上海科技大学
ShanghaiTech University

CS283: Robotics Spring 2024: SLAM I

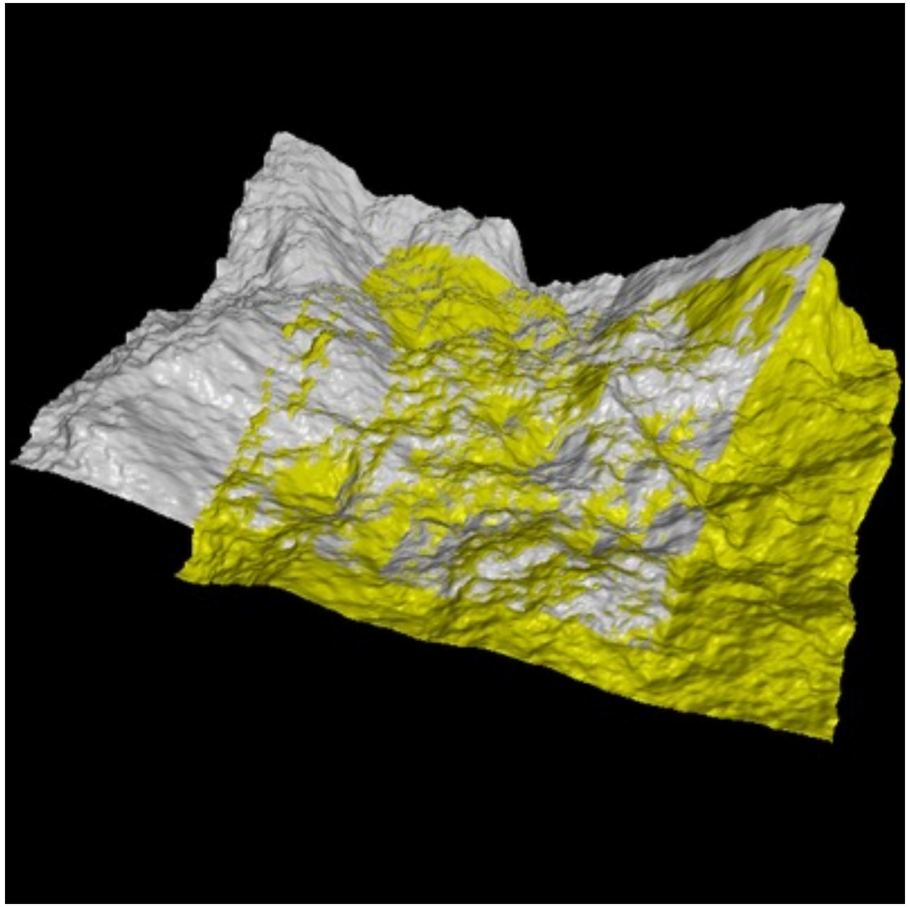
Sören Schwertfeger / 师泽仁

ShanghaiTech University

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)

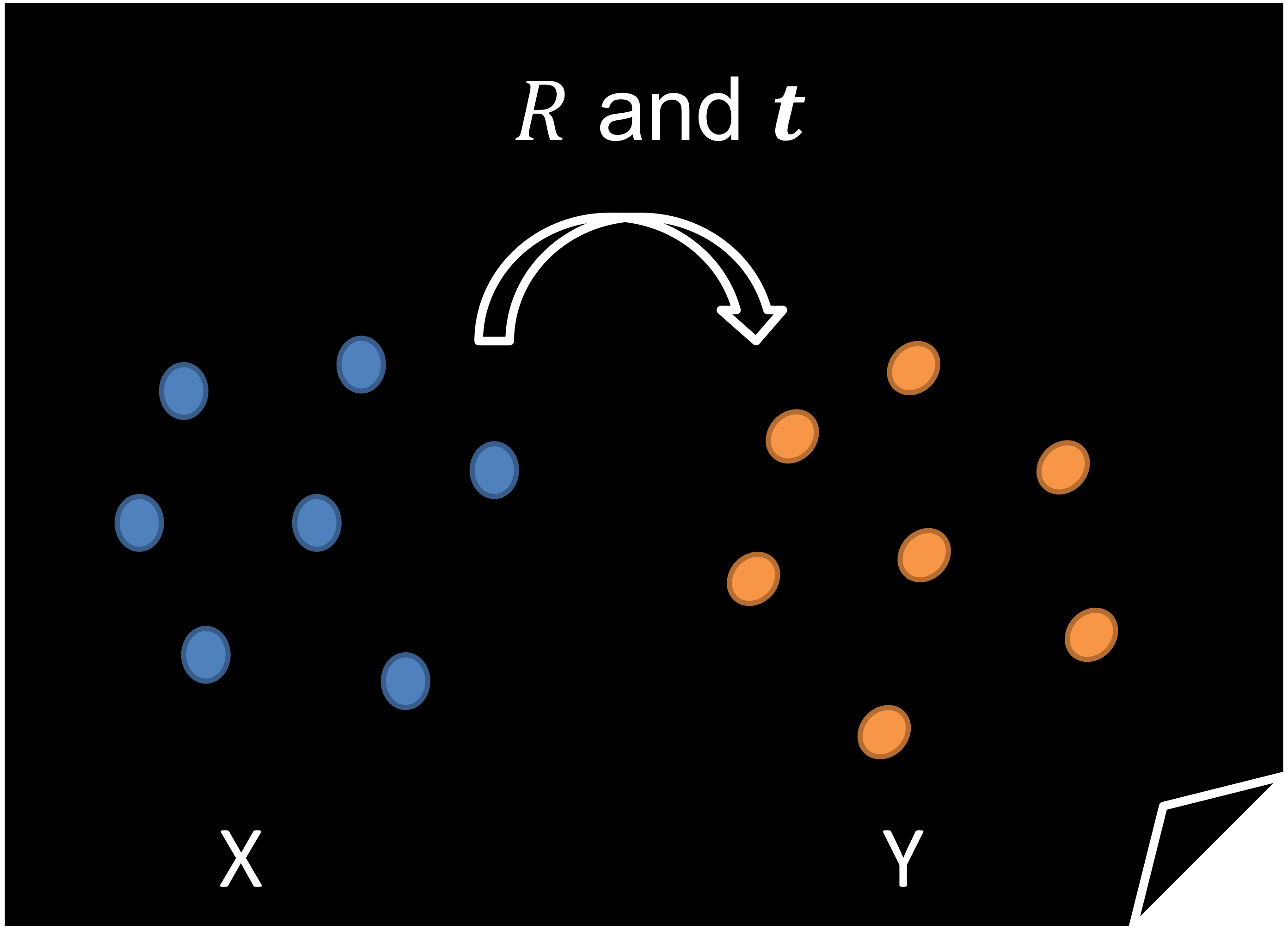


Material derived from Ronen Gvili : *www.cs.tau.ac.il/~dcor/Graphics/adv-slides/**ICP**.ppt*

Horn's method

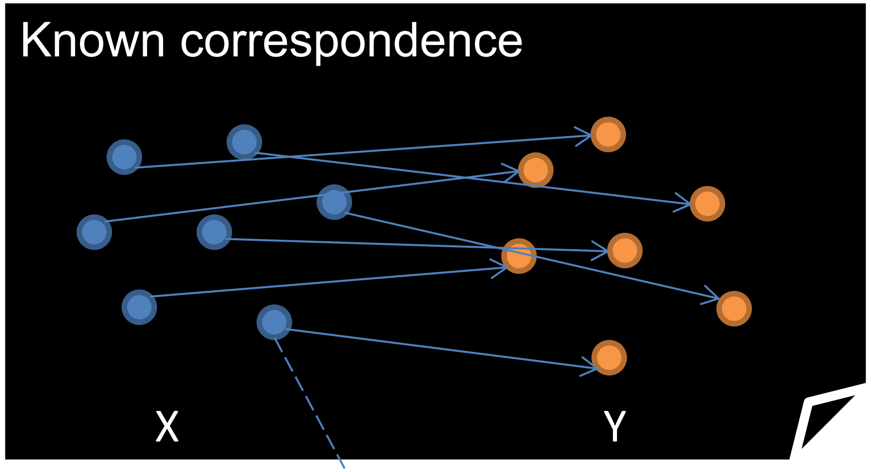
Material by Toru Tamaki, Miho Abe, Bisser Raytchev, Kazufumi Kaneda

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

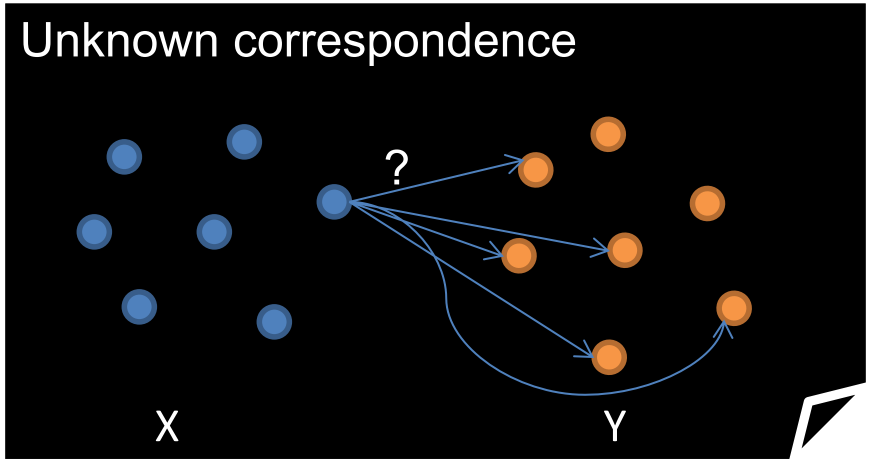


Horn's method: correspondence is known.

$$X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \end{bmatrix} \qquad Y = \begin{bmatrix} \boldsymbol{y}_1^T \\ \boldsymbol{y}_2^T \\ \vdots \end{bmatrix}$$



$$\boldsymbol{x}_1 = (x_{1x}, x_{1y}, x_{1z})^T$$



Horn’s method: correspondence is known.

1

$$X = \begin{pmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \end{pmatrix} \qquad Y = \begin{pmatrix} \boldsymbol{y}_1^T \\ \boldsymbol{y}_2^T \\ \vdots \end{pmatrix}$$

$\bar{\boldsymbol{x}}$ $\bar{\boldsymbol{y}}$

Compute centers

2

$\hat{X}$ $\hat{Y}$

$X - \bar{\boldsymbol{x}}$ $Y - \bar{\boldsymbol{y}}$

Centering

3

$$S = \hat{X} \hat{Y}$$

$$\begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}$$

4

$$K = \begin{pmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & S_{yy}-S_{xx}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}-S_{zy} & S_{zz}-S_{xx}-S_{yy} \end{pmatrix}$$

