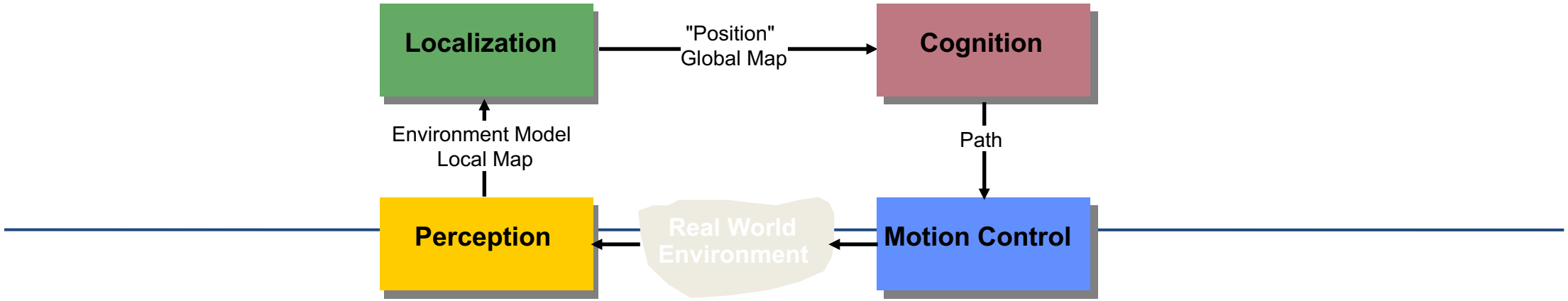
上海科技大学

ShanghaiTech University

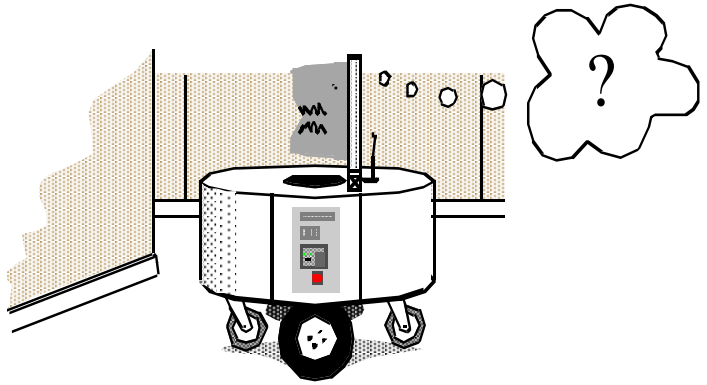
CS283: **Robotics Fall 2016: Localization**

Sören Schwertfeger / 师泽仁

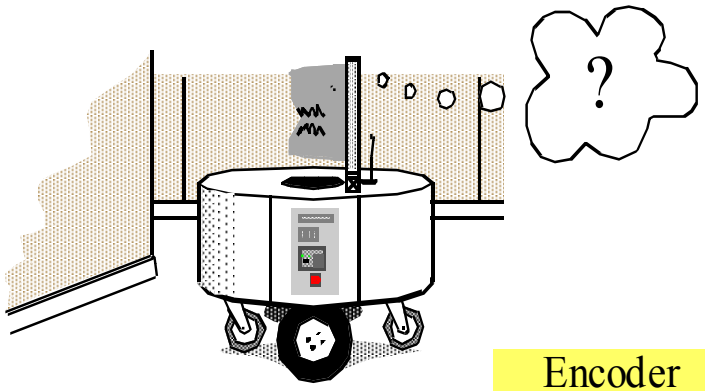
ShanghaiTech University

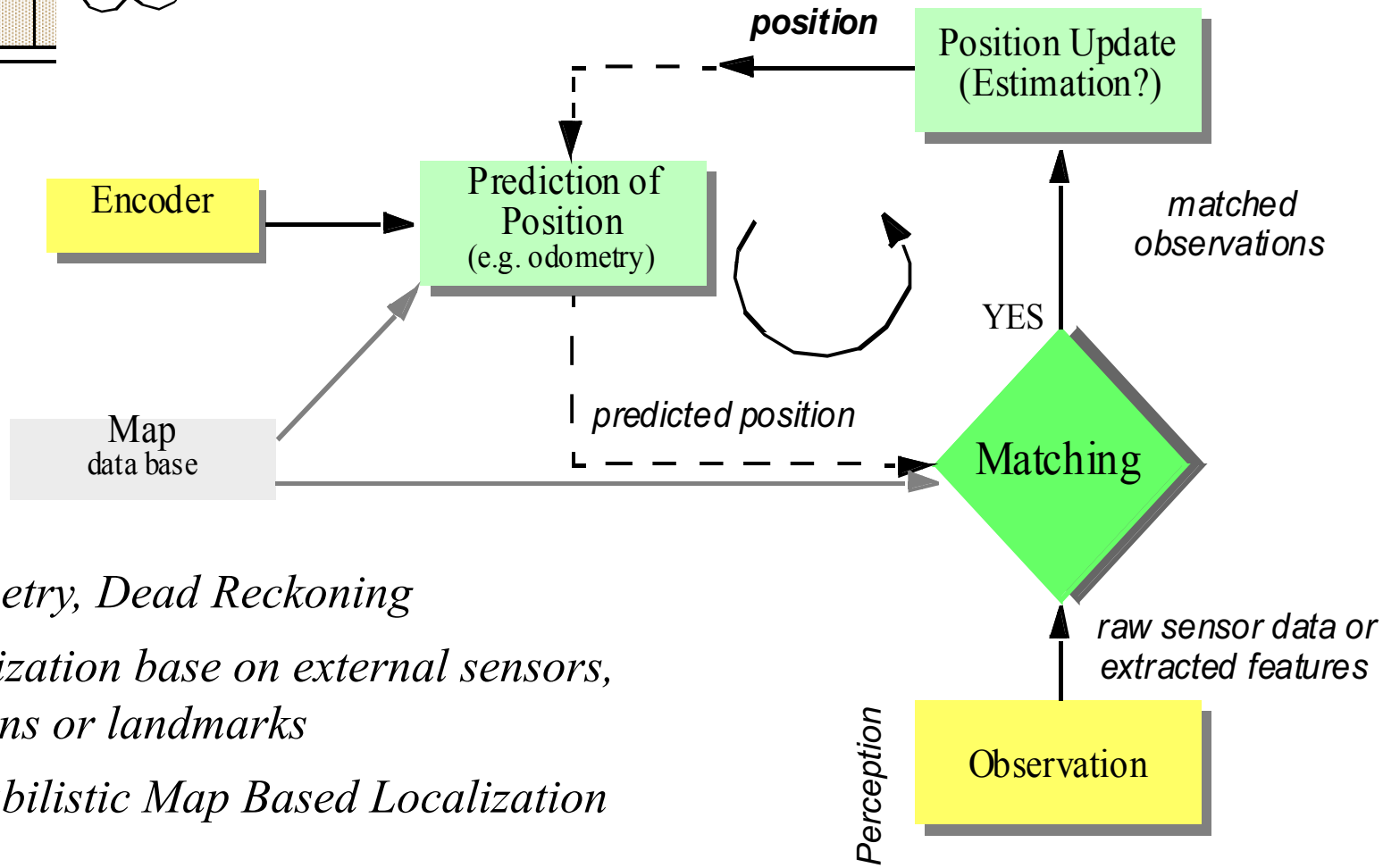


LOCALIZATION



Map based localization

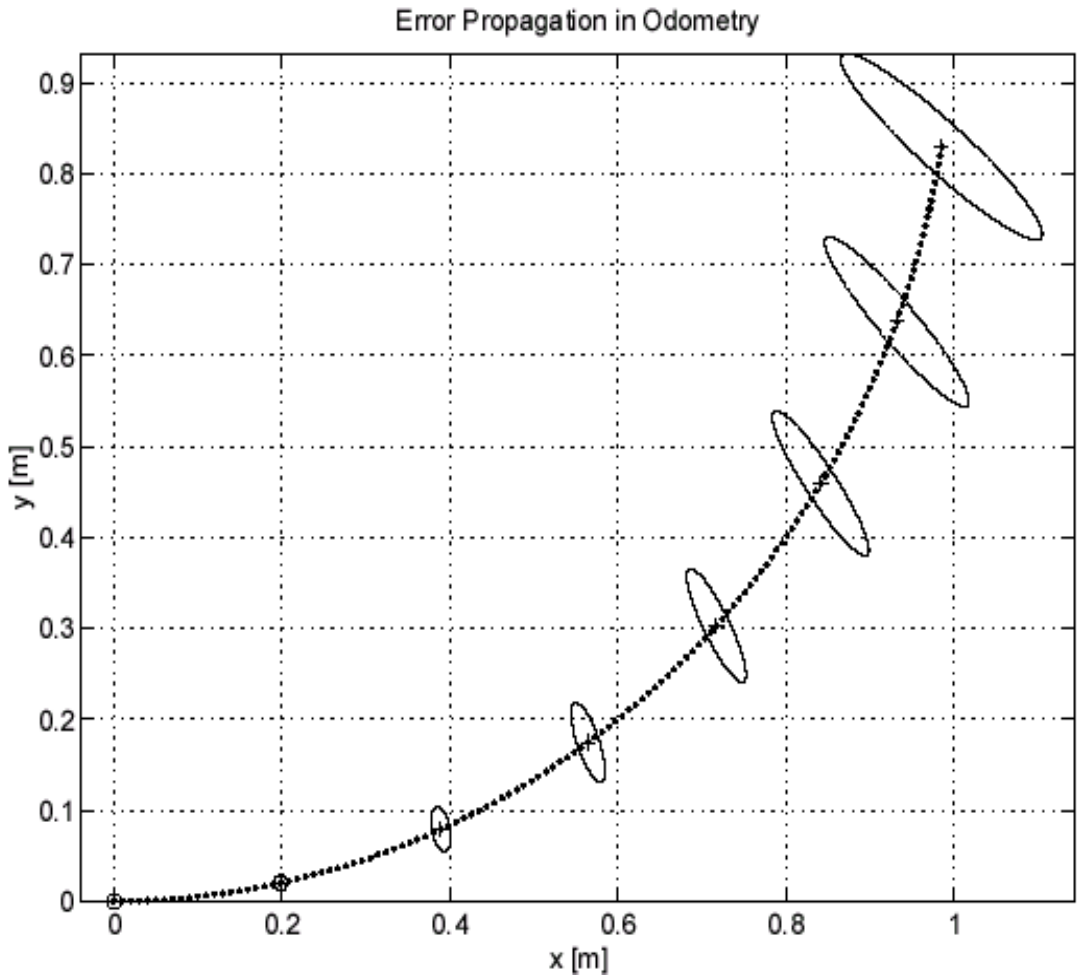




- *Odometry, Dead Reckoning*
- *Localization base on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

Odometry: Growth of Pose uncertainty for Movement on a Circle

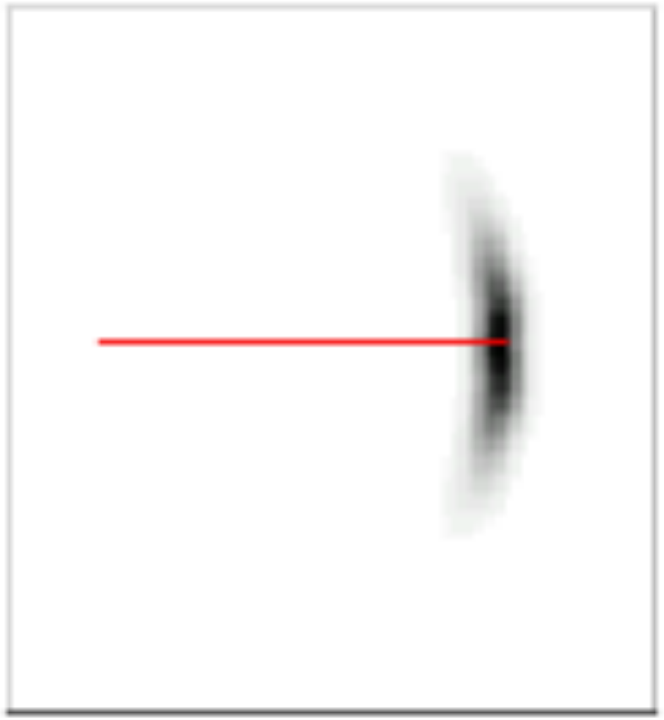
- Note: Errors ellipse in does not remain perpendicular to the direction of movement!

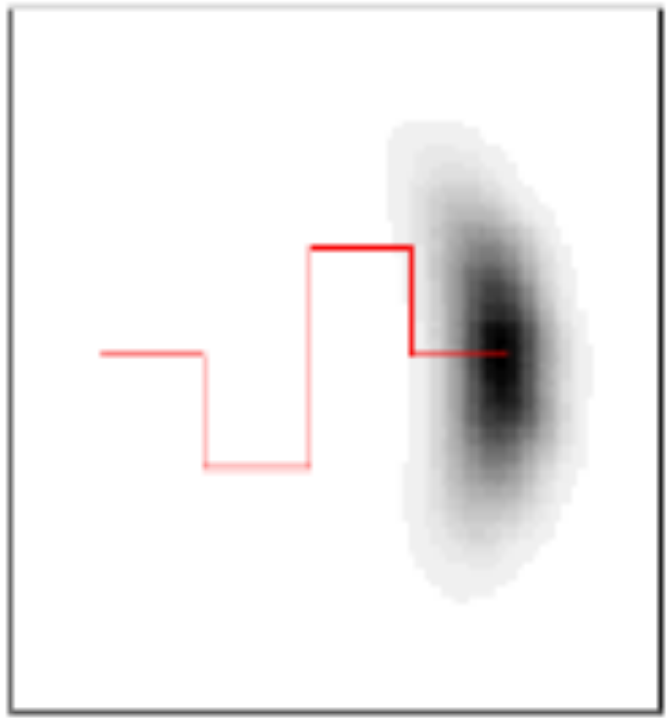


Odometry: example of non-Gaussian error model

- Note: Errors are not shaped like ellipses!

Courtesy AI Lab, Stanford

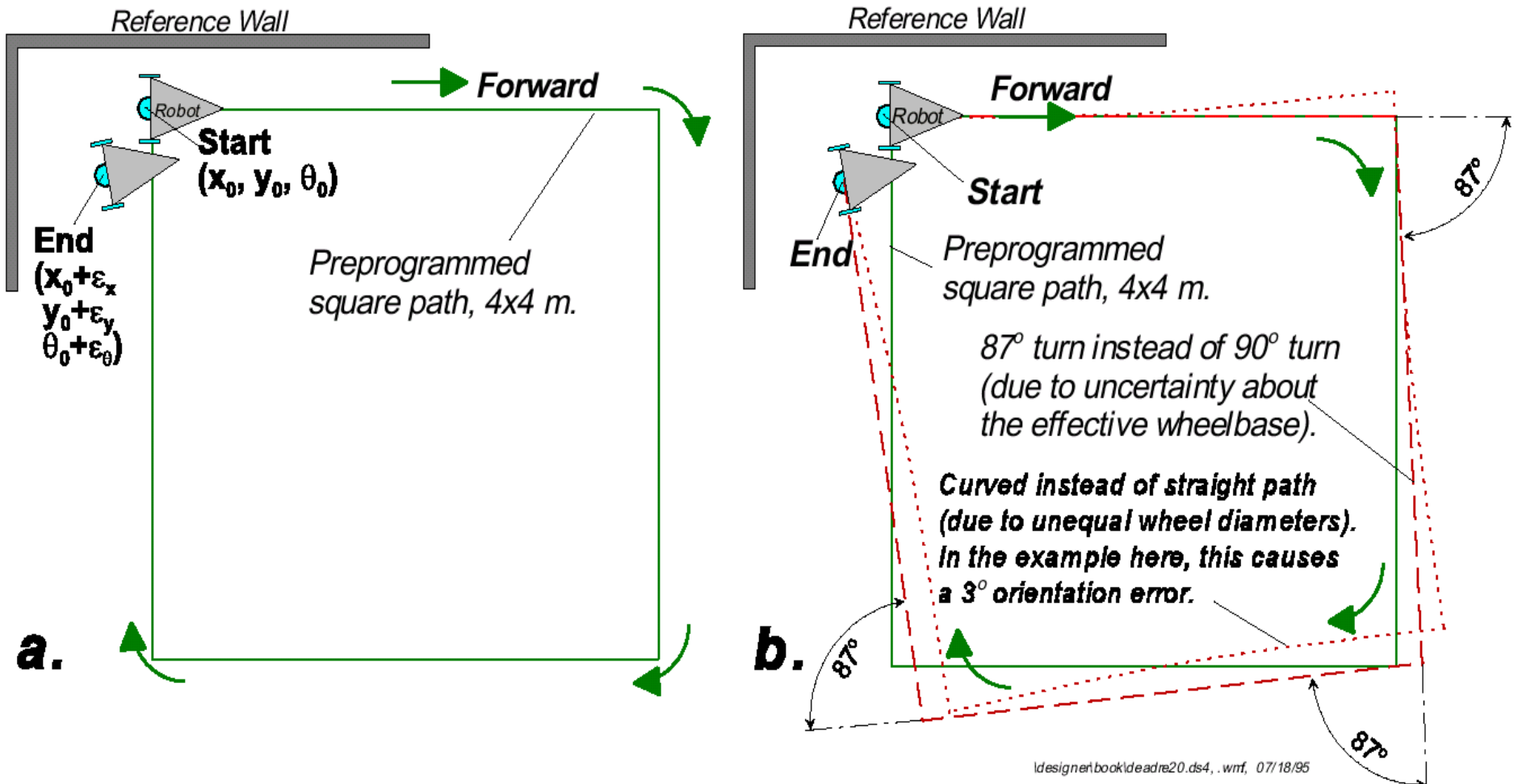




[Fox, Thrun, Burgard, Dellaert, 2000]

