Dynamic Geometry Processing

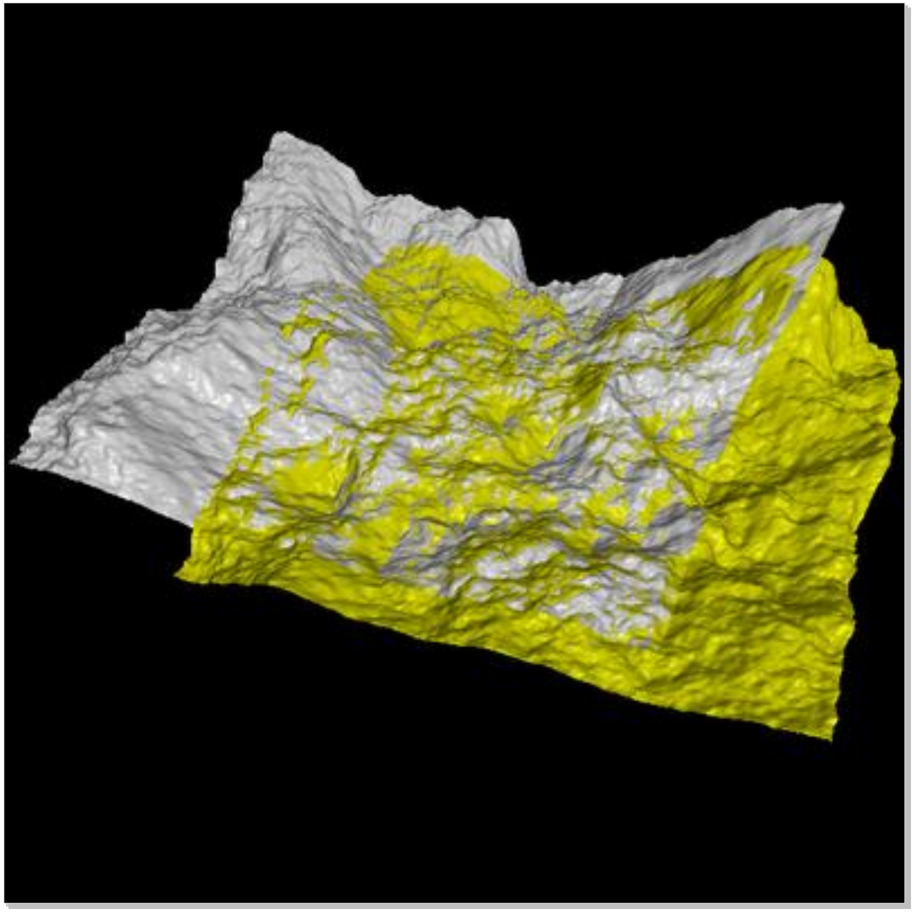
EG 2012 Tutorial

Local, Rigid, Pairwise

The ICP algorithm and its extensions

Pairwise Rigid Registration Goal

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



Outline

ICP: Iterative Closest Points

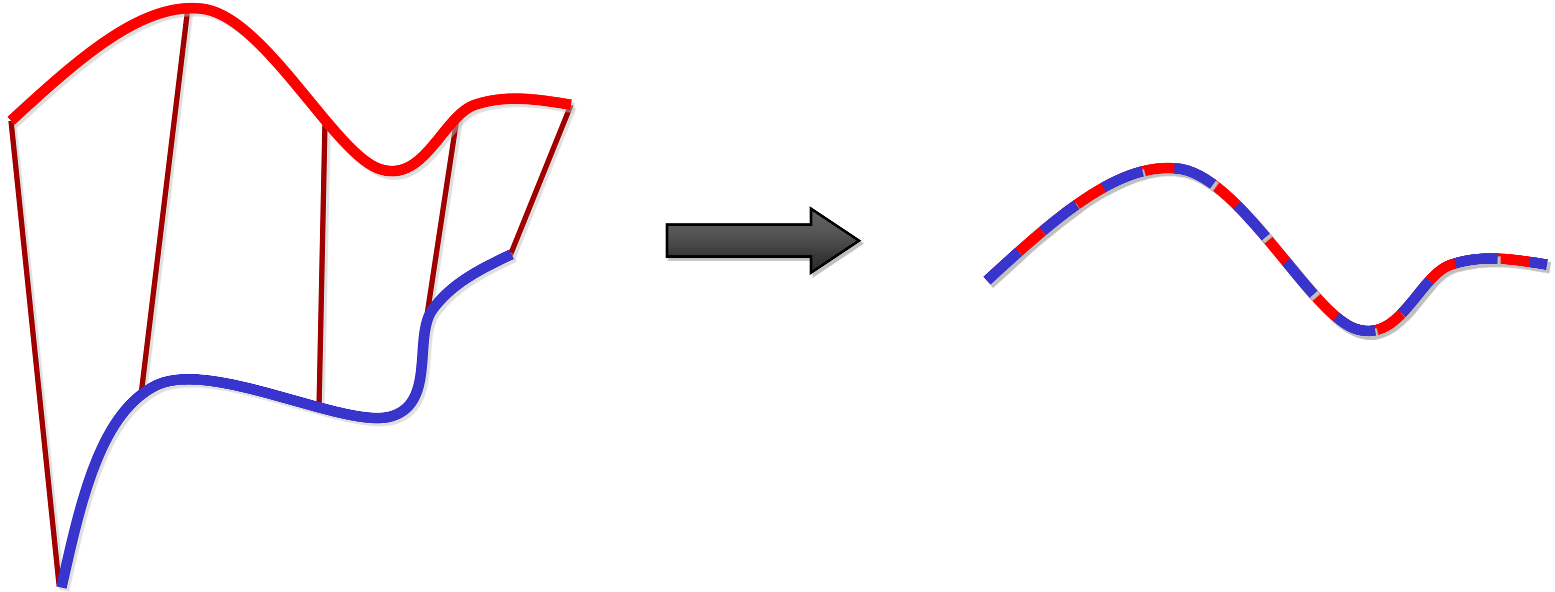
Classification of ICP variants

- Faster alignment
- Better robustness

ICP as function minimization

Aligning 3D Data

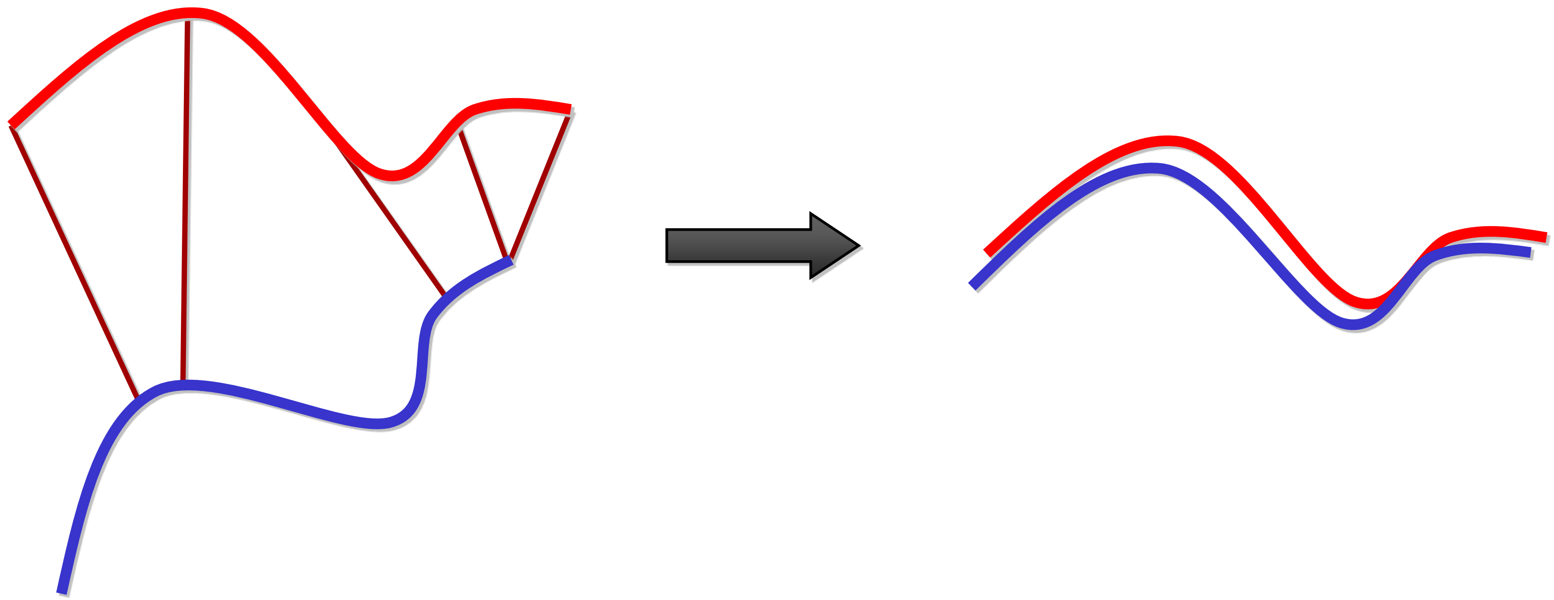
If correct correspondences are known, can find correct relative rotation/translation