5

Compute 1st Eigenvector $\boldsymbol{q}$

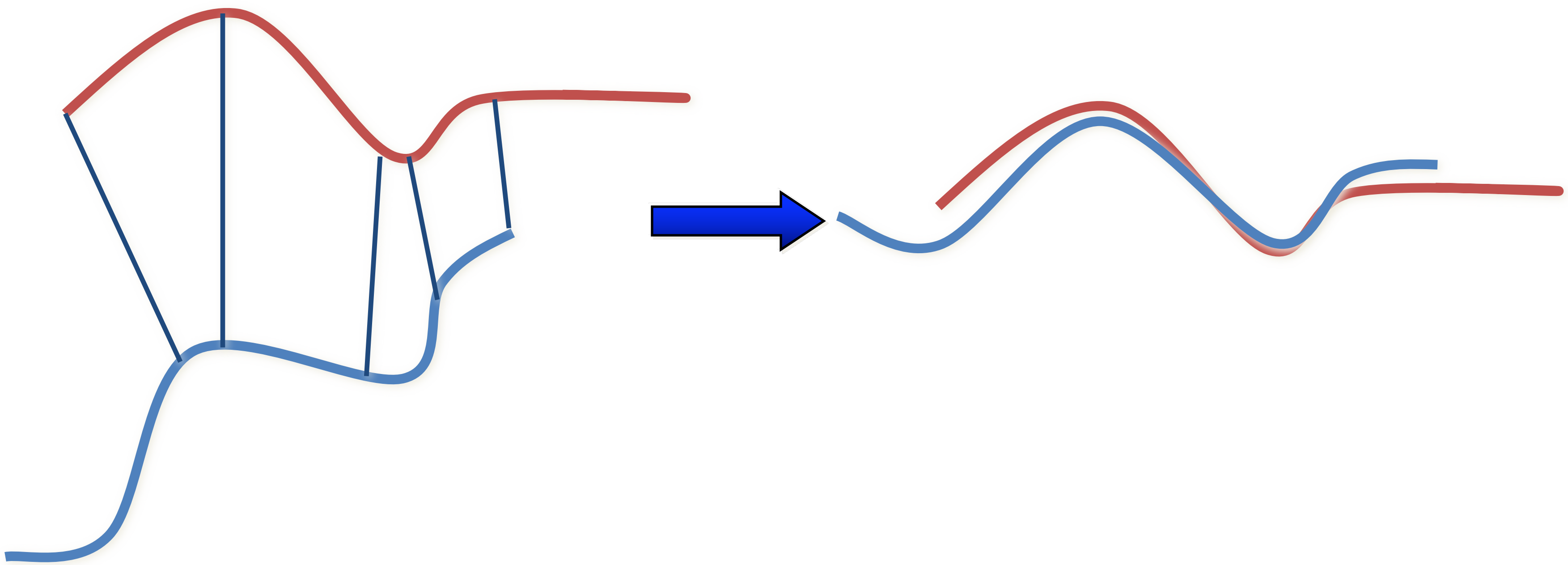
: quaternion q

Convert q to R

$$\boldsymbol{t} = \bar{\boldsymbol{x}} - R\bar{\boldsymbol{y}}$$

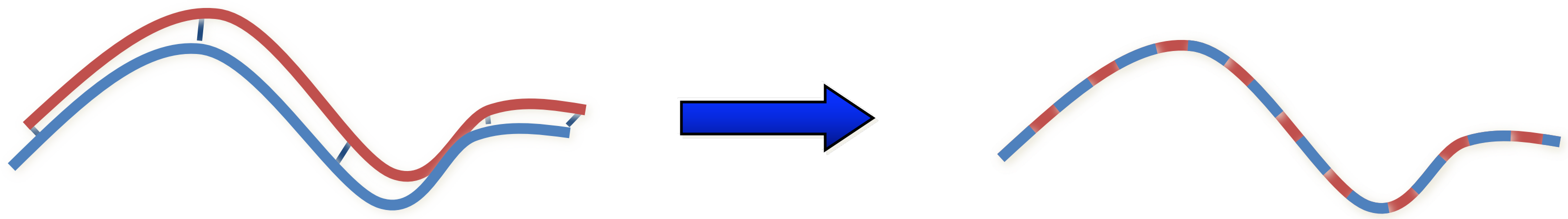
Aligning 3D Data

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

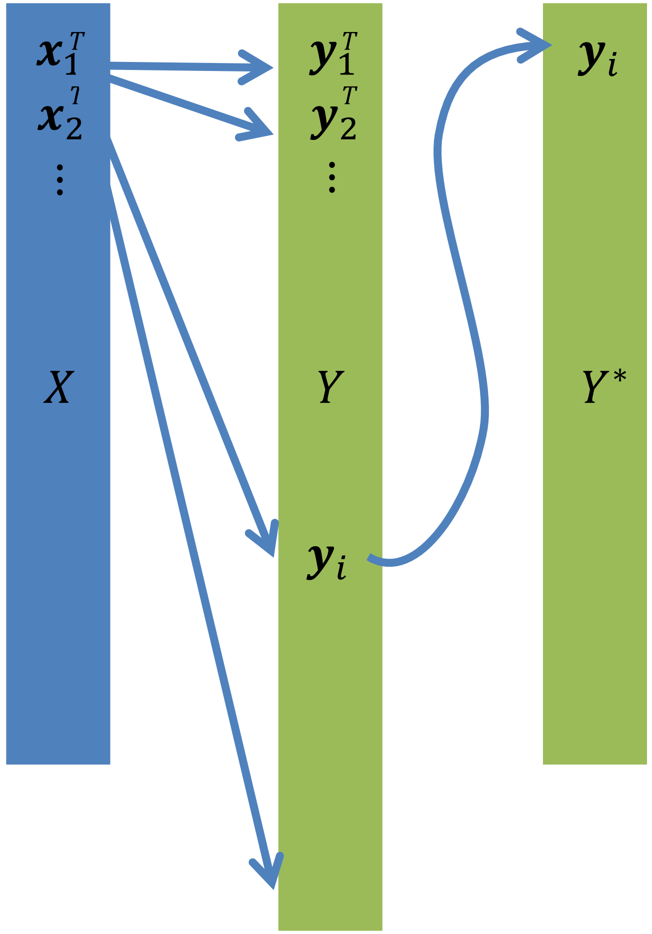


Aligning 3D Data

- Converges if starting position “close enough“



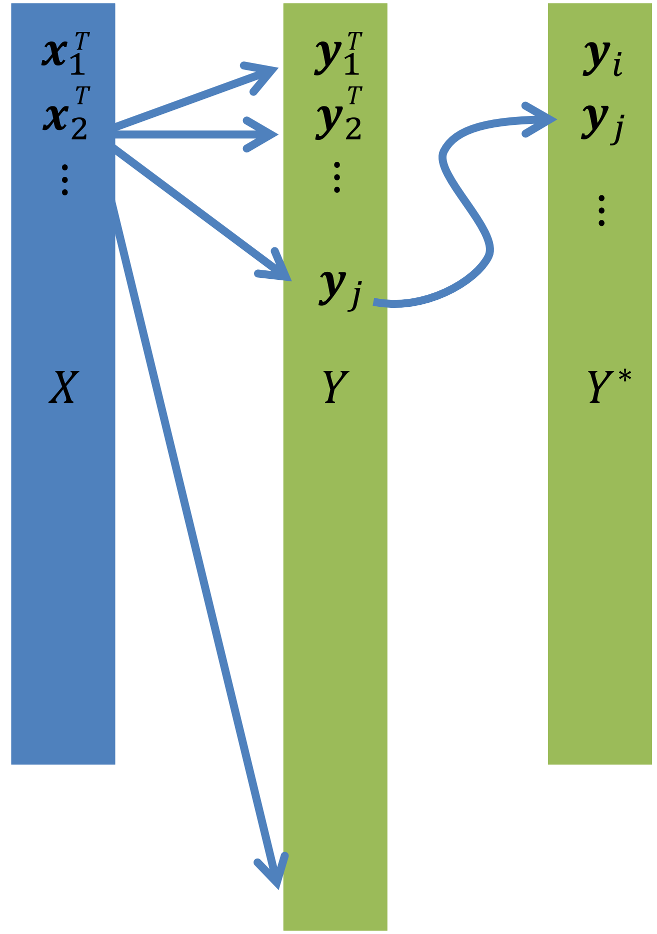
ICP: correspondence is unknown.



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

Put the point to Y^*

ICP: correspondence is unknown.



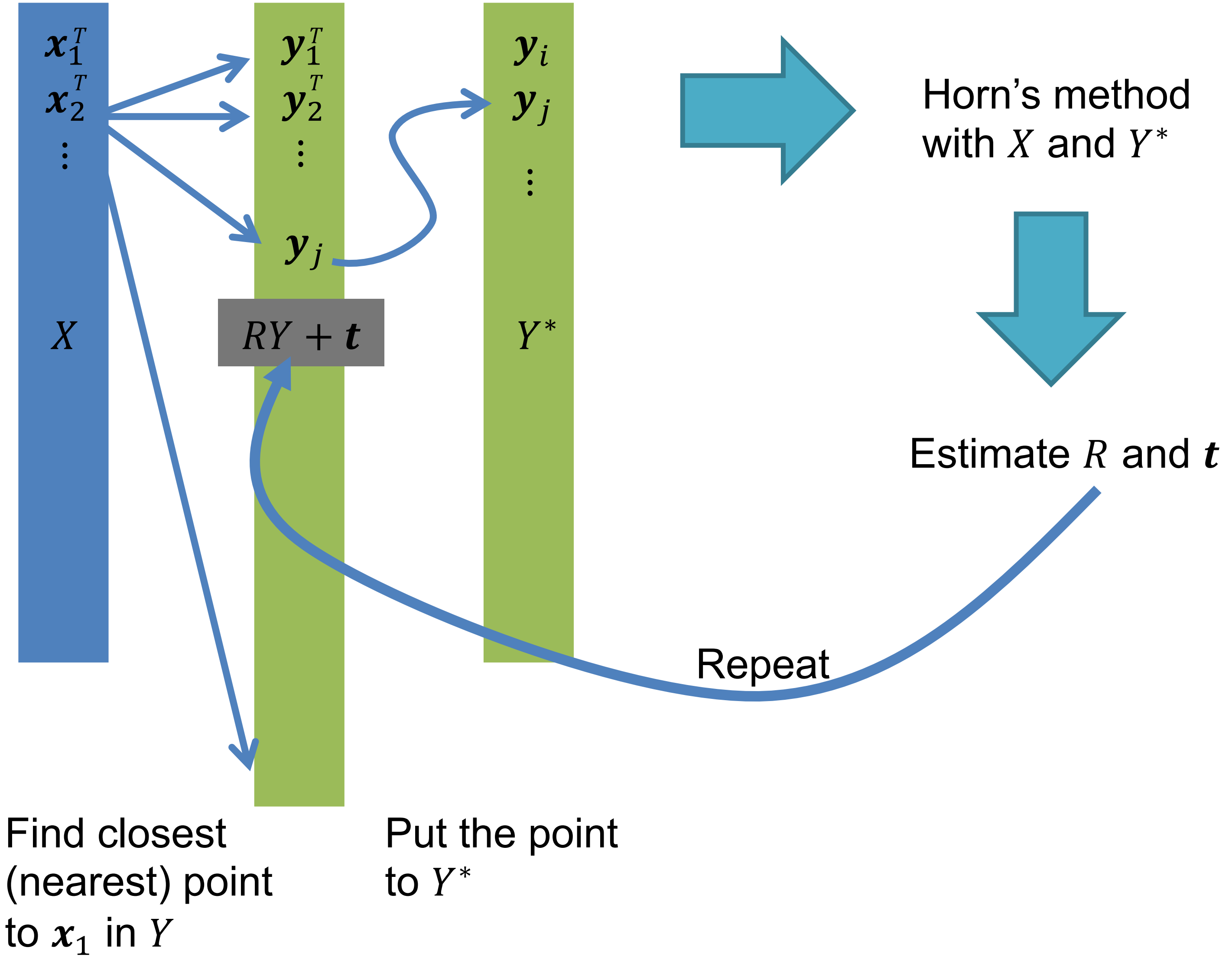
Find closest (nearest) point to $\boldsymbol{x}_1$ in Y