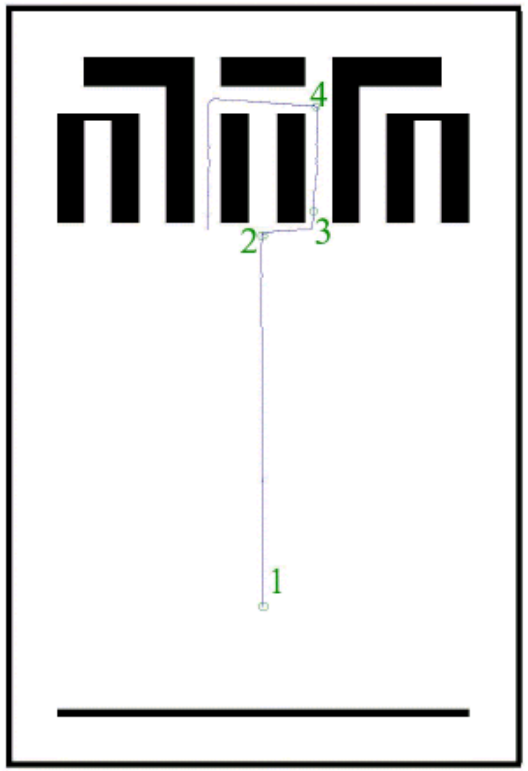
Odometry: Calibration of Errors

- The unidirectional square path experiment

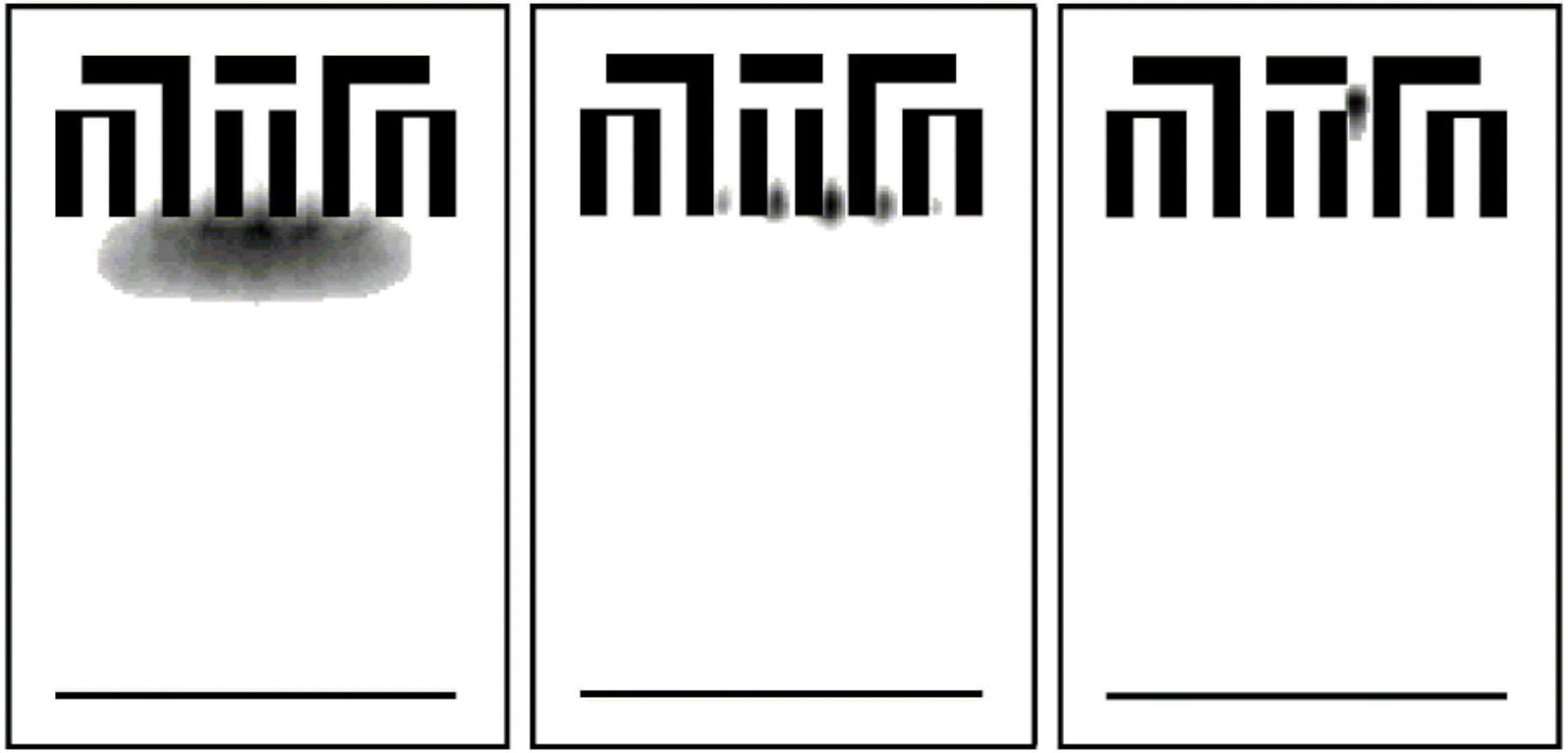


Grid-based Representation - Multi Hypothesis

Courtesy of W. Burgard

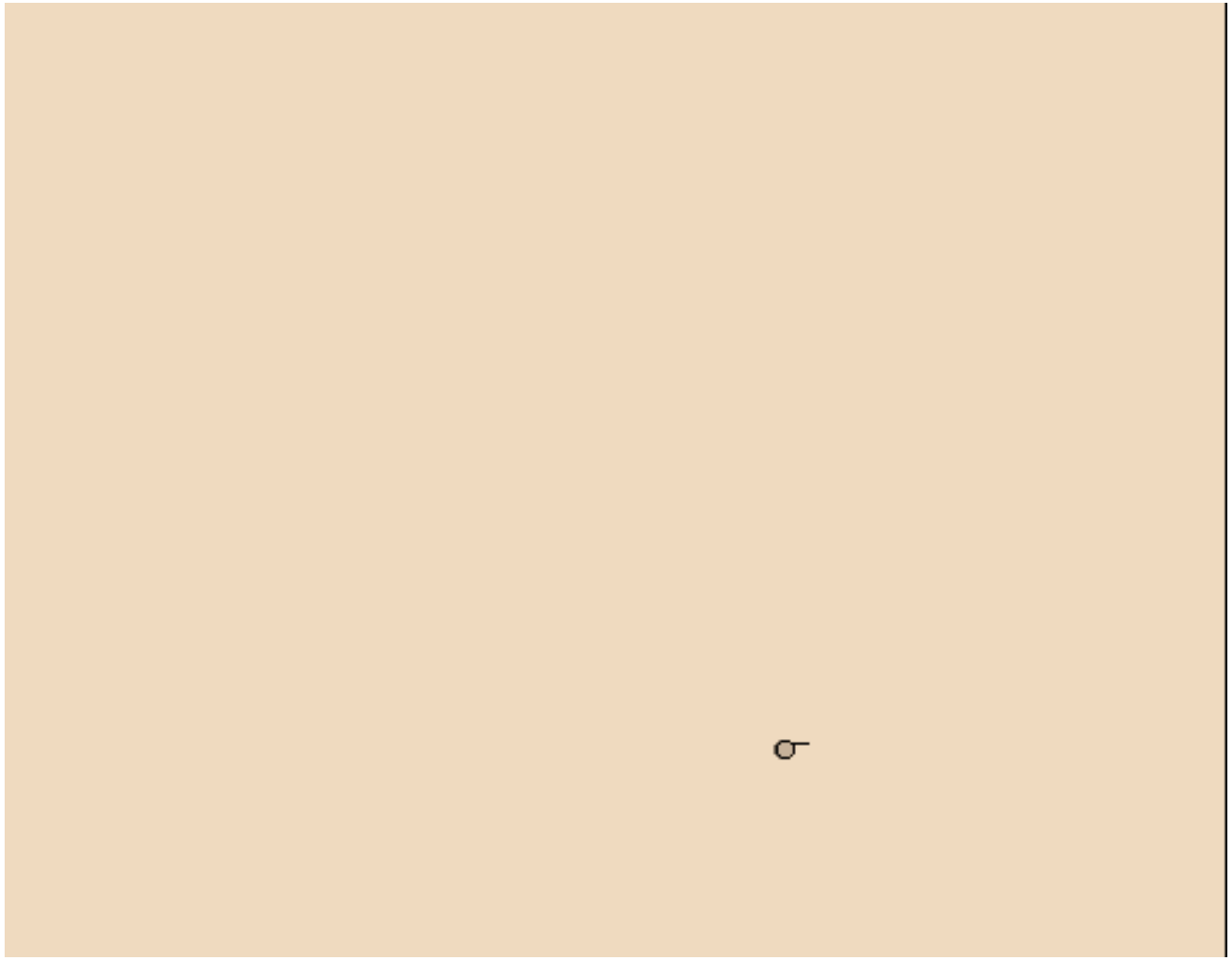


Path of the robot



Belief states at positions 2, 3 and 4

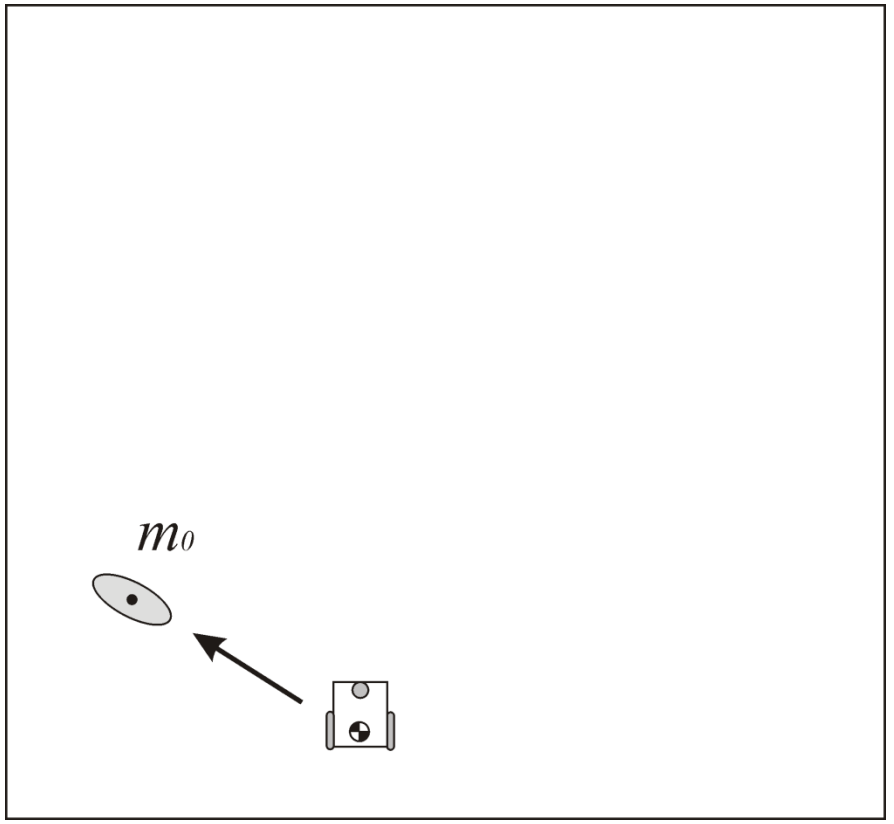
Scan Matching:
compare to sensor data from previous scan



Courtesy of S. Thrun

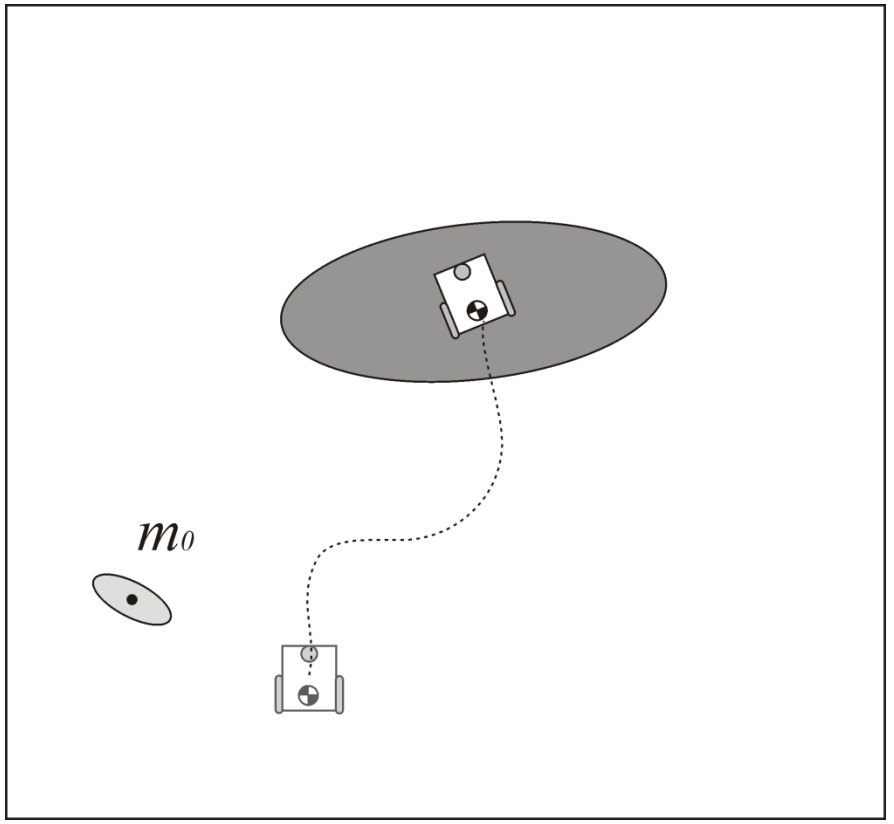
SLAM overview

- Let us assume that the robot uncertainty at its initial location is zero.
- From this position, the robot observes a feature which is mapped with an uncertainty related to the exteroceptive sensor error model



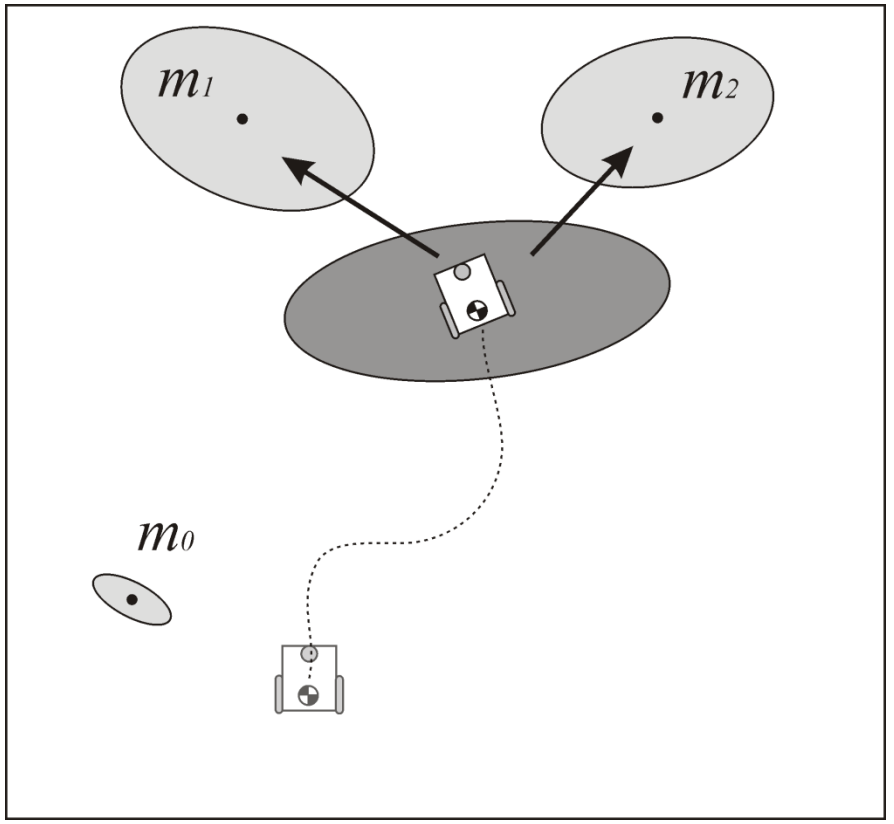
SLAM overview

- As the robot moves, its pose uncertainty increases under the effect of the errors introduced by the odometry



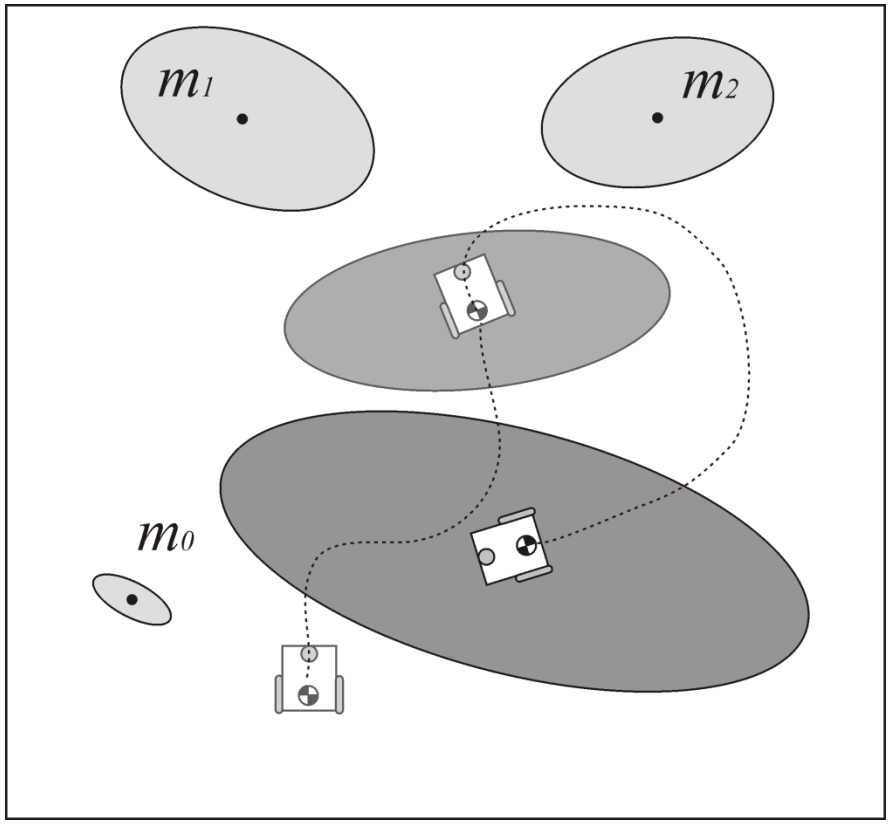
SLAM overview

- At this point, the robot observes two features and maps them with an uncertainty which results from the combination of the measurement error with the robot pose uncertainty
- From this, we can notice that the map becomes correlated with the robot position estimate. Similarly, if the robot updates its position based on an observation of an imprecisely known feature in the map, the resulting position estimate becomes correlated with the feature location estimate.



