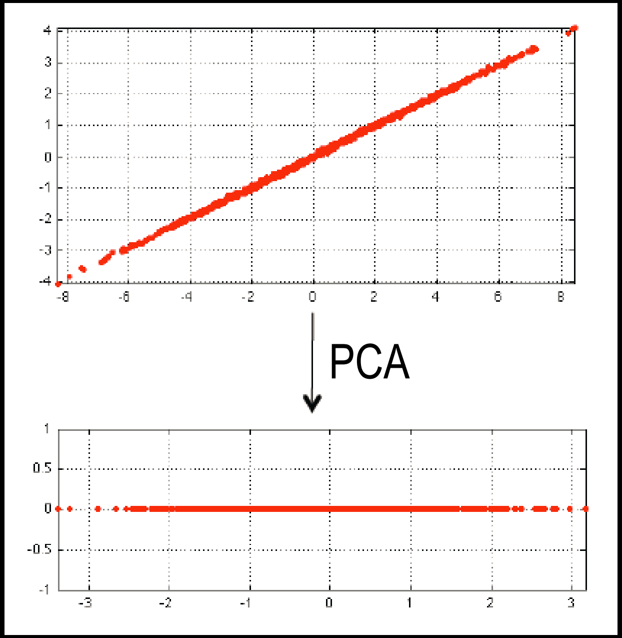
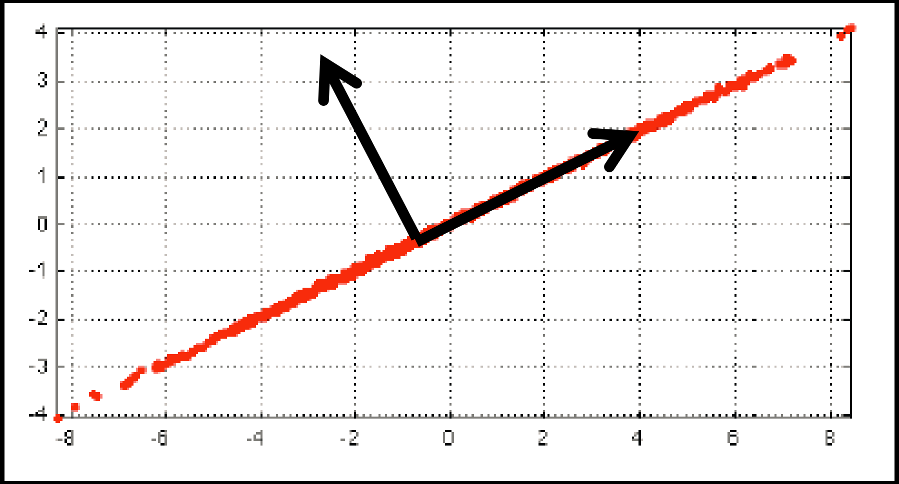
Point Cloud Model Fitting and Registration

Using examples from Tao Ju and pointclouds.org

Last time…

- We saw how to use PCA for dimensionality reduction





- We can also use PCA to fit linear models to data
- What if we want to use non-linear models?
- What if our model is just a reference point cloud?

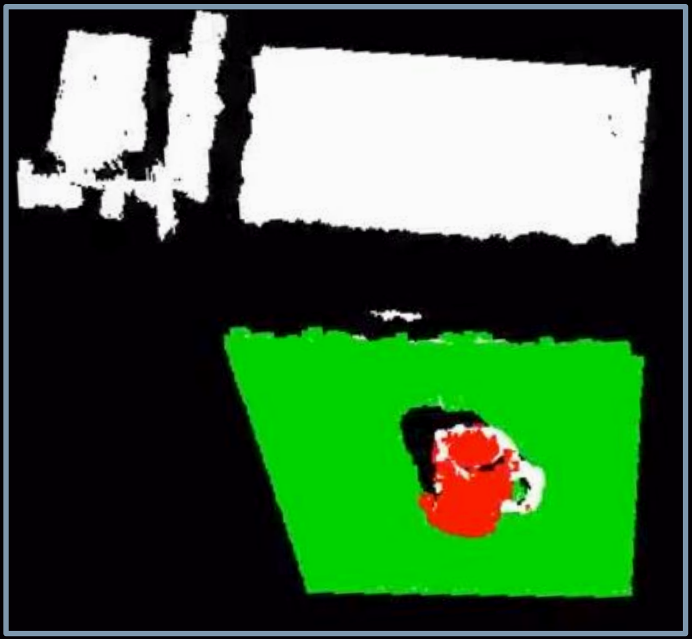
Outline

- RANSAC
- ICP
- Point Cloud Features

Fitting Non-Linear Models

- Want to fit a parametrized model to points
  - E.g. fit a plane to points (can do with PCA)
  - PCA won't work for non-linear models (e.g. cylinder)
- Problem: outliers (e.g. noise)
  - Don't know which points to fit to!

3D points from laser scan data:



Cylinder
Table
Other stuff

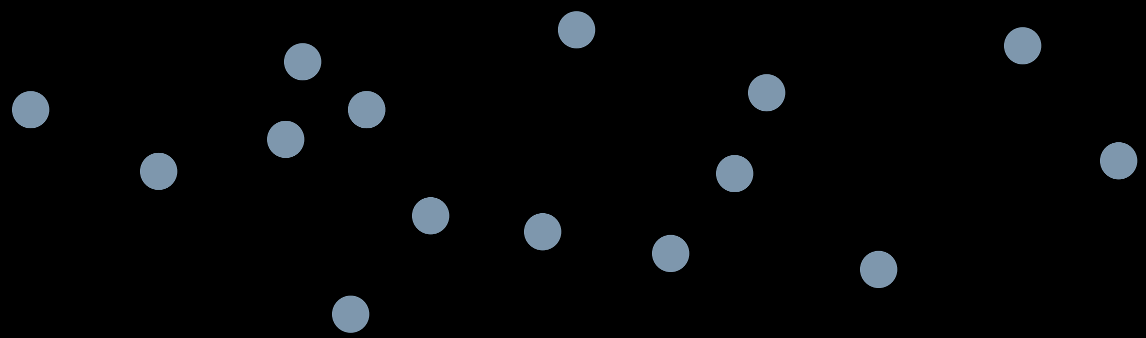
RANSAC Algorithm Sketch

- RANdom SAmple Consensus (RANSAC) samples models and returns the one with the best fit

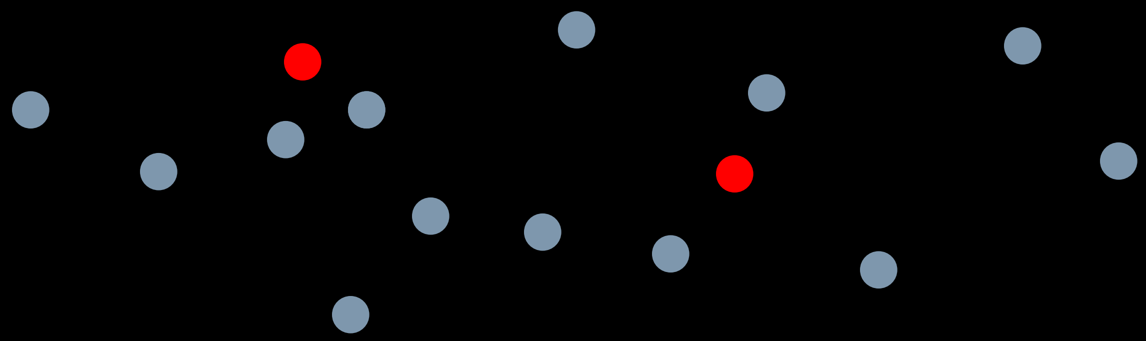
```
Input: Set of Points P, model type
Output: Model parameters
```

```
For some number of iterations
  1. Pick a random subset of points
  2. Fit the model to these random points
  3. Compute how many other points are close to the
model and how far they are
  4. If this is the best model so far, save it
Return best model found
```

