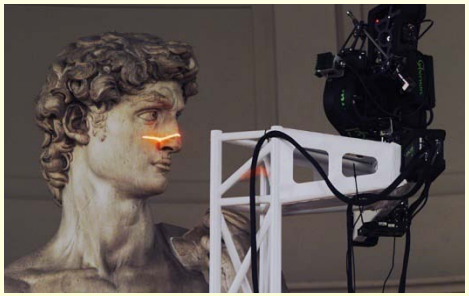
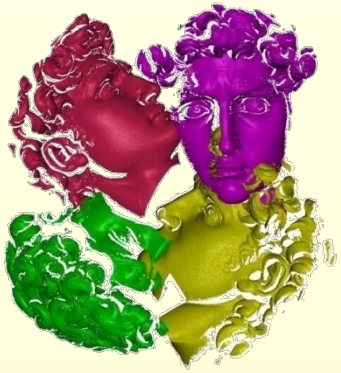
CS164: Shape Registration

Leonidas Guibas
Computer Science Dept.
Stanford University

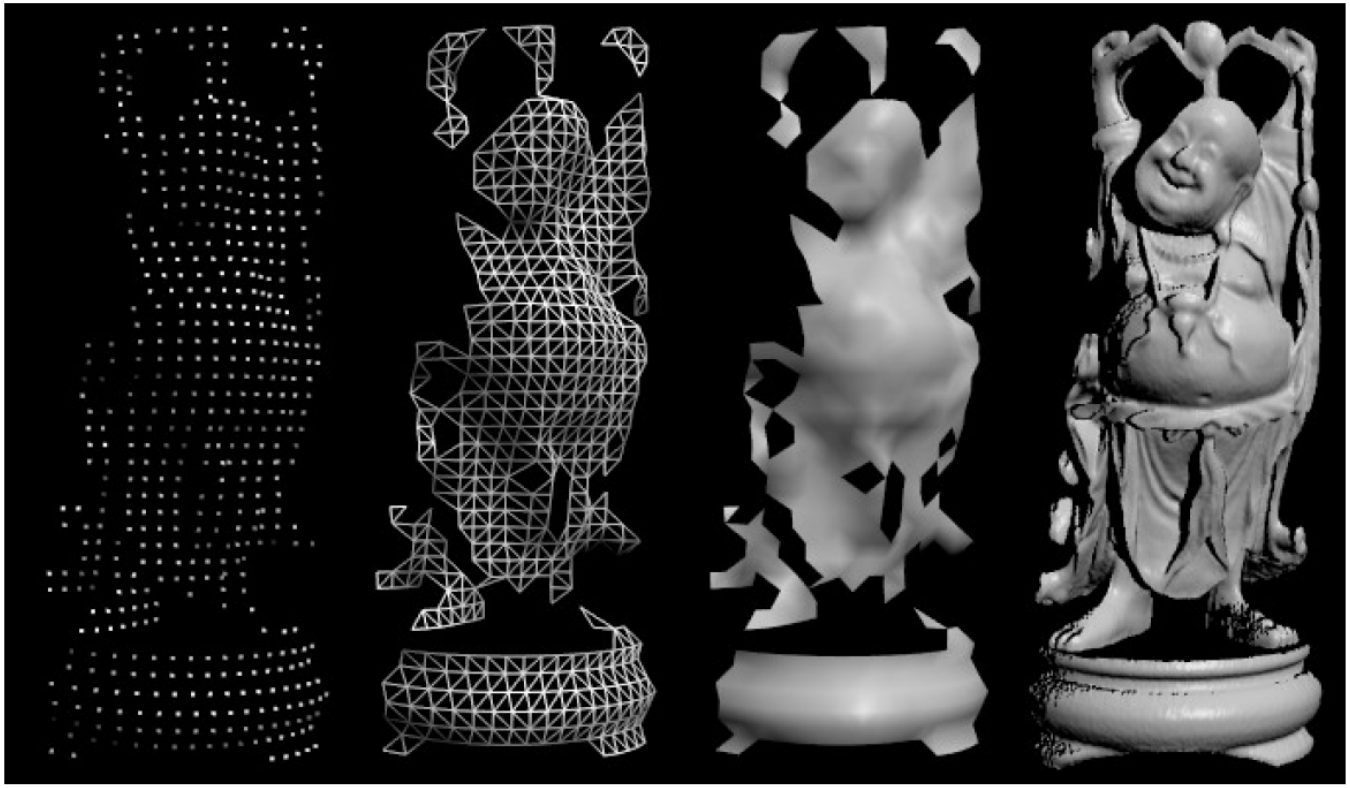
Aligning and Matching (Partial or Entire) Shapes

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





Range Image Registration



From point clouds to triangle meshes …

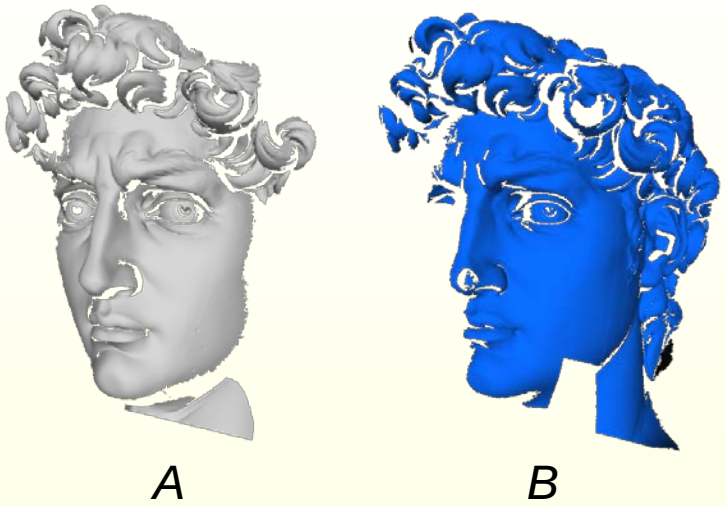
The Registration Problem

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*



A *B*

Distance or Similarity of Shapes

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.

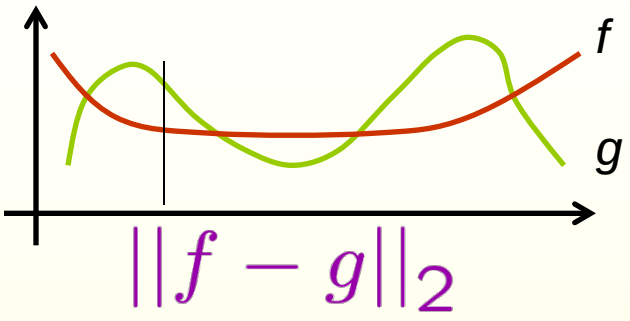
As another example, shape registration and matching is very important in modern structural biology

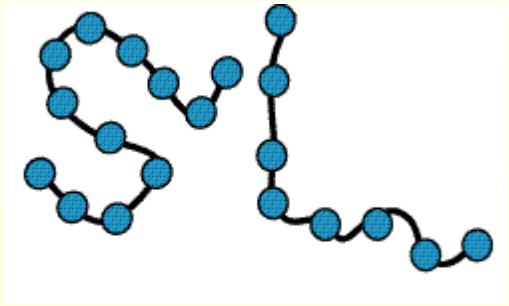
Human (red) and fly (yellow) thierodoxins, compared

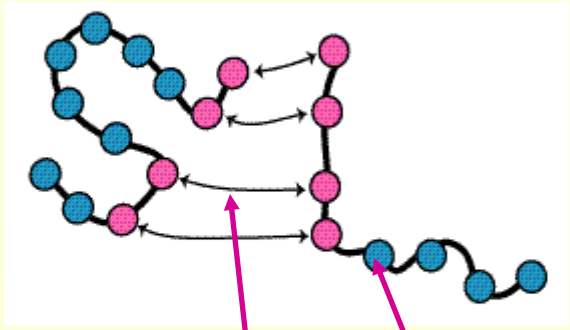
Issues about Similarity Measures

- We are 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 or similarity a metric?

Not for partial matches



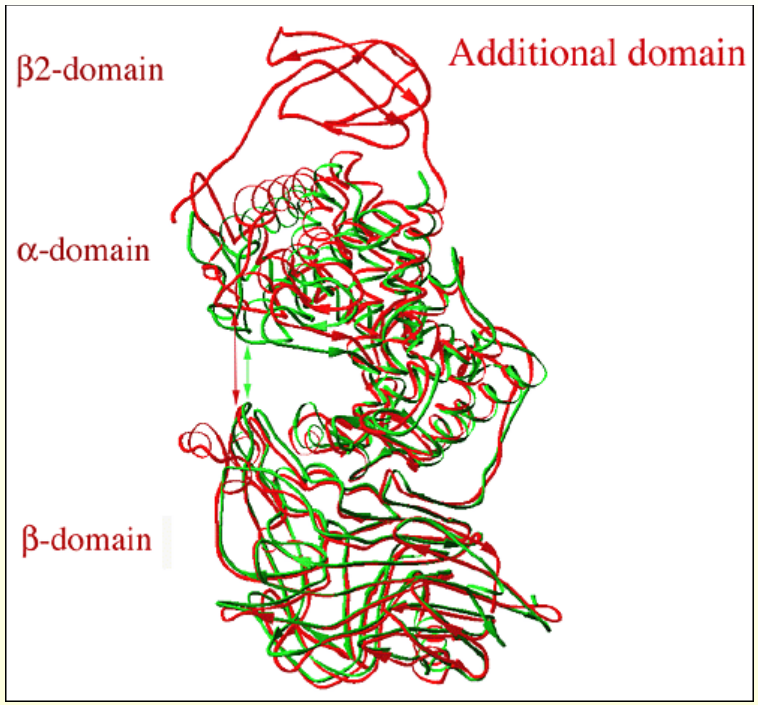


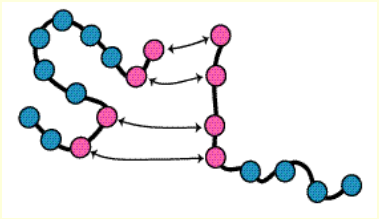


$$\delta(A,C) \leq \delta(A,B) + \delta(B,C)$$

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 (dis-) similarity measure δ



