上海科技大学
ShanghaiTech University

CS283: **Robotics Fall 2020: Localization**

Sören Schwertfeger / 师泽仁

ShanghaiTech University

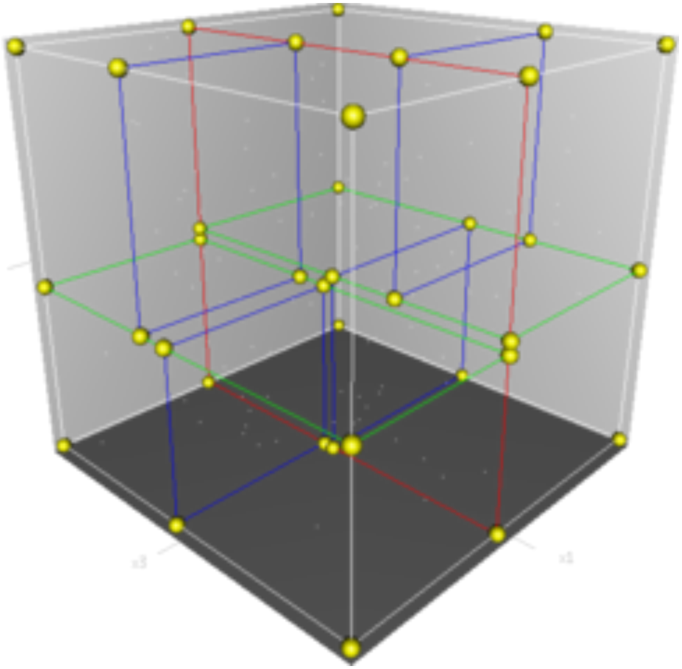
Map Representation: what is “saved” in the map

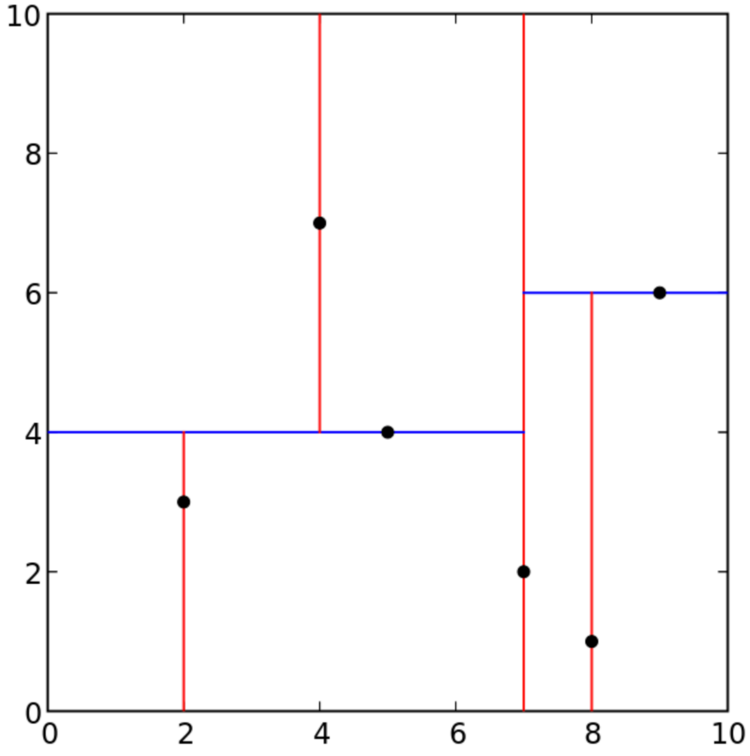
- Points (surface of objects, buildings): 2D or 3D
  - What: x,y or x,y,z coordinates; Optional: intensity; maybe RGB; maybe descriptor; temperature; …
  - From range sensors (laser, ultrasound, stereo, RGB-D): dense
  - From cameras (structure from motion; feature points): sparse
  - Variant: kd-tree
- 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.:
  - Camus, 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

k-d tree

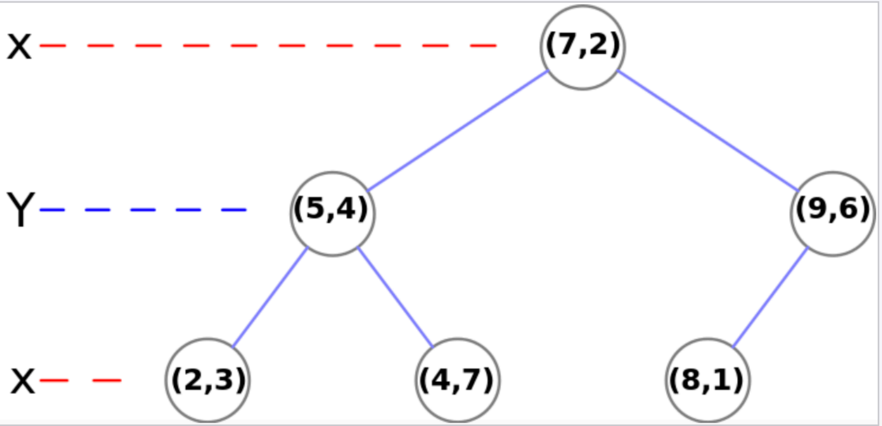
- k-dimensional binary search tree
- Robotics: typically 3D or 2D
- Every level of tree:
  - For a different axis (e.g. x,y,z,x,y,z,x,y,z) (=> split space with planes)
  - Put points in left or right side based on median point (w.r.t. its value of on the current axis) =>
  - Balanced tree
- Fast neighbor search -> ICP!

Algorithm	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(n)$
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$





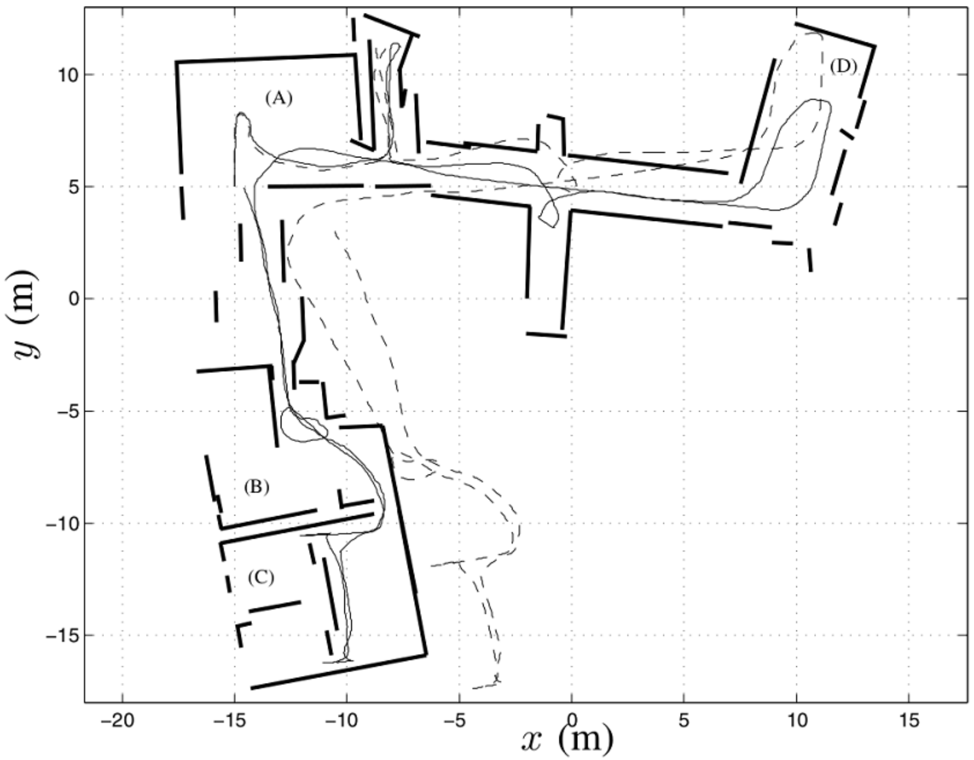
k-d tree decomposition for the point set (2,3), (5,4), (9,6), (4,7), (8,1), (7,2).



The resulting *k*-d tree.