RANSAC Line fitting example

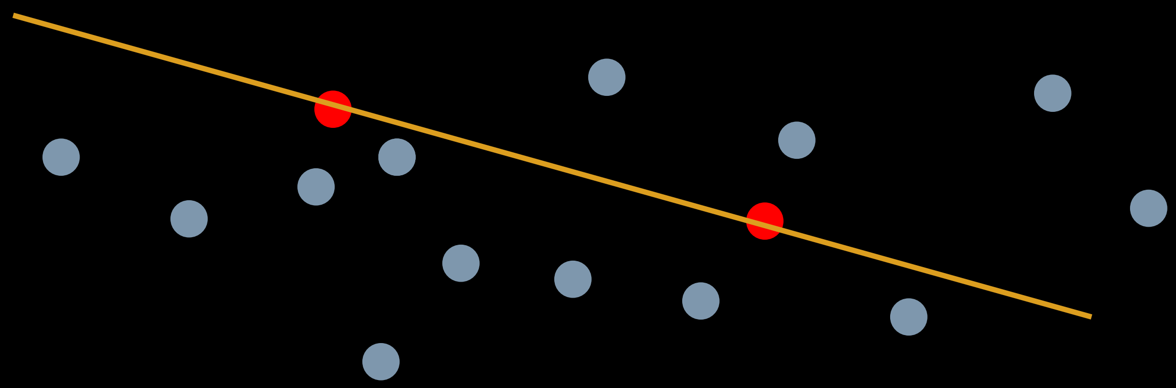


RANSAC Line fitting example



Pick R (hypothetical inliers)

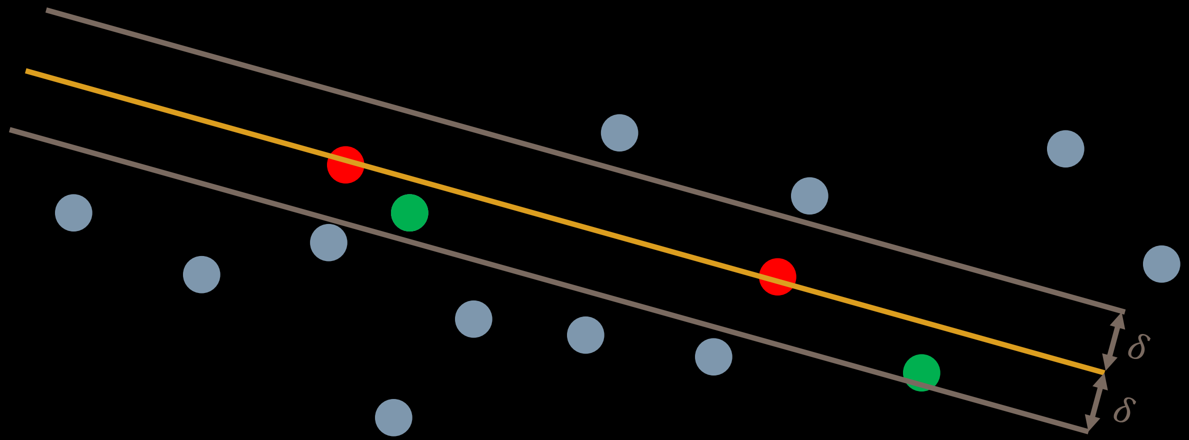
RANSAC Line fitting example



Pick R (hypothetical inliers)

Fit Model to R

RANSAC Line fitting example

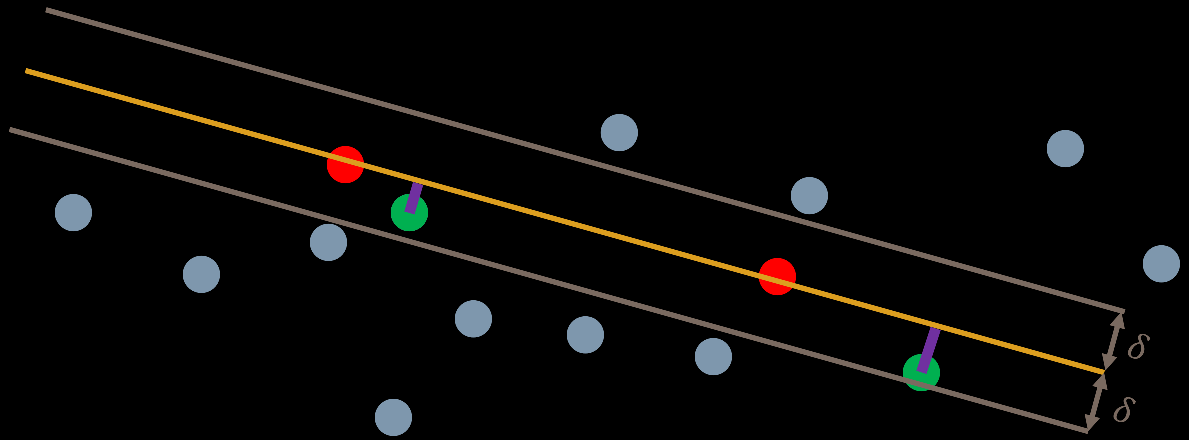


Pick R (hypothetical inliers)

Fit Model to R

Find C (consensus set)

RANSAC Line fitting example



Pick R (hypothetical inliers)

Fit Model to R

Find C (consensus set)

Compute Error of Model on $C \cup R$

RANSAC Algorithm

Input: Set of Points P, model type, K: # of iterations, δ: threshold for inliers, N: minimum number of consensus points required

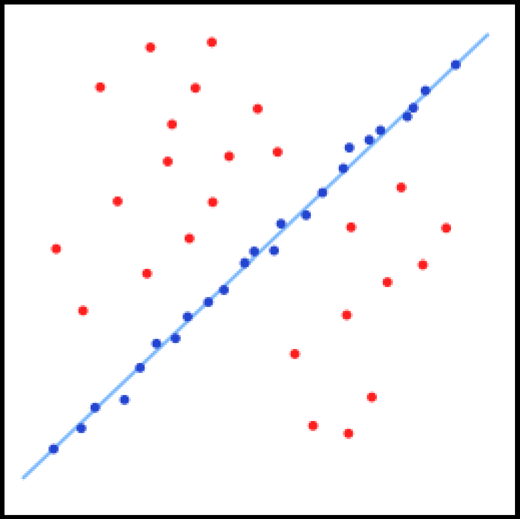
Output: Model parameters θ

```
e_best ← ∞
For i ∈ {1,2,…,K}
    Pick a random subset R ⊂ P          //R is the set of hypothetical inliers (enough to fit model)
    θ ←Fit(model, R)                    //θ are the model parameters
    C ← {∅}                             //C is the consensus set
    For p ∈ P\R                         //For all points that weren't used yet
        If Error(p, model(θ))< δ        //Check if p is close to the model prediction
            C ← C ∪ p                   //Add p to the consensus set
    If |C| > N                          //If we have enough consensus points
        θ ←Fit(model, R ∪ C)            //Re-fit the model parameters
        e_new ← Error(R ∪ C, model(θ))  //Get new error
        If e_new < e_best               //If this is the best model so far, save it
            e_best ← e_new
            θ_best ← θ
return θ_best
```

RANSAC Example

- Line fitting with extreme noise:

