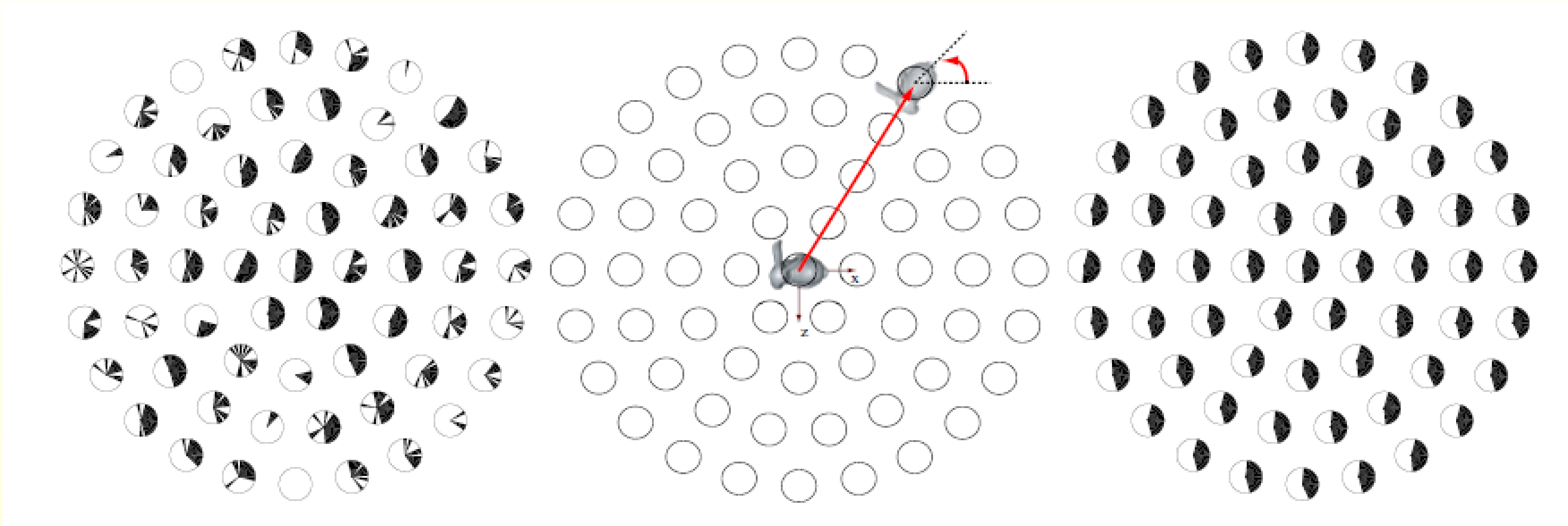


Translation in x-z plane.
Rotation about y-axis.

Converges

Does not converge

Matching the Bunny to Itself

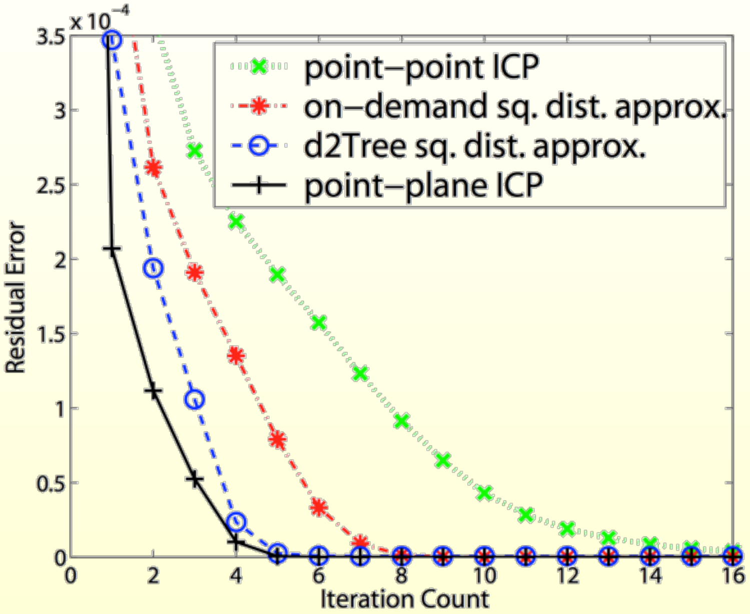


point-to-plane

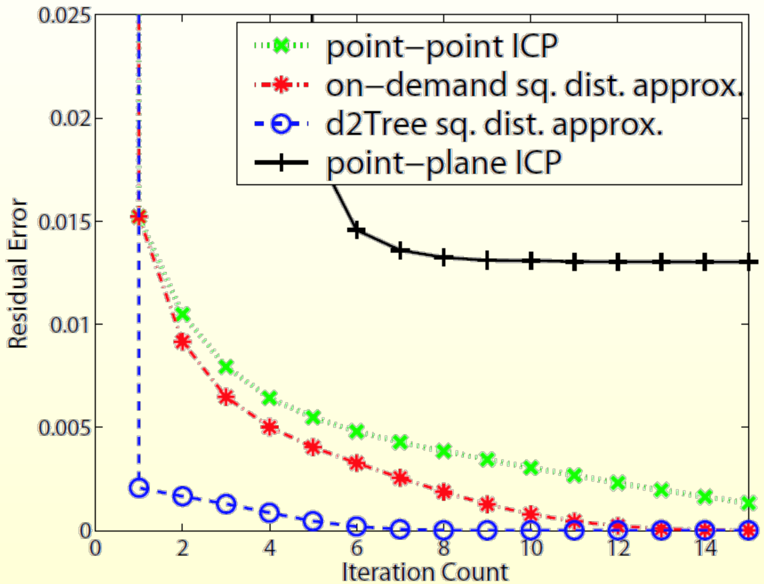
d2Tree

Registration of Point Cloud Data from a Geometric Optimization Perspective
Mitra et al., SGP 2004

Matching the Bunny to Itself



Well-aligned



Noisy, far away

Registration of Point Cloud Data from a Geometric Optimization Perspective
Mitra et al., SGP 2004

Local Rigid Matching – ICP

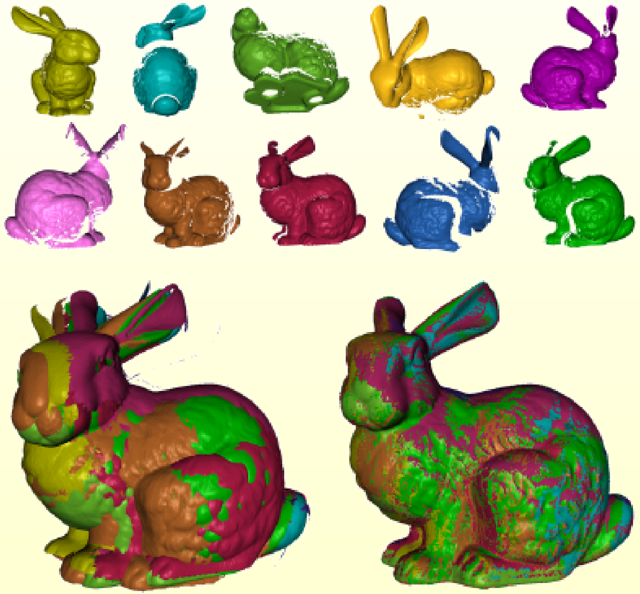
The upshot is that

- Locally, the point-to-plane metric provides a second order approximation to the squared distance function.
- Optimization based on point-to-plane will converge quadratically to a local minimum.
- Convergence funnel can be narrow, but can improve it with either d2tree or point-to-point.