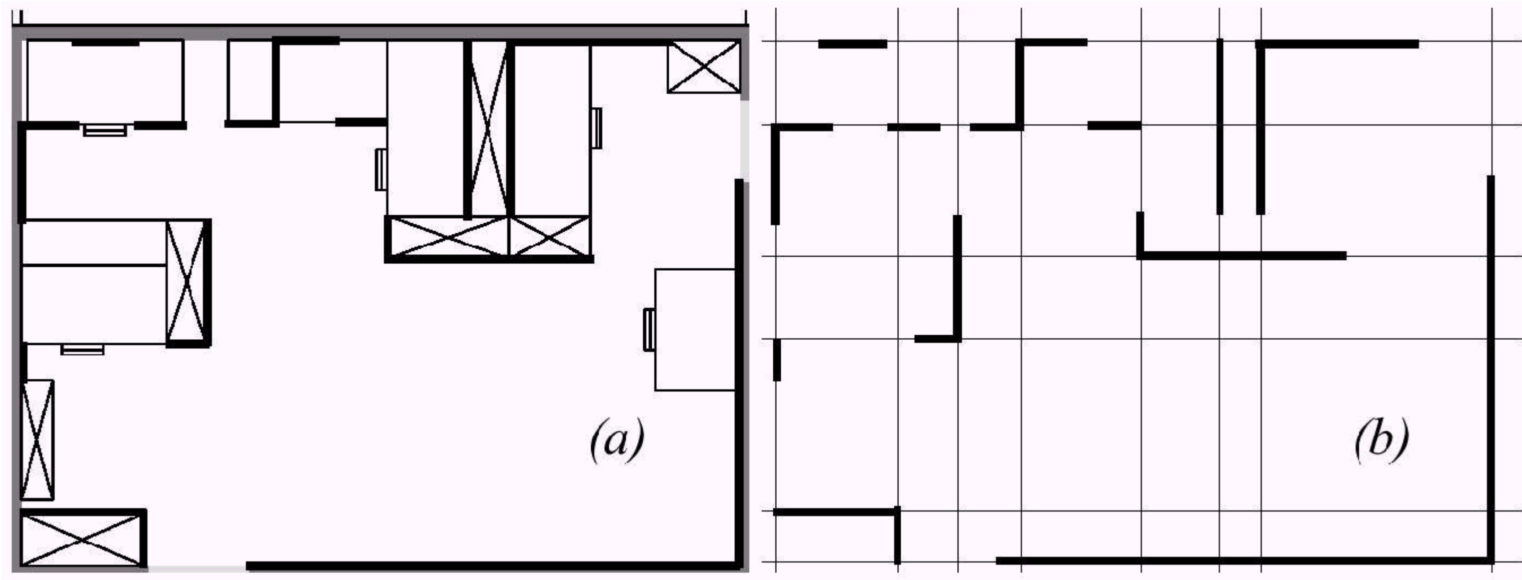
Line Map

- Abstract from points =>
- Extract features (e.g. lines, planes)
- E.g. using RANSAC, Hough Transform
- E.g. region growing, e.g. via normals

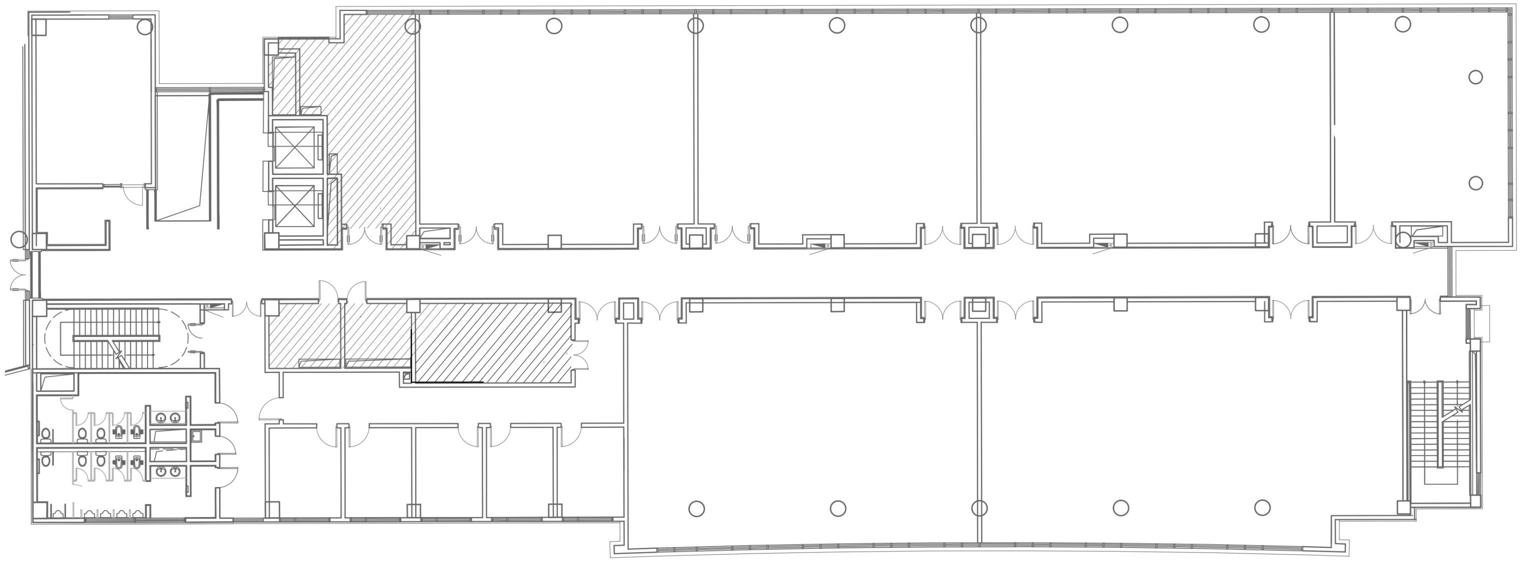


- Finite lines (a)
- Infinite lines (b)

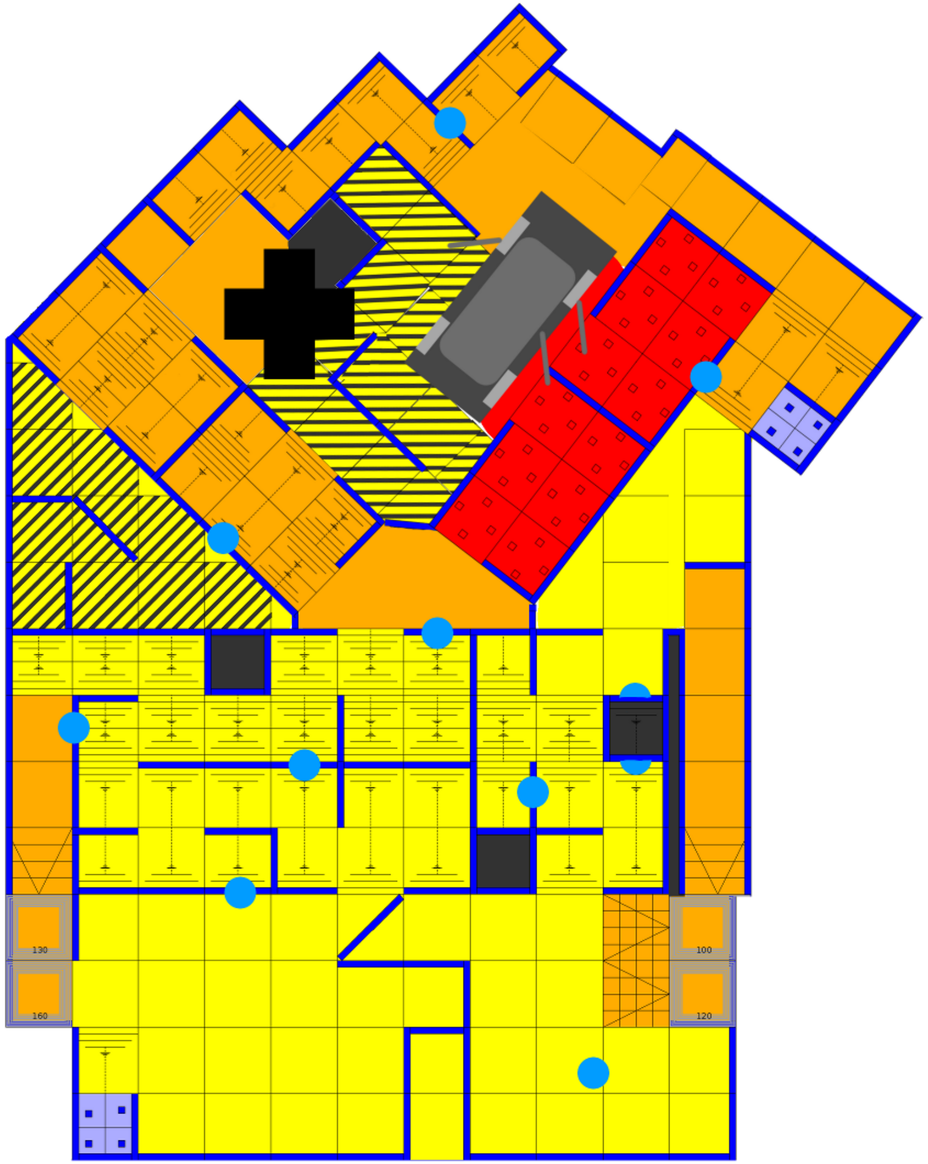
- Very compact
- Can do scan-matching (e.g. against laser scan)