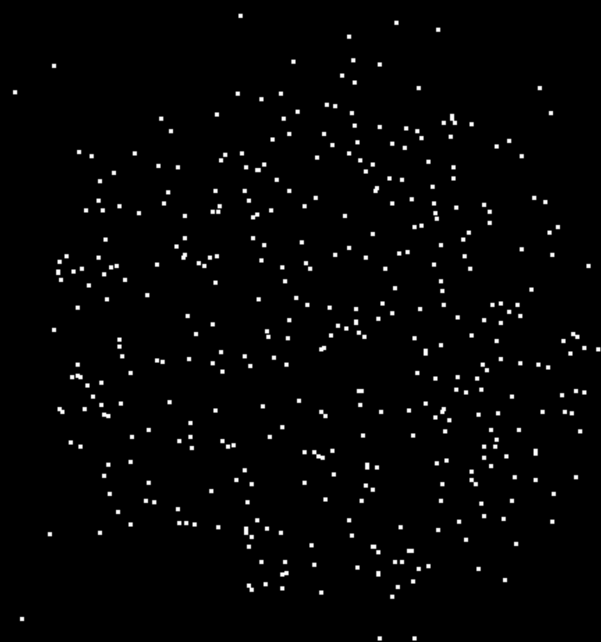
Input data



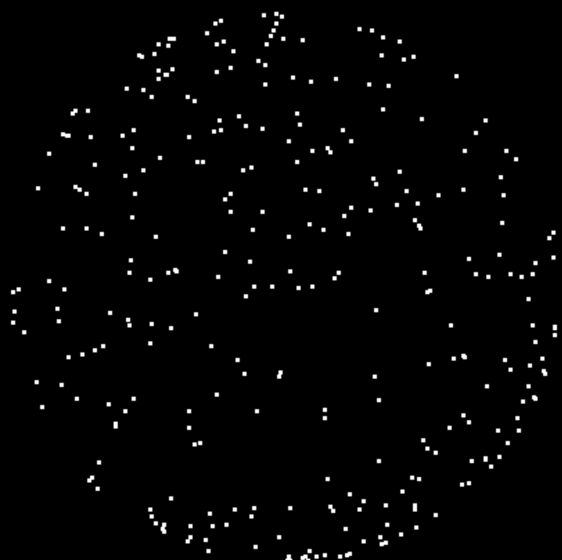
RANSAC output model with inliers in blue

RANSAC Example

- Fitting a sphere surface

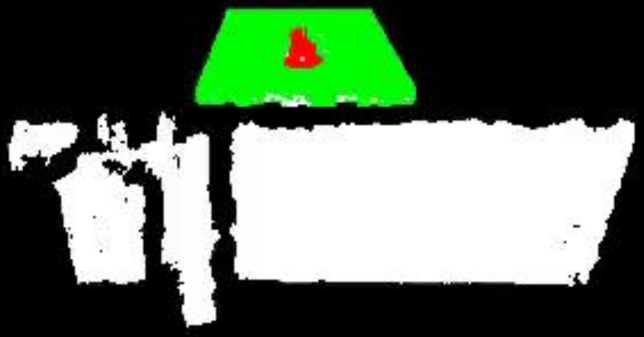


Input data



Inliers of RANSAC output model

RANSAC Example

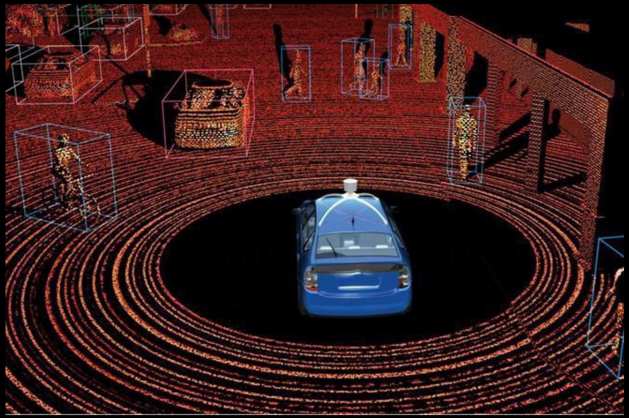


RANSAC Advantages and Disadvantages

- Advantages
  - Fitting is robust to extreme noise in the data
  - Model type can be anything, as long a there is a `Fit(model,data)` function
- Disadvantages
  - Solution may not be optimal if number of iterations is too small
  - Thresholds (δ and N) are problem-specific
- Time vs. accuracy trade-off
  - More iterations increase computation time but make it more likely a good model will be produced
- The `Fit(model,data)` function should be fast!
  - Need to evaluate it at least once per iteration

Going Further: Fitting Multiple Models

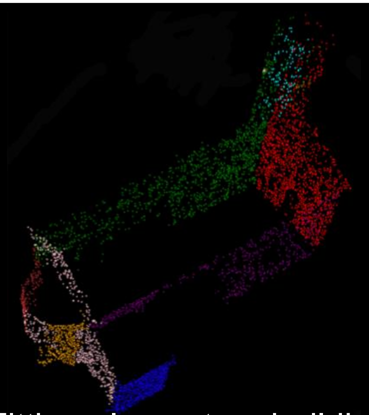
- What if we want to find multiple instances of a model in a point cloud?
- What is a simple way to do this?



Self-driving Example: Find all the cars in the point cloud

- A better way:

  - Represent each point by the set of random models that fit it
  - Hierarchically cluster points that belong to the same model

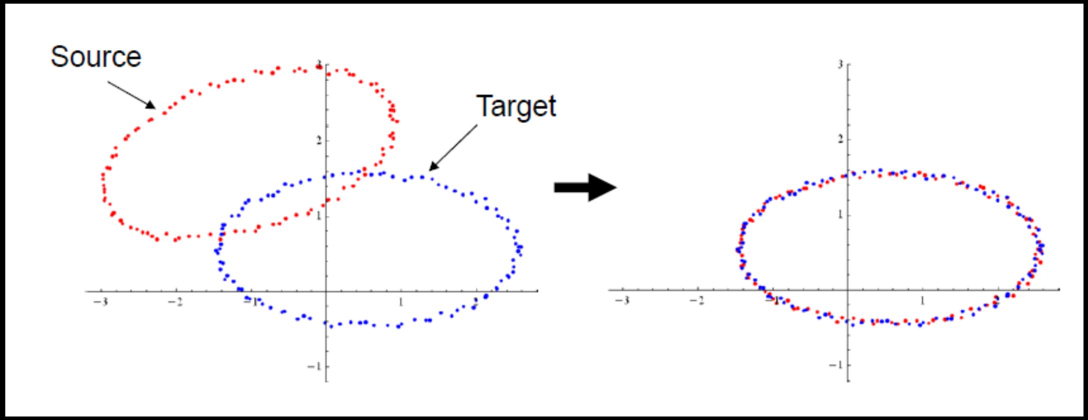


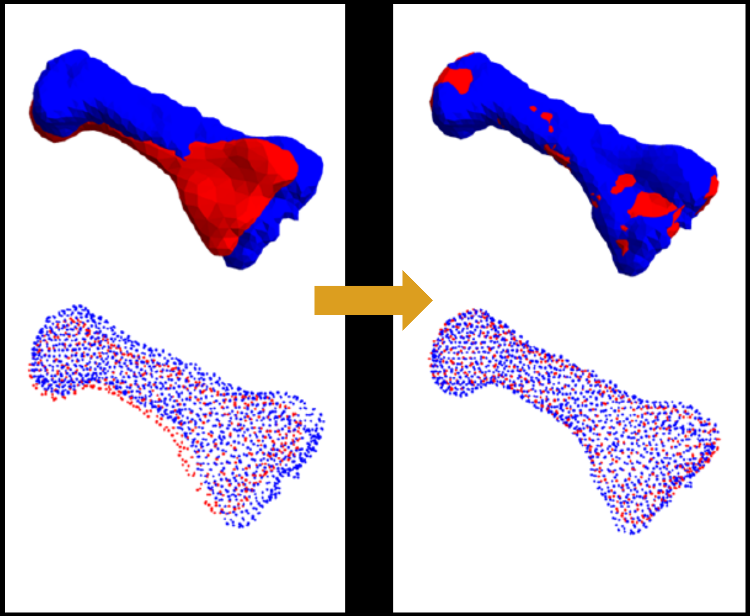
Fitting planes to a building

Toldo, R. and Fusiello, A. Robust Multiple Structures Estimation with J-Linkage. In ECCV, 2008.

