上海科技大学
ShanghaiTech University

CS283: **Robotics Spring 2026: Localization**

Sören Schwertfeger / **师泽仁**

ShanghaiTech University

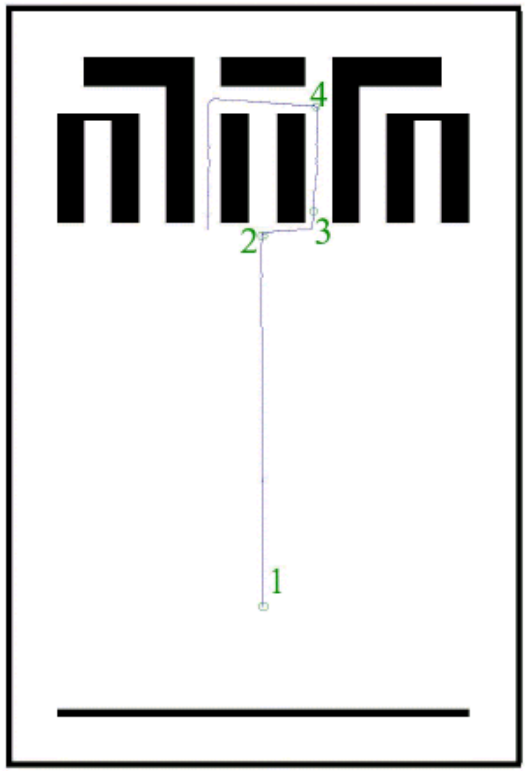
LOCALIZATION METHODS

Localization

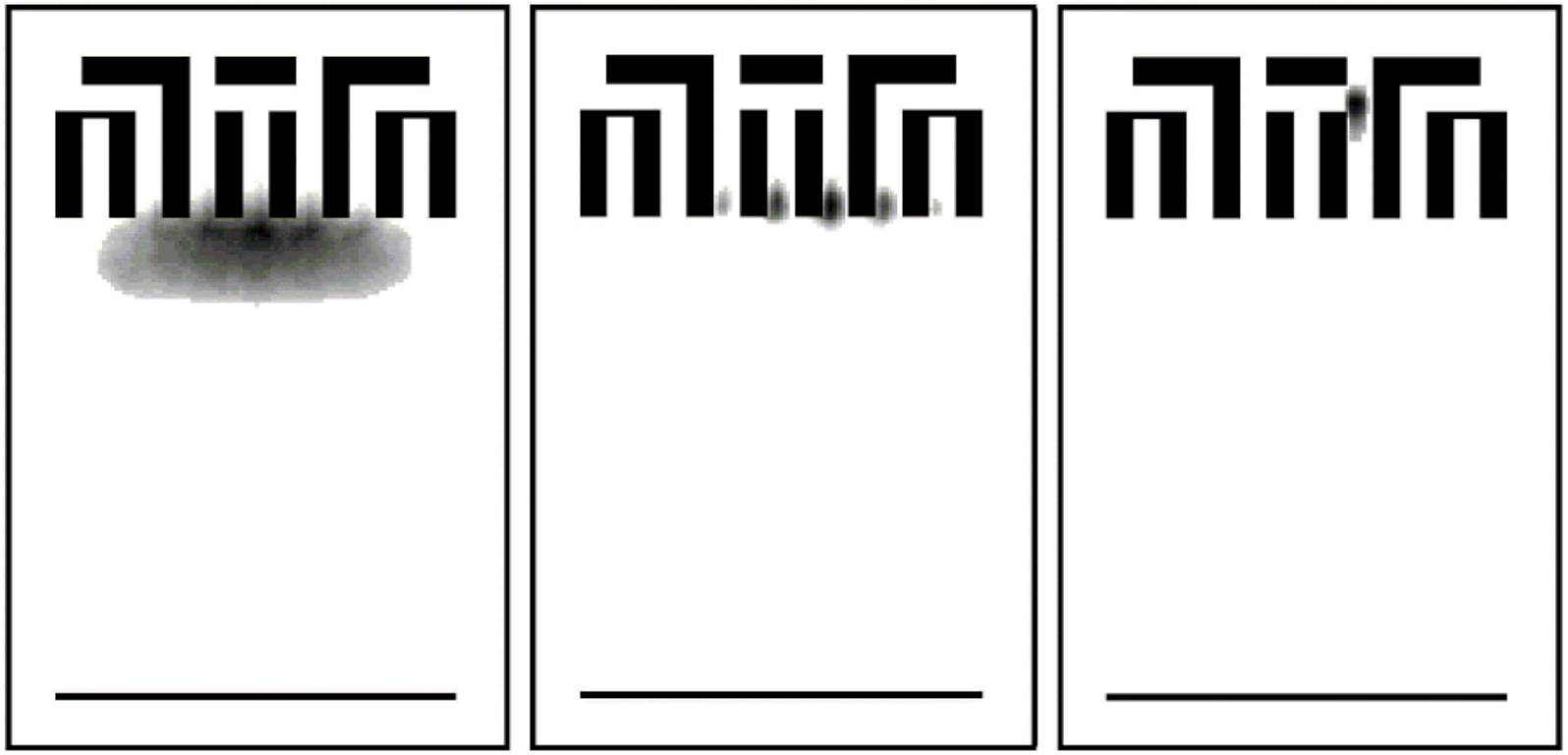
- Based on control commands
  => Open Loop!
- Wheel odometry
  - Compass, Accelerometer, Gyro => IMU
- Scan Matching of Range Sensors == Registration (rigid => no scaling or shearing)
  - ICP: scan to scan or scan to map
    - Needs good initial guess
  - NDT registration
  - Feature-based registration
  - Direct/ optimization based registration
- Grid-based Localization
- Kalman Filter Based Localization
- Monte-Carlo Localization (MCL) == Particle Filter
  - Adaptative MCL => AMCL
- Visual Odometry (VO)
  - With IMU: Visual Inertial Odometry (VIO)
- SLAM techniques
- 3D Reconstruction
  - Structure from Motion/ Bundle Adjustment
  - Localization is by-product
- Absolute Localization:
  - GPS
  - Markers (e.g. QR code)
  - Landmarks (e.g. ShanghaiTech Tower)

Grid-based Localization - Multi Hypothesis

Probability of robot location saved in grid cells – based on combination of:
1) cell values of previous step; 2) odometry; 2) scan matching



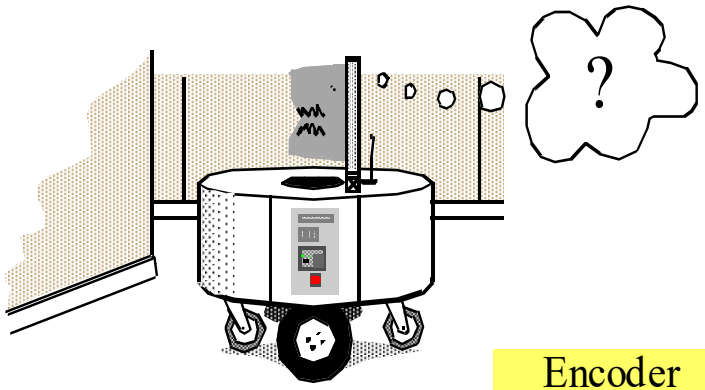
Path of the robot

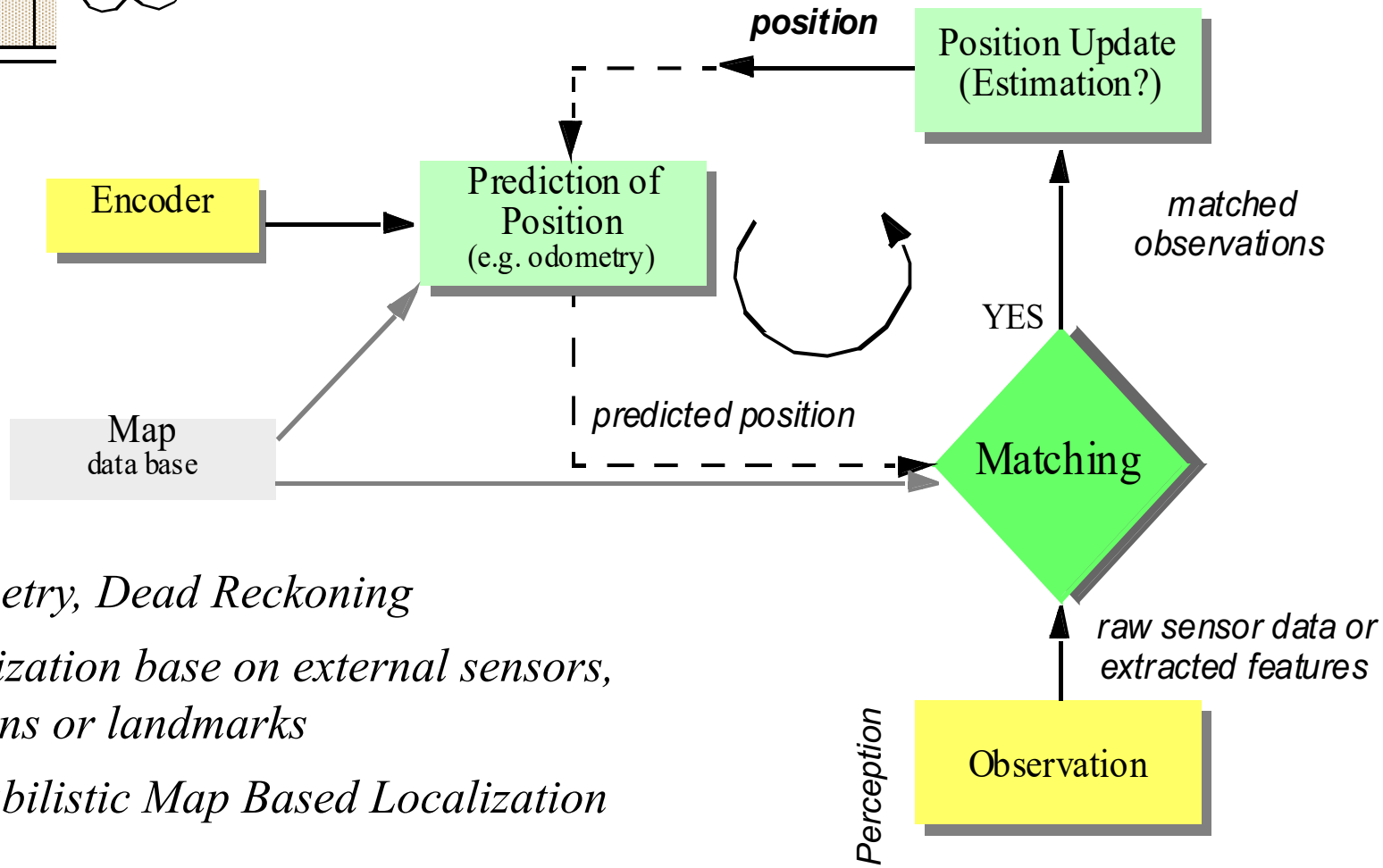


Belief states at positions 2, 3 and 4

Courtesy of W. Burgard

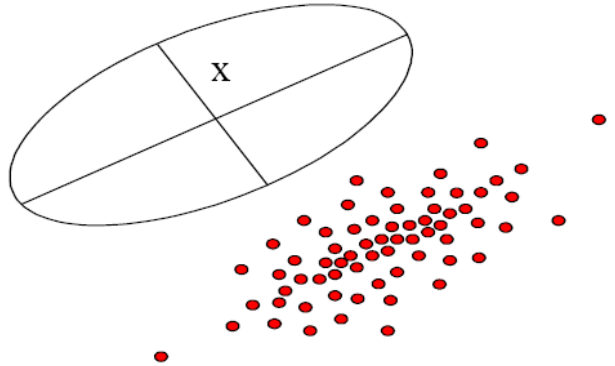
Map based localization





- *Odometry, Dead Reckoning*
- *Localization base on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