Aligning 3D Data

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

Alternative: assume closest points correspond

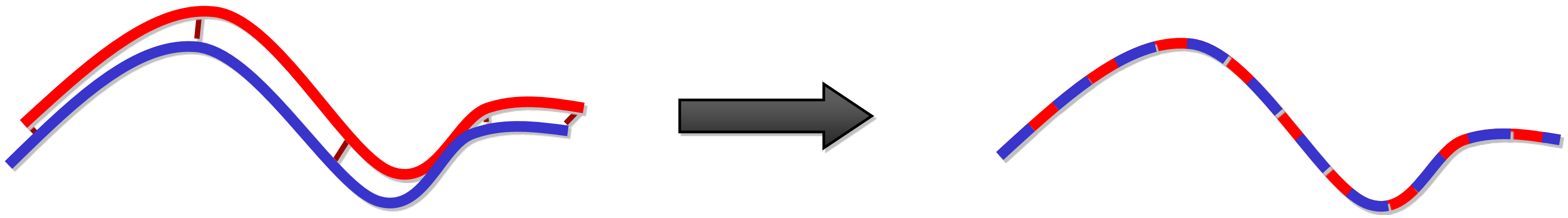


Aligning 3D Data

… and iterate to find alignment

- Iterative Closest Points (ICP) [Besl & McKay 92]

Converges if starting position "close enough"



Basic ICP

Select e.g. 1000 random points

Match each to closest point on other scan, using data structure such as *k*-d tree

Reject pairs with distance > *k* times median

Construct error function:

$$E = \sum \left| Rp_i + t - q_i \right|^2$$

Minimize (closed form solution in [Horn 87])

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

Performance of 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]

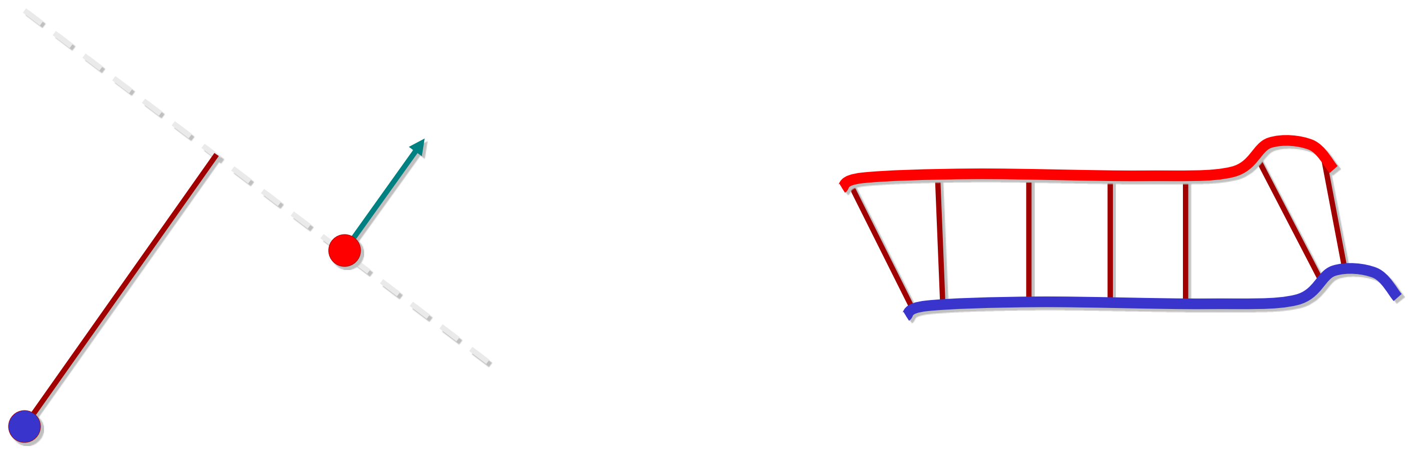
ICP Variants

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

Point-to-Plane Error Metric

Using point-to-plane distance instead of point-to-point lets flat regions slide along each other

[Chen & Medioni 91]



Point-to-Plane Error Metric

Error function:

$$E = \sum \left((Rp_i + t - q_i) \cdot n_i \right)^2$$

where R is a rotation matrix, t is translation vector

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

$$E \approx \sum \left((p_i - q_i) \cdot n_i + r \cdot (p_i \times n_i) + t \cdot n_i \right)^2, \qquad \text{where } r = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}$$

Result: overconstrained linear system

Point-to-Plane Error Metric

Overconstrained linear system

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

$$\mathbf{A} = \begin{pmatrix} \leftarrow & p_1 \times n_1 & \rightarrow & \leftarrow & n_1 & \rightarrow \\ \leftarrow & p_2 \times n_2 & \rightarrow & \leftarrow & n_2 & \rightarrow \\ & \vdots & & & \vdots & \end{pmatrix}, \qquad x = \begin{pmatrix} r_x \\ r_y \\ r_z \\ t_x \\ t_y \\ t_z \end{pmatrix}, \qquad b = \begin{pmatrix} -(p_1 - q_1) \cdot n_1 \\ -(p_2 - q_2) \cdot n_2 \\ \vdots \end{pmatrix}$$

Solve using least squares

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}x = \mathbf{A}^{\mathrm{T}}b$$

$$x = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}b$$

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: curvatures, higher-order derivatives, and other local features

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

Selecting Source Points

Use all points

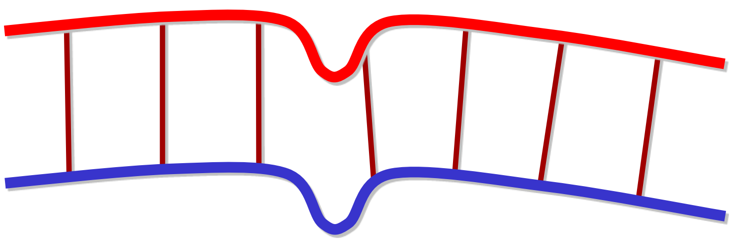
Uniform subsampling

Random sampling

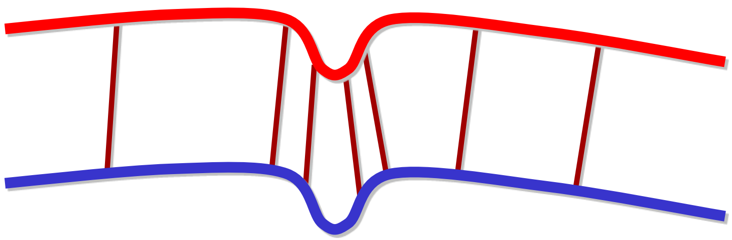
Stable sampling [Gelfand et al. 2003]

- Select samples that constrain all degrees of freedom of the rigid-body transformation

Stable Sampling



Uniform Sampling



Stable Sampling

Covariance Matrix

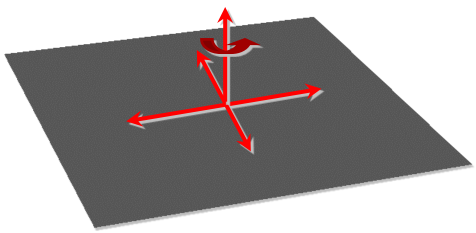
Aligning transform is given by $A^T A x = A^T b$, where

$$\mathbf{A} = \begin{pmatrix} \leftarrow & p_1 \times n_1 & \rightarrow & \leftarrow & n_1 & \rightarrow \\ \leftarrow & p_2 \times n_2 & \rightarrow & \leftarrow & n_2 & \rightarrow \\ & \vdots & & & \vdots & \end{pmatrix}, \qquad x = \begin{pmatrix} r_x \\ r_y \\ r_z \\ t_x \\ t_y \\ t_z \end{pmatrix}, \qquad b = \begin{pmatrix} -(p_1 - q_1) \cdot n_1 \\ -(p_2 - q_2) \cdot n_2 \\ \vdots \end{pmatrix}$$