Point set registration

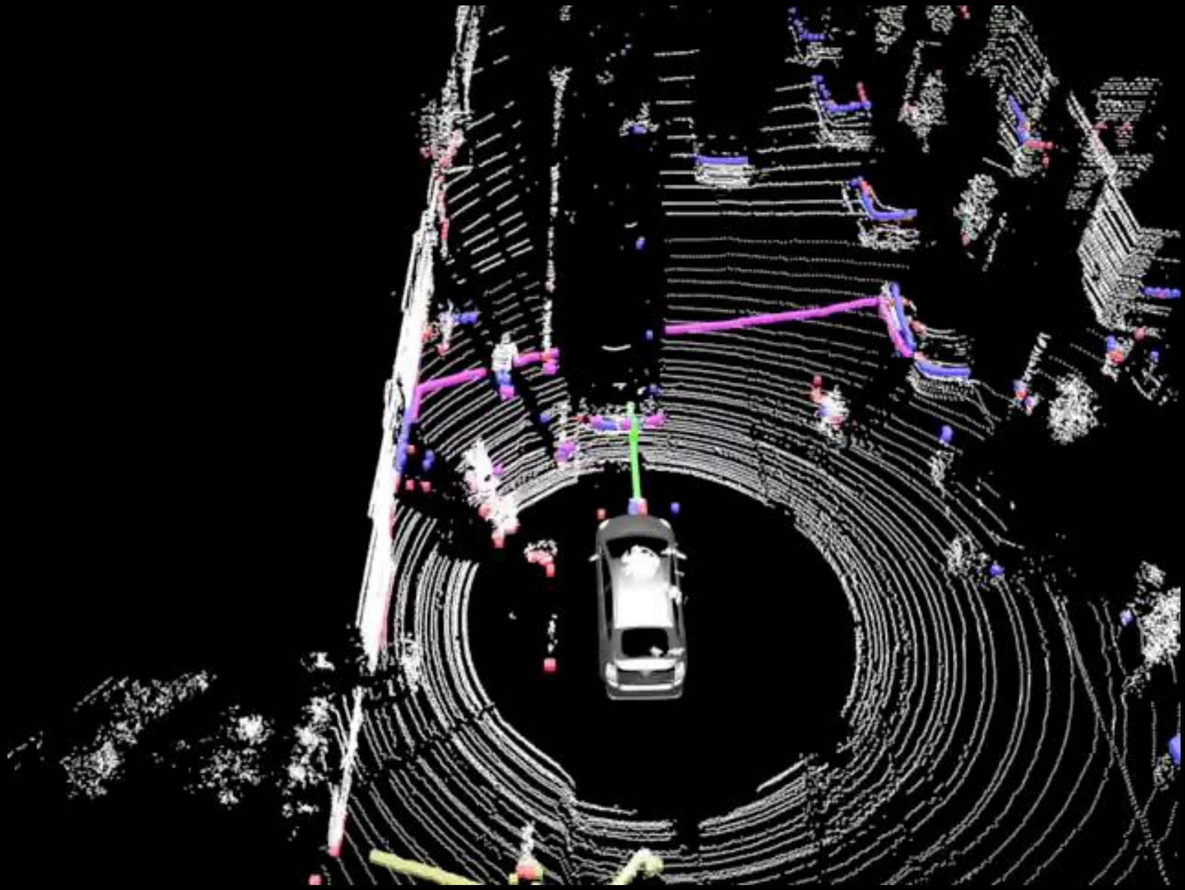
- What if you want to register a geometric model to data that's defined as a set of points?
- Example of registration:





Motivation: Localization and Registration

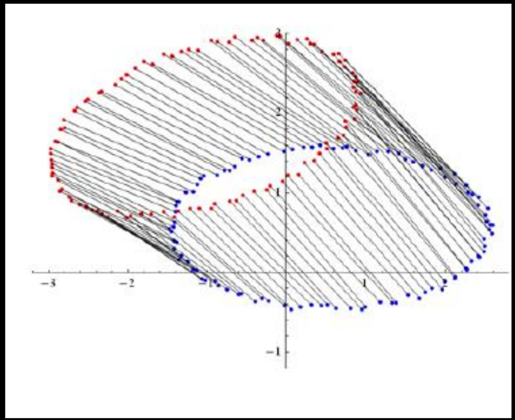
- To navigate, we want to build a map and/or localize the robot in a map (SLAM: Simultaneous Localization and Mapping)



Video from Matthew Johnson-Roberson

Point set registration: Known Correspondences

- Consider two sets of 3D points



Source: $P = \{p_1, p_2, \ldots p_n\}$
Target: $Q = \{q_1, q_2, \ldots, q_n\}$

  - Assume that p_i corresponds to q_i

    (we will remove this assumption later)

- Goal: Compute a rotation R and translation t to apply to P to best align the point sets
- In math:

$$\underset{R \in SO(3), t \in \boldsymbol{R}^3}{\operatorname{argmin}} \sum_{i=1}^{n} ||(Rp_i + t) - q_i||^2$$

This makes the problem difficult!

- Luckily, we don't need optimization algorithms to solve this, we can use calculus and SVD directly

Point set registration: Known Correspondences

- Let's find the translation t first
- Assume R is fixed, we find t by finding the root of

$$\frac{d(\sum_{i=1}^{n} \|(Rp_i + t) - q_i\|^2)}{dt} = 0$$

- Let's solve for t:

$$\frac{d(\sum_{i=1}^{n} \|(Rp_i + t) - q_i\|^2)}{dt} = \sum_{i=1}^{n} 2((Rp_i + t) - q_i) = 0$$

$$= 2\sum_{i=1}^{n} Rp_i + 2\sum_{i=1}^{n} t - 2\sum_{i=1}^{n} q_i = 0$$

Point set registration: Known Correspondences

$$\ldots = 2\sum_{i=1}^{n} Rp_i + 2\sum_{i=1}^{n} t - 2\sum_{i=1}^{n} q_i = 0$$

$$t = \frac{\sum_{i=1}^{n} q_i}{n} - R\frac{\sum_{i=1}^{n} p_i}{n}$$

- Rename variables for convenience:

$$\bar{p} = \frac{\sum_{i=1}^{n} p_i}{n} \qquad \bar{q} = \frac{\sum_{i=1}^{n} q_i}{n}$$

These are just the means of the datasets!

$$t = \bar{q} - R\bar{p}$$

Point set registration: Known Correspondences

- Now that we have t, let's find R
- Plug $t = \bar{q} - R\bar{p}$ into the objective function:

$$\sum_{i=1}^{n} \|(Rp_i + t) - q_i\|^2 = \sum_{i=1}^{n} \|(Rp_i + \bar{q} - R\bar{p}) - q_i\|^2$$

$$= \sum_{i=1}^{n} \|R(p_i - \bar{p}) \;-\; (q_i - \bar{q})\|^2$$

- Rename for convenience: $x_i = p_i - \bar{p}, \quad y_i = q_i - \bar{q}$
- Now the problem becomes:

$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} \|Rx_i - y_i\|^2$$

Point set registration: Known Correspondences

$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} ||Rx_i - y_i||^2$$

- To solve, first compute the 3x3 covariance matrix (don't need to divide by n-1)

$$S = XY^T$$

where X and Y are 3 x n matrices that have x_i and y_i as their columns

- Compute the SVD of S: $\mathrm{SVD}(S) = U\Sigma V^T$
- Then R is:

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{bmatrix} U^T$$

This matrix prevents reflections

- Proof of this method is here

Point set registration: Known Correspondences

- In summary, to solve

$$P = \{p_1, p_2, \dots p_n\}$$
$$Q = \{q_1, q_2, \dots, q_n\}$$

$$\underset{R \in SO(3), t \in \mathbf{R}^3}{\operatorname{argmin}} \sum_{i=1}^{n} \|(Rp_i + t) - q_i\|^2$$