Monte Carlo Localization (MCL)



probability distribution (ellipse) as particle set (red dots)

- Input: Global, known map and laser scan
- Particle filter: set of particles representing a robot state
  - Here: robot pose (position & orientation)
  - Particle filter SLAM (e.g. FastSLAM): also map!
  - Particles are sampled based on probability distribution
- Assign weights (scores) to particles based on how well the scan matches to the map, given this pose
- Markov property: Current state only depends on previous state

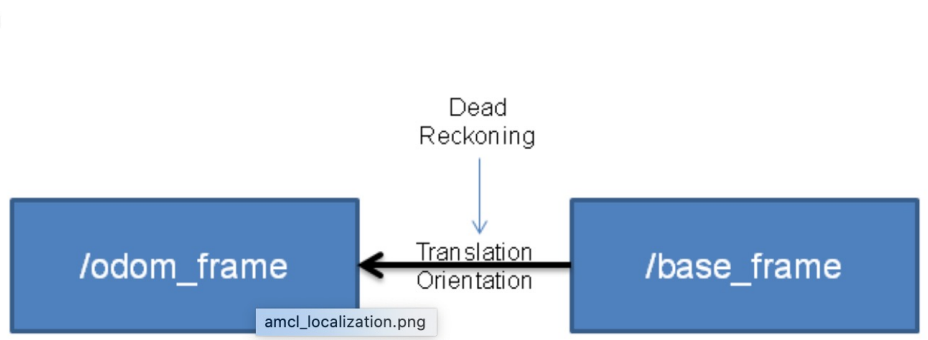
- Algorithm:

1. For all particles:
   1. Apply motion update (e.g. odometry)
   2. Apply the sensor update (scan match) and calculate new weights
2. Re-Sample particles based on their weights

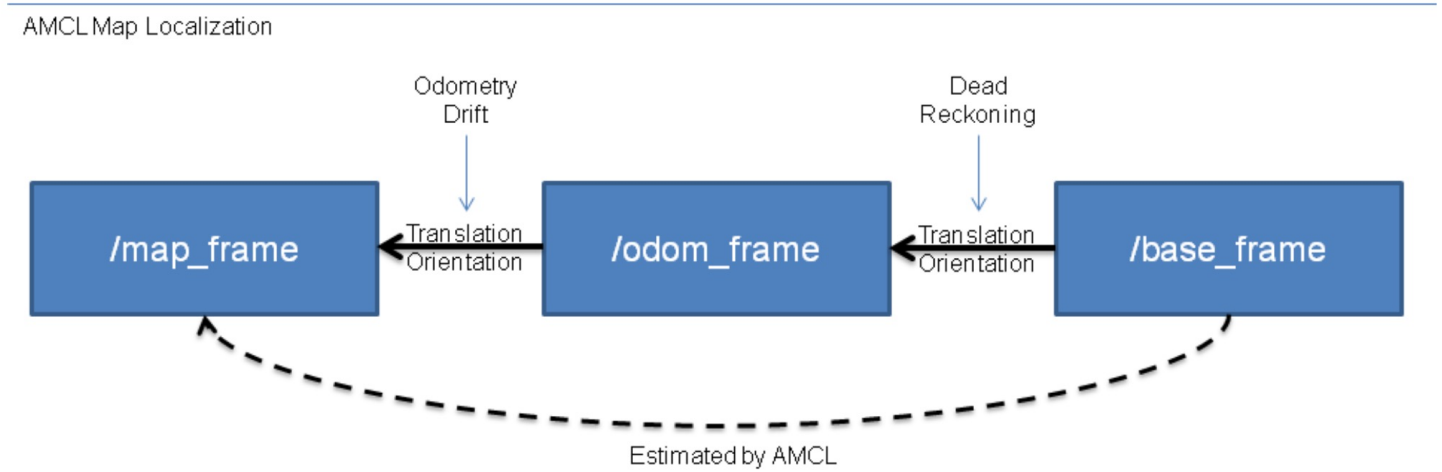
- Can solve the kidnapped robot problem (also wake-up robot problem)
- Problem: Particle of correct pose might not exist…

Adaptive Monte Carlo Localization (AMCL)

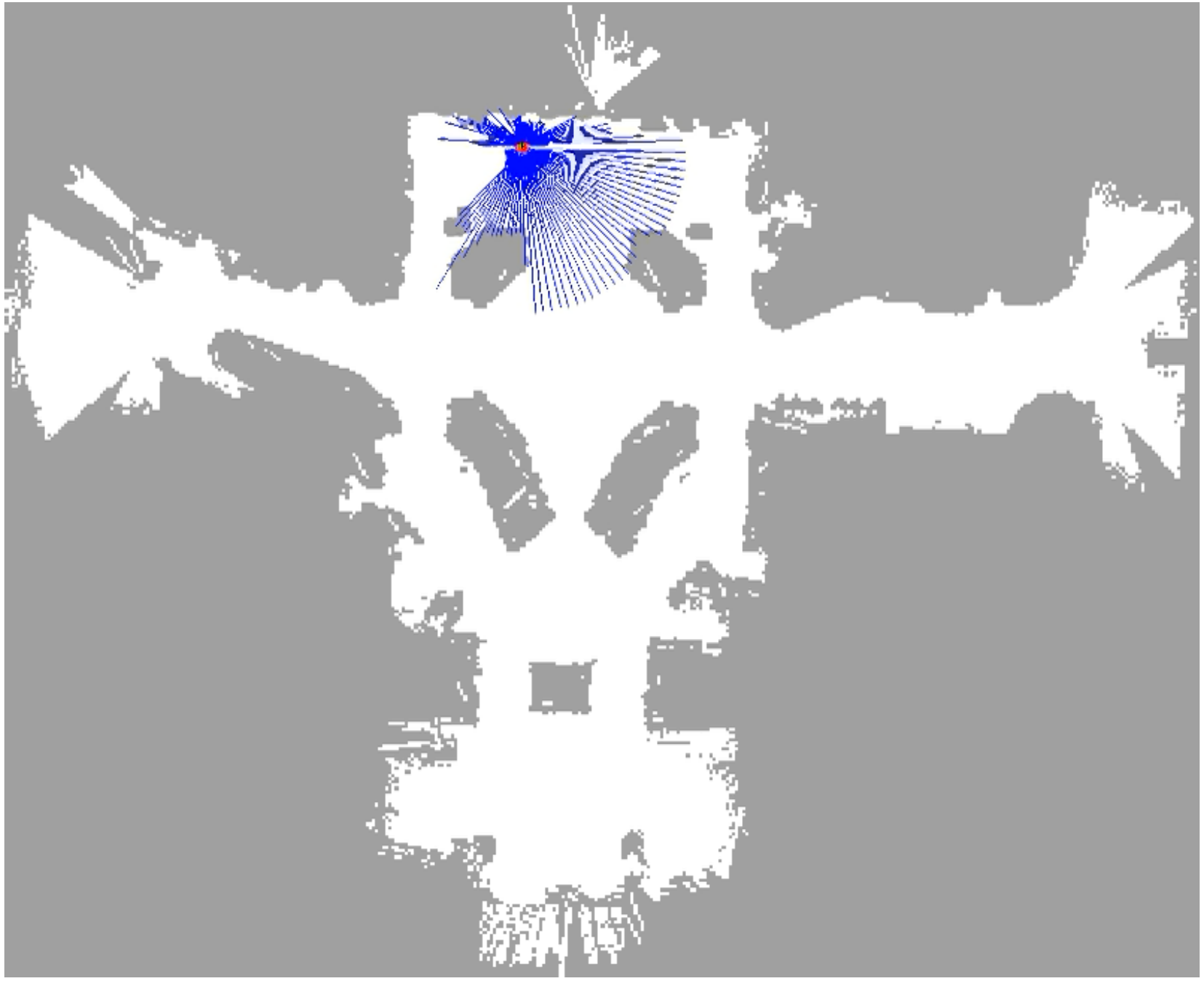
- Sample particles adaptively
  - Based on error estimate
  - Kullback-Leibler divergence (KLD)
  - => when particles have converged, have a fewer number of particles
  - Sample size is re-calculated each iteration