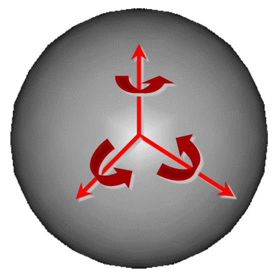
Covariance matrix $C = A^T A$ determines the change in error when surfaces are moved from optimal alignment

Sliding Directions

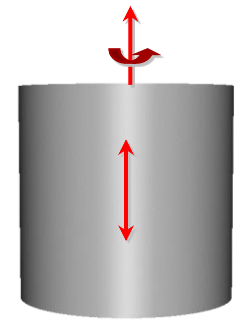
Eigenvectors of C with small eigenvalues correspond to sliding transformations



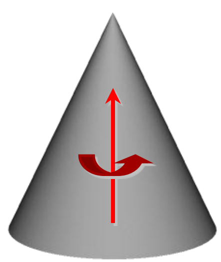
3 small eigenvalues
2 translation
1 rotation



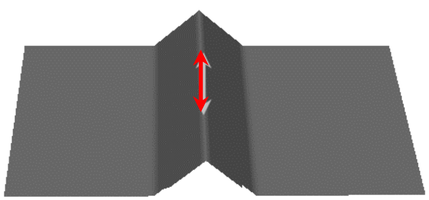
3 small eigenvalues
3 rotation



2 small eigenvalues
1 translation
1 rotation



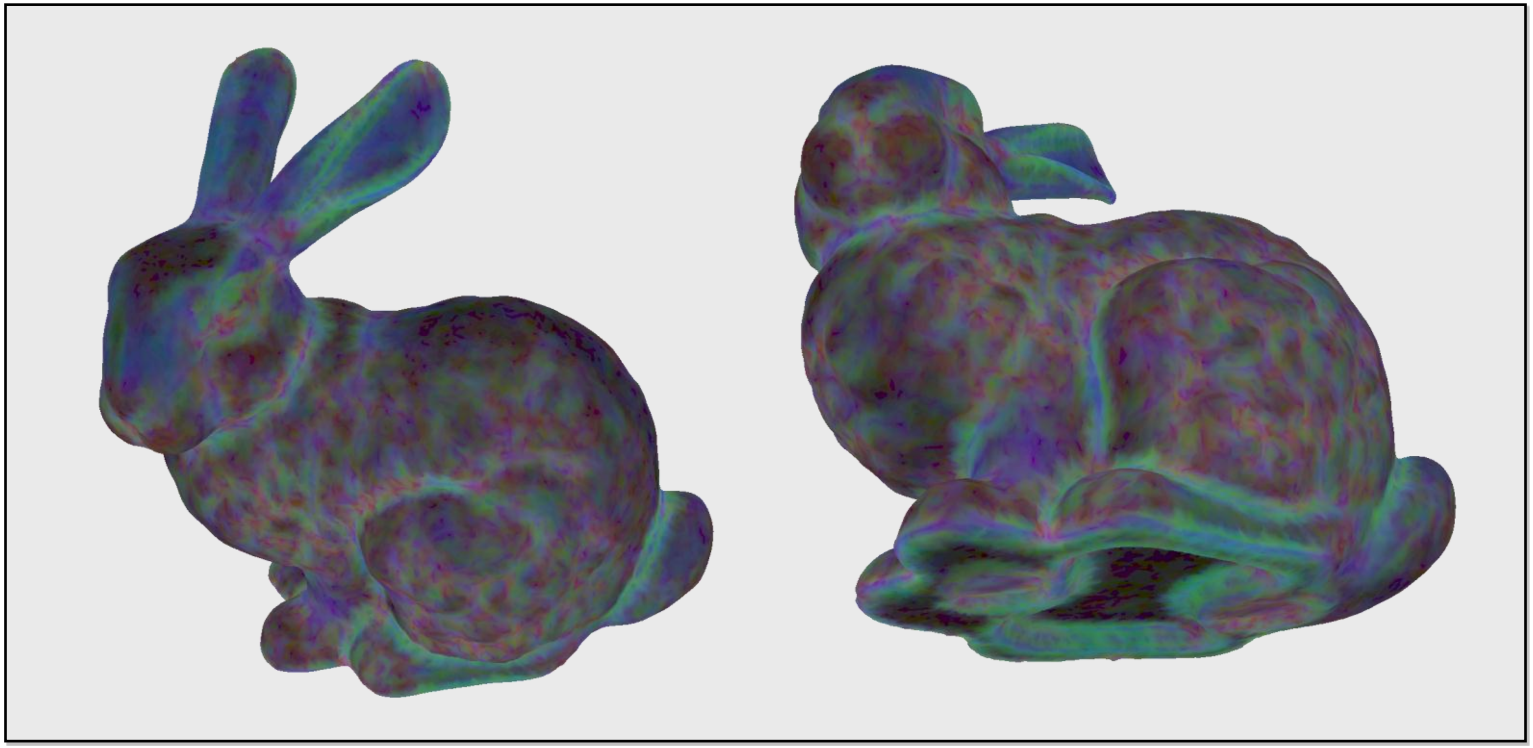
1 small eigenvalue
1 rotation



1 small eigenvalue
1 translation

[Gelfand et al. '04]

Stability Analysis



Key:

3 DOFs stable	5 DOFs stable
4 DOFs stable	6 DOFs stable

Sample Selection

Select points to prevent small eigenvalues

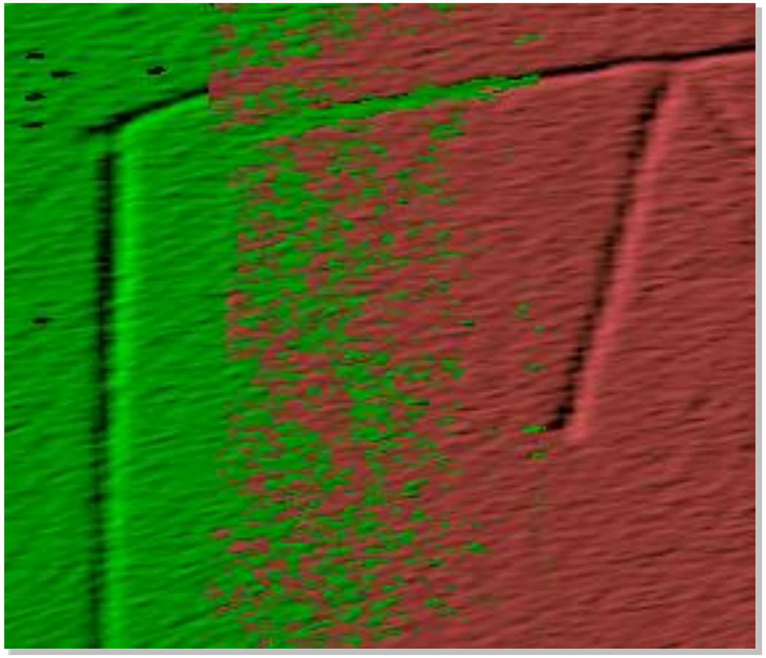
- Based on C obtained from sparse sampling