Ground Truth Maps

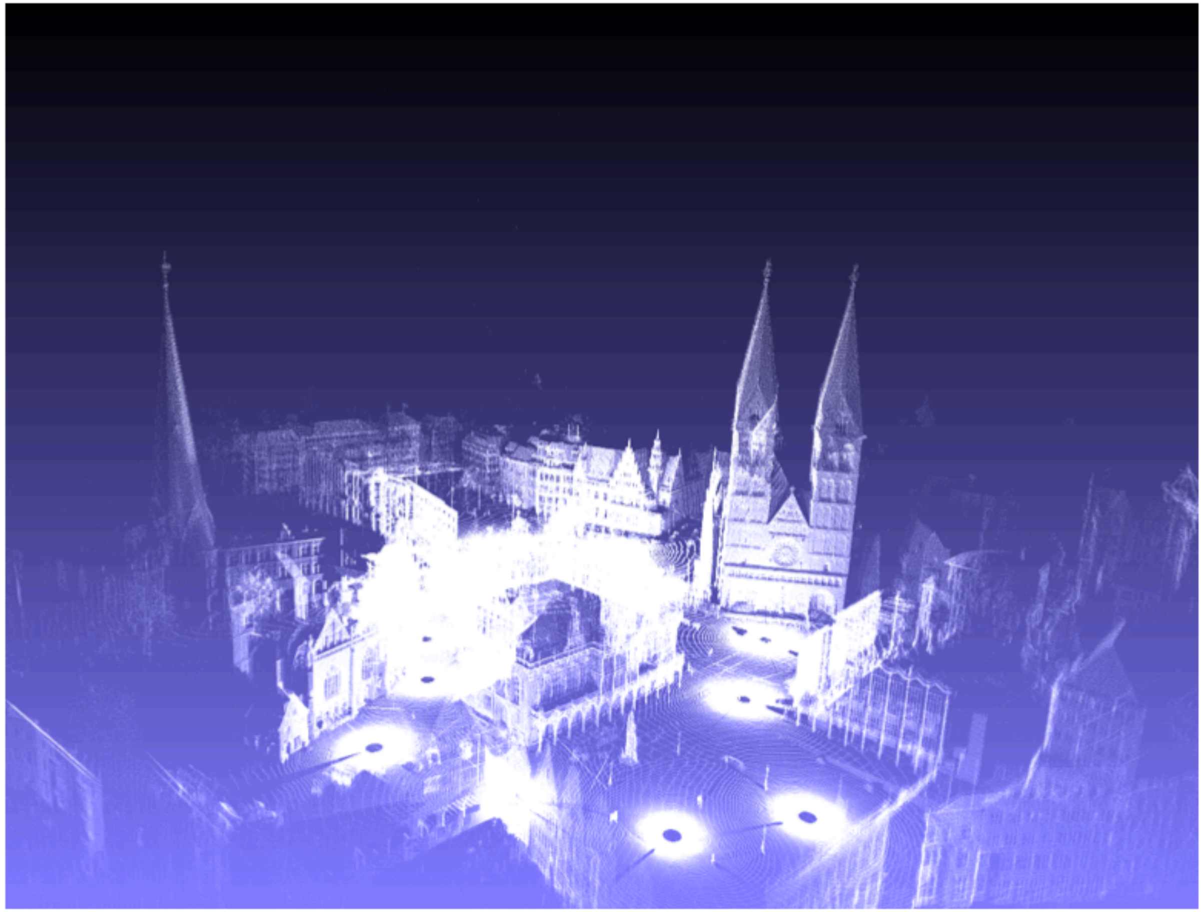


CAD drawing (STAR Center)
Vector format (lines, circles, …)

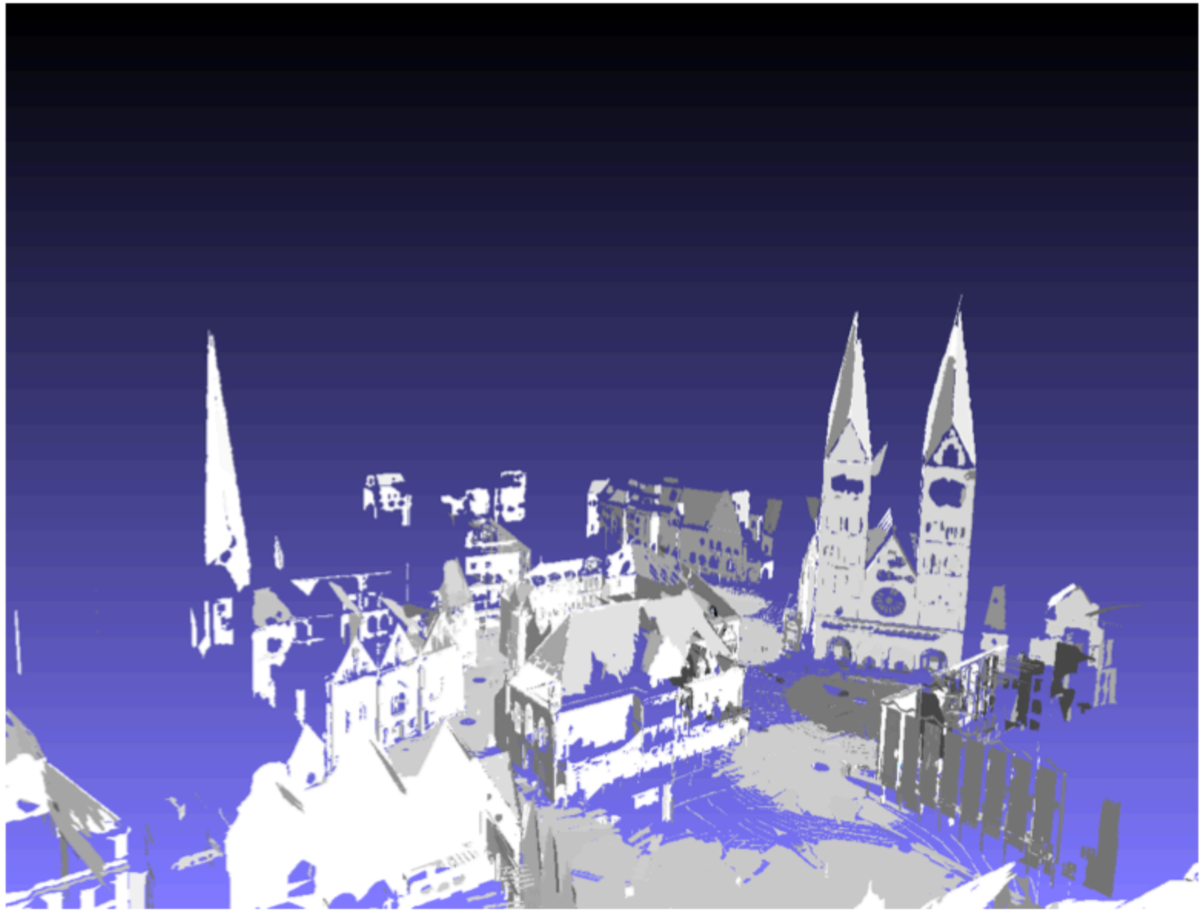


Grid Map (from vector map)