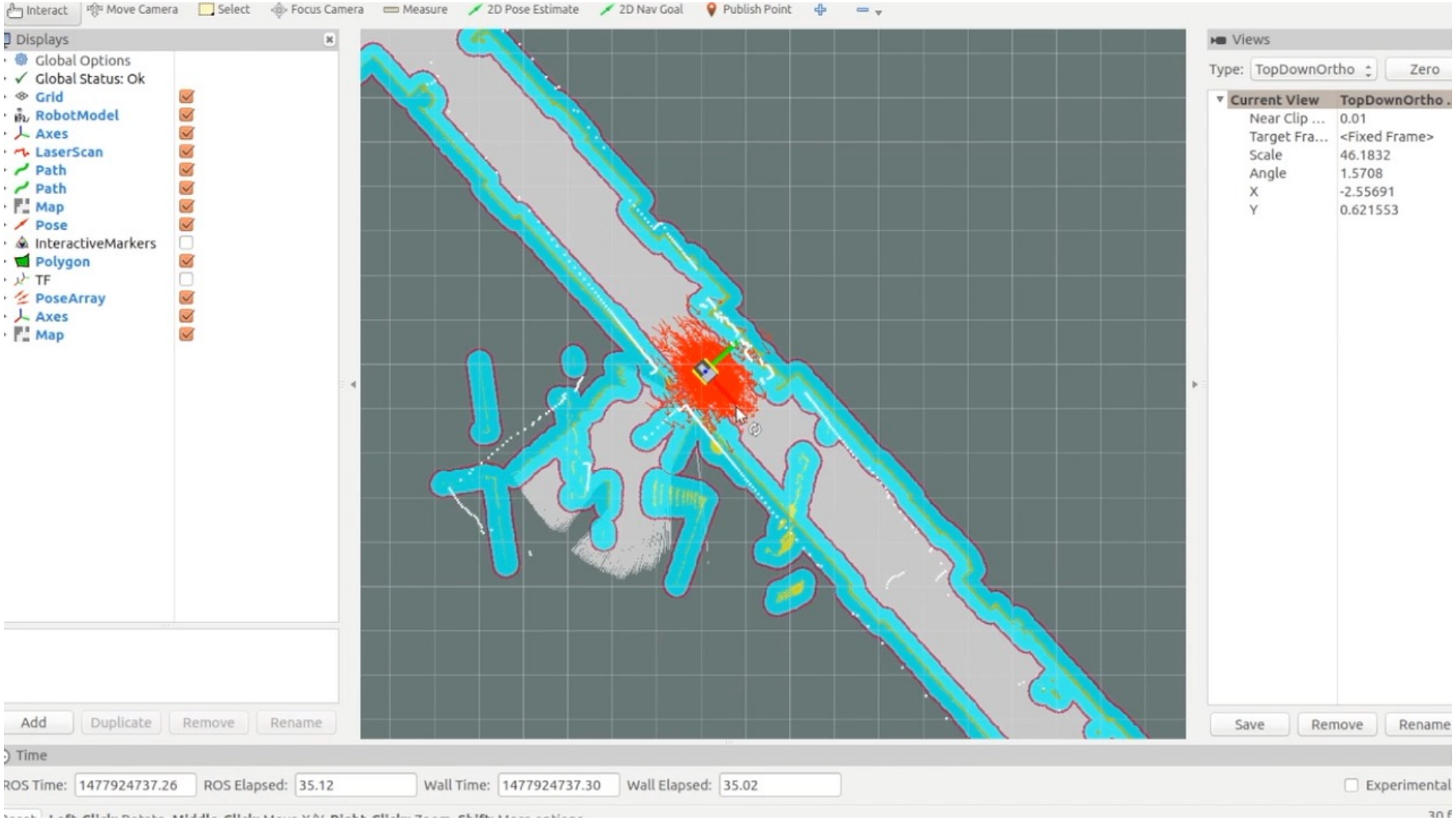
- http://wiki.ros.org/amcl
- Used by the ROS Navigation stack



MCL & Robot Kidnapping



AMCL in ROS


Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Global Status: Ok
Grid
RobotModel
Axes
LaserScan
Path
Path
Map
Pose
InteractiveMarkers
Polygon
TF
PoseArray
Axes
Map
Add
Duplicate
Remove
Rename
Views
Type: TopDownOrtho
Zero
Current View
TopDownOrtho
Near Clip ...
0.01
Target Fra...
<Fixed Frame>
Scale
46.1832
Angle
1.5708
X
-2.55691
Y
0.621553
Save
Remove
Rename
Time
ROS Time:
1477924737.26
ROS Elapsed:
35.12
Wall Time:
1477924737.30
Wall Elapsed:
35.02
Experimental

Scan Matching/ Registration

- Take one sensor scan
- Match against:
  - Another sensor scan
  - Against the map

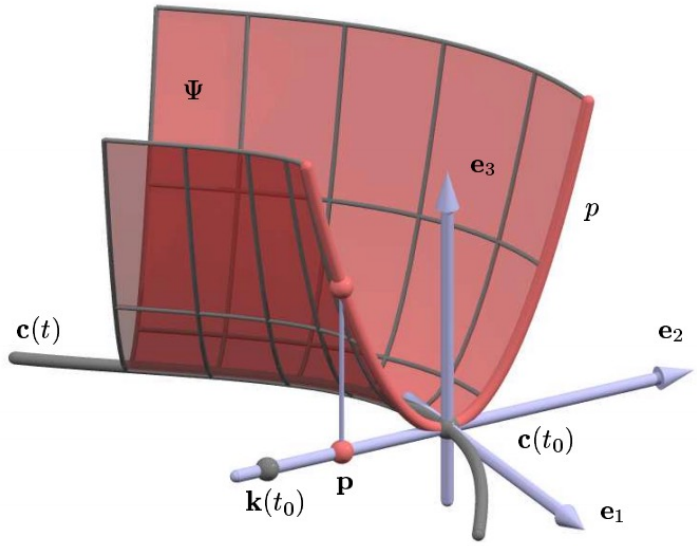
- Output:
  - The Transform (2D: 3DoF; 3D: 6DoF; each maybe with scale)
  - Uncertainty about the result (e.g. covariance matrix) and/ or registration error/ fitting error

- Used for Localization

- Most famous algorithm: ICP (Iterative Closest Point)

Registration Methods for Range Data

- ICP
- NDT
- Robust point matching (soft point correspondences)
- Coherent point drift
- Kernel correlation
- Approximations of the squared distance functions to curves and surfaces
- Direct Methods/ Optimization based (also for images)

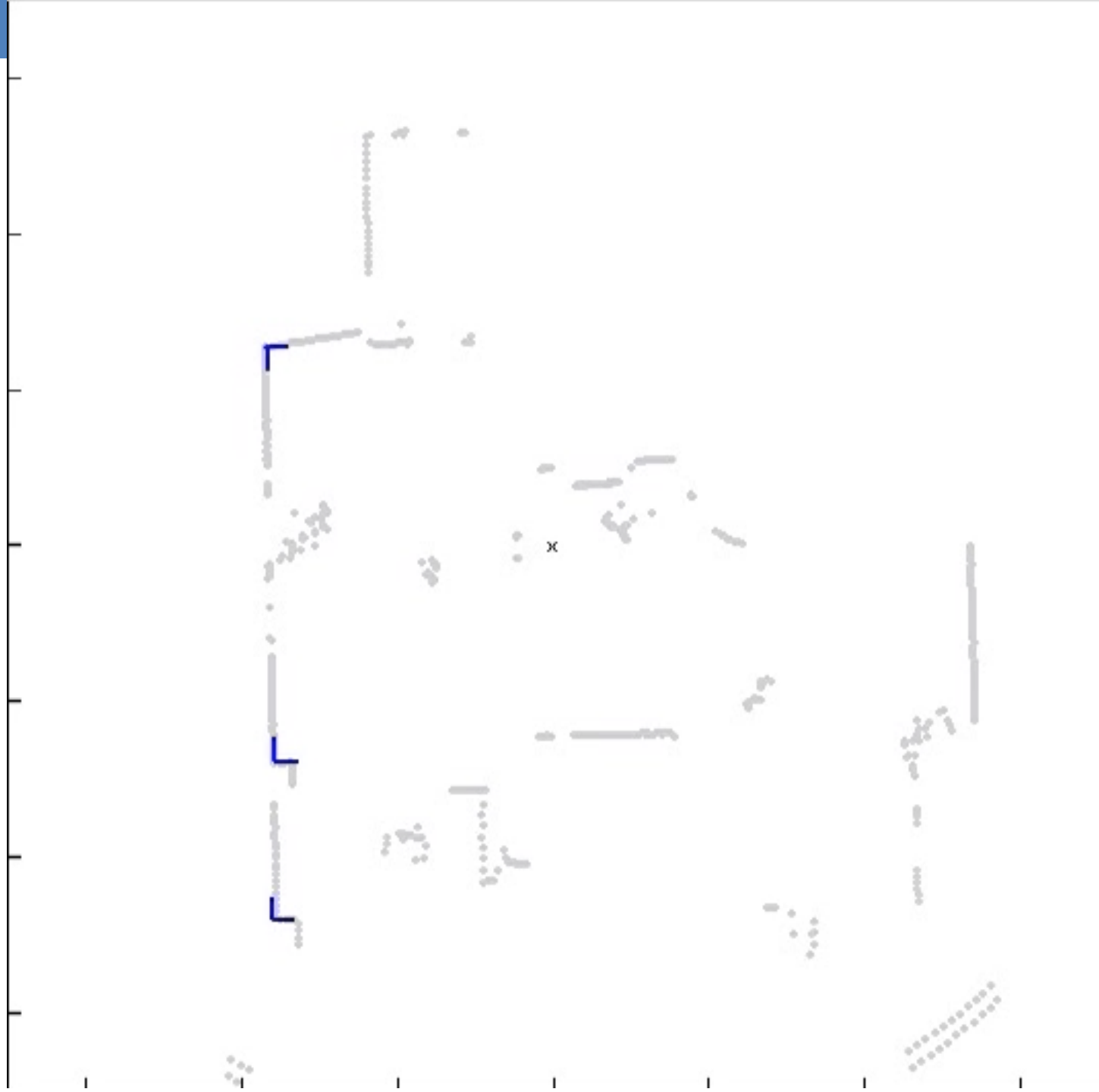


- Feature extracting methods (also for images)
  - Corners in point clouds
  - Lines
  - Planes
  - Feature Descriptors/ also via Deep Learning

- Spectral methods (also for images)