1. Compute means and centered vectors:

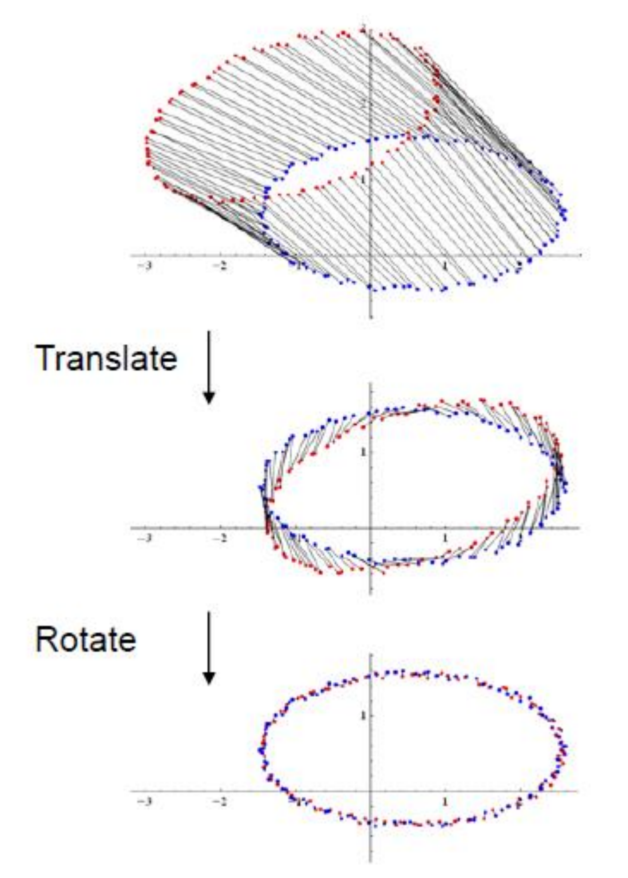
$$\bar{p} = \frac{\sum_{i=1}^{n} p_i}{n} \qquad \bar{q} = \frac{\sum_{i=1}^{n} q_i}{n} \qquad \begin{aligned} x_i &= p_i - \bar{p} \\ y_i &= q_i - \bar{q} \end{aligned}$$

2. Compute SVD of covariance matrix of centered vectors:

$$S = XY^T \qquad \mathrm{SVD}(S) = U\Sigma V^T$$

3. Compute R and t:

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{bmatrix} U^T \qquad t = \bar{q} - R\bar{p}$$



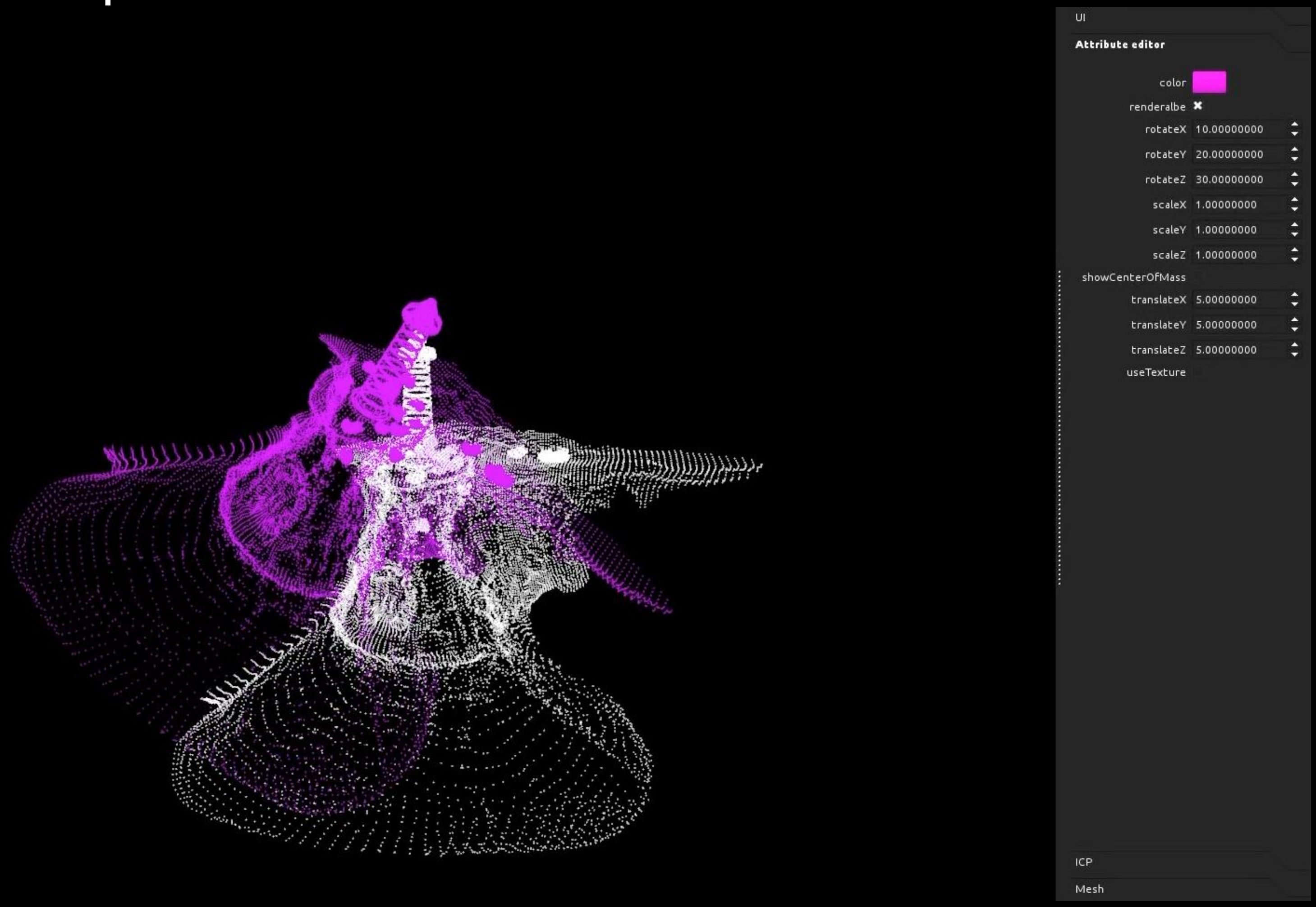
Iterative Closest Point (ICP)

- We assumed we knew the correspondences between points
  - In practice this is rarely true
- ICP iteratively computes correspondences and registers point sets
- There are many variants of ICP, here is a simple one
- Won't always succeed!
  - Need to add a way to terminate based on
    - time
    - number of iterations
    - lack of progress
- Need to tune ϵ

```
Input: P and Q (not necessarily
same size)
Output: P aligned to Q
```

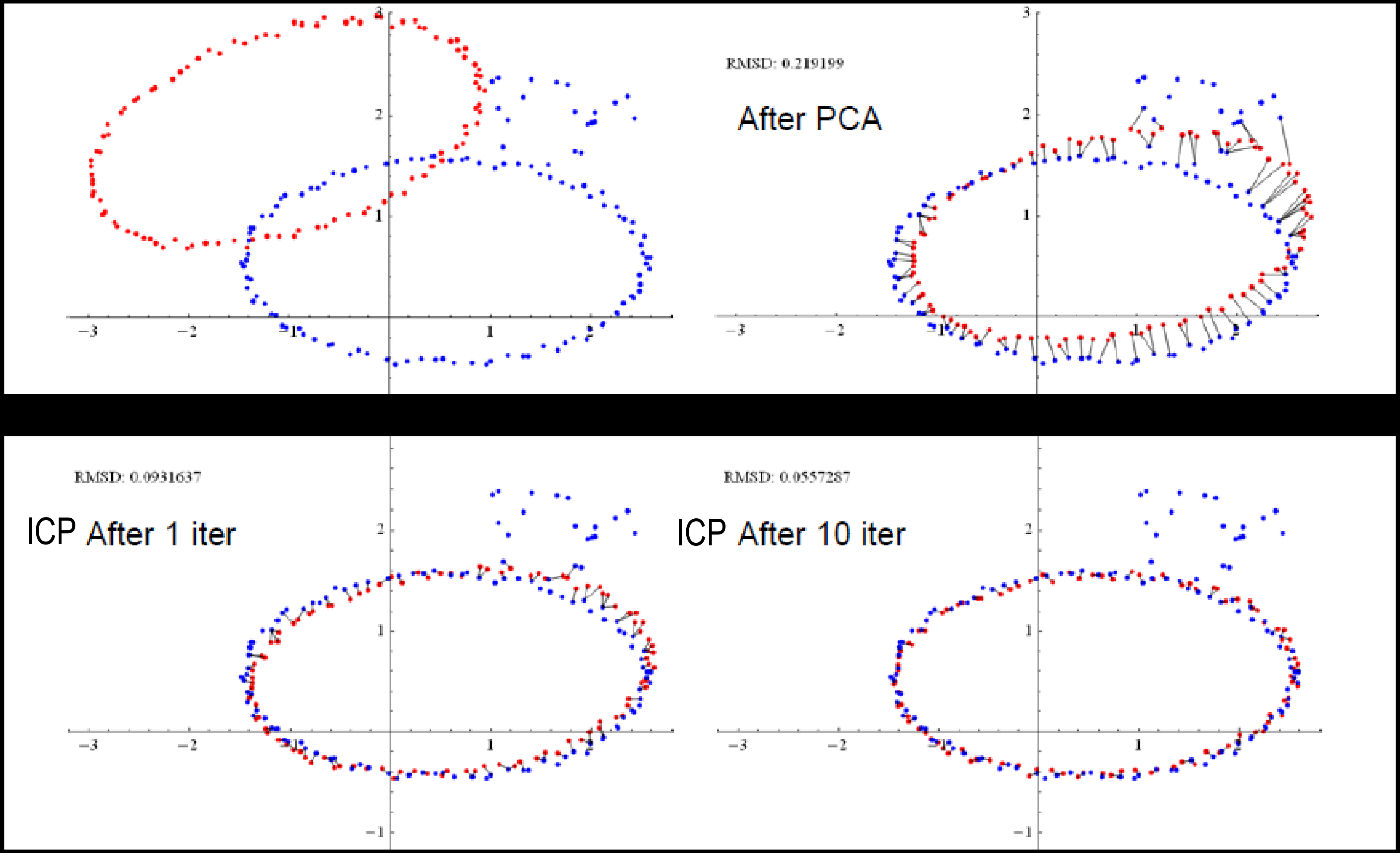
```
Set P to some initial pose
While not Done
    //Compute Correspondences
    C = ∅
    For each p_i ∈ P
        find the closest q_i
        C = C ∪ {p_i, q_i}

    //Compute Transform
    //(see previous slide)
    R, t ← GetTransform(C_p, C_q)

    If Σ_{i=1}^{n} ||(R C_{p_i} + t) − C_{q_i}||^2 < ϵ
        return P

    // Update all P
    For each p_i ∈ P
        p_i = R p_i + t
```