Simpler variant: normal-space sampling

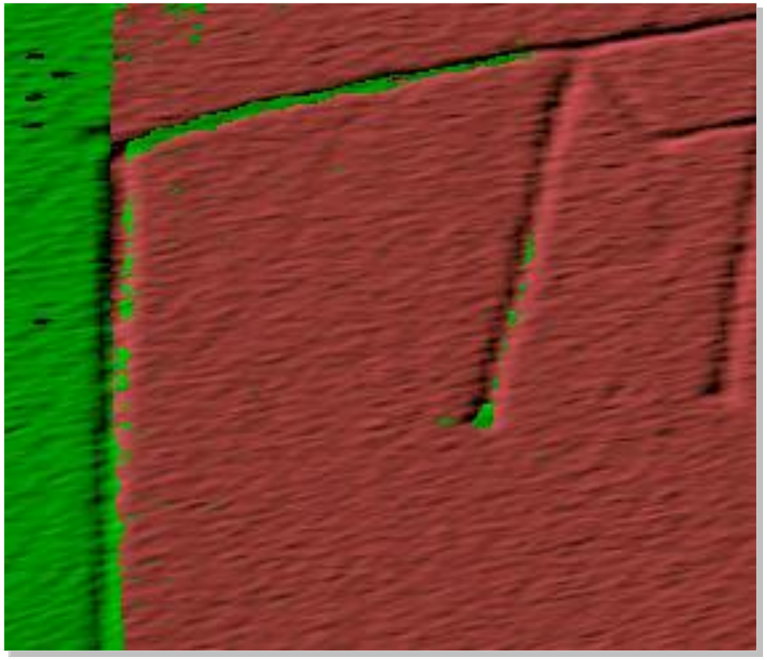
- Select points with uniform distribution of normals
- Pro: faster, does not require eigenanalysis
- Con: only constrains translation

Result

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

Improving ICP Speed

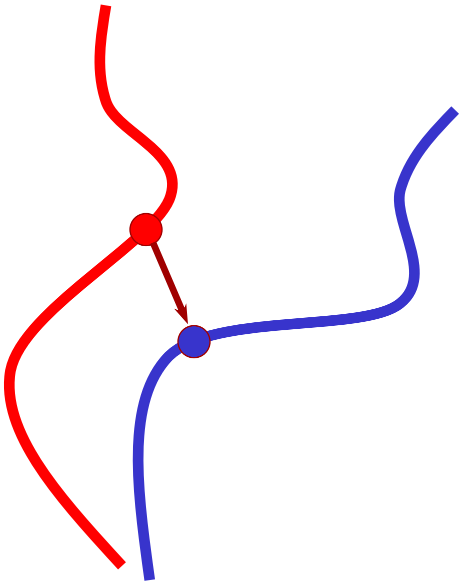
Projection-based matching

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

Finding Corresponding Points

Finding closest point is most expensive stage of the ICP algorithm

- Brute force search – O(n)
- Spatial data structure (e.g., k-d tree) – O(log n)

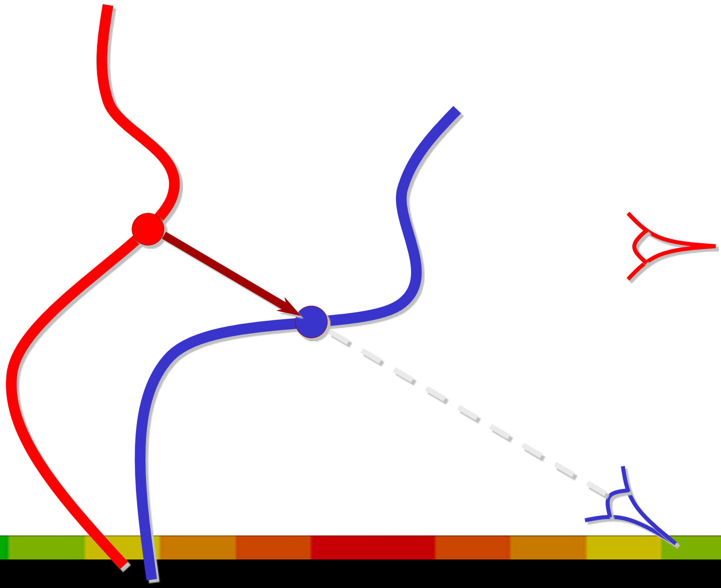


Projection to Find Correspondences

Idea: use a simpler algorithm to find correspondences

For range images, can simply project point [Blais 95]

- Constant-time
- Does not require precomputing a spatial data structure

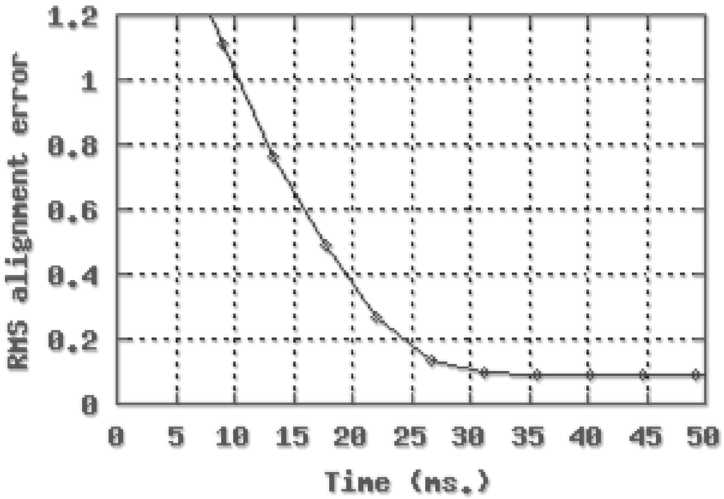


Projection-Based Matching

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



Application

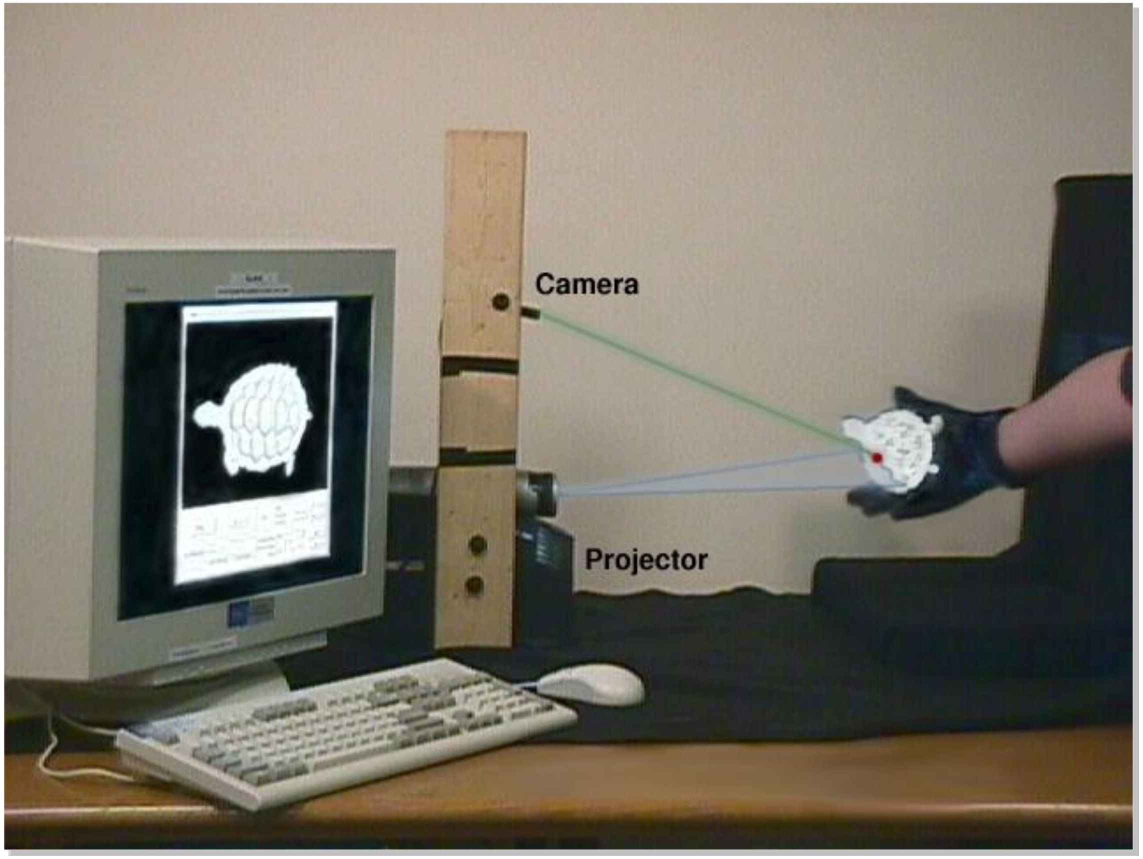
Given:

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

Result: 3D model acquisition

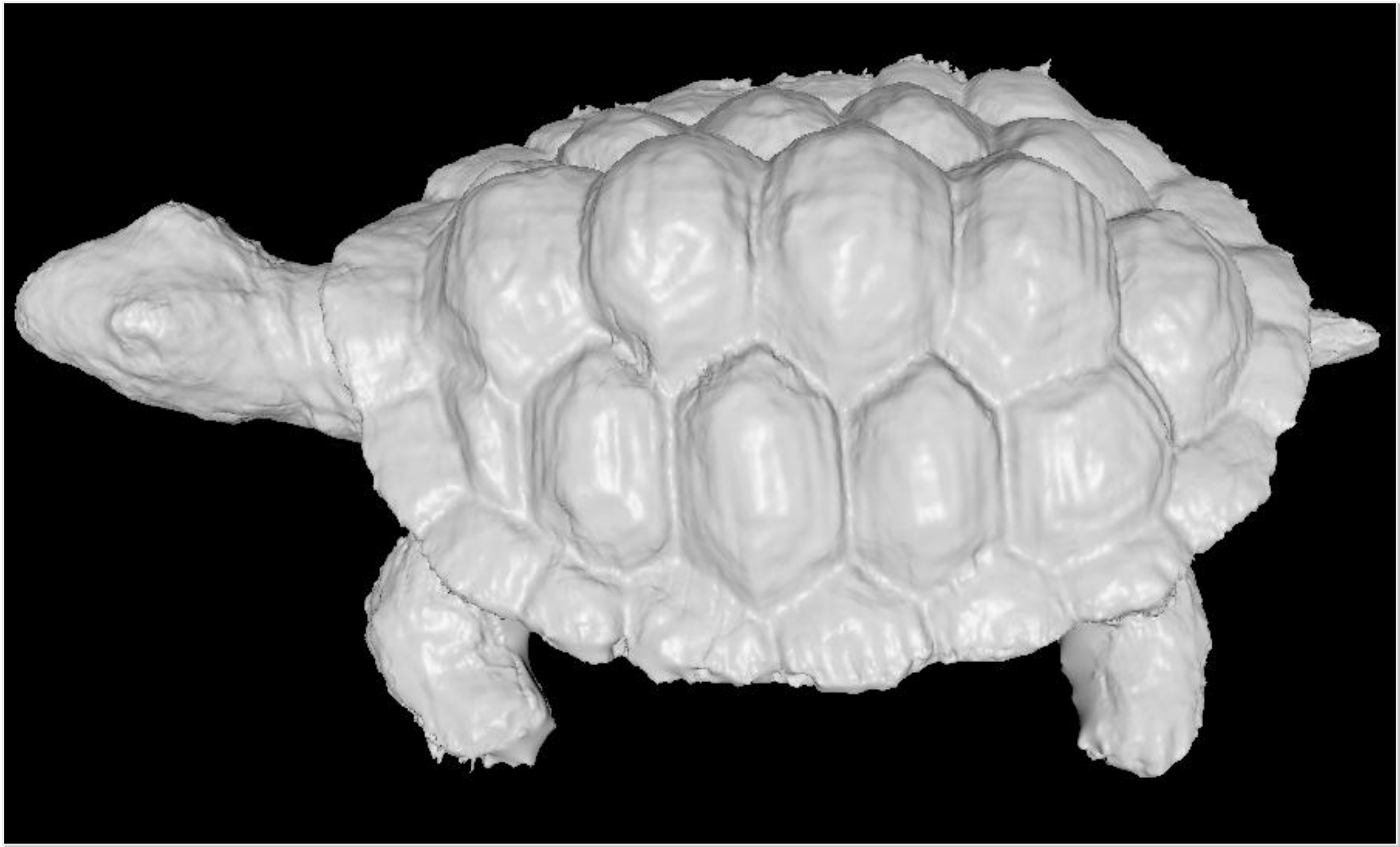
- Tight feedback loop with user
- Can see and fill holes while scanning

Scanner Layout



Photograph

Real-Time Result



Theoretical Analysis of ICP Variants

One way of studying performance is via empirical tests on various scenes

How to analyze performance analytically?

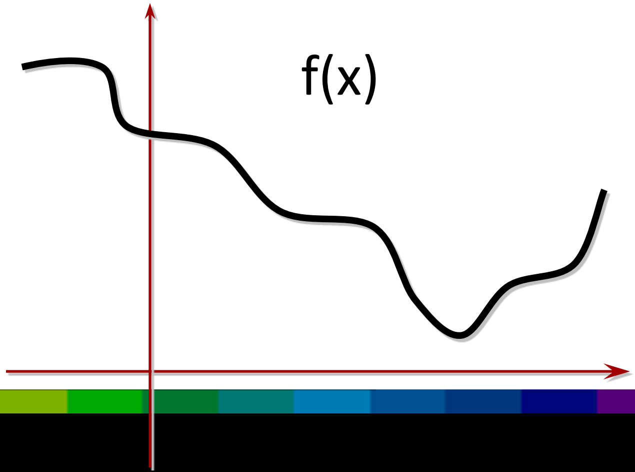
For example, when does point-to-plane help? Under what conditions does projection-based matching work?

What Does ICP Do?

Two ways of thinking about ICP:

- Solving the correspondence problem
- Minimizing point-to-surface squared distance

ICP is like (Gauss-) Newton method on an approximation of the distance function

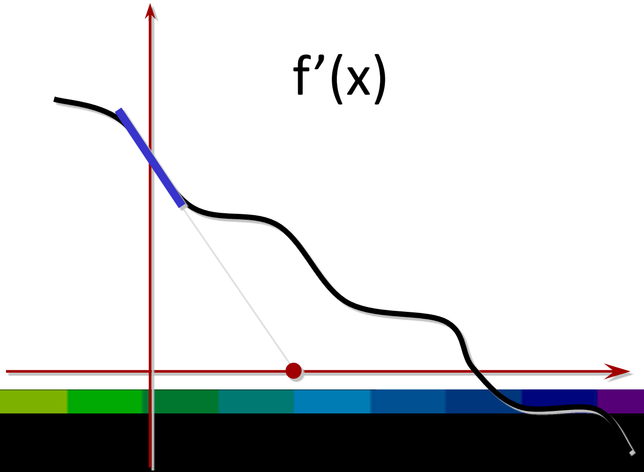


What Does ICP Do?

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



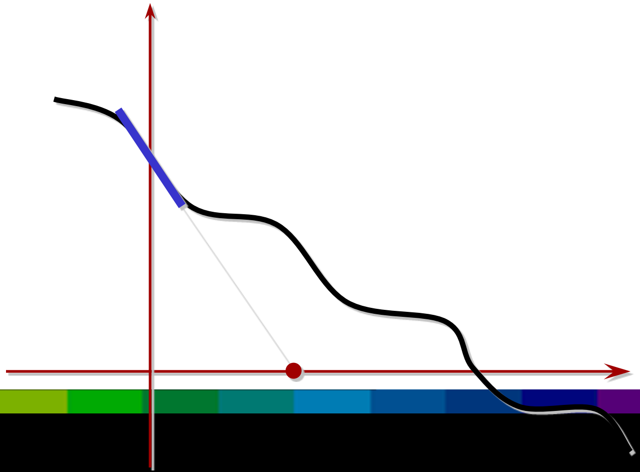
What Does ICP Do?

