Digital Geometry
- Surface Registration

Junjie Cao @ DLUT

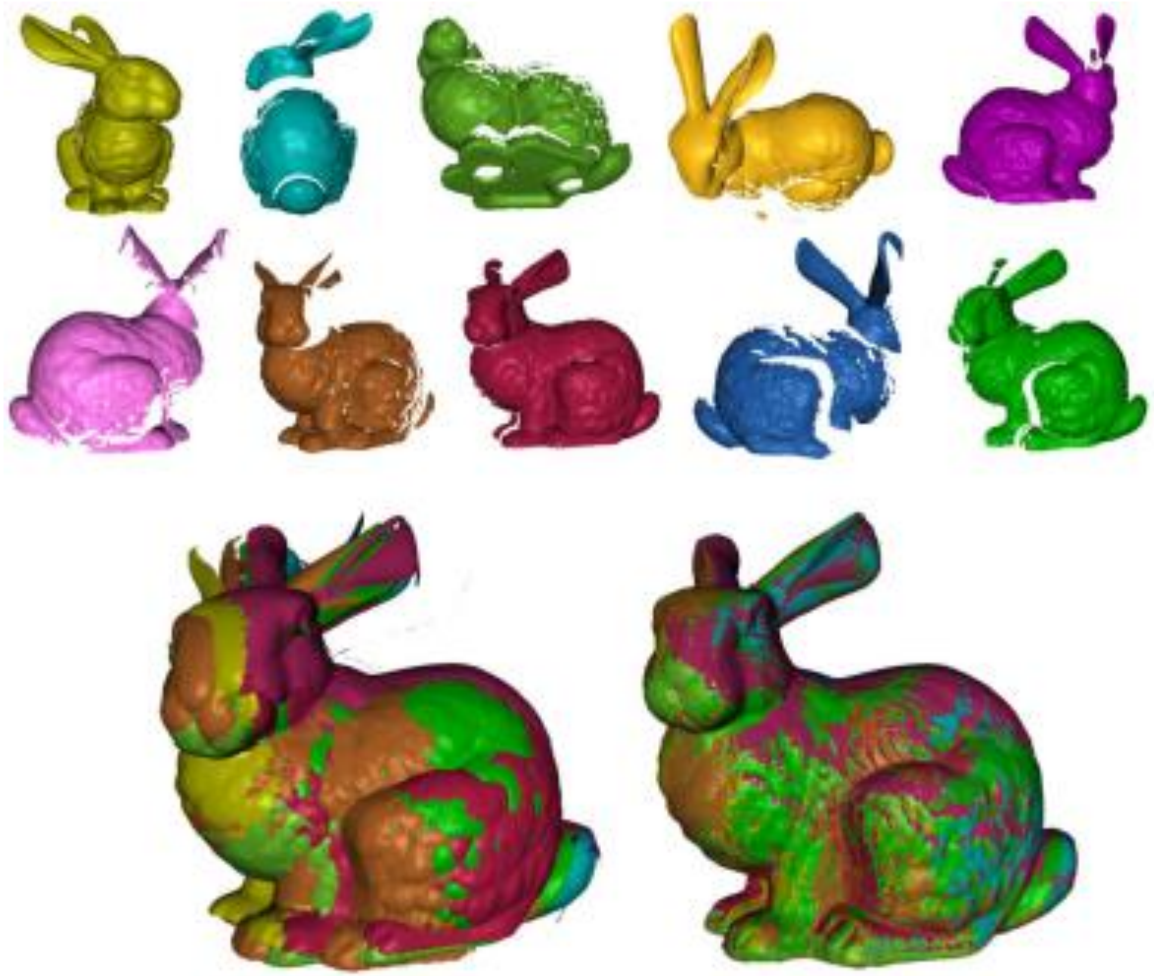
Spring 2017

http://jjcao.github.io/DigitalGeometry/

The purpose of computing is insight, not numbers

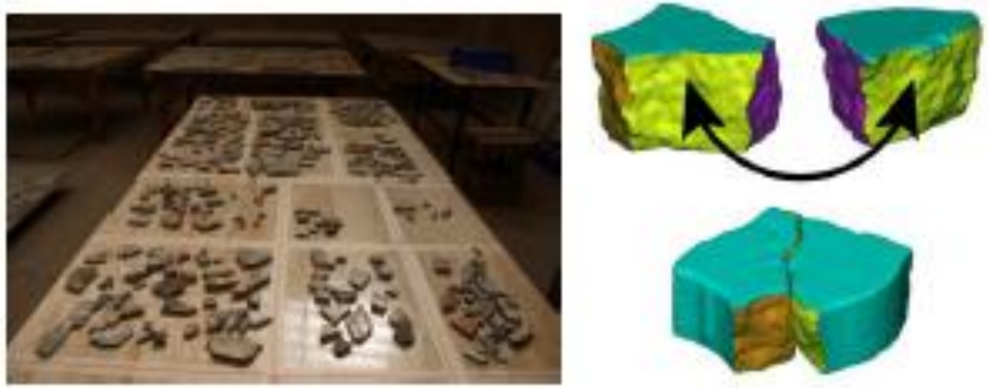
Definition

- Surface registration is the process of identifying and **matching corresponding regions**
- across **multiple scans** given in **arbitrary initial positions**,
- and estimating the corresponding **rigid transforms** that best align the scans to each other.

