上海科技大学
ShanghaiTech University

CS283: **Robotics Spring 2025: Localization**

Sören Schwertfeger / 师泽仁

ShanghaiTech University

Map Representation: what is “saved” in the map

- Points (surface of objects, buildings): 2D or 3D
  - What: x,y or x,y,z coordinates; Optional: intensity; maybe RGB; maybe descriptor; temperature; …
  - From range sensors (laser, ultrasound, stereo, RGB-D): dense
  - From cameras (structure from motion; feature points): sparse
  - Variant: kd-tree
- Grid-map: 2D or 3D
  - Option: probabilistic grid map
  - Option: elevation map
  - Option: cost map
  - Option: Truncated Signed Distance Field
  - Option: Normal Distributions Transform (NDT)
  - Variant: Quad-tree; Oct-tree
- Higher-level Abstractions
  - Lines; Planes; Mesh
  - Curved: splines; Superquadrics
- Semantic Map
  - Assign semantic meaning to entities of a map representation from above
  - E.g. wall, ceiling, door, furniture, car, human, tree, …
- Topologic Map
  - High-level abstraction: places and connections between them
- Hierarchical Map
  - Combine Maps of different scales. E.g.:
  - Campus, building, floor
- Pose-Graph Based Map
  - Save (raw) sensor data in graph, annotated with the poses; generate maps on the fly
- Dynamic Map
  - Capture changing environment
- Hybrid Map
  - Combination of the above

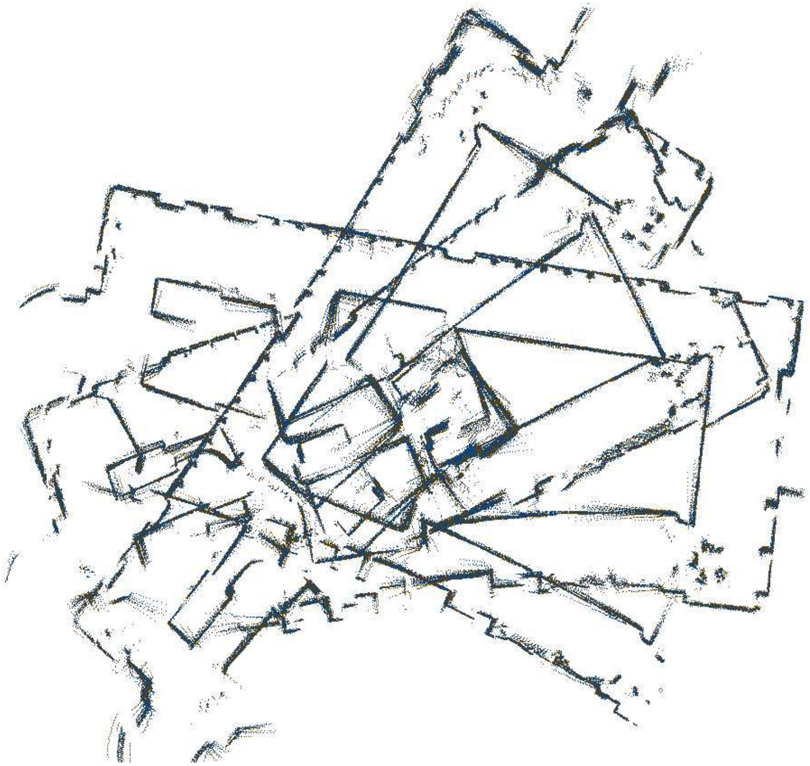
Mapping

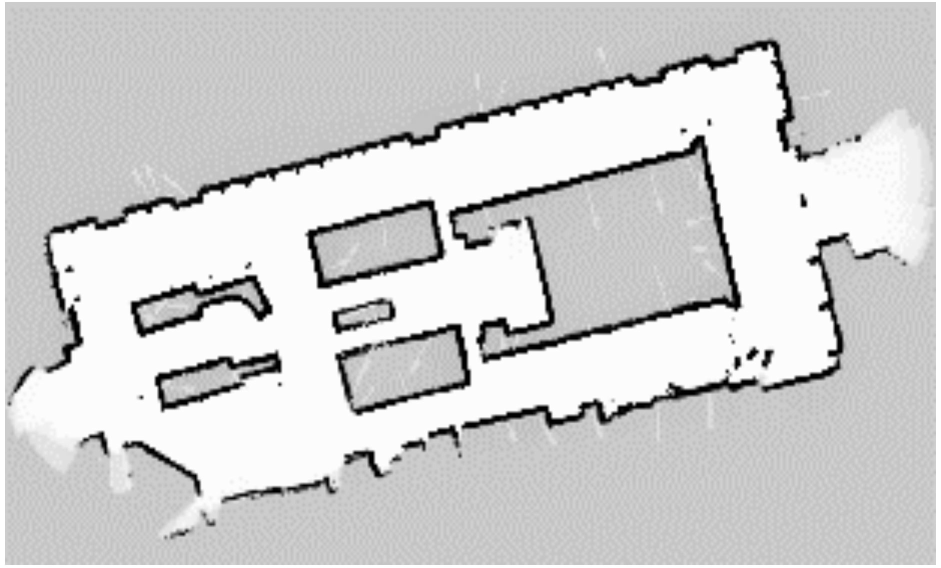
- Process of building a map
- Basic principle:
  1. Initialize the map with unknown or free
  2. Take a sensor scan
  3. Maybe pre-process it (e.g. plane detection)
  4. Localize the robot w.r.t. the map frame (maybe difficult!)
  5. Transform the (processed) sensor scan to the global frame
  6. "Merge" the new data with the old map data, e.g.:
     - Add scanned points to map point cloud
     - Update cells in a probabilistic occupancy grid
  7. Sometimes: Also do ray-casting to mark all cells from sensor to obstacle as free
  8. Repeat for every new sensor scan

- Localization step may need the map (e.g. matching the scan against the map) => both should be done at the same time =>
- Simultaneous Localization and Mapping : SLAM

Cyclic Environments

- Small local error accumulate to arbitrary large global errors!
- This is usually irrelevant for navigation
- However, when closing loops, global error does matter