SLAM using corner structures

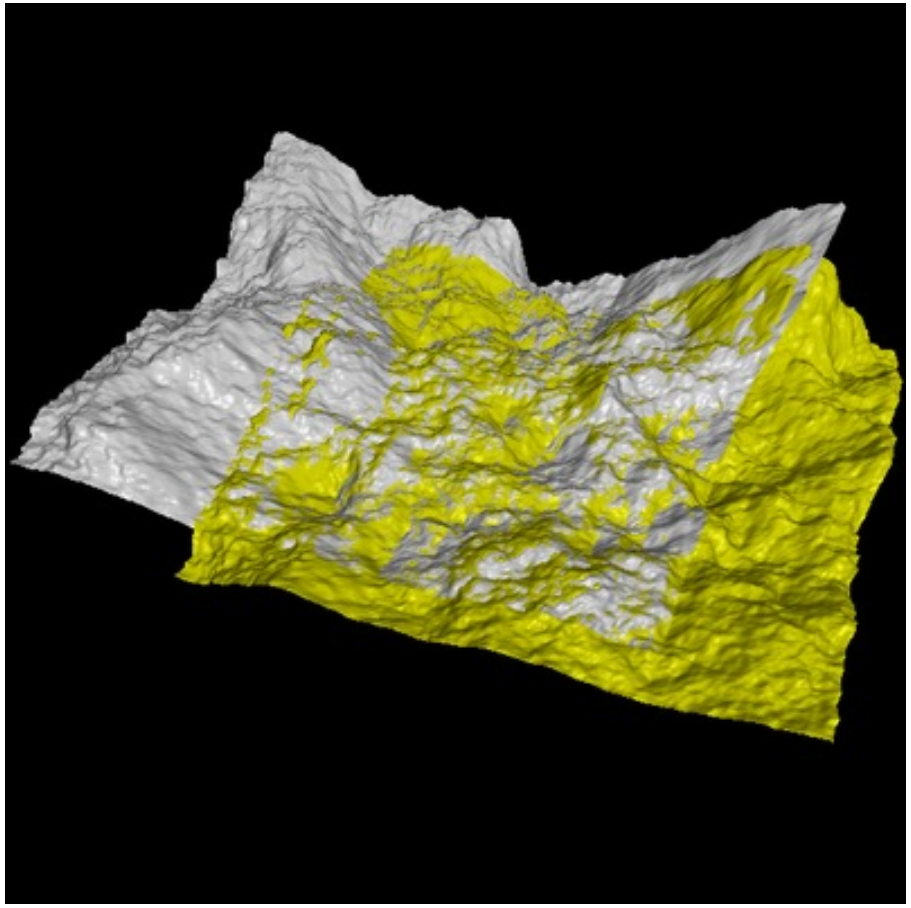
- 2D LRF Scan
- Detect corners in the scan
- Map corners, localization against corners



ICP

ICP: Iterative Closest Points Algorithm

- Align two partially-overlapping point sets (2D or 3D)
- Given initial guess for relative transform
- Warning: Using 3D ICP for 2D data may mirror the data (e.g. 180 degree roll)!
  - Use 2D ICP!
- ROS: Point Cloud Library (PCL)



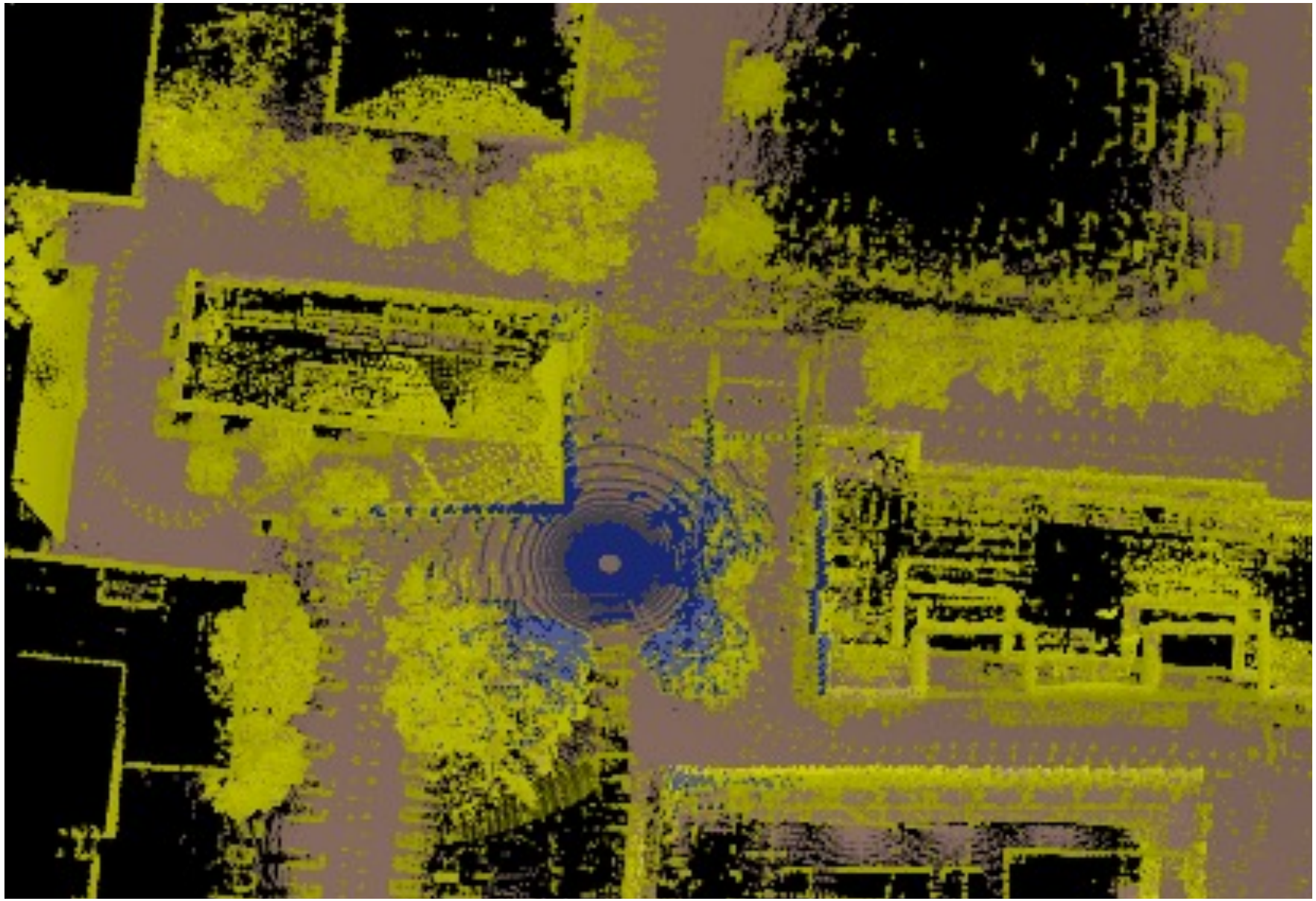
Material derived from Ronen Gvili : *www.cs.tau.ac.il/~dcor/Graphics/adv-slides/**ICP**.ppt*

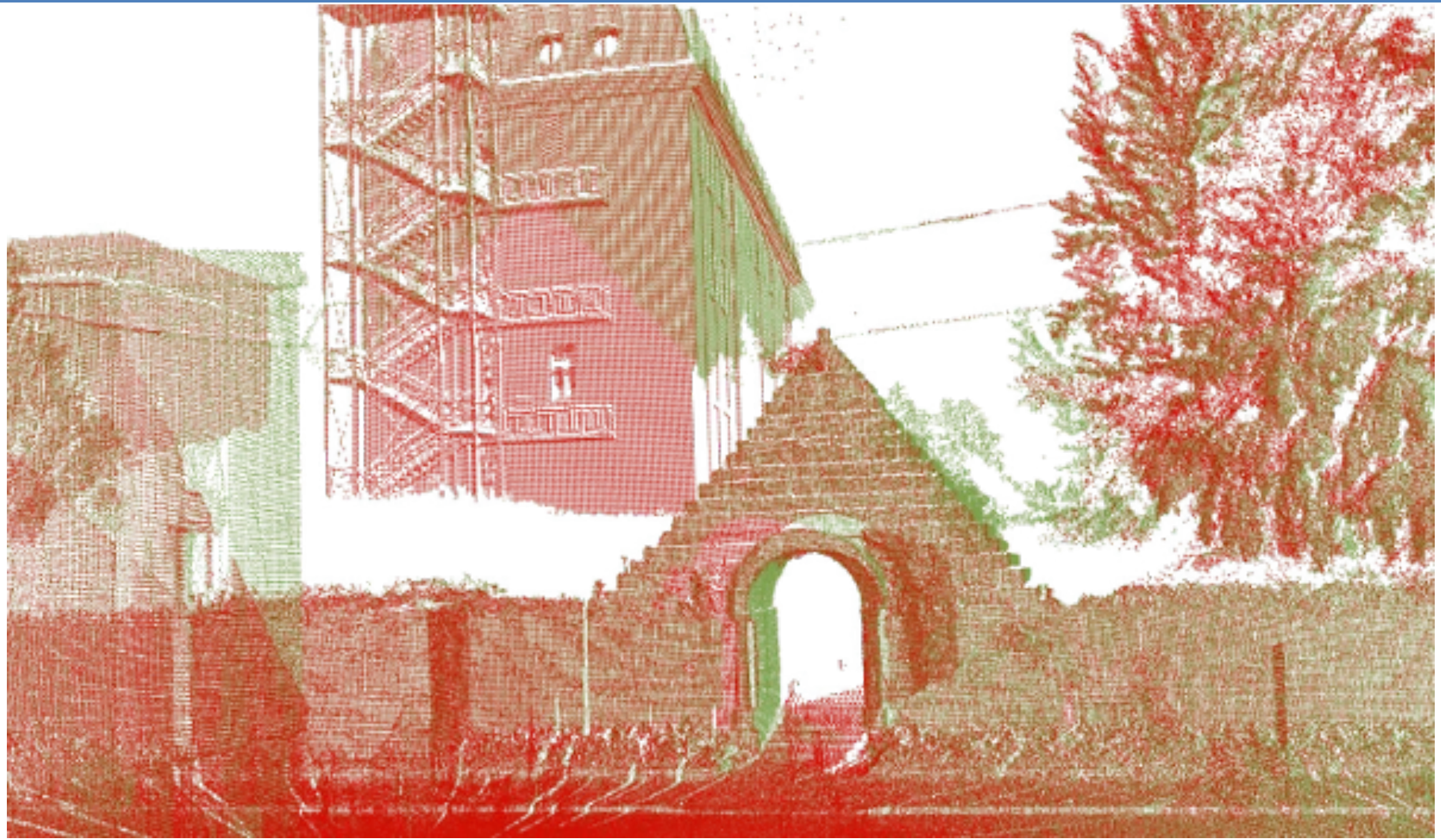
Data Types

- Point sets
- Line segment sets (polylines)
- Implicit curves : f(x,y,z) = 0
- Parametric curves : (x(u),y(u),z(u))
- Triangle sets (meshes)
- Implicit surfaces : s(x,y,z) = 0
- Parametric surfaces (x(u,v),y(u,v),z(u,v)))