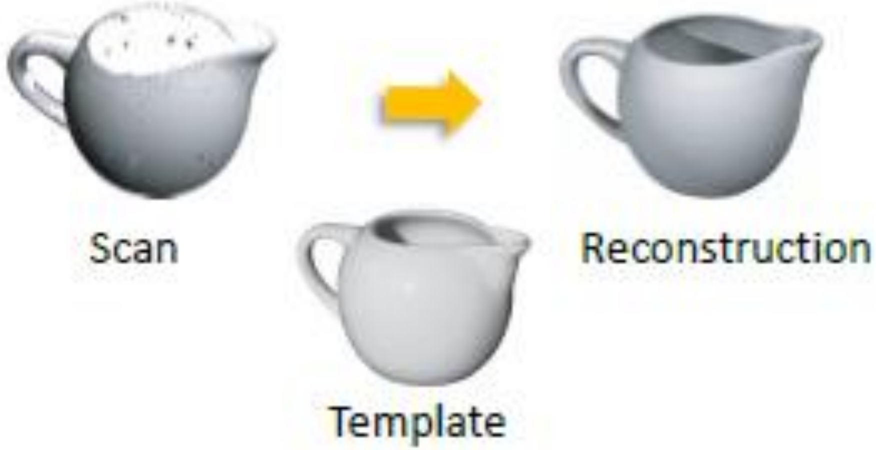


Applications

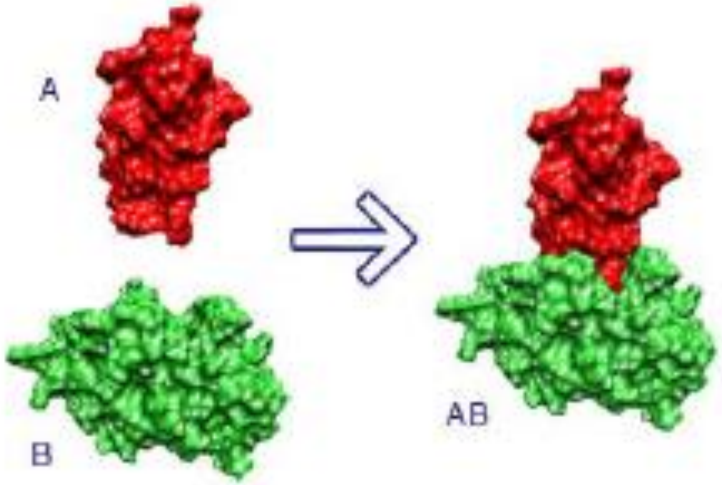
Surface reconstruction



Fragment assembly

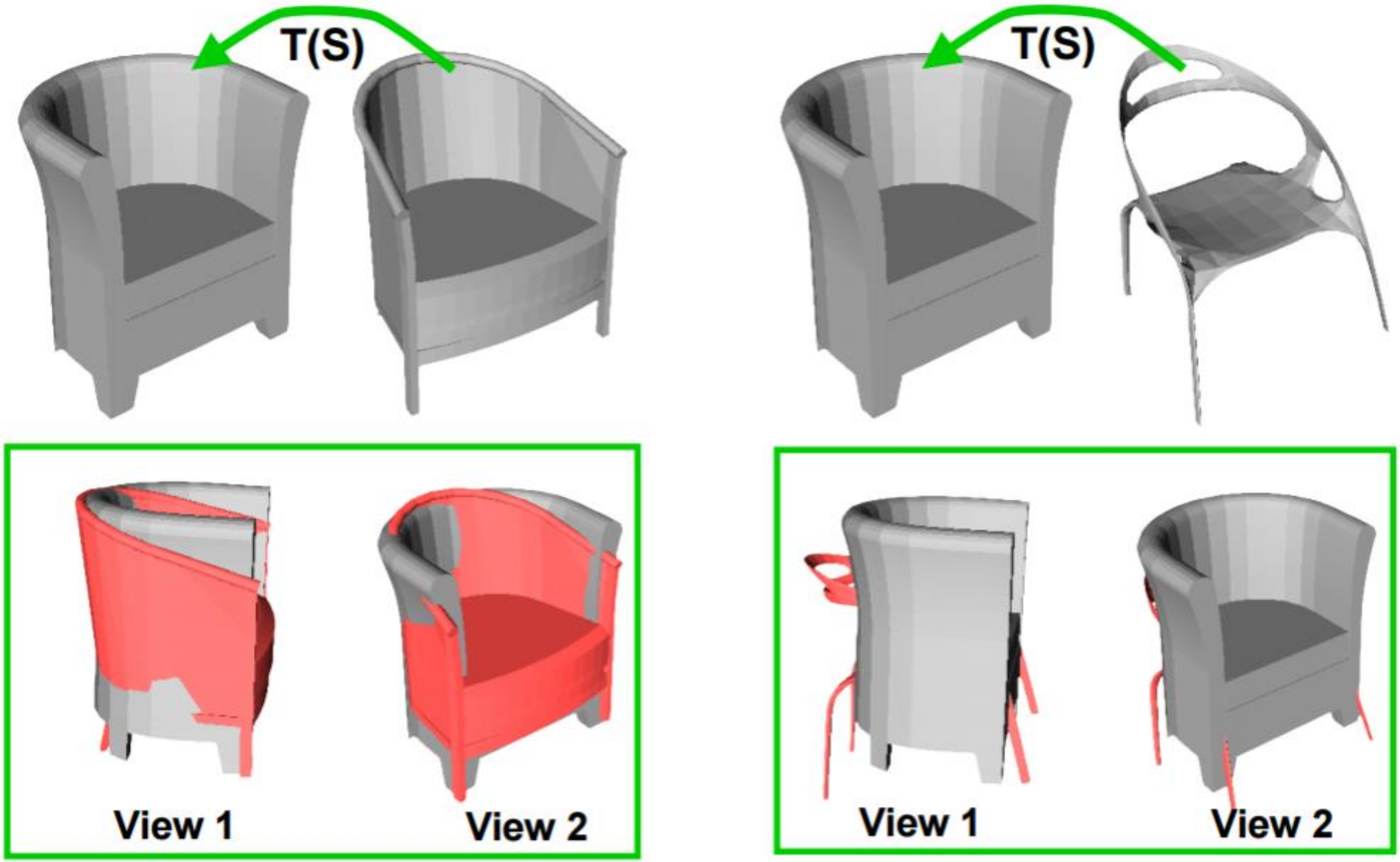


Object completion



Protein docking

Rigid Shape Matching – Search a transformation

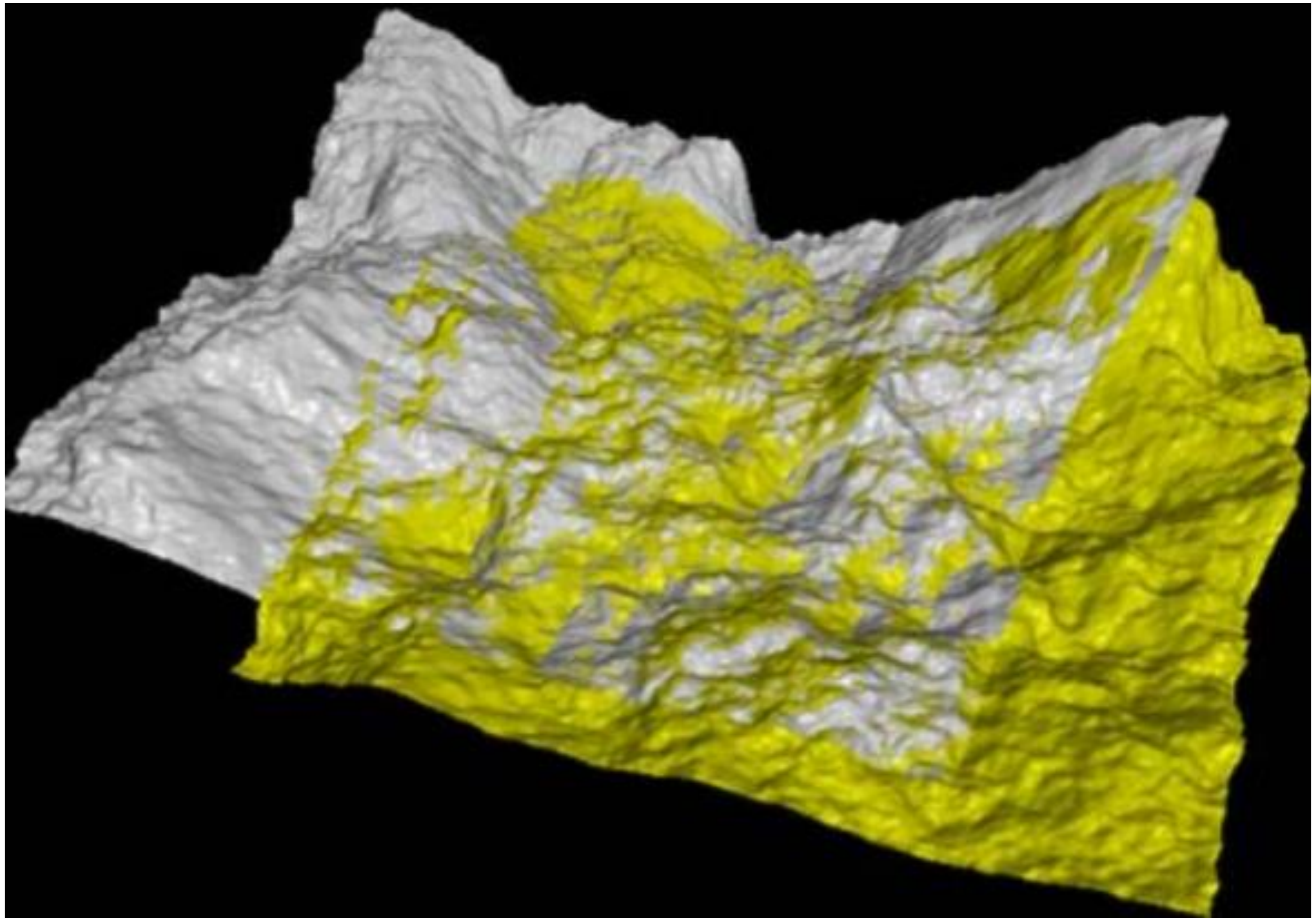


Correspondence Problem Classification

- How many meshes?
  - Two: Pairwise registration
  - More than two: multi-view registration
- Initial registration available?
  - Yes: Local optimization methods
  - No: Global methods
- Class of transformations?
  - Rotation and translation: Rigid-body
  - Non-rigid deformations

Pairwise Rigid Registration Goal

- Align two partially overlapping meshes given initial guess for relative transform



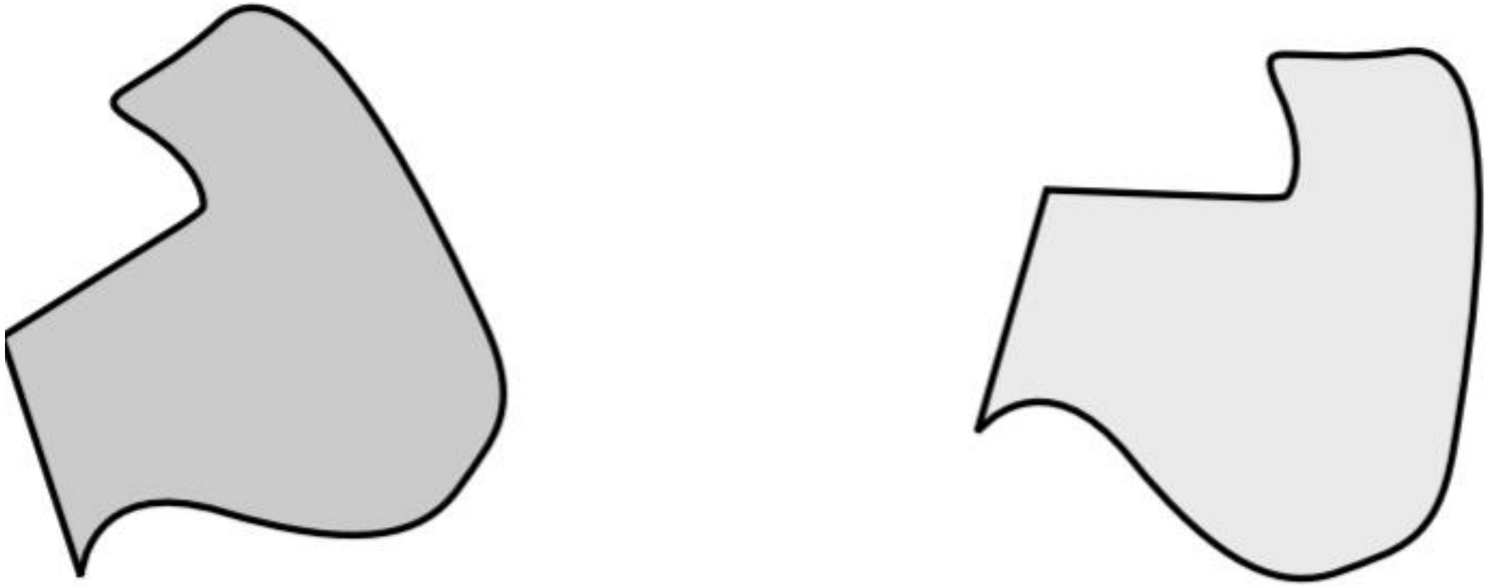
Outline

- Basic ICP: Iterative Closest Points
- Classification of ICP variants
  - Faster alignment
  - Better robustness
- Global Registration

ICP: Iterative Closest Points

Objective

M_1 M_2

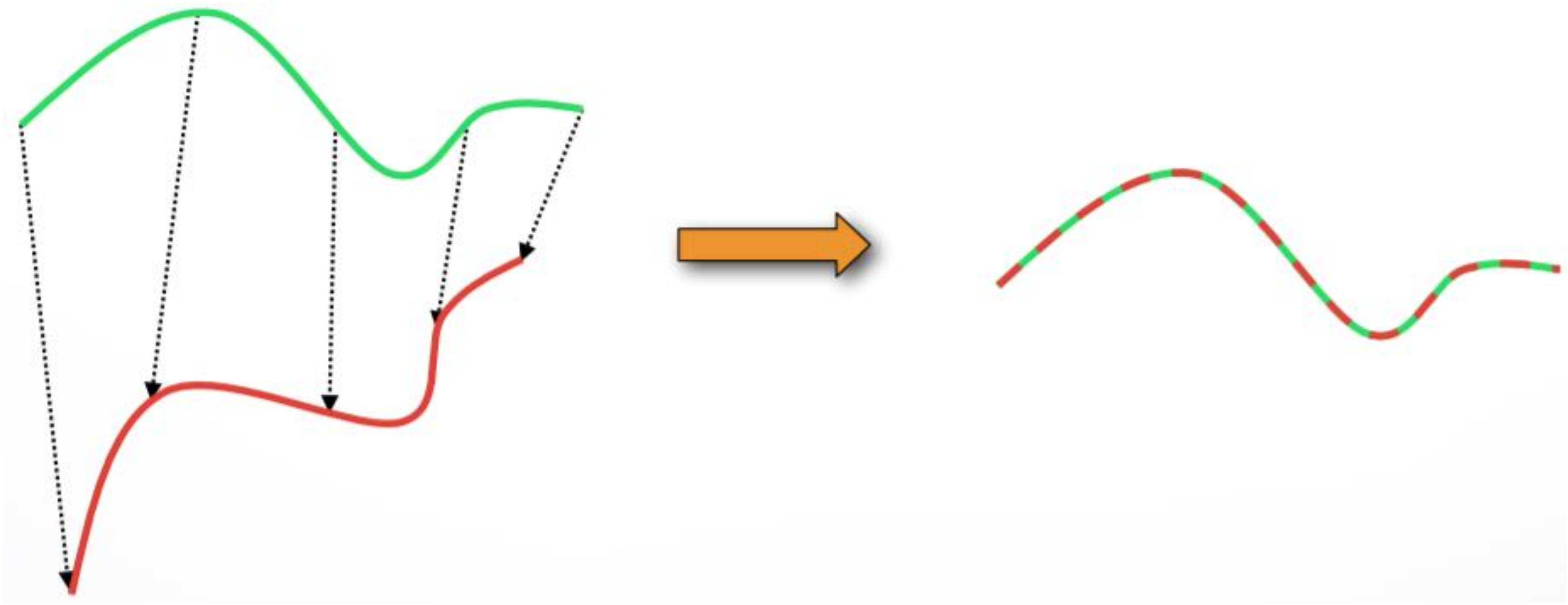


$$M_1 \approx T(M_2)$$

T : **translation + rotation**

Aligning 3D Data

- If correct **correspondences** are **known**, can find correct relative rotation/translation

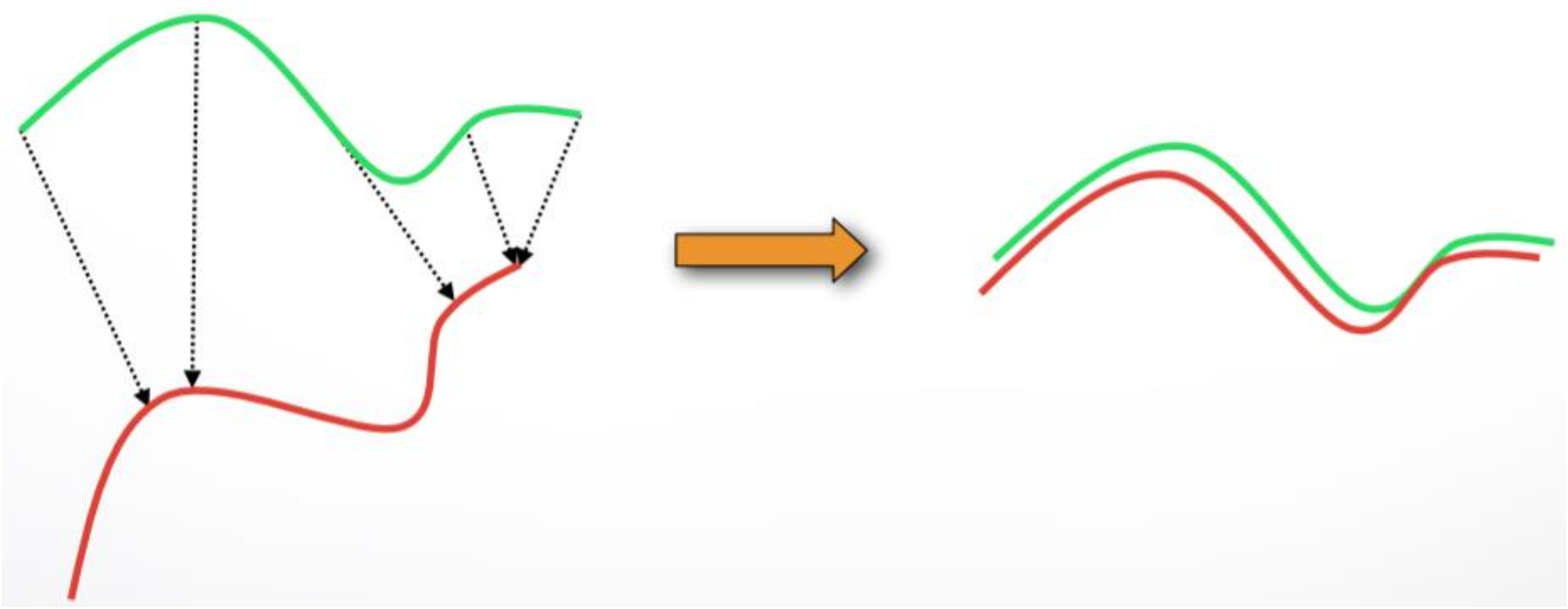


Aligning 3D Data

- How to find correspondences: User input? Feature detection? Signatures?

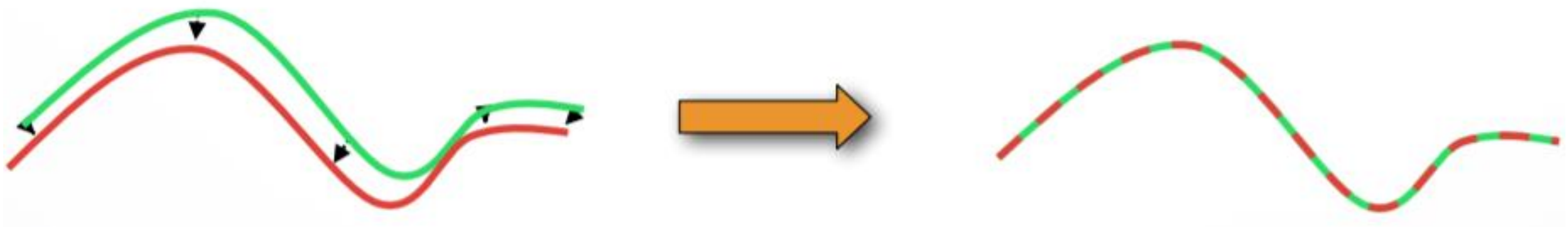
a seemingly-radical guess

- Alternatives: assume closest points correspond



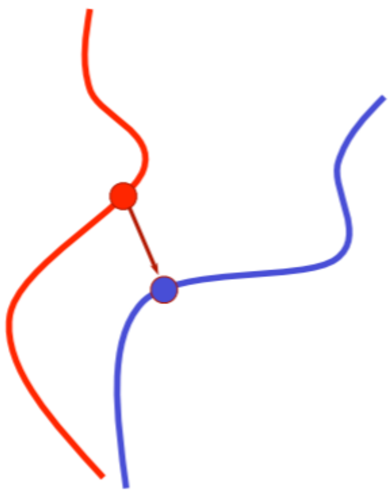
Aligning 3D Data

- … and iterate to find alignment
  - Iterative Closest Points (ICP) [Besl & Mckay]
- Converges if starting position “close enough”



Basic ICP

- **Select** e.g., 1000 random points



- **Match** each to closest point on other scan
- **Reject** pairs with distance > k times median

- Construct **error function**:

$$E = \sum \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|^2$$

- Minimize (closed form solution in [Horn 87]) (Also this note: Least-Squares Rigid Motion Using SVD by Olga Sorkine)

Shape Matching: Translation first. why?