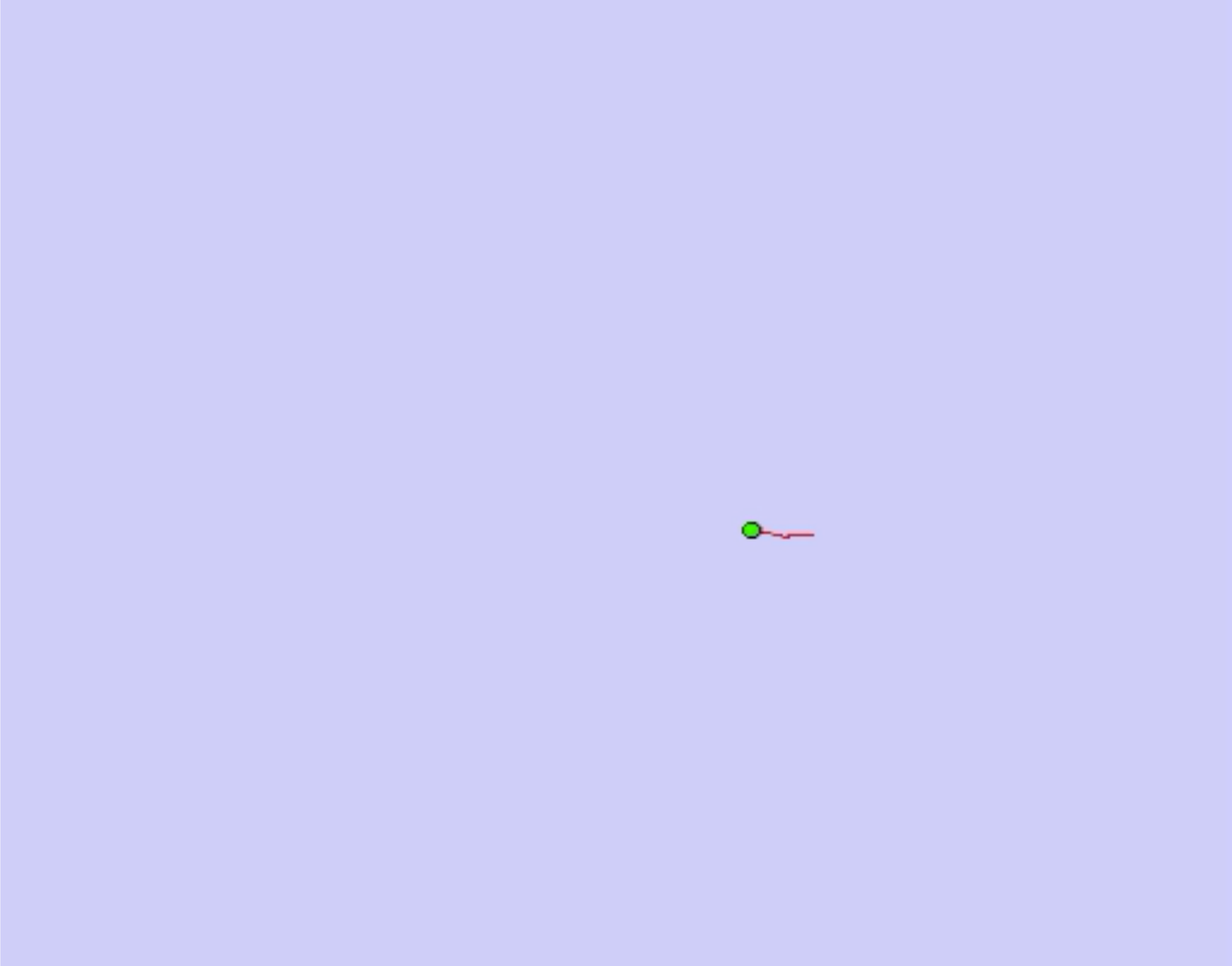


Raw Odometry

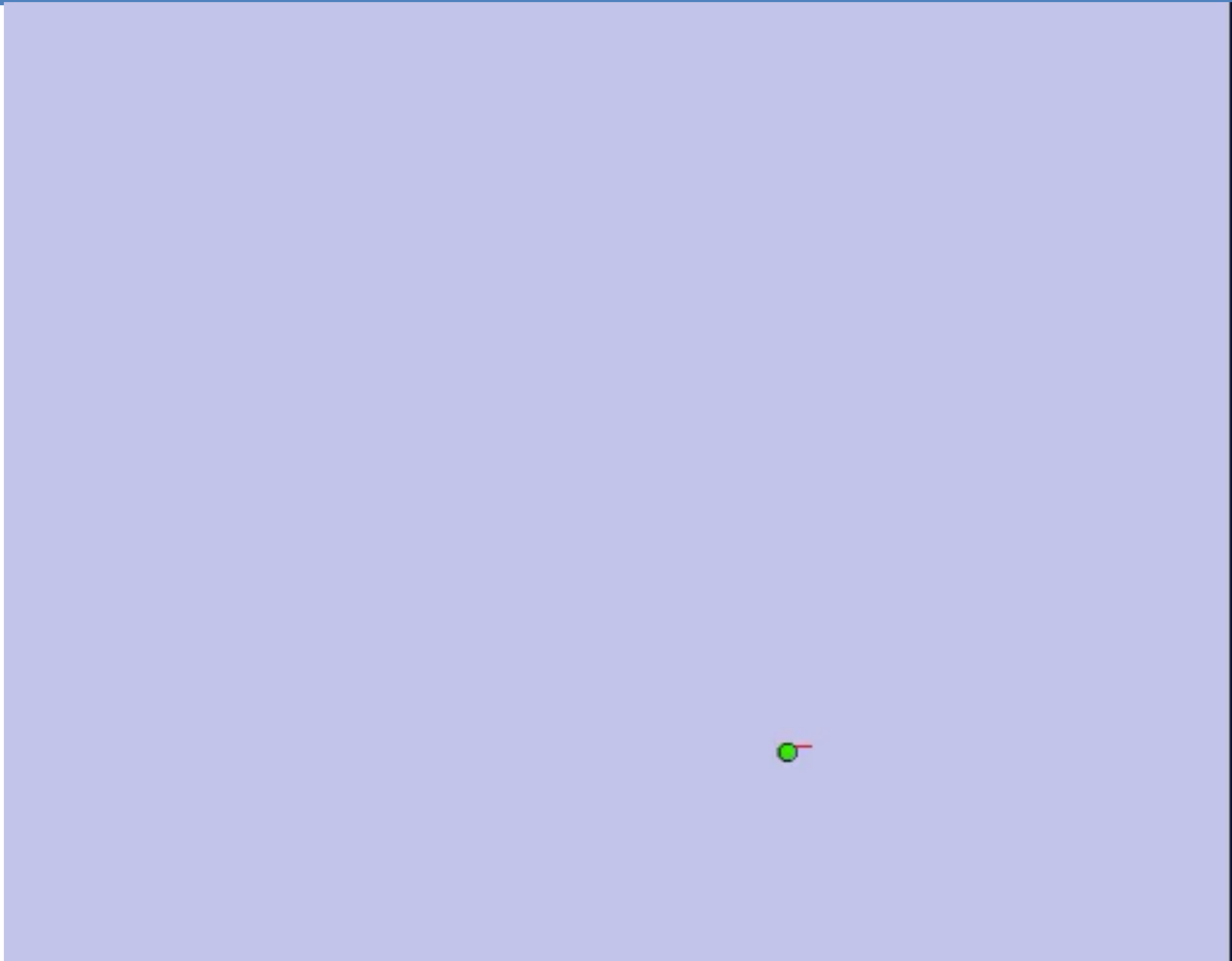
- Famous Intel Research Lab dataset (Seattle) by Dirk Hähnel

Courtesy of S. Thrun

http://robots.stanford.edu/videos.html

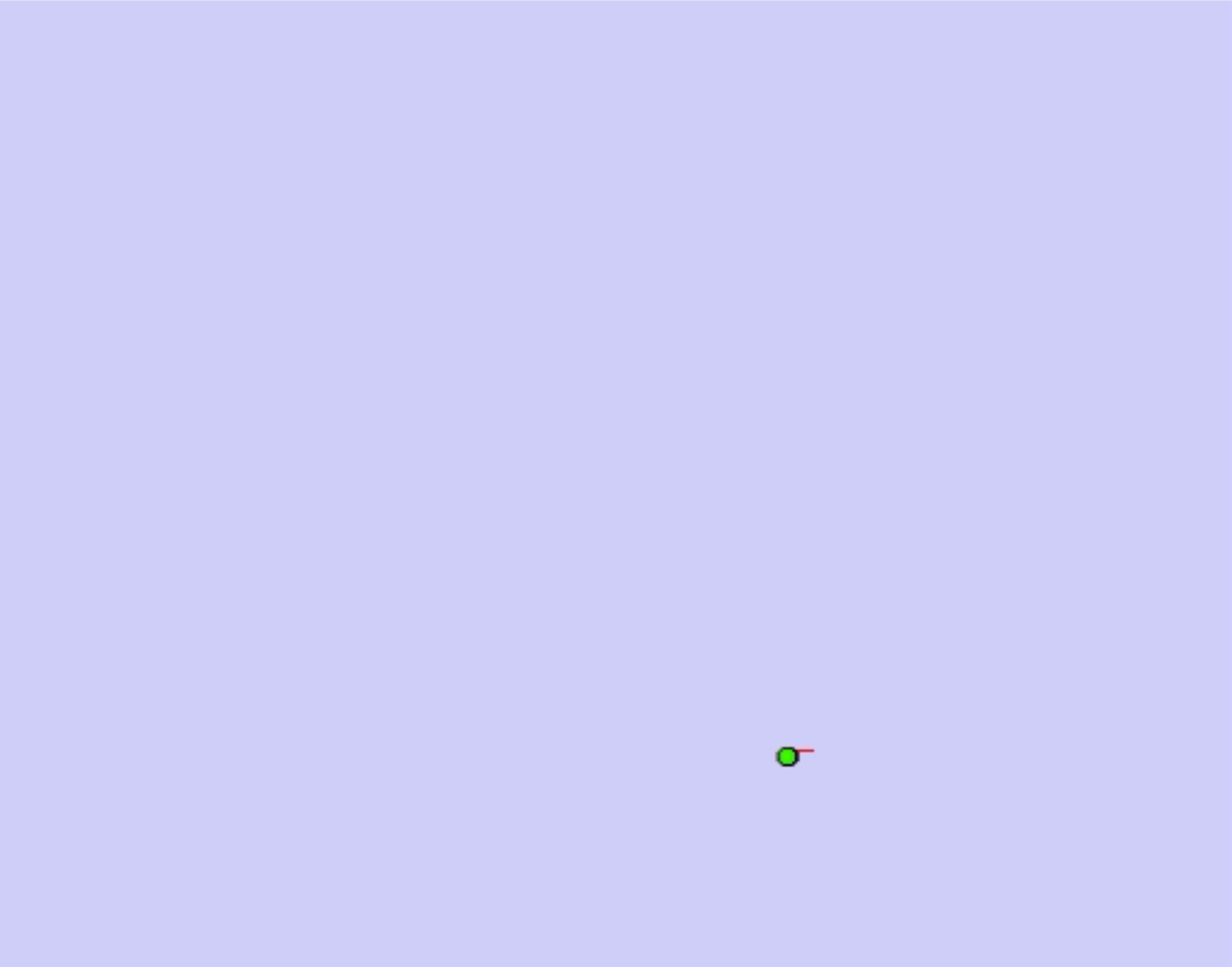


Scan Matching: compare to sensor data from previous scan

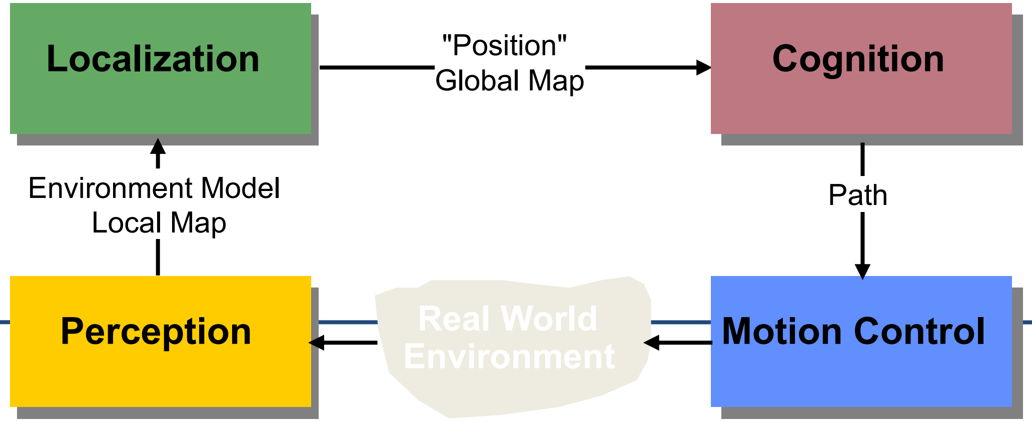


Courtesy of S. Thrun

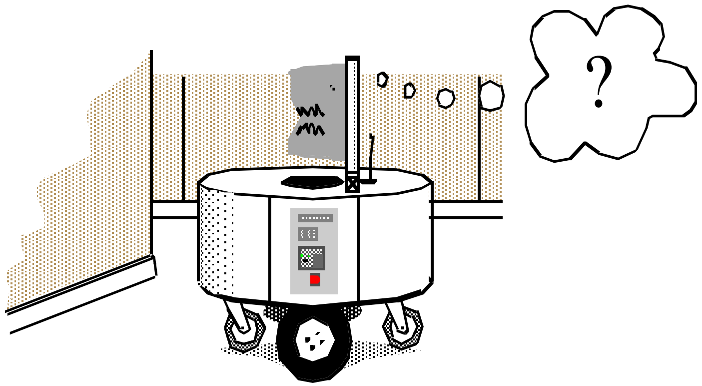
FastSLAM: Particle-Filter SLAM



Courtesy of S. Thrun



LOCALIZATION



Problem: NOISE!