Put the point to Y^*

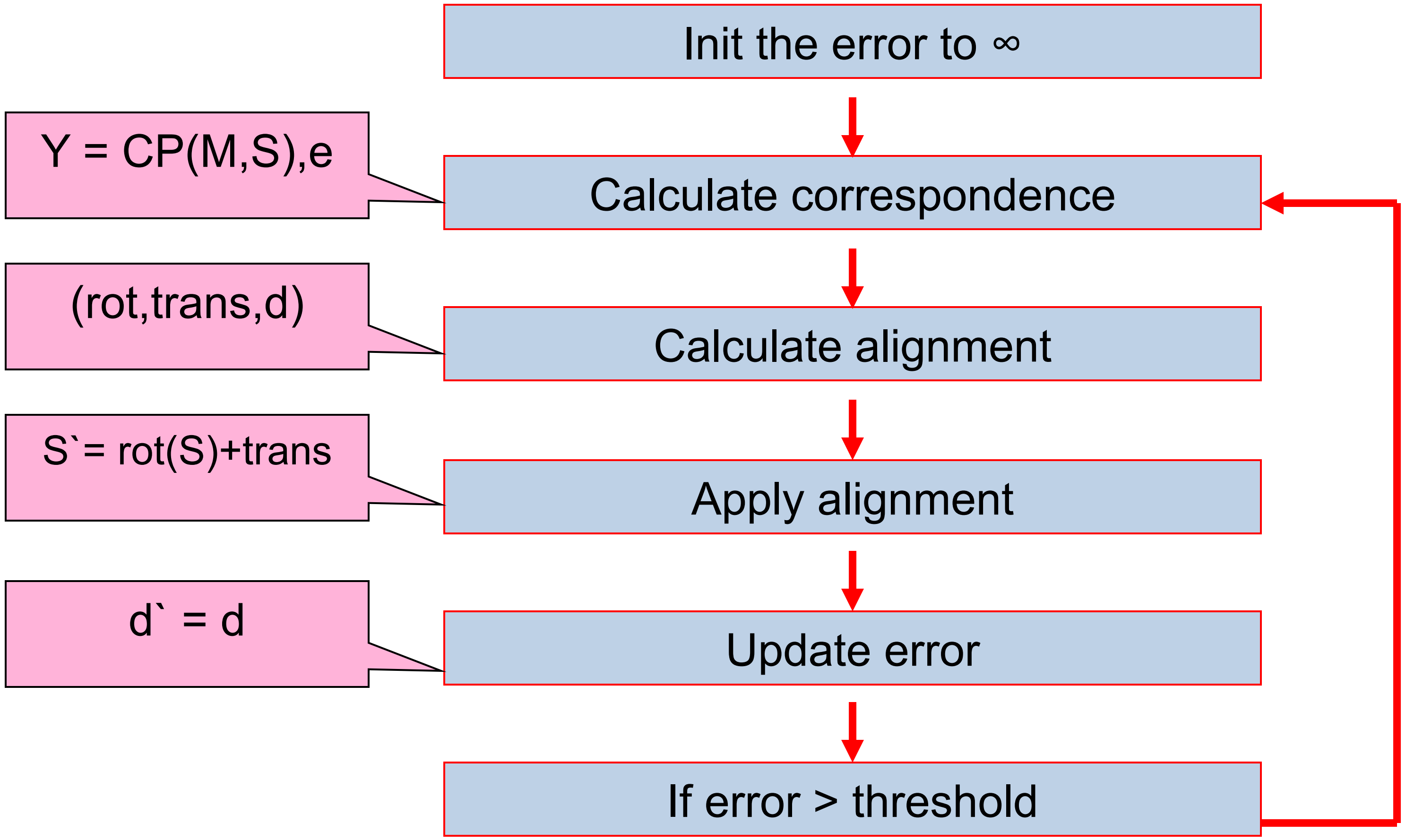
Horn's method with X and Y^*

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

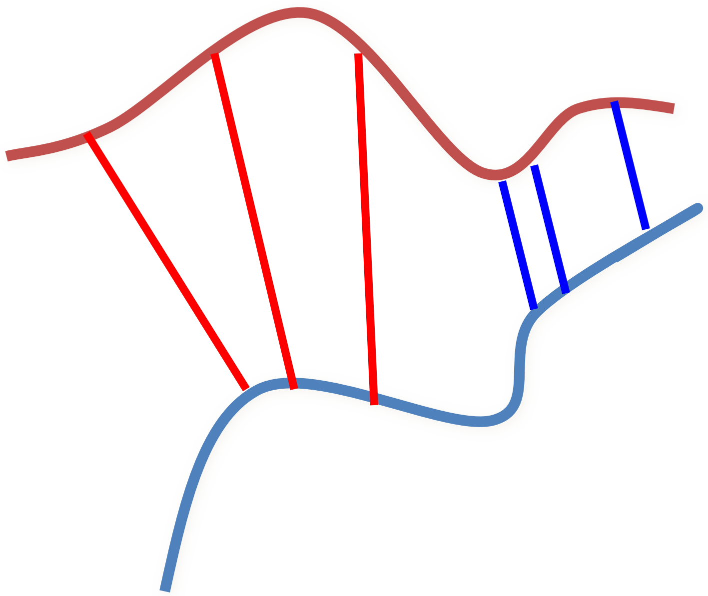
- Corresponding points with point to point distance higher than a given threshold.
- Rejection of worst n% pairs based on some metric.
- Pairs containing points on end vertices.
- Rejection of pairs whose point to point distance is higher than n*σ.
- Rejection of pairs that are not consistent with their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

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

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

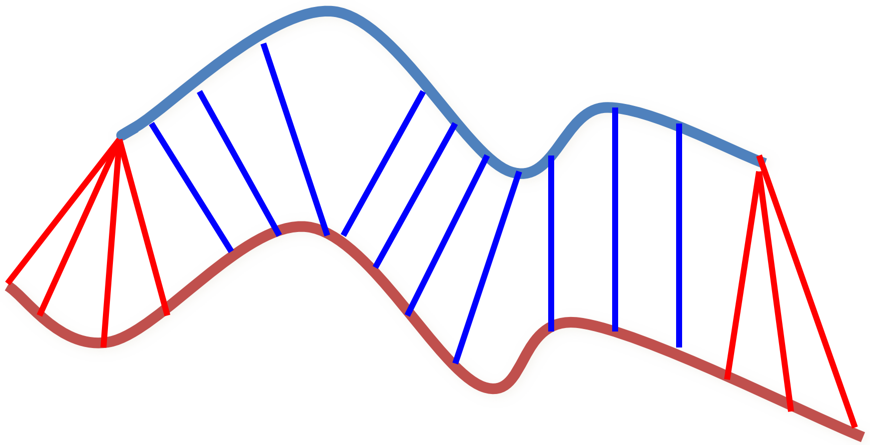
Rejecting Pairs

Distance thresholding



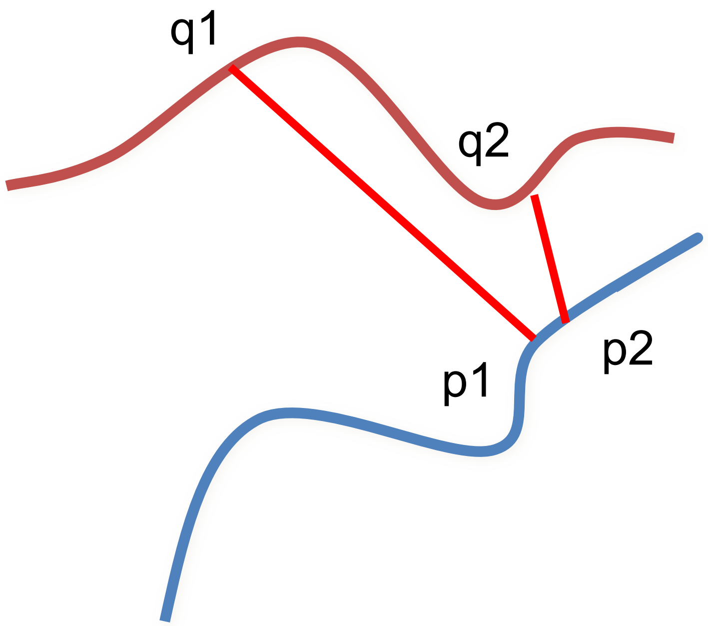
Rejecting Pairs

Points on end vertices

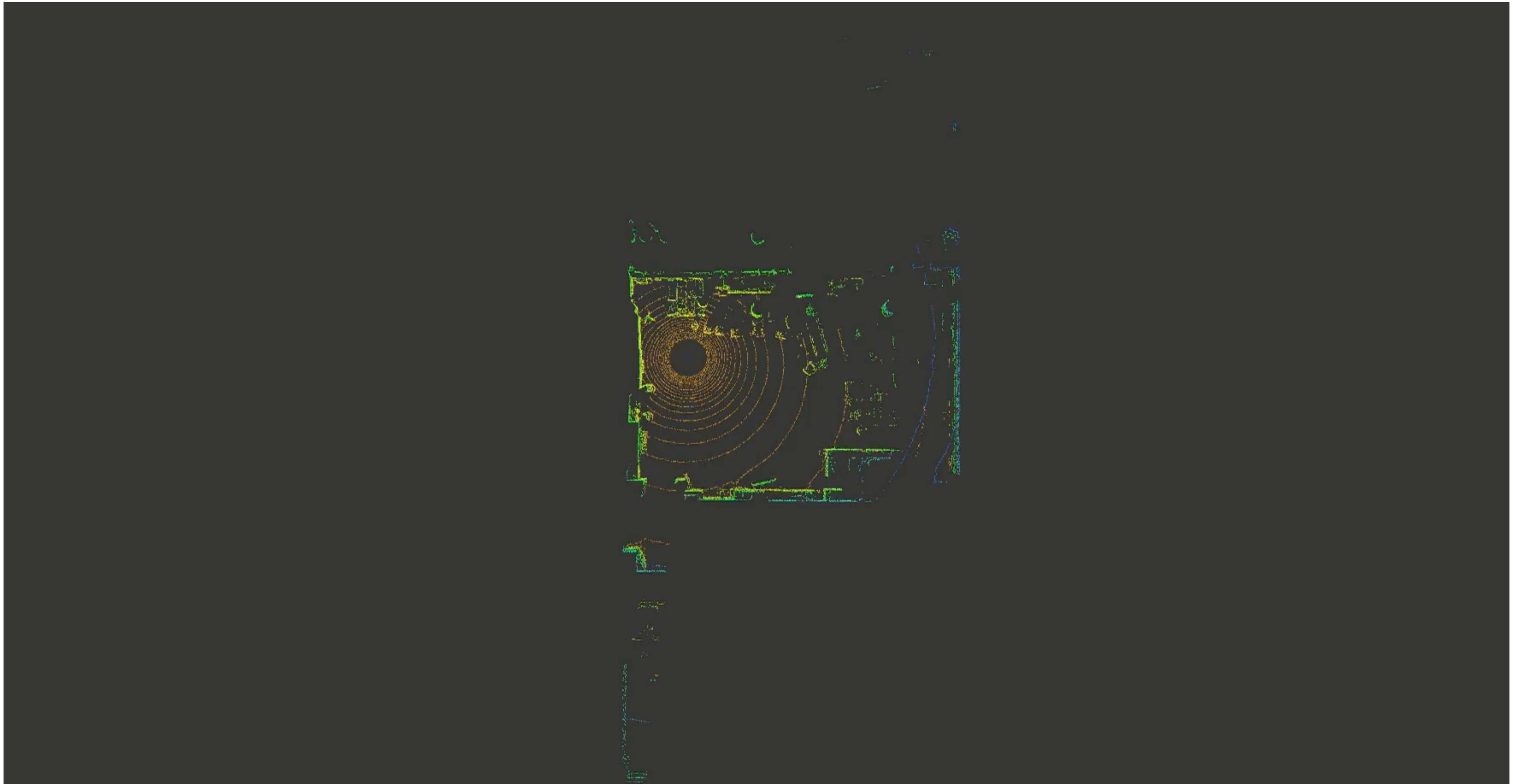


Rejecting Pairs

Inconsistent Pairs



BLAM: ICP in action



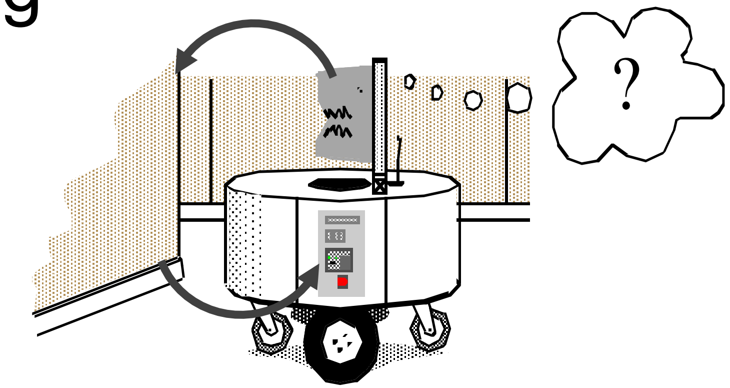

PrimeSense

DEFINITION OF SLAM

What is SLAM?