Two ways of thinking about ICP:

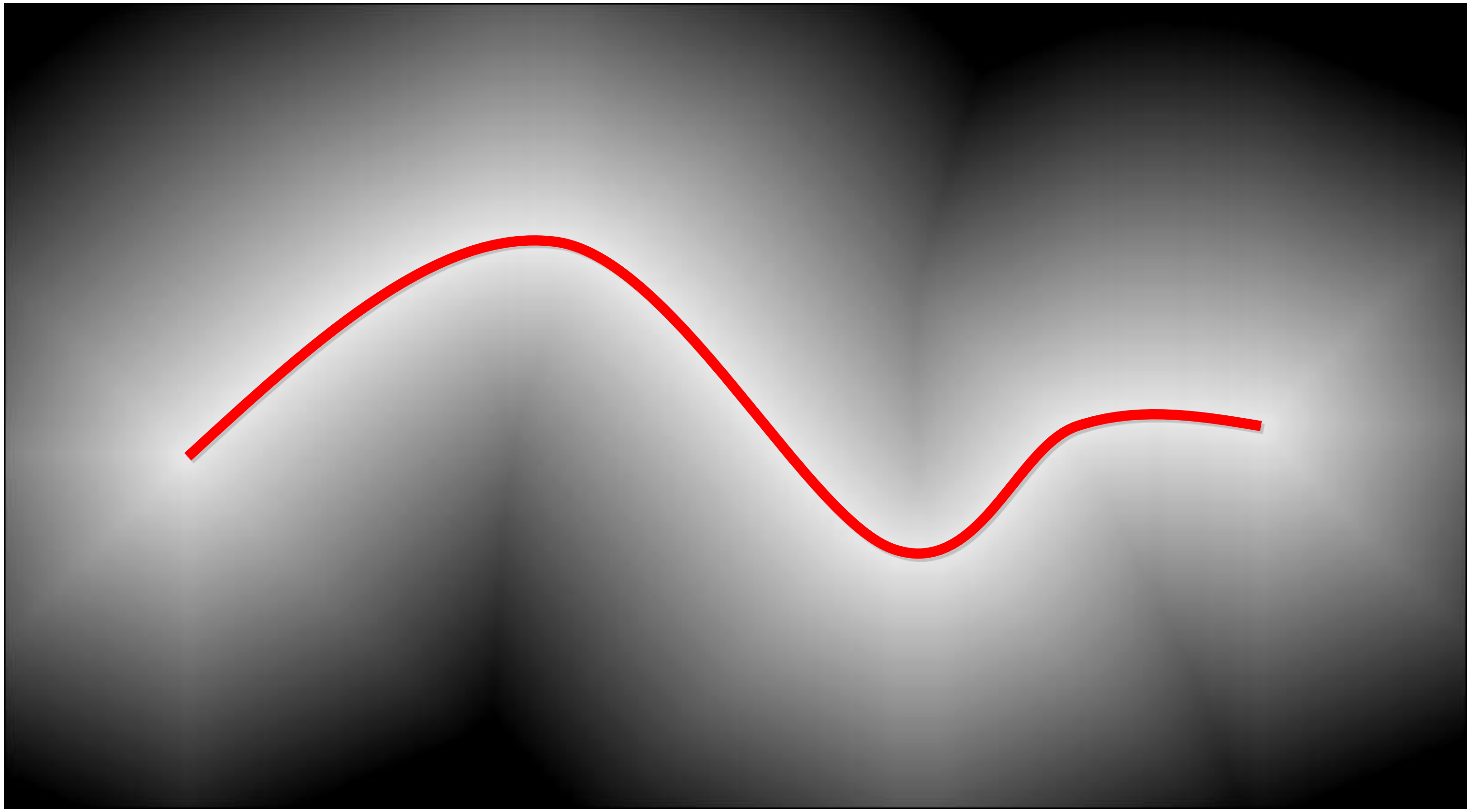
- Solving the correspondence problem
- Minimizing point-to-surface squared distance

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

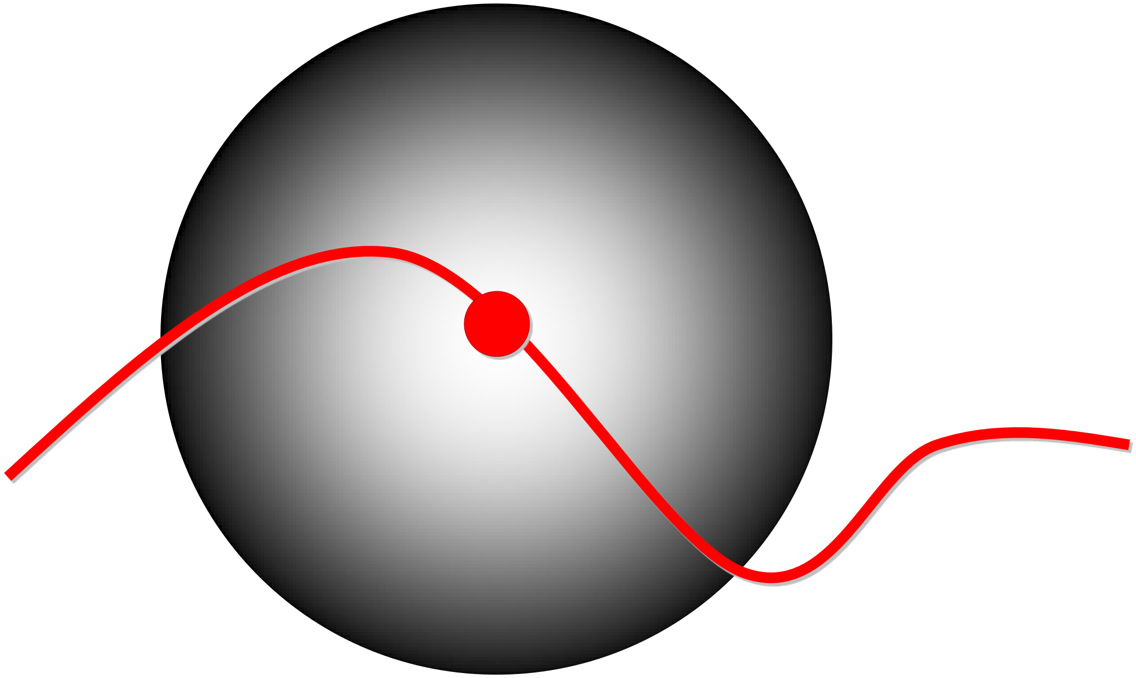
- ICP variants affect shape of global error function or local approximation