- Exteroceptive Sensor Noise
  - Sensor noise is mainly influenced by environment
    e.g. surface, illumination …
  - and by the measurement principle itself
    e.g. interference two Kinects
  - Sensor noise drastically reduces the useful information of sensor readings. The solution is:
    - to model sensor noise appropriately
    - to take multiple readings into account
    - employ temporal and/or multi-sensor fusion

Effector Noise: Odometry, Deduced Reckoning

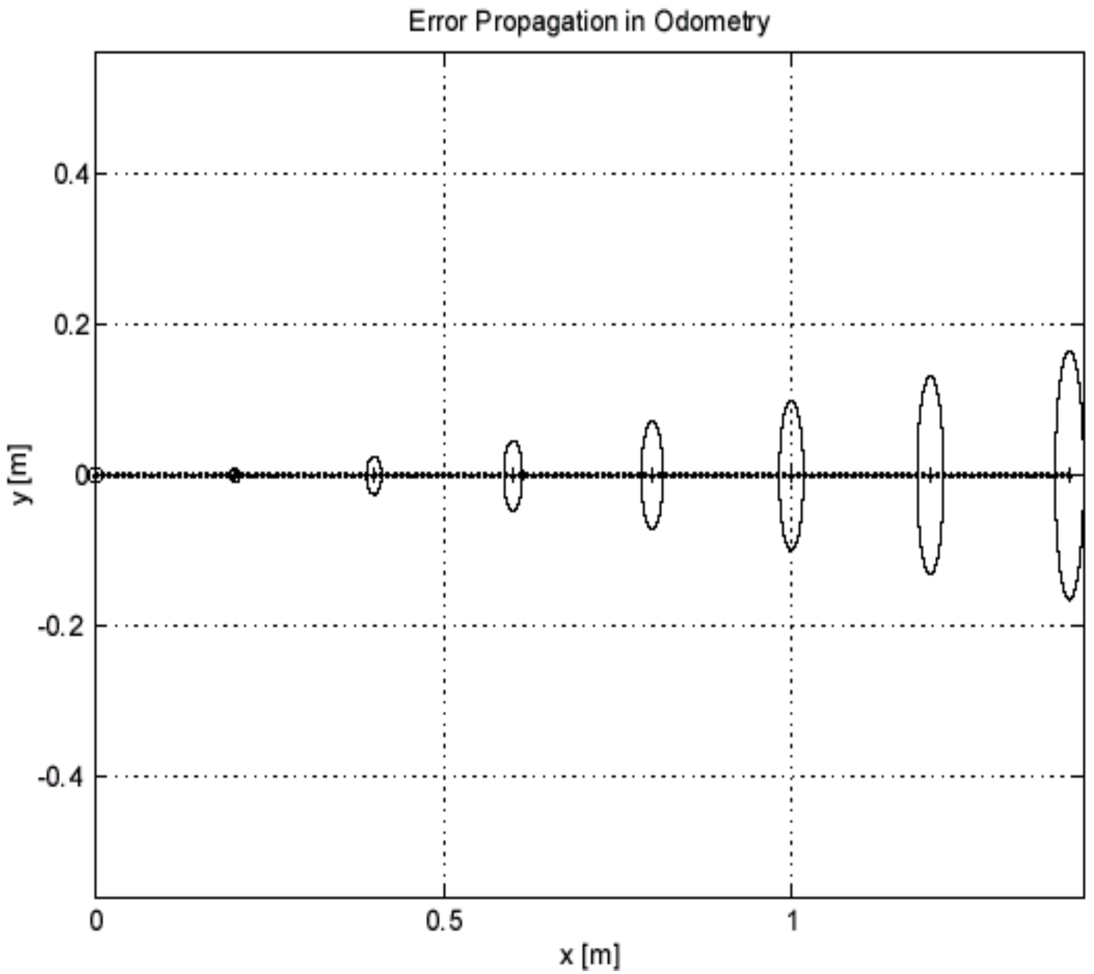
- Odometry and dead reckoning:
  Position update is based on proprioceptive sensors
  - Odometry: wheel sensors only
  - Dead reckoning: also heading sensors

- The movement of the robot, sensed with wheel encoders and/or heading sensors is integrated to the position.
  - Pros: Straight forward, easy
  - Cons: Errors are integrated -> unbound

- Using additional heading sensors (e.g. gyroscope) might help to reduce the cumulated errors, but the main problems remain the same.

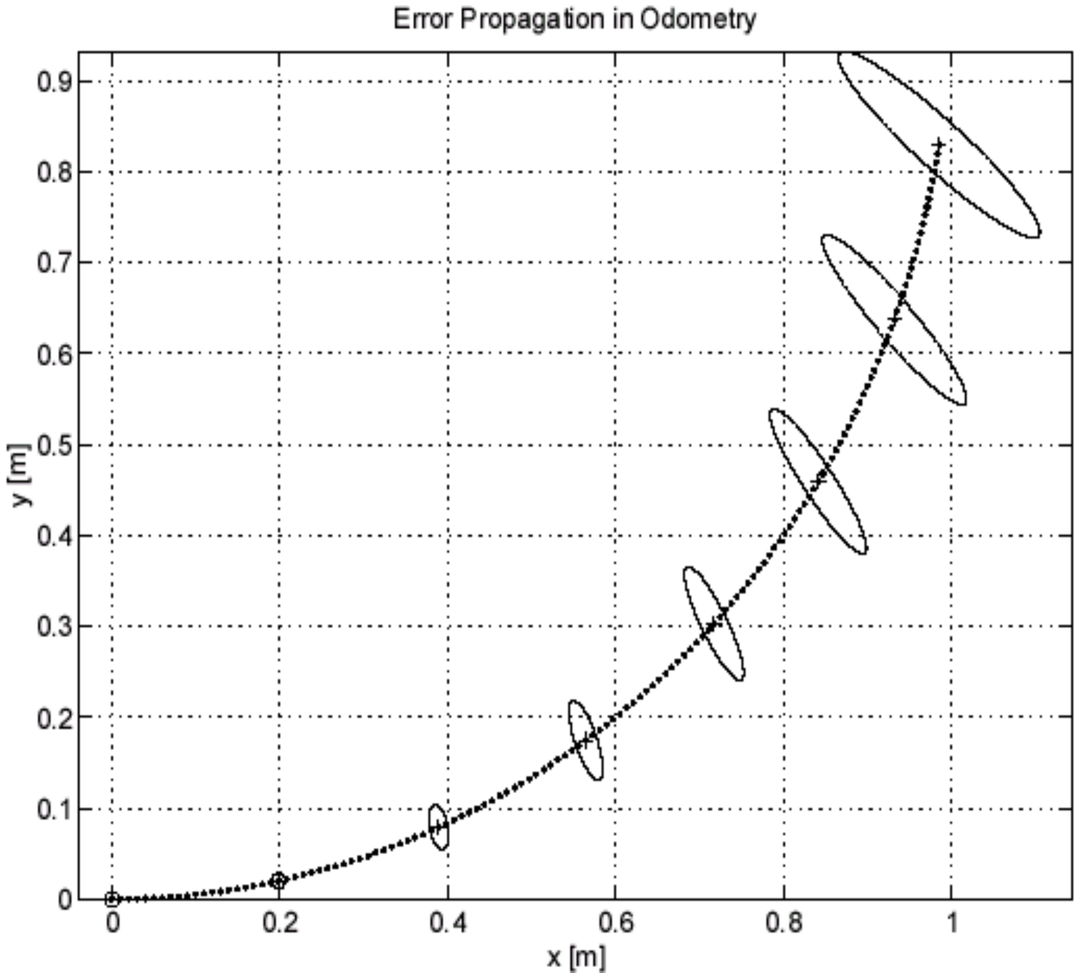
Odometry: Growth of Pose uncertainty for Straight Line Movement

- Note: Errors perpendicular to the direction of movement are growing much faster!



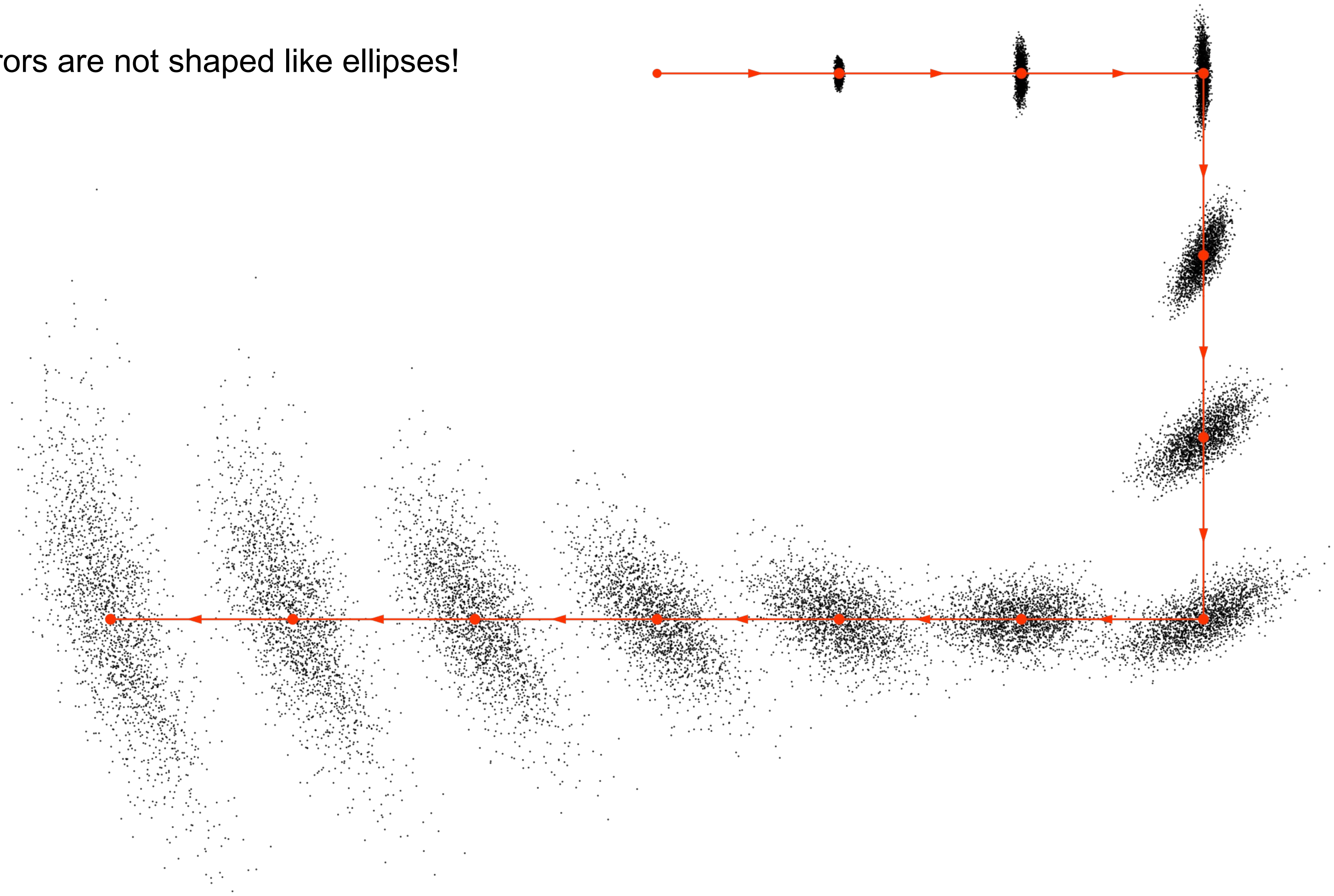
Odometry: Growth of Pose uncertainty for Movement on a Circle