- **Localization**: inferring location given a map
- **Mapping**: inferring a map given locations
- **SLAM**: learning a map and locating the robot simultaneously
- SLAM has long been regarded as a chicken-and-egg problem:

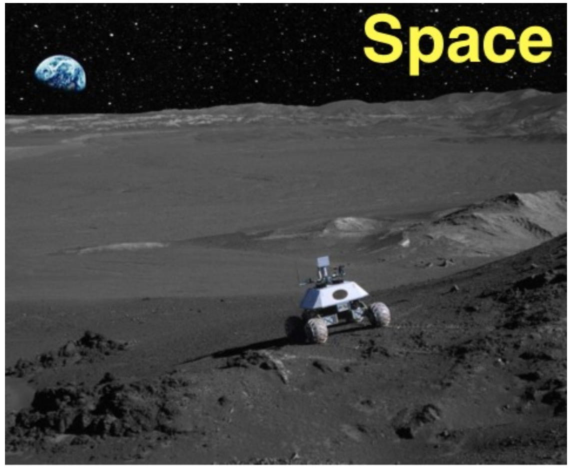
  - a map is needed for localization and
  - a pose estimate is needed for mapping

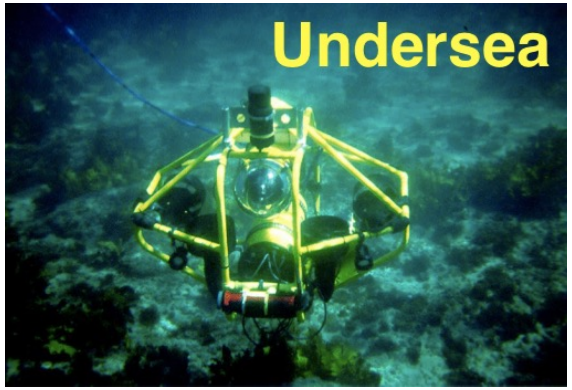


Material derived from Wolfram Burgard:
http://ais.informatik.uni-freiburg.de/teaching/ss20/robotics/slides/13-slam.pdf

SLAM Applications

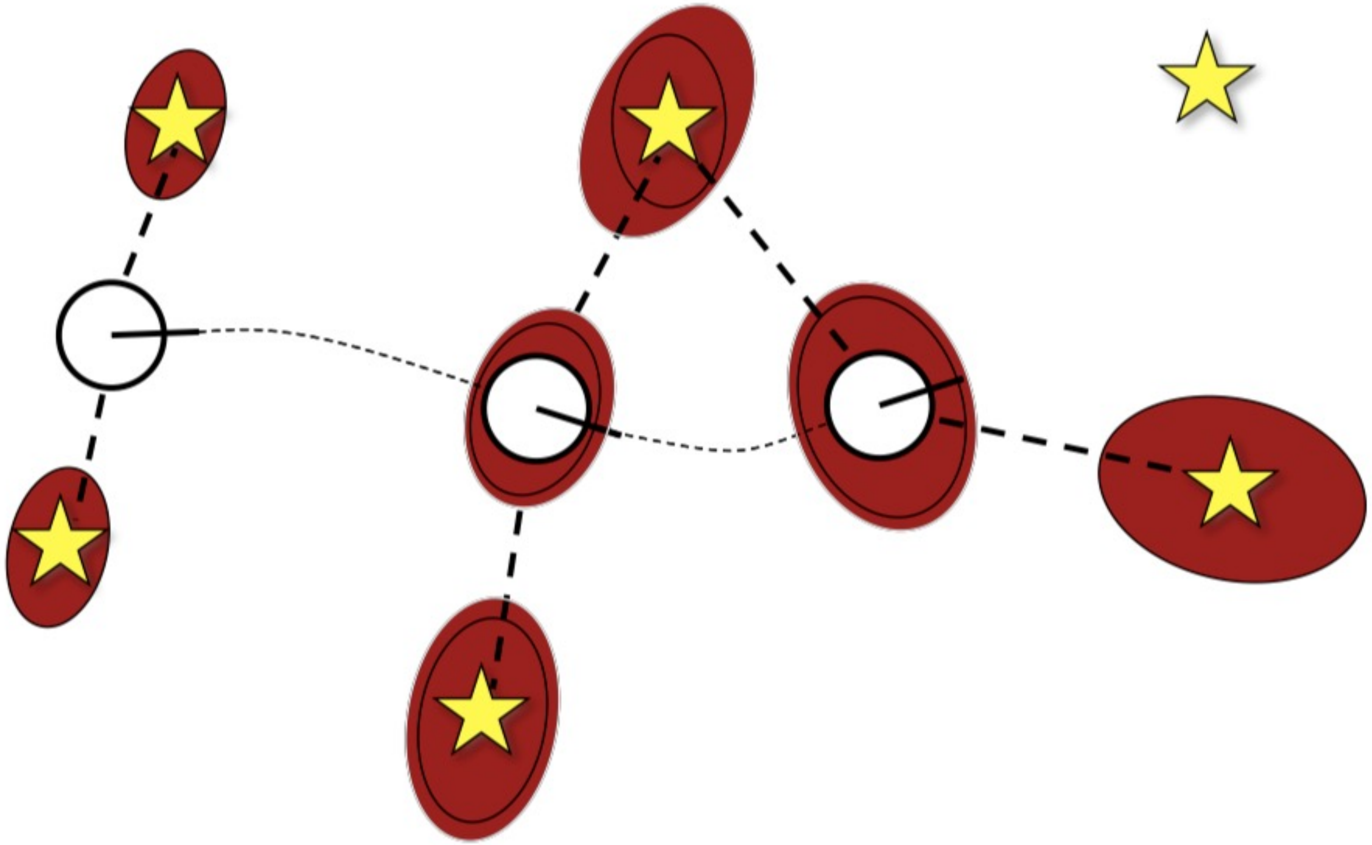
- At home: vacuum cleaner, lawn mower
- Air: surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of mines
- Space: terrain mapping for localization
- ……





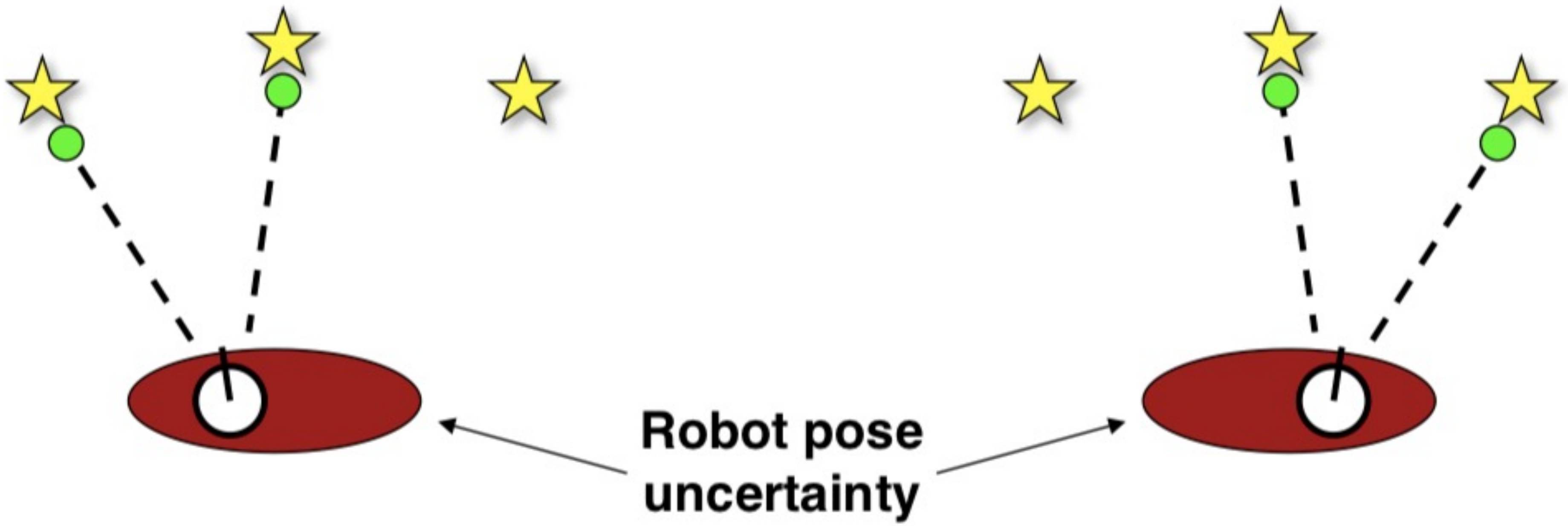
Why is SLAM a Hard Problem?

- Robot path and map are both unknown
- Errors in map and pose estimates correlated



Why is SLAM a Hard Problem?

- The mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences (divergence)



Overview of SLAM Methods

- Camera
  - Feature-Based Methods
    - MonoSLAM
    - PTAM
    - ORB-SLAM
  - Direct Methods
    - DTAM
    - LSD-SLAM
    - DSO
  - Semi-Direct Methods
    - SVO
  - Others
    - PoseNet
    - CNN-SLAM
    - …
- Laser
  - Pose Graph
    - Cartographer
    - Karto-SLAM
    - Hector-SLAM
    - BLAM
    - LIO
  - Particle Filter
    - FastSLAM
    - Gmapping
  - Extended Kalman Filter
    - EKF-SLAM
    - LINS
  - Others
    - LOAM
    - IMLS-SLAM
    - …

SLAM Front-end & Back-end

- Front-end
  - calculate relative poses between several frames/ to map
    - scan matching
    - image registration
    - …
  - estimate absolute poses
  - construct the local map
- Back-end
  - optimize the absolute poses and mapping
  - only if a loop was closed

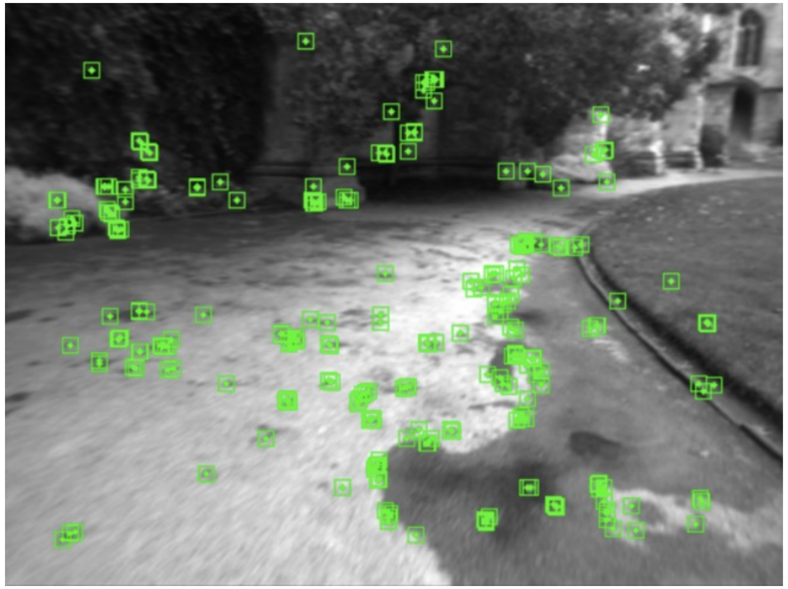
https://ww2.mathworks.cn/help/nav/ug/implement-simultaneous-localization-and-mapping-with-lidar-scans.html