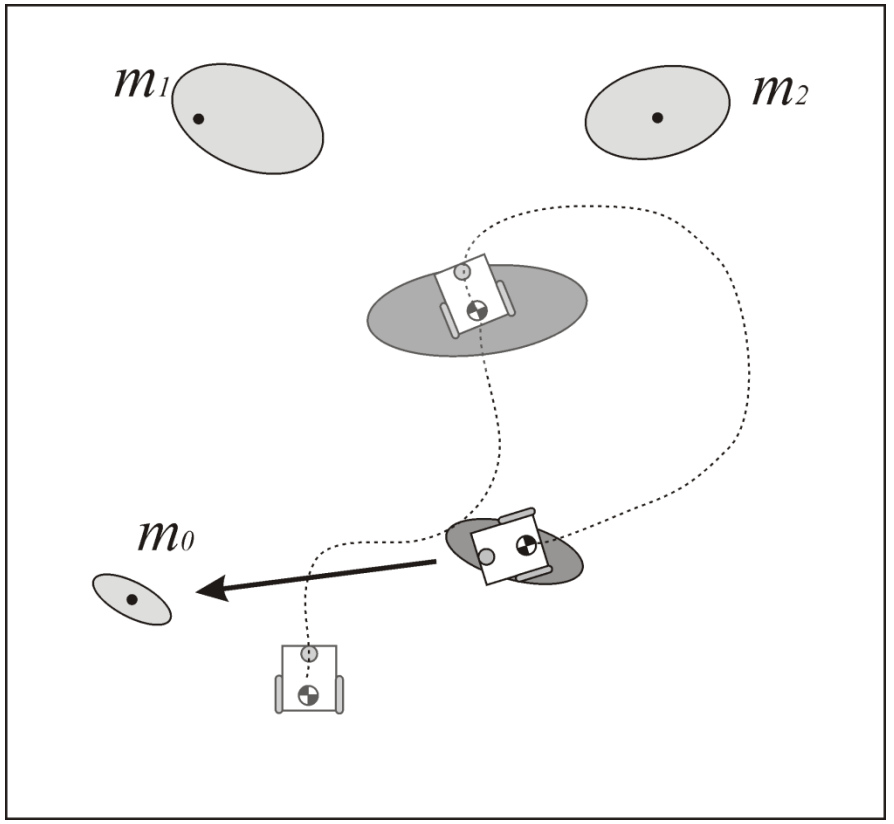
SLAM overview

- The robot moves again and its uncertainty increases under the effect of the errors introduced by the odometry



SLAM overview

- In order to reduce its uncertainty, the robot must observe features whose location is relatively well known. These features can for instance be landmarks that the robot has already observed before.
- In this case, the observation is called *loop closure detection.*
- When a loop closure is detected, the robot pose uncertainty shrinks.
- At the same time, the map is updated and the uncertainty of other observed features and all previous robot poses also reduce

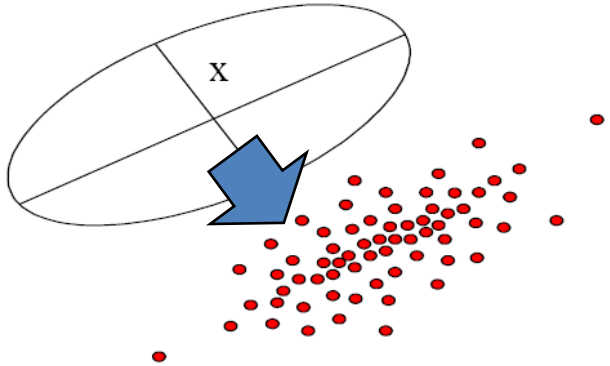


The Three SLAM paradigms

- Most of the SLAM algorithms are based on the following three different approaches:
  - Extended Kalman Filter SLAM: (called EKF SLAM)
  - Particle Filter SLAM: (called FAST SLAM)
  - Graph-Based SLAM

Particle Filter SLAM: FastSLAM

- **FastSLAM approach**
  - Using particle filters.
  - Particle filters: mathematical models that represent probability distribution as a set of discrete particles that occupy the state space.



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

- Particle filter update
  - Generate new particle distribution using motion model and controls

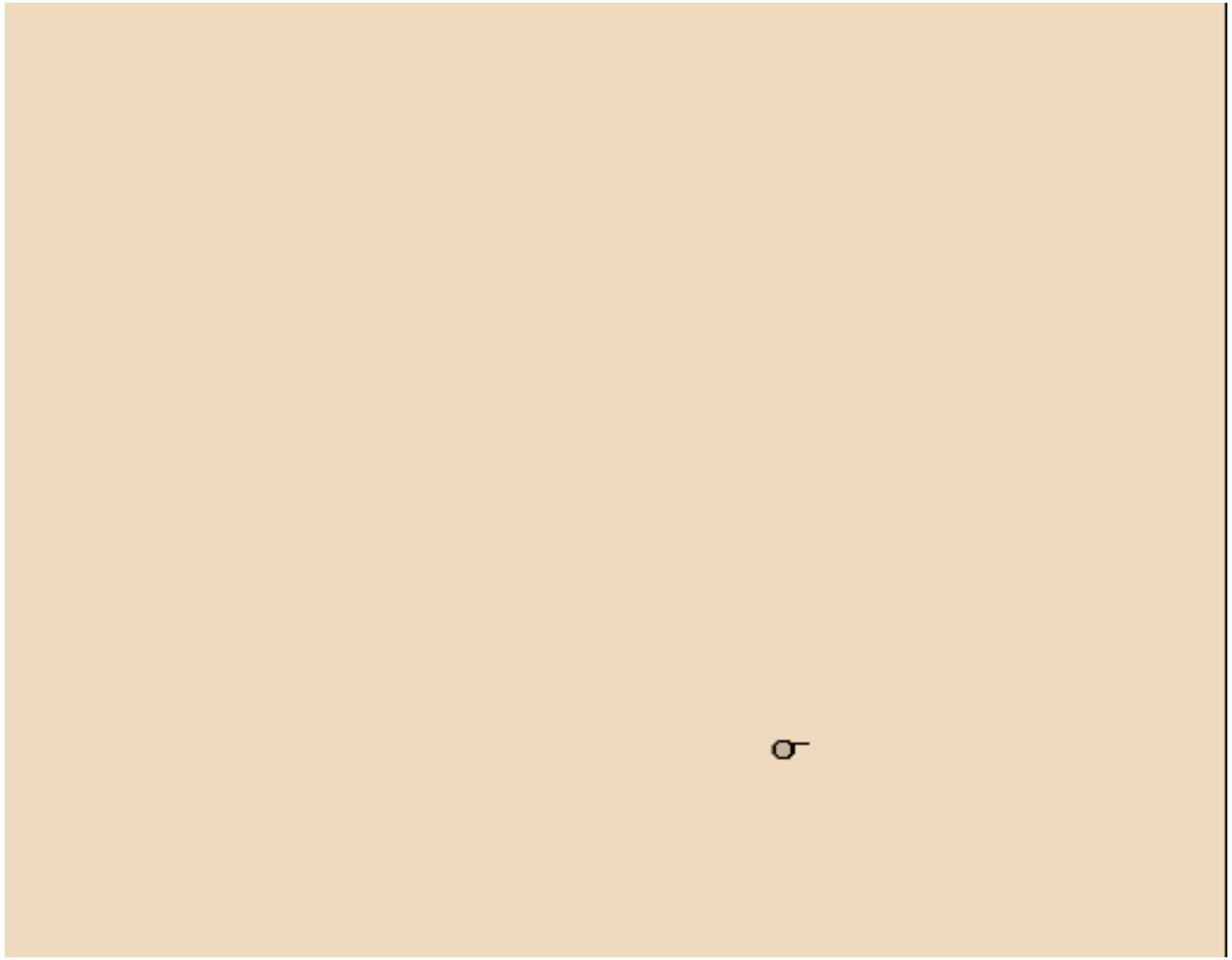
a) For each particle:

1. Compare particle's prediction of measurements with actual measurements
2. Particles whose predictions match the measurements are given a high weight

b) Filter resample:

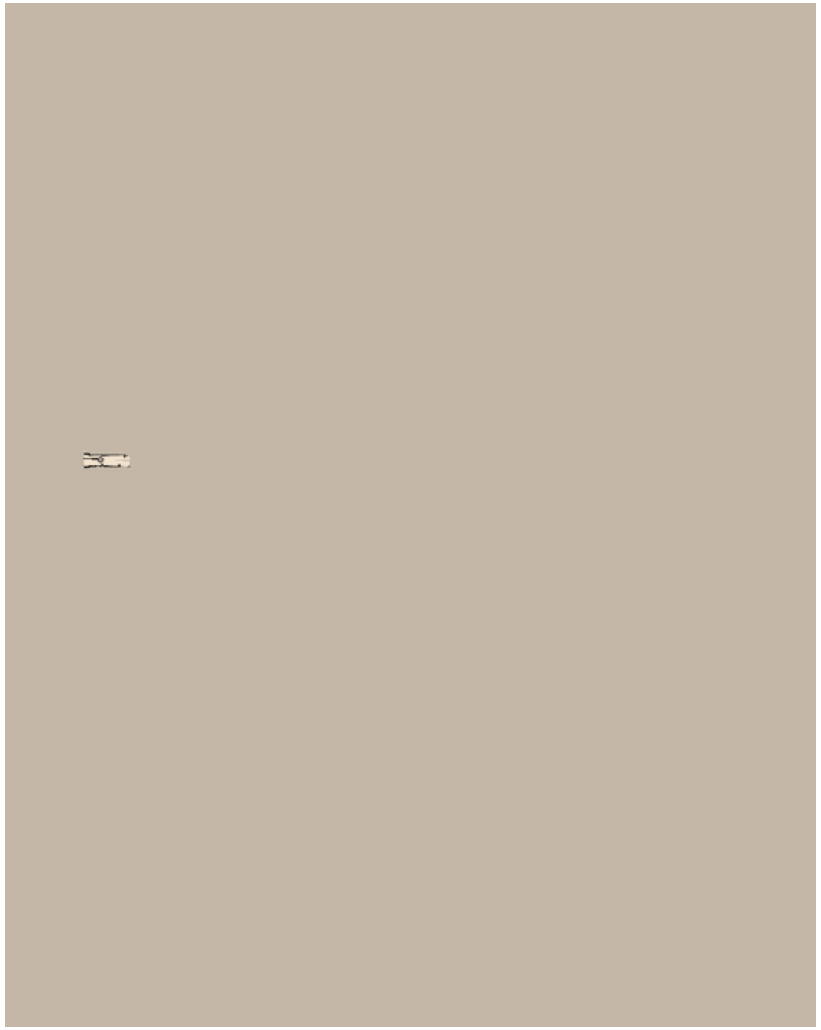
- Resample particles based on weight
- Filter resample
  - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements and randomly draw particles from previous distribution based on weights creating a new distribution.

FAST SLAM example



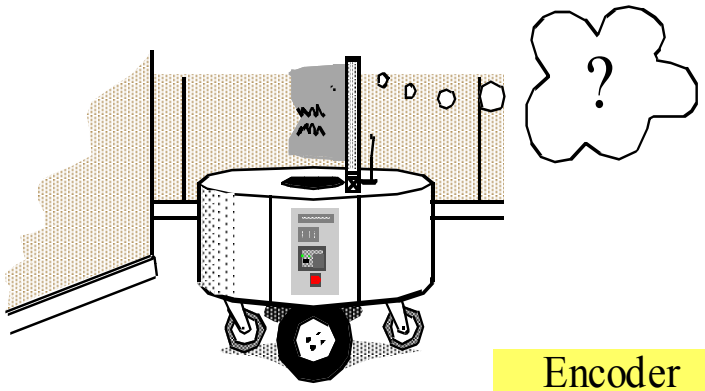
Courtesy of S. Thrun

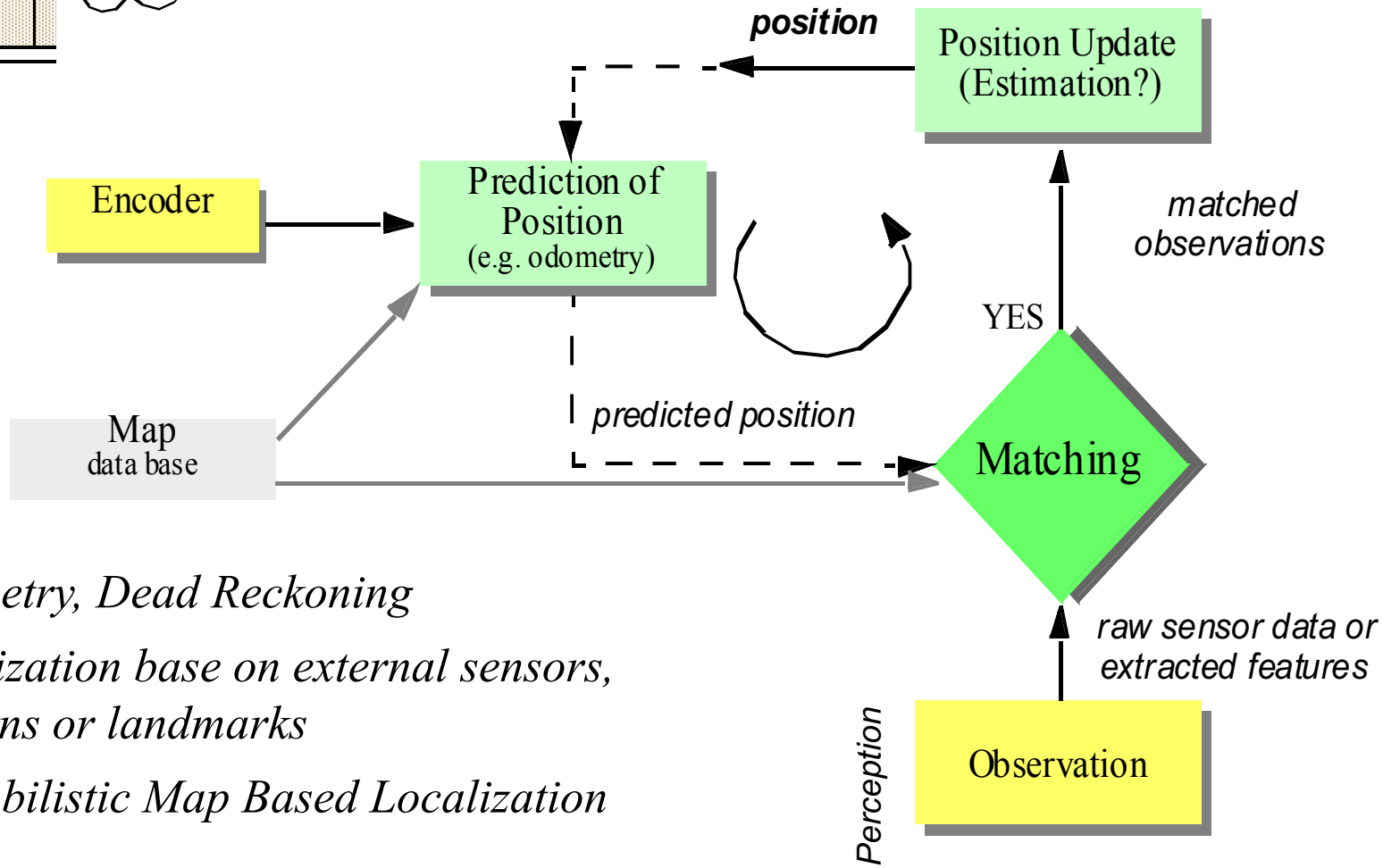
FAST SLAM example



Courtesy of S. Thrun

Map based localization

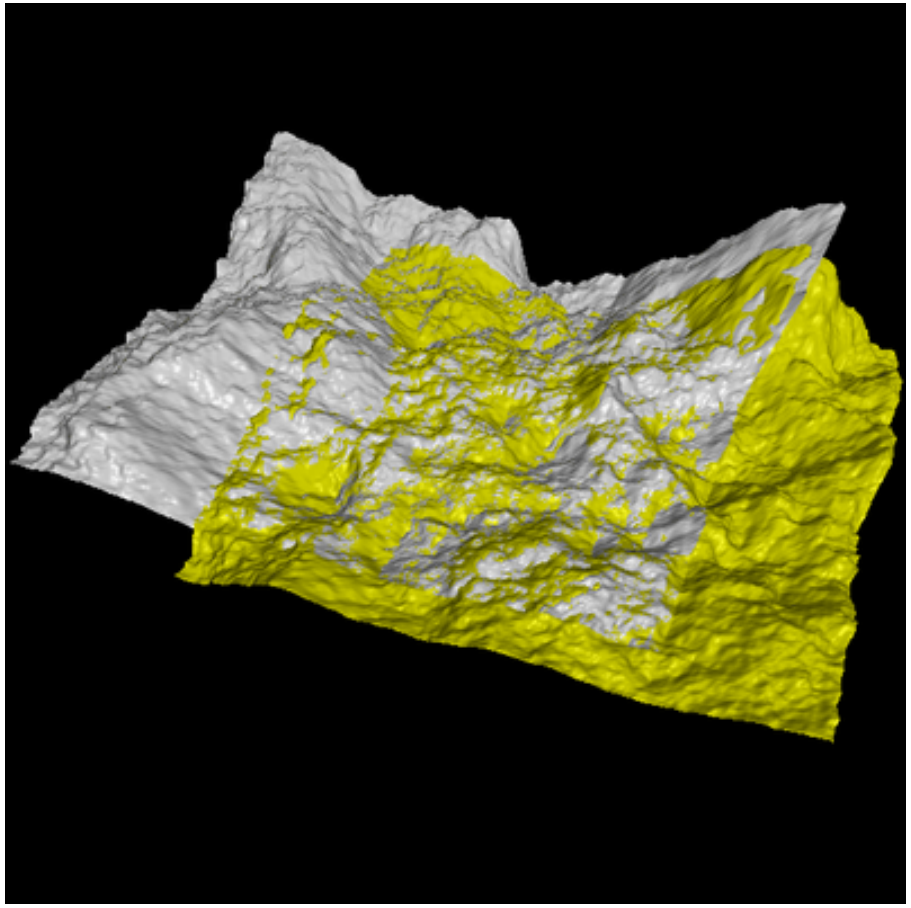




- *Odometry, Dead Reckoning*
- *Localization base on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

ICP: Iterative Closest Points Algorithm

- Align two partially-overlapping point sets (2D or 3D)
- Given initial guess for relative transform