Motivation

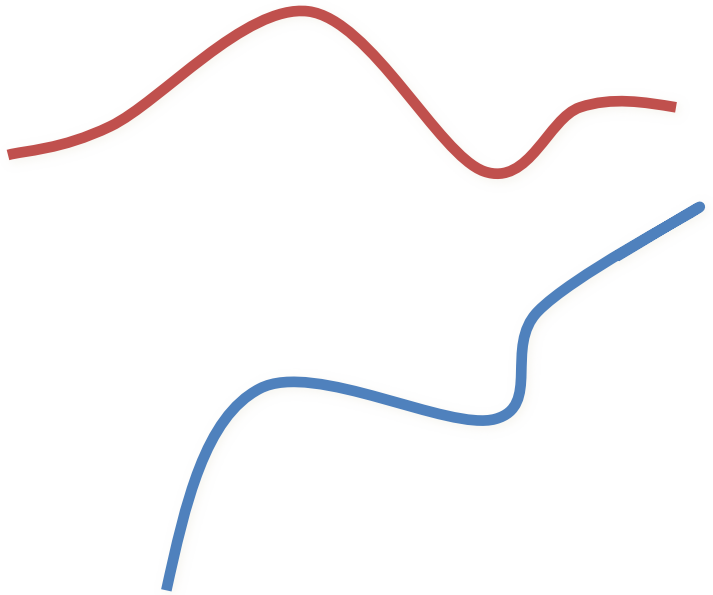
- Scan Matching - Registration
- Shape inspection
- Motion estimation
- Appearance analysis
- Texture Mapping
- Tracking





Aligning 3D Data

- Continuous lines or a set of points…

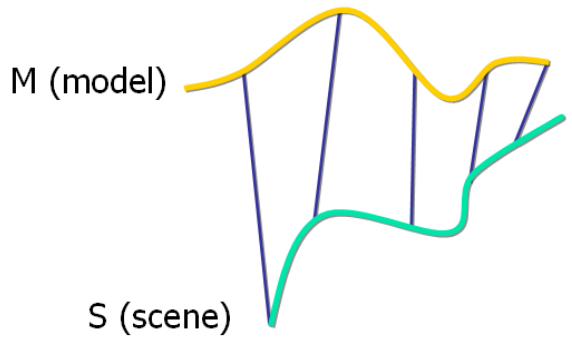


Corresponding Point Set Alignment

- Let M be a model point set. (or map or previous scan)
- Let S be a scene point set. (current scan)

We assume :

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



Corresponding Point Set Alignment

The Mean Squared Error (MSE) objective function :

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

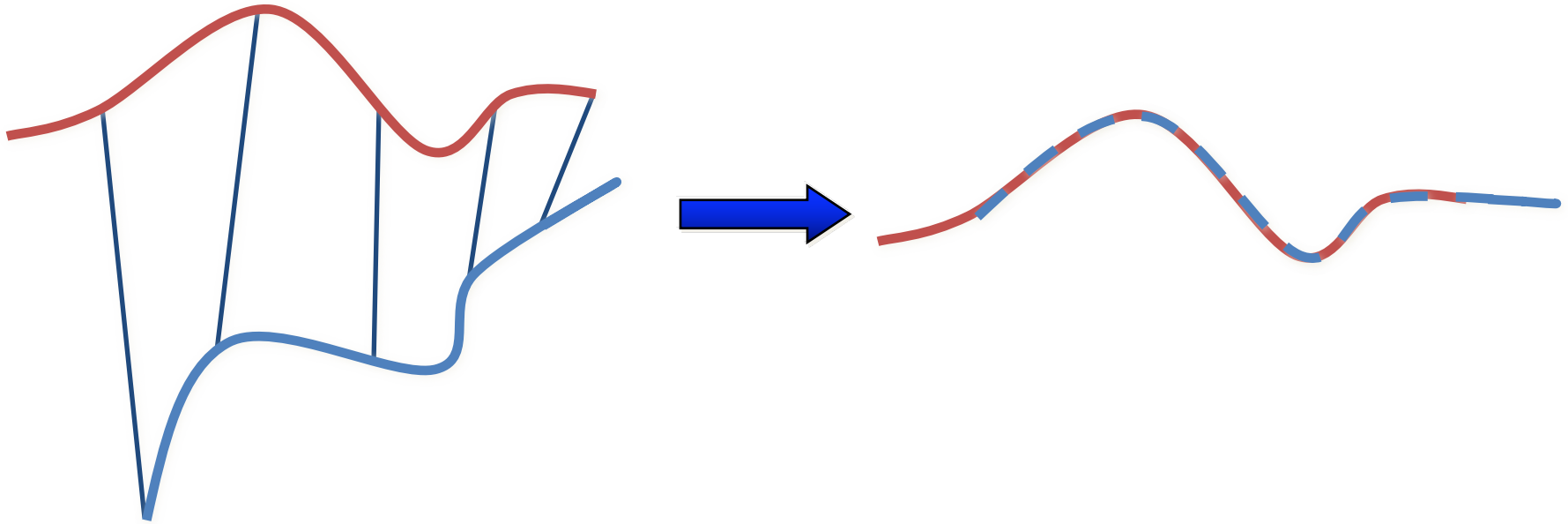
$$f(q) = \frac{1}{N_S}\sum_{i=1}^{N_S}\left\| m_i - R(q_R)s_i - q_T \right\|^2$$

The alignment is :

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

Aligning 3D Data

- If correct correspondences are known, can find correct relative rotation/ translation as closed from solution:
  - **Horn's quaternion method**
  - SVD Arun et al.
  - Orthonormal matrices Horn et al.
  - Dual quaternions Walker et al.

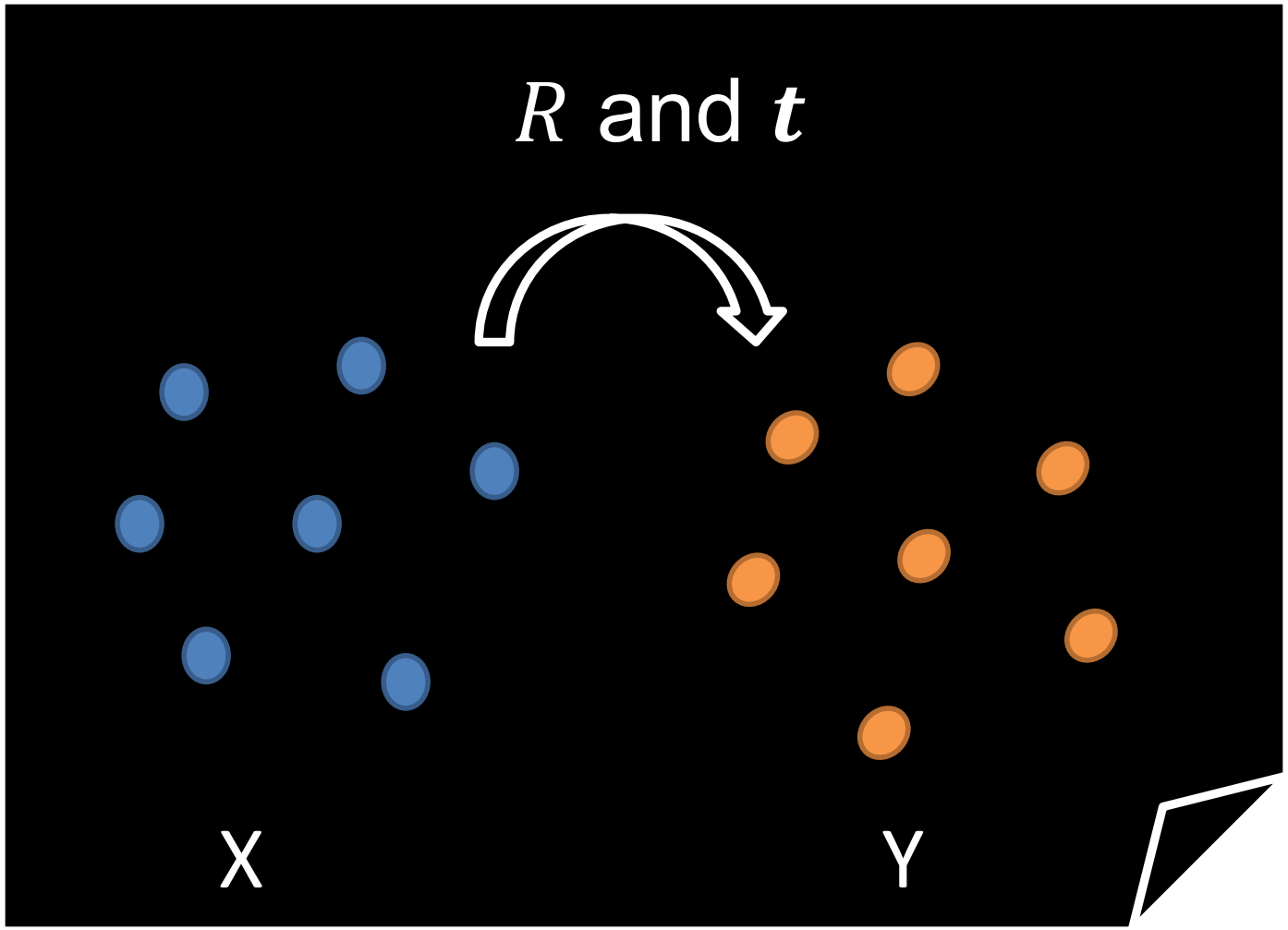


See:
A. Lorusso, D. Eggert, and R. Fisher.
A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations.
In *Proceedings of the 4th British Machine Vision Conference (BMVC '95)*, pages 237 - 246, Birmingham, England, September 1995.

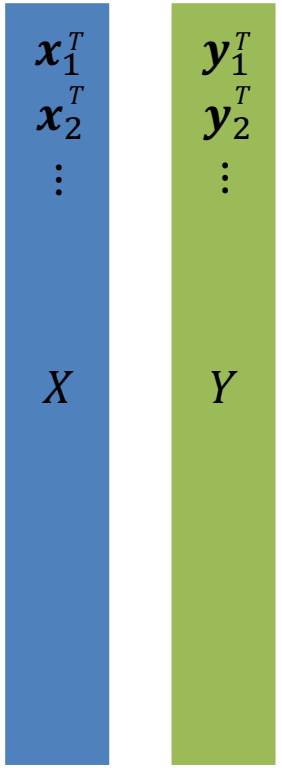
Horn's method

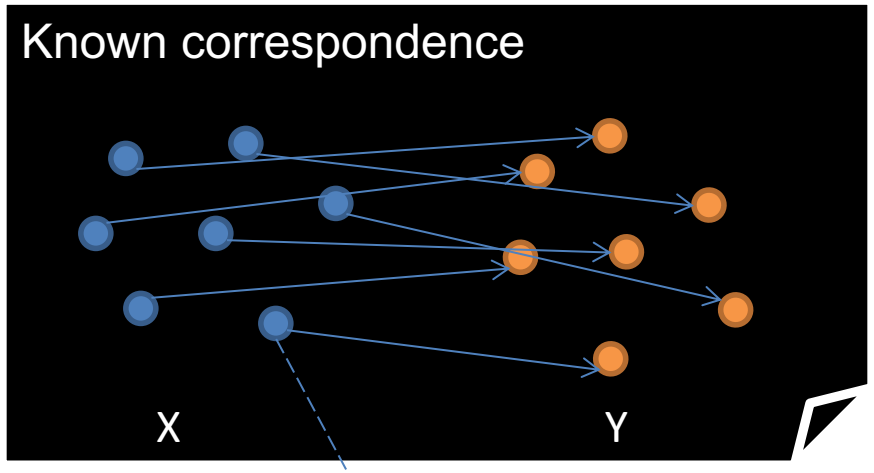
Material by Toru Tamaki, Miho Abe,
Bisser Raytchev, Kazufumi Kaneda