What if we are outside the convergence funnel?

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

Global Matching – Approaches

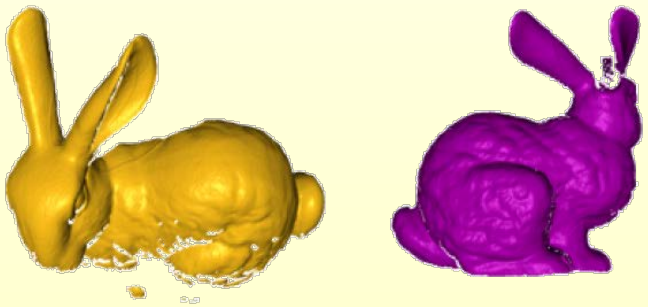
Several classes of approaches:

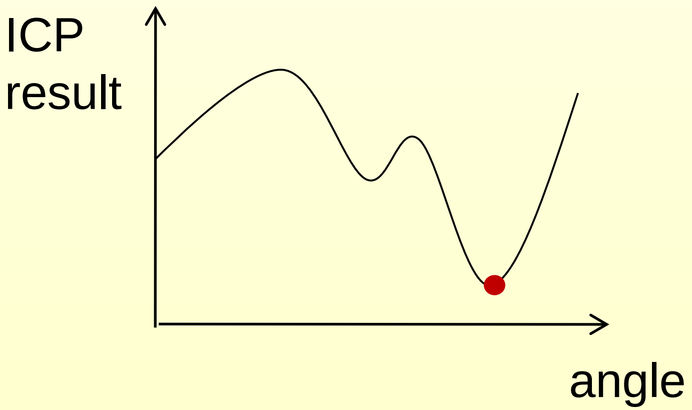
1. Exhaustive Search
2. Normalization
3. Random Sampling
4. Invariance

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

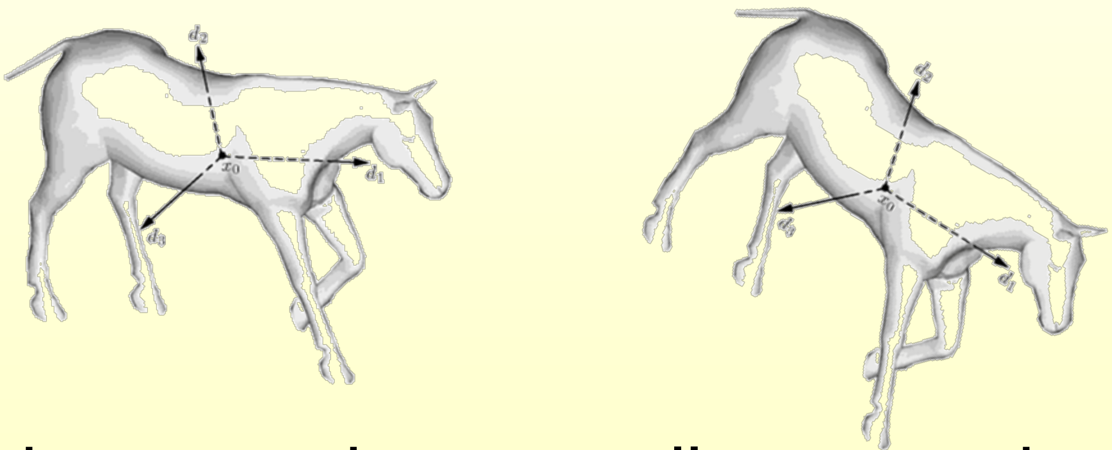
- Sample the space of possible initial alignments
- Correspondence is determined by the alignment at which models are closest

- Provides optimal result
- Can be unnecessarily slow
- Does not generalize to non-rigid deformations

Normalization – Canonical Poses

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

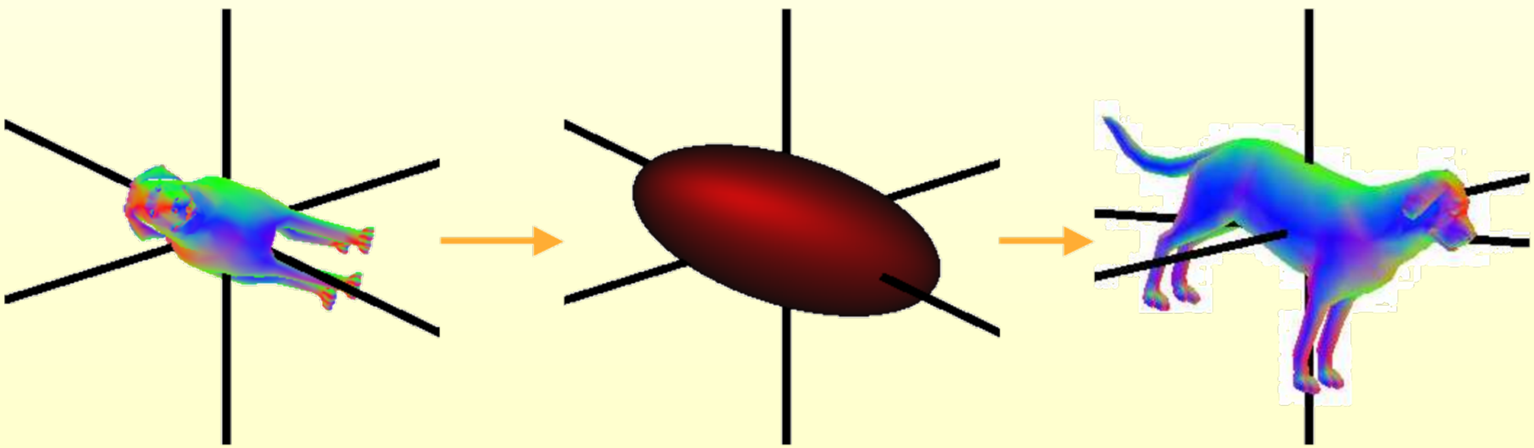
Can center all shapes at the origin and use PCA to find the principal directions of the shape.