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



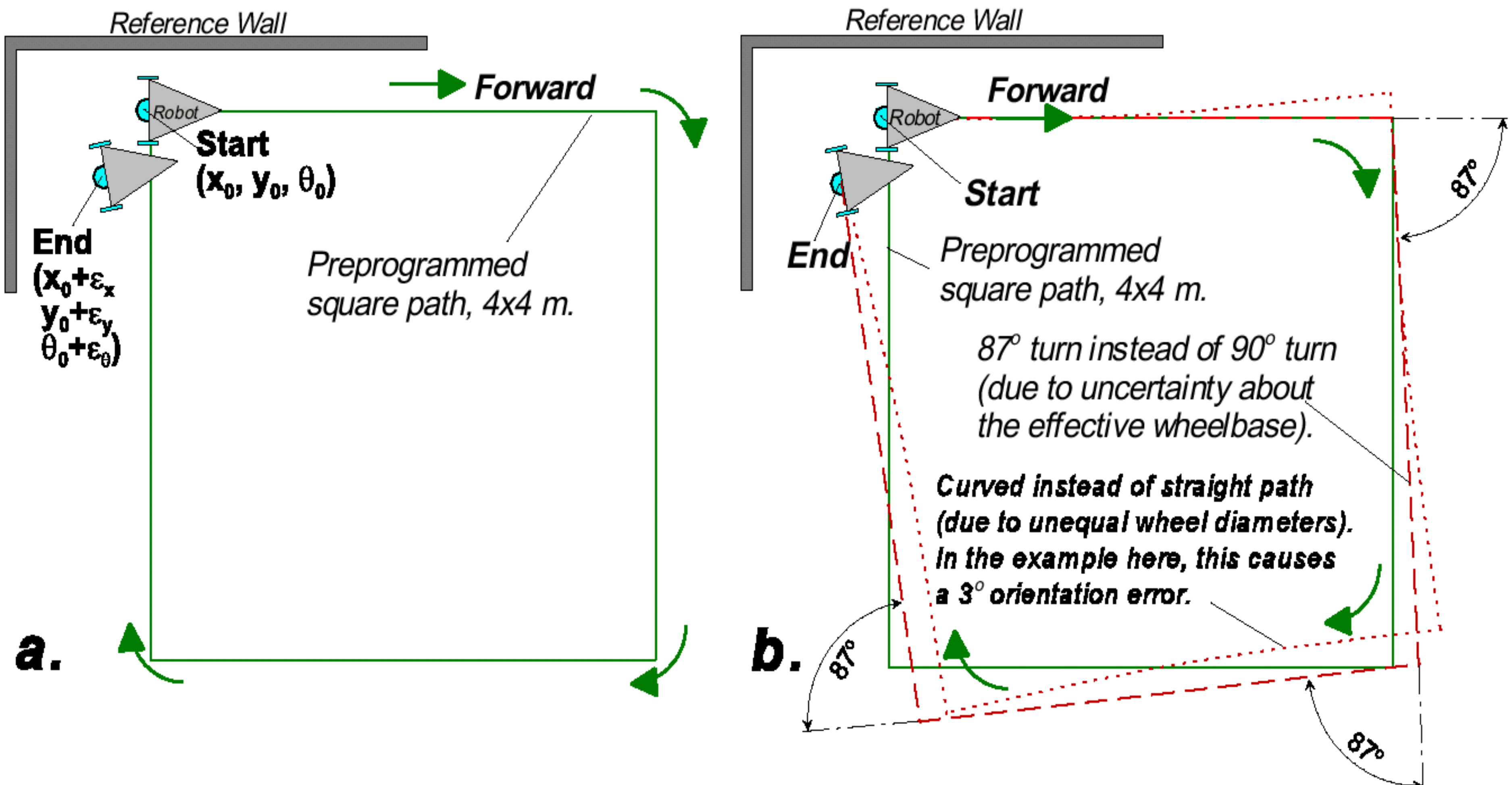
Odometry: example of non-Gaussian error model

- Note: Errors are not shaped like ellipses!