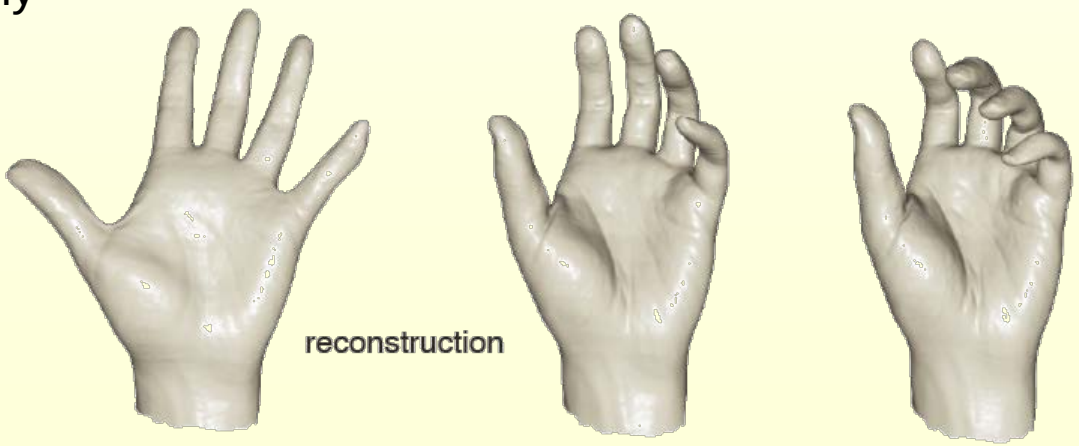


In computing the score, how do we

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)
  - More general transforms require more degrees of freedom. When shape deformations are allowed, the degrees of freedom can grow very rapidly



Shape Alignment and Registration

Simplest Case: Rigid Alignment, Given Correspondences

- We are given two sets of corresponding points x_1, x_2, ..., x_n and y_1, y_2, ..., y_n in E^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 in robotics, graphics, medical image analysis, statistical theories of shape, 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):

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

T R

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

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

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

The Rotation Part

- Define

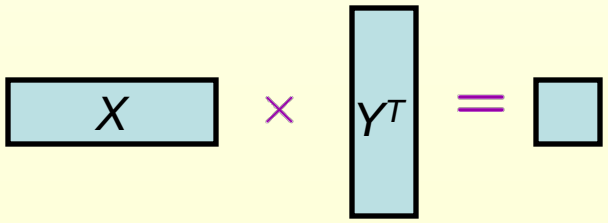
$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

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

$$X = [x_1 - \overline{x}, \ldots x_n - \overline{x}]^T$$

$$Y = [y_1 - \overline{y}, \ldots y_n - \overline{y}]^T$$

- 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

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

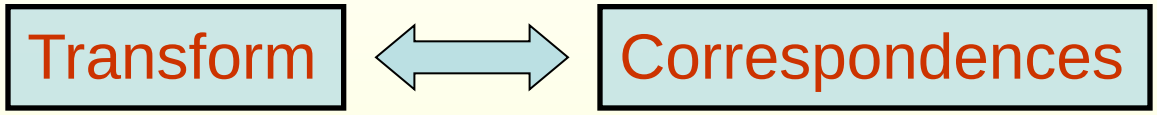
- Then $\boxed{R = USV^T}$

*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)

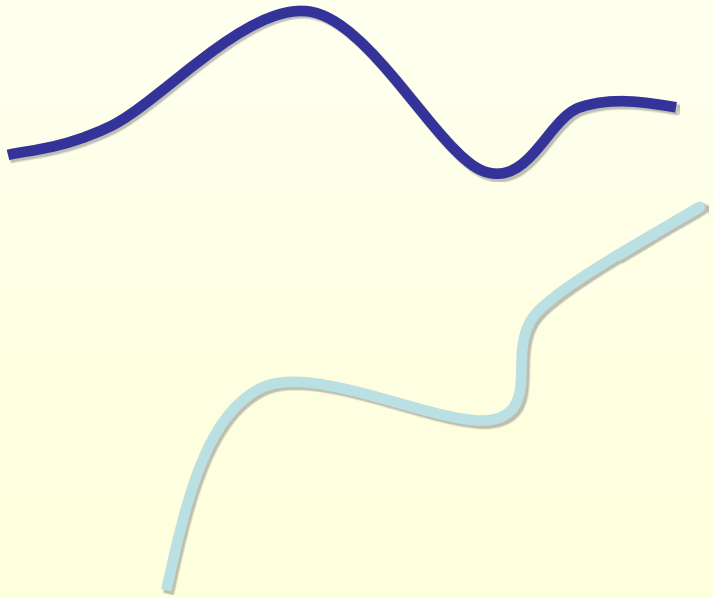


Guess one, estimate the other, and *iterate!*

EM like

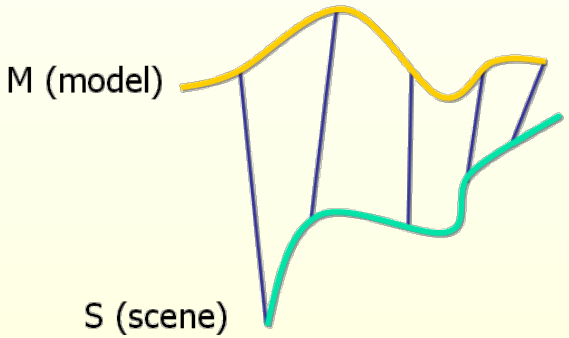
- Correspondences from proximity (Iterated Closest Pair)
- Correspondences from local shape descriptors (Shape Features)
- Transform from voting schemes (Geometric Hashing)
- Combinations

Aligning 3D Data



Point Set Alignment

- Let *M* be a model point set (geometric model)
- Let *S* be a scene point set (sensor data)



In the simplest setting:

1. $N_M = N_S$
2. Each point S_i corresponds to M_i

MSE Objective Function

The MSE objective function is:

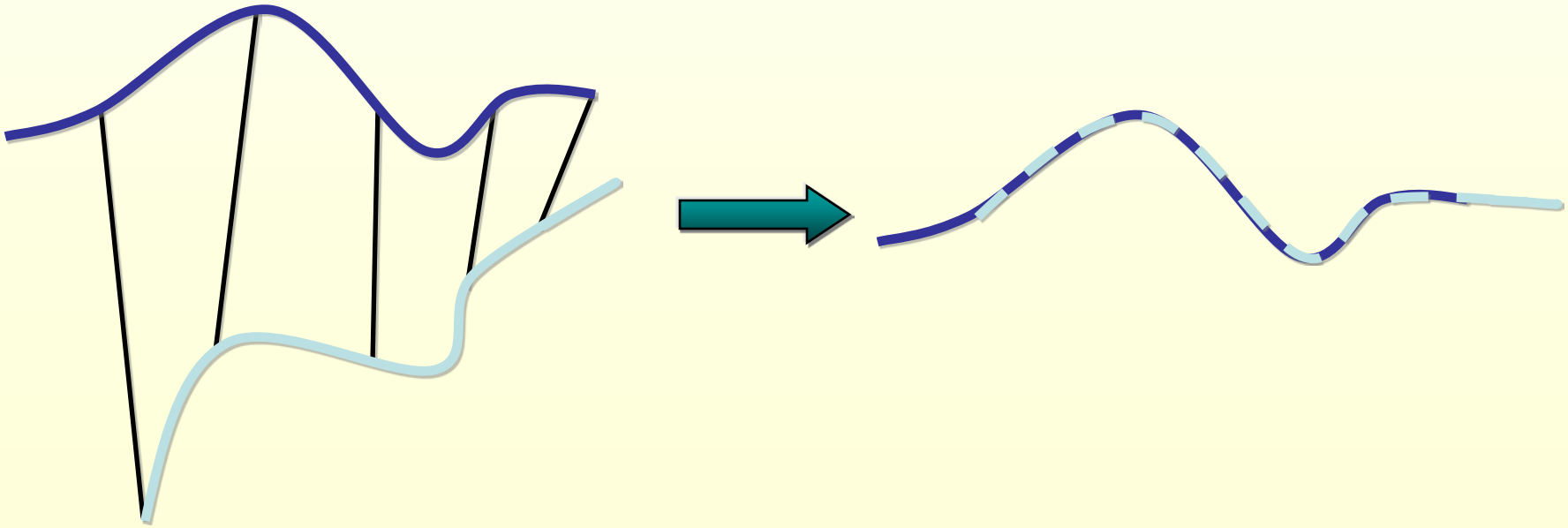
$$f(R,T) = \frac{1}{N_S} \sum_{i=1}^{N_S} \left\| m_i - Rot(s_i) - Trans \right\|^2$$