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

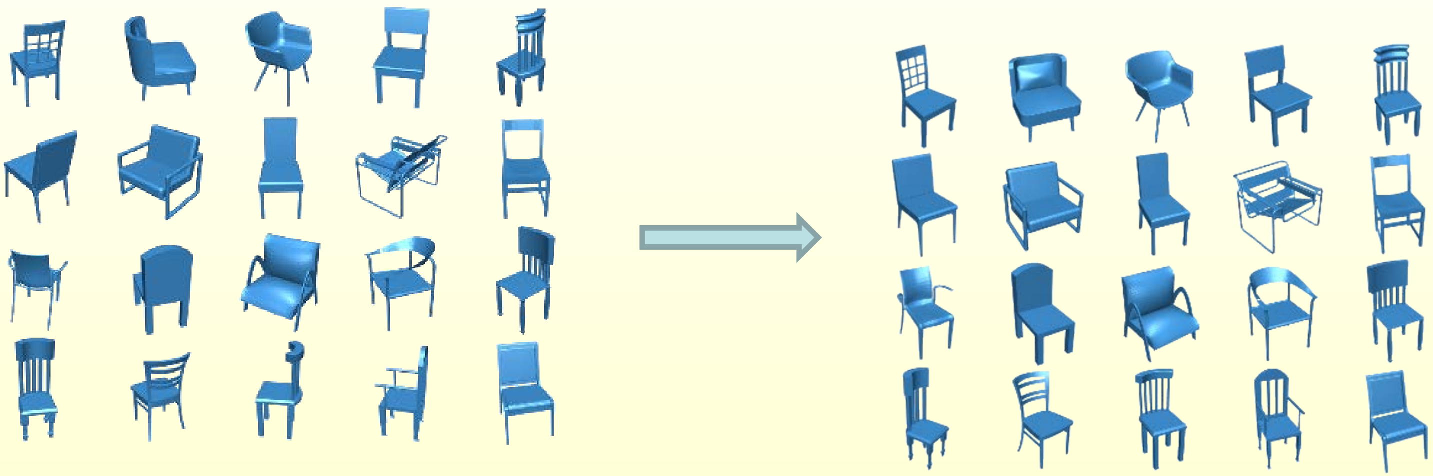
PCA-Based Alignment

- 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.

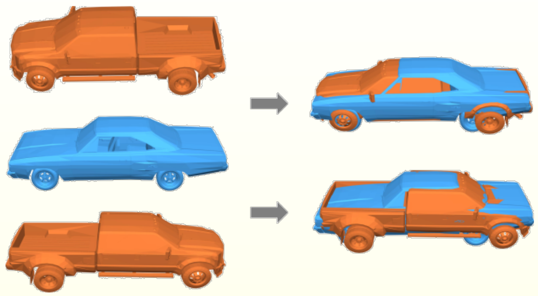


Works well if we have complete shapes and no noise.

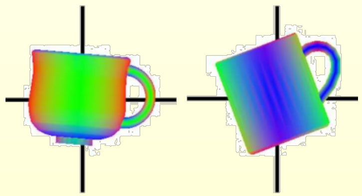
Fails for partial scans, outliers, high noise, etc.

Problems with PCA

- Principal axes are not consistently oriented



- Axes are unstable when principal values are similar



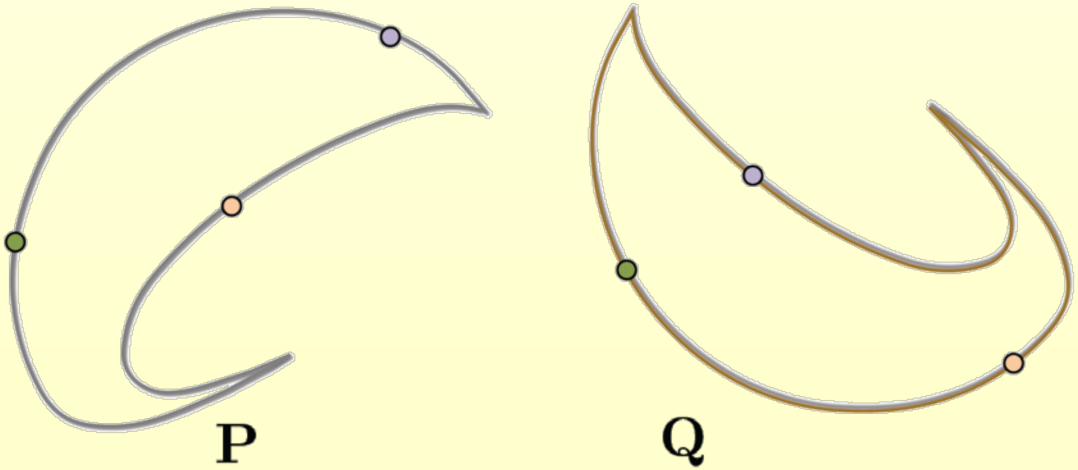
- Partial similarity

Random Sampling (RANSAC)

ICP only needs 3 point pairs!

Robust and simple approach. Iterate between:

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



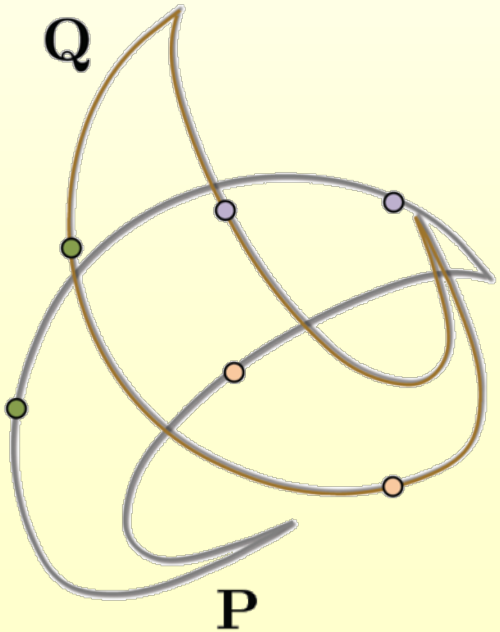
Guess and verify

Random Sampling (RANSAC)

ICP only needs 3 point pairs!