Odometry: Calibration of Errors

- The unidirectional square path experiment



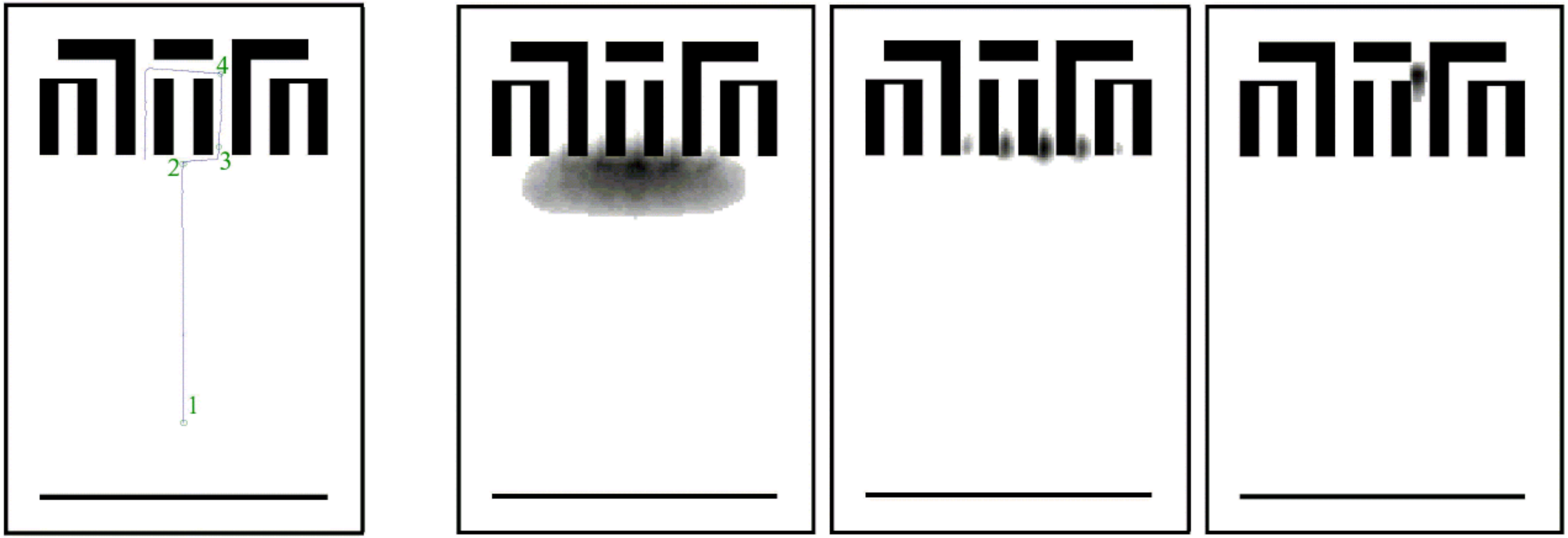
LOCALIZATION METHODS

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

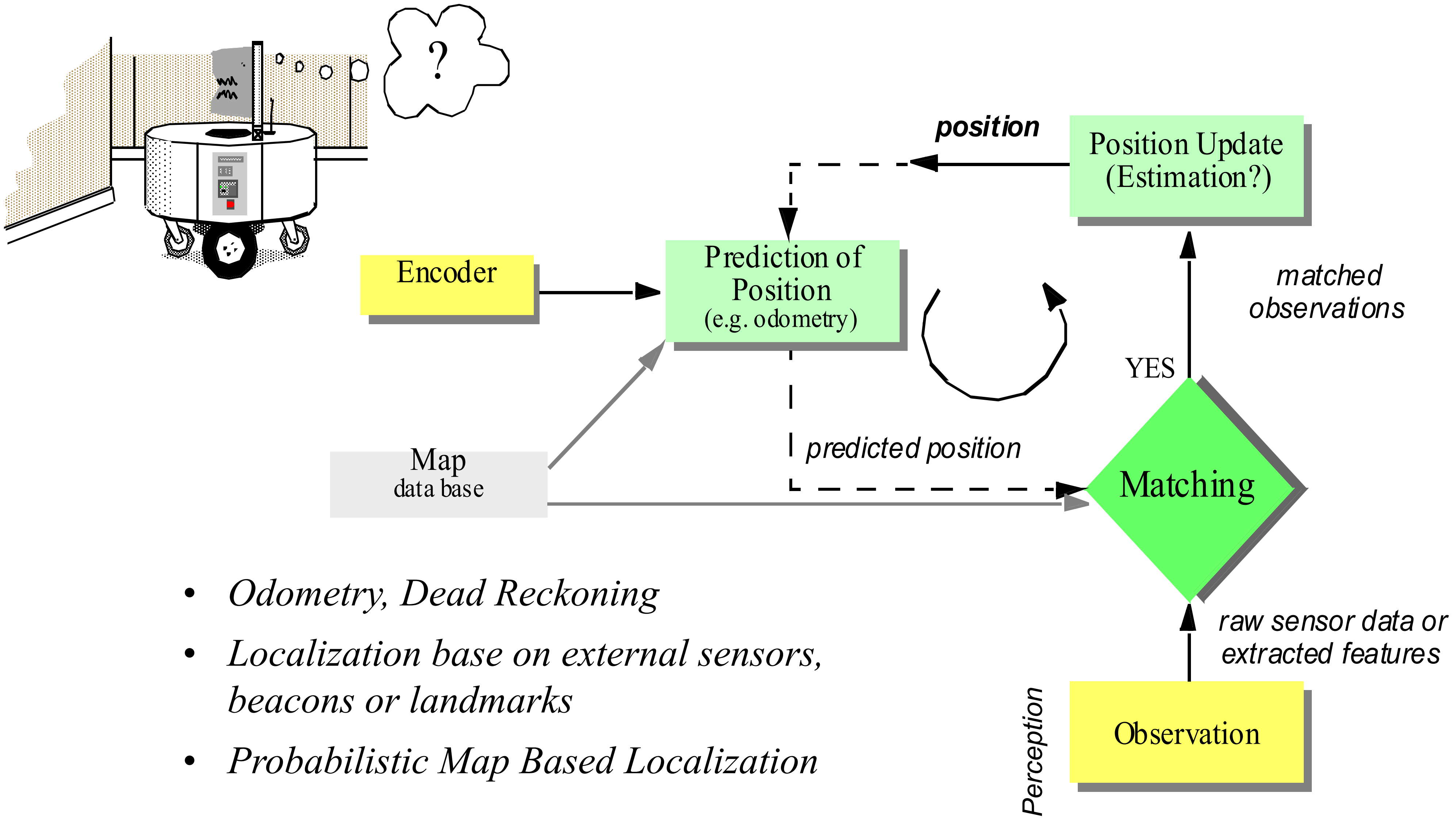


Path of the robot

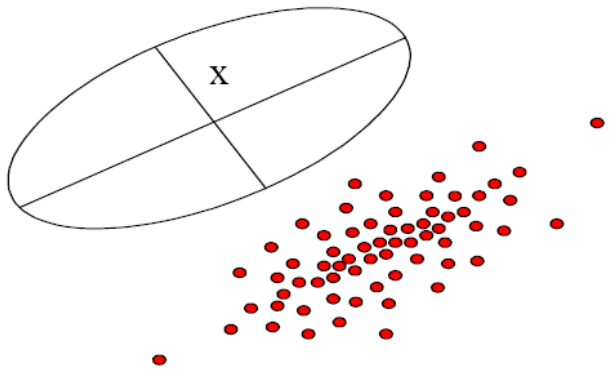
Belief states at positions 2, 3 and 4

Courtesy of W. Burgard

Map based localization



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



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

Monte Carlo Localization (MCL)

- 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…

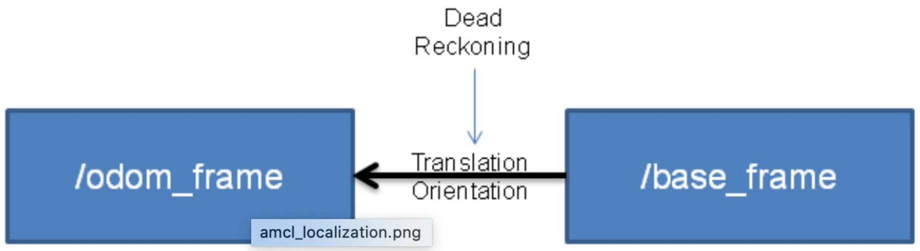
Adaptive Monte Carlo Localization (AMCL)

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

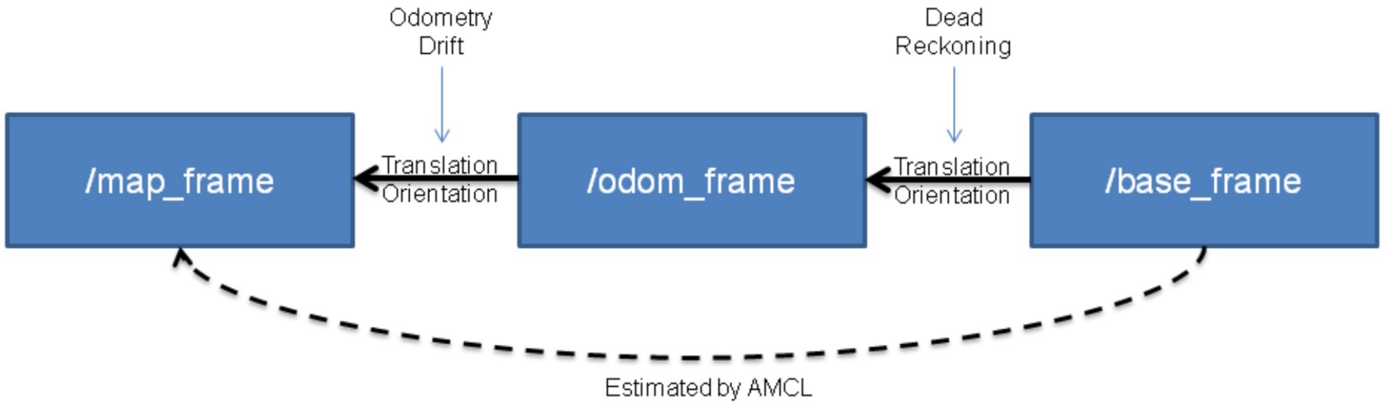
  - Sample size is re-calculated each iteration