The optimal alignment is given by:

$$(rot, trans, d_{mse}) = \Phi(M, S)$$

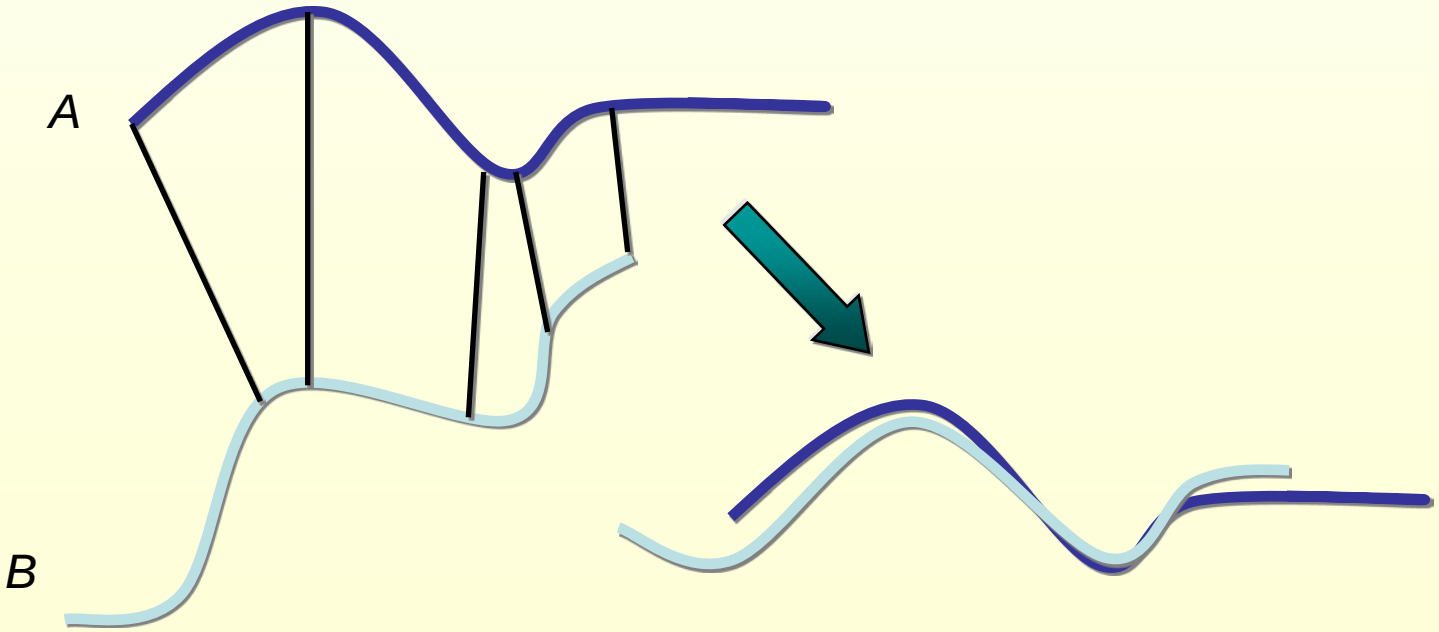
Aligning 3D Data

- As we saw, if correct correspondences are known, we can find optimal rotation/translation in essentially closed form



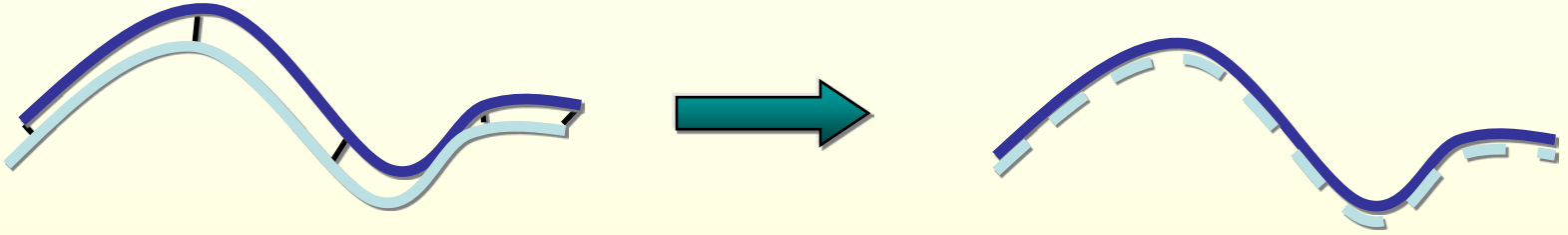
Aligning 3D Data

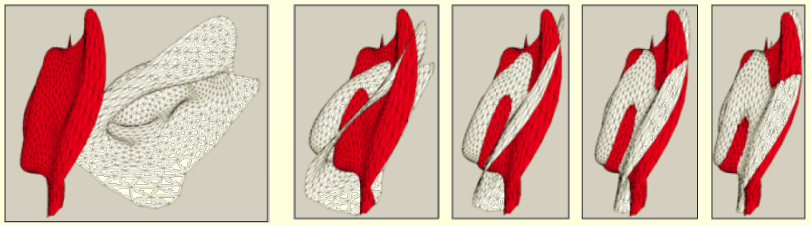
- How to find correspondences: User input? Feature detection? Local shape signatures?
- When *A* and *B* are partial scans of the same stationary object, use the simplest alternative: assume closest points correspond

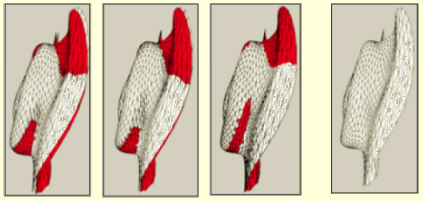


Aligning 3D Data

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







Successive iterations bring the objects closer together

Closest Points

- Given two points r_1 and r_2 , their Euclidean distance is:

$$d(r_1, r_2) = \|r_1 - r_2\| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

- Given a point r_1 and set of points A, the Euclidean distance of r_1 to A is:

$$d(r_1, A) = \min_{i \in 1..n} d(r_1, a_i)$$

Finding Matches

- The scene shape *S* is aligned to be in the best alignment with the model shape *M*
- The distance of each point *s* of the scene from the model is :

$$d(s, M) = \min_{m \in M} \|m - s\|$$

Finding Matches

$$d(s, M) = \min_{m \in M} \|m - s\| = d(s, y)$$

$$y \in M$$

$$Y = C(S, M)$$

$$Y \subseteq M$$

C – the closest point operator

Y – the set of closest points to S

Finding Matches

- Finding each match is performed in O(N_M) time, in the worst case.
- Given *Y* we can calculate optimal alignment

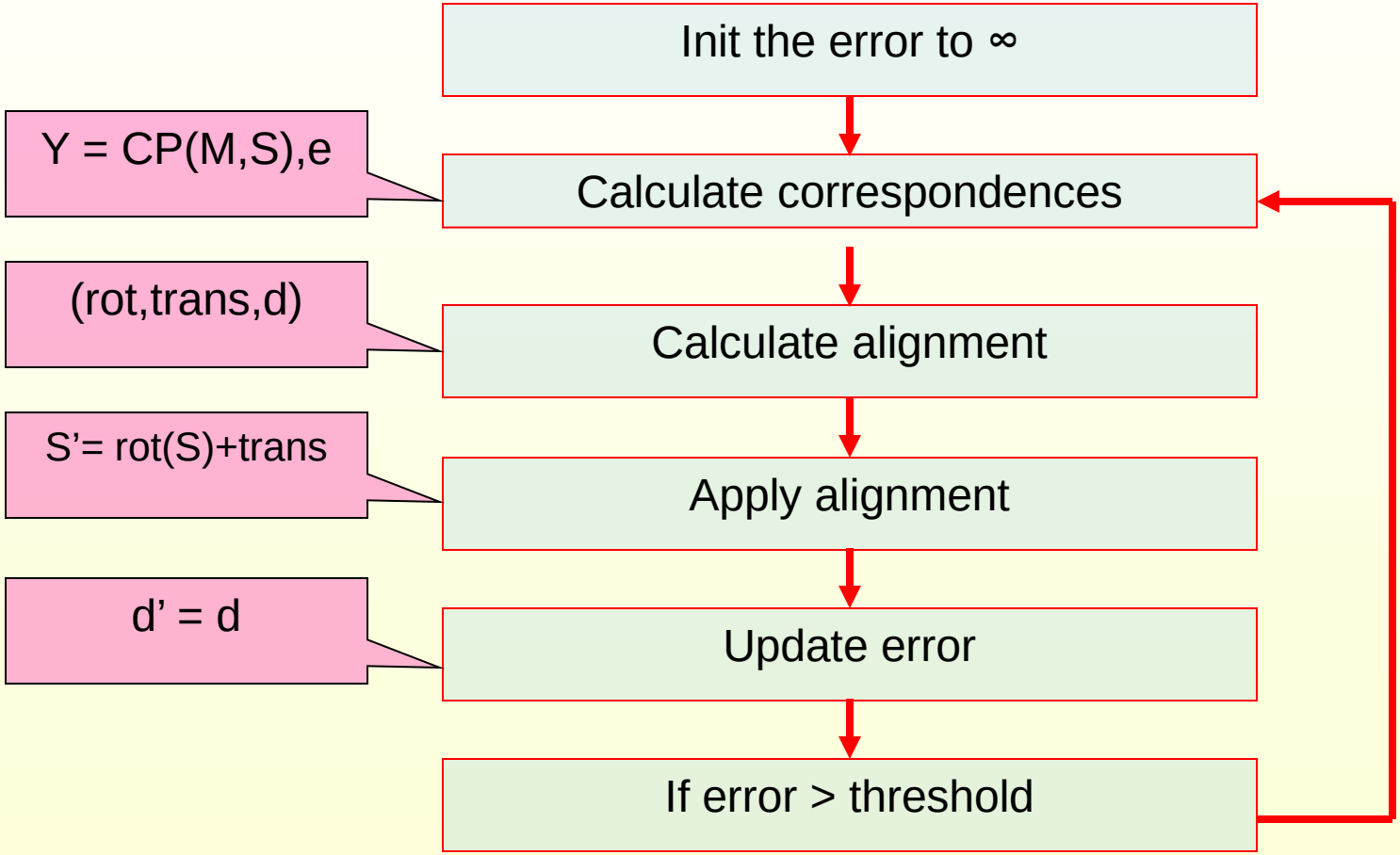
$$(rot, trans, d) = \Phi(S, Y)$$

- *S* is then updated to be :

$$S_{new} = rot(S) + trans$$

The Iterated Closest Pair (ICP) Algorithm

[Besl & McKay, ‘92]



The Algorithm in Code

```
function ICP(Scene,Model)
begin
E' ← + ∞;
(Rot,Trans) ← In Initialize-Alignment(Scene,Model);
repeat
    E ← E';
    Aligned-Scene ← Apply-Alignment(Scene,Rot,Trans);
    Pairs ← Return-Closest-Pairs(Aligned-Scene,Model);
    (Rot,Trans,E') ← Update-Alignment(Scene,Model,Pairs,Rot,Trans);
Until |E'- E|  < Threshold
return (Rot,Trans);
end
```