Assume R is fixed and denote $F(\mathbf{t}) = \sum_{i=1}^{n} w_i \left\|(R\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\right\|^2$. We can find the optimal translation by taking the derivative of F w.r.t. $\mathbf{t}$ and searching for its roots:

$$\begin{aligned} 0 &= \frac{\partial F}{\partial \mathbf{t}} = \sum_{i=1}^{n} 2w_i \left(R\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\right) = \\ &= 2\mathbf{t}\left(\sum_{i=1}^{n} w_i\right) + 2R\left(\sum_{i=1}^{n} w_i \mathbf{p}_i\right) - 2\sum_{i=1}^{n} w_i \mathbf{q}_i. \end{aligned} \tag{2}$$

Denote

$$\bar{\mathbf{p}} = \frac{\sum_{i=1}^{n} w_i \mathbf{p}_i}{\sum_{i=1}^{n} w_i}, \quad \bar{\mathbf{q}} = \frac{\sum_{i=1}^{n} w_i \mathbf{q}_i}{\sum_{i=1}^{n} w_i}. \tag{3}$$

By rearranging the terms of (2) we get

$$\mathbf{t} = \bar{\mathbf{q}} - R\bar{\mathbf{p}}. \tag{4}$$

In other words, the optimal translation $\mathbf{t}$ maps the transformed weighted centroid of P to the weighted centroid of Q. Let us plug the optimal $\mathbf{t}$ into our objective function:

$$\mathbf{t} = \bar{\mathbf{q}} - R\bar{\mathbf{p}}. \tag{4}$$

In other words, the optimal translation $\mathbf{t}$ maps the transformed weighted centroid of P to the weighted centroid of Q. Let us plug the optimal $\mathbf{t}$ into our objective function:

$$\sum_{i=1}^{n} w_i \left\|(R\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\right\|^2 = \sum_{i=1}^{n} w_i \left\|R\mathbf{p}_i + \bar{\mathbf{q}} - R\bar{\mathbf{p}} - \mathbf{q}_i\right\|^2 = \tag{5}$$

$$= \sum_{i=1}^{n} w_i \left\|R(\mathbf{p}_i - \bar{\mathbf{p}}) - (\mathbf{q}_i - \bar{\mathbf{q}})\right\|^2. \tag{6}$$

We can thus concentrate on computing the rotation R by restating the problem such that the translation would be zero:

$$\mathbf{x}_i := \mathbf{p}_i - \bar{\mathbf{p}}, \quad \mathbf{y}_i := \mathbf{q}_i - \bar{\mathbf{q}}. \tag{7}$$

So we look for the optimal rotation R such that

$$R = \operatorname*{argmin}_R \sum_{i=1}^{n} w_i \left\|R\mathbf{x}_i - \mathbf{y}_i\right\|^2. \tag{8}$$

Shape Matching: Rotation

- Approximate nonlinear rotation by general matrix

$$\min_{\mathbf{R}} \sum_i \|\hat{\mathbf{p}}_i - \mathbf{R}\hat{\mathbf{q}}_i\|^2 \quad \rightarrow \quad \min_{\mathbf{A}} \sum_i \|\hat{\mathbf{p}}_i - \mathbf{A}\hat{\mathbf{q}}_i\|^2$$

- The least squares linear transformation is

$$\mathbf{A} = \left(\sum_{i=1}^{m} \hat{\mathbf{p}}_i \hat{\mathbf{q}}_i^T\right) \cdot \left(\sum_{i=1}^{m} \hat{\mathbf{q}}_i \hat{\mathbf{q}}_i^T\right)^{-1} \in \mathbb{R}^{3\times 3}$$

- SVD & Polar decomposition extracts rotation from $\mathbf{A}$

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad \rightarrow \quad \mathbf{R} = \mathbf{U}\mathbf{V}^T$$

Iterative Closest Points (ICP)-The classic version

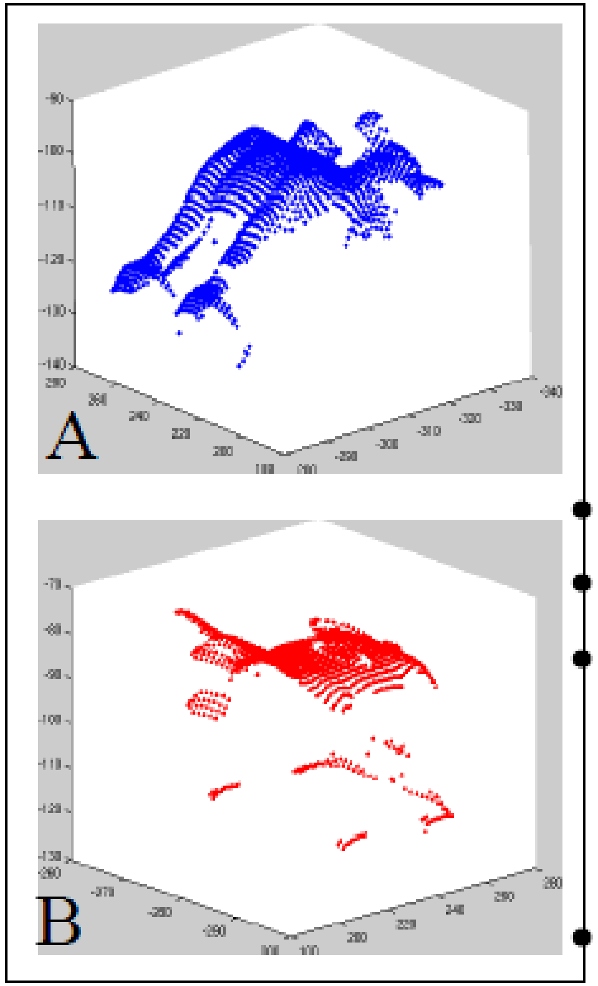
- Given two point sets P and Q, establish correspondence between each point at P and its closest point at Q.
- Iterate between two steps:
  1. Use the estimated correspondence to estimate the best rigid transformation and align the shapes.
  2. Derive new correspondence from the new alignment.
- Stop when there is no significant change.
- **The initial alignment is critical!**

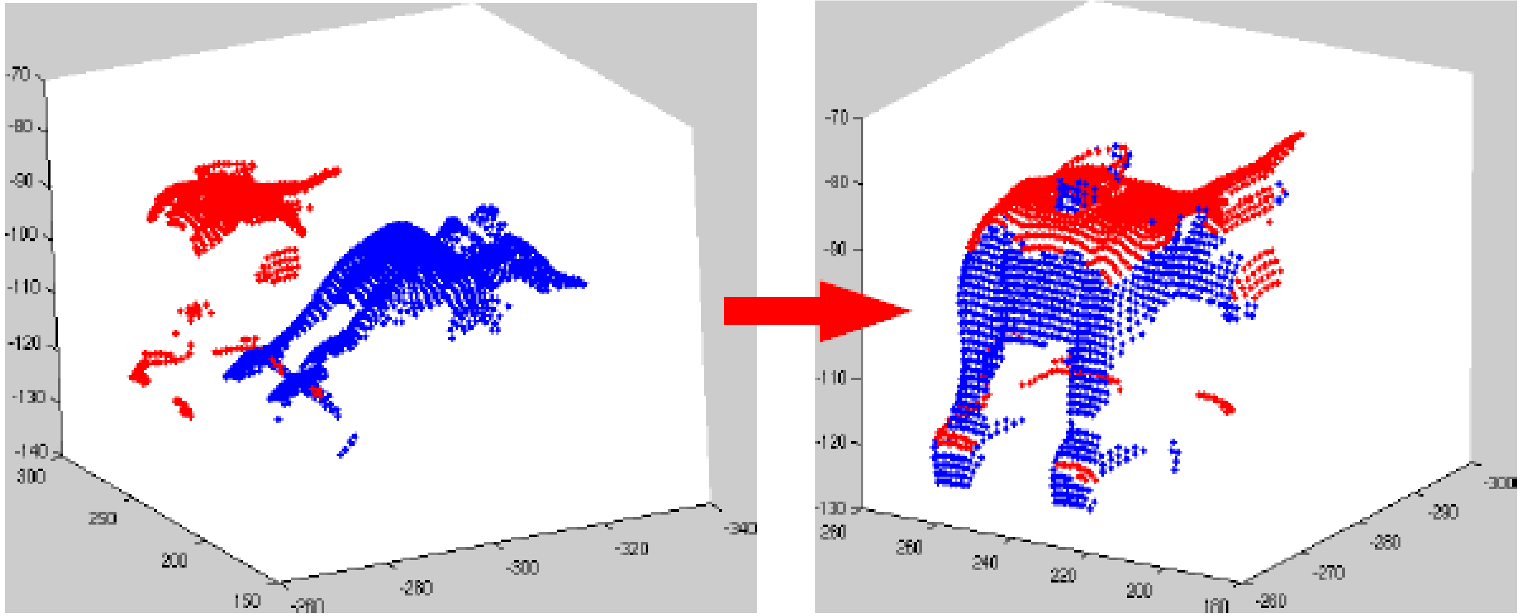
Suggestions:

1. Use a set of set of feature points/ user markers.
2. Use PCA to align the shapes.
3. Use the symmetry axes to align the shapes.

Registration – ICP algorithm

- Example : registration of 3D model parts (toy cow)





Input: point clouds acquired from non-aligned viewpoints (from 3D range scanner) & initial estimation of registration

Output: Transformation

ICP Variants

ICP Variants

Variants on the following stages of ICP have been proposed:

1. **Selecting source 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**

ICP Variants

- Can analyze various aspects of performance:
  - Speed
  - Stability
  - Tolerance of noise and/or outliers
  - Maximum initial misalignment
- Comparisons of many variants in
  - [Rusinkiewicz & Levoy, 3DIM 2001]: Efficient Variants of the ICP Algorithm

ICP Variants

1. Selecting source points (from one or both meshes)

2. Matching to points in the other mesh

3. Weighting the correspondences

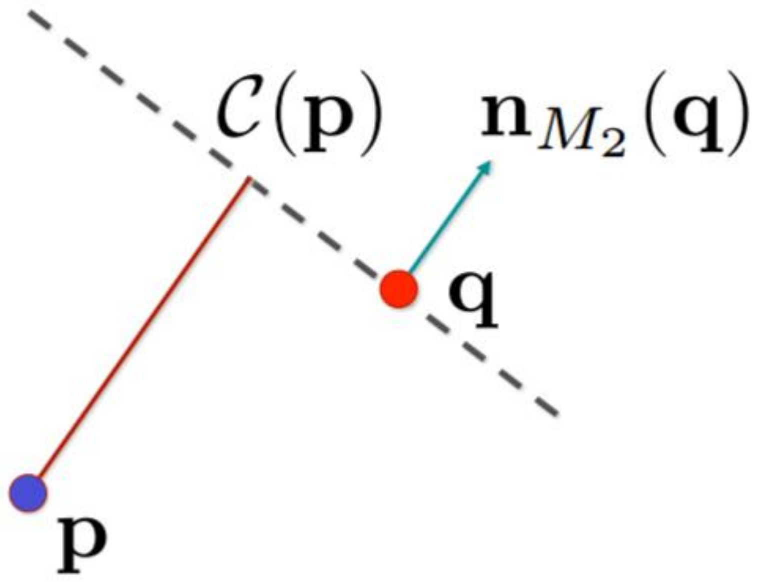
4. Rejecting certain (outliers) point pairs

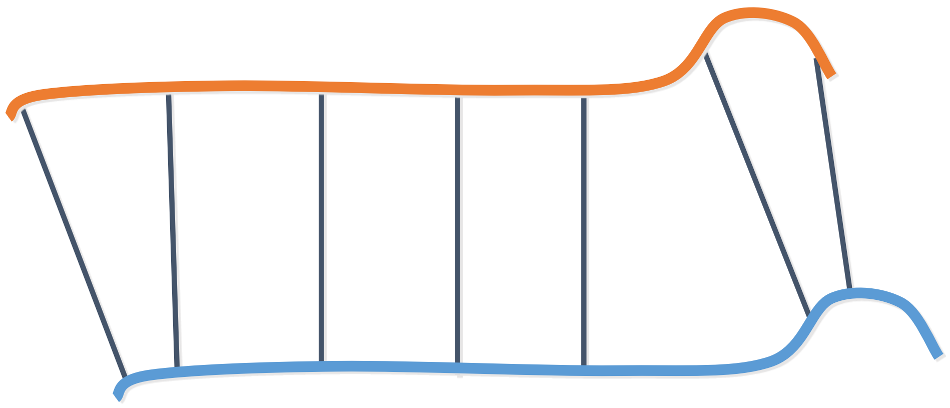
5. **Assigning an error metric to the current transform**