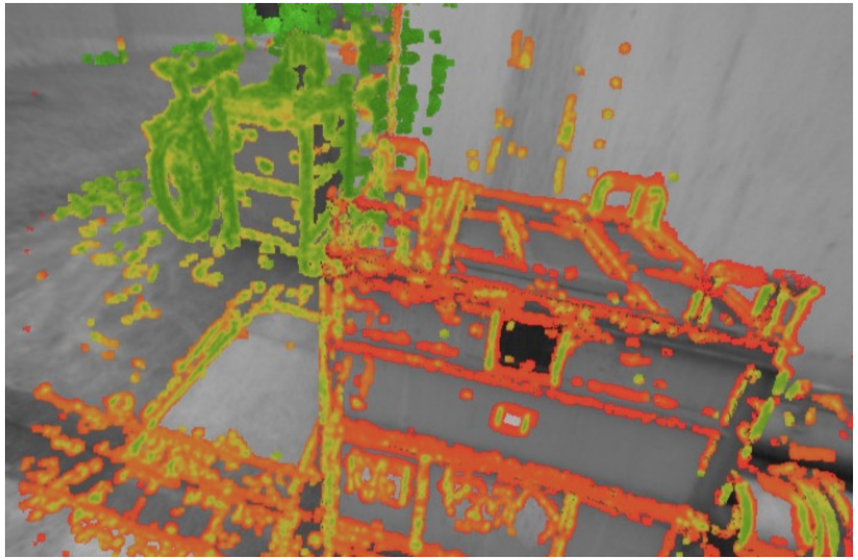
FRONT END – LASER - ICP

FRONT END - CAMERA

Methods

- **Feature-based Methods**
  - SIFT
  - ORB (ORB-SLAM)
  - BRISK
  - AKAZE
- **Direct Methods**
  - Optical Flow
  - Inverse Depth (LSD-SLAM)
  - Fourier-Mellin Transform
- **Semi-Direct Methods**
  - SVO





more details in the next lectures

Feature-based Methods

- Feature Extraction
  - Feature Detectors & Feature Descriptor; more in vision lectures

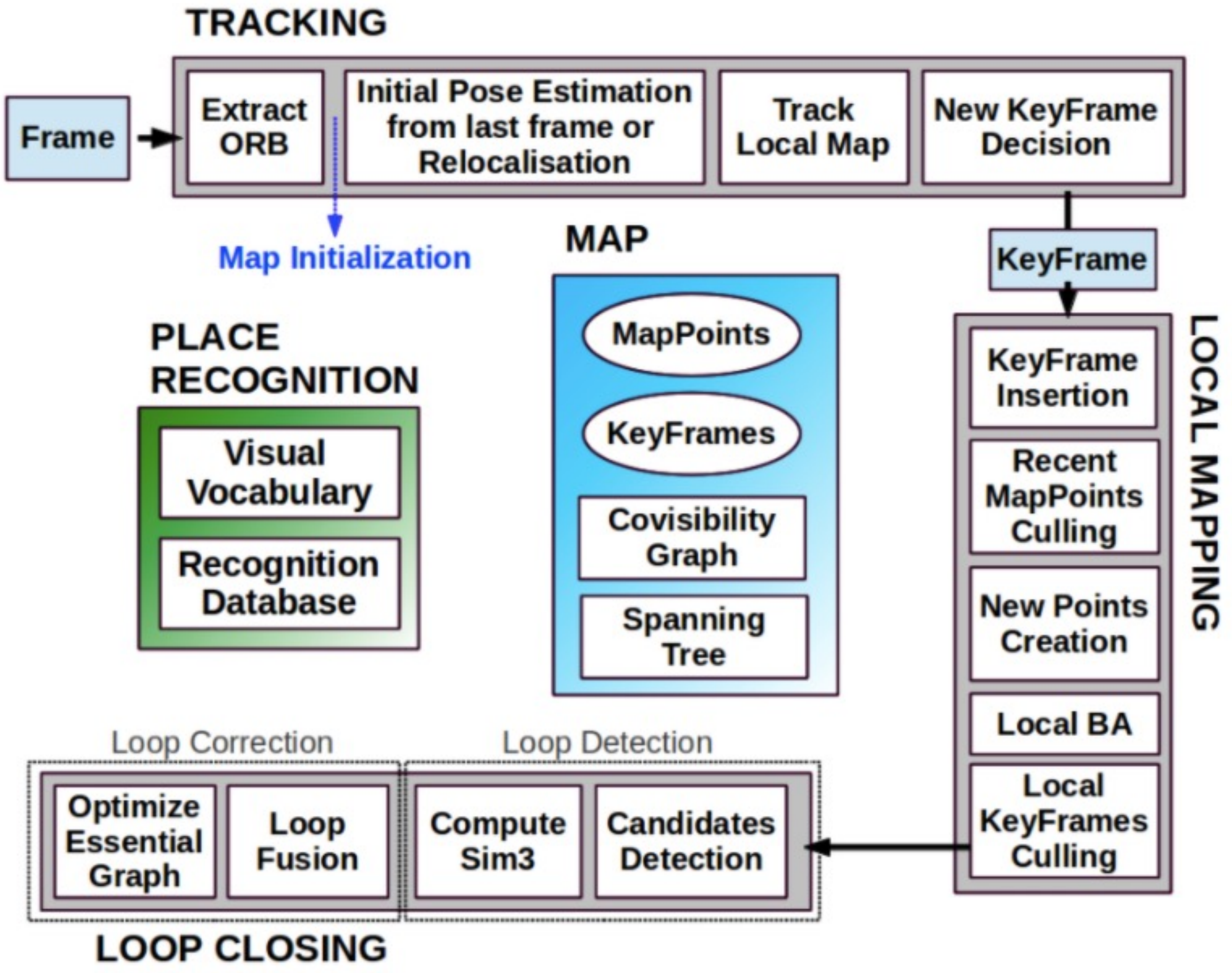
    - ORB, SIFT, AKAZE, BRISK, etc …
- Feature Matching

    - BFM, KNN, etc …
- Relative Pose Calculation

    - 5-pt, 7-pt, 8-pt, PnP, etc …

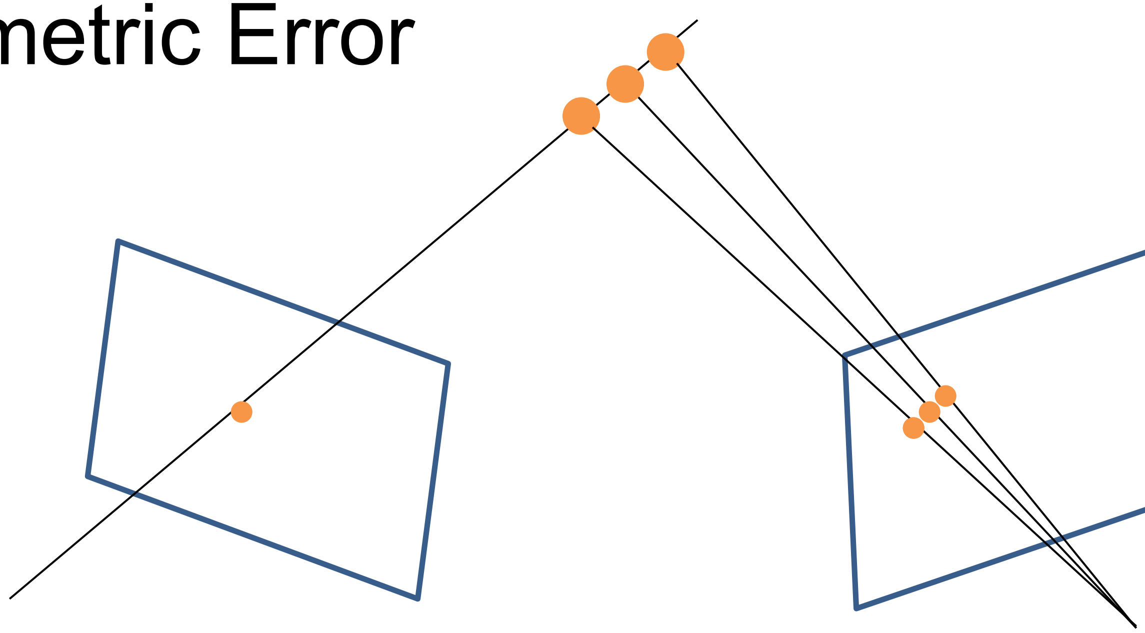
Feature-based Method: ORB-SLAM

Mur-Artal R, Montiel J M M, Tardos J D. ORB-SLAM: a versatile and accurate monocular SLAM system[J]. IEEE transactions on robotics, 2015, 31(5): 1147-1163.



Direct Method: LSD-SLAM

- Construct Photometric Error

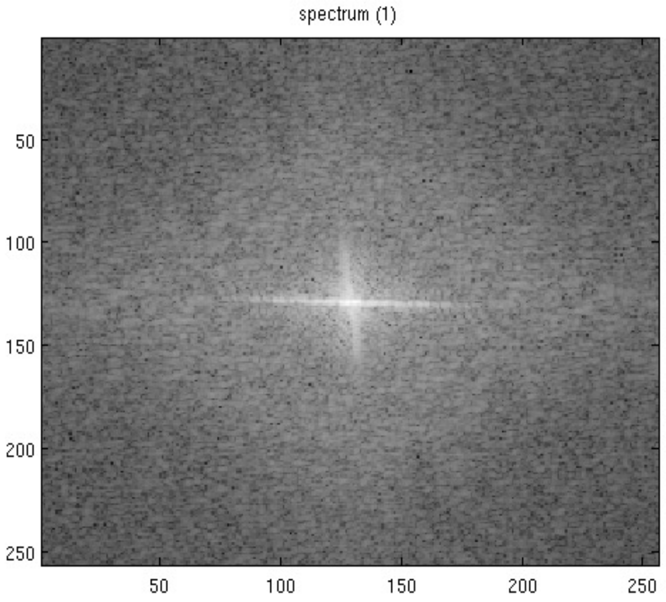


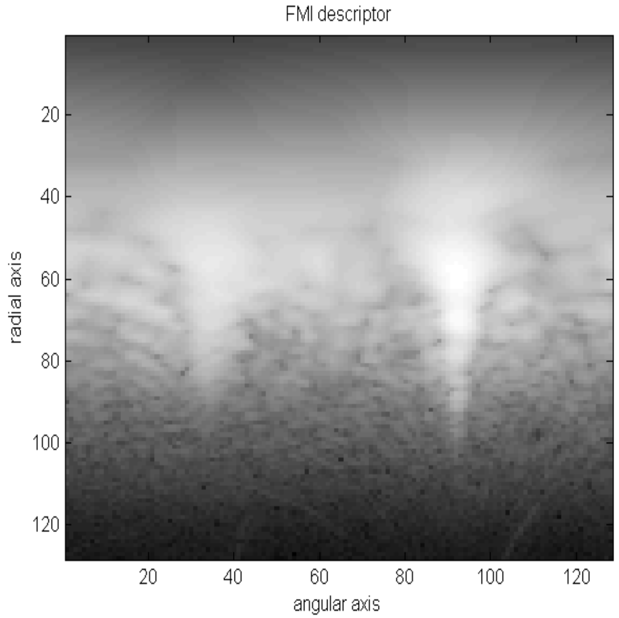
- Construct Depth Error
- Minimize Objective Error Function

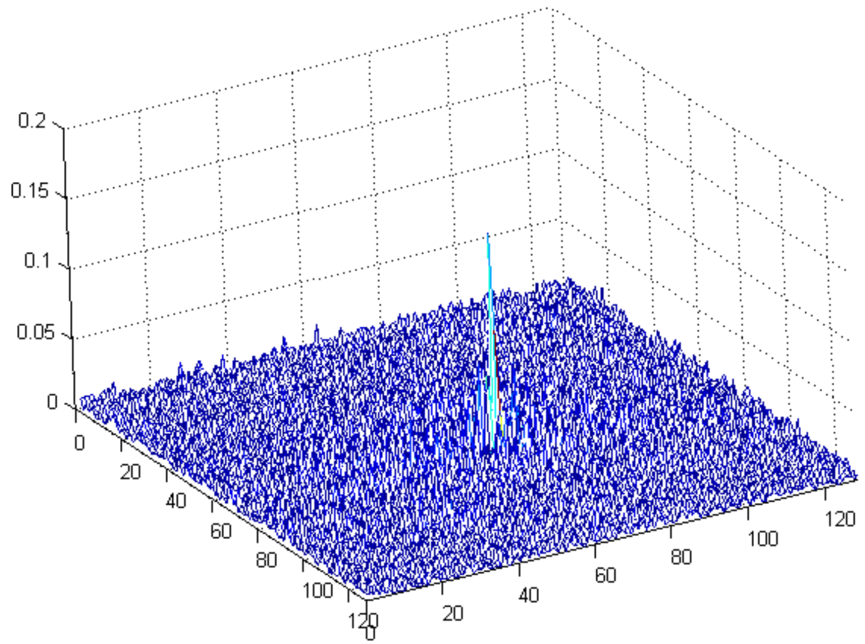
Engel J, Schöps T, Cremers D. LSD-SLAM: Large-scale direct monocular SLAM[C]//European conference on computer vision. Springer, Cham, 2014: 834-849.