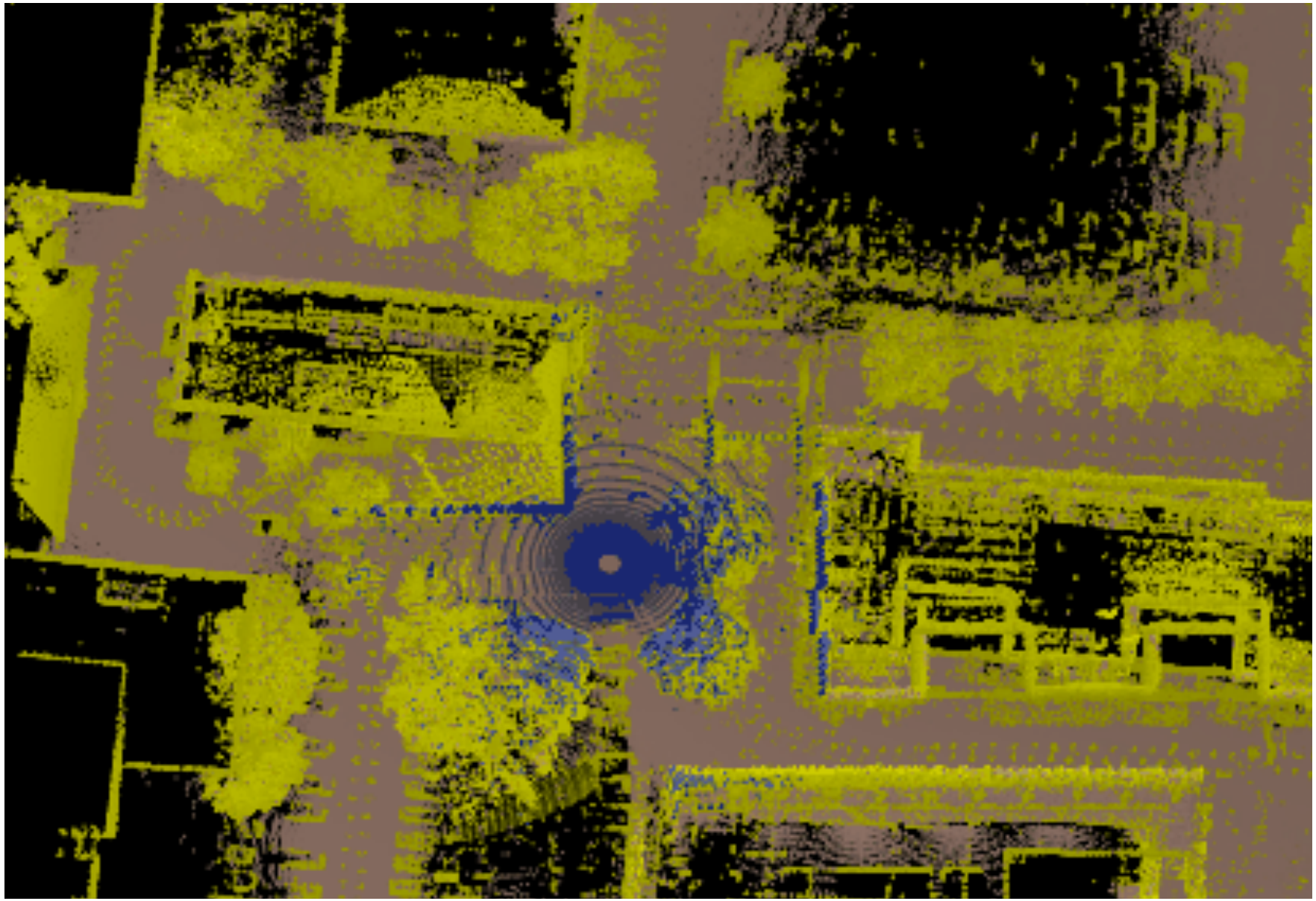
Material derived from Ronen Gvili : *www.cs.tau.ac.il/~dcor/Graphics/adv-slides/**ICP**.ppt*

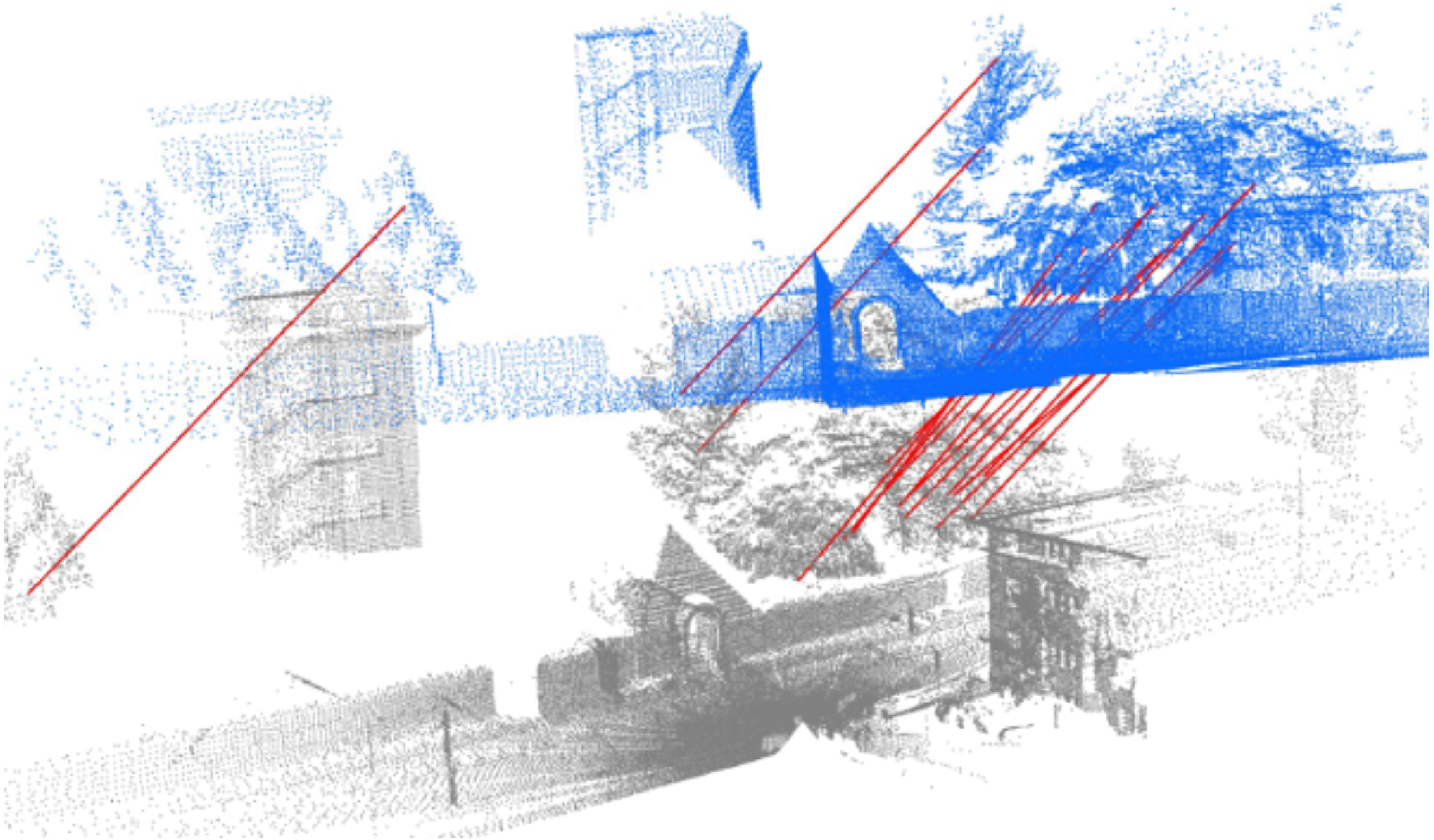
Data Types

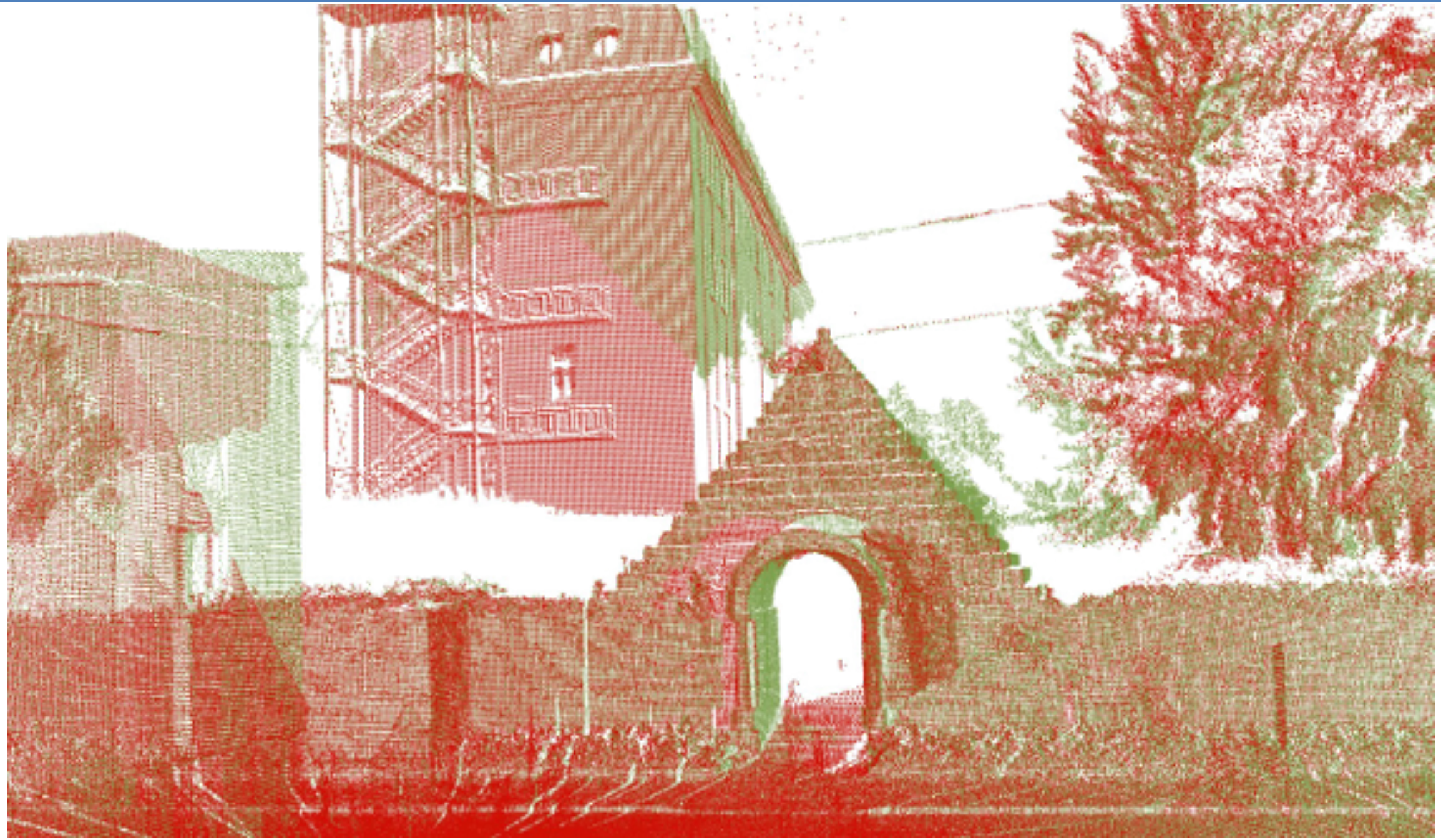
- Point sets
- Line segment sets (polylines)
- Implicit curves : f(x,y,z) = 0
- Parametric curves : (x(u),y(u),z(u))
- Triangle sets (meshes)
- Implicit surfaces : s(x,y,z) = 0
- Parametric surfaces (x(u,v),y(u,v),z(u,v)))

Motivation

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

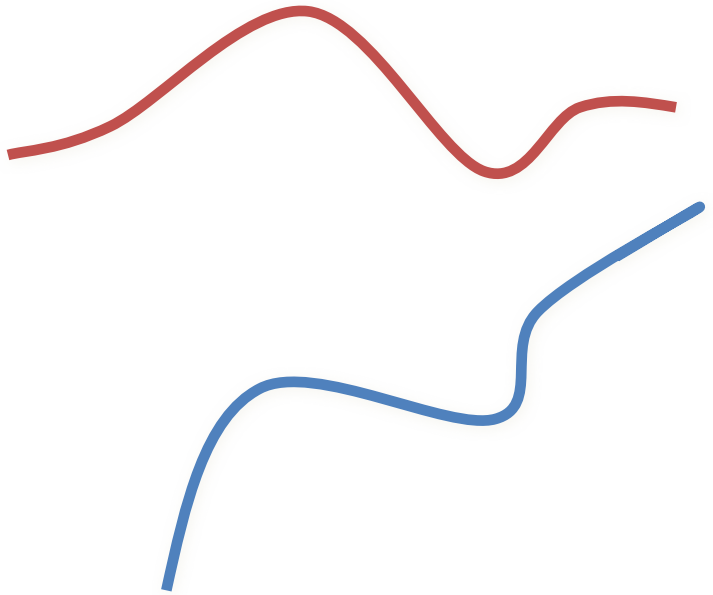






Aligning 3D Data

- Continuous lines or a set of points…

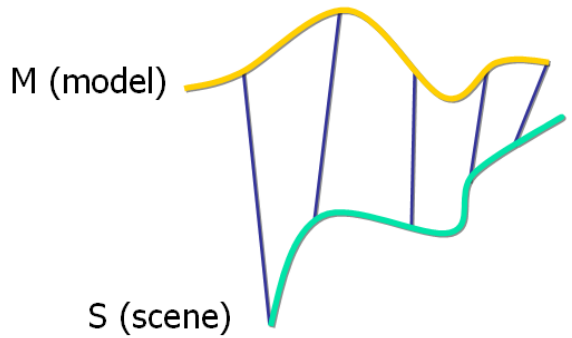


Corresponding Point Set Alignment

- 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 :

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

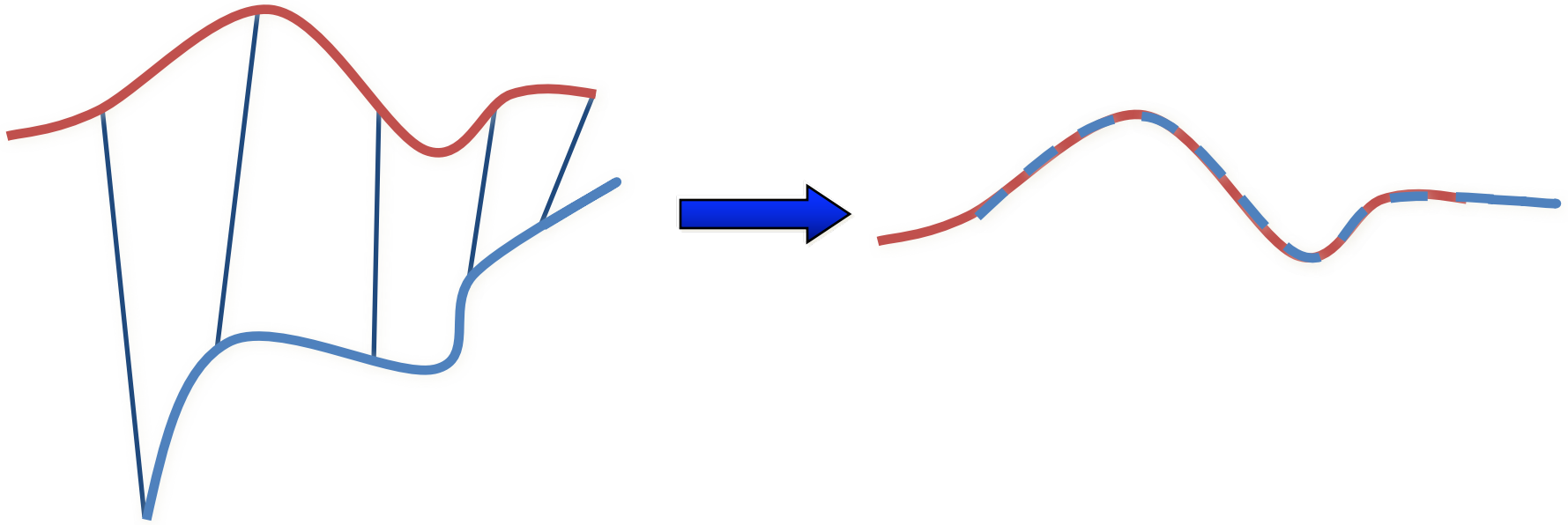
$$f(q) = \frac{1}{N_S}\sum_{i=1}^{N_S}\left\|m_i - R(q_R)s_i - q_T\right\|^2$$

The alignment is :

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

Aligning 3D Data

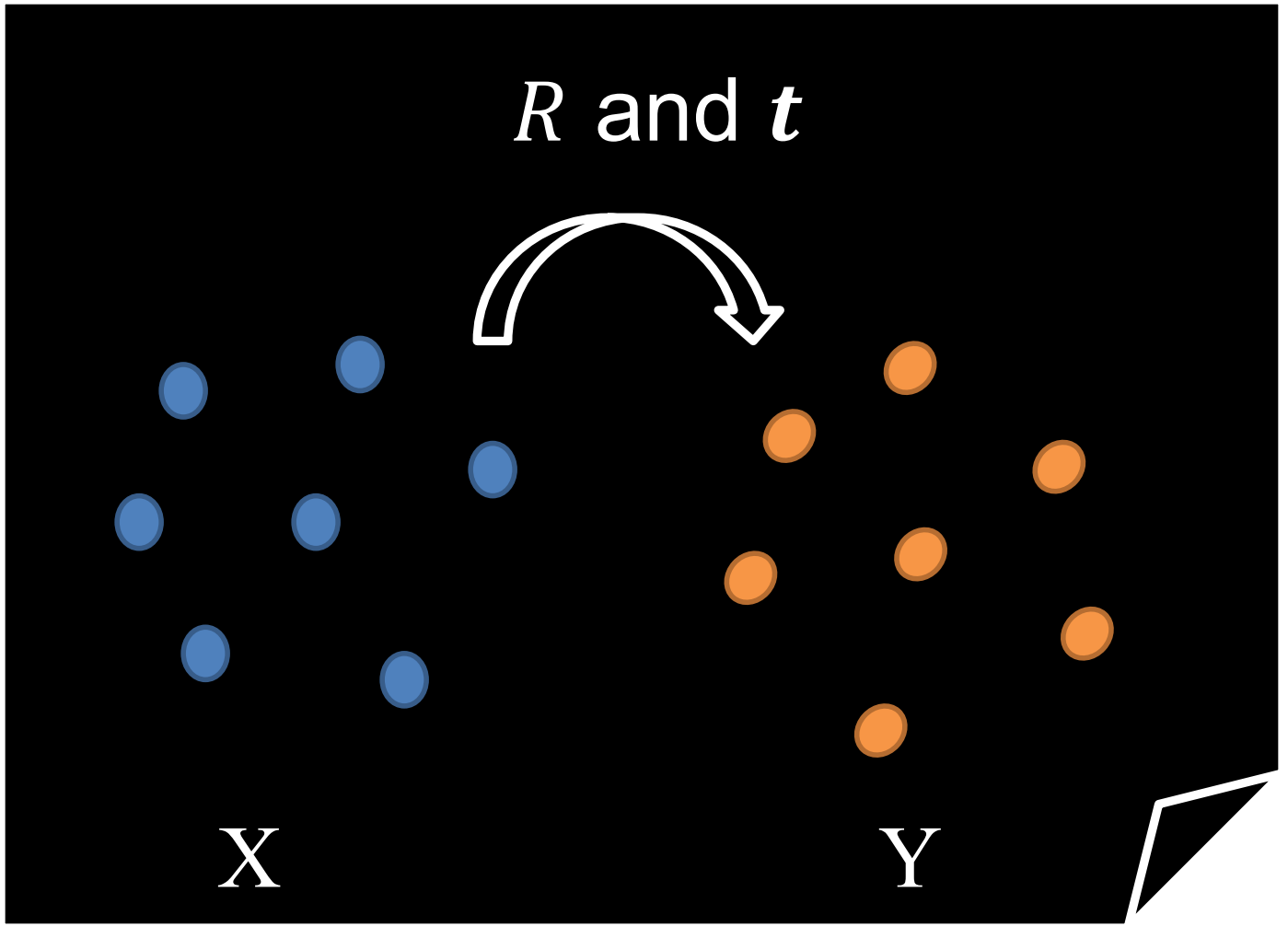
- If correct correspondences are known, can find correct relative rotation/ translation, e.g. using **Horn's method**