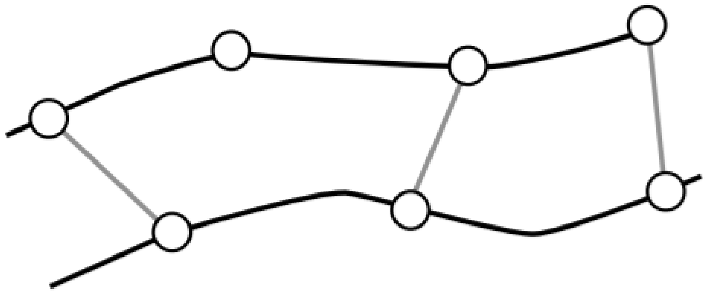
6. Minimizing the error metric w.r.t. transformation

Point-to-Plane Error Metric

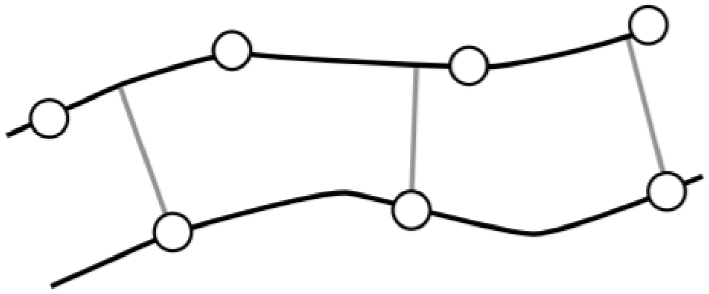
- Using point-to-plane distance instead of point-to-point
- allows flat regions slide along each other [Chen & Medioni 91]







point-to-point



point-to-plane

Point-to-Plane Error Metric

- Error function:

$$E = \sum \left((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i)^\top \mathbf{n_i} \right)^2$$

where $\mathbf{R}$ is a rotation matrix, $\mathbf{t}$ is a translation vector

- Linearize (i.e. assume that $\sin\theta \approx \theta$, $\cos\theta \approx 1$):

$$E \approx \sum \left((\mathbf{p}_i - \mathbf{q}_i)^\top \mathbf{n}_i \right) \; + \mathbf{r}^\top (\mathbf{p}_i \times \mathbf{n}_i) + \mathbf{t}^\top \mathbf{n}_i)^2 \qquad \mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

- Result: overconstrained linear system

Point-to-Plane Error Metric

- Overconstrained linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} \leftarrow & \mathbf{p}_1 \times \mathbf{n}_1 & \rightarrow & \leftarrow & \mathbf{n}_1 & \rightarrow \\ \leftarrow & \mathbf{p}_2 \times \mathbf{n}_1 & \rightarrow & \leftarrow & \mathbf{n}_2 & \rightarrow \\ & \vdots & & & \vdots & \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} r_x \\ r_y \\ r_z \\ t_x \\ t_y \\ t_z \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -(\mathbf{p}_1 - \mathbf{q}_1)^\top \mathbf{n}_1 \\ -(\mathbf{p}_2 - \mathbf{q}_2)^\top \mathbf{n}_2 \\ \vdots \end{bmatrix}$$

- Solve using least squares

$$\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b}$$

$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

Rotation matrix from axis and angle

For some applications, it is helpful to be able to make a rotation with a given axis. Given a unit vector $\mathbf{u} = (u_x, u_y, u_z)$, where $u_x^2 + u_y^2 + u_z^2 = 1$, the matrix for a rotation by an angle of θ about an axis in the direction of $\mathbf{u}$ is[3]

$$R = \begin{bmatrix} \cos\theta + u_x^2\left(1-\cos\theta\right) & u_x u_y\left(1-\cos\theta\right) - u_z\sin\theta & u_x u_z\left(1-\cos\theta\right) + u_y\sin\theta \\ u_y u_x\left(1-\cos\theta\right) + u_z\sin\theta & \cos\theta + u_y^2\left(1-\cos\theta\right) & u_y u_z\left(1-\cos\theta\right) - u_x\sin\theta \\ u_z u_x\left(1-\cos\theta\right) - u_y\sin\theta & u_z u_y\left(1-\cos\theta\right) + u_x\sin\theta & \cos\theta + u_z^2\left(1-\cos\theta\right) \end{bmatrix}$$

A derivation of this matrix from first principles can be found in section 9.2. here.[4]

This can be written more concisely as

$$R = \cos\theta\mathbf{I} + \sin\theta[\mathbf{u}]_{\times} + (1-\cos\theta)\mathbf{u}\otimes\mathbf{u}\,,$$

where $[\mathbf{u}]_{\times}$ is the cross product matrix of **u**, $\otimes$ is the tensor product and *I* is the Identity matrix, or alternatively as $R_{jk} = \begin{cases} \cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2}\left(2u_j^2 - 1\right), & \text{if } j = k \\ 2u_j u_k \sin^2\frac{\theta}{2} + \epsilon_{jkl} u_l \sin\theta, & \text{if } j \neq k \end{cases}$

where ϵ_{jkl} is the Levi-Civita symbol with $\epsilon_{123} = 1$. This is a matrix form of Rodrigues' rotation formula, (or the equivalent, differently parameterized Euler–Rodrigues formula) with[5]

$$\mathbf{u}\otimes\mathbf{u} = \mathbf{u}\mathbf{u}^T = \begin{bmatrix} u_x^2 & u_x u_y & u_x u_z \\ u_x u_y & u_y^2 & u_y u_z \\ u_x u_z & u_y u_z & u_z^2 \end{bmatrix}, \qquad [\mathbf{u}]_{\times} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}.$$

$$E = \sum \left((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i)^\top \mathbf{n}_i \right)^2 \Longrightarrow E \approx \sum \left((\mathbf{p}_i - \mathbf{q}_i)^\top \mathbf{n}_i \right) + \mathbf{r}^\top (\mathbf{p}_i \times \mathbf{n}_i) + \mathbf{t}^\top \mathbf{n}_i)^2$$

$$R = cos\theta I + sin\theta [r]_\times + (1 - cos\theta I)\ r(r)^T$$

- Key: $(Rp_i)^T n_i = \left((I + [r]_\times)p_i\right)^T n_i$
- $([r]_\times p_i)^T n_i \ ?= \ (r)^T(p_i \times n_i)$

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- $(r \times p_i)^T n_i \ ?= \ (r)^T(p_i \times n_i)$
- $(r \times p_i)^T n_i = n_i \cdot (r \times p_i);\ \ (r)^T(p_i \times n_i) = r \cdot (p_i \times n_i)$
- $n_i \cdot (r \times p_i) ?= r \cdot (p_i \times n_i)$

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{b} \cdot (\mathbf{c} \times \mathbf{a}) = \mathbf{c} \cdot (\mathbf{a} \times \mathbf{b})$$

Improving ICP Stability

- Closest compatible point
- Stable sampling

ICP Variants

1. Selecting source 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

Closest Compatible Point

- Closest point often a bad approximation to corresponding point
- Can improve matching effectiveness by restricting match to compatible points
  - Compatibility of colors [Godin et al. ’94]
  - Compatibility of normals [Pulli ’99]
  - Other possibilities: curvature, higher-order derivatives, and other local features (remember: data is noisy)

ICP Variants

1. **Selecting source points (from one or both meshes)**

2. Matching to points in the other mesh

3. Weighting the correspondences

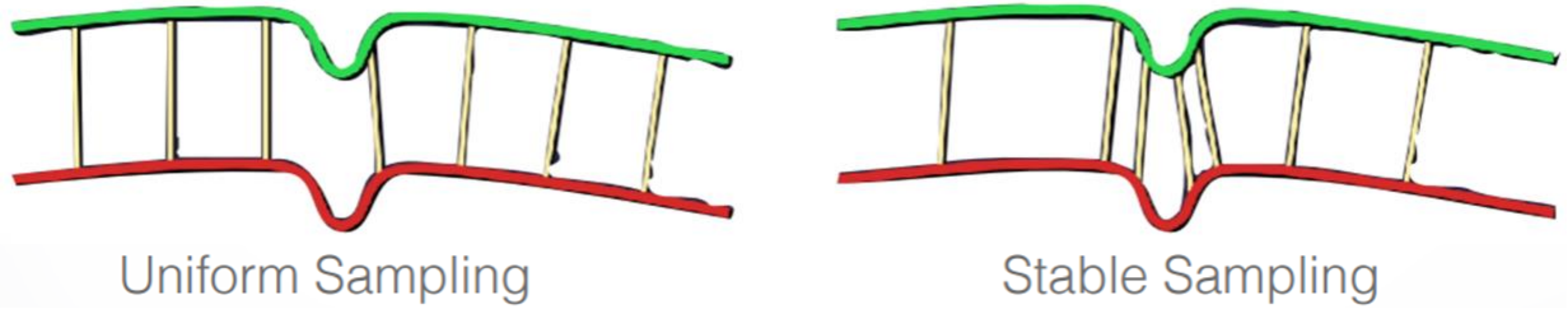
4. Rejecting certain (outliers) point pairs

5. Assigning an error metric to the current transform

6. Minimizing the error metric w.r.t. transformation

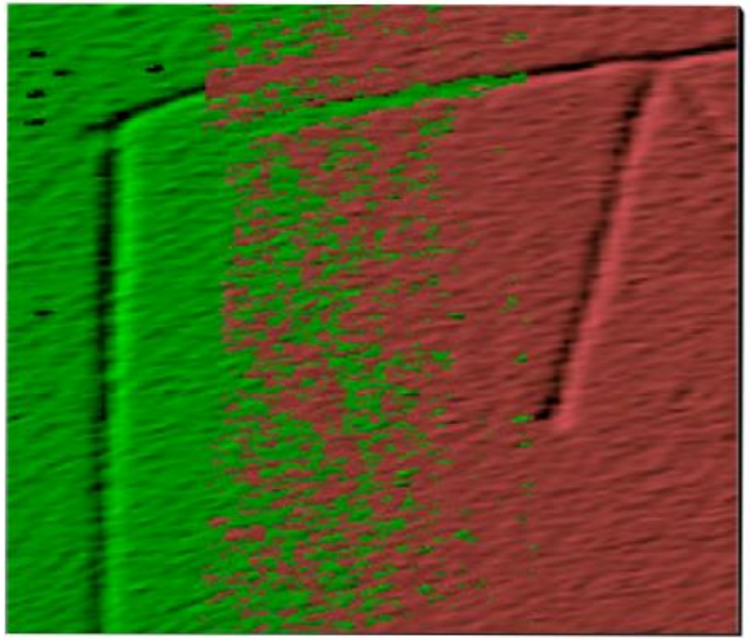
Selecting Source Points

- Use all points
- Uniform subsampling
- Random sampling
- Stable sampling [Gelfand et al. 2003]
  - Select samples that constrain all degrees of freedom of the rigid-body transformation

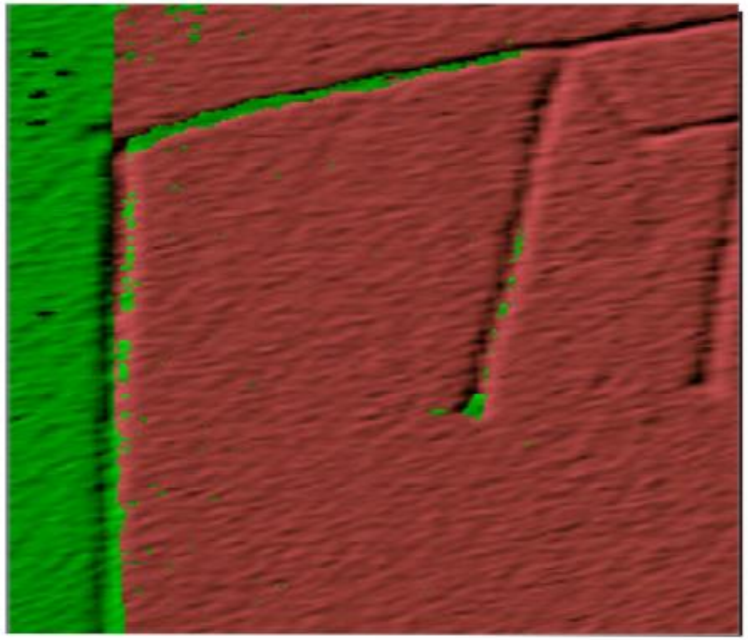


Sample Selection

- Simpler variant: normal-space sampling
  - select points with uniform distribution of normals
  - Pro: faster, does not require eigenanalysis
  - Con: only constrains translation
- Stability-based or normal-space sampling important for smooth areas with small features