Direct Method: Fourier Mellin Transform

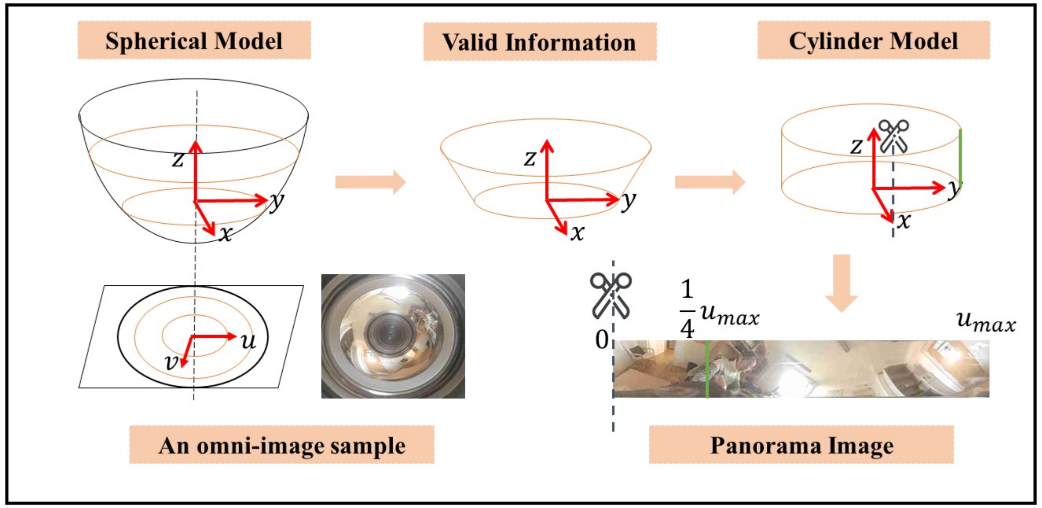
- Spectral based registration: detection of scaling, rotation and translation in 2 subsequent frames
- Processing spectrum magnitude decouples translation from affine transformations
  - Detection of signal shift between 2 signals by phase information
  - Resampling to polar coordinates → Rotation turns 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

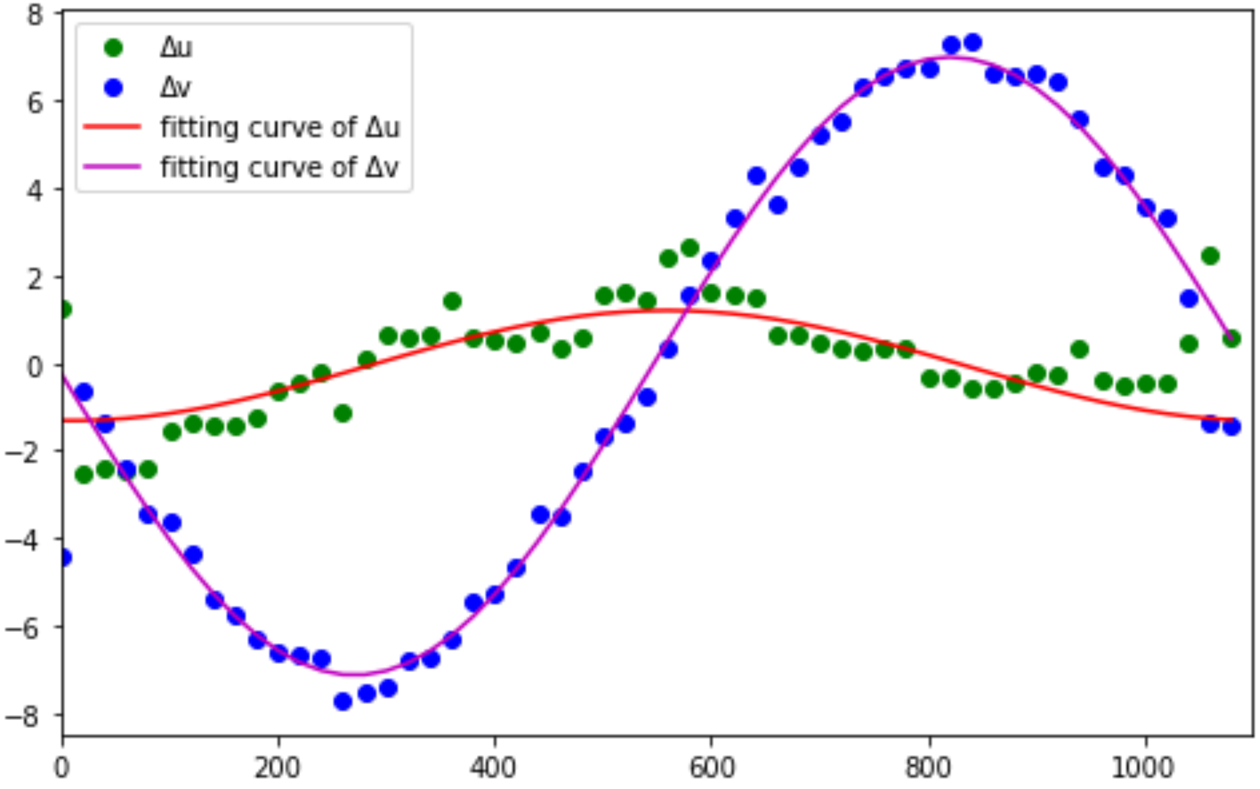






Pose Estimation for Omni-directional Cameras using Sinusoid Fitting





$$y = B + A\sin(\omega x + \phi) \quad \text{(1a)}$$

$$\Delta v(u_p) = \lambda_p t_z + \gamma \left\| P_{xy}(R) \cdot \sin\left(\gamma u_p + \frac{P_{xy}(R)}{\|P_{xy}(R)\|}\right) \right\| \quad \text{(1b)}$$

$$\Delta u(u_p) = \gamma P_z(R) + \lambda_p \|P_{xy}(t)\| \cdot \sin(\gamma u_p + \hat{t}_{xy}) \quad \text{(1c)}$$

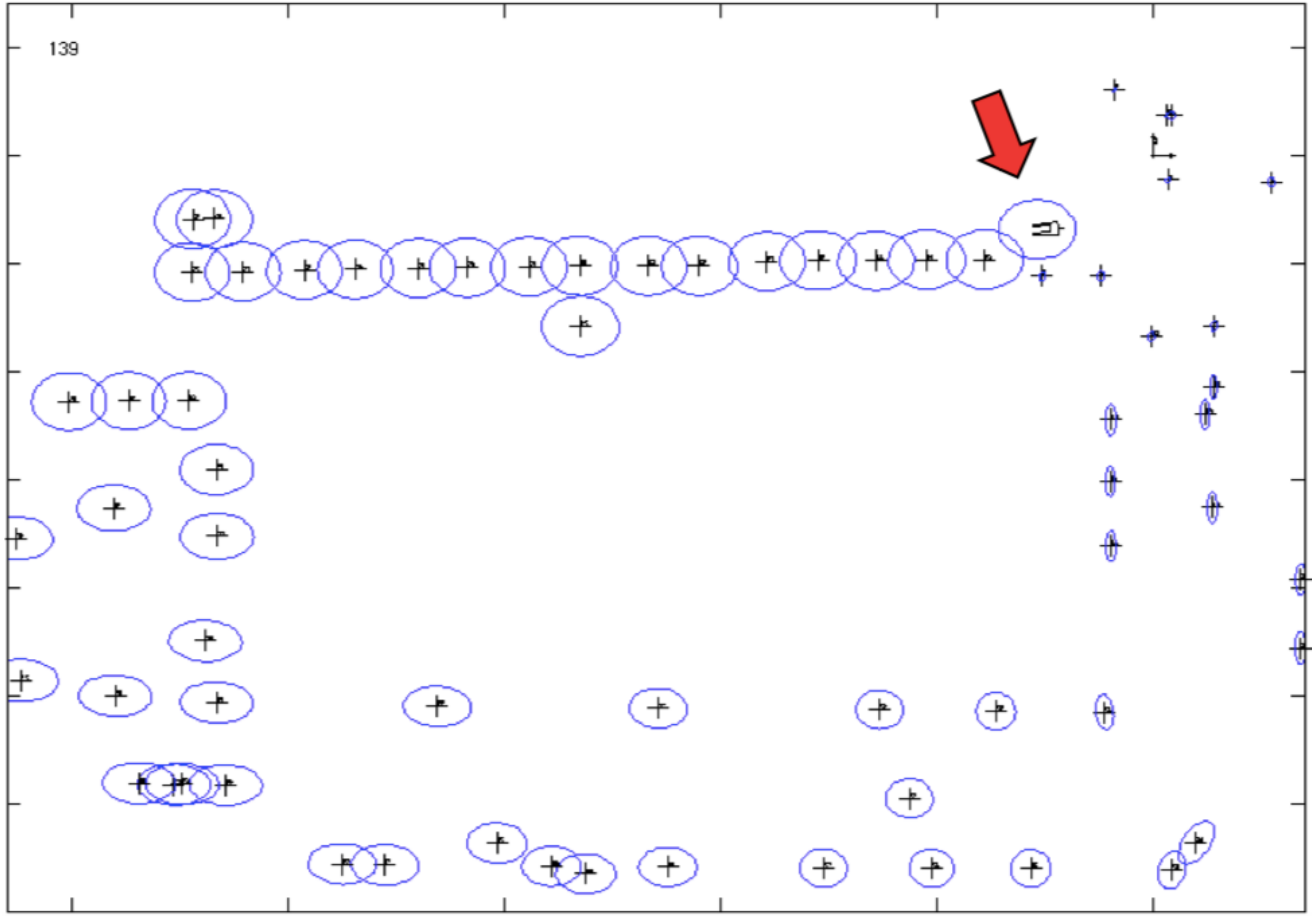
BACK END

Overview of Back-end

- Loop Detection
  - Find candidates of scan pairs/ scan with old map
  - E.g. based on global pose estimated (chain rule) OR image similarity (bag of words)
- Loop Closure
  - E.g. use scan matching to find the transform AND its uncertainty
- Optimization
  - Pose Graph optimization (e.g. minimize error of poses, based on uncertainty)
  - Bundle Adjustment
- Map Rendering
  - E.g. generate grid map based on optimized graph