Bremen: 3D Point Cloud & 3D Plane Map

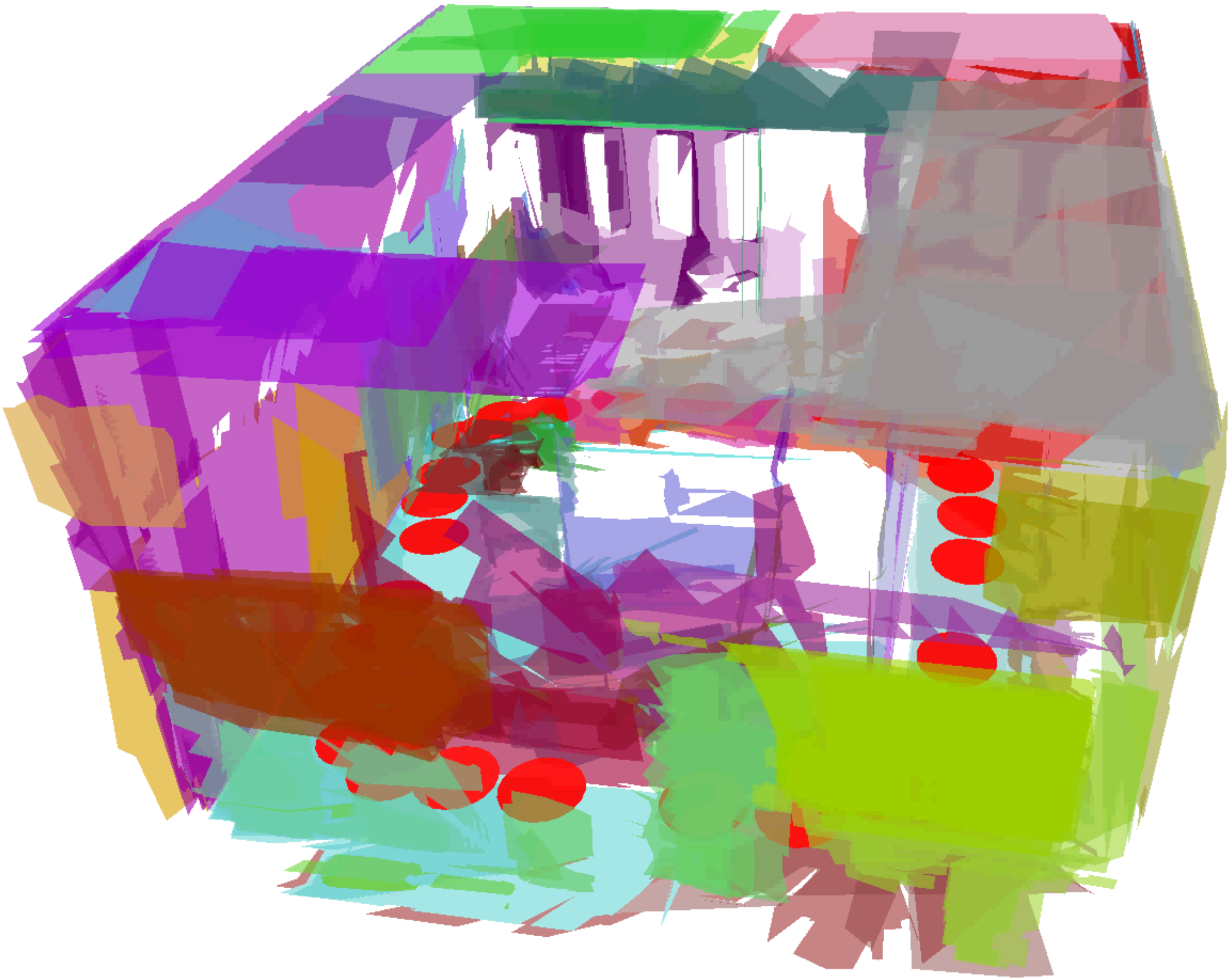


(a) 3D Point Cloud



(b) 3D Plane Map

Plane map: 29 poses; each with several planes



Vector Maps

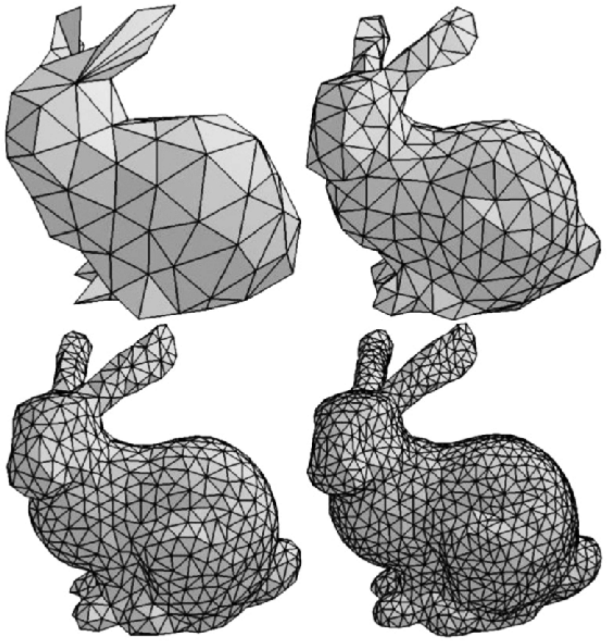
- E.g. Open Street Maps
- Represented as OSM files (xml) or PBF (binary)

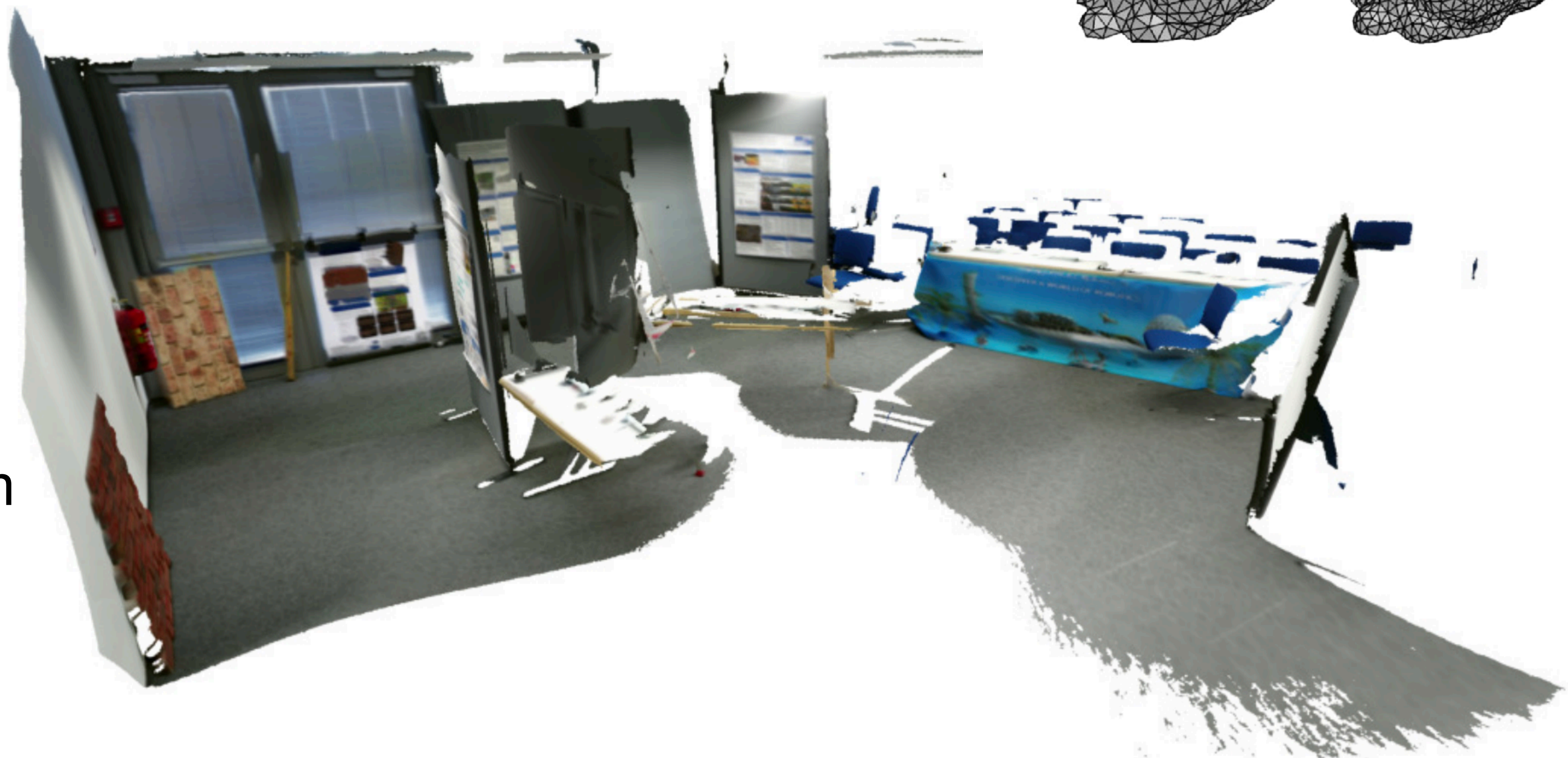
- Nodes in WGS 84 (vertices)
  - Only entity with position
  - Just for ways or
  - Object (e.g. sign)

- Ways:
  - Open polygon (street)
  - Closed polygons
  - Areas
  - With tags (e.g. name, type, …)

Mesh

- Often build via Signed Distance Field
- Close relation to 3D reconstruction from Computer Vision
- Often with texture (RGB information from camera)