Random Sampling



Normal-space Sampling

Selection vs. Weighting

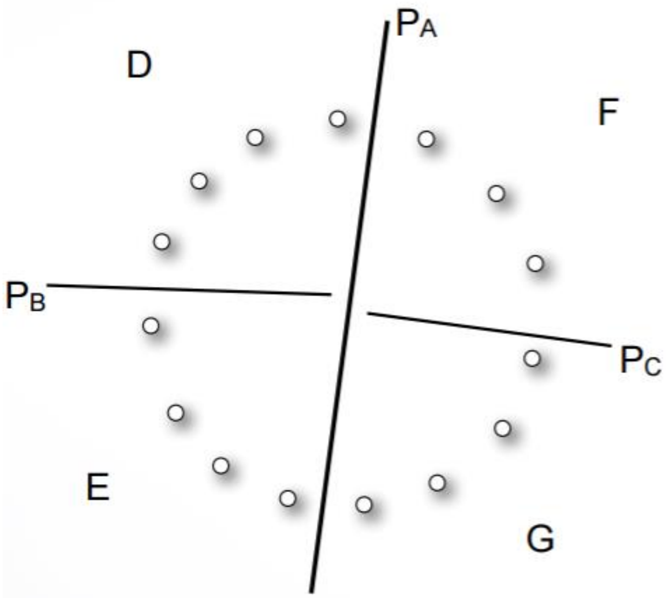
- Could achieve same effect with weighting
- Hard to ensure enough samples in features except at high sampling rates
- However, have to build special data structure
- Preprocessing / run-time cost tradeoff

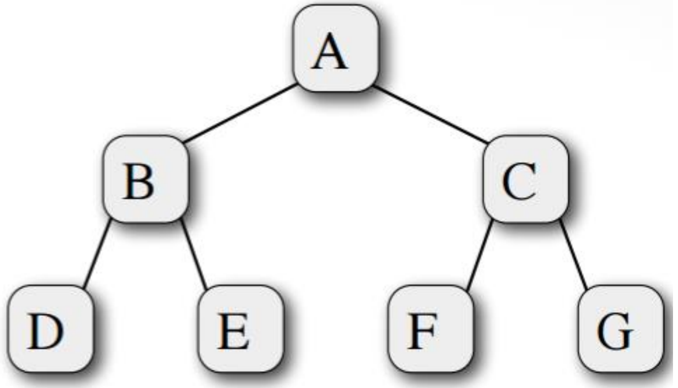
ICP Variants

- Can analyze various aspects of performance:
  - **Speed**
  - Stability
  - Tolerance of noise and/or outliers
  - Maximum initial misalignment
- Comparisons of many variants in
  - [Rusinkiewicz & Levoy, 3DIM 2001]

Closest Point Search

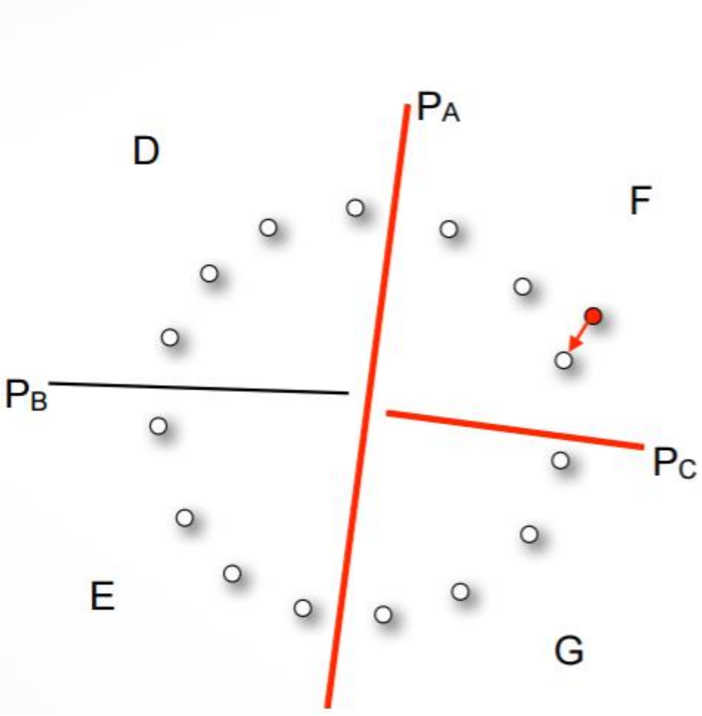
- most expensive stage of the ICP algorithm
  - Brute force search – O(n)

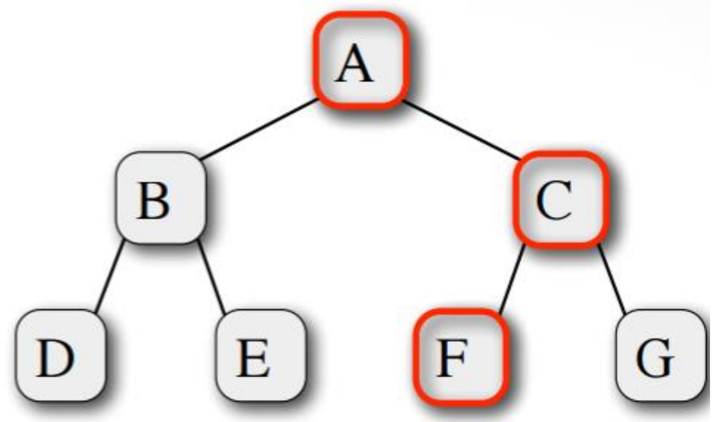




- Use Hierarchical BSP tree
  - Binary space partitioning tree (general kD-tree)
  - Recursively partition 3D space by planes
  - Tree should be balanced, put plane at median
  - log(n) tree levels, complexity O(nlog n)

BSP Closest Point Search

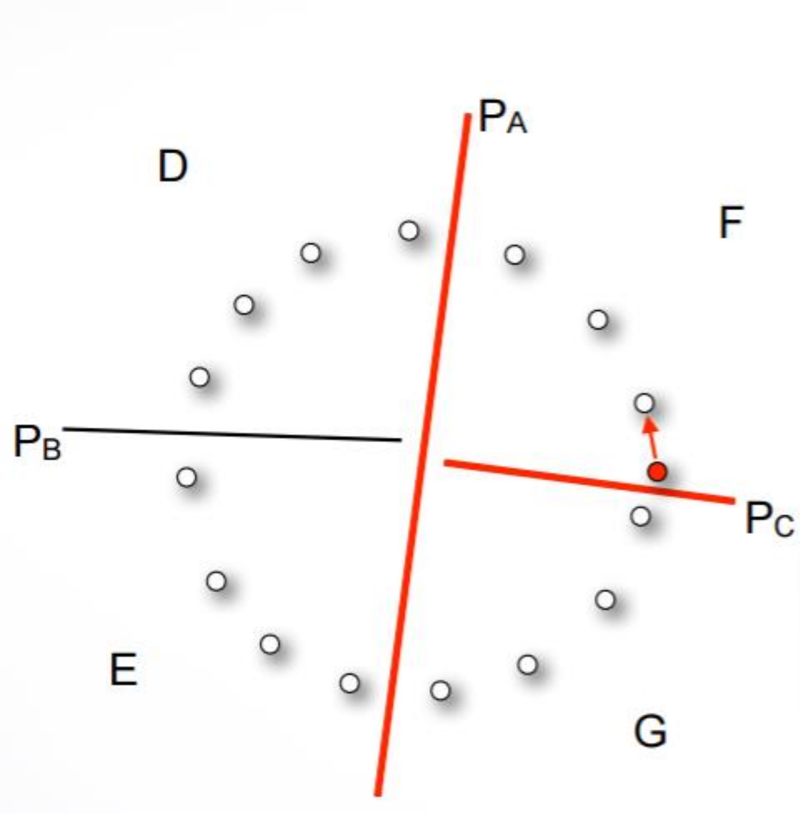


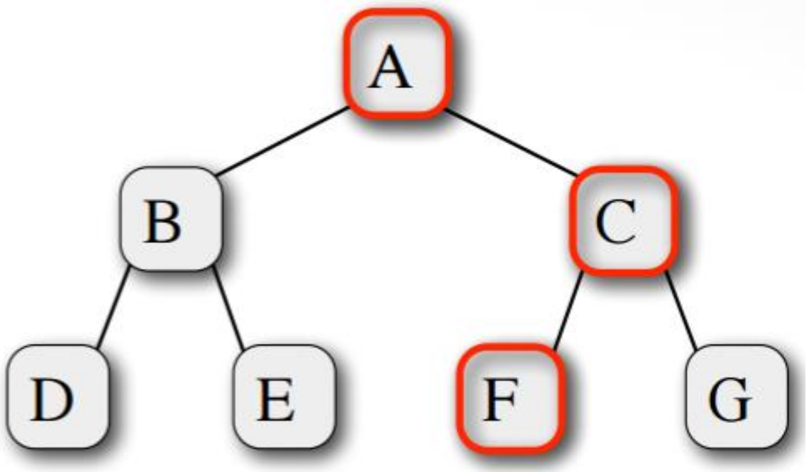


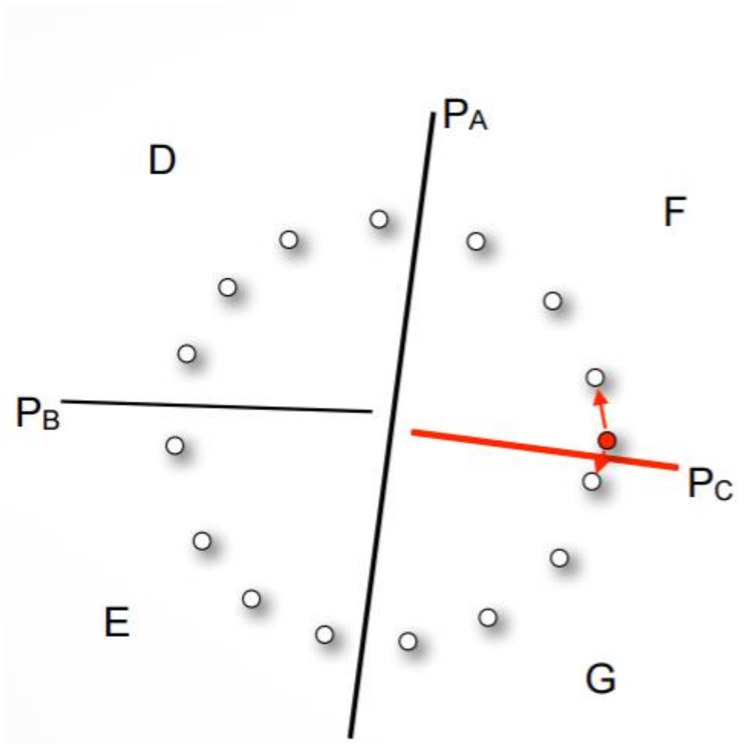
Implement BSPNode::dist() with the following info:

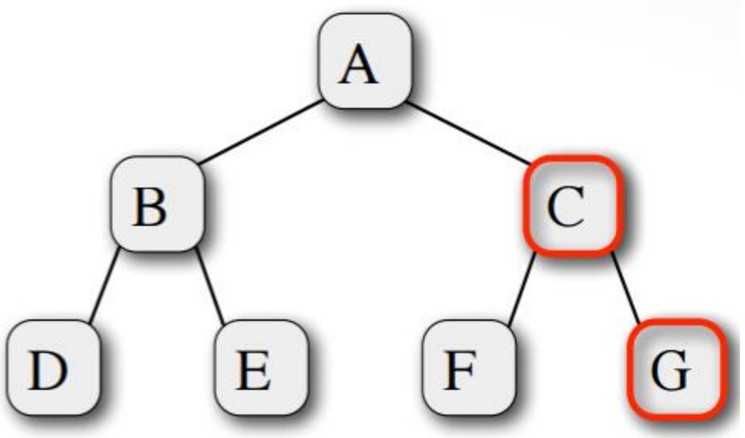
BSPNode{
BSPNode left_child, right_child;
vector<Point> p; // p[i] is the ith point
...
bool leaf_node();
void dist(Point x, Scalar& dmin): x: the query point, dmin: min distance between x and its closest point in the Tree.
};
float dist = dist_to_plane(x)

How to handle this?