ICP Example



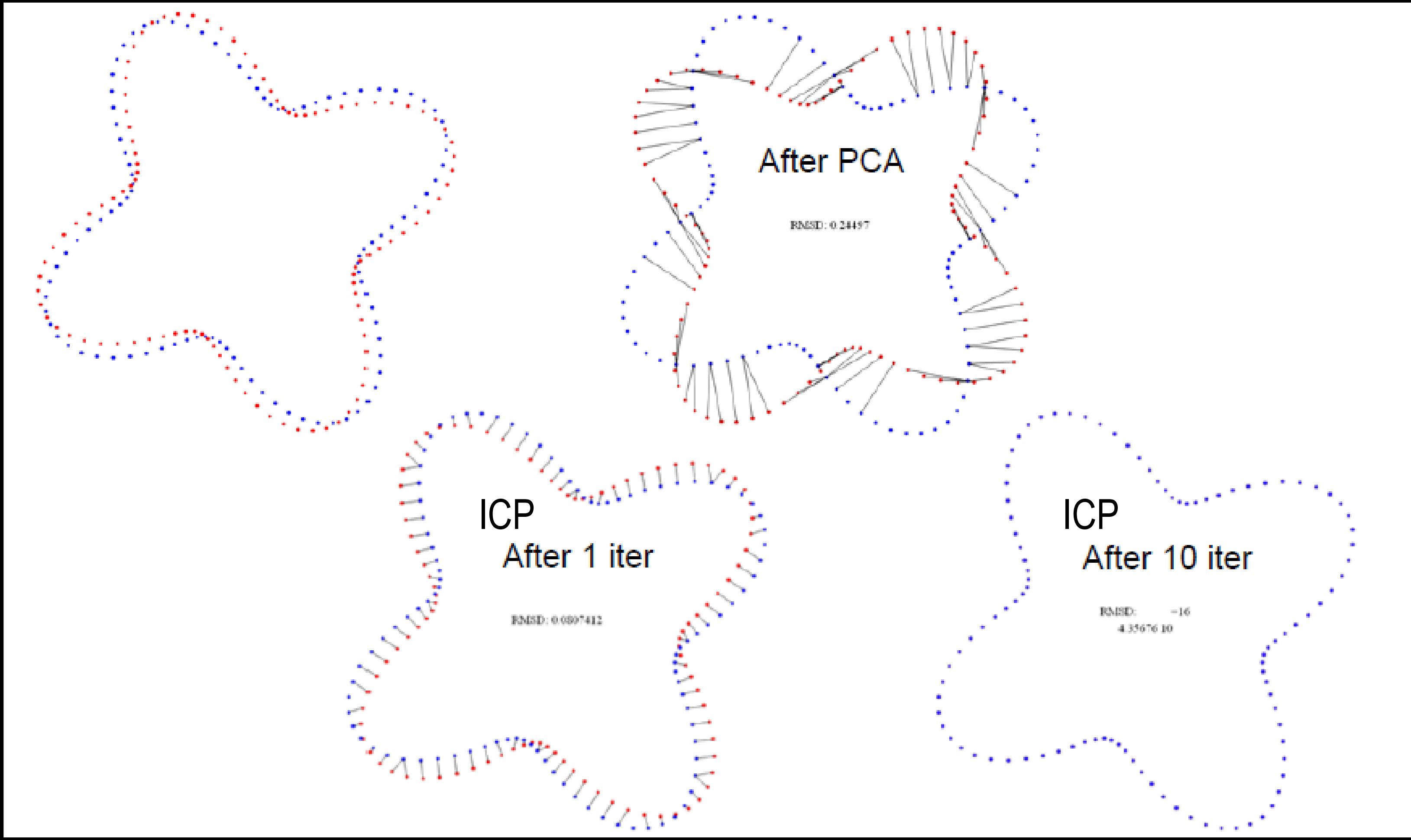
Jeppe Walther, ICP Point Cloud Alignment, https://www.youtube.com/watch?v=Xq91aMwYezU

PCA vs. ICP



RMSD: 0.219199
After PCA
RMSD: 0.0931637
ICP After 1 iter
RMSD: 0.0557287
ICP After 10 iter

PCA vs. ICP


After PCA
RMSD: 0.24497
ICP
After 1 iter
RMSD: 0.0807412
ICP
After 10 iter
RMSD: 4.35676 10 −16

ICP Problems and Solutions

- Problem: Correspondences based on outliers (e.g. from sensor noise) can disrupt the process
  - Solution: Solve

$$\underset{R \in SO(3), t \in \boldsymbol{R}^3}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \|(Rp_i + t) - q_i\|^2$$

where $w_i \geq 0$ captures the probability that $\{p_i, q_i\}$ is an outlier (e.g. based on how far $\mathrm{dist}(p_i, q_i)$ is from the mean distance of all pairs)

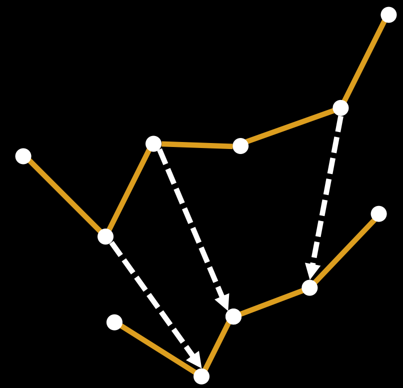
  - Use same solution method as before but change the following:

$$\bar{p} = \frac{\sum_{i=1}^{n} p_i}{\sum_{i=1}^{n} w_i} \qquad \bar{q} = \frac{\sum_{i=1}^{n} q_i}{\sum_{i=1}^{n} w_i}$$