Robust and simple approach. Iterate between:

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



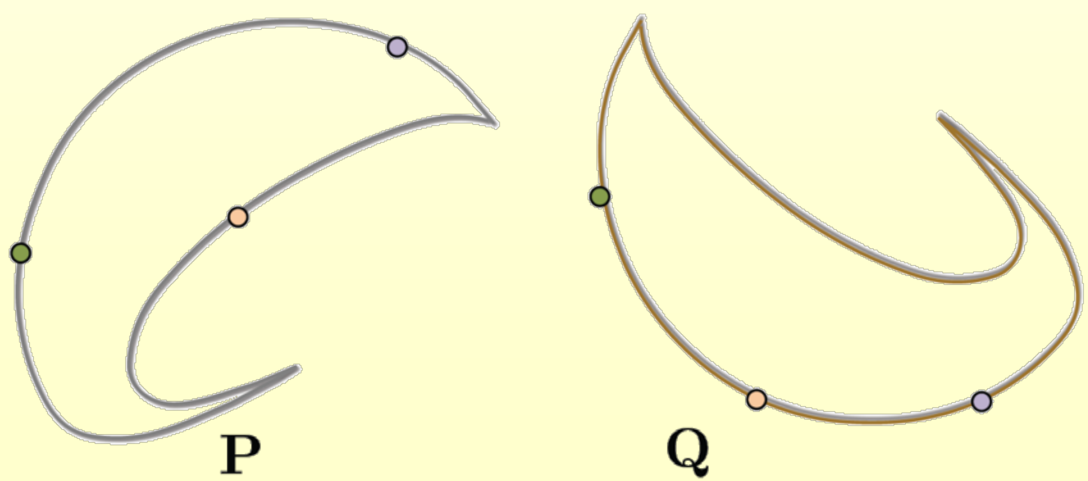
Guess and verify

Random Sampling (RANSAC)

ICP only needs 3 point pairs!

Robust and Simple approach. Iterate between:

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



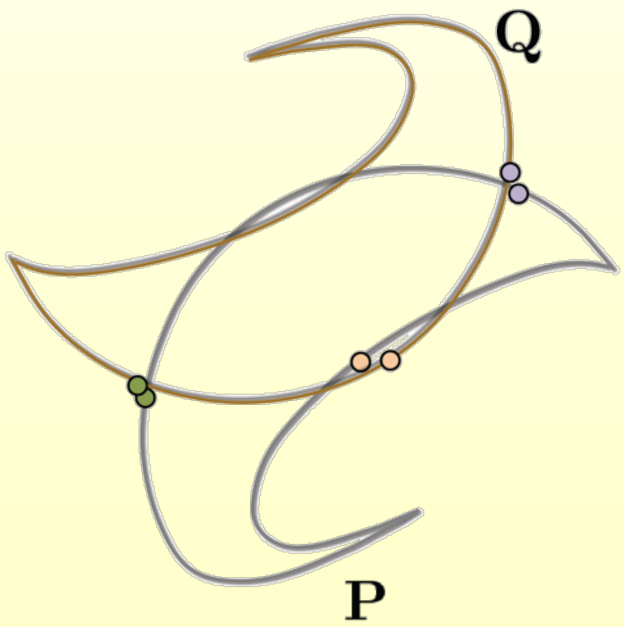
Guess and verify

Random Sampling (RANSAC)

ICP only needs 3 point pairs!

Robust and simple approach. Iterate between:

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



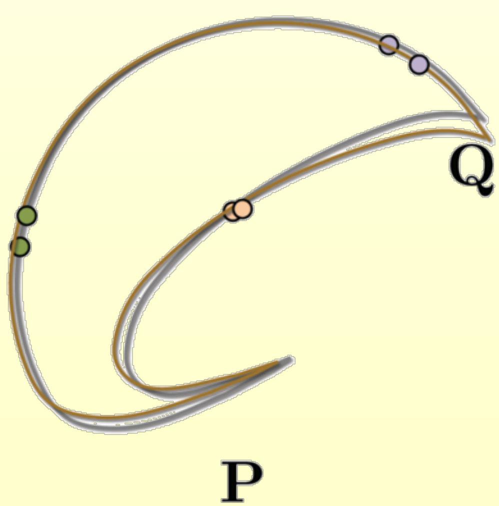
Guess and verify

Random Sampling (RANSAC)

ICP only needs 3 point pairs!

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

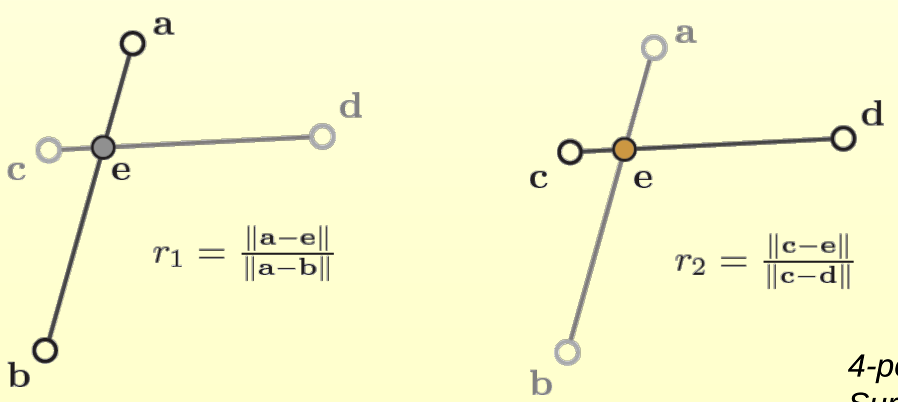
Guess and verify

Random Sampling (RANSAC)

A pair of triples (from **P** and **Q**) are enough to determine a **rigid transform**, resulting in $O(n^3)$ RANSAC.

Surprisingly, a special set of 4 points, **congruent sets**, makes the problem simpler leading to $O(n^2)$!

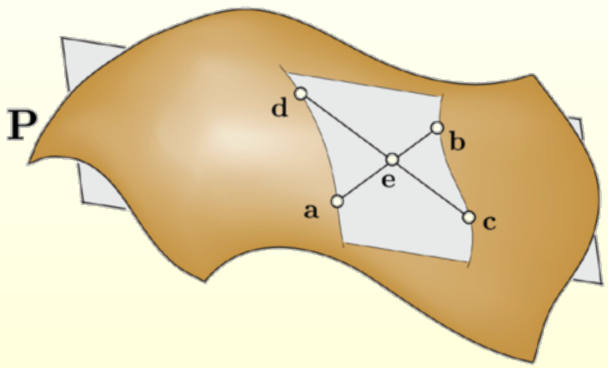
Co-planar points
remain coplanar



4-points Congruent Sets for Robust Surface Registration,
Aiger et al., SIGGRAPH 2008

Method Overview

On the source shape, pick 4 (approx.) coplanar points.



Compute

$$r_1 = \frac{\|a-e\|}{\|a-b\|} \qquad r_2 = \frac{\|d-e\|}{\|d-c\|}$$

For every *pair* (q_1, q_2) of points on the destination compute

$$p_1 = q_1 + r_1(q_2 - q_1)$$

$$p_2 = q_1 + r_2(q_2 - q_1)$$

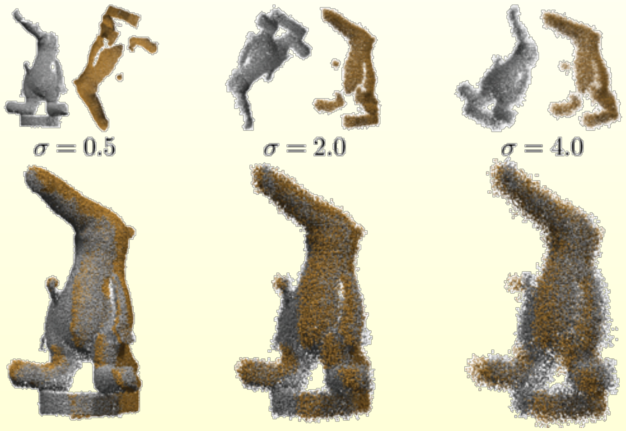
Those pairs (q_1, q_2), (q_3, q_4) for which ${p_1}_{(q_1,q_2)} = {p_2}_{(q_3,q_4)}$ are a good candidate correspondence for (a, b, c, d).