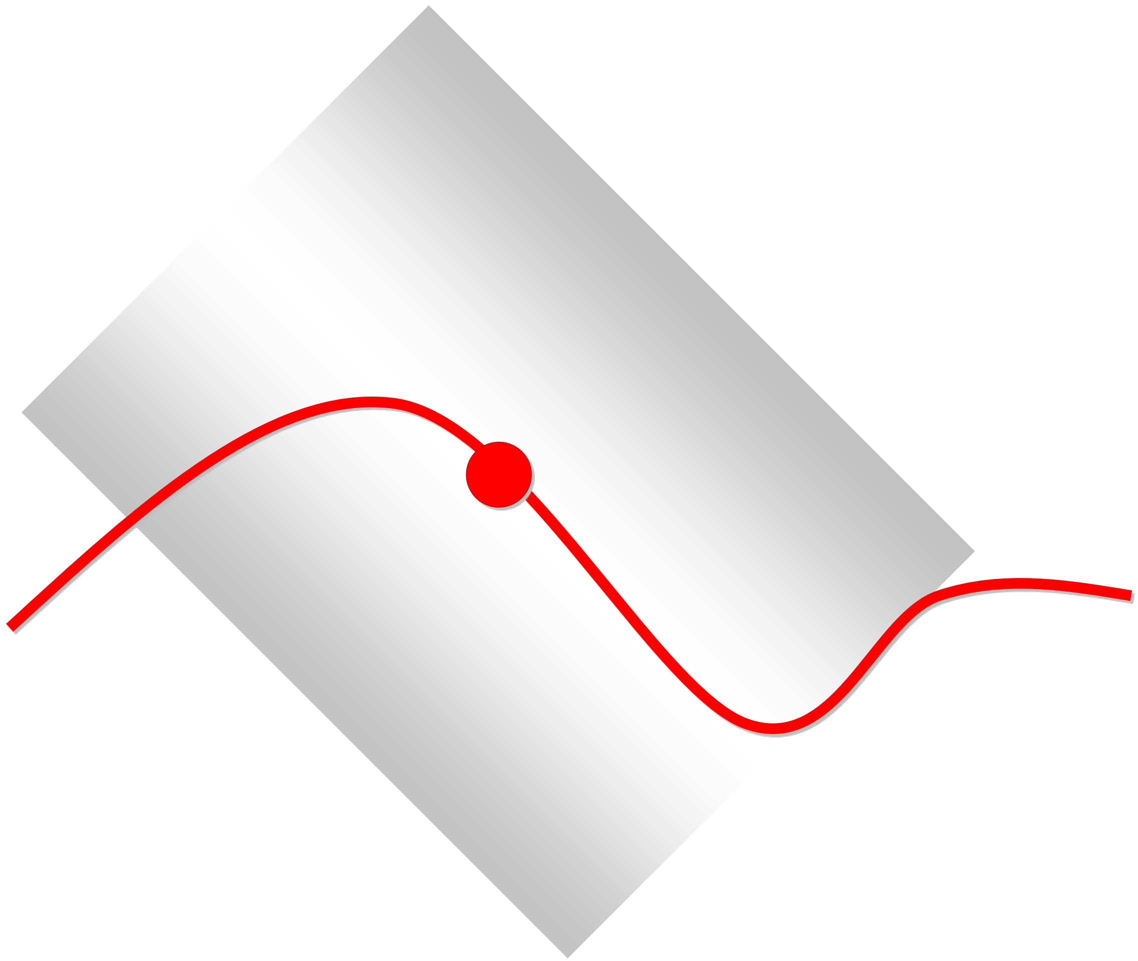
Point-to-Surface Distance



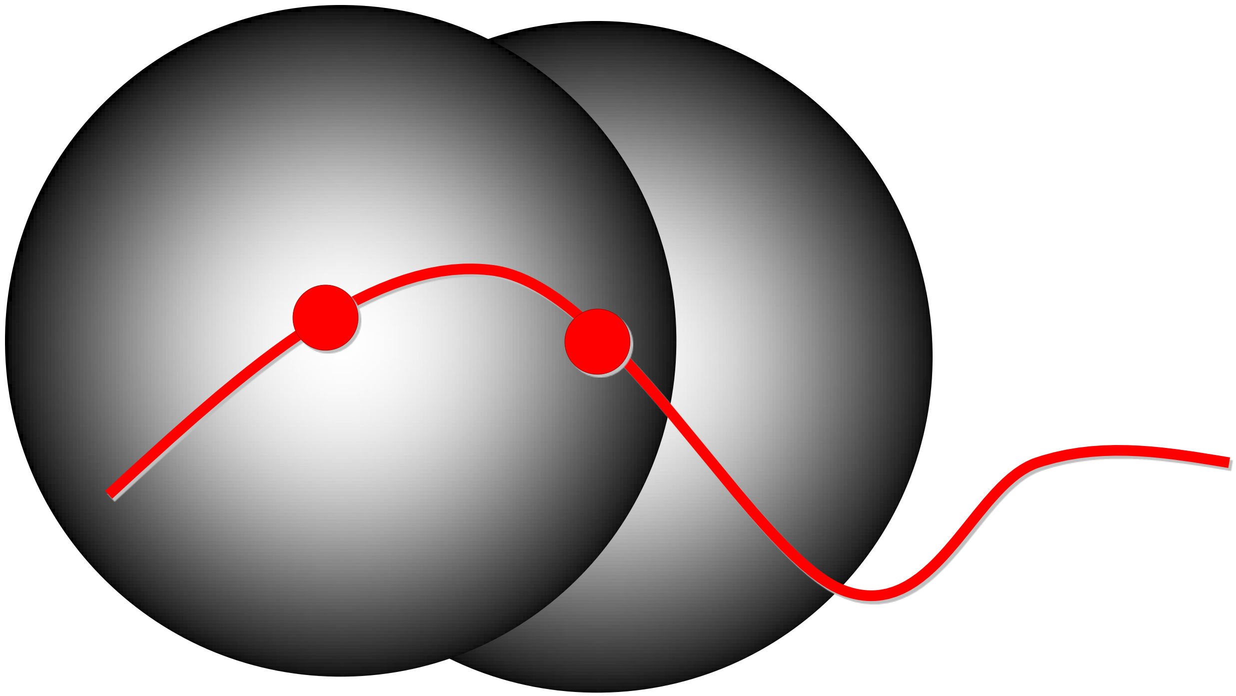
Point-to-Point Distance



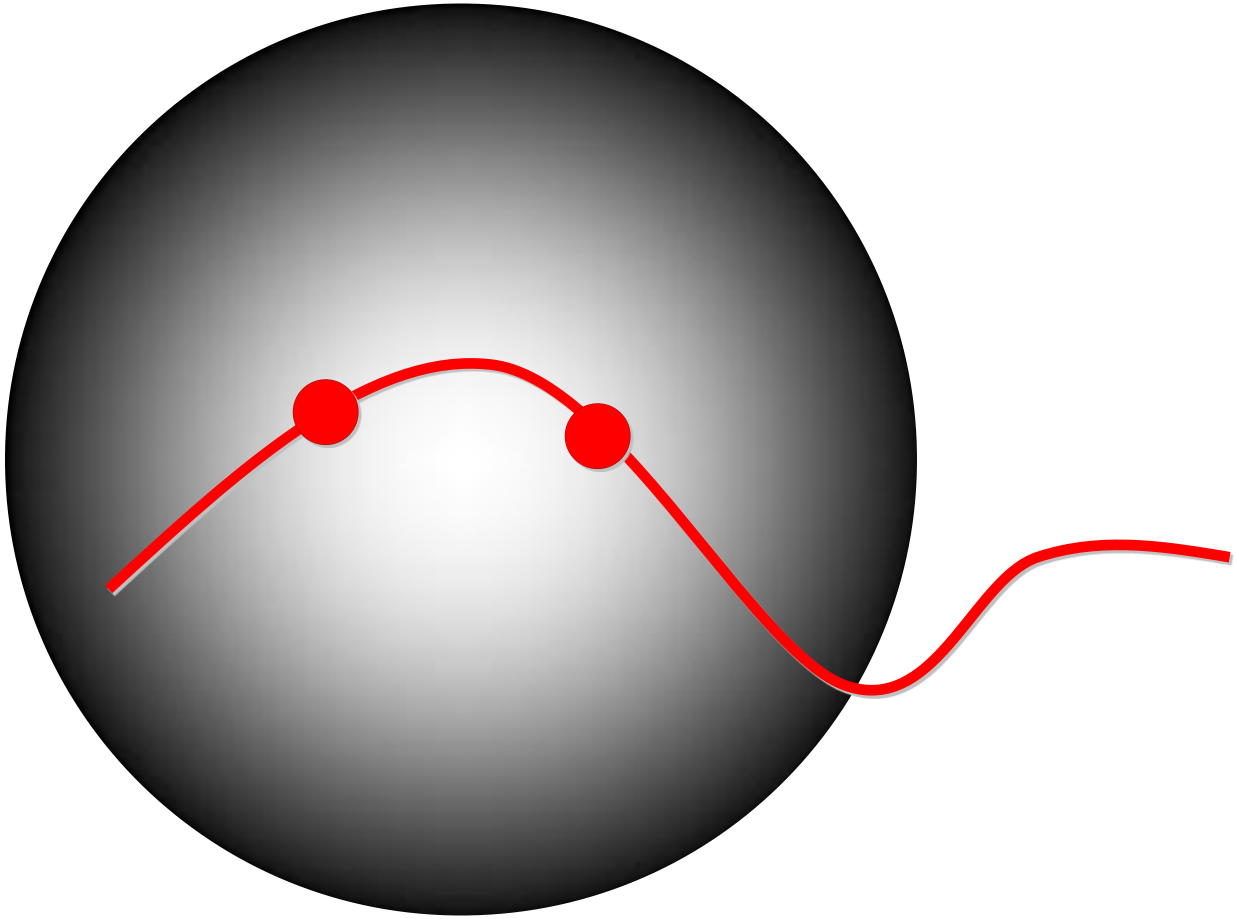
Point-to-Plane Distance



Point-to-Multiple-Point Distance



Point-to-Multiple-Point Distance



Soft Matching and Distance Functions

Soft matching equivalent to standard ICP on (some) filtered surface

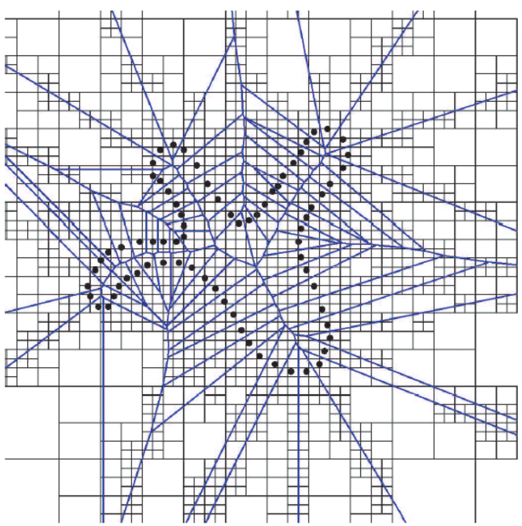
Produces filtered version of distance function
$\Rightarrow$ fewer local minima