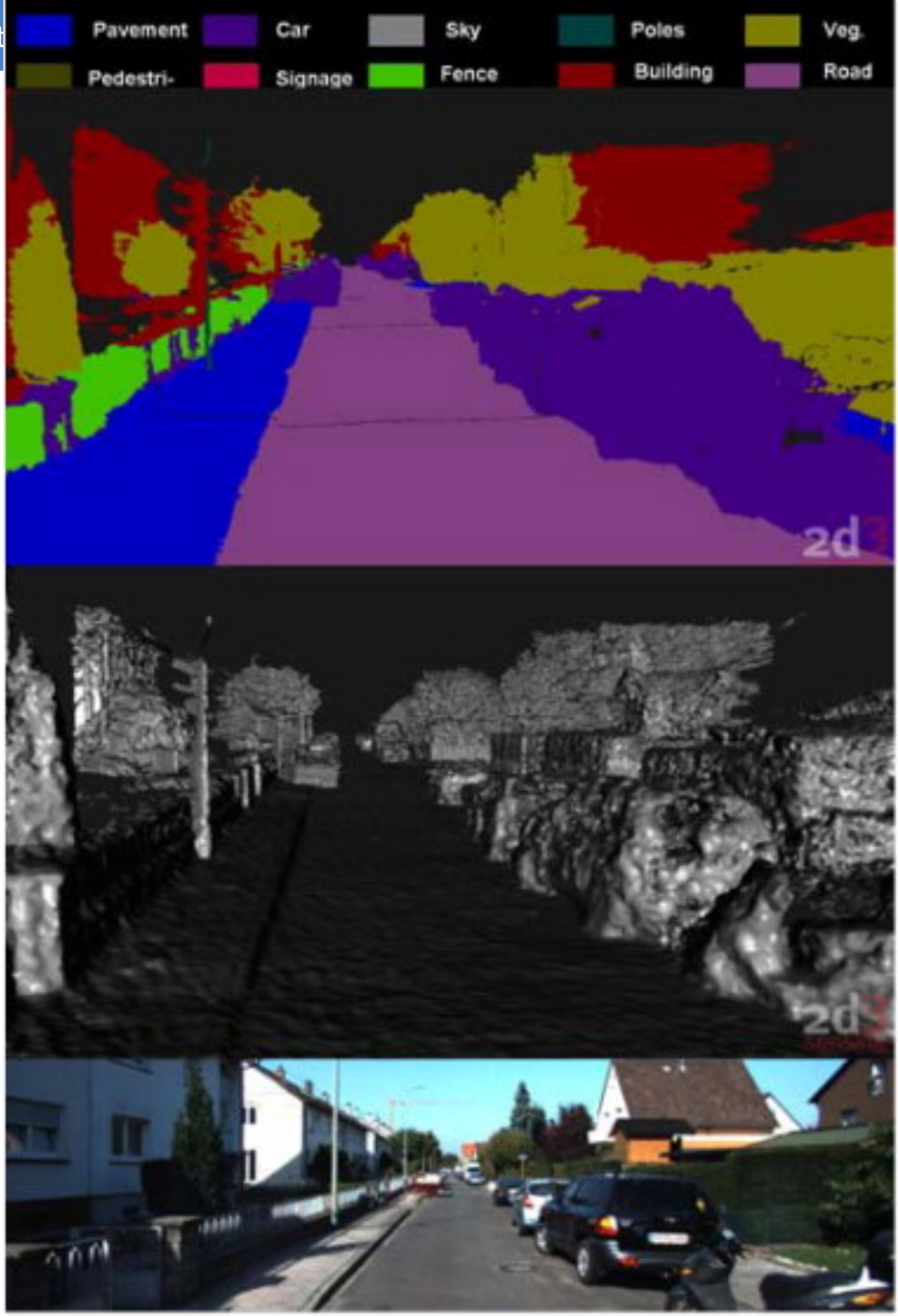


Stereye: Mesh Simplification

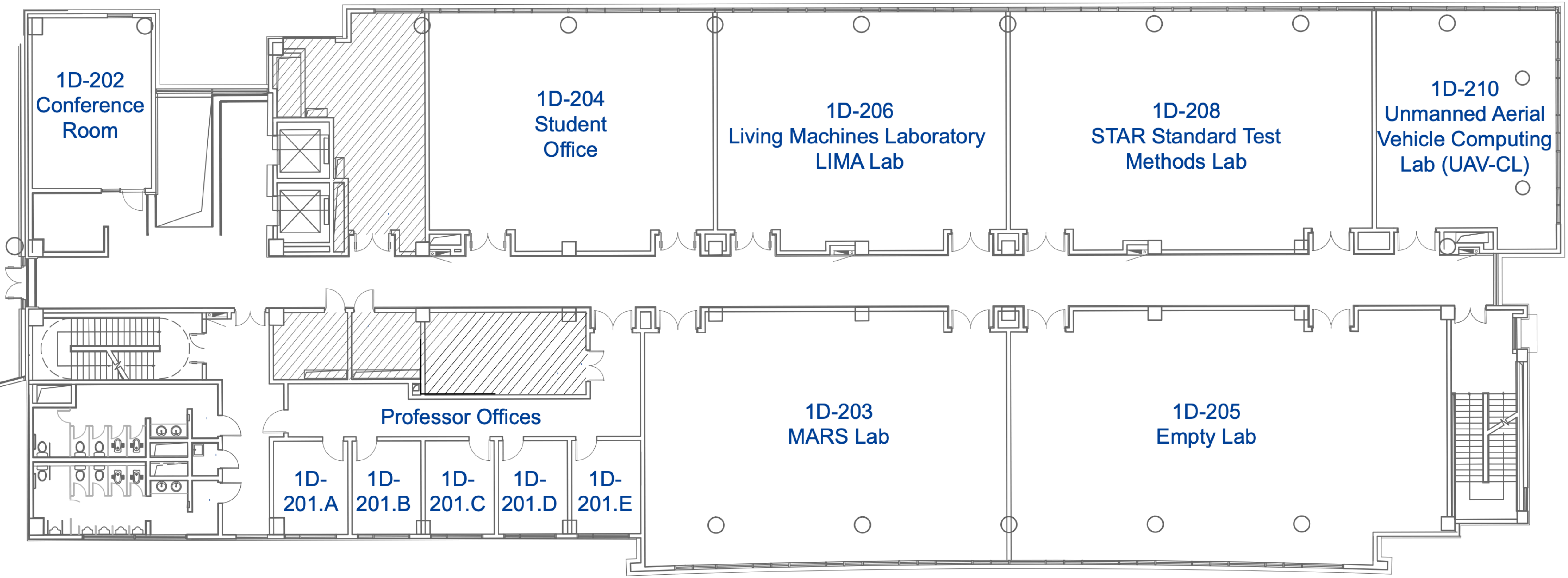
- Plane recognition via plane growing (add points with similar normals)
- Plane contour via alpha shape algorithm
- Planar intersections to make the model tight
- Mesh via Ear Clipping algorithm

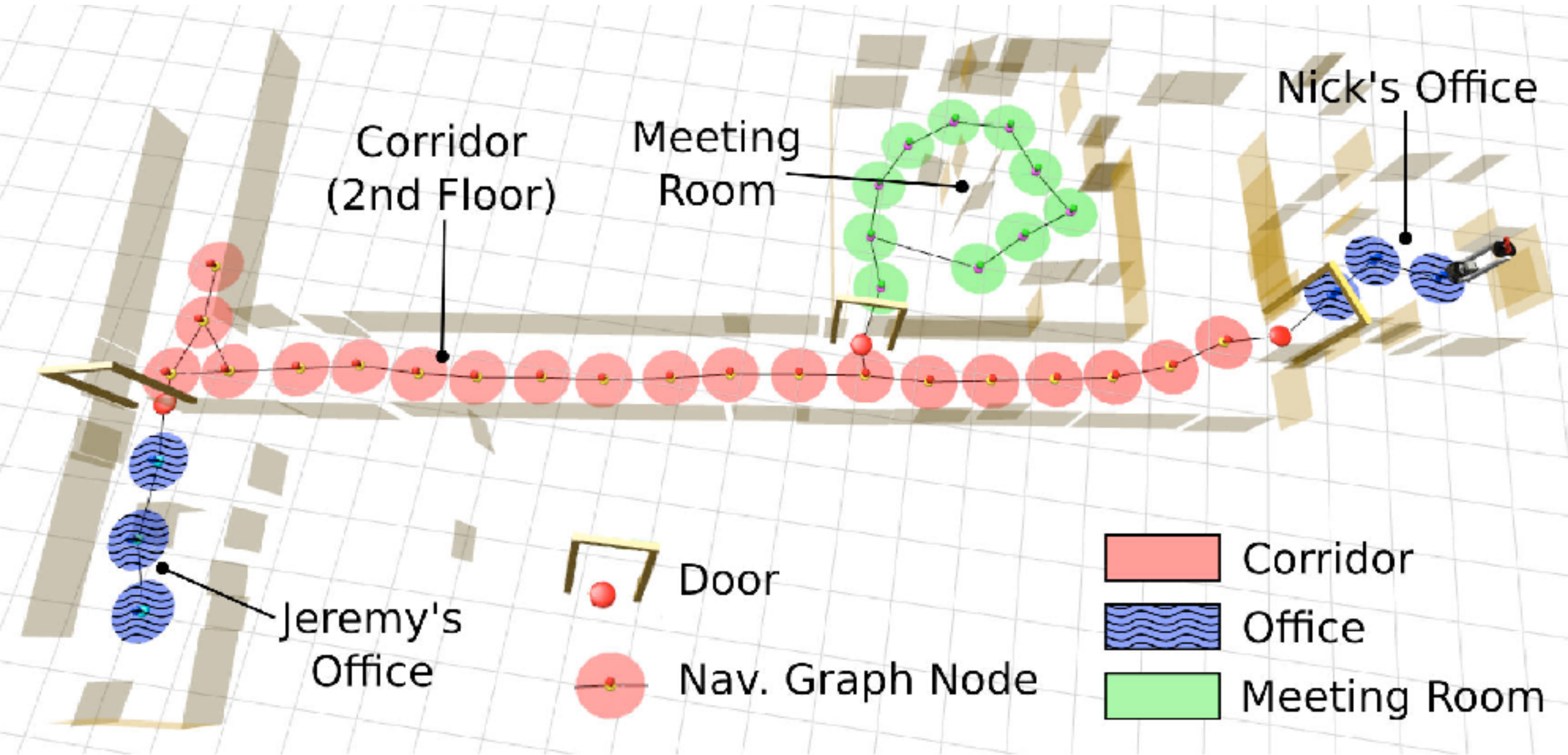
Semantic Map

- Semantic Segmentation
  - In room (e.g. detect furniture)
  - Outdoors



Semantic Annotation: Room Names




Corridor
(2nd Floor)
Meeting
Room
Nick's Office
Jeremy's
Office
Door
Nav. Graph Node
Corridor
Office
Meeting Room