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

Convergence Theorem

- Correspondence error :

$$e_k = \frac{1}{N_S} \sum_{i=1}^{N_S} \left\| y_{ik} - s_{ik} \right\|^2$$

- Alignment error:

$$d_k = \frac{1}{N_S} \sum_{i=1}^{N_S} \left\| y_{ik} - Rot_k(s_{io}) - Trans_k \right\|^2$$

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

ICP Variants

- Variants on the following stages of ICP have been proposed:

1. Selecting sample points (from one or both meshes)
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. transformation

Performance of Variants

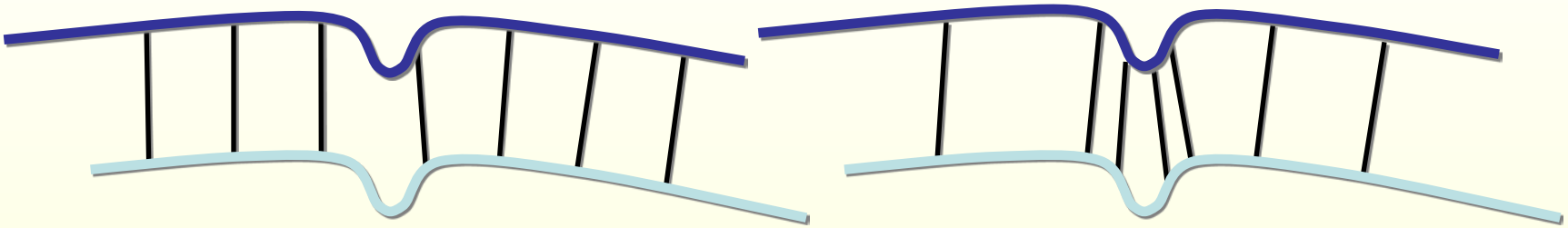
- Can analyze various aspects of performance:
  - Speed of convergence
  - Stability w.r.t. initial position
  - Tolerance of noise and/or outliers
  - Maximum initial misalignment tolerated

ICP Variants

Selecting sample points (from one or both shapes)

- Use all available points [Besl 92]
- Uniform subsampling [Turk 94]
- Random sampling in each iteration [Masuda 96]
- Ensure that samples have normals distributed as uniformly as possible [Rusinkiewicz 01]

Selection of Points



Uniform Sampling

Normal-Space Sampling

Slippage motions and stability of ICP

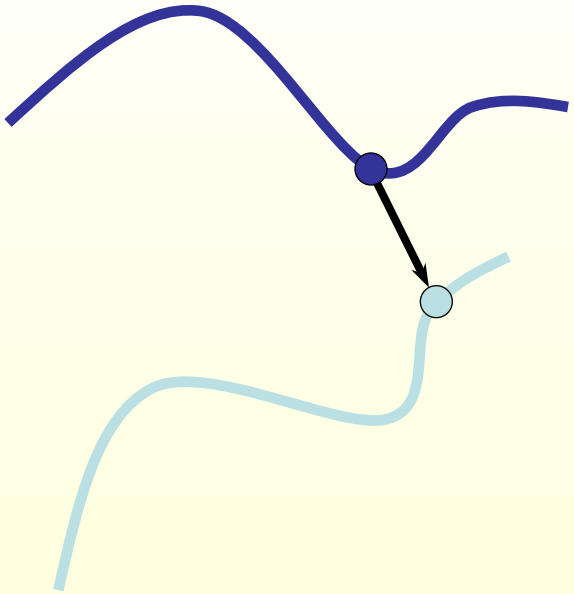
ICP Variants

Matching to points in the other mesh

- Closest point in the other mesh [Besl 92]
- Normal shooting [Chen 91]
- Reverse calibration [Blais 95]
- Restricting matches to compatible points (color, intensity, normals , curvature ..) [Pulli 99]

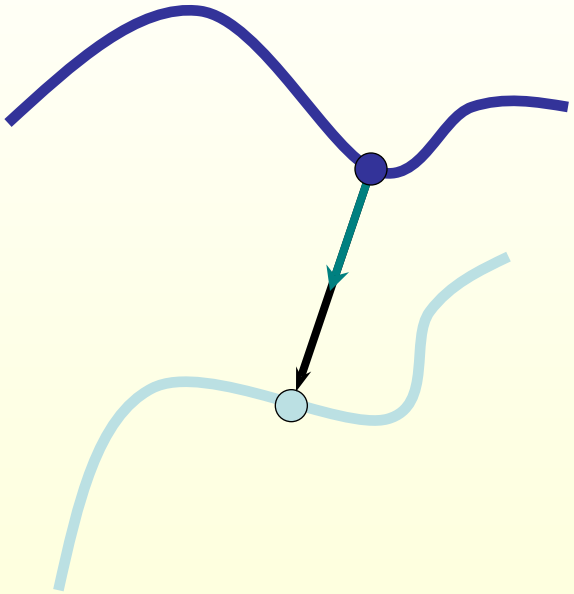
Point Matching

- Closest point :



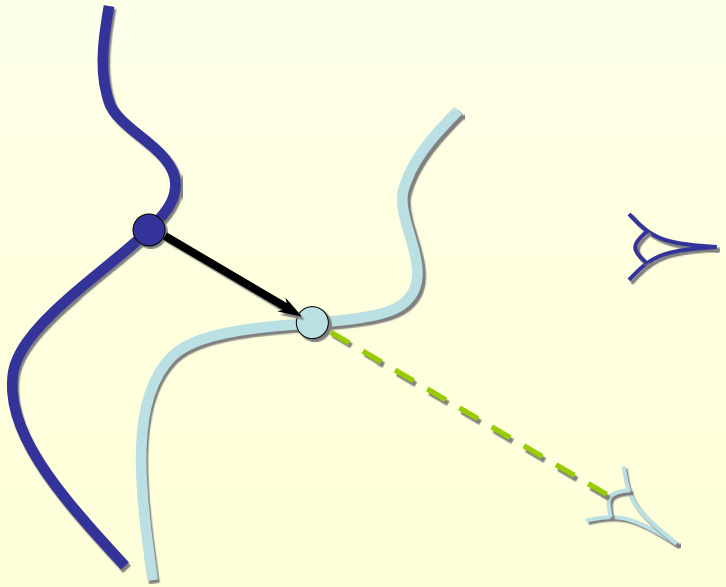
Point Matching

- Normal Shooting



Point Matching

- Projection (reverse calibration)
  Project the sample point onto the destination mesh , from the point of view of the destination mesh’s camera.



Point Matching

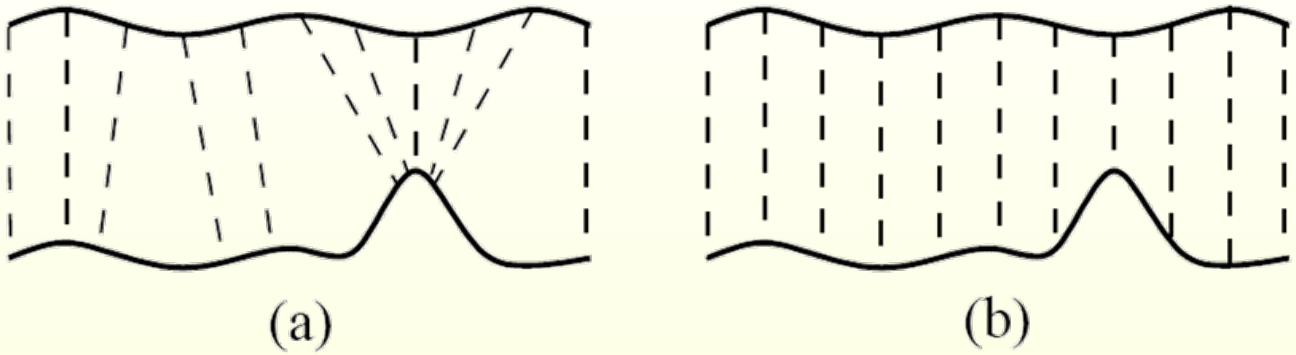


Figure 8: (a) In the presence of noise and outliers, the closest-point matching algorithm potentially generates large numbers of incorrect pairings when the meshes are still relatively far from each other, slowing the rate of convergence. (b) The "projection" matching strategy is less sensitive to the presence of noise.

ICP Variants

Weighting the correspondences

- Constant weight
- Assigning lower weights to pairs with greater point-to-point distance:

$$Weight = 1 - \frac{Dist(p_1, p_2)}{Dist_{\max}}$$

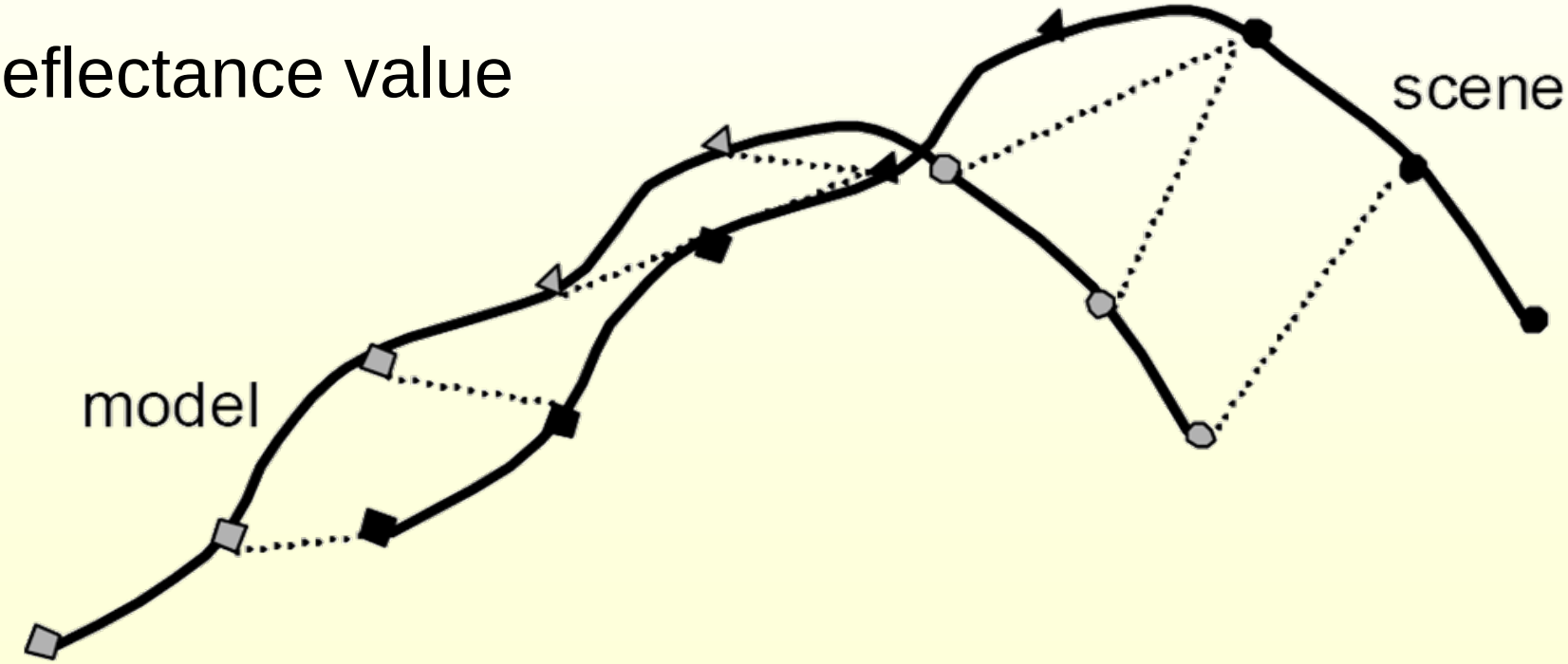
- Weighting based on compatibility of normals:

$$Weight = n_1 \bullet n_2$$

- Scanner uncertainty

Weighting of Pairs

The rectangles and circles indicate the scanner reflectance value



ICP Variants

Rejecting certain (outlier) point pairs

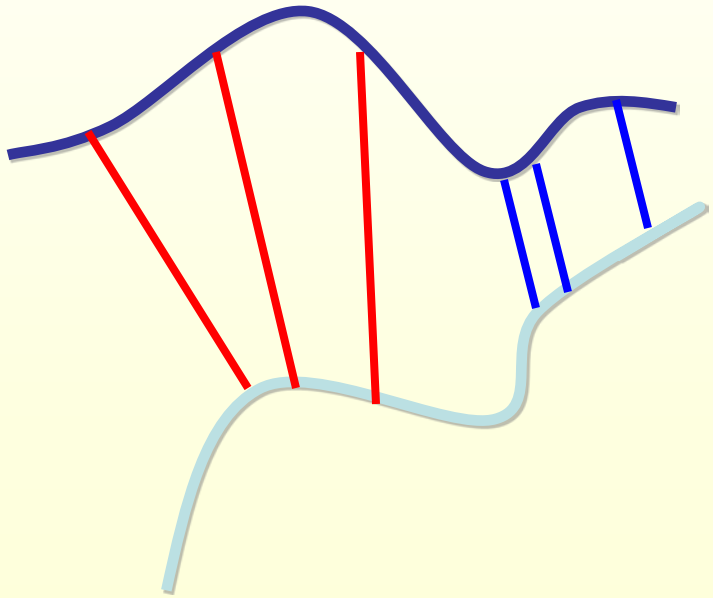
- Corresponding points with point to point distance higher than a given threshold
- Rejection of worst n% pairs based on some metric
- Pairs containing points on end vertices
- Rejection of pairs whose point to point distance is higher than $n*\sigma$
- Rejection of pairs that are not consistent with their neighboring pairs [Dorai 98]:

  (p_1,q_1) , (p_2,q_2) are inconsistent iff

$$\left|Dist(p_1, p_2) - Dist(q_1, q_2)\right| > threshold$$

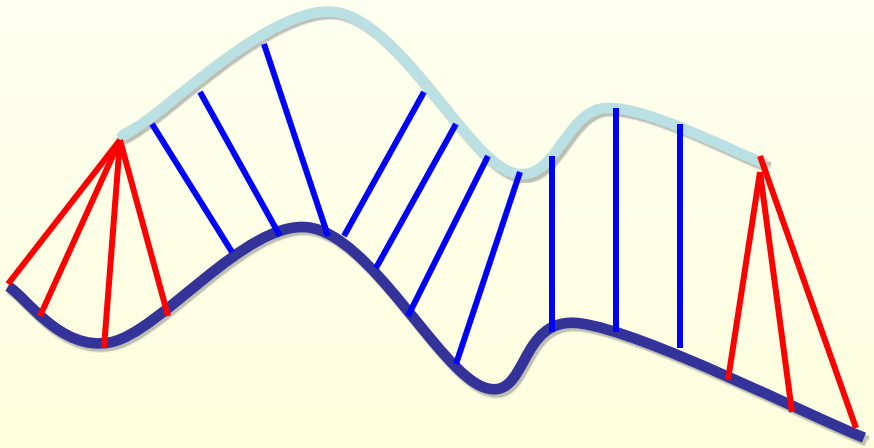
Rejecting Pairs

Distance thresholding



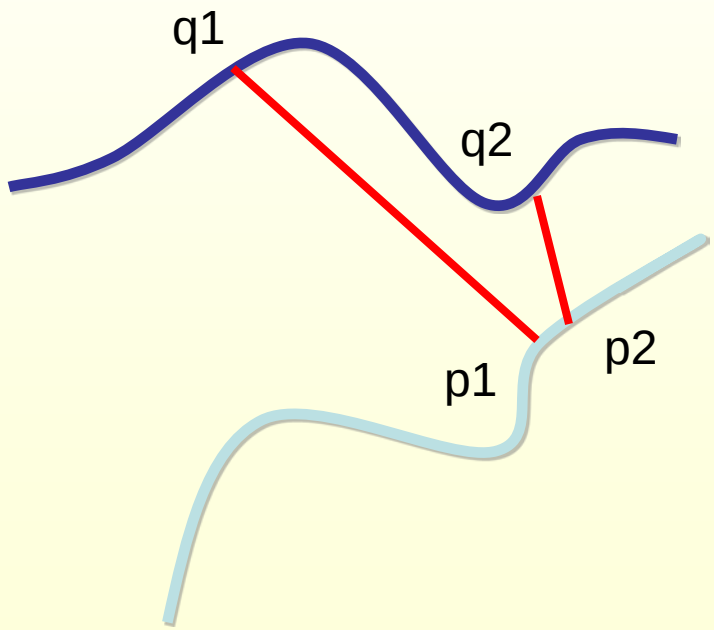
Rejecting Pairs

Points on end vertices



Rejecting Pairs

Inconsistent Pairs



ICP Variants

1. Assigning an error metric to the current transform
2. Minimizing the error metric w.r.t. transformation

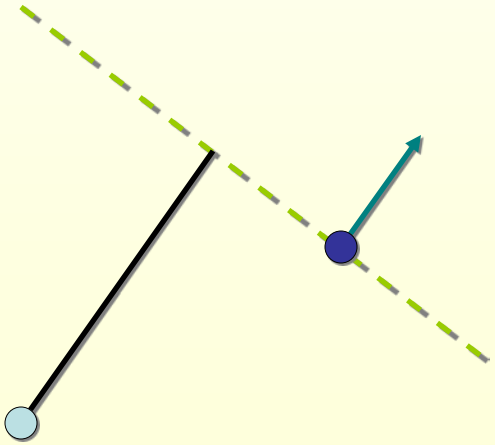
Error Metric and Minimization

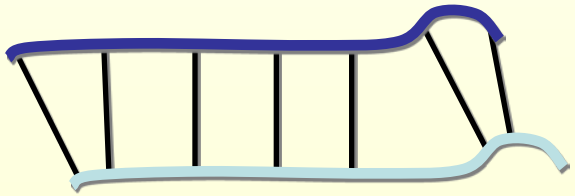
- Sum of squared distances from each sample point to the tangent plane containing the destination point (“Point to Plane”) [Chen 91]