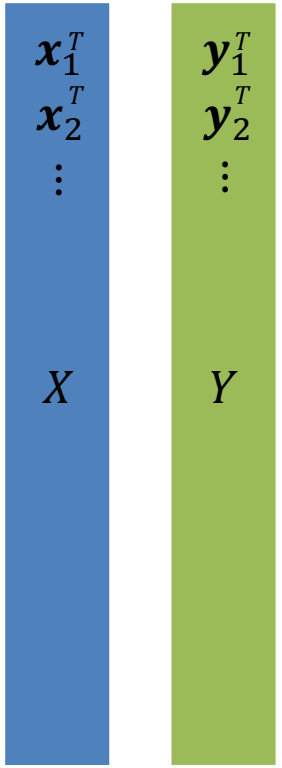
Horn's method

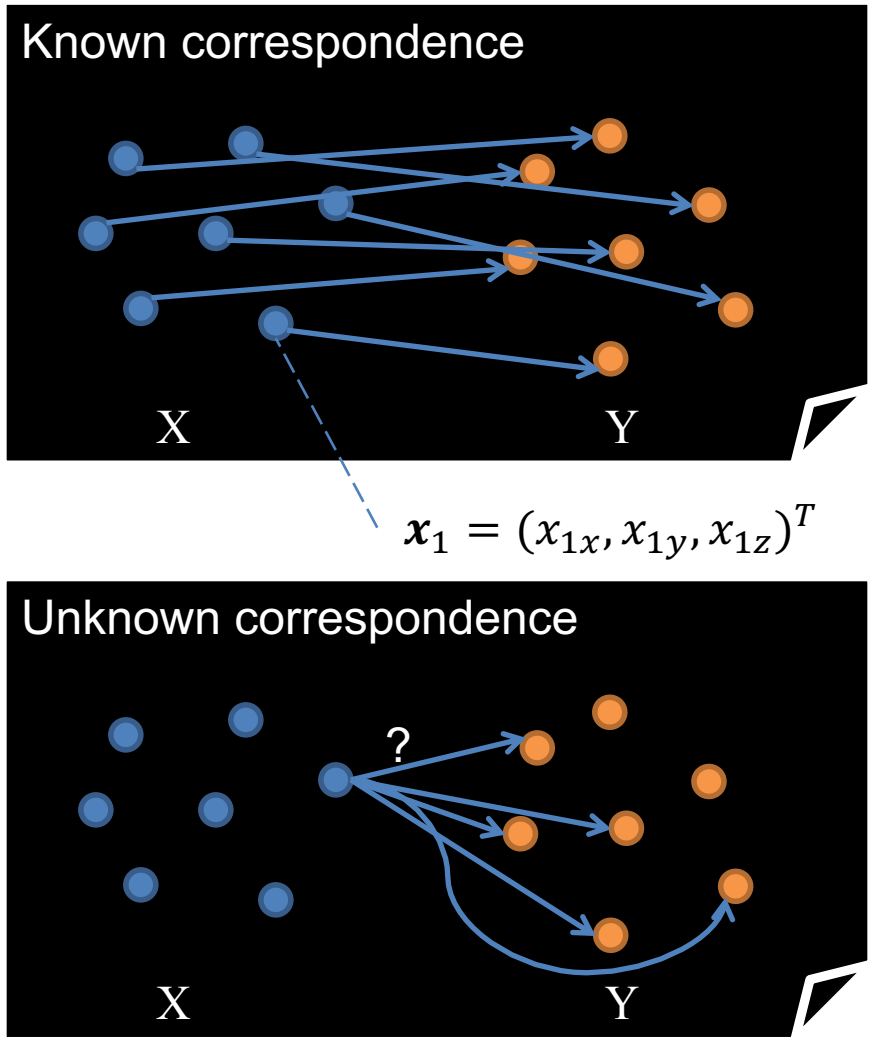
Material by Toru Tamaki, Miho Abe, Bisser Raytchev, Kazufumi Kaneda

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

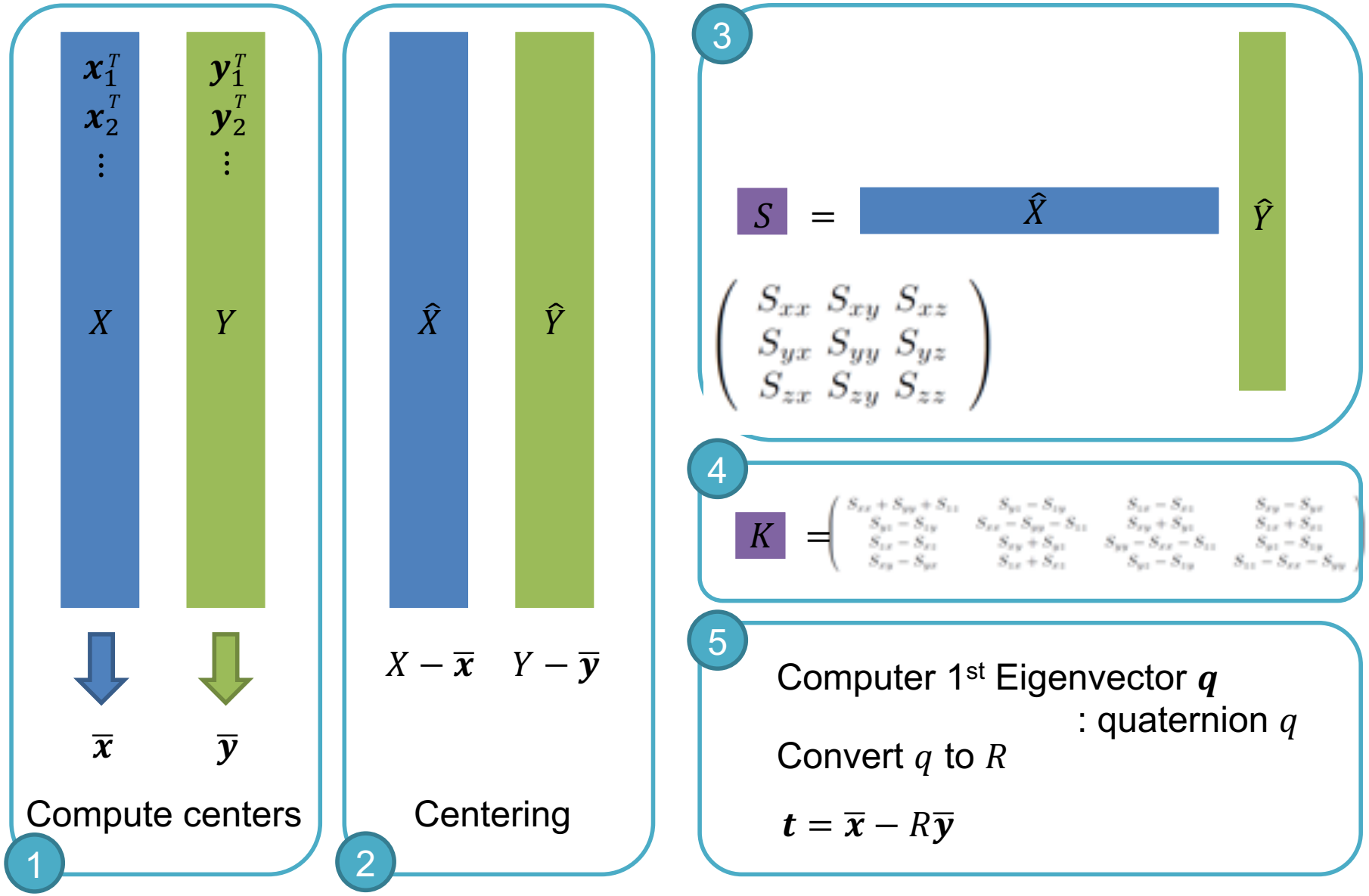


Horn's method: correspondence is known.



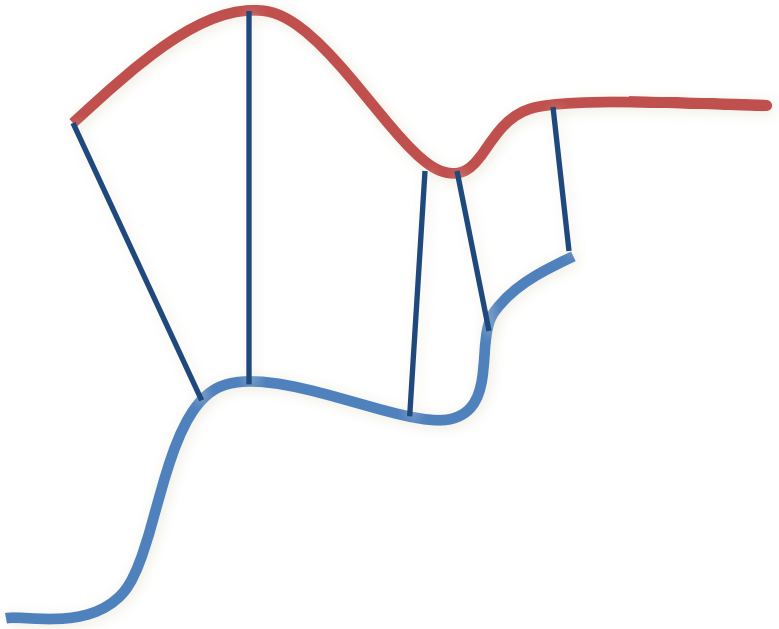


Horn's method: correspondence is known.



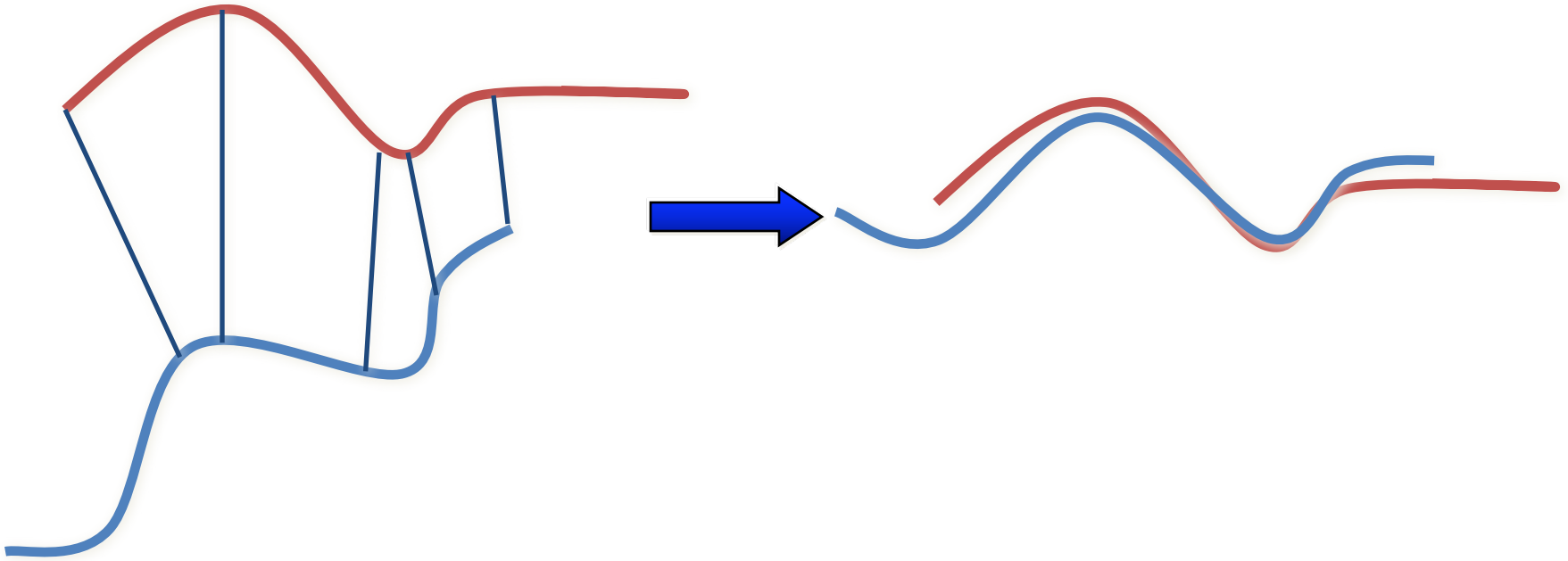
Aligning 3D Data

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



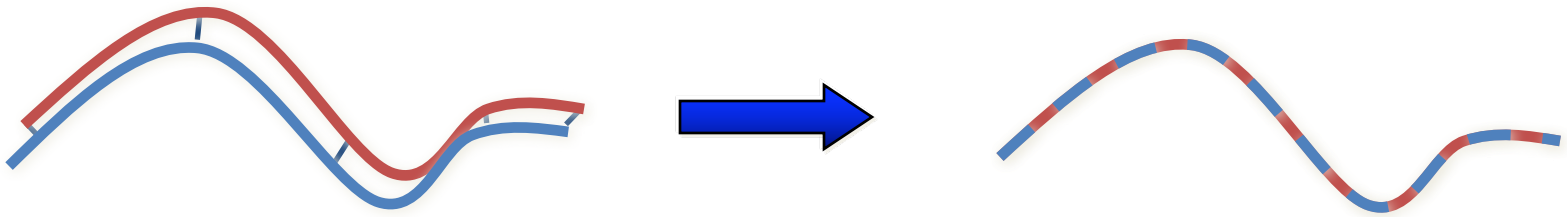
Aligning 3D Data

- 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

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

$$y \in M$$

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

$$Y \subseteq M$$

C – the closest point operator

Y – the set of closest points to S

Finding Matches

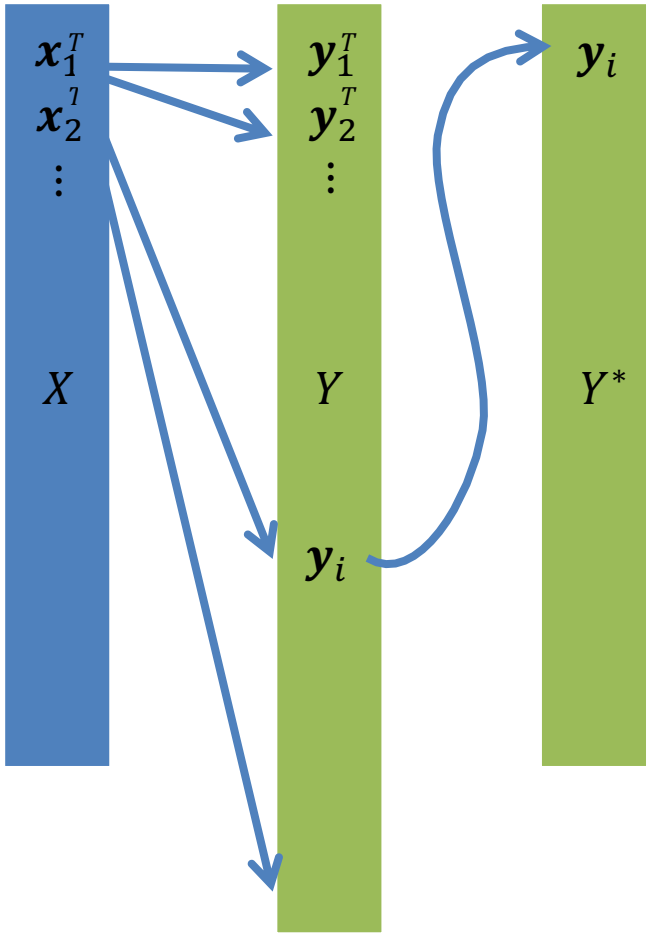
- Finding each match is performed in $O(N_M)$ worst case.
- Given Y we can calculate alignment

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

- S is updated to be :

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

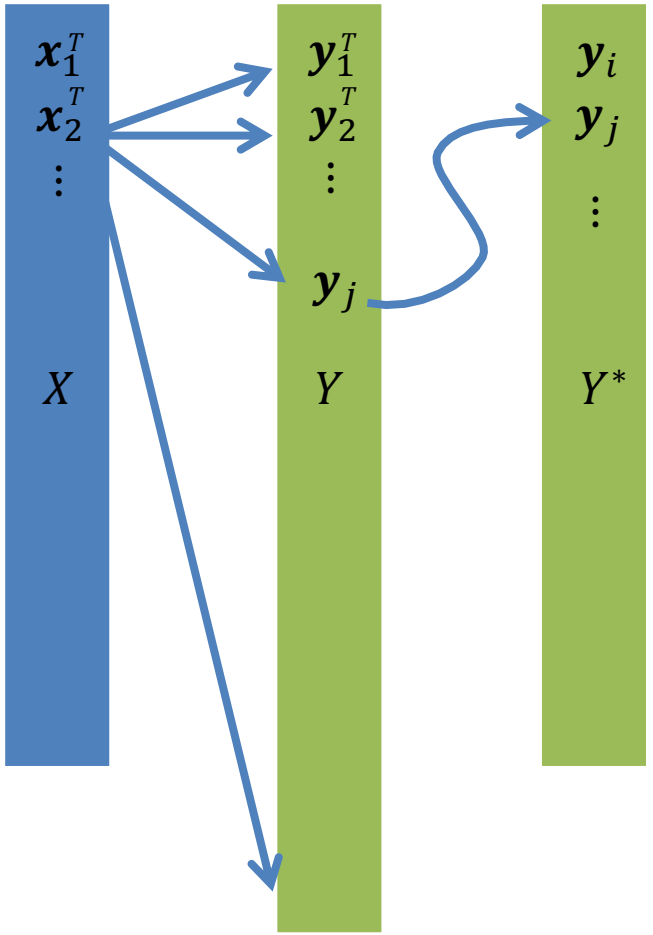
ICP: correspondence is unknown.



Find closest (nearest) point to x_1 in Y

Put the point to Y^*

ICP: correspondence is unknown.