Semantic Information

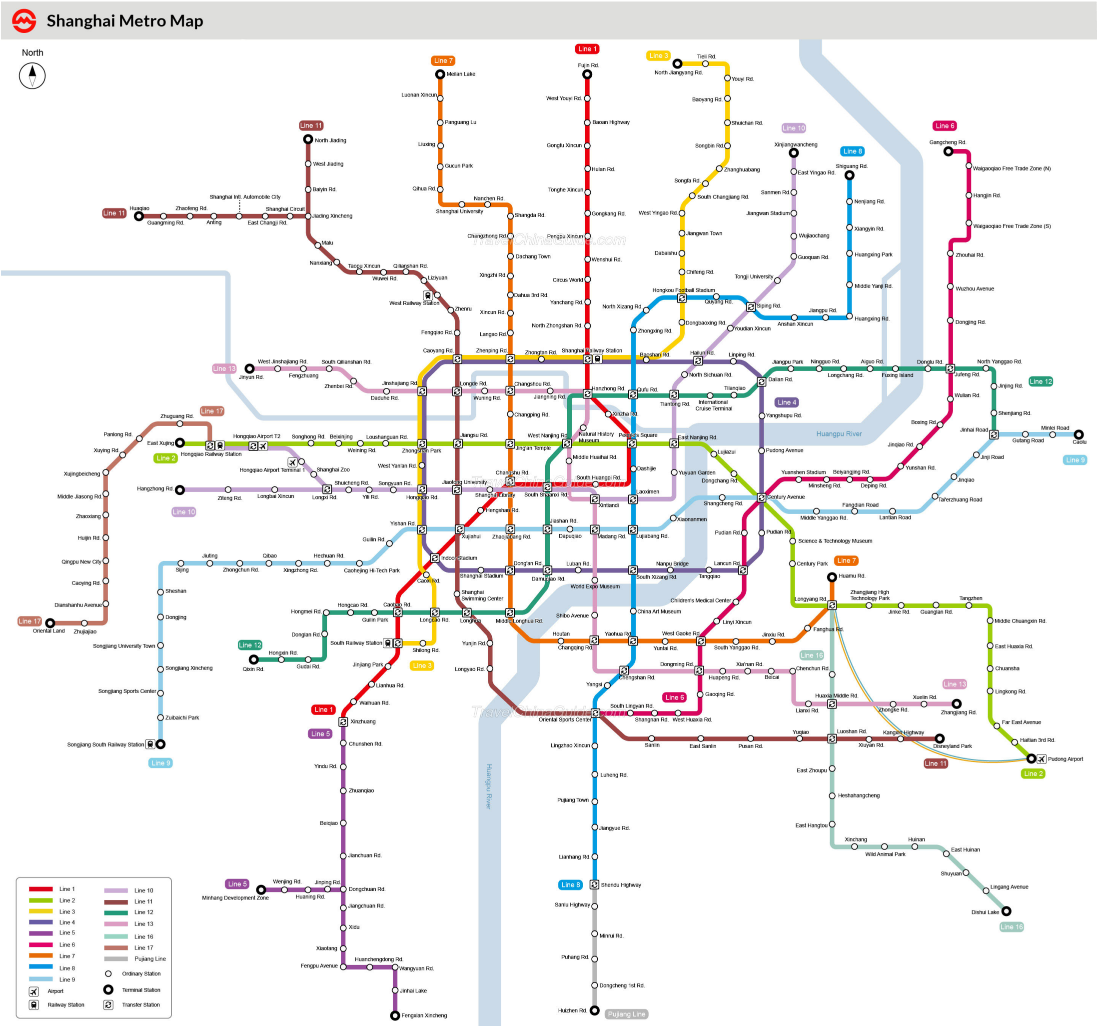
- Assign labels to data
- Segmentation: automatically group data (e.g. points) according to their semantic class

- Even save just very high level data; e.g. room at (x,y); Eiffel tower; …

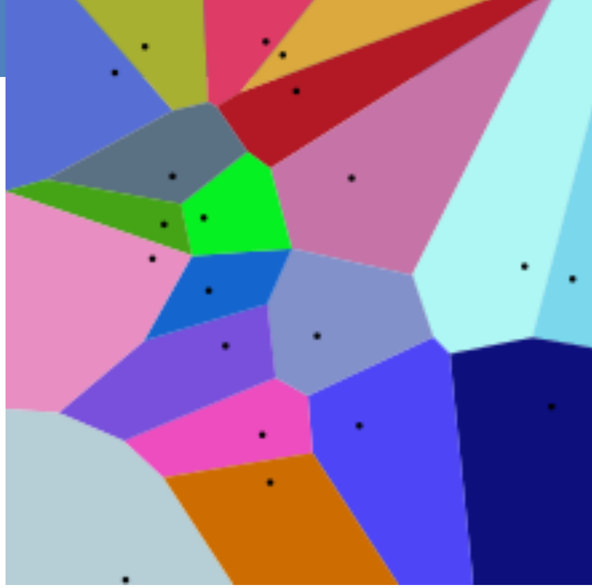
- Applications:
  - Human Robot Interaction (“go to kitchen”)
  - Scene understanding
  - Navigation (detect road; detect door)
  - Localization
  - …

Topologic Map

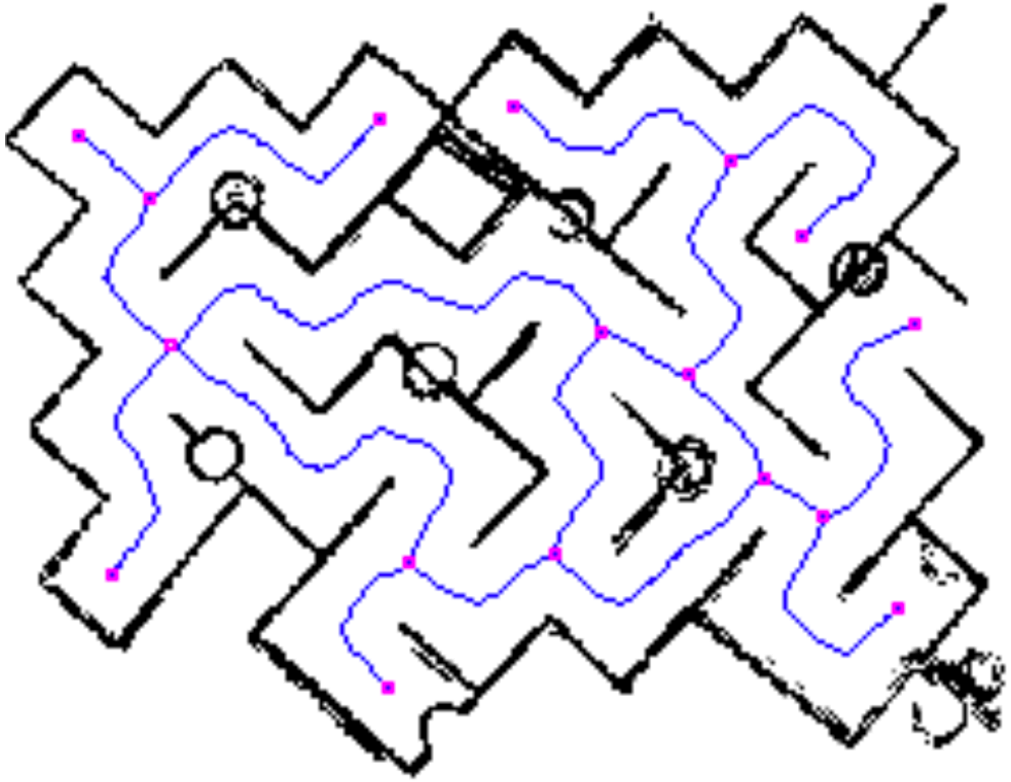
- A (undirected) graph
- Places (vertices) and their connections (edges)
- E.g. subway map of Shanghai: stations (vertices) and lines (edges)
- Do not have coordinates
- Topometric map: vertices and/ or edges are attributed with coordinates
- Very abstract – e.g. no obstacles anymore