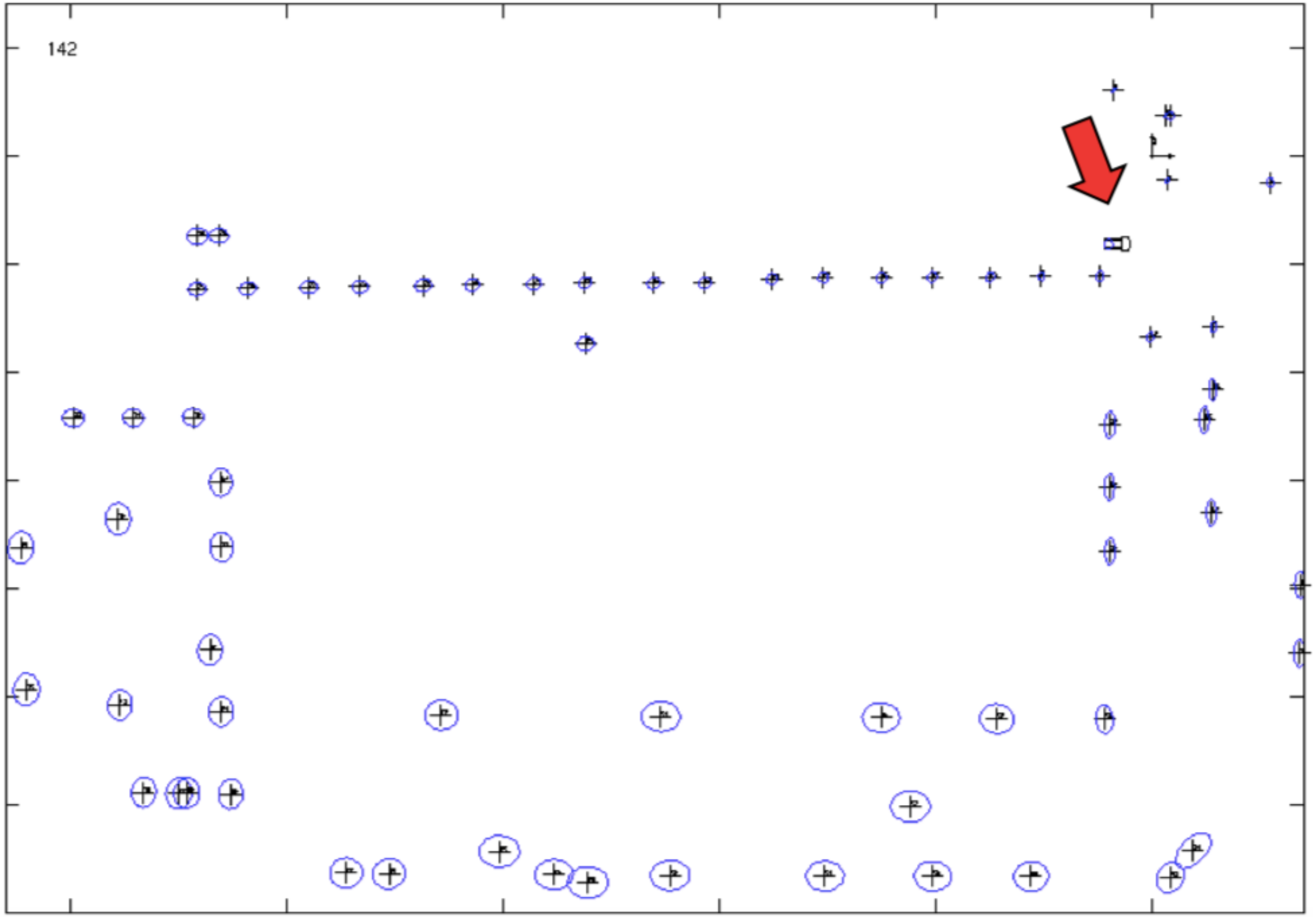
Loop Closure

- Before loop closure



Loop Closure

- After loop closure

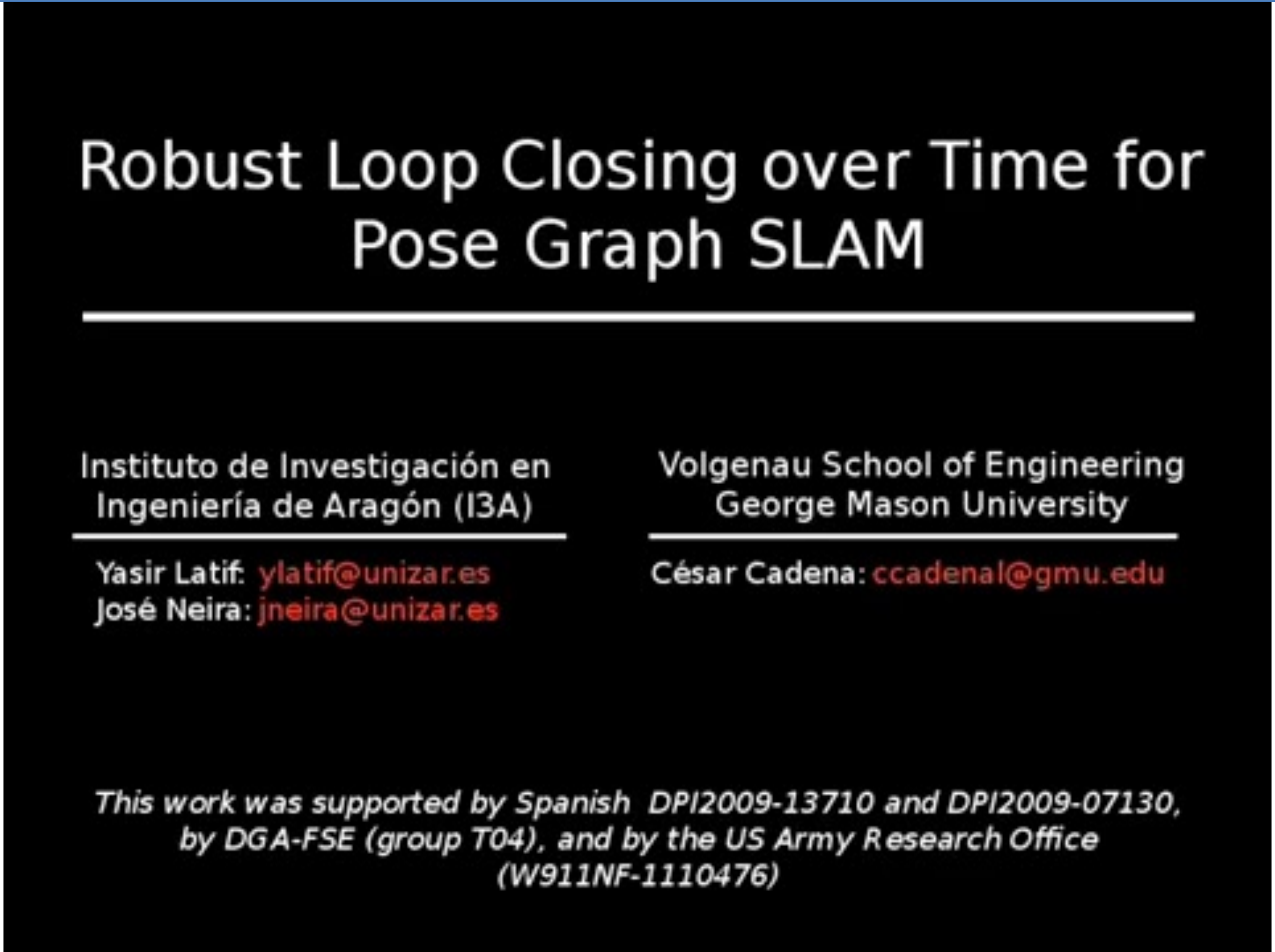


Loop Closure

- Recognizing an already mapped area, typically after a long exploration path (the robot “closes a loop”)
- Structurally identical to data association, but
  - high levels of ambiguity
  - possibly useless validation gates
  - environment symmetries
- Uncertainties collapse after a loop closure (whether the closure was correct or not)

Loop Closure

- By revisiting already mapped areas, uncertainties in robot and landmark estimates can be reduced
- This can be exploited when exploring an environment for the sake of better (e.g. more accurate) maps
- Exploration: the problem of where to acquire new information


Robust Loop Closing over Time for Pose Graph SLAM
Instituto de Investigación en Ingeniería de Aragón (I3A)
Yasir Latif: ylatif@unizar.es
José Neira: jneira@unizar.es
Volgenau School of Engineering George Mason University
César Cadena: ccadenal@gmu.edu
This work was supported by Spanish DPI2009-13710 and DPI2009-07130, by DGA-FSE (group T04), and by the US Army Research Office (W911NF-1110476)

OVERVIEW:
THREE SLAM PARADIGMS

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

EKF SLAM: overview

- **Extended state vector** $\boldsymbol{y_t}$: robot pose $\boldsymbol{x_t}$ + position of all the features $\boldsymbol{m_i}$ in the map:

$$y_t = [x_t, m_0, \ldots, m_{n-1}]^T$$

- Example: 2D line-landmarks, size of $\boldsymbol{y_t}$ = $3+2n$: three variables to represent the robot pose + $2n$ variables for the n line-landmarks having vector components

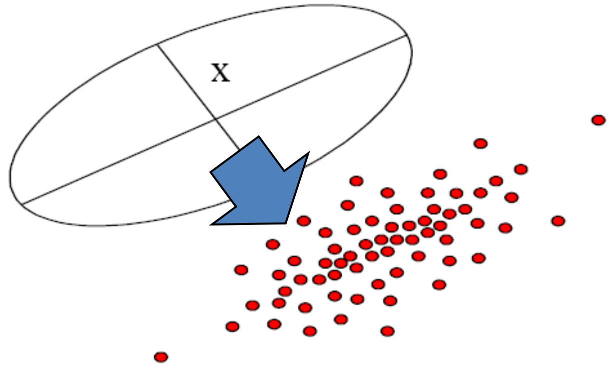
$$(\alpha_i, r_i)$$

$$y_t = [x_t, y_t, \theta_t, \alpha_0, r_0, \ldots, \alpha_{n-1}, r_{n-1}]^T$$

- As the robot moves and takes measurements, the state vector and covariance matrix are updated using the standard equations of the extended Kalman filter
- Drawback: EKF SLAM is computationally very expensive.

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:

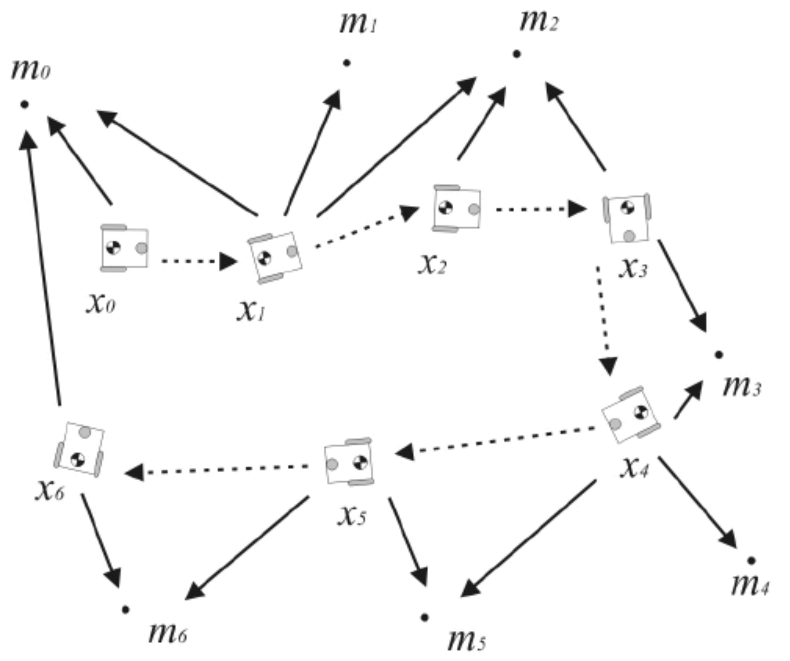
     1. Compare particle's prediction of measurements with actual measurements
     2. Particles whose predictions match the measurements are given a high weight

  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.