  No closed form solution available

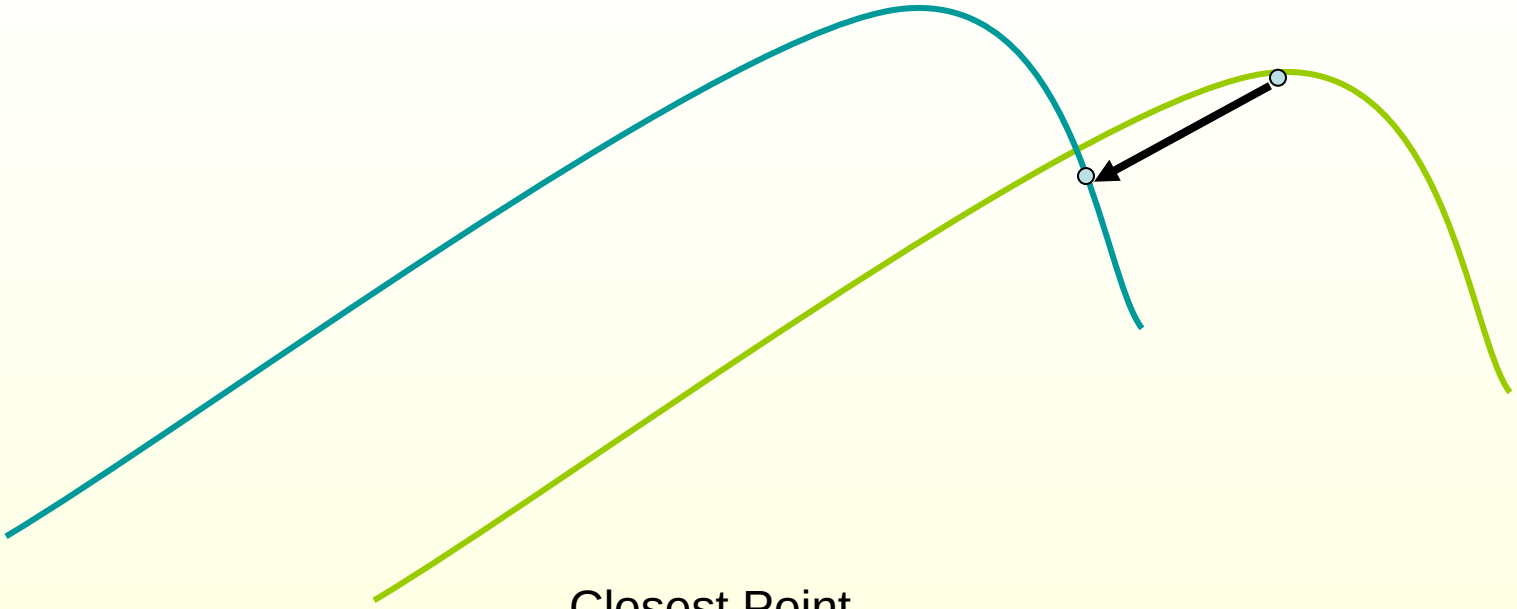
Error Metric and Minimization

- Using point-to-plane distance instead of point-to-point lets flat regions slide along each other [Chen & Medioni 91]

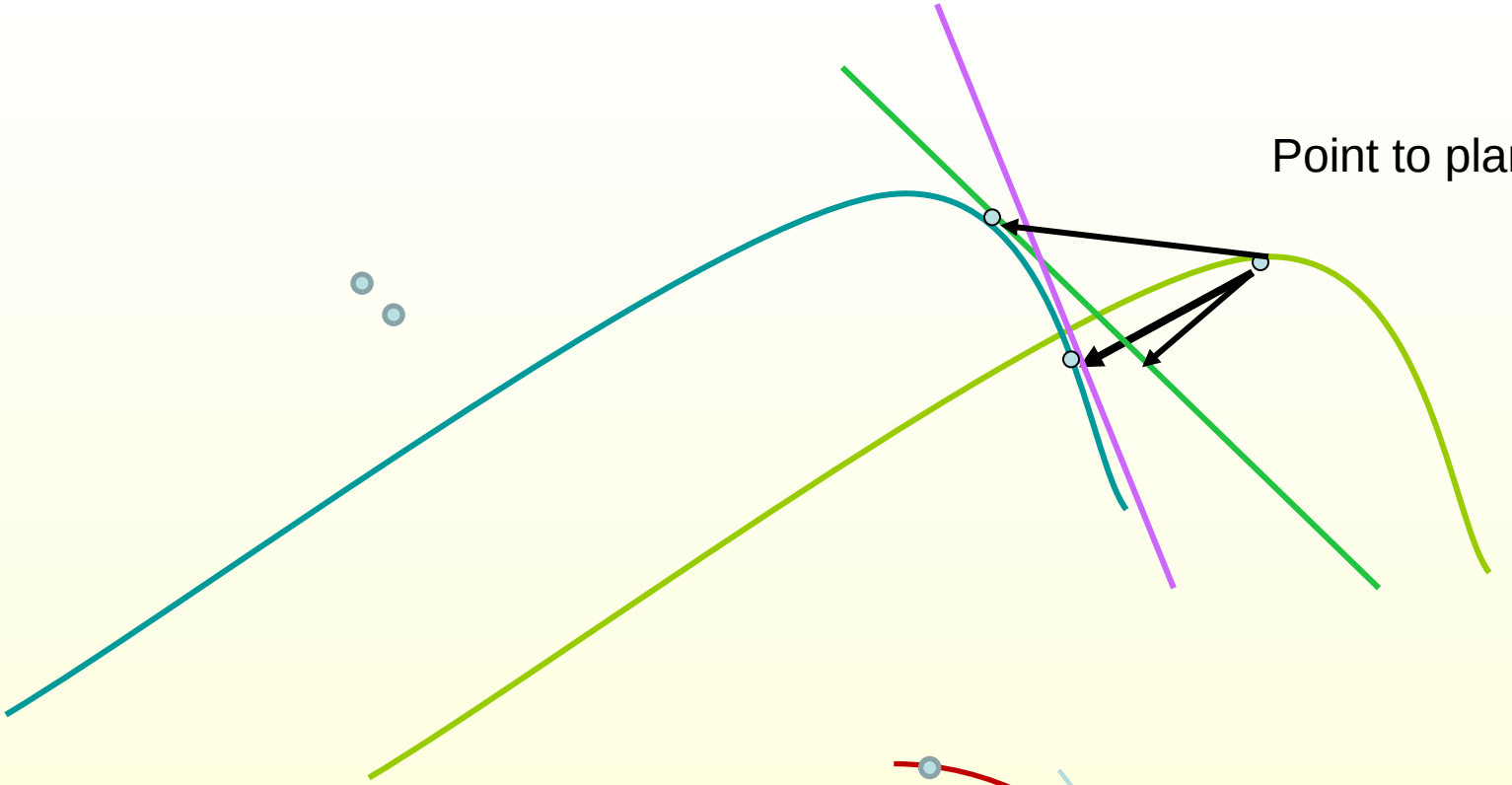


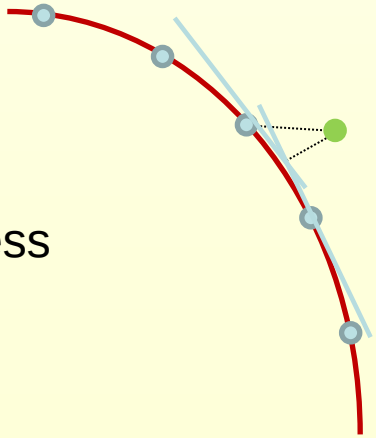


Error Metric and Minimization



Error Metric and Minimization



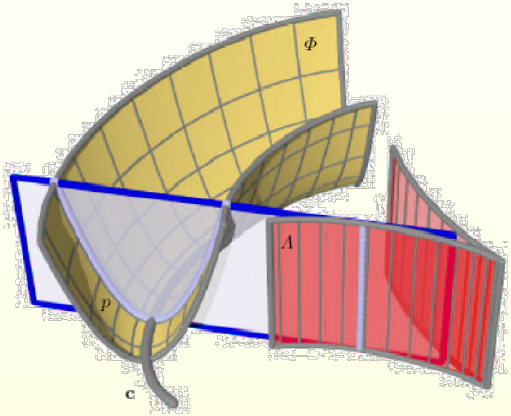


Works well when spacing of shapes is less than the sampling density

ICP Without Correspondences

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

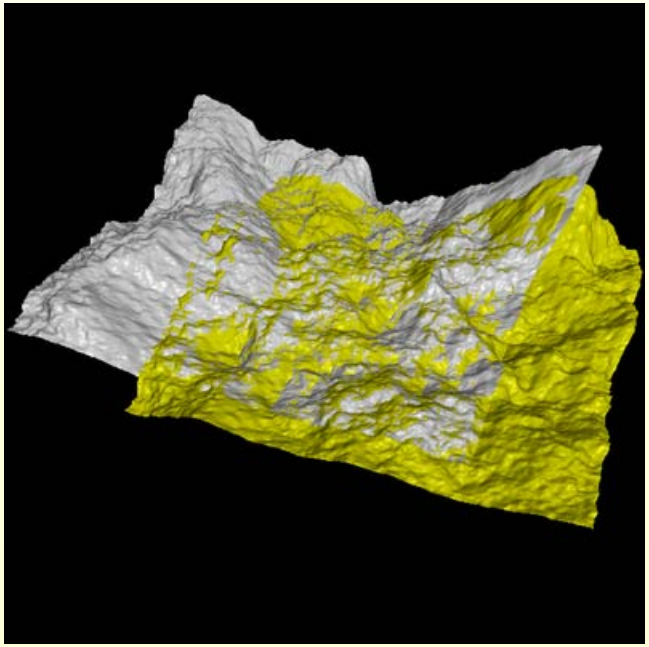
[Pottman & Hofer, 02]



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

curvature

- perform iterative gradient-descent in this field
- point to foot-point distance
  - case $d \to \infty$: classical ICP
  - case d 0: point-to-plane ICP



Shape Models and Shape Variability

Generic Shape Models and Shape Variability

Point Distribution Models

Represent a shape instance by a judiciously chosen set of point (features), each of which is a *k*-dim vector. In the simplest case $\mathbf{p}_i=(x_i,y_i)$ and $k=2$