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

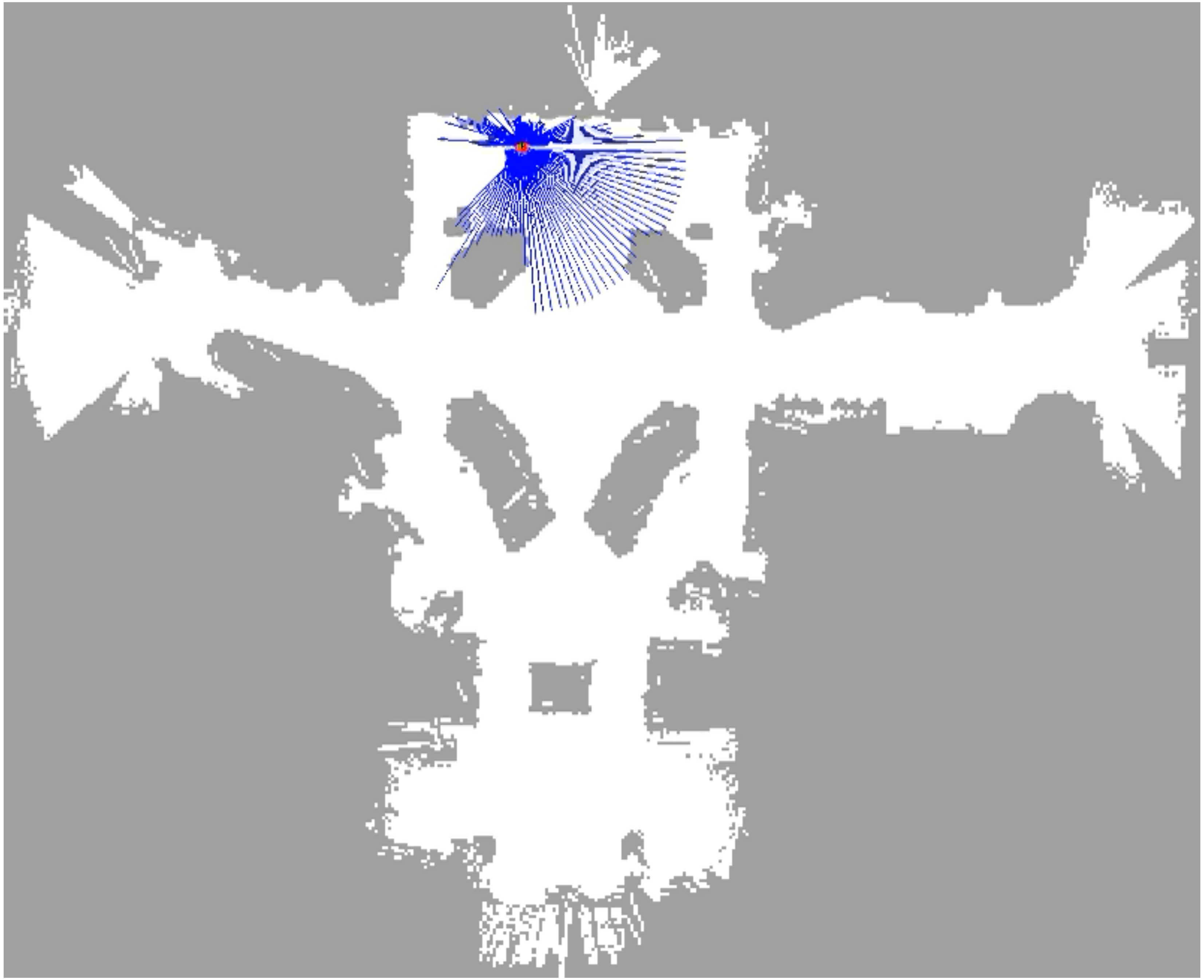
Odometry Localization



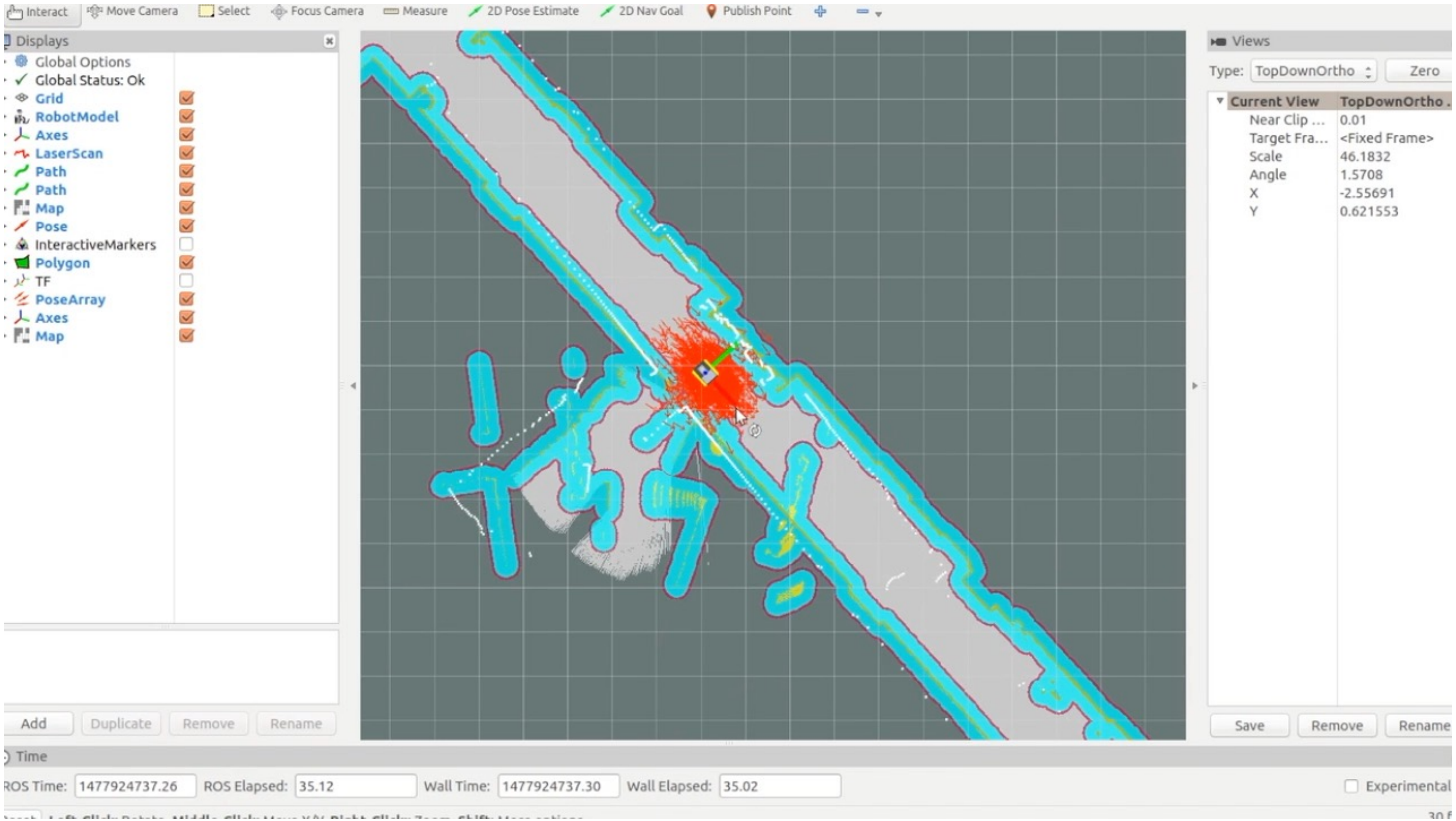
AMCL Map Localization



MCL & Robot Kidnapping



AMCL in ROS

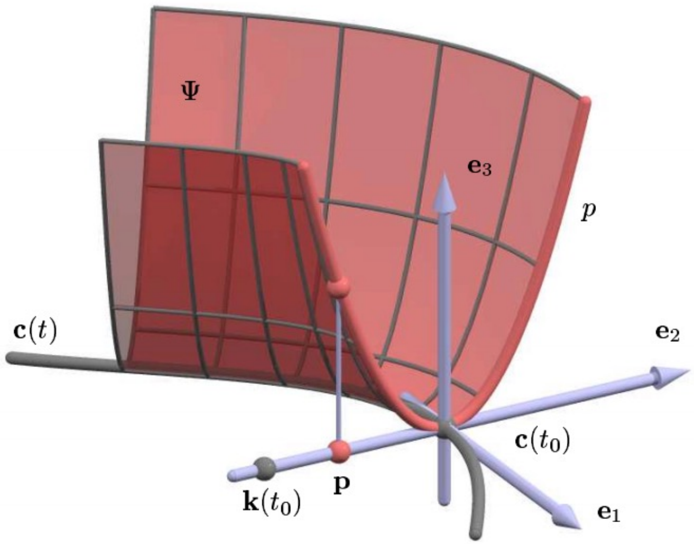

Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Global Status: Ok
Grid
RobotModel
Axes
LaserScan
Path
Path
Map
Pose
InteractiveMarkers
Polygon
TF
PoseArray
Axes
Map
Add
Duplicate
Remove
Rename
Views
Type: TopDownOrtho
Zero
Current View 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)

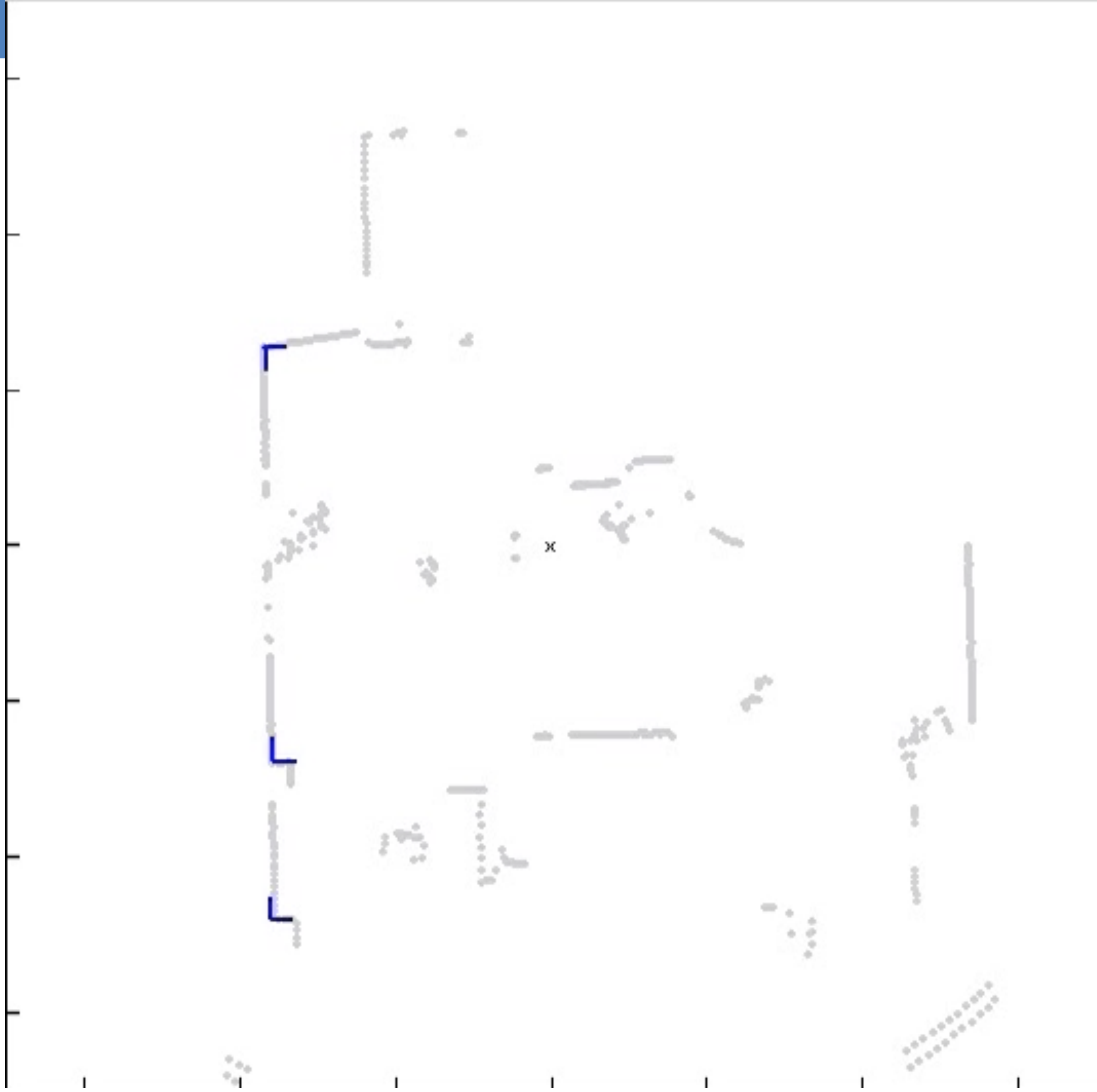


- Feature extracting methods (also for images)
  - Corners in point clouds
  - Lines
  - Planes
  - Feature Descriptors/ also via Deep Learning

- Spectral methods (also for images)

SLAM using corner structures

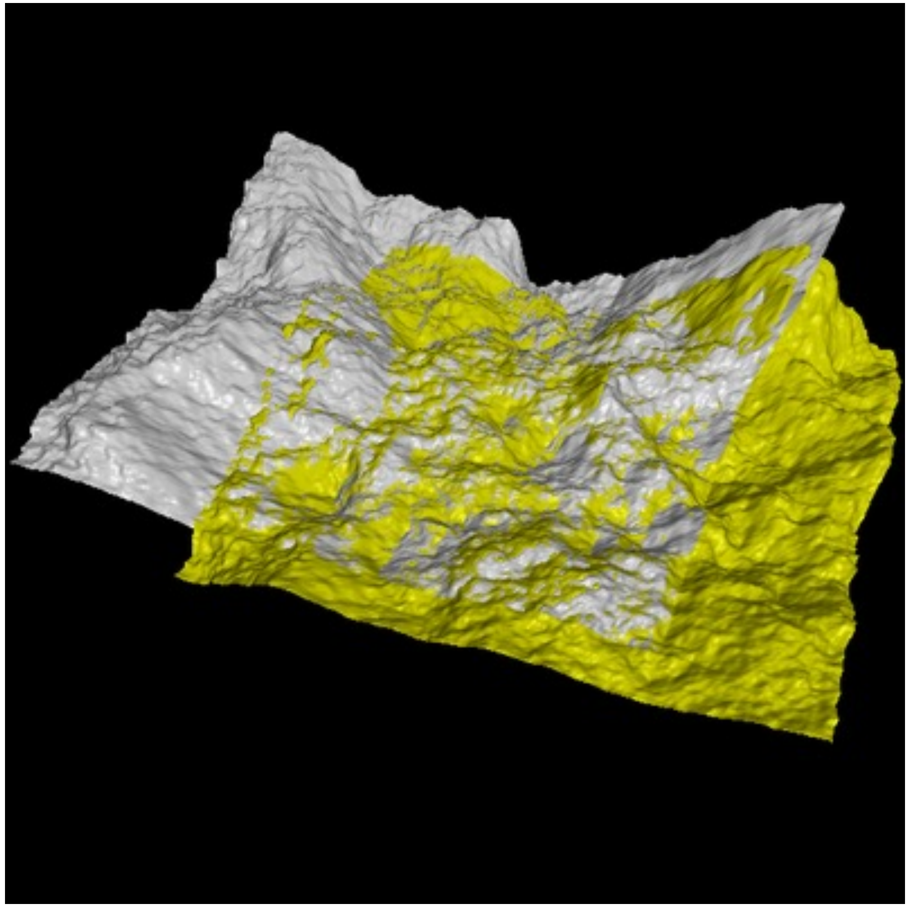
- 2D LRF Scan
- Detect corners in the scan
- Map corners, localization against corners



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)