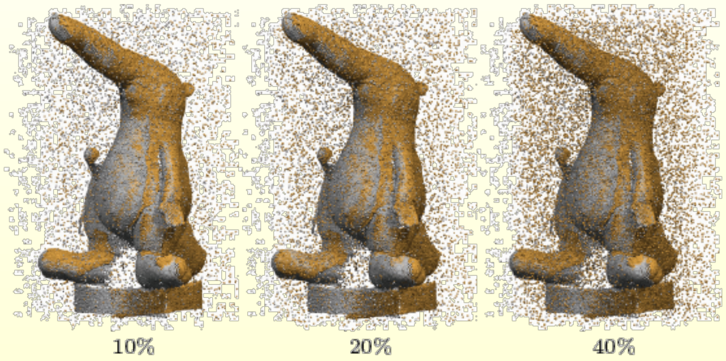
Under mild assumptions the procedure runs in $O(n^2)$ time.

4-points Congruent Sets for Robust Surface Registration,
Aiger et al., SIGGRAPH 2008

Method Overview

Can pick a few base points for partial matching.





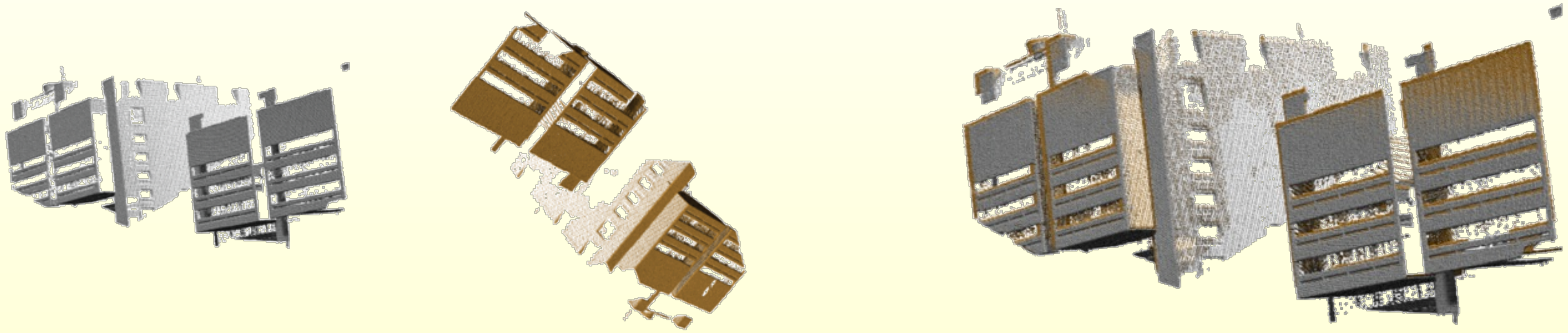
Random sampling
handles noise

and outliers

4-points Congruent Sets for Robust Surface Registration,
Aiger et al., SIGGRAPH 2008

Method Overview

Can pick a few base points for partial matching.



Partial matches

4-points Congruent Sets for Robust Surface Registration,
Aiger et al., SIGGRAPH 2008

Global Matching – Approaches

Several classes of approaches:

1. Exhaustive Search
2. Normalization
3. Random Sampling
4. Invariant Features

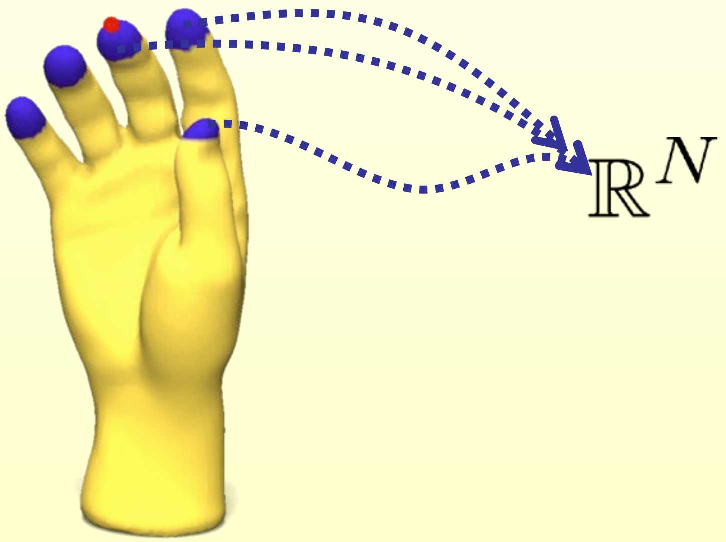
Global Matching – Invariant Features

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

Conflicting interests – invariance vs. informativeness.

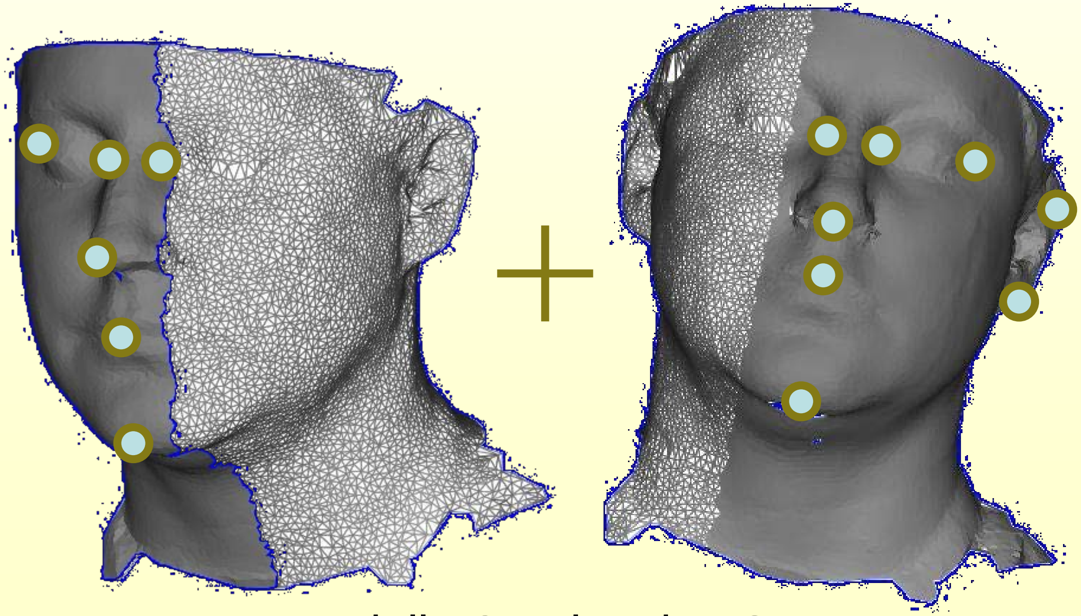
The most common pipeline:

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



Matching Using Feature Points

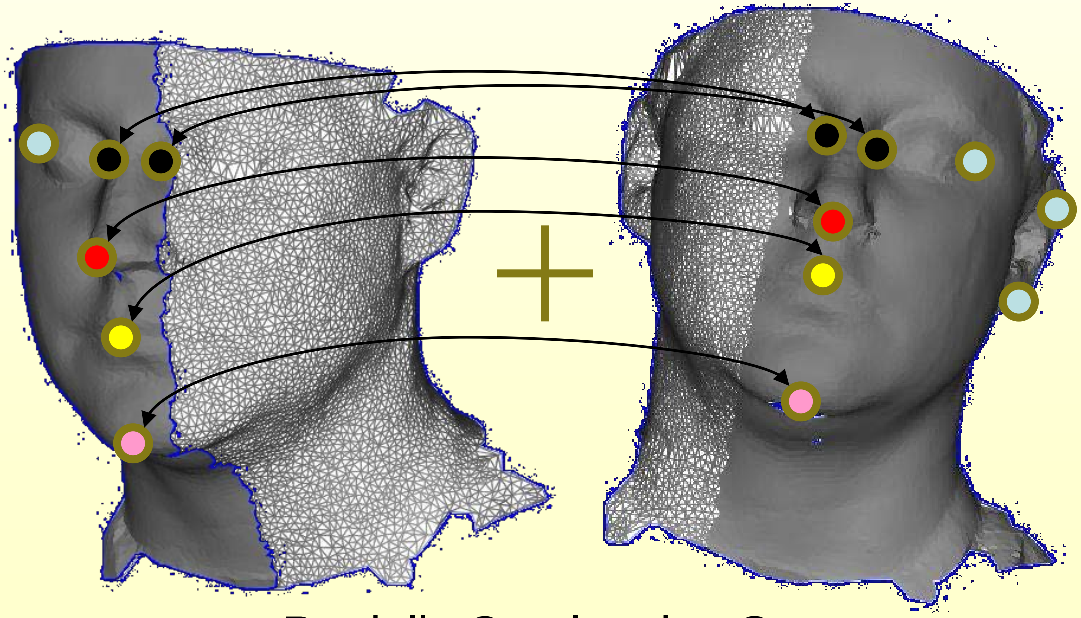
1. Find feature points on the two scans (we'll come back to that issue)



Partially Overlapping Scans

Approach

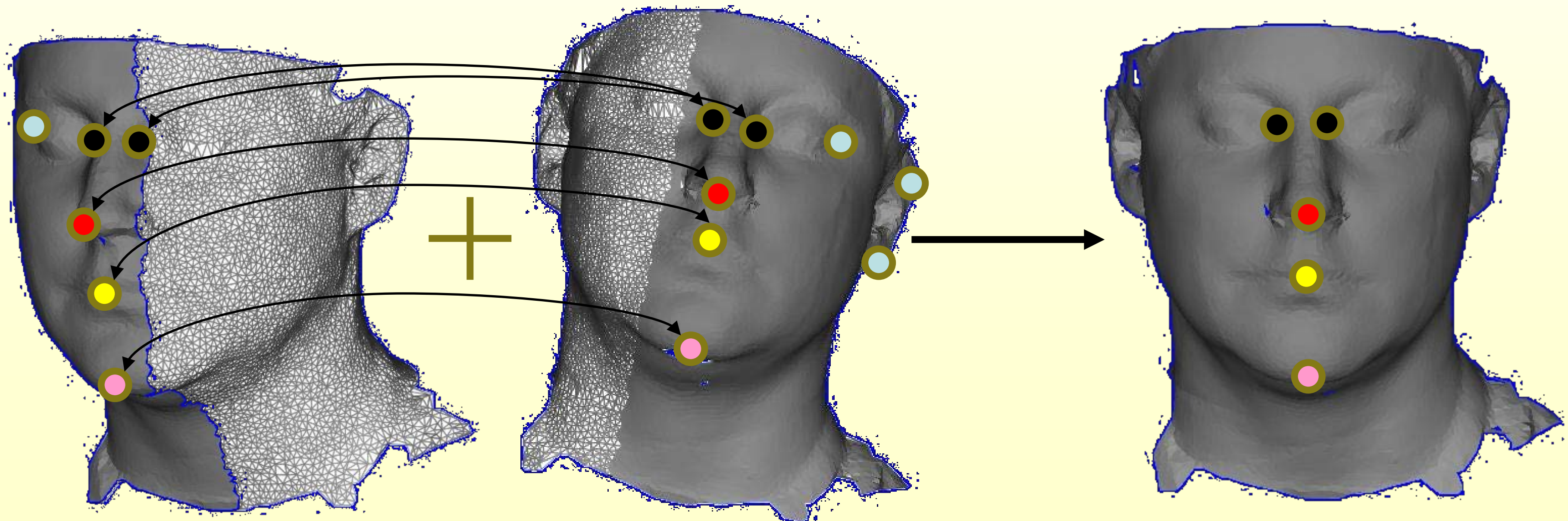
1. (Find feature points on the two scans)
2. Establish correspondences