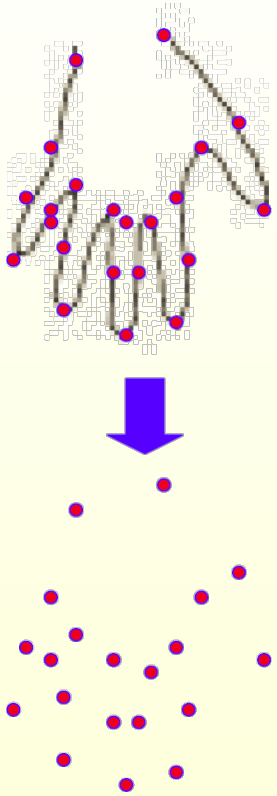
$$\{\mathbf{p}_i : i = 1 \ldots n\}$$

The *n* feature points are stacked into a long vector of length *kn*

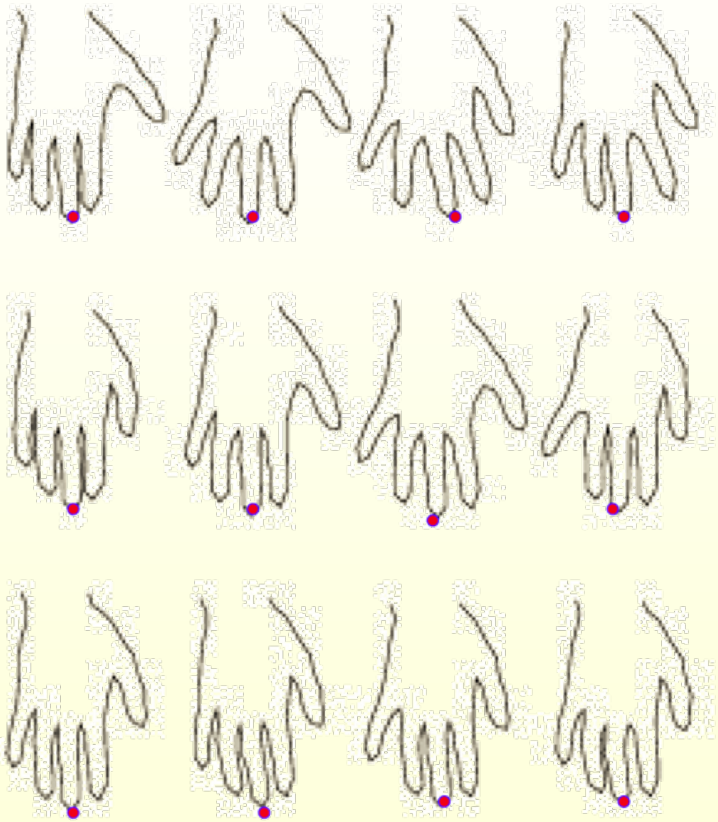
$$\mathbf{q} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]^T$$

Assume that for each *i* and for each of *M* training instances of the shape that the points are in correspondence:

$$\mathbf{p}_i^1, \mathbf{p}_i^2, \ldots, \mathbf{p}_i^M$$



Learning a Shape Family



Here there are M=12 training instances.

One point feature, the 14th, is shown, with correspondences on each instance.

$$\mathbf{p}_{14}^{1}, \mathbf{p}_{14}^{2}, \ldots, \mathbf{p}_{14}^{12}$$

Evidently, if the number of feature points n is large, and the training set size M is also large, this is going to be tedious unless it can be automated…

Aligning Shape Instances

The Procrustes Algorithm is used so that the sum of distances to the mean of each shape is minimised

$$\sum_{i=1..M} (\mathbf{q}_i - \overline{\mathbf{q}})^2$$

1. Translate each shape instance (= stacked vector) $\mathbf{q}_i$ so that its centre of mass is at the origin
2. Choose one example as the mean shape and rescale so that $|\overline{\mathbf{q}}| = 1$
3. Record the first estimate as $\mathbf{q}_0$ to define the default reference frame
4. Align all instances of the shape with the current estimate of the mean shape
5. Re-estimate the mean from the aligned shapes
6. Re-scale and re-iterate the process if necessary

PCA Models of Shape Variability

PCA = Principal Component Analysis

1. Compute the mean of the data

$$\bar{\mathbf{q}} = \frac{1}{M} \sum_{i=1..M} \mathbf{q}_i$$

2. Compute the covariance of the data

$$\mathbf{S} = \frac{1}{M-1} \sum_{i} (\mathbf{q}_i - \bar{\mathbf{q}})(\mathbf{q}_i - \bar{\mathbf{q}})^T$$

3. Compute the eigenvectors $\mathbf{u}_i$ and eigenvalues λ_i of the covariance matrix, sorted in decreasing order of eigenvalue size

4. Remove the small eigenvalues, retaining "most" (eg 98%) of the variation

$$\text{Choose } t \text{ so that } \sum_{i=1}^{t} \lambda_i \geq 0.98 * \sum_{i=1}^{M} \lambda_i$$

PCA Analysis

We have the eigenvectors $\mathbf{u}_i$ sorted in order so that $\lambda_1 \geq \lambda_2 ... \geq \lambda_M$

and have chosen t so that $\sum_{i=1}^{t} \lambda_i \geq 0..98 * \sum_{i=1}^{M} \lambda_i$

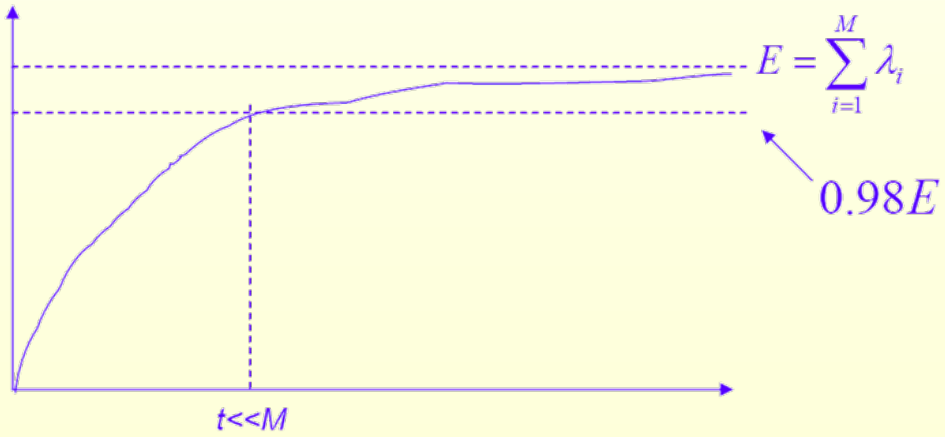
Now define the matrix $\mathbf{U}$ from the top t eigenvectors:

$$\mathbf{U} = [\mathbf{u}_1 \,|\, \mathbf{u}_2 \,|\, ... \,|\, \mathbf{u}_t]$$

Then we approximate any shape instance $\mathbf{x}$ by a t-dimensional vector $\mathbf{b}$

$$\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{U} \cdot \mathbf{b}$$

$$\mathbf{b} = \mathbf{U}^T \cdot (\mathbf{x} - \bar{\mathbf{x}})$$



Shape Approximation

We now ***approximate*** any instance of the shape, including the training instances, by projecting onto the first *t* eigenvectors:

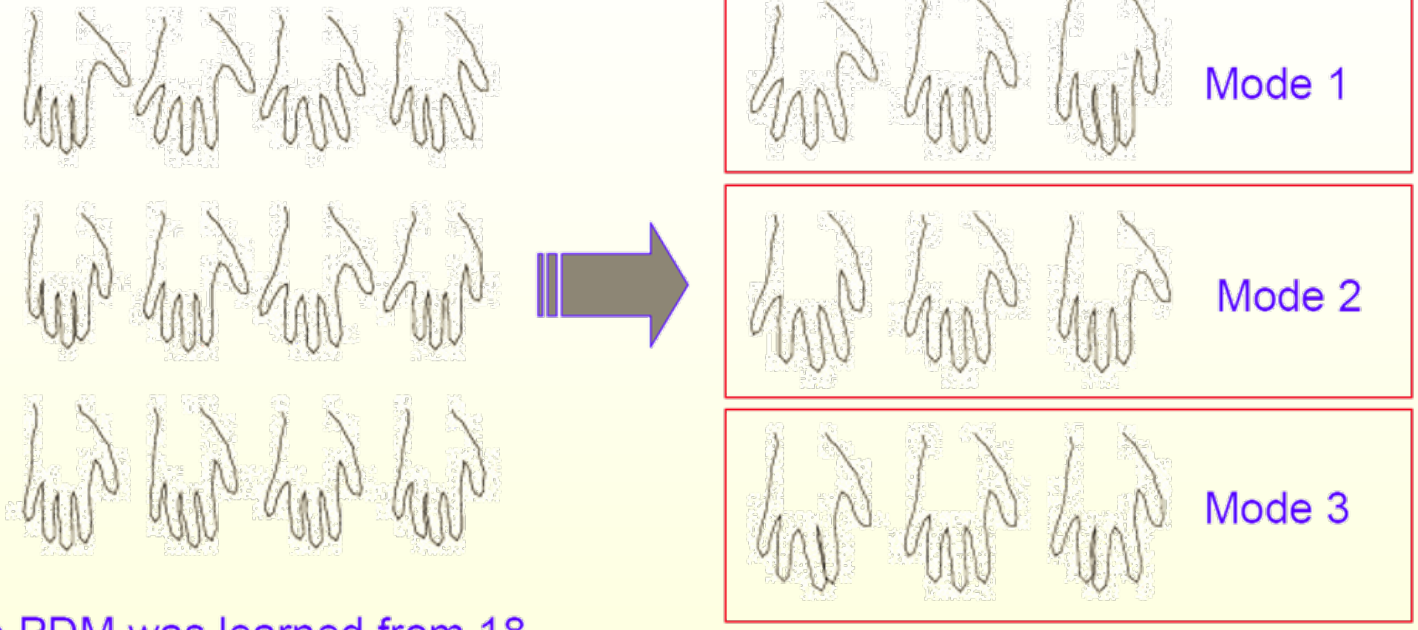
$$\mathbf{q} = \overline{\mathbf{q}} + \sum_{i=1}^{t} b_i \mathbf{u}_i$$