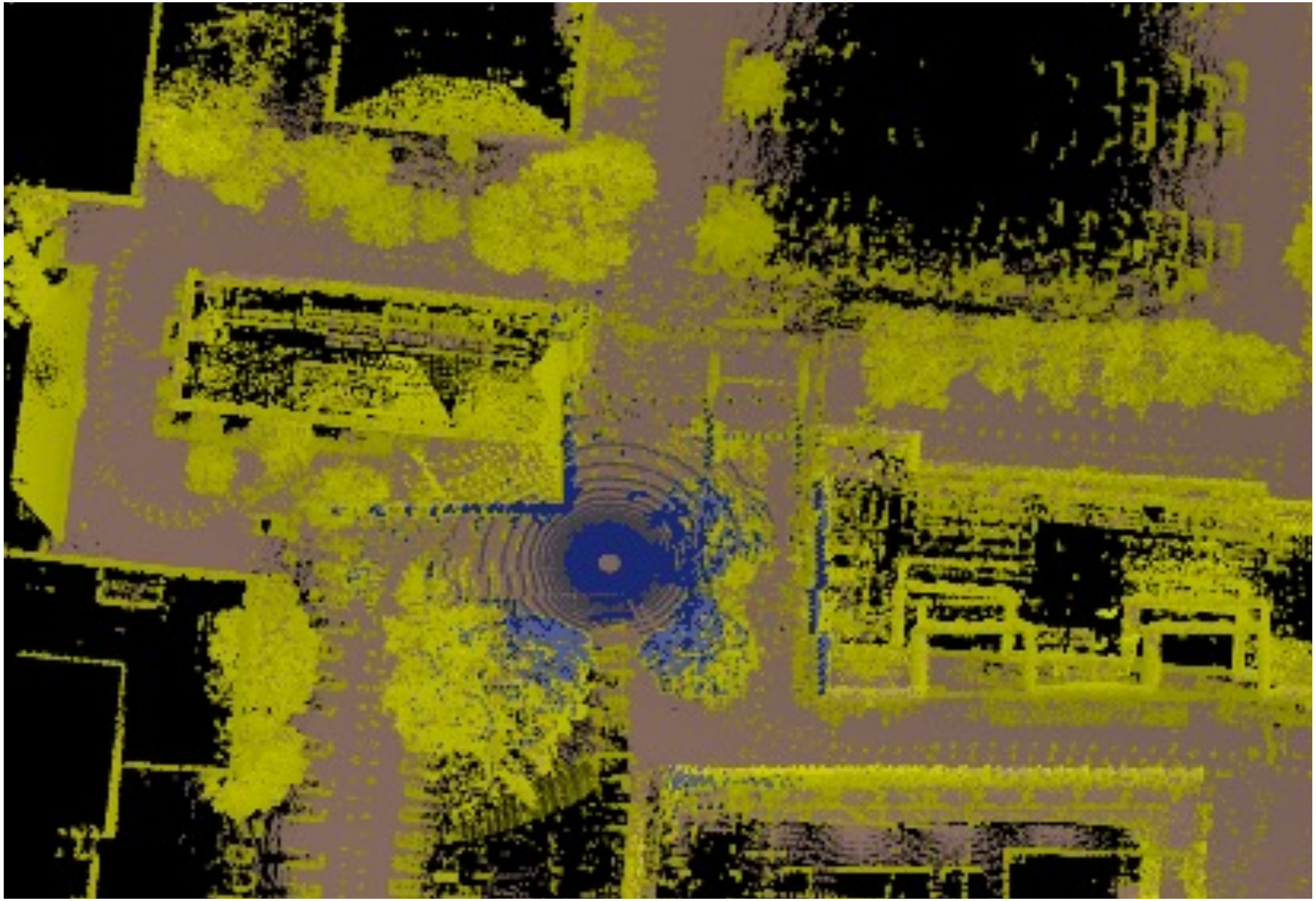
Material derived from Ronen Gvili : *www.cs.tau.ac.il/~dcor/Graphics/adv-slides/**ICP**.ppt*

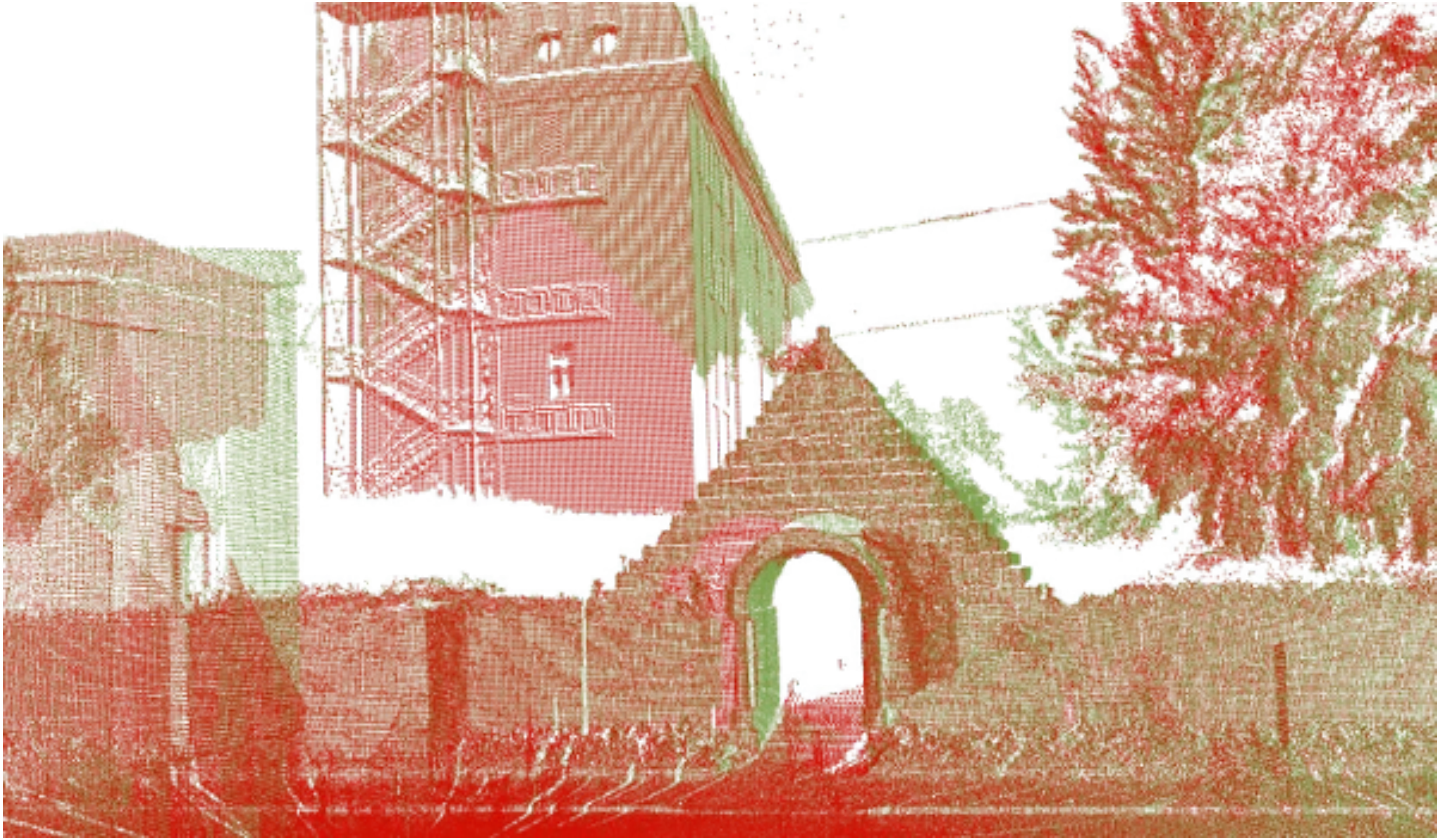
Data Types

- 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

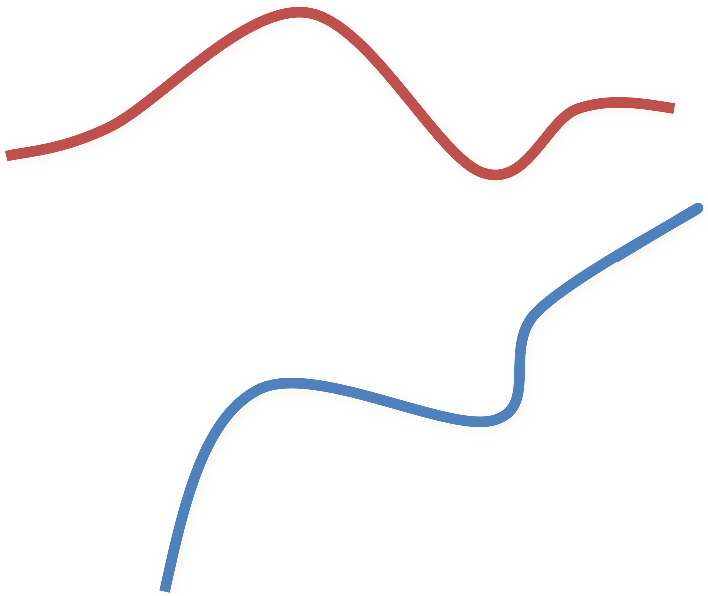
- Scan Matching - Registration
- Shape inspection
- Motion estimation
- Appearance analysis
- Texture Mapping
- Tracking





Aligning 3D Data

- Continuous lines or a set of points…

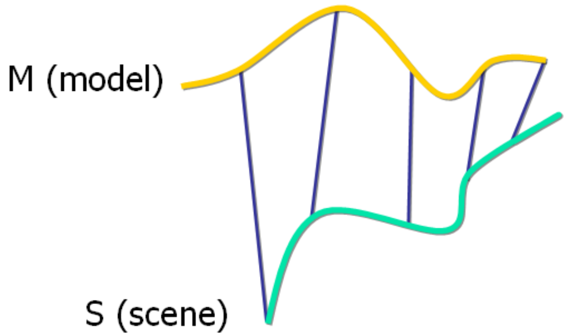


Corresponding Point Set Alignment

- 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 :