$$S = XWY^T$$

where $W = \mathrm{diag}(w_1, w_2, \dots, w_n)$

ICP Problems and Solutions

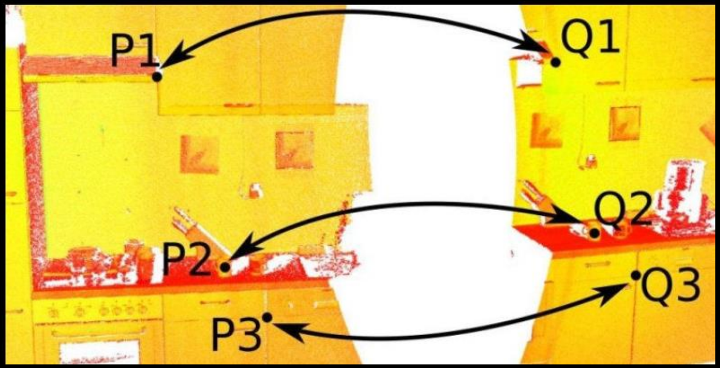


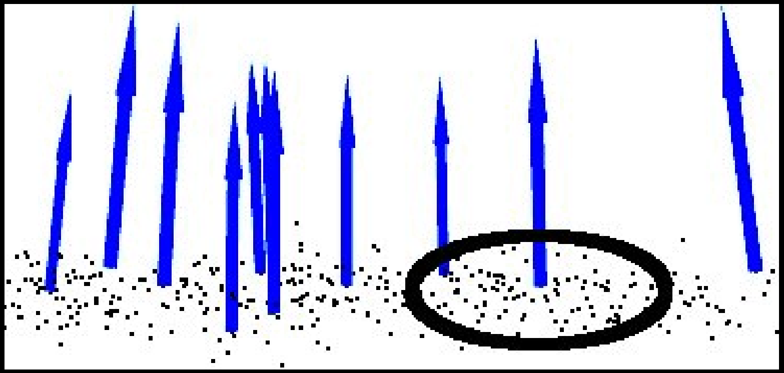
- Problem: Not taking into account connectivity of points
  - Solution: Match points based on local geometry around points
    - Some methods give worse results when there is noise
    - We'll see ways to do this later
- Problem: ICP is very sensitive to initial transform
  - Solution: Try multiple initial transforms
    - This will be slow
- Problem: Considering every point in P and Q may be slow
  - Solution: Define a subset of points to consider for registration (regular sub-sampling, random sampling, etc.)
  - Solution: Use Oct-Trees or k-d trees to speed up nearest-neighbor queries

Break

Point Cloud Features

- Point cloud features can help us determine better correspondences between points
  - E.g. Considering the surface patch around a point

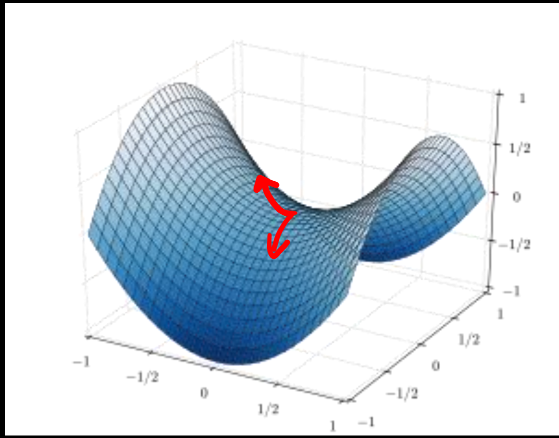




- For a given point, select k neighbors within a distance bound, put these points in a matrix X
  - Compute surface normal n of the surface patch formed by the k points
    - Compute eigenvectors and eigenvalues of surface patch (e.g. using $\mathrm{SVD}(XX^T)$)
    - n is the eigenvector corresponding to the smallest eigenvalue
  - Compute curvature κ of the surface patch formed by the k points
    - Compute eigenvalues of surface patch (e.g. using $\mathrm{SVD}(XX^T)$)
    - $\kappa = \frac{\lambda_0}{\lambda_0+\lambda_1+\lambda_2}$ where λ_i are eigenvalues and $\lambda_0 < \lambda_1 < \lambda_2$
- Determine correspondences using features, e.g. use distance between $[n_{p_i}, \kappa_{p_i}]$ and $[n_{q_i}, \kappa_{q_i}]$

Point Cloud Features: Point Feature Histograms (PFH)

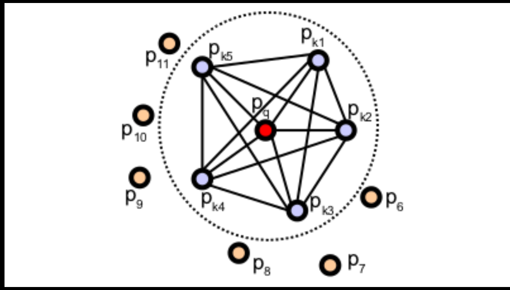
- Goal: Capture the geometric properties of a neighborhood of points, accounting for how properties vary in different directions.



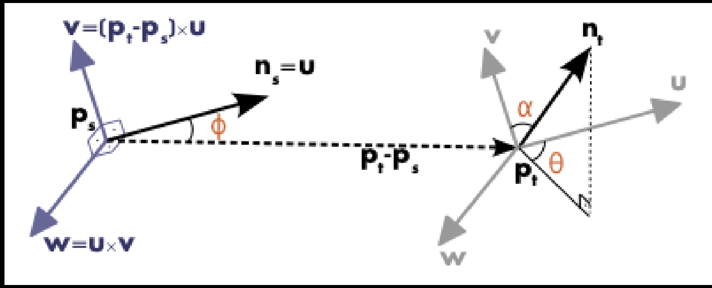
Sometimes curvature changes differently in different directions

- Point Feature Histogram (PFH): Each point receives a signature (a high-dimensional vector) based on the statistics of how the surface normals change in the surface patch around that point
  - PFH is invariant to the pose of the underlying surface
  - PFH is not very sensitive to different sampling densities or noise levels

Computing PFH



- Want to capture how surface normals change between every pair of points in the neighborhood
- Compute a frame for each pair of points:



$$d = \|p_t - p_s\|$$
$$u = n_s$$
$$v = u \times \frac{p_t - p_s}{d}$$
$$w = u \times v$$

- Angular features capture change in surface normal between these points:

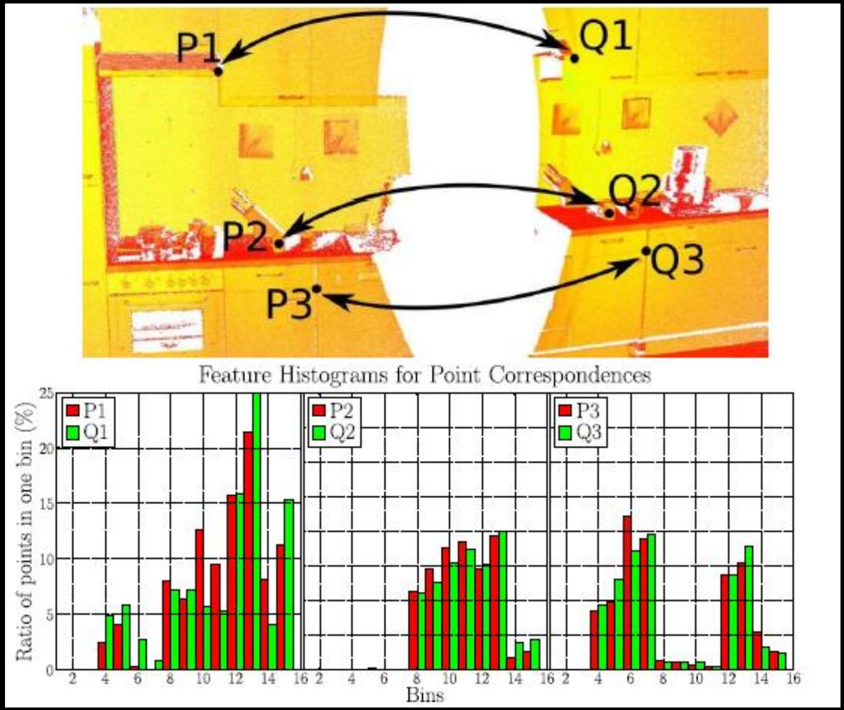
$$\alpha = v \cdot n_t$$
$$\phi = u \cdot \frac{p_t - p_s}{d}$$
$$\theta = \arctan(w \cdot n_t, u \cdot n_t)$$

The feature for this pair is:
$\langle \alpha, \phi, \theta, d \rangle$

Often not used in robotics b/c laser scanner point spacing increases with distance from the scanner (d is not informative)