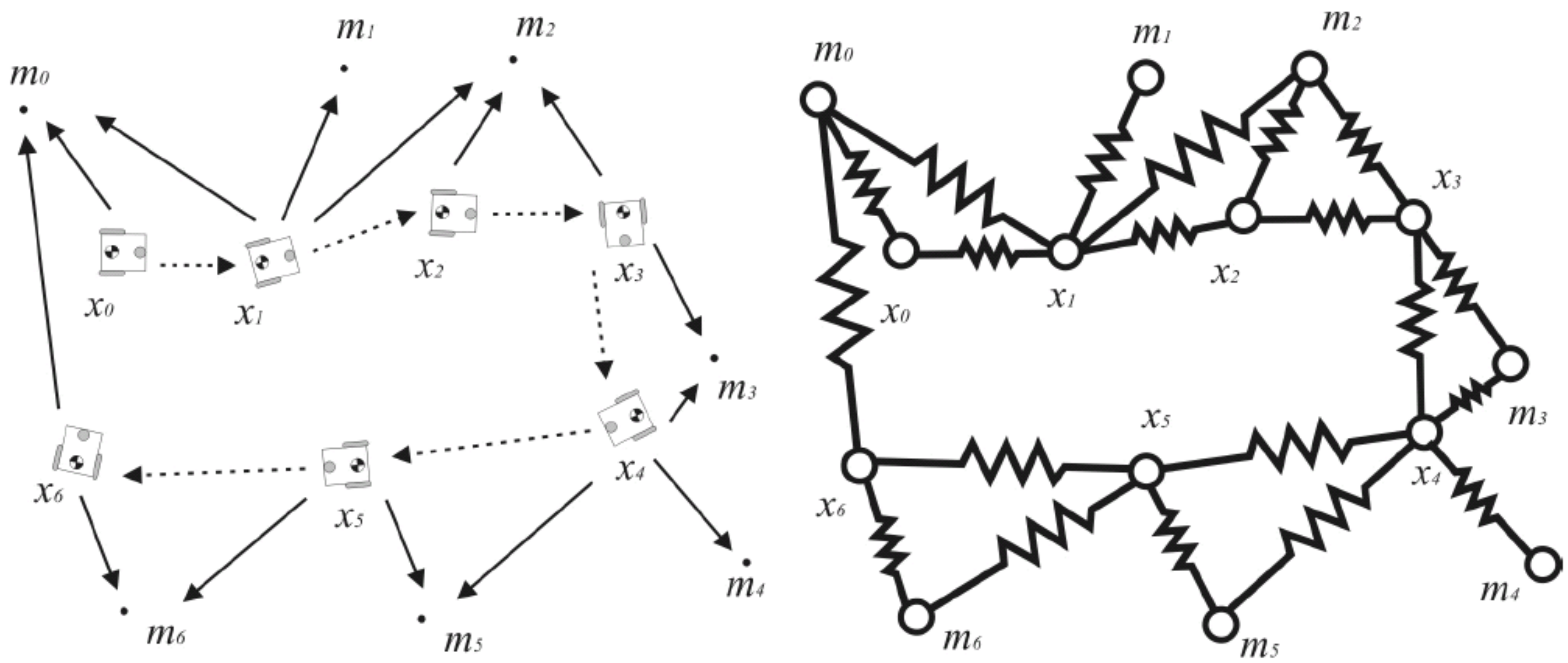
Graph-Based SLAM (1/3)

- 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)

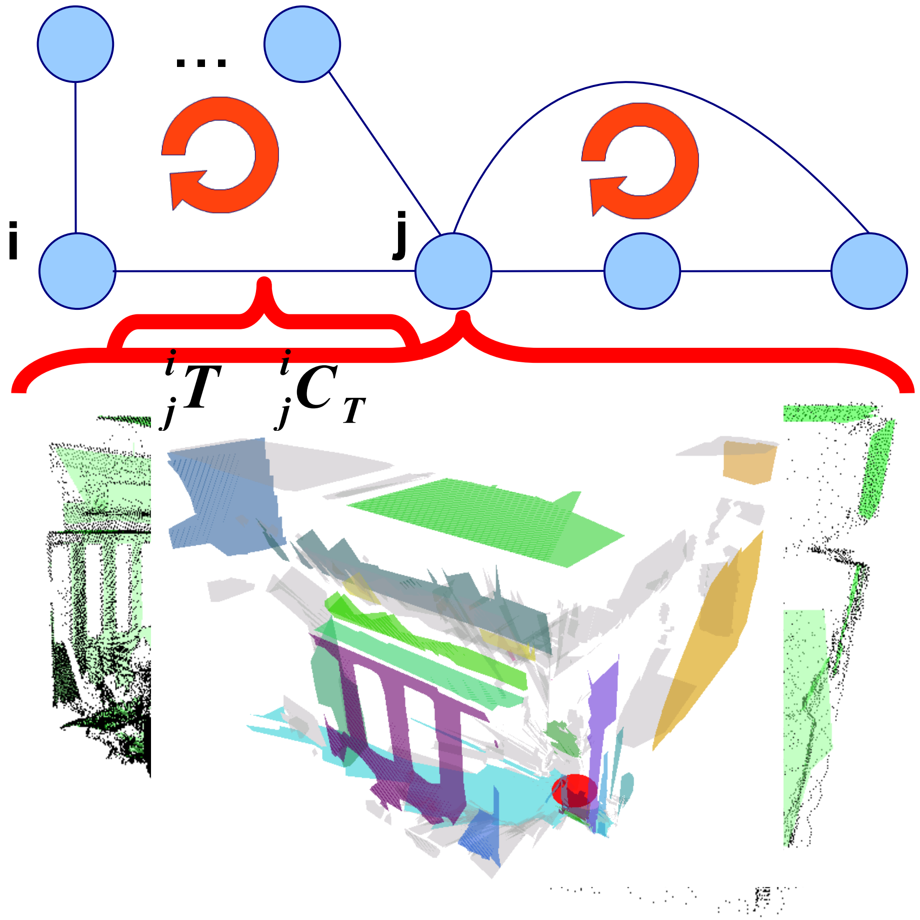
- Significant advantage of graph-based SLAM techniques over EKF SLAM:
  - EKF SLAM: computation and memory for to update and store the covariance matrix is quadratic with the number of features.
  - Graph-based SLAM: update time of the graph is constant and the required memory is linear in the number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

- Libraries for graph-based slam: g2o, ceres

SLAM EXAMPLES

Jacobs 3D Mapping – Plane Mapping

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



Pathak, K., A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online 3D SLAM by Registration of Large Planar Surface Segments and Closed Form Pose-Graph Relaxation", *Journal of Field Robotics, Special Issue on 3D Mapping*, vol. 27, no. 1, pp. 52-84, 2010.

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

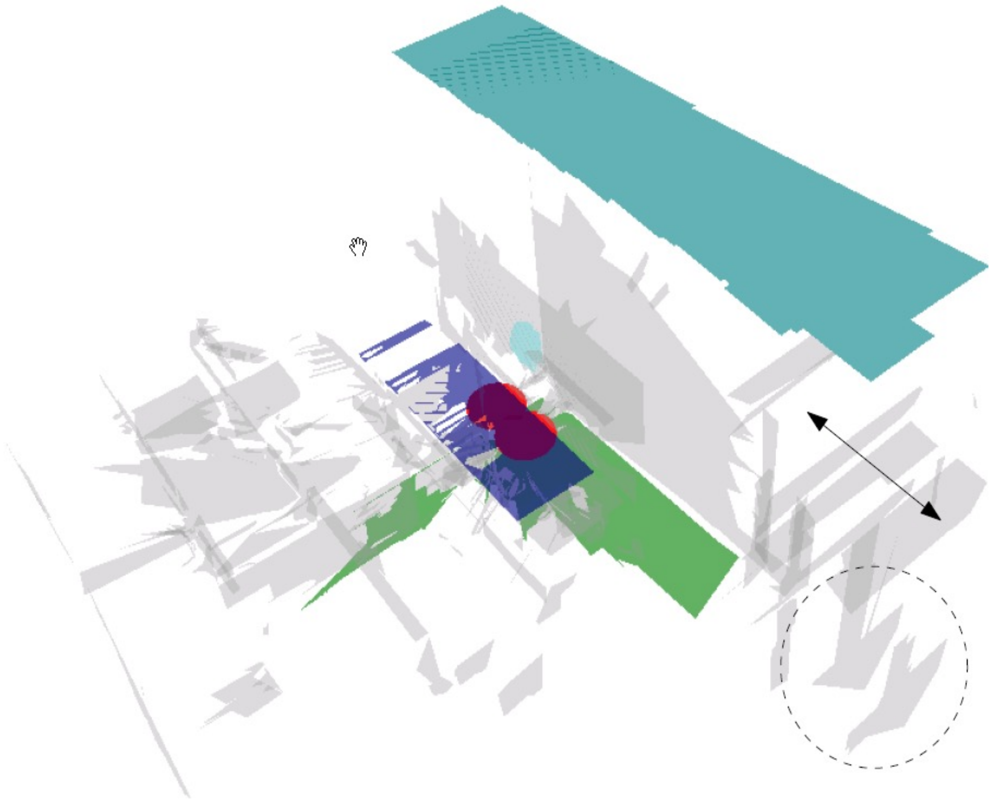
- **Range Image Segmentation**
  - Is based on region growing algorithm
  - Uses incremental formulas, therefore is fast
  - Has linear computational complexity

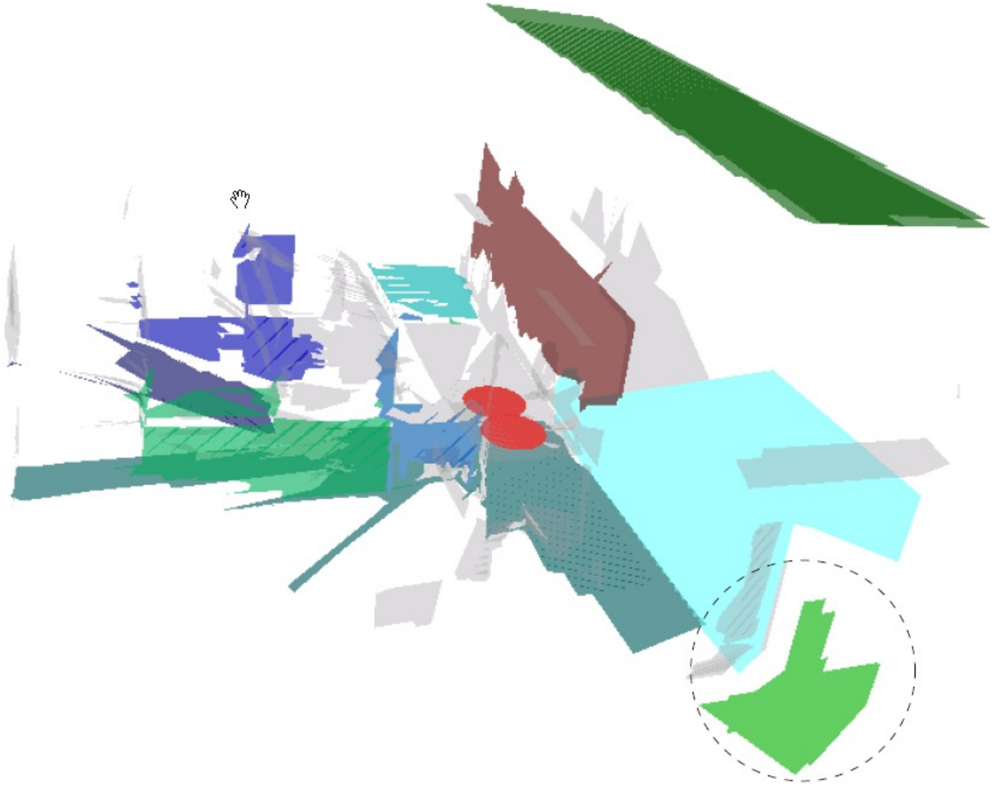
Given a range image, returns a polygonal model i.e. a set of planar features and boundaries.

Plane Registration (Scan Matching)

- Determining the correspondence set maximizing the global rigid body motion constraint.
- Finding the optimal decoupled rotations (Wahba's problem) and translations (closed form least squares) with related uncertainties.
- No motion estimates from any other source are needed.
- Very fast
- MUMC: Finding Minimally Uncertain Maximal Consensus
  - Of matched planes
- Idea: select two non-parallel plane matches => fixes rotation and only leaves one degree of translation!

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