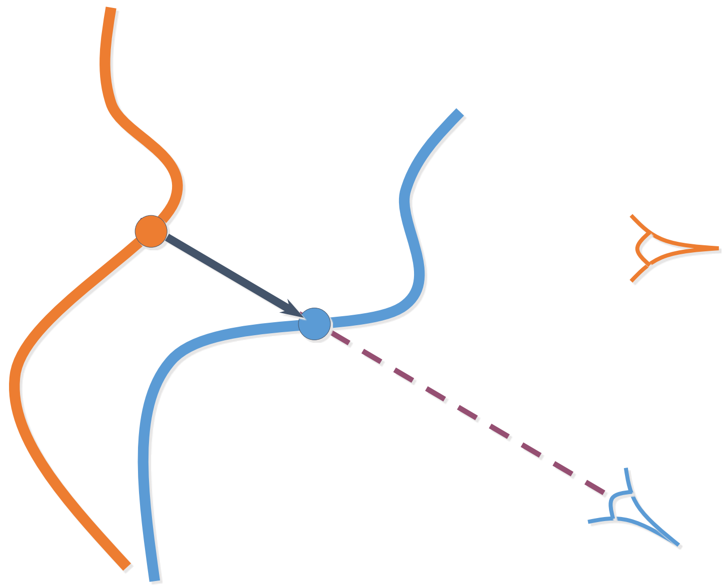
BSP Closest Point Search

```
BSPNode::dist(Point x, Scalar& dmin)
{
  if (leaf_node())
    for each sample point p[i]
      dmin = min(dmin, dist(x, p[i]));

  else
  {
    d = dist_to_plane(x);
    if (d < 0)
    {
      left_child->dist(x, dmin);
      if (|d| < dmin) right_child->dist(x, dmin);
    }
    else
    {
      right_child->dist(x, dmin);
      if (|d| < dmin) left_child->dist(x, dmin);
    }
  }
}
```

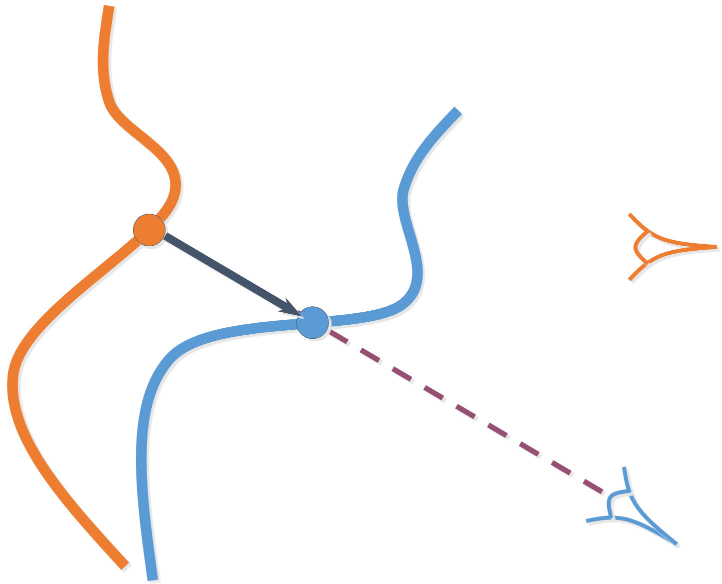
Closest Point Search

- most expensive stage of the ICP algorithm
  - Brute force search – O(n)
  - Binary space partitioning tree (general kD-tree) - O(nlog n)



Projection to Find Correspondence

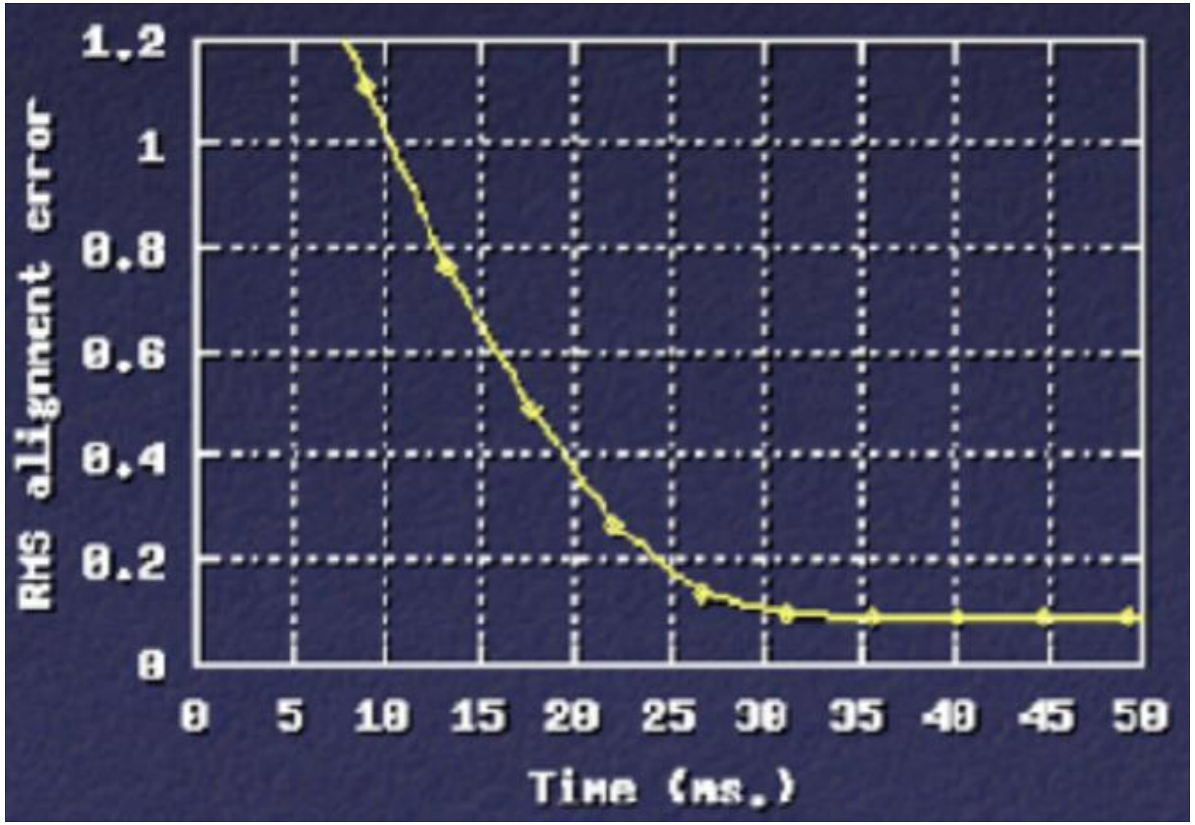
- Idea: use a simpler algorithm to find correspondences
- For range images, can simply project point by “reverse calibration” [Blais 95]
  - Constant-time
  - Does not require precomputing a spatial data structure



$$\tilde{\boldsymbol{x}}_s = \underset{\substack{\uparrow \\ \text{Calibration Matrix}}}{\boldsymbol{K}} \left[\, \boldsymbol{R} \,\middle|\, \boldsymbol{t} \,\right] \boldsymbol{p}_w = \underset{\substack{\uparrow \\ \text{Camera Matrix}}}{\boldsymbol{P}} \boldsymbol{p}_w$$

Projection-Based Matching

- Slightly worse performance per iteration
- Each iteration is one to two orders of magnitude faster than closest point
- Result: can align two range images in a few milliseconds, vs. a few seconds

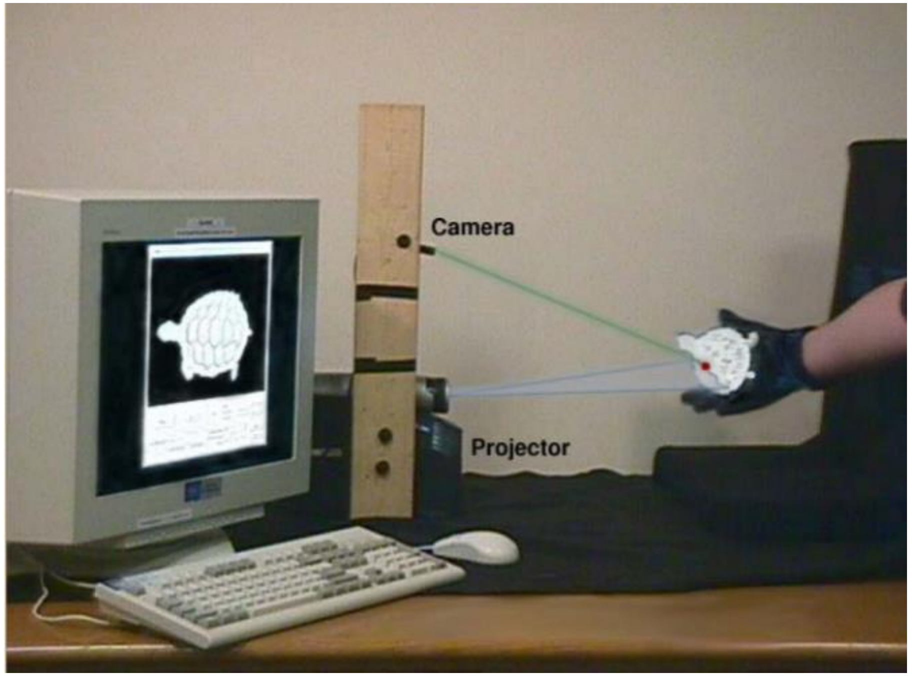


Applications

- Given:
  - A scanner that returns range images in real time
  - Fast ICP
  - Real-time merging and rendering

- Result: 3D model acquisition
  - Tight feedback loop with user
  - Can see and fill holes while scanning

Scanner Layout



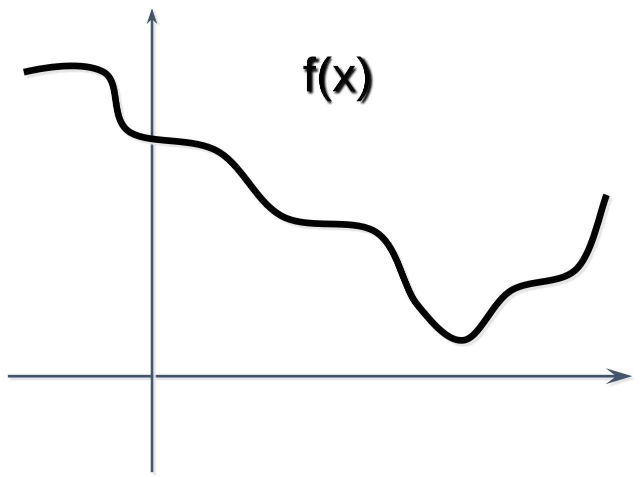
Artec Group

[Newcombe et al. '11]
KinectFusion

What Does ICP Do?

- Two ways of thinking about ICP:
  - Solving the correspondence problem
  - Minimizing point-to-surface squared distance

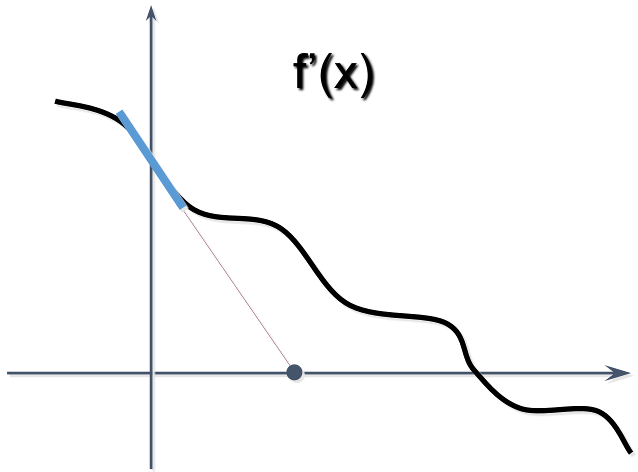
- ICP is like Newton's method on an approximation of the distance function



What Does ICP Do?

- Two ways of thinking about ICP:
  - Solving the correspondence problem
  - Minimizing point-to-surface squared distance

- ICP is like Newton's method on an approximation of the distance function

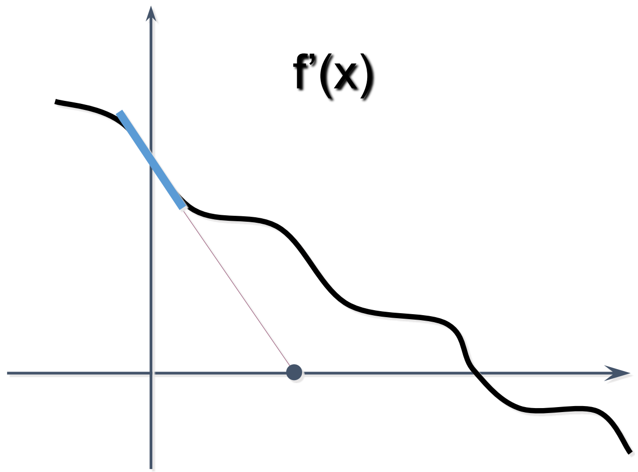


What Does ICP Do?