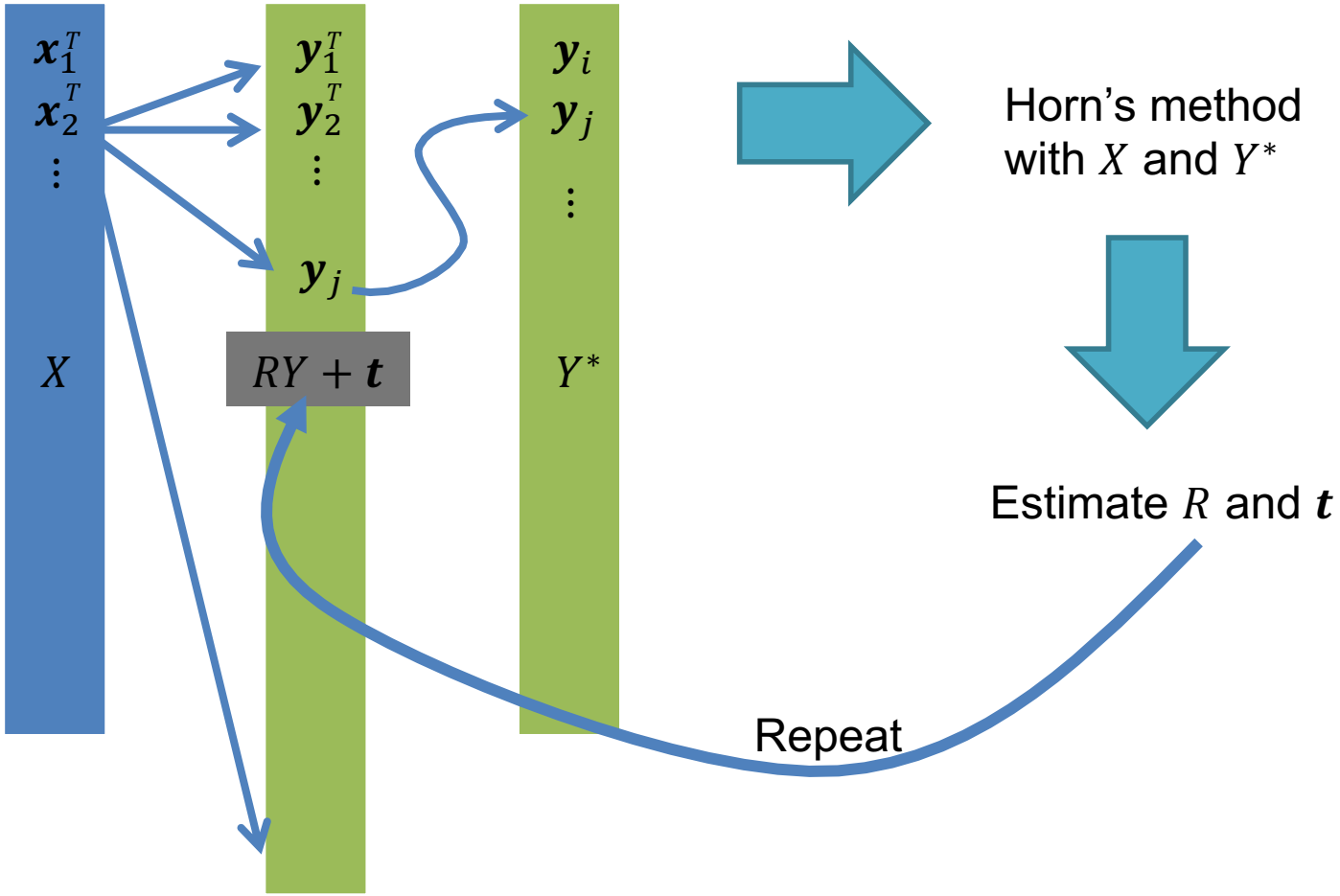
Find closest (nearest) point to $\boldsymbol{x}_1$ in Y

Put the point to Y^*

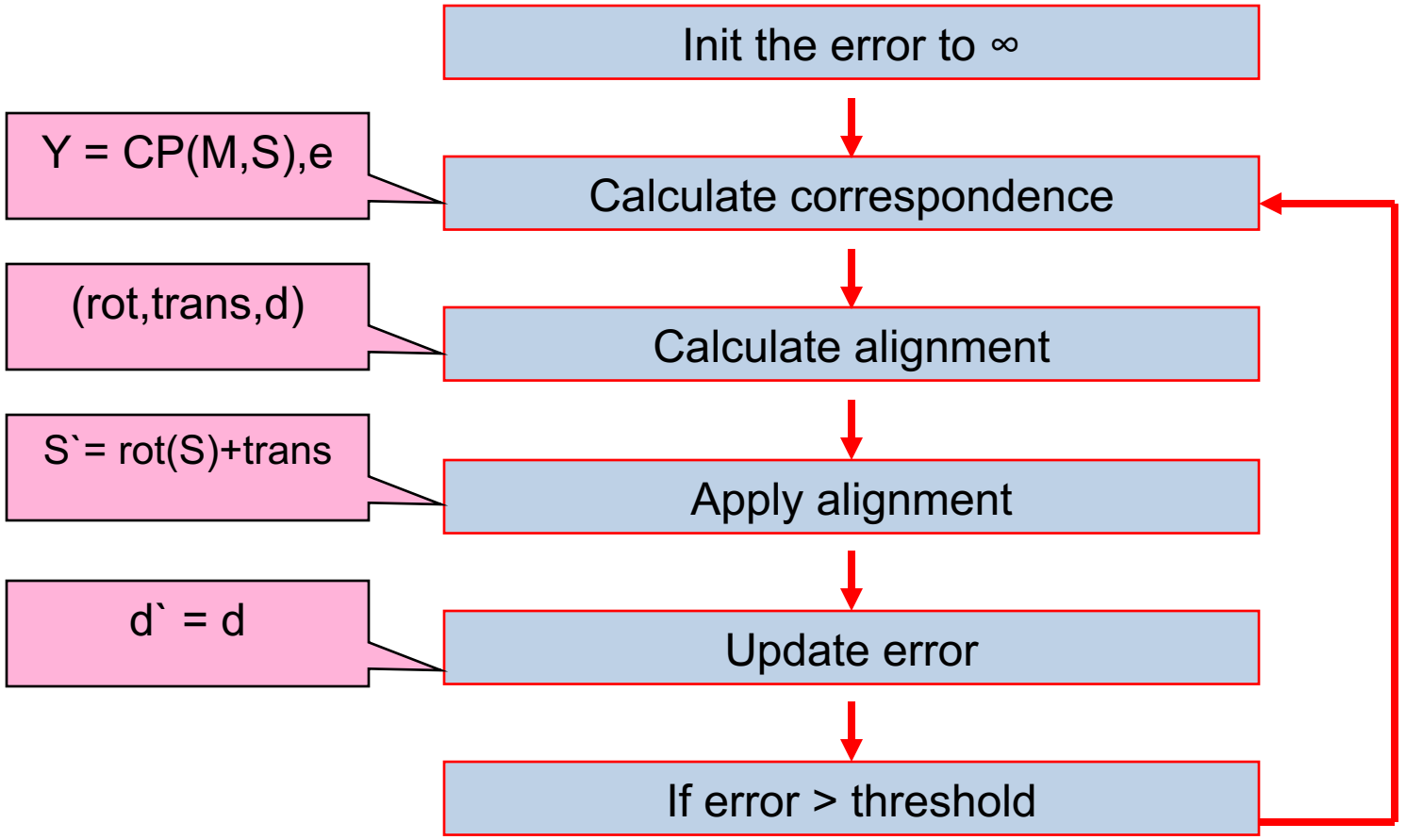
Horn's method with X and Y^*

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: $O(N_S)$

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

Performance of Variants

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

ICP Variants

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

ICP Variants

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

ICP Variants

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

ICP Variants

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

Rejecting Pairs

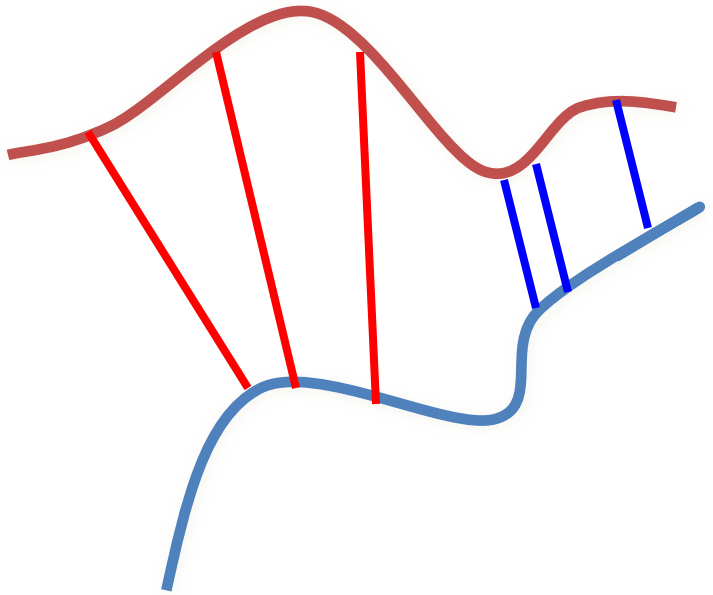
- Corresponding points with point to point distance higher than a given threshold.
- Rejection of worst n% pairs based on some metric.
- Pairs containing points on end vertices.
- Rejection of pairs whose point to point distance is higher than n*σ.
- Rejection of pairs that are not consistent with their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

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

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

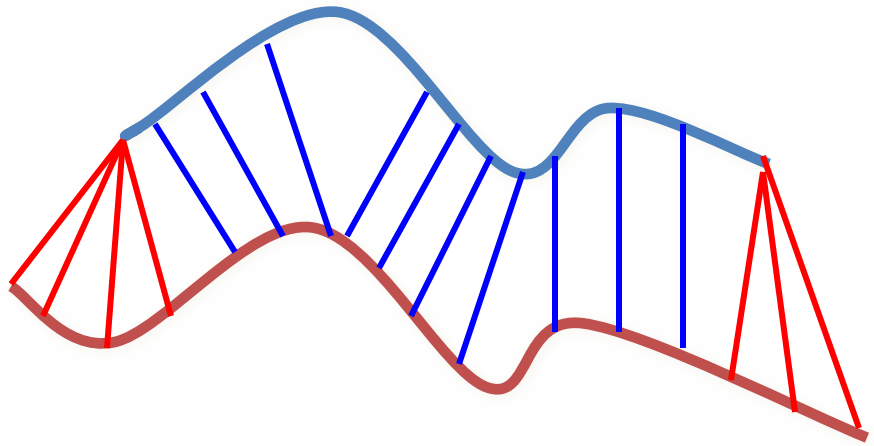
Rejecting Pairs

Distance thresholding



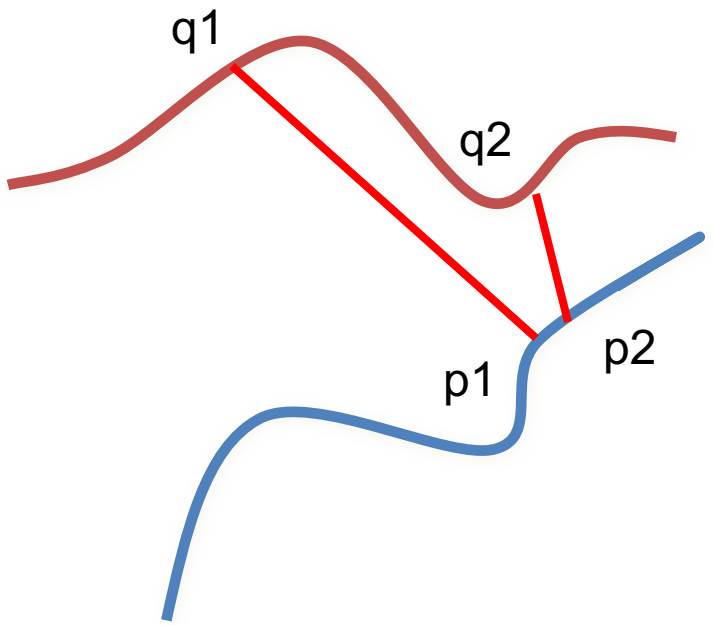
Rejecting Pairs

Points on end vertices



Rejecting Pairs

Inconsistent Pairs

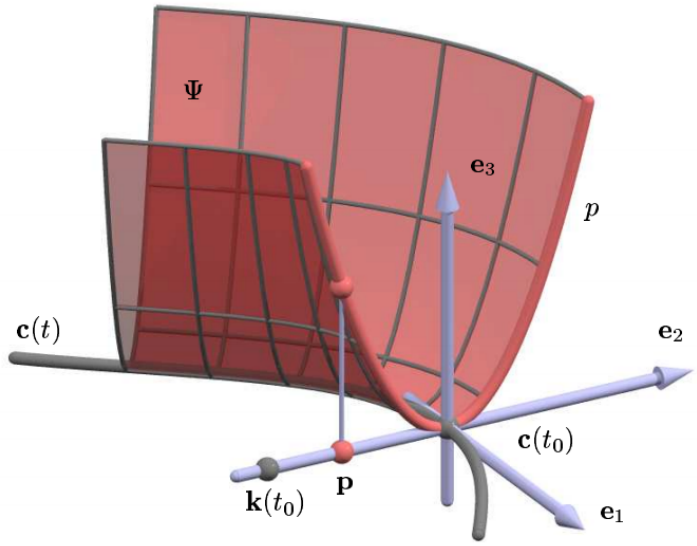


ICP Variants

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

Other registration methods exist

- Robust point matching (soft point correspondences)
- Coherent point drift
- Kernel correlation
- Approximations of the squared distance functions to curves and surfaces

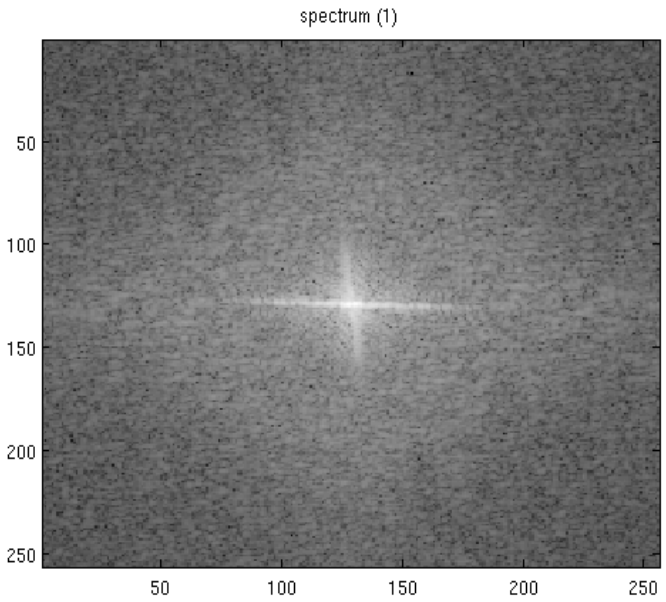


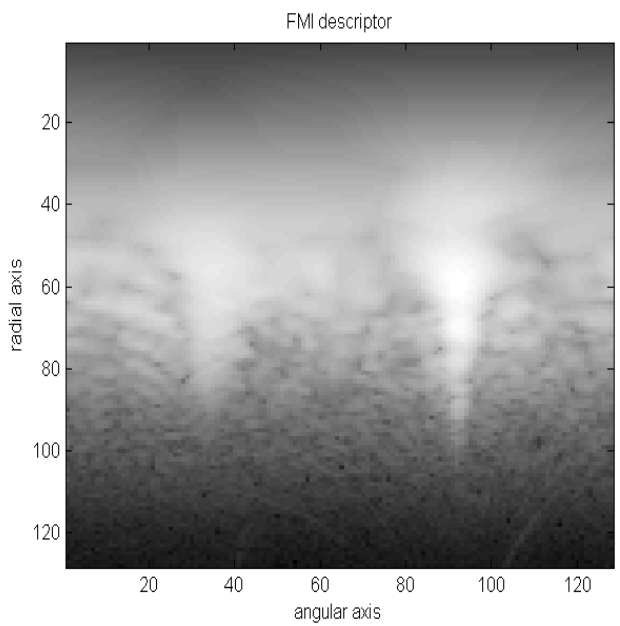
- Feature extracting methods
  - Corners in point clouds
  - Lines
  - Planes

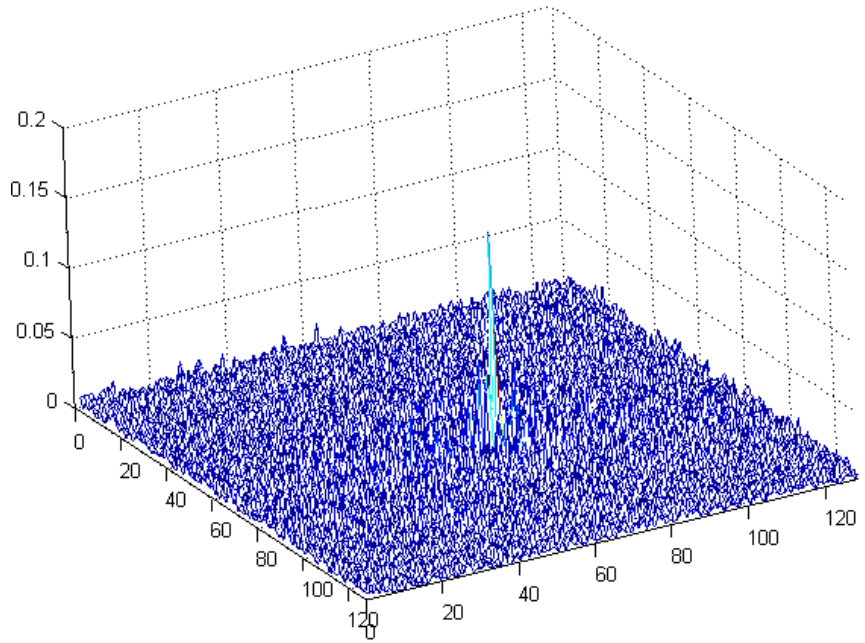
- Spectral methods

Fourier Mellin Transform

- Spectral based registration: detection of scaling, rotation and translation in 2 subsequent frames
- Processing spectrum magnitude decouples translation from affine transformations
  - Detection of signal shift between 2 signals by phase information
  - Resampling to polar coordinates → Rotation turns into signal shift !
  - Resampling the radial axis from linear to logarithmic presentation
    → Scaling turns into signal shift !
  - Calculate a Phase Only Match Filter (POMF) on the resampled magnitude spectra