Partially Overlapping Scans

Approach

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



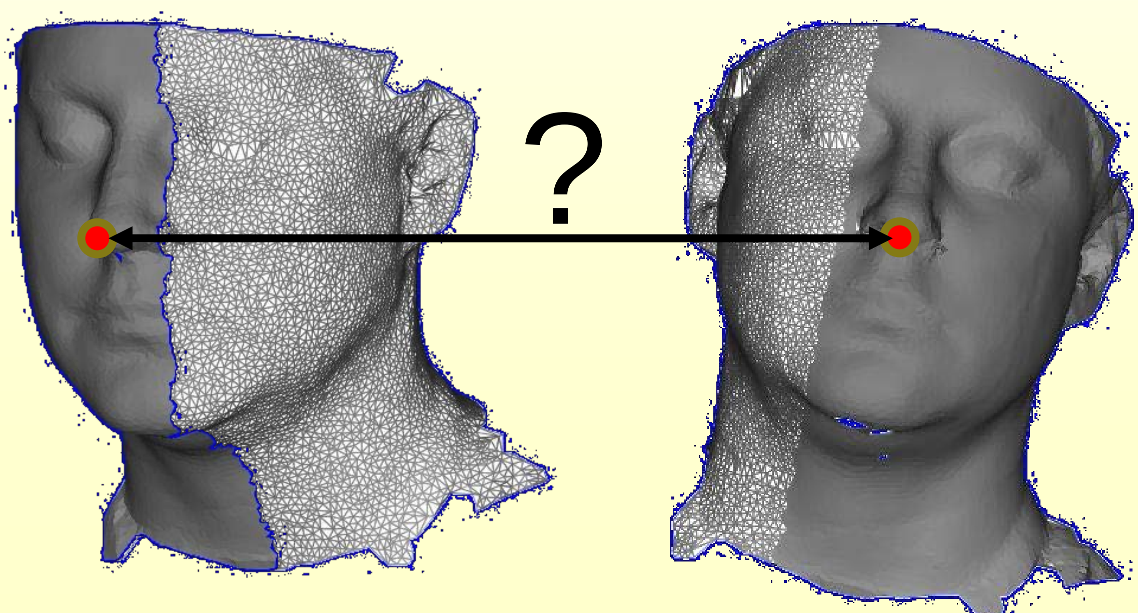
Partially Overlapping Scans

Aligned Scans

Correspondence

Goal:

Identify when two points on different scans represent the same feature

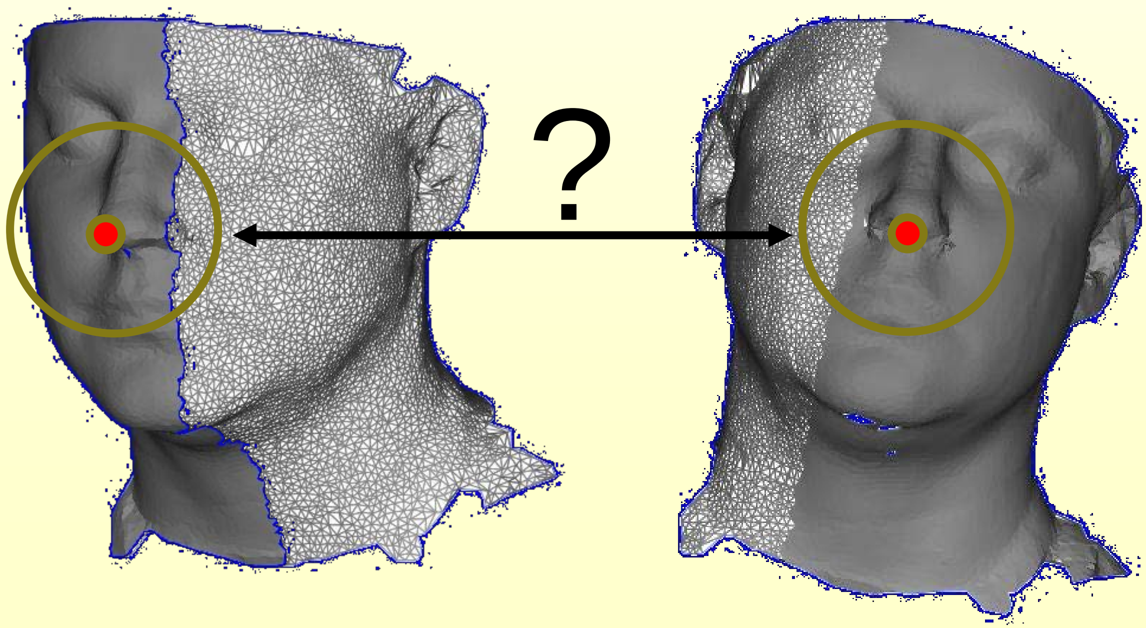


Correspondence

<u>Goal</u>:

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

Are the surrounding regions similar?

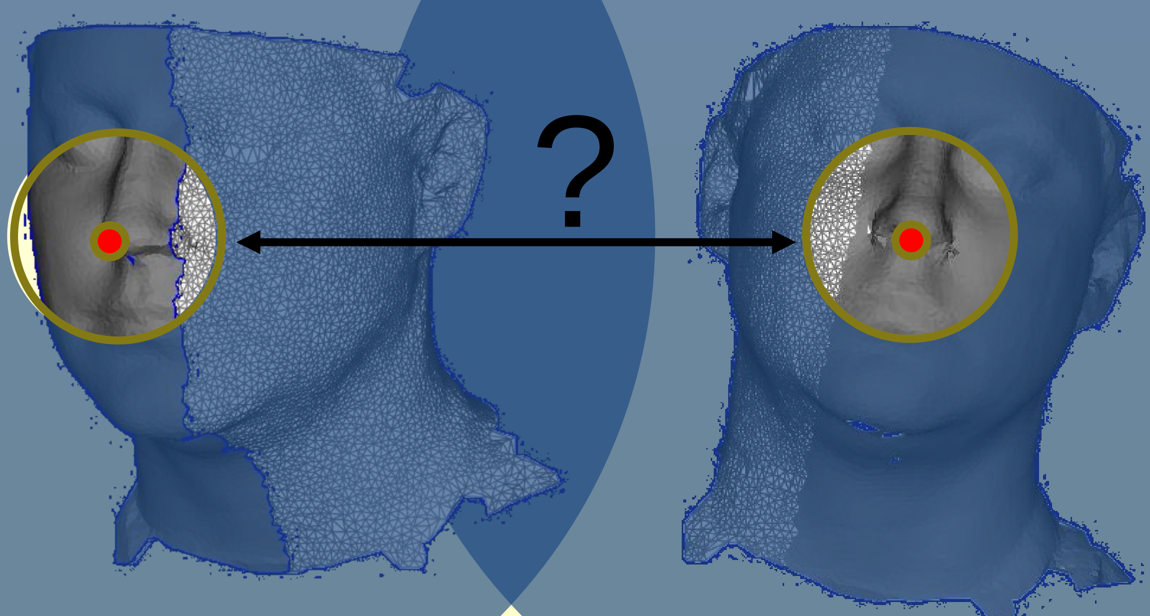


Correspondence

Goal:

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

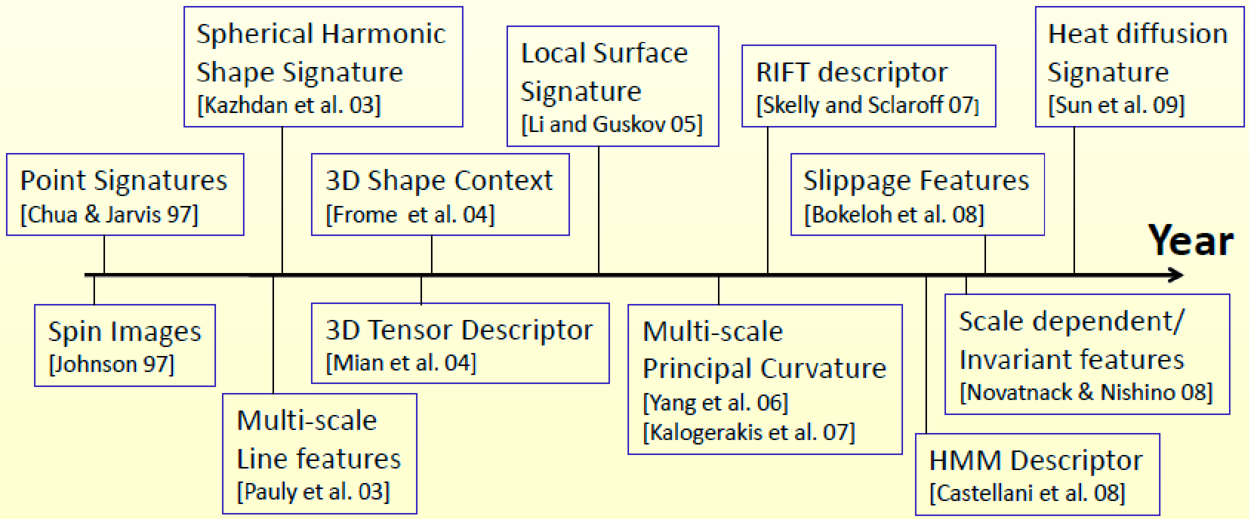
Are the surrounding regions similar?



Main Question

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

A large variety of *descriptors* have been suggested.



To give an example, we describe two.

table by Will Chang

Spin Images

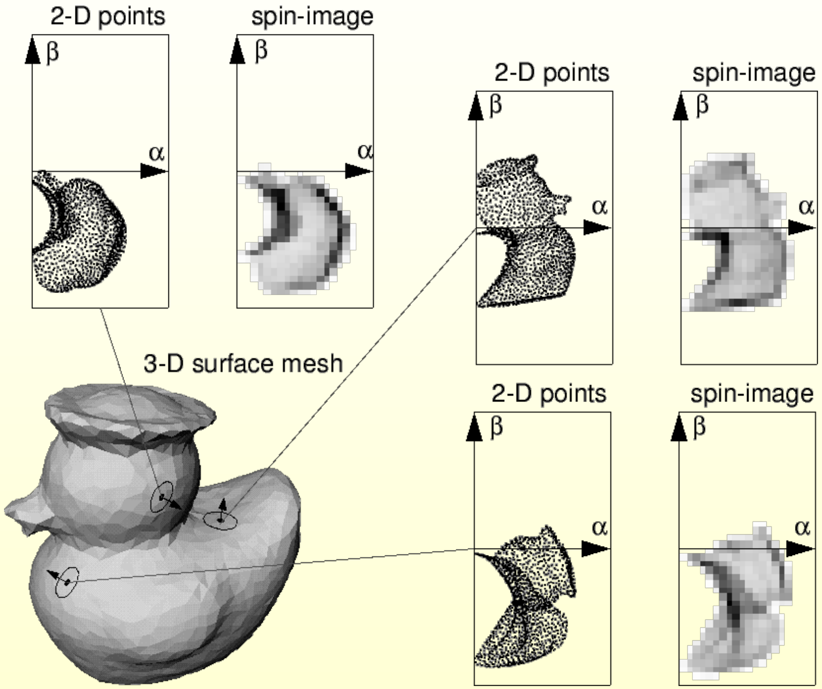
Creates an image associated with a neighborhood of a point.

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

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

β distance to tangent plane

α distance to normal line

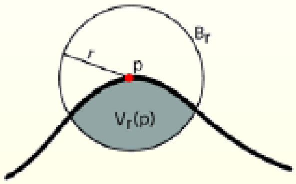


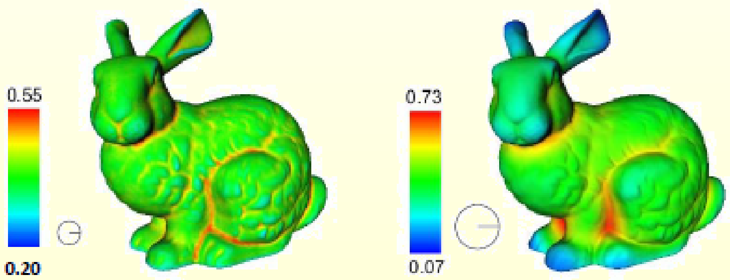
Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes
Johnson et al, PAMI 99

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

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

Robust Global Registration,
Gelfand et al. 2005

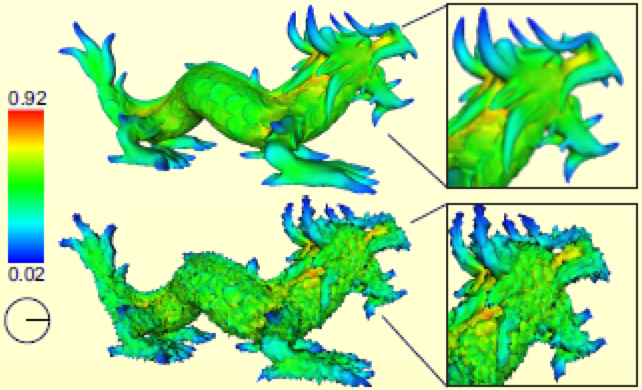
Feature Based Methods

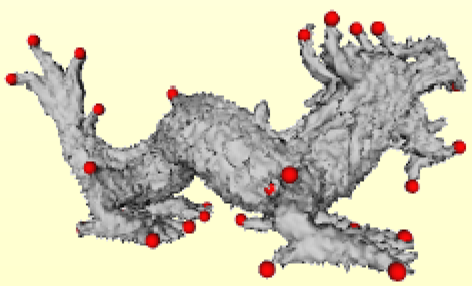
Once we have a feature descriptor, we can find the most *unusual* one: feature detection.

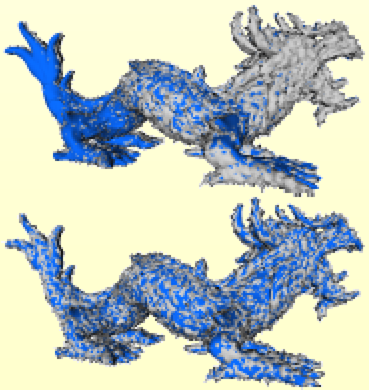
Establish correspondences by first finding *reliable* ones.

Propagate the matches everywhere.

To backtrack use branch-and bound.







Robust Global Registration, Gelfand et al. 2005

Method Taxonomy

Local vs. Global

refinement (e.g. ICP) | alignment (search)

Rigid vs. Deformable

rotation, translation | general deformation

Pair vs. Collection

two shapes | multiple shapes

Conclusion

- Shape matching is an active area of research.
- Local rigid matching works well. Many approaches to global matching. Works well, depending on the domain.
- Non-rigid matching is much harder. Isometric deformation model is common and useful, but limiting.
- Research problems: other deformation models, consistent matching with many shapes, robust deformable matching.