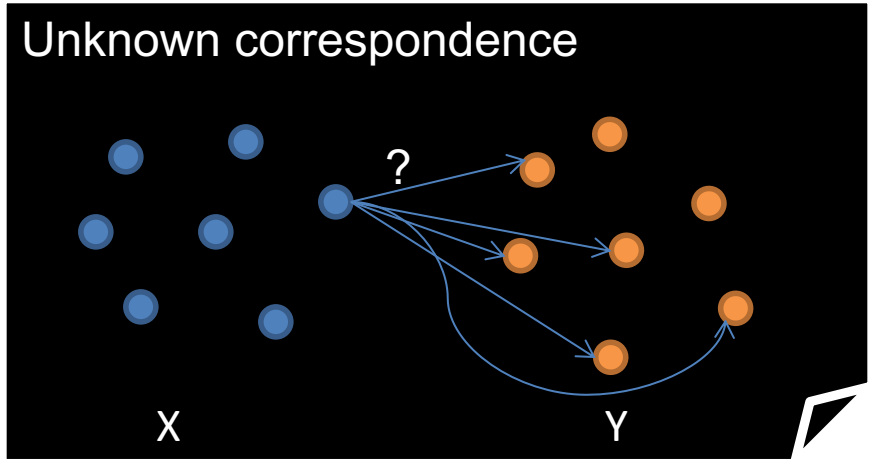
- Input
  - Two point sets: X and Y
- Output
  - Rotation matrix R
  - Translation vector $\boldsymbol{t}$
  - Fitting error



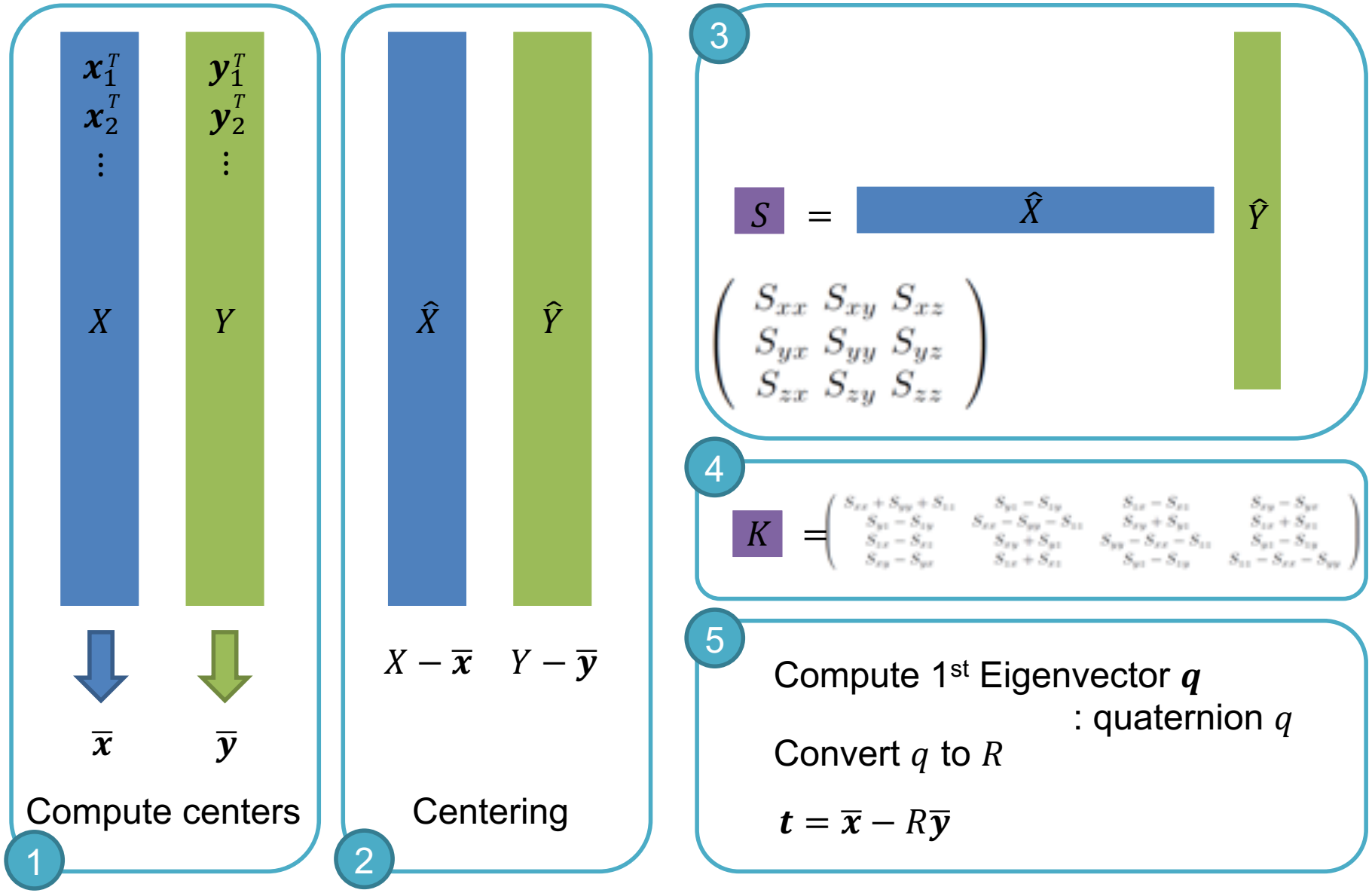
Horn's method: correspondence is known.





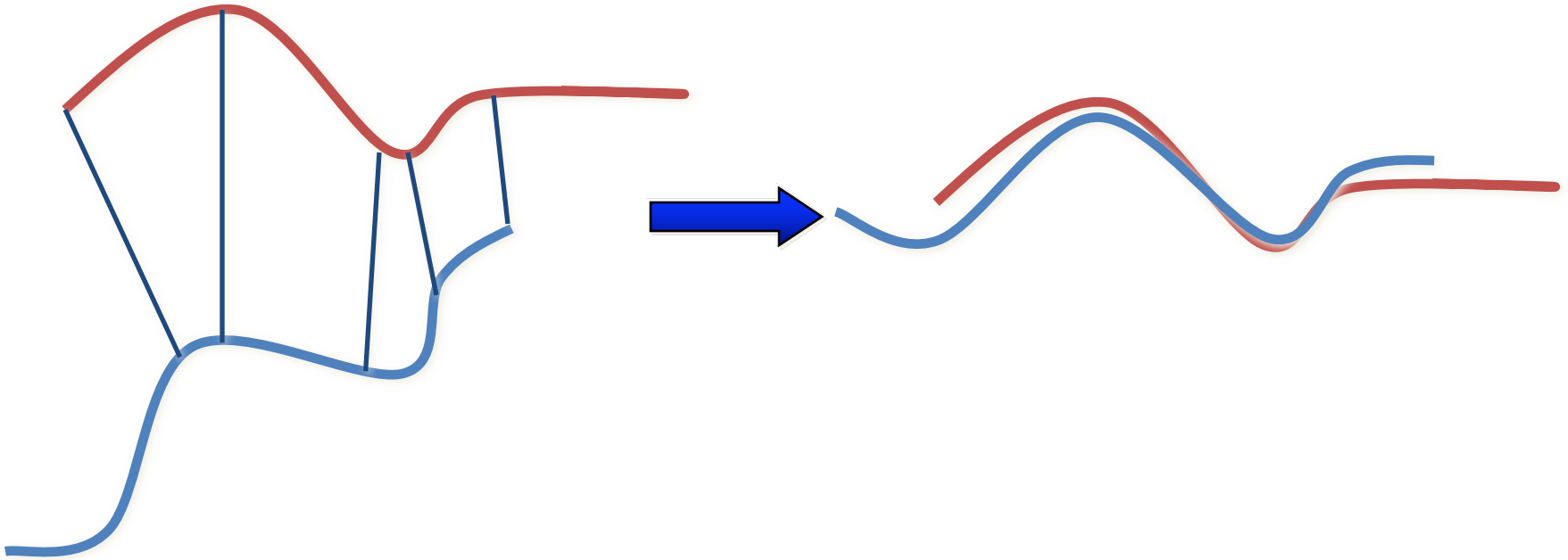


Horn's method: correspondence is known.



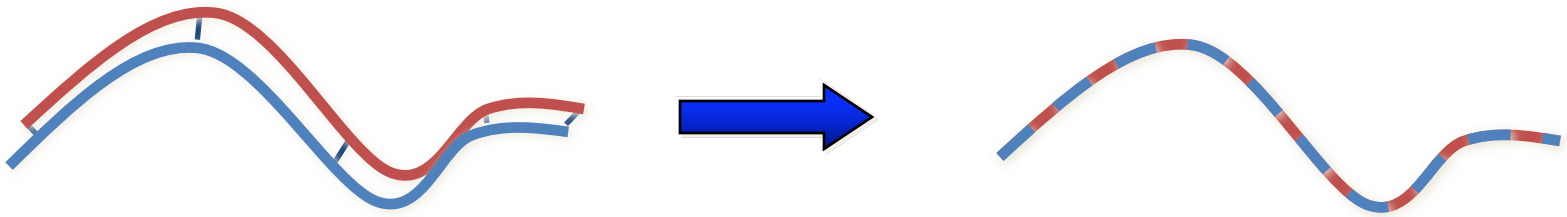
Aligning 3D Data

- How to find correspondences: User input? Feature detection? Signatures?
- Alternative: assume closest points correspond



Aligning 3D Data

- Converges if starting position “close enough“



Closest Point

- Given 2 points r_1 and r_2, the 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 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 :