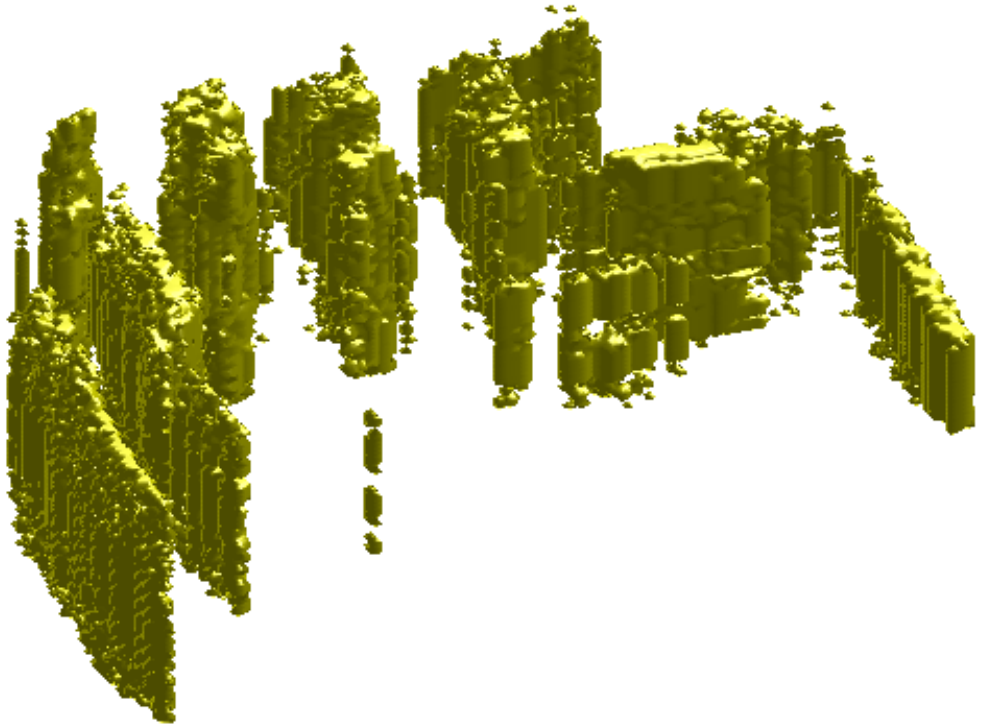






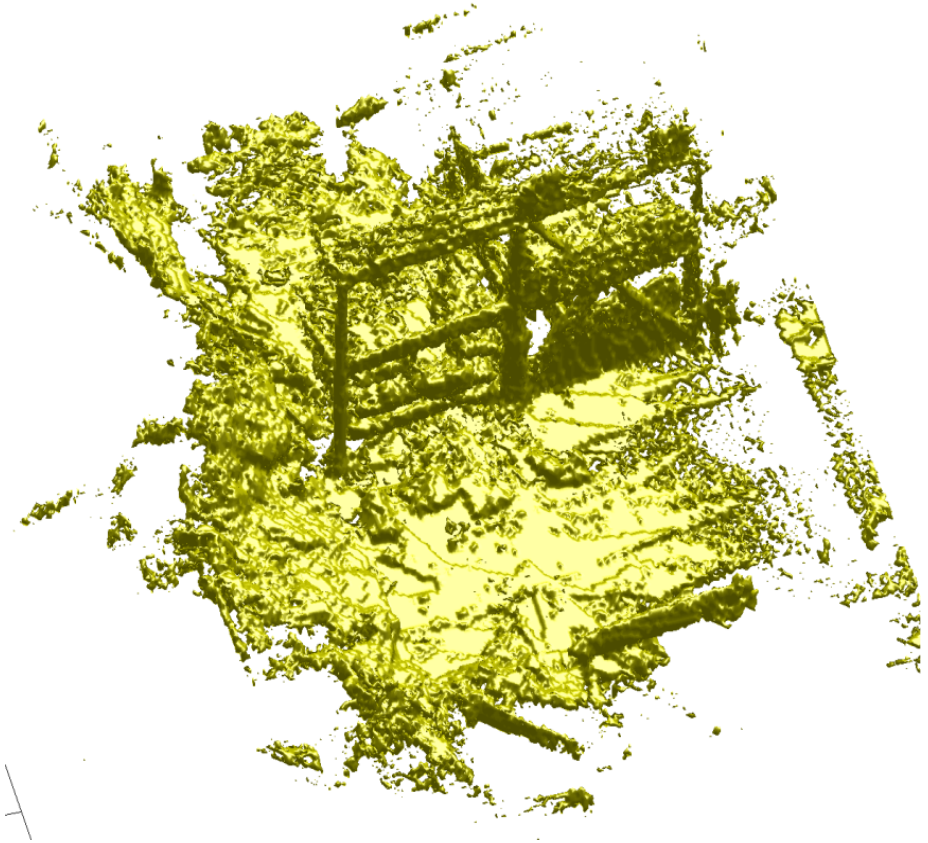
3D Scan matching using Spectral Method FMI

Flood Gate (sonar)



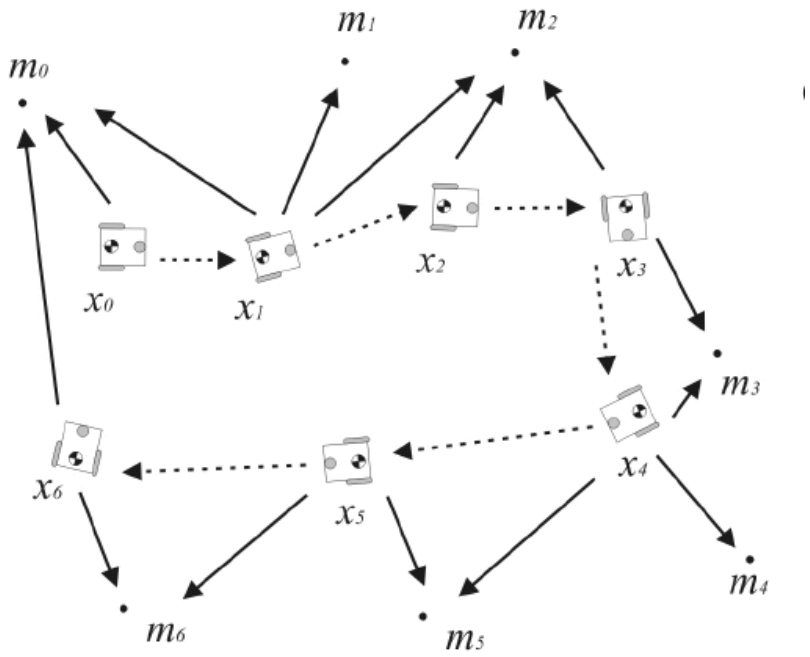
Crashed car park

Disaster City (3D LRF)



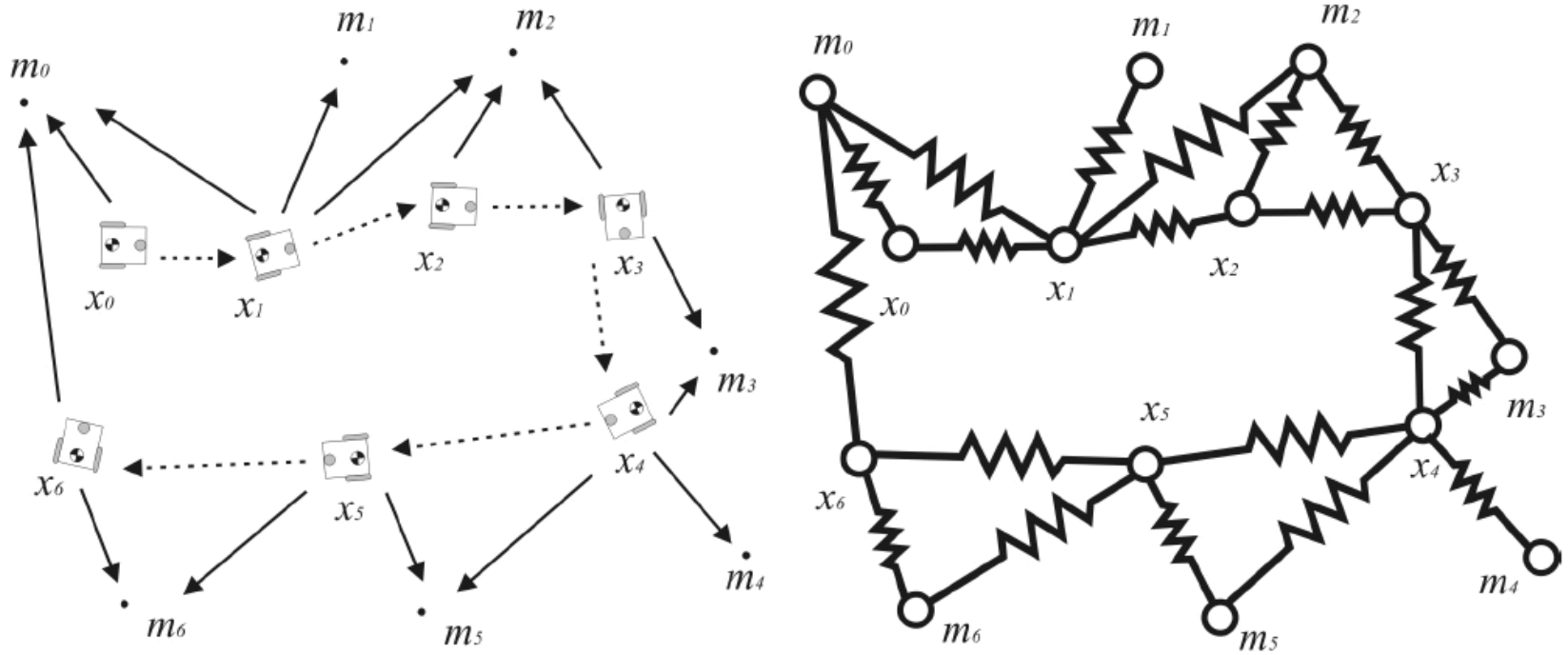
Graph-Based SLAM (1/3)

- SLAM problem can be interpreted as a sparse graph of nodes and constraints between nodes.
- The nodes of the graph are the robot locations and the features in the map.
- Constraints: relative position between consecutive robot poses , (given by the odometry input ***u***) and the relative position between the robot locations and the features observed from those locations.



Graph-Based SLAM (2/3)

- Constraints are not rigid but soft constraints!
- Relaxation: compute the solution to the full SLAM problem =>
  - Compute best estimate of the robot path and the environment map.
  - Graph-based SLAM represents robot locations and features as the nodes of an elastic net. The SLAM solution can then be found by computing the state of minimal energy of this net

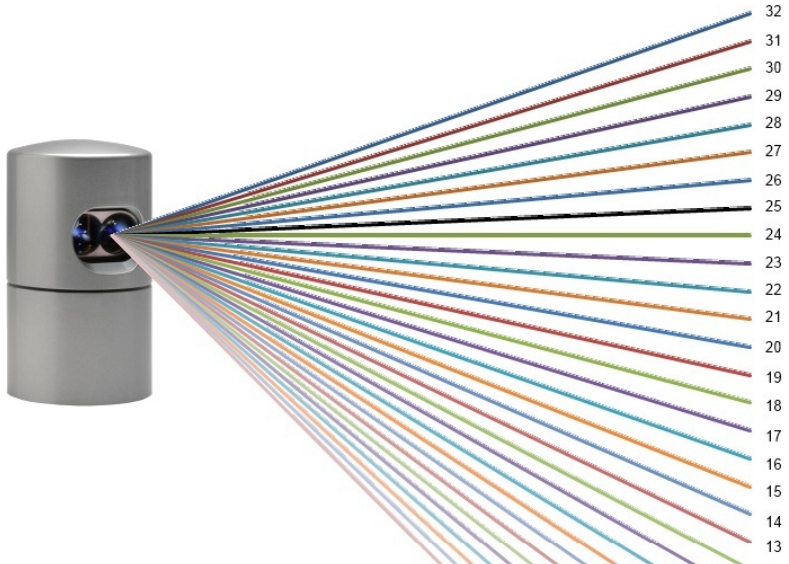


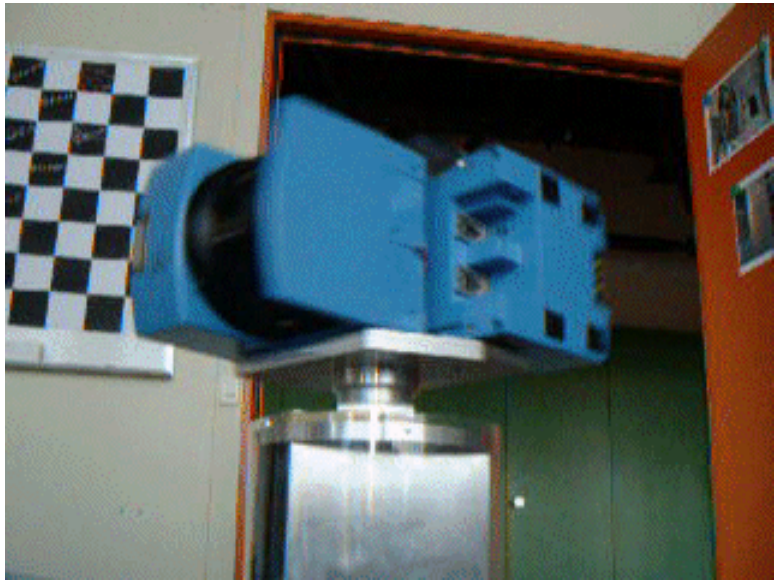
Graph-Based SLAM (3/3)

- Significant advantage of graph-based SLAM techniques over EKF SLAM:
  - EKF SLAM: computation and memory for to update and store the covariance matrix is quadratic with the number of features.
  - Graph-based SLAM: update time of the graph is constant and the required memory is linear in the number of features.
- However, the final graph optimization can become computationally costly if the robot path is long.

3D Mapping

- Using 2.5D Sensors:
  - Create depth images – suffer from occlusion
  - Actuated LRF
  - 3D LRF (Velodyne, Riegel, Faro)

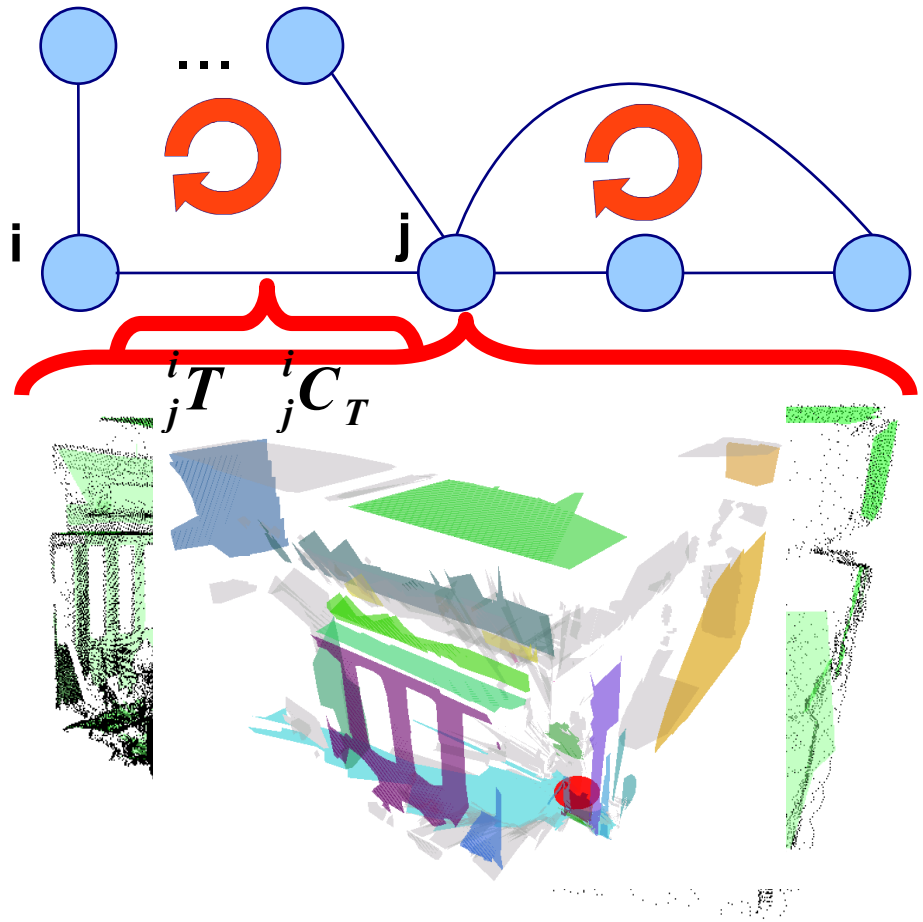




Davide Scaramuzza and Roland Siegwart, ETH Zurich

Jacobs 3D Mapping – Plane Mapping

- Pose Graph
- 3D Range Sensing
- Plane Extraction
- Planar Scan Matching
- Relax Loop-Closing Errors



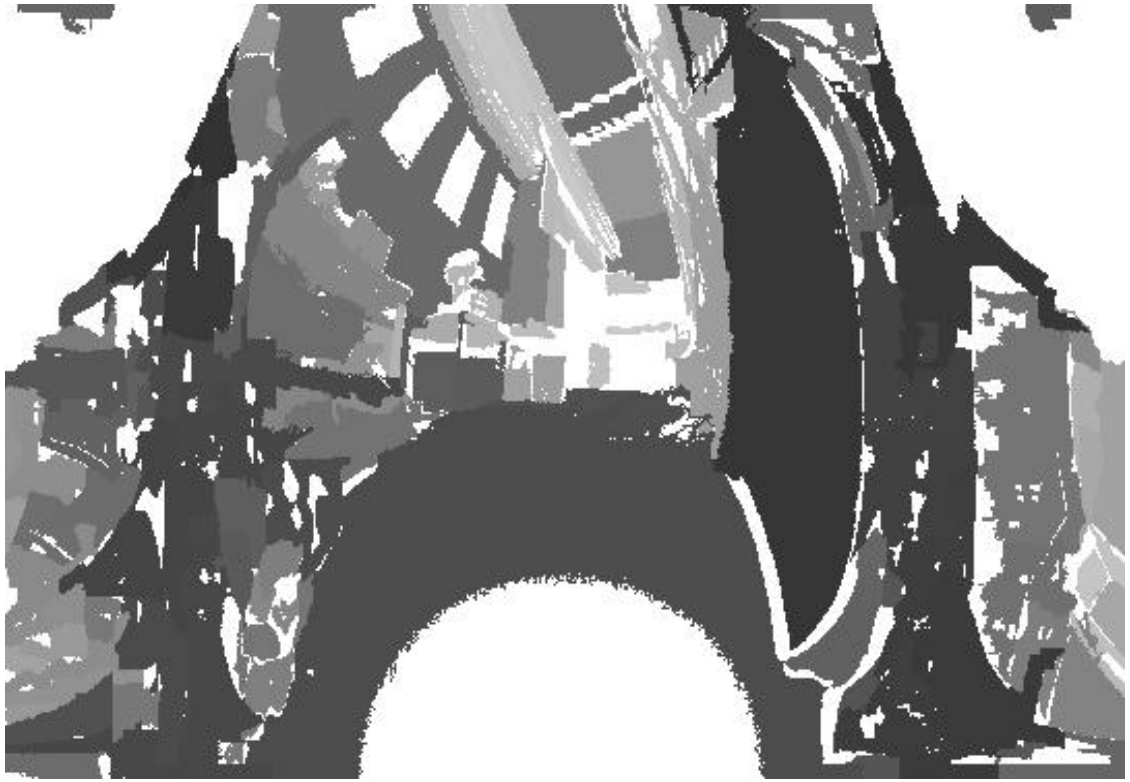
Plane Extraction from 3D Point Clouds

- **Plane Fitting**
  - Assumes 3D sensor has radial Gaussian noise dependent on range
  - Uses Approximate Least Squares solution to find the best fit.
  - Estimates covariance matrix of the plane parameters