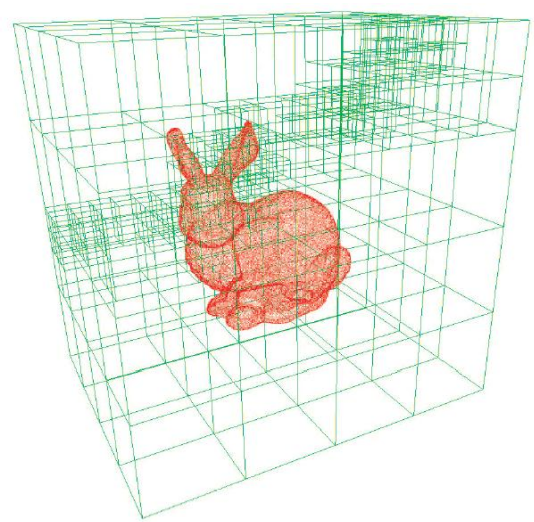
Multiresolution minimization [Turk & Levoy 94]
or softassign with simulated annealing
(good description in [Chui 03])

Mitra et al.'s Optimization

Precompute piecewise-quadratic approximation to distance field throughout space

Store in "d2tree" data structure





[Mitra et al. 2004]

Mitra et al.'s Optimization

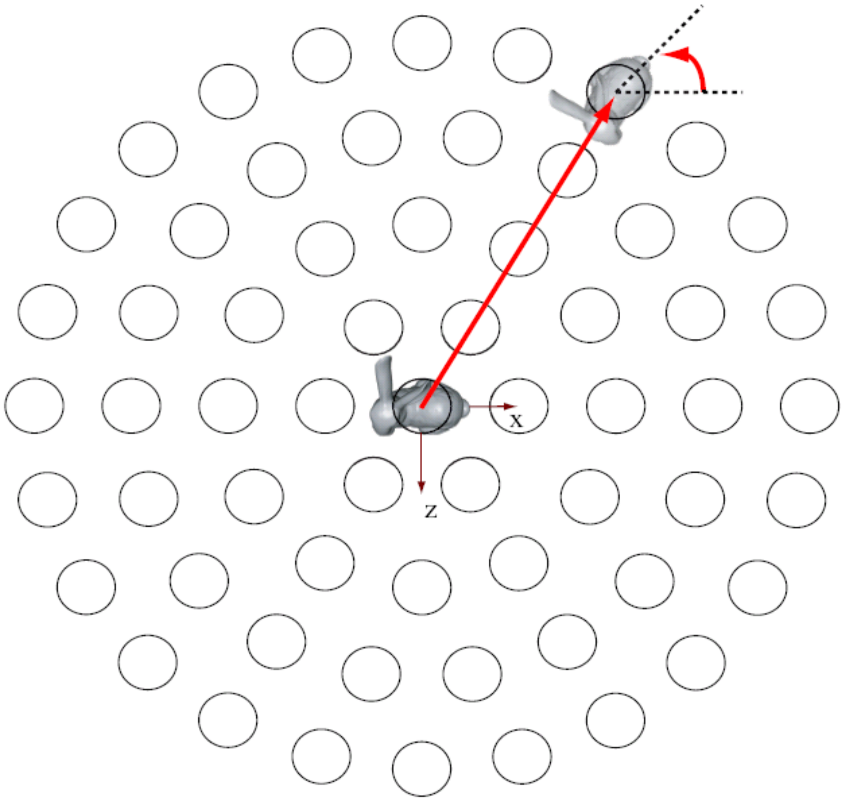
Precompute piecewise-quadratic approximation to distance field throughout space

Store in "d2tree" data structure

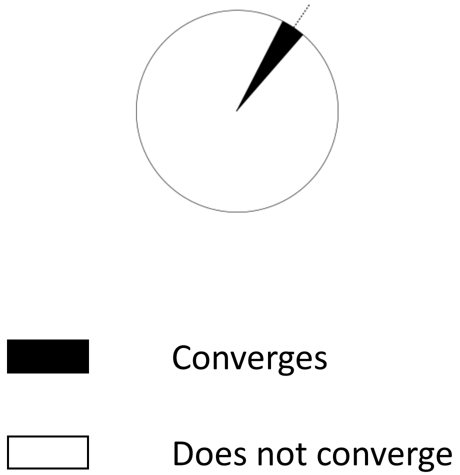
At run time, look up quadratic approximants and optimize using Newton's method

- More robust, wider basin of convergence
- Often fewer iterations, but more precomputation

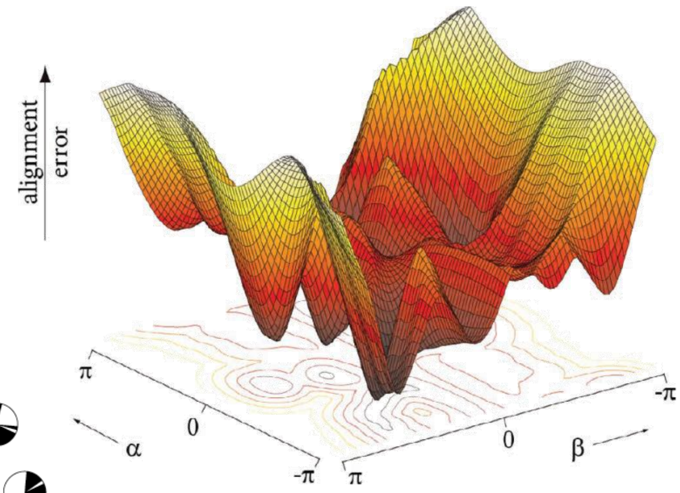
Convergence Funnel

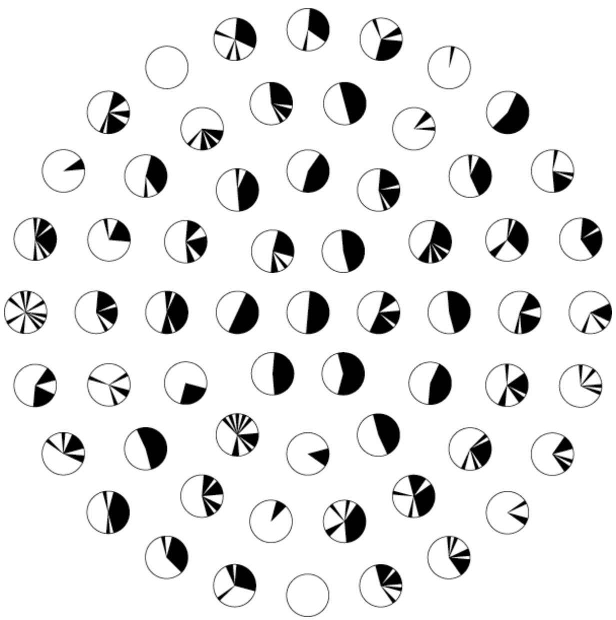


Translation in x-z plane.
Rotation about y-axis.

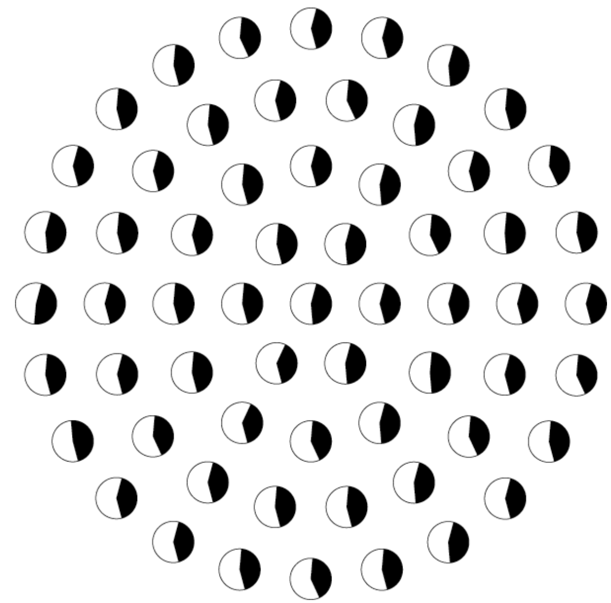


Convergence Funnel





Plane-to-plane ICP



distance-field formulation