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

Finding Matches

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

$$y \in M$$

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

$$Y \subseteq M$$

C – the closest point operator

Y – the set of closest points to S

Finding Matches

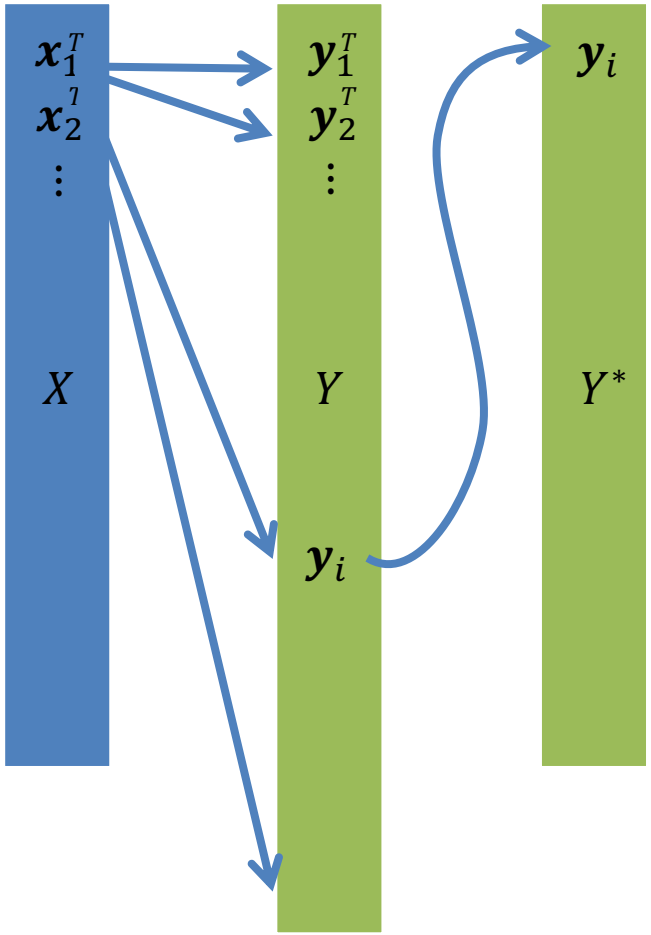
- Finding each match is performed in O(MN) worst case.
- Given Y we can calculate alignment

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

- S is updated to be :

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

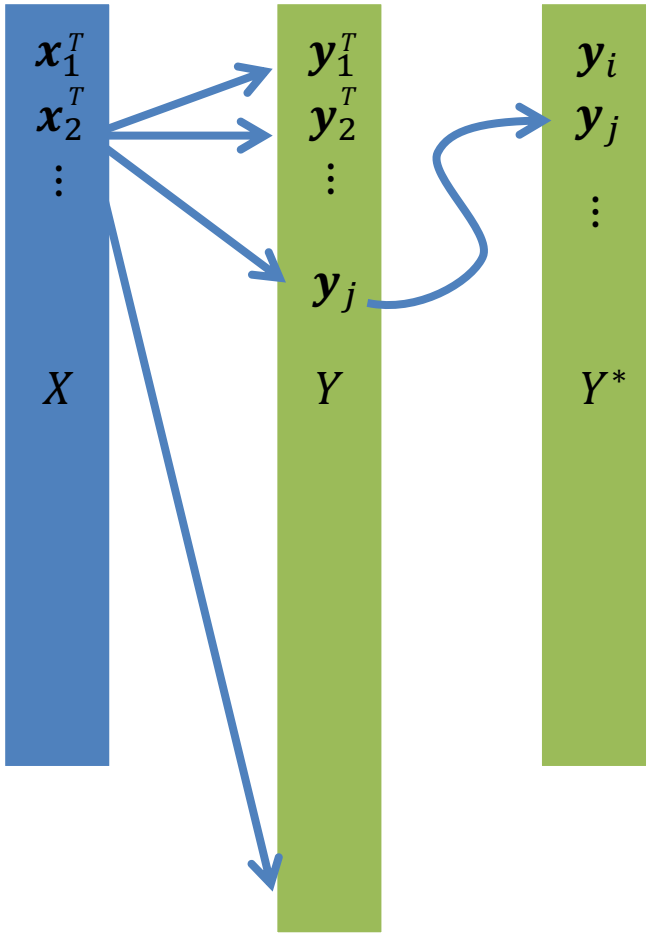
ICP: correspondence is unknown.



Find closest (nearest) point to x_1 in Y

Put the point to Y^*

ICP: correspondence is unknown.



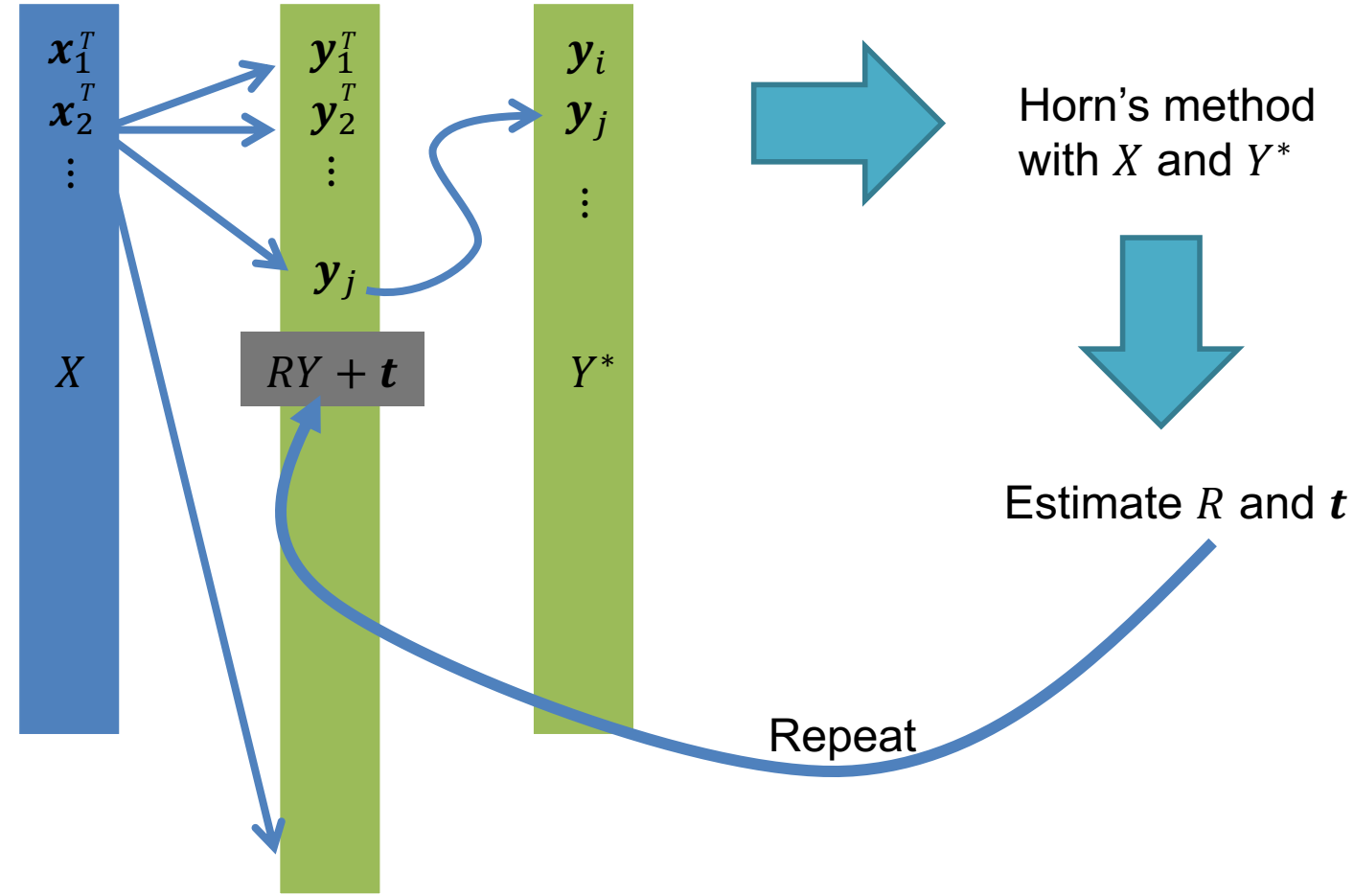
Find closest (nearest) point to $\boldsymbol{x}_1$ in Y

Put the point to Y^*

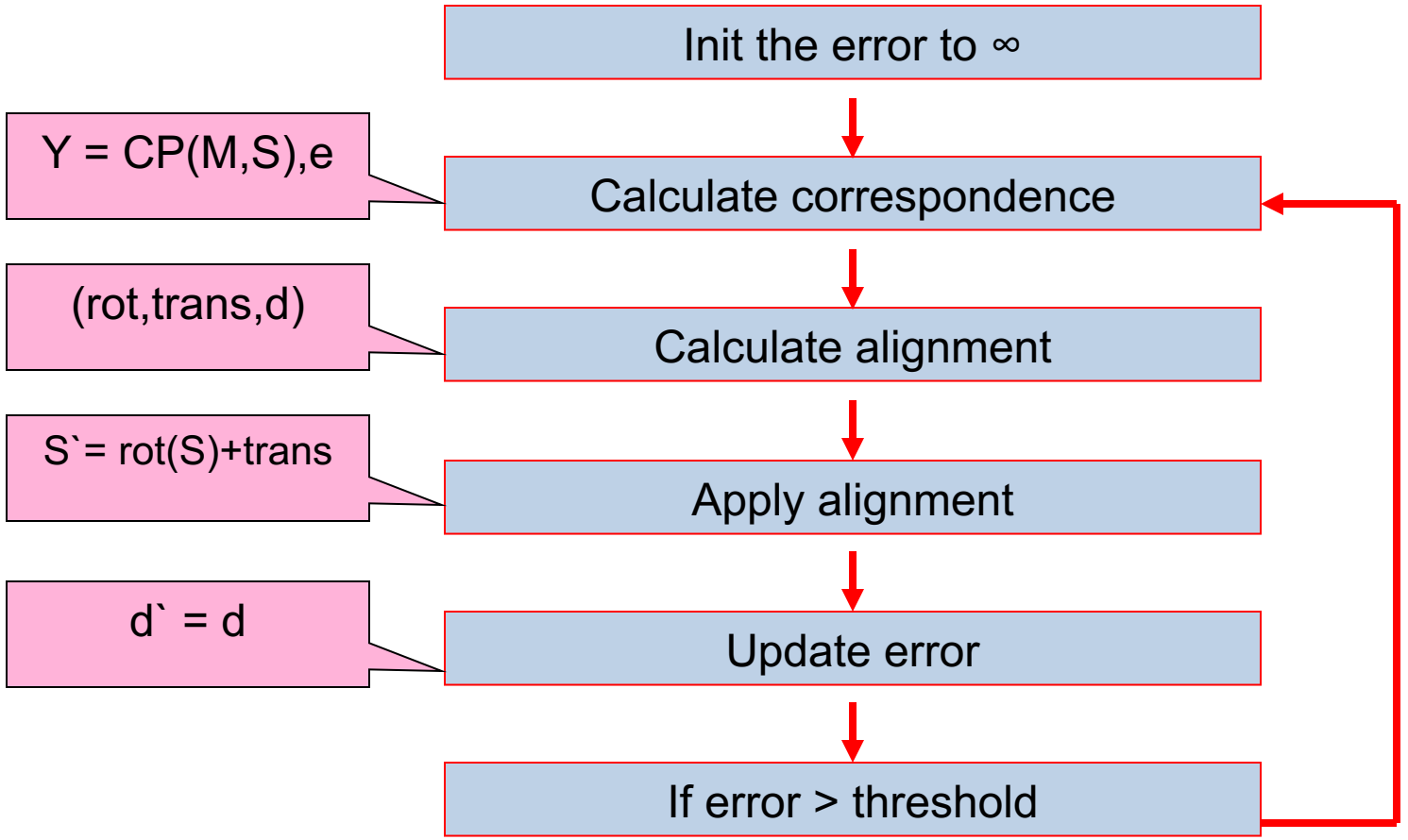
Horn's method with X and Y^*

Estimate R and $\boldsymbol{t}$

ICP: correspondence is unknown.