Voronoi Diagram (-> Topology Graph)

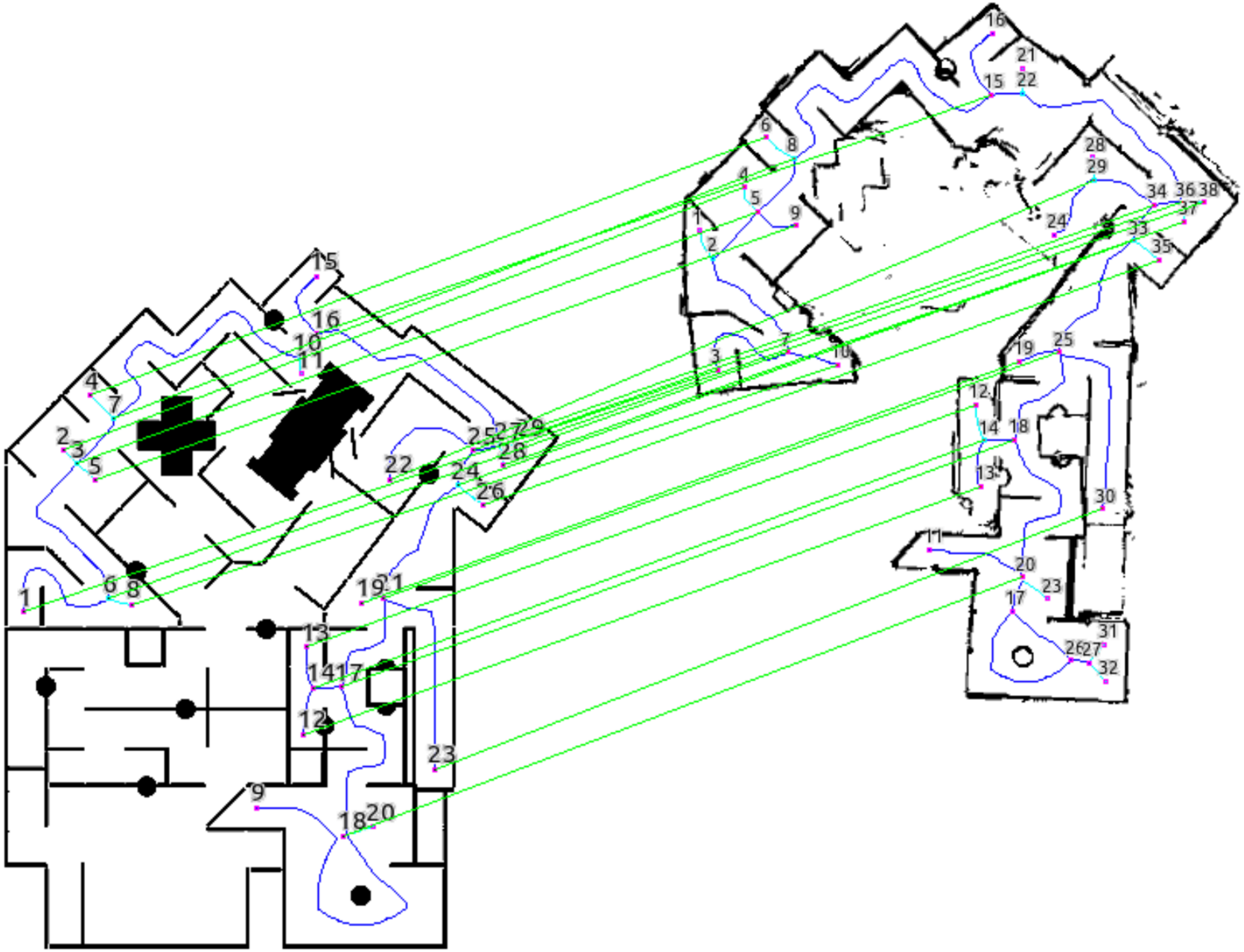


- Voronoi Diagram (VD): partition space such that edge is always equidistant to 2 closest obstacles.
- Topology Graph: vertices at junctions and dead ends
- Applications:
  - Very fast path planning
  - Human robot interaction (follow corridor, then go left)
  - Map matching

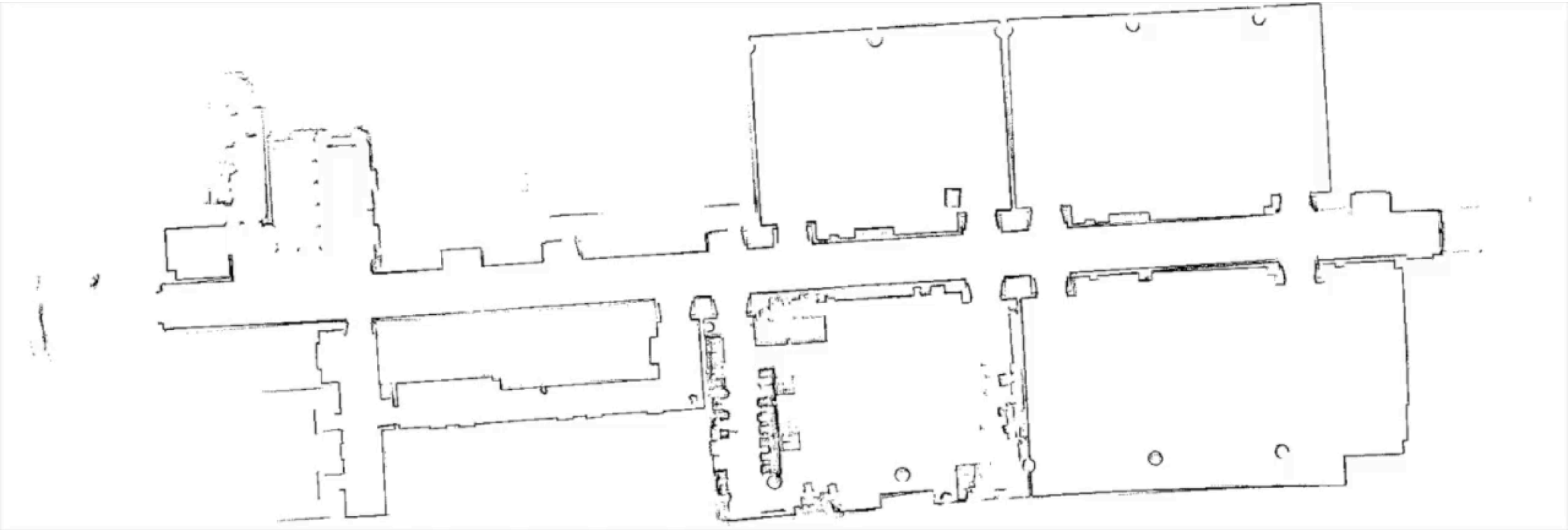


Map Matching

- Of 2D grid maps based on Topology Graph
- Left: ground truth map
- Right: Robot generated map
- RoboCup Rescue environment!