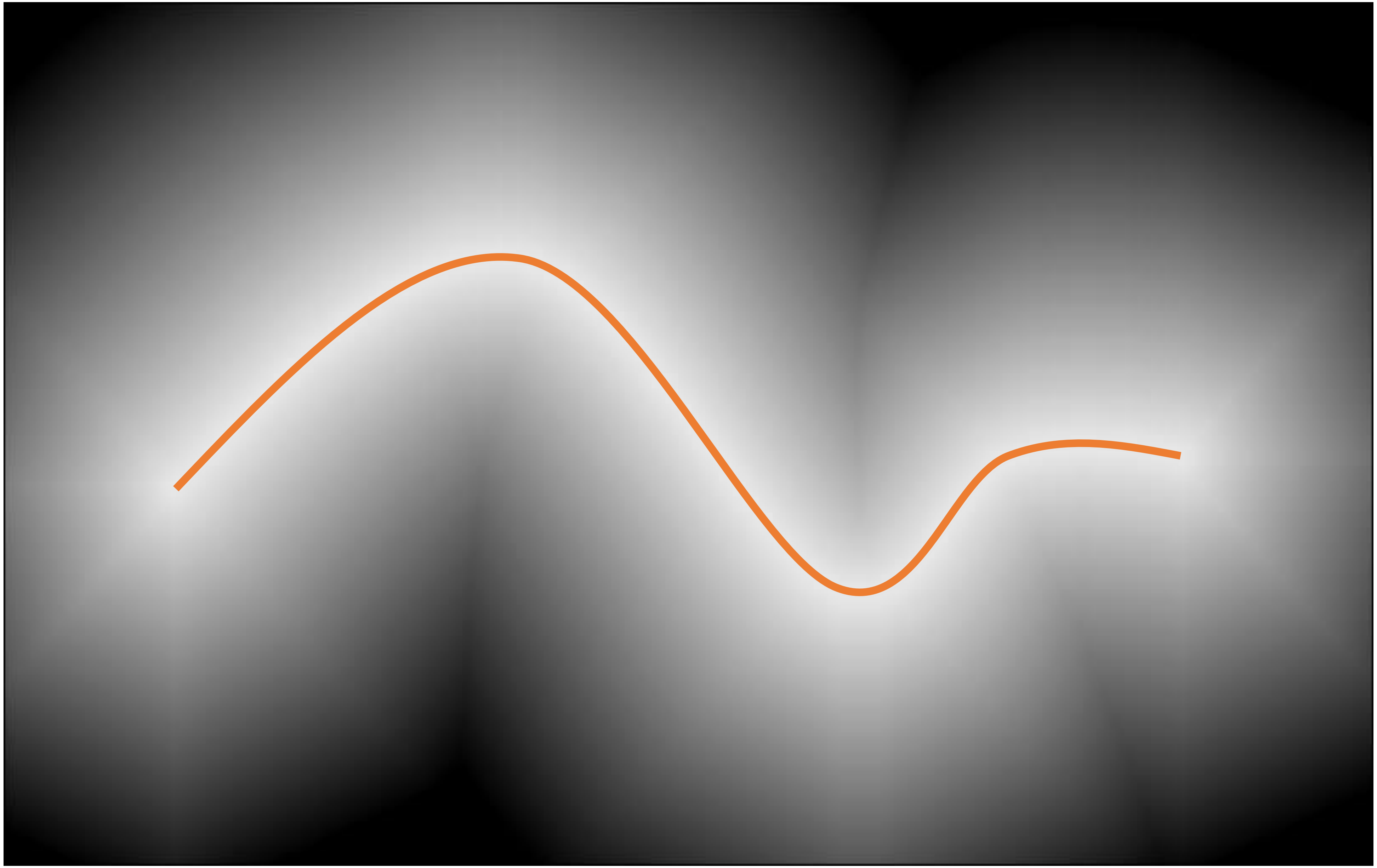
- Two ways of thinking about ICP:
  - Solving the correspondence problem
  - Minimizing point-to-surface squared distance

- ICP is like Newton's method on an approximation of the distance function
  - ICP variants affect shape of global error function