The weight vector **b** is identified as the characteristic of this instance of the shape

$$\mathbf{b} = [b_1, \ldots, b_t]^T$$

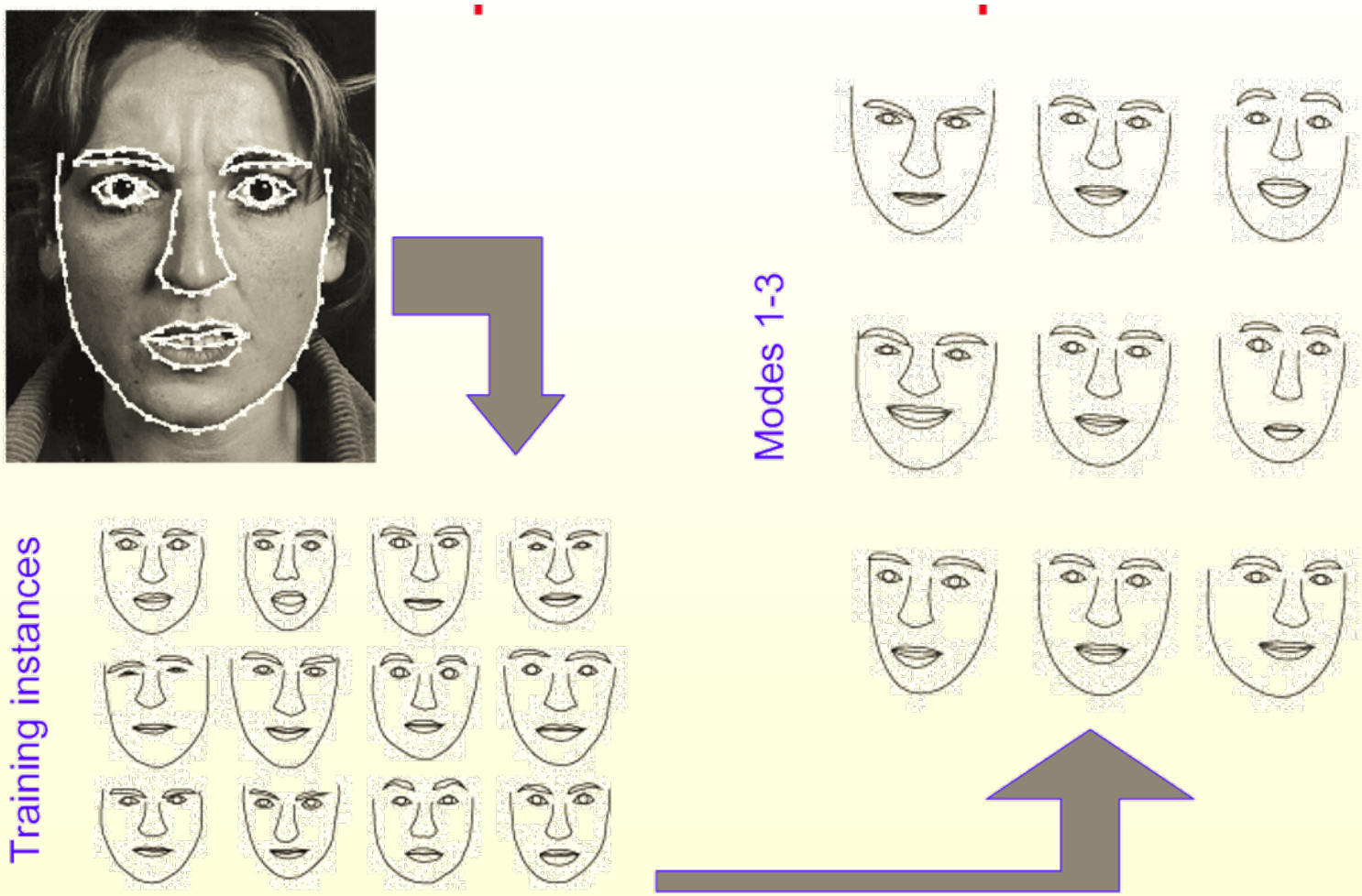
Varying the weights b_i enables us to explore the allowable variations in the shape

Example: Hand Shapes



The PDM was learned from 18 shapes, each comprising 72 points, at finger tips, finger junctions, and equally spaced along the finger sides

Example: Face Shapes



Finding a Shape in an Image

We suppose that we have learned a shape model, comprising the average shape and set of *t* modes. We are presented with a new image and try to fit the shape model to the image: this requires both that we find the appropriate weights **b** that define the model instance and that we find the transformation from shape space to the image, to align the model instance with the image.

If the transform is T and the weight vector $\mathbf{b}$, the instance of shape $(\mathbf{q}, \mathbf{u}_1, \ldots, \mathbf{u}_t)$ in the image is

$$\mathbf{q} = T(\overline{\mathbf{q}} + \sum_{i=1}^{t} b_i \mathbf{u}_i)$$

Typically, *T* is a similarity: translation + rotation + uniform scaling

Fitting Pose and Parameters

Suppose we have identified a set of points **Y** in the image.
Evidently, we can seek to minimise the squared distance:

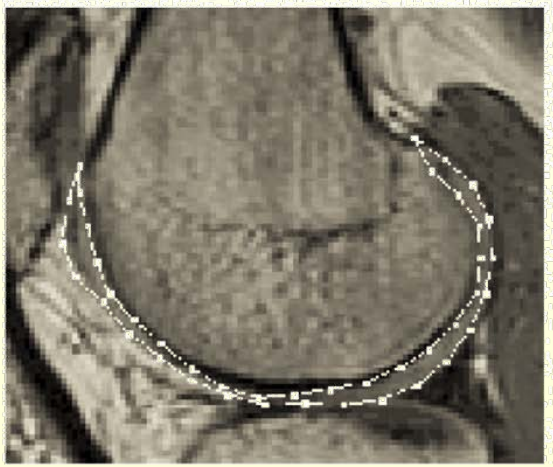
$$|\mathbf{Y} - T(\overline{\mathbf{q}} + \sum_{i=1}^{t} b_i \mathbf{u}_i)|^2$$

1. Initialise **b=0**
2. Generate initial model instance: $\mathbf{q} = (\overline{\mathbf{q}} + \sum_{i=1}^{t} b_i \mathbf{u}_i)$
3. Find *T* that best aligns **q** to **Y** (eg similarity transform)
4. Invert pose parameters, to project $\mathbf{y} = T^{-1}(\mathbf{Y})$ into model frame
5. Update the model parameters: $\mathbf{b} = \mathbf{U}^T(\mathbf{y} - \overline{\mathbf{q}})$, where $\mathbf{U} = [\mathbf{u}_1 | ... | \mathbf{u}_t]$
6. Repeat from step 2 until converged

Kee Cartilege in MRI Images

Initial

After 1 iteration



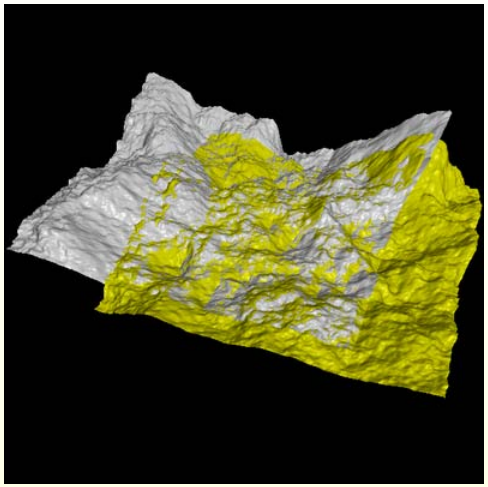
After 6 iterations

After 14 iterations

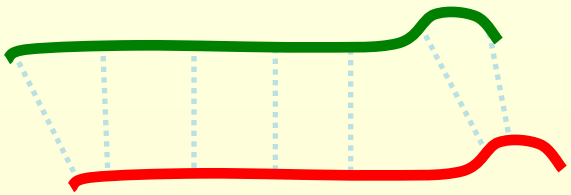
Correspondences from Proximity (ICP)

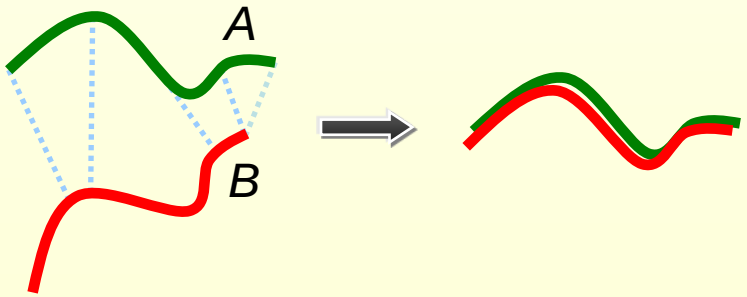
- Associate with a sample of points of *A* their closest point of *B*.
- This defines a set of corresponding points.
- Compute the best rigid alignment, and iterate.

RMSD decreases



- Works if A and B are close (but can get stuck in local minima).
- Relatively slow convergence (linear).
- Can go especially wrong in flat regions

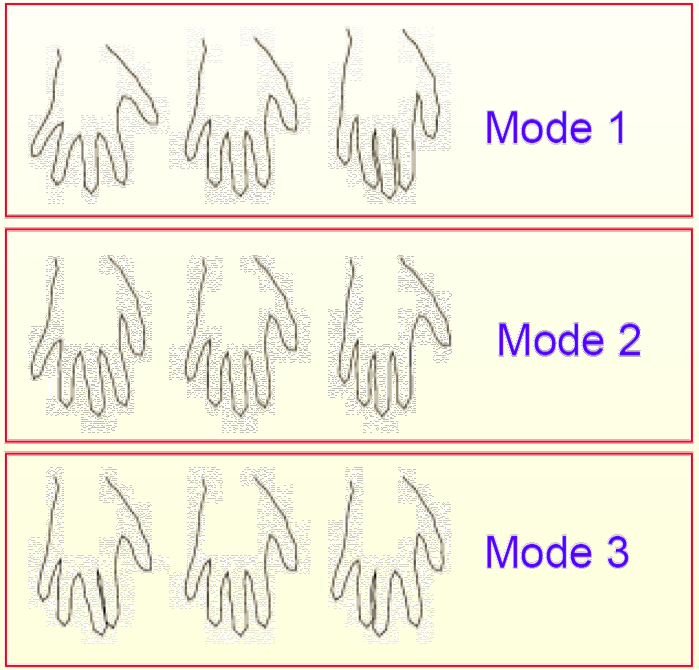




Iterated Closest Pair (ICP),
[Besl & McKay, 92]

Shape Models

- The variability in a set of related shapes can be captured through PCA analysis of sets of corresponding points
- Fitting such shape models requires the simultaneous estimation of pose and transform



The End