Computing PFH

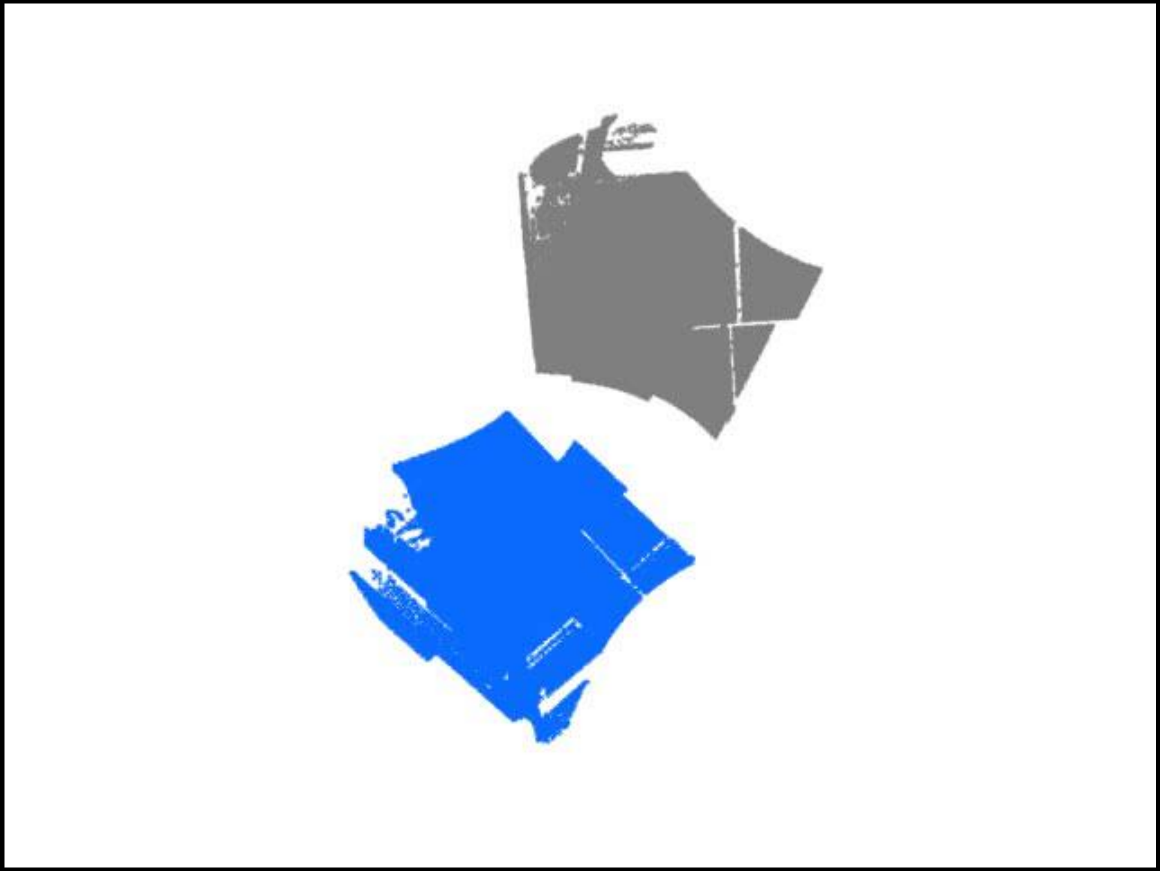
- Collect $\langle \alpha, \phi, \theta, d \rangle$ features for every pair of points in a neighborhood around p
- Compute a histogram of the values
  - Each bin in the histogram is a range of values for $\langle \alpha, \phi, \theta, d \rangle$
  - If you have b bins per dimension, need b^4 bins total
- A vector of the percentage of pairs that fall into each bin is the signature of p
- WARNING: Computing the signature can take significant time if point cloud is dense



[Rusu et al., IROS, 2008]

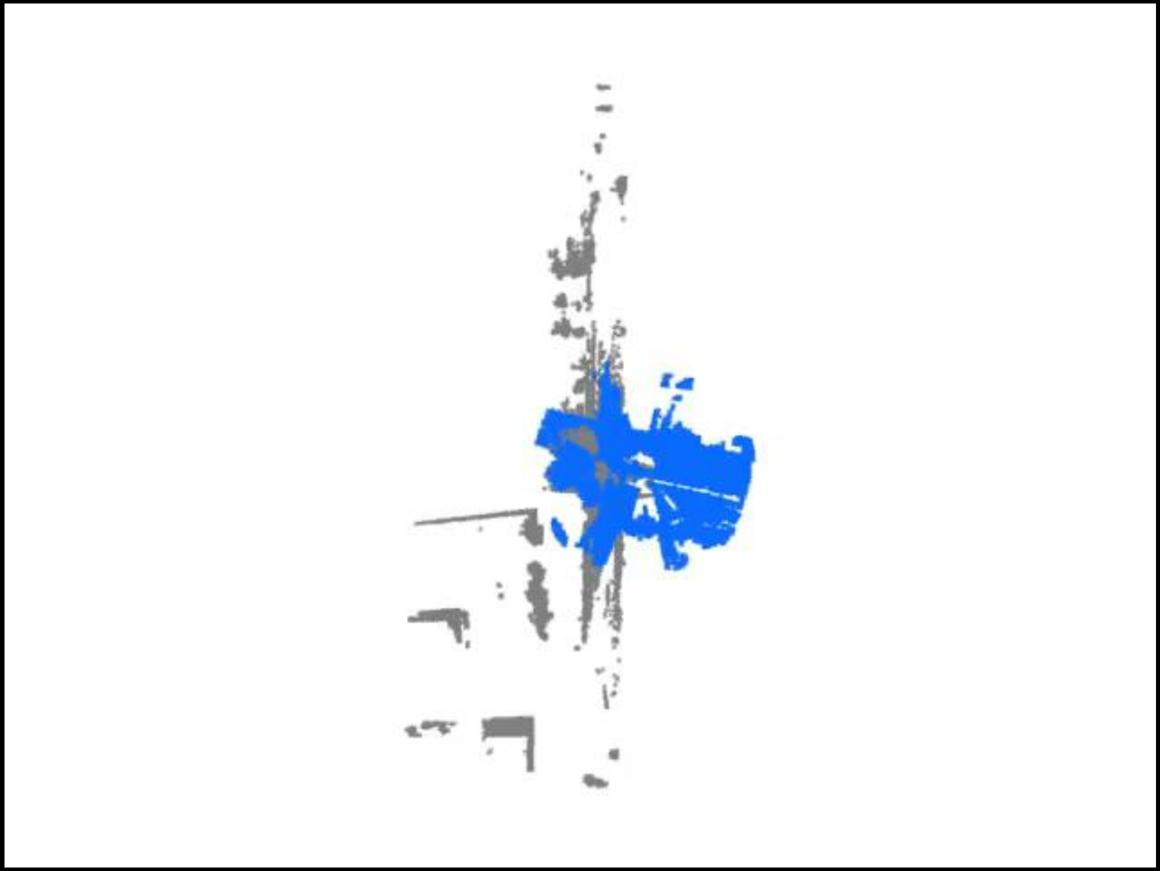
Now ICP can determine correspondences using distance between PFH signatures!

PFH Example



pointclouds.org

PFH Example



pointclouds.org

Summary

- RANSAC is a way to fit non-linear models to data
  - Works well with noise, model type is arbitrary
  - Need to set problem-specific thresholds

- ICP is a way to iteratively register a references set of points to target set.
  - At each iteration:
    1. Compute correspondences
    2. Move source points to minimize error between corresponding points
  - Sensitive to initial correspondences (set by initial transform of reference points)
  - Doesn’t take into account surface information (by default)

- Point cloud features (such as PFH) can be used with ICP
  - Can estimate correspondences based on surface patch similarity (don’t need to initialize ICP)
  - PFH is invariant to the pose of the underlying surface
  - PFH is not very sensitive noise

Homework

- Read AI book Ch. 13
- Read AI book Ch. 14.1-14.3
- Homework 4 is out