The Algorithm



The Algorithm

```
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.

Time analysis

Each iteration includes 3 main steps

A. Finding the closest points :

$O(N_M)$ per each point

$O(N_M * N_S)$ total.

B. Calculating the alignment: $O(N_S)$

C. Updating the scene: $O(N_S)$

Optimizing the Algorithm

The best match/nearest neighbor problem :

Given **N** records each described by **K** real values (attributes), and a dissimilarity measure **D**, find the **m** records closest to a query record.

Optimizing the Algorithm

- K-D Tree :

  Construction time: O(kn log n)

  Space: O(n)

  Region Query : $O(n^{1-1/k}+k)$

Time analysis

Each iteration includes 3 main steps

A. Finding the closest points :

$O(N_M)$ per each point

$O(N_M \log N_S)$ total.

B. Calculating the alignment: $O(N_S)$

C. Updating the scene: $O(N_S)$

Further optimization: Approximate k-d tree search

ICP Variants

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

1. Selecting sample points (from one or both point clouds)
2. Matching to points to a plane or 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

- Can analyze various aspects of performance:
  - Speed
  - Stability
  - Tolerance to noise and/or outliers
  - Maximum initial misalignment

Rejecting Pairs

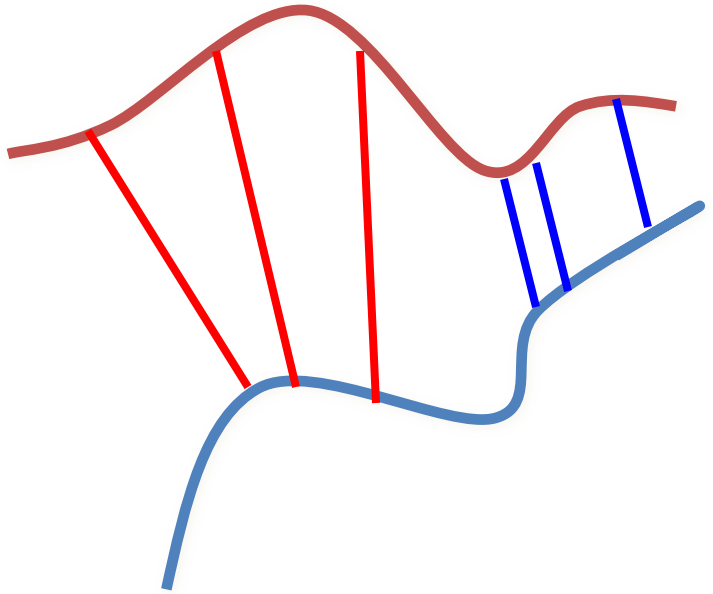
- Corresponding points with point to point distance higher than a given threshold.
- Rejection of worst n% pairs based on some metric.
- Pairs containing points on end vertices.
- Rejection of pairs whose point to point distance is higher than n*σ.
- Rejection of pairs that are not consistent with their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

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

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

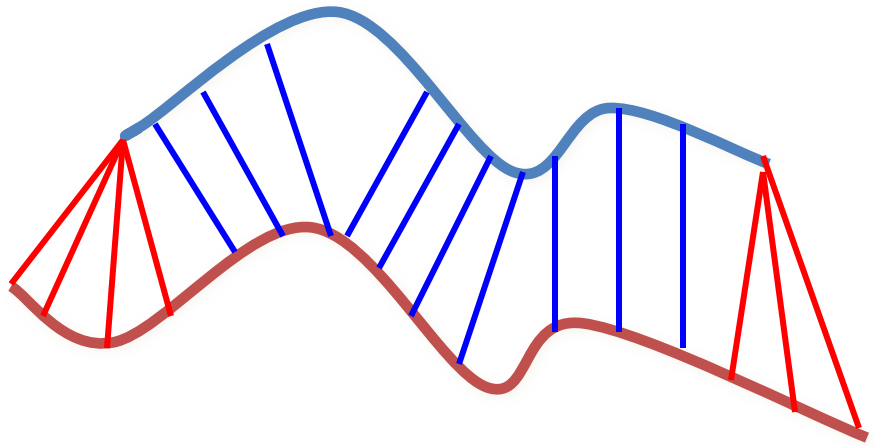
Rejecting Pairs

Distance thresholding



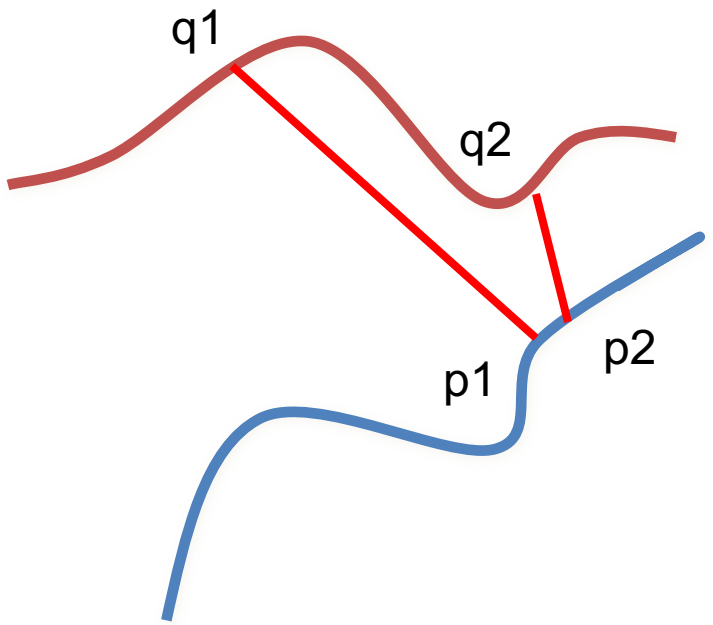
Rejecting Pairs

Points on end vertices

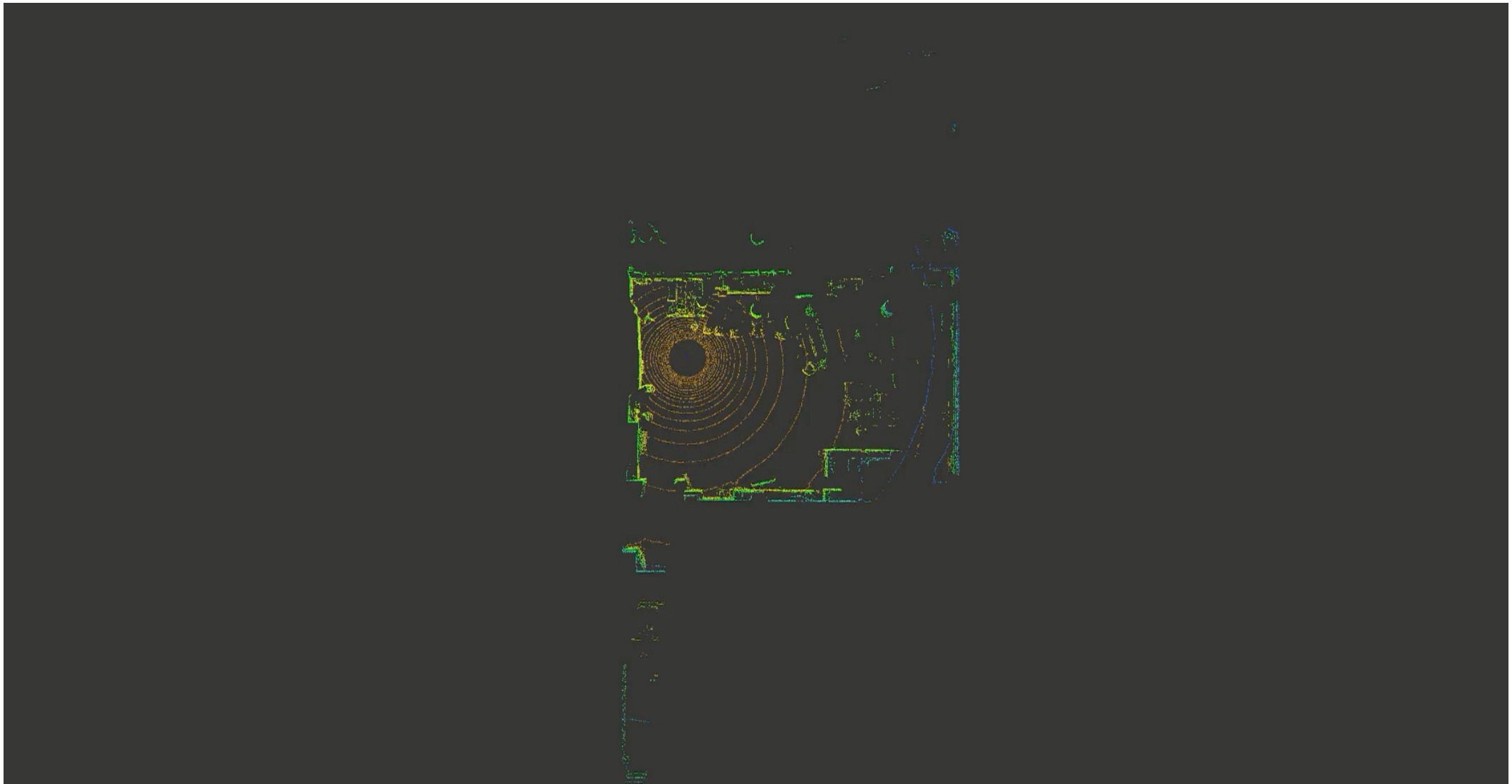


Rejecting Pairs

Inconsistent Pairs



BLAM: ICP in action




PrimeSense