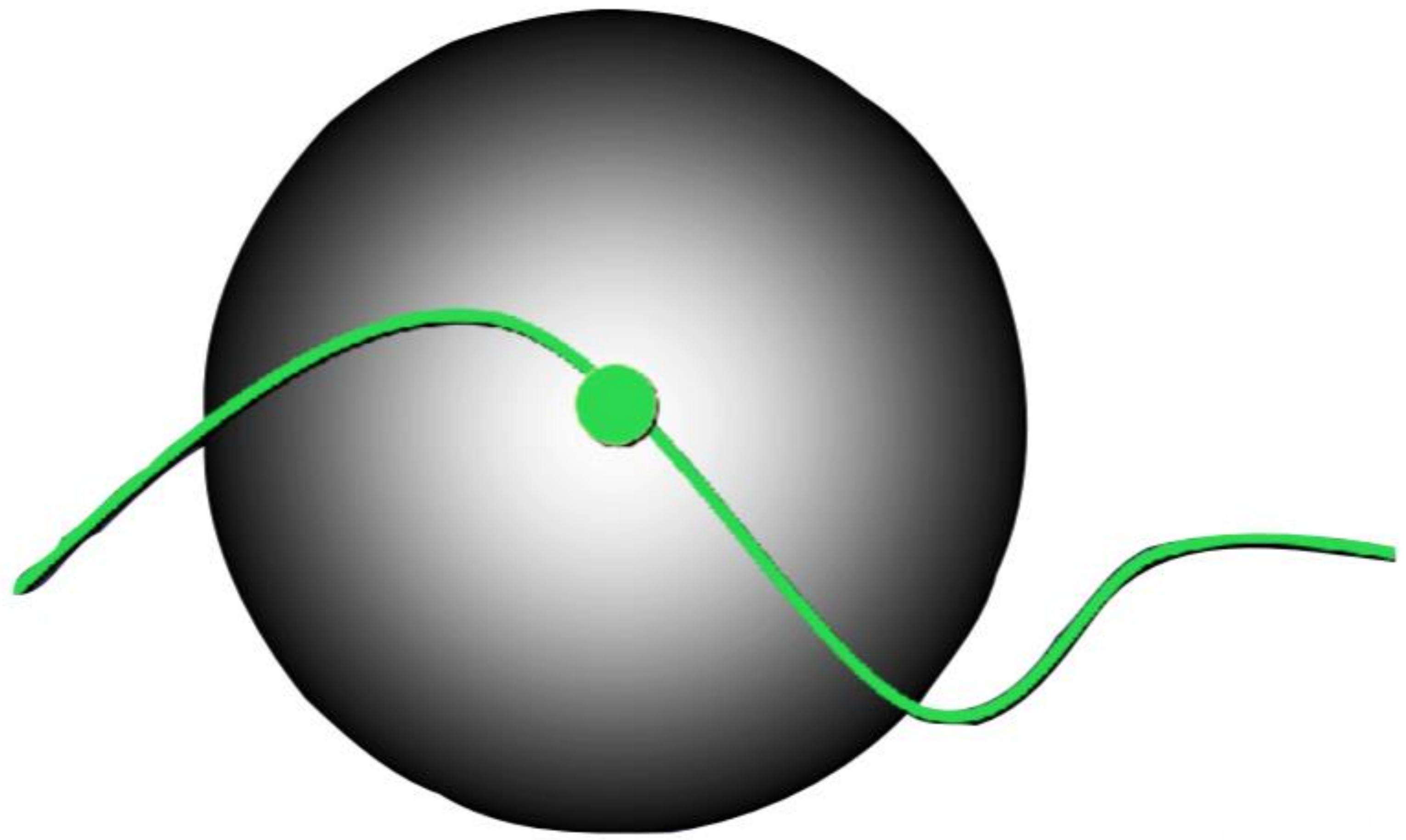
  or local approximation



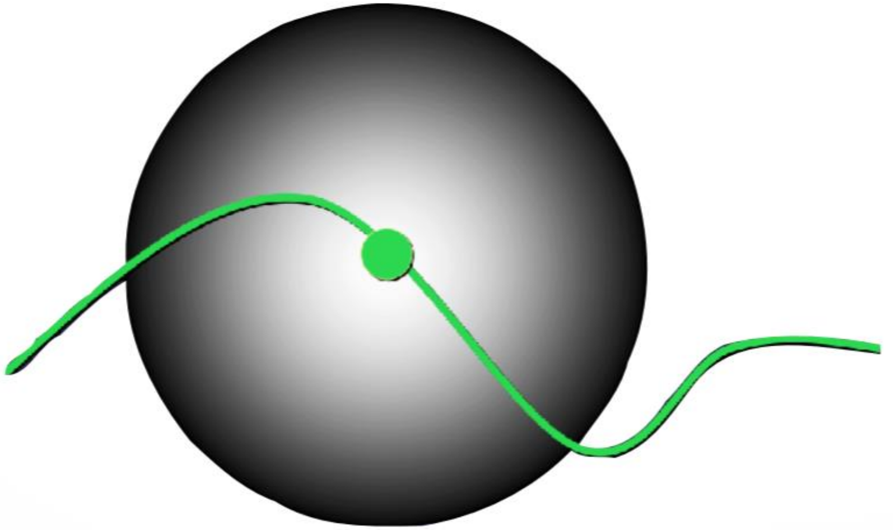
Point-to-Surface Distance

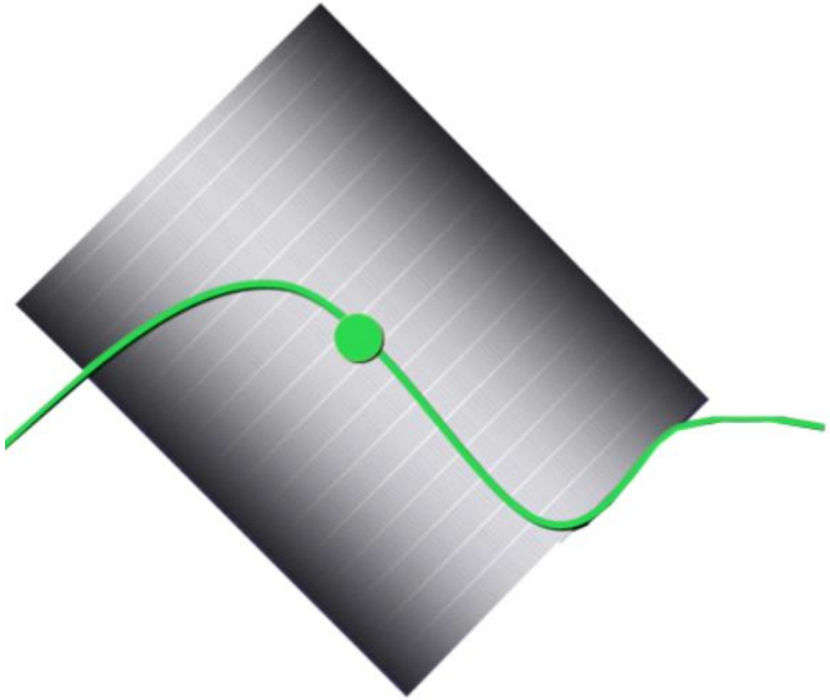


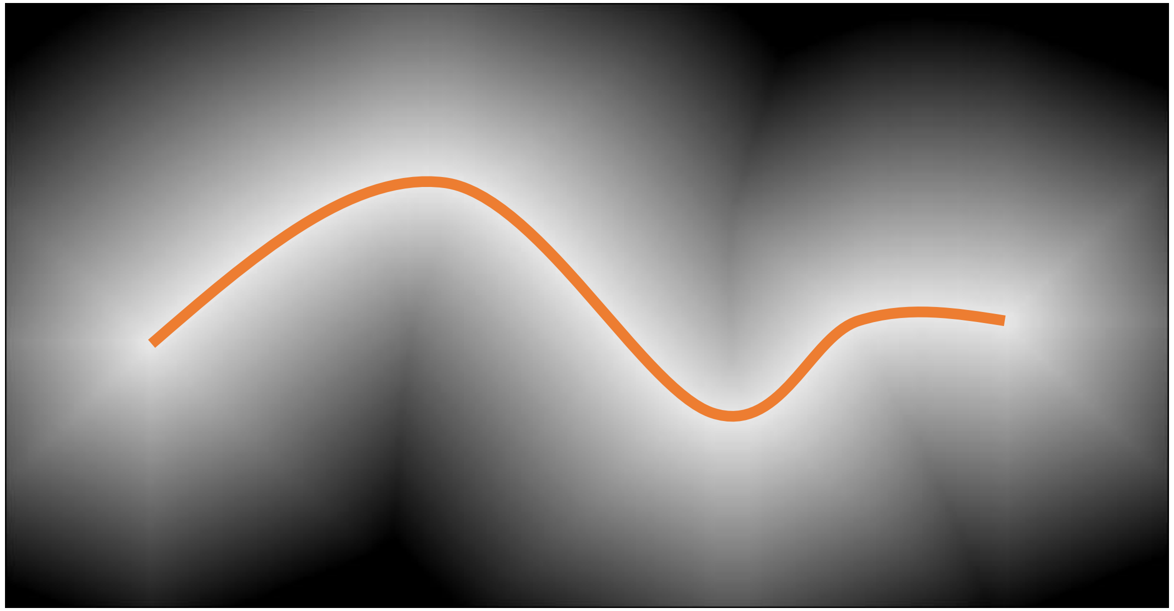
Point-to-Point Distance



Point-to-Plane Distance

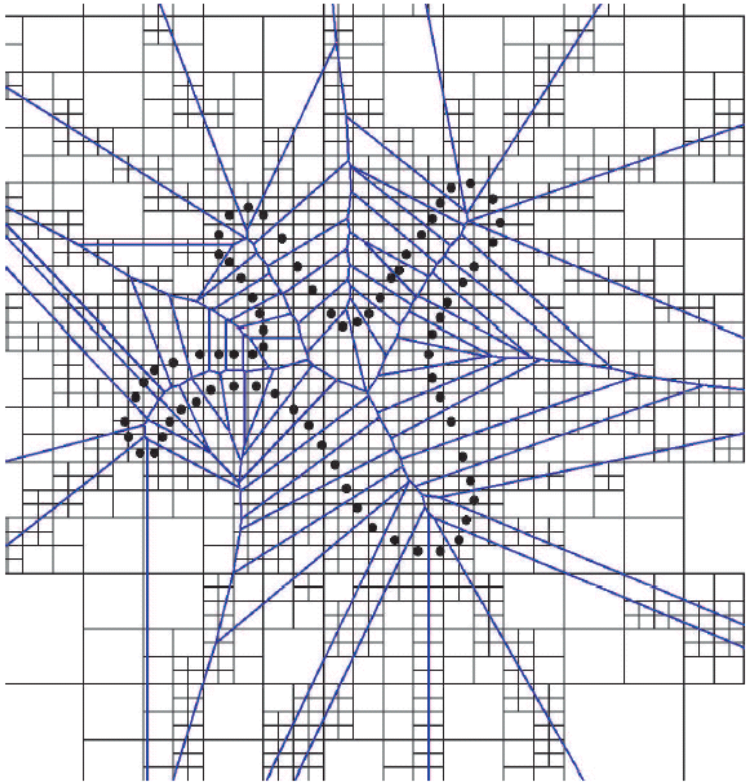




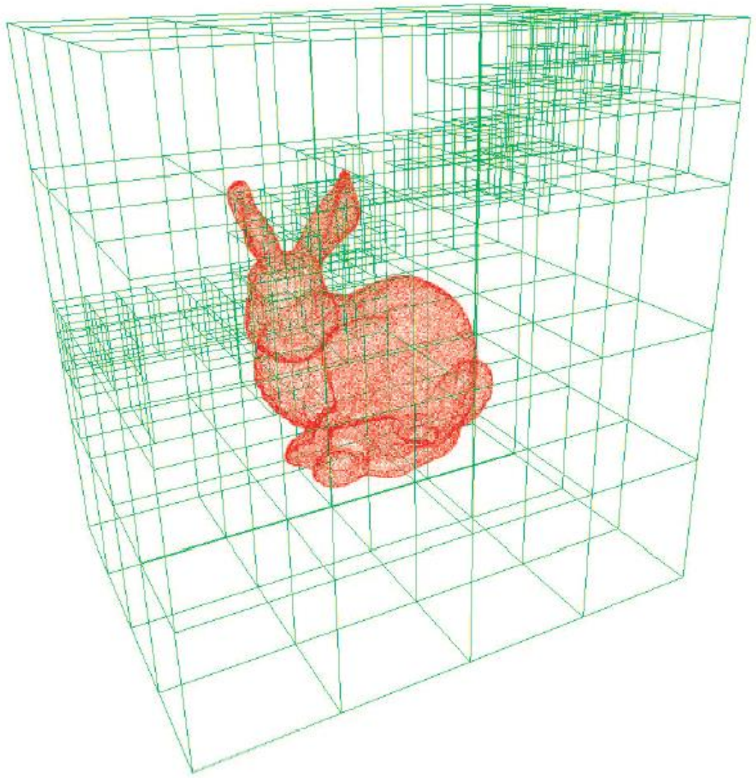


Mitra et al.'s Optimization

- Precompute piecewise-quadratic approximation to distance field throughout space
- Store in “d2tree” data structure



2D



3D

[Mitra et al. 2004]

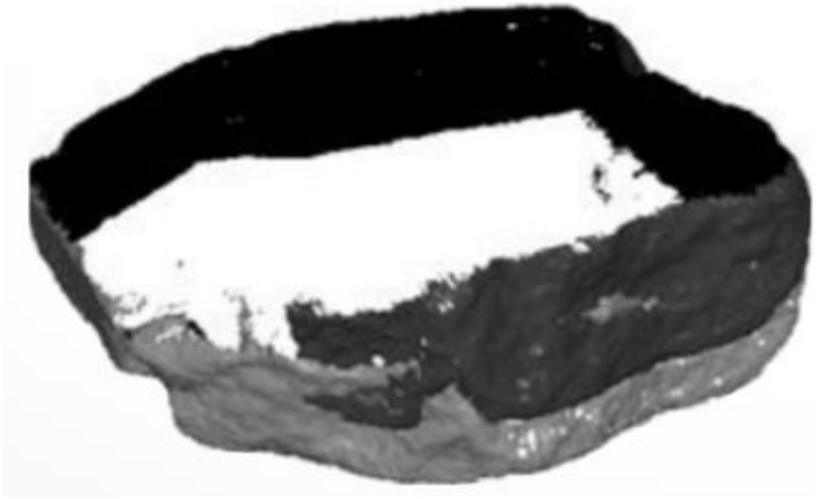
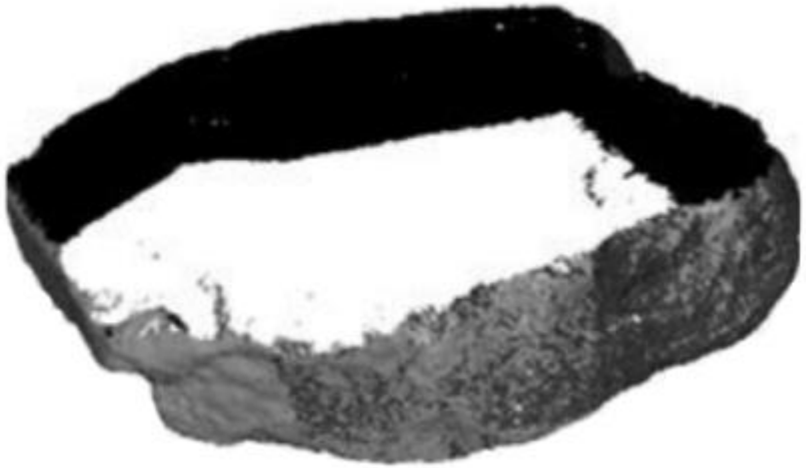
Mitra et al.’s Optimization

- Precompute piecewise-quadratic approximation to distance field throughout space
- Store in “d2tree” data structure

- At run time, look up quadratic approximants and optimize using Newton’s method
  - More robust, wider basin of convergence
  - Often fewer iterations, but more precomputation

Global Registration Goal

- Given: n scans around an object
- Goal: align them all
- First attempt: apply ICP to each scan to one other
- Want method for distributing accumulated error among all scans

Approach #1: Avoid the Problem

- In some cases, have 1 (possibly low-resolution) scan that covers most surface

- Align all other scans to this “anchor” [Turk 94]

- Disadvantage: not always practical to obtain anchor scan

Approach #2: The Greedy Solution

- Align each new scan to the union of all previous scans [Masuda ’96]
- Disadvantages:
  - Order dependent
  - Doesn’t spread out error
    - This sometimes avoids catastrophic accumulation of error, but really isn’t guaranteed to do anything.

Approach #3: The Brute-Force Solution

- While not converged:
  - For each scan:
    - For each point:
      - For every other scan
        - Find closest point
  - Minimize error w.r.t. transforms of all scans

- Disadvantage:
  - Solve (6n)x(6n) matrix equation, where n is number of scans

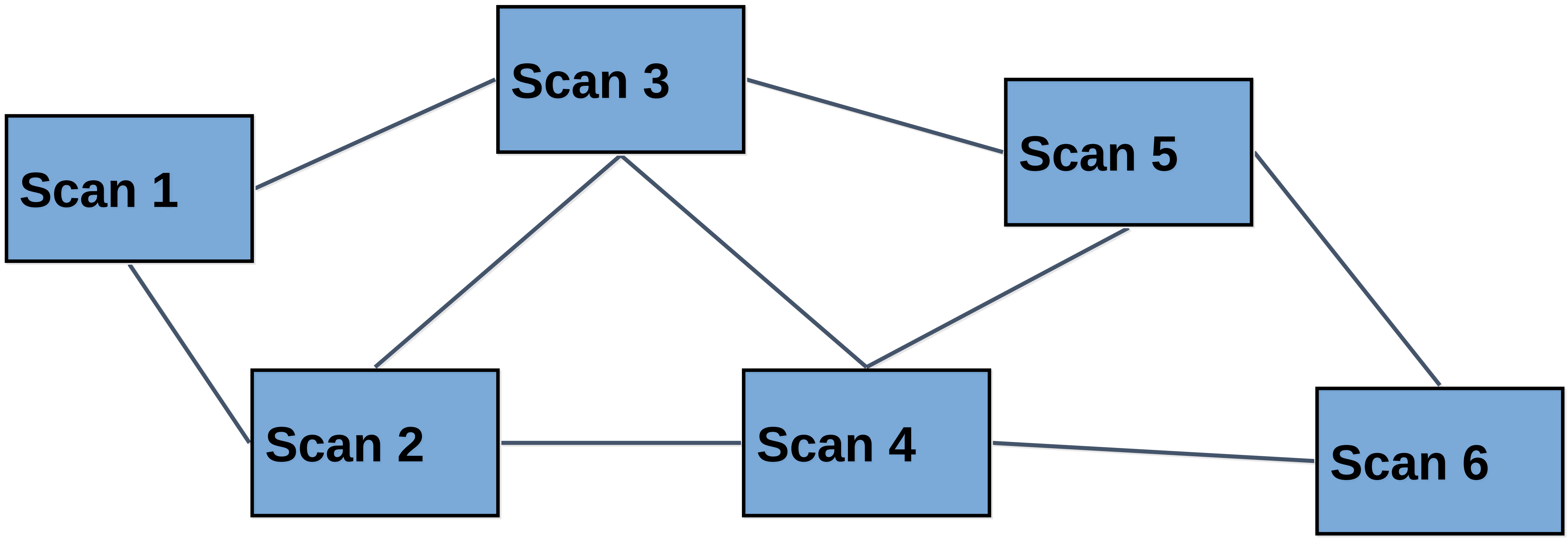
Approach #3a: Slightly Less Brute-Force Solution

- While not converged:
    - For each scan:
        - For each point:
            - For every other scan
                - Find closest point
    - Minimize error w.r.t. transforms of this scans

- Faster than previous method (matrices are 6x6) [Bergevin ’96, Benjemaa ‘97]
- Previous: Solve (6n)x(6n) matrix equation, where n is number of scans

Graph Methods

- Many globalreg algorithms create a graph of pairwise alignments between scans



Pulli’s Algorithm

- **Perform pairwise ICPs, record sample (e.g. 200) of corresponding points**
- For each scan, starting w. most connected
  - Align scan to existing set
  - While (change in error) > threshold
    - Align each scan to others
- **All alignments during globalreg phase use precomputed corresponding points**

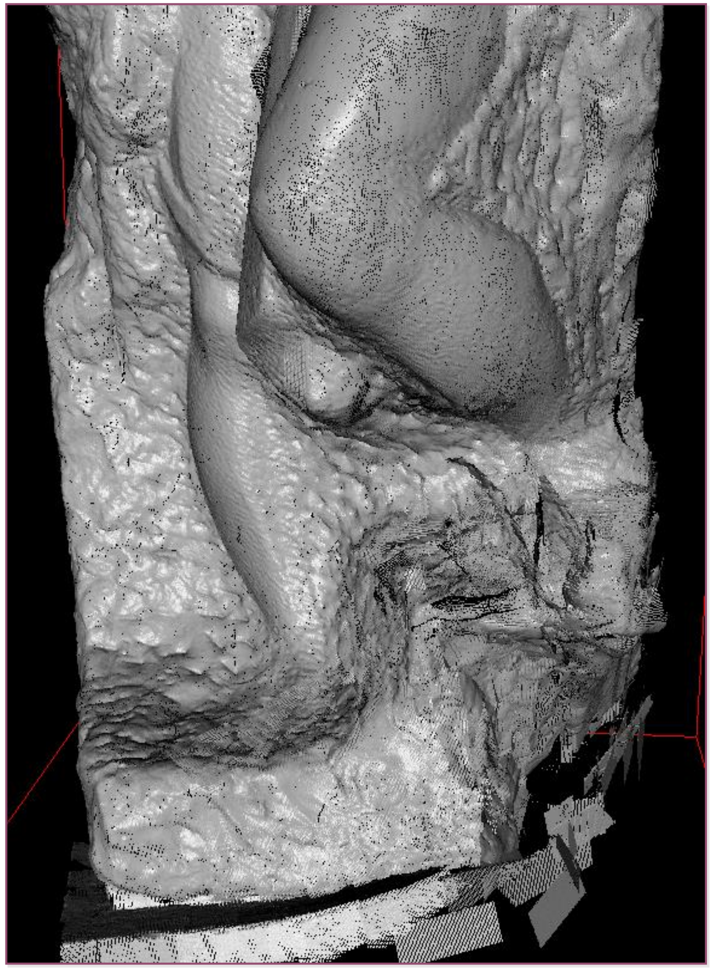
Sharp et al. Algorithm

- Perform pairwise ICPs, record only optimal rotation/translation for each
- Decompose alignment graph into cycles
- While (change in error) > tolerance
  - For each cycle:
    - Spread out error equally among all scans in the cycle
  - For each scan belonging to more than 1 cycle:
    - Assign average transform to scan

Bad ICP in Globalreg

One bad ICP can throw off the entire model!

Correct Globalreg



Globalreg Including Bad ICP

Acknowledgement

- Data-Driven Shape Anlaysis - CS468 @Stanford.edu, Spring 2014
  CSCI 621: Digital Geometry Processing SS 2017 @ USC
- 3D Scan Matching and Registration, Part II, ICCV 2005 Short Course

Literature

- Rusinkiewicz & Levoy, Efficient Variants of the ICP Algorithm, 3DIM 2001
- Chen & Medioni, “Object modeling by registration of multiple range images”, ICRA1991
- Besl & McKay: A method for registration of 3D shapes, PAMI 1992
- Horn: Closed-form solution of absolute orientation using unit quaternions, Journal Opt. Soc. Amer. 4(4), 1987
- Gelfand et al: Geometrically Stable Sampling for the ICP Algorithm, 3DIM, 2001.
- Pulli, Multiview Registration for Large Data Sets, 3DIM 1999

Thanks