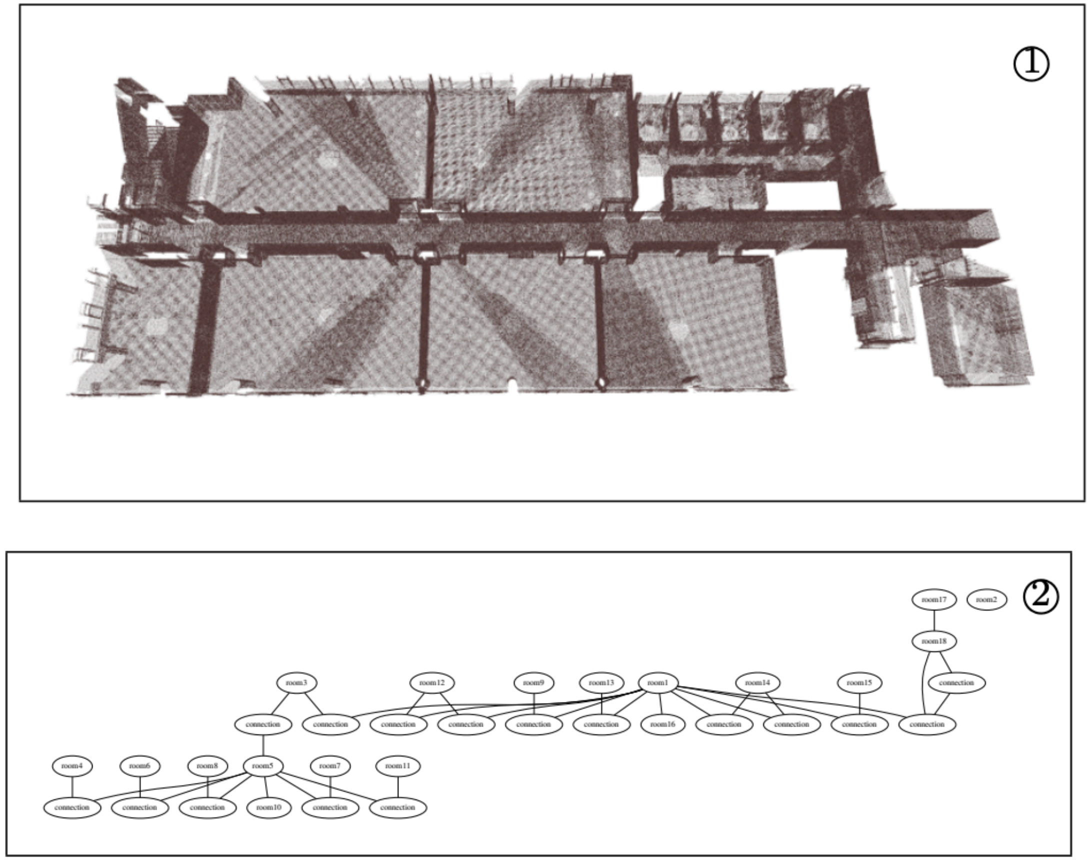


Area Graph: from 2D Grid Map to Topology Graph



Topological Map in different Dimensions

- (2): 0D; (3): 1D; (4): 2D; (5): 3D



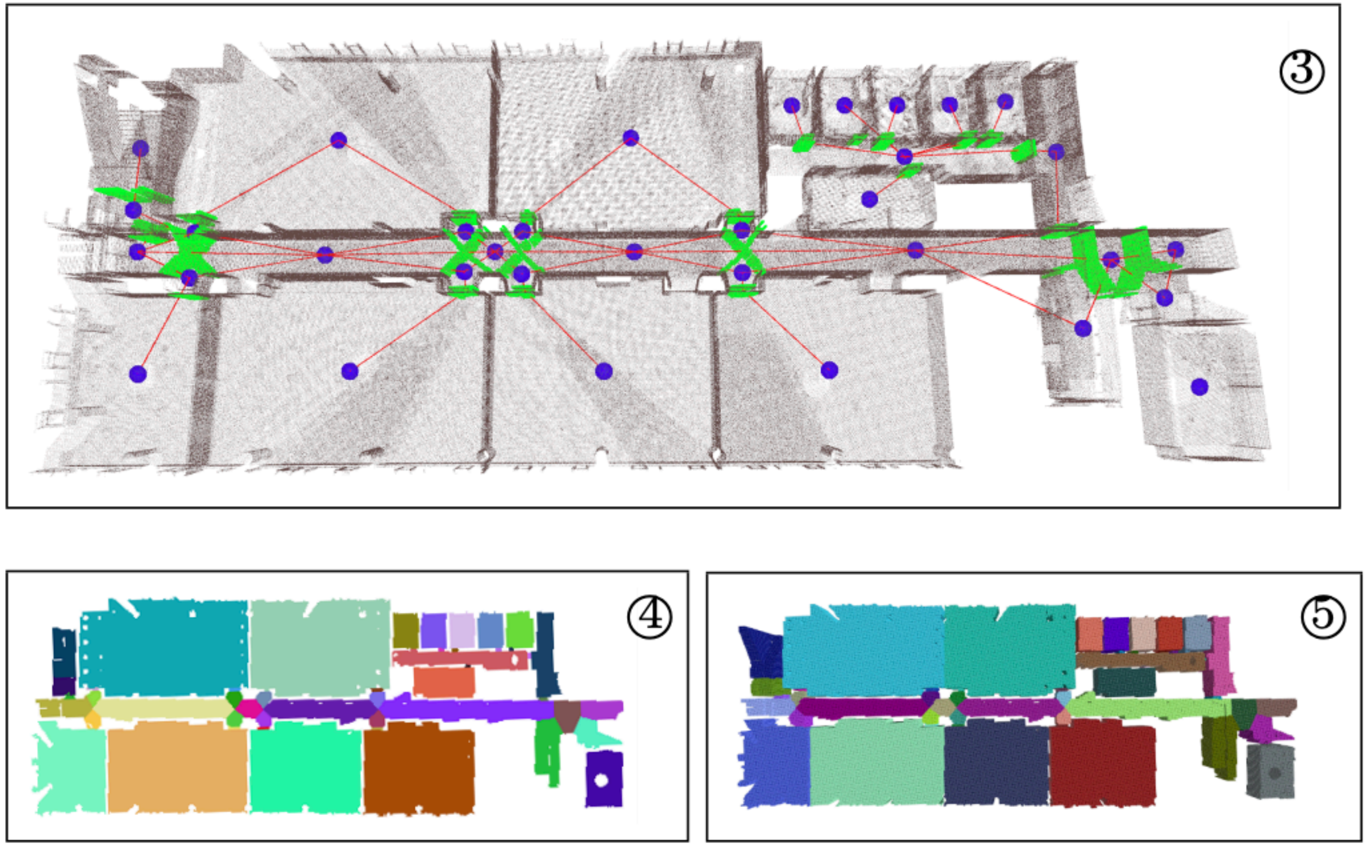
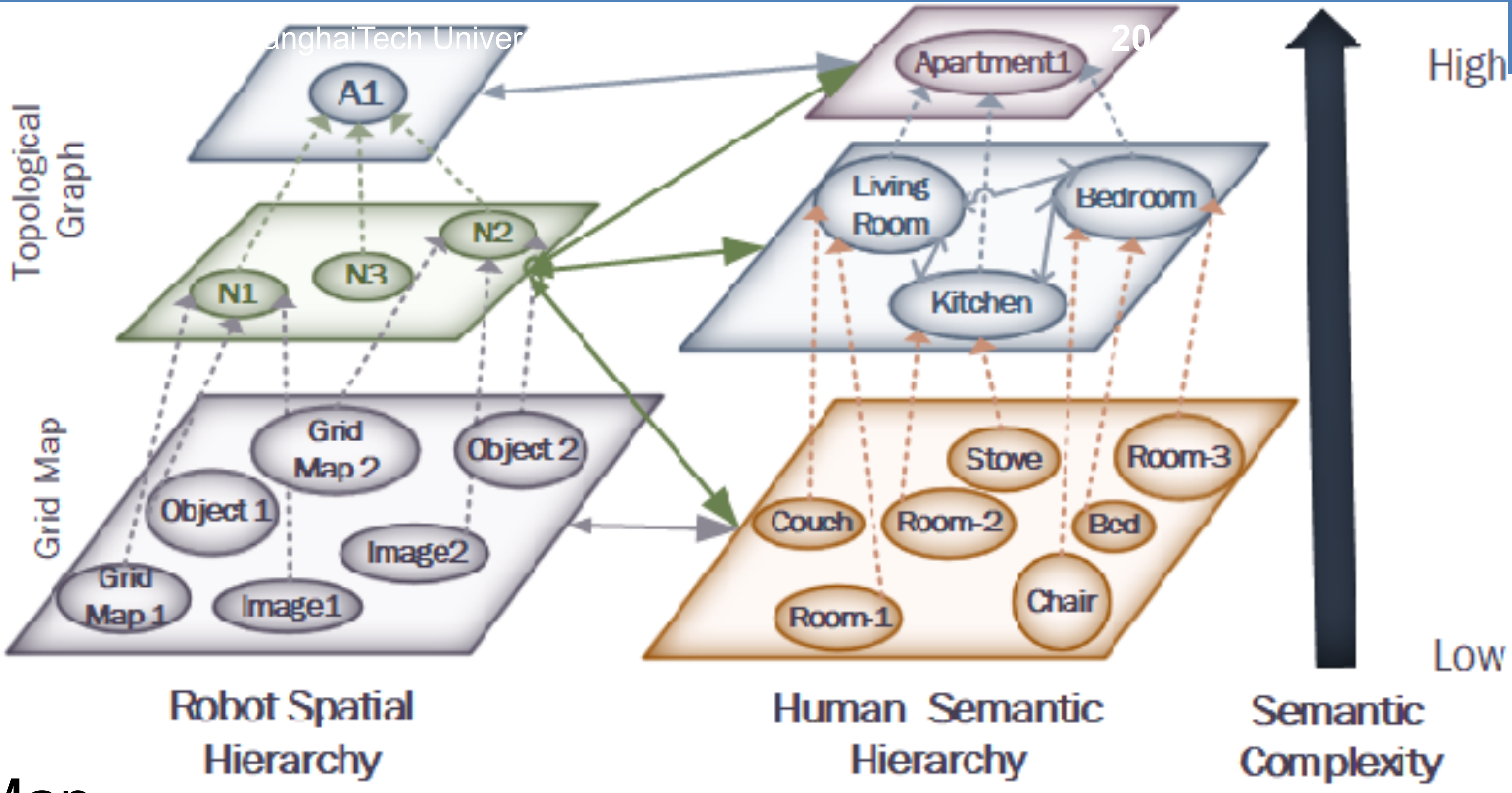