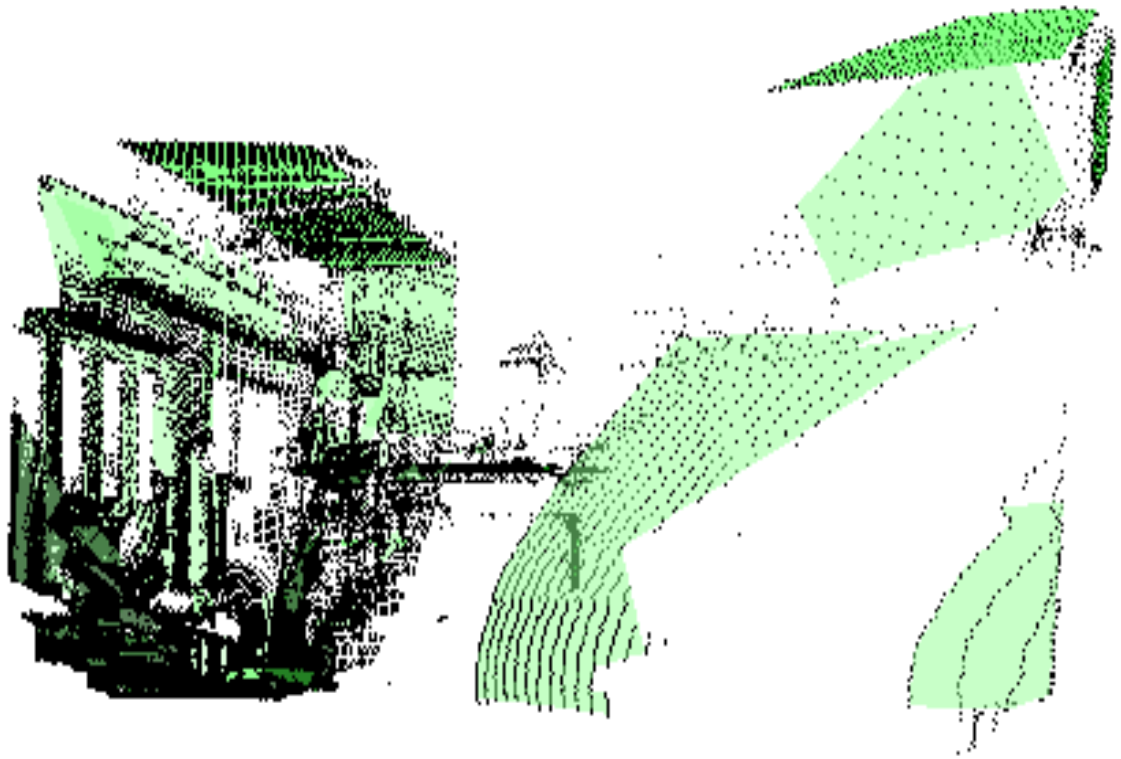
- **Range Image Segmentation**
  - Is based on region growing algorithm
  - Uses incremental formulas, therefore is fast
  - Has linear computational complexity

Given a range image, returns a polygonal model i.e. a set of planar features and boundaries.

An Example



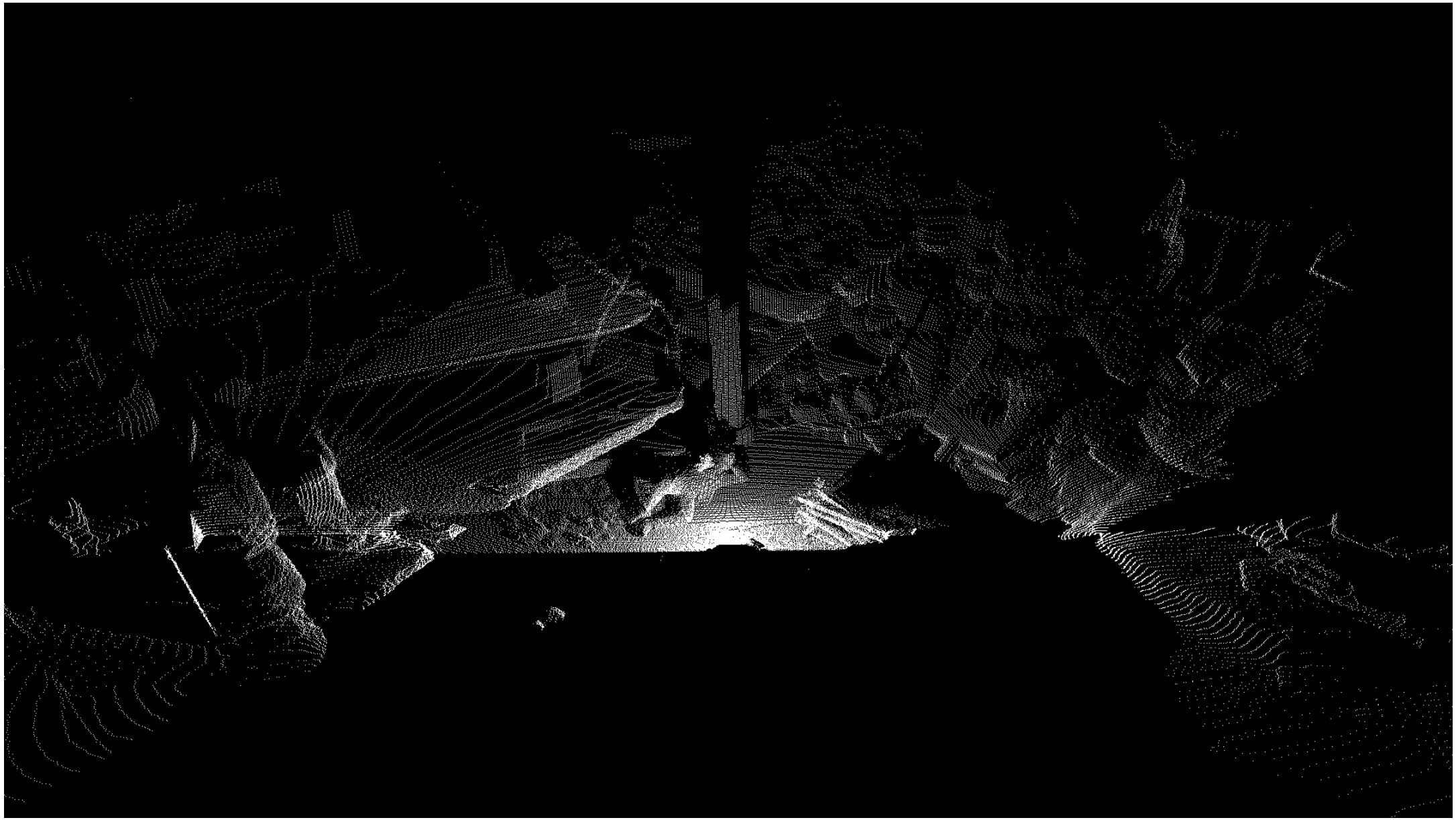
Segmentation Image



Polygonal 3D Model

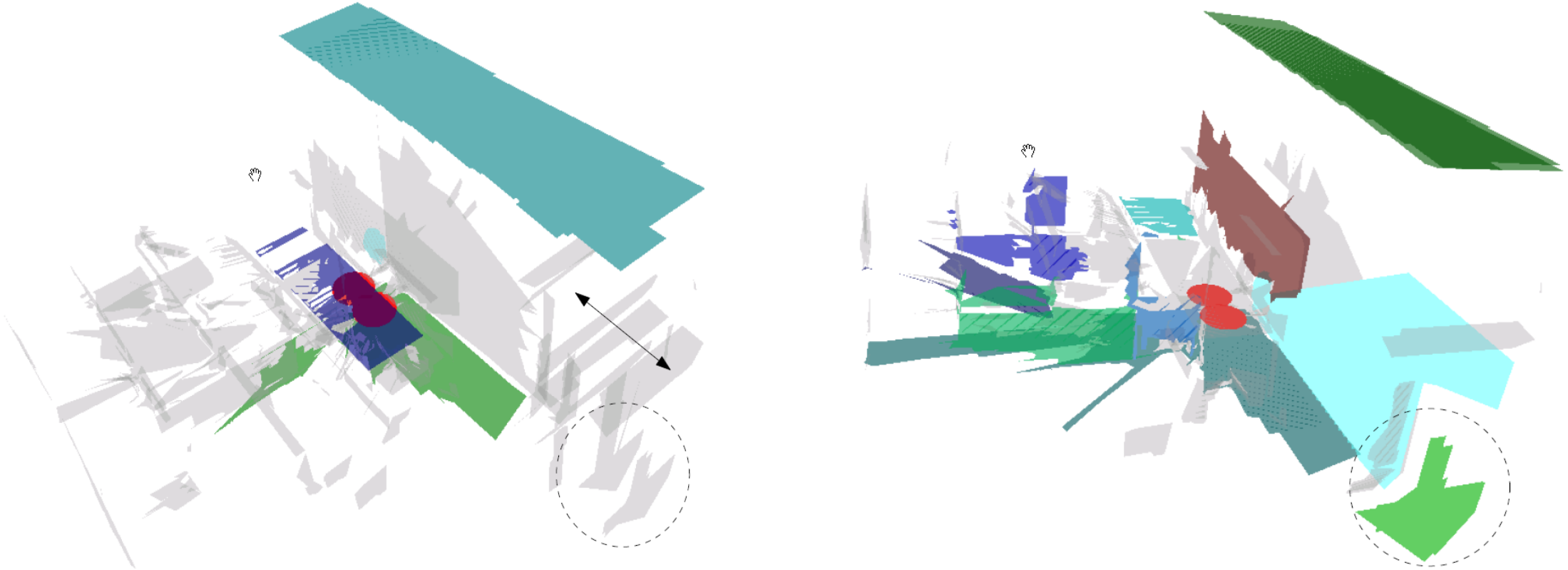
Segments range image consisting of ~$2 \cdot 10^5$ pixels in ~3s. On 1.6 Ghz machine.

One Point Cloud with human



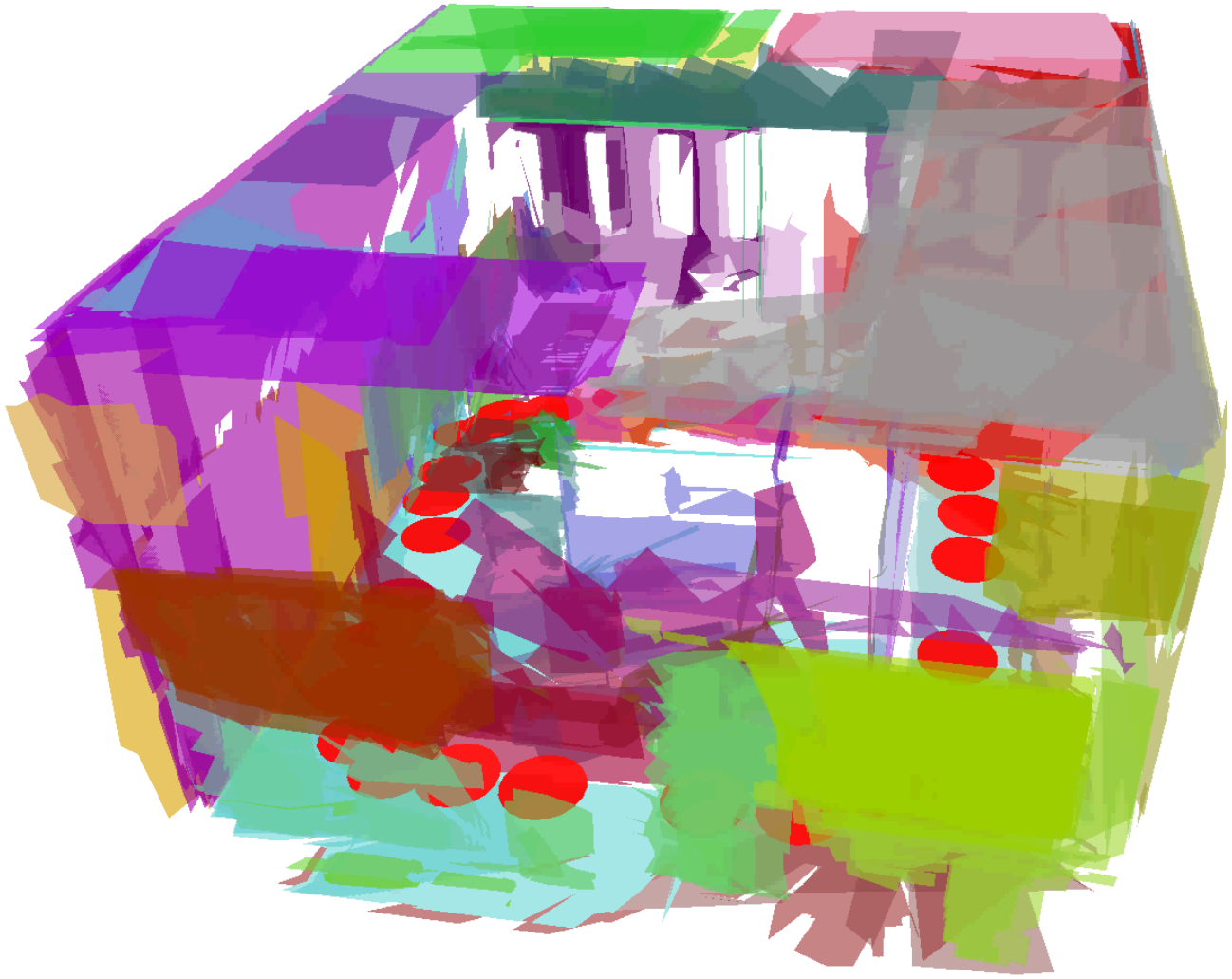
One Plane Cloud with human

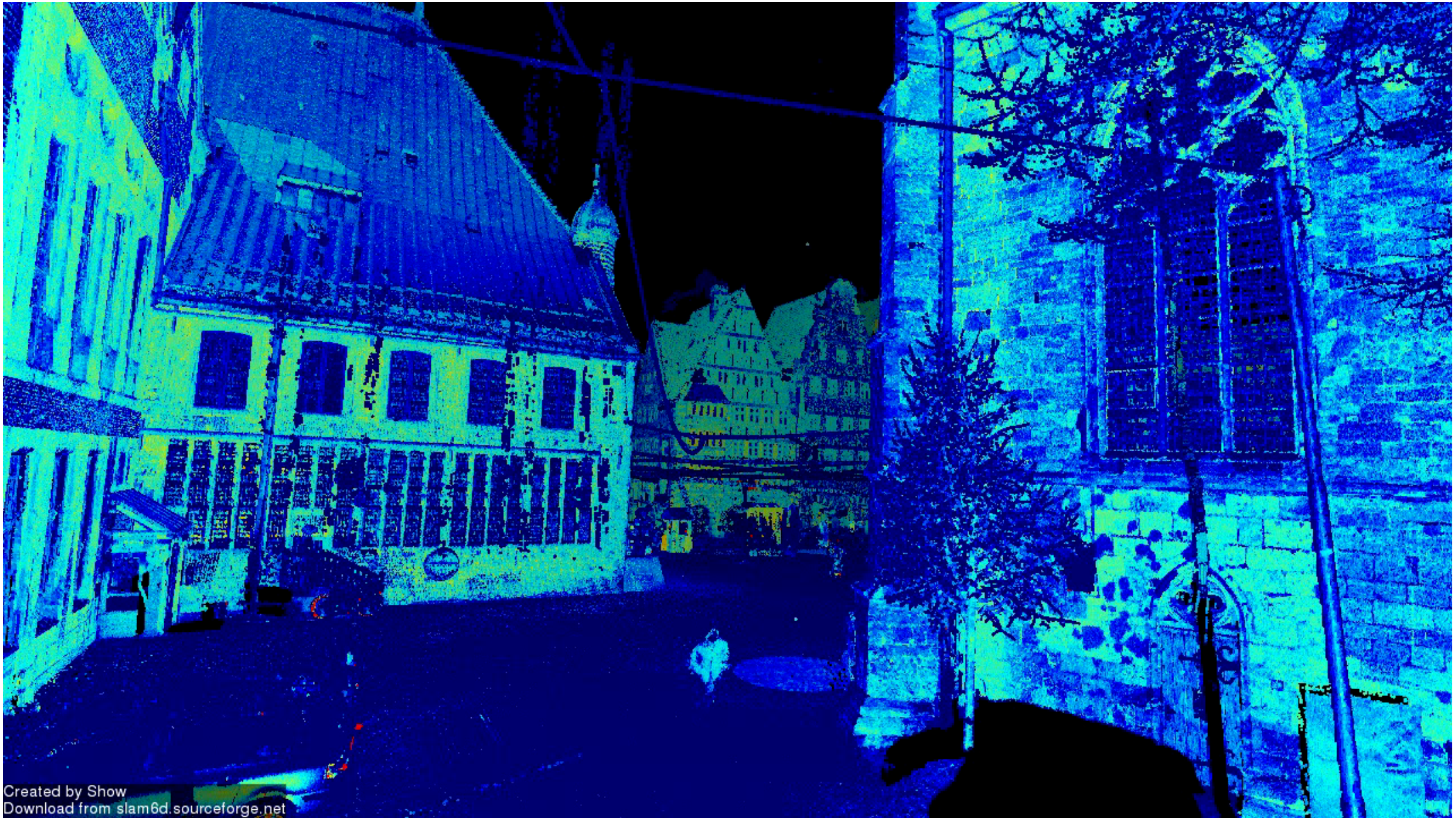
Relaxation of Errors (Translation)



- **Only translation errors are relaxed**
  - Good rotation estimates from the plane matching
  - Non-linear optimization can be exchanged with linear if rotation is assumed to be known precisely.
  - This leads to a fast relaxation method

Experiment Lab Run: 29 3D point-clouds; size of each: 541 x 361 = 195,301




Created by Show
Download from slam6d.sourceforge.net

Aerial Map (Mosaic)

- Rubble pile and train
- 435 frames
- Real time generation of map