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

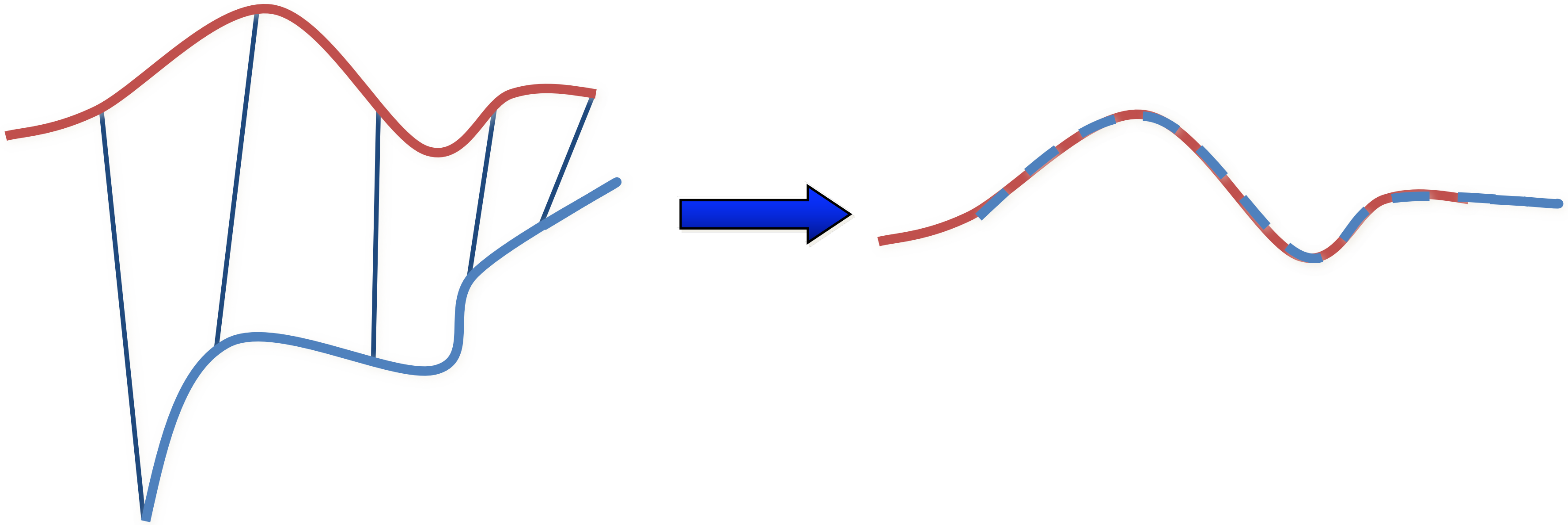
$$f(q) = \frac{1}{N_S} \sum_{i=1}^{N_S} \left\| m_i - R(q_R) s_i - q_T \right\|^2$$

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.

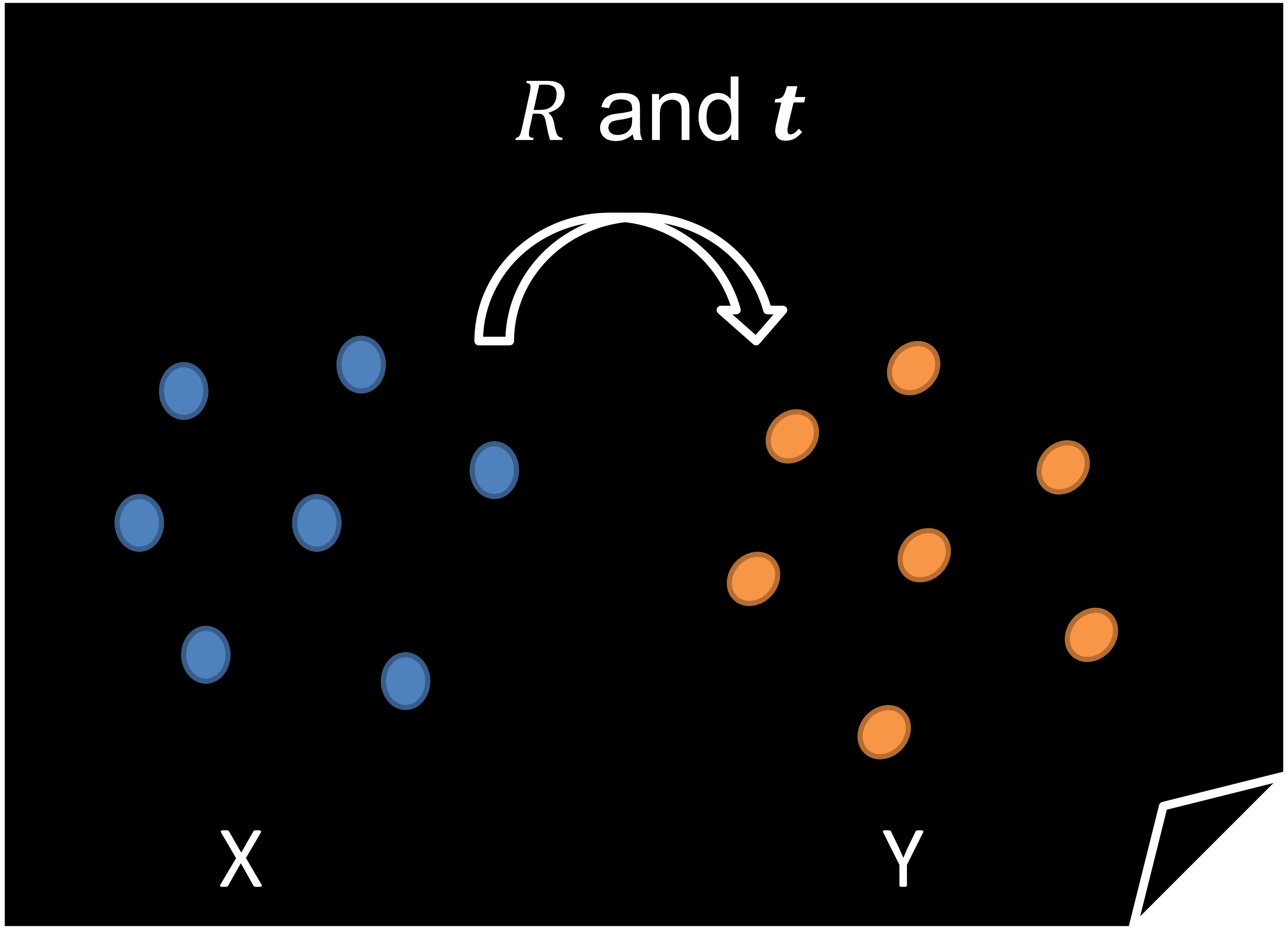


See:
A. Lorusso, D. Eggert, and R. Fisher.
A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations.
In *Proceedings of the 4th British Machine Vision Conference (BMVC '95)*, pages 237 - 246, Birmingham, England, September 1995.

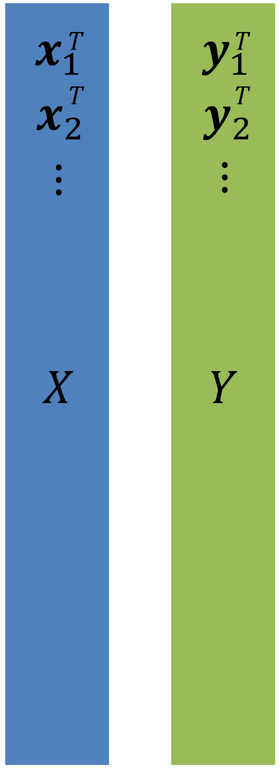
Horn's method

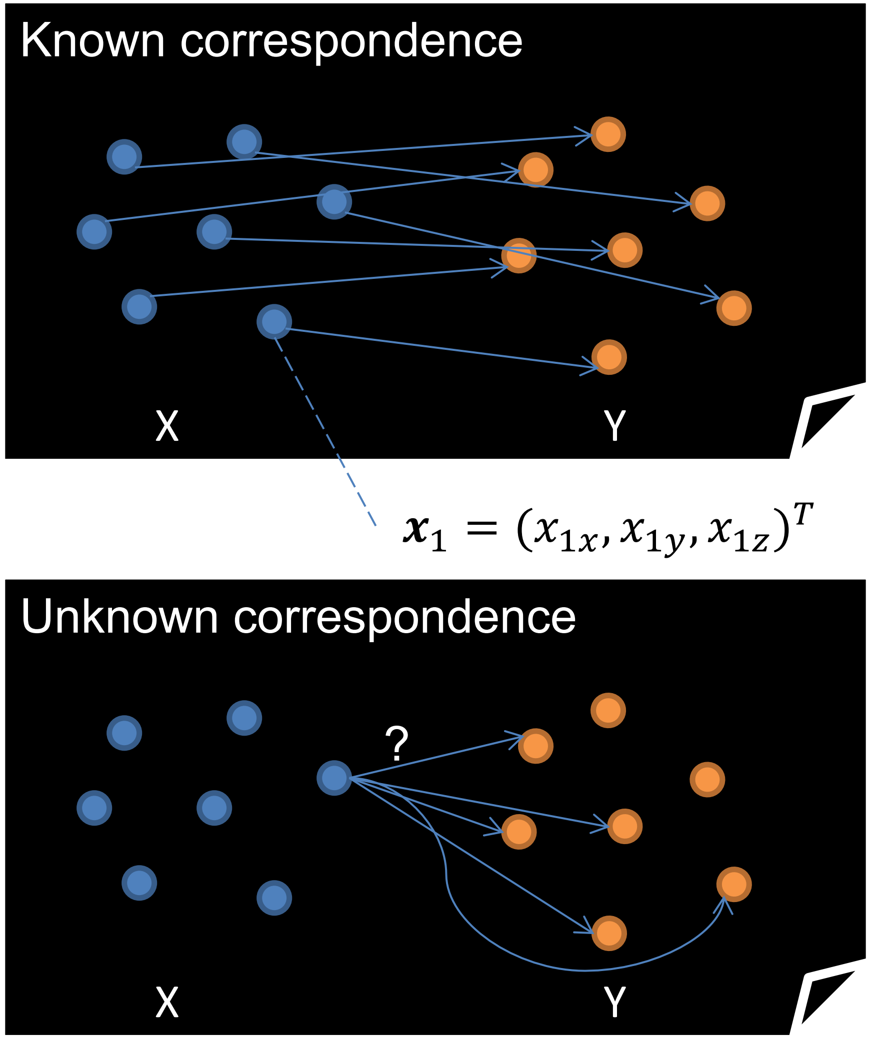
Material by Toru Tamaki, Miho Abe, Bisser Raytchev, Kazufumi Kaneda

- 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.

1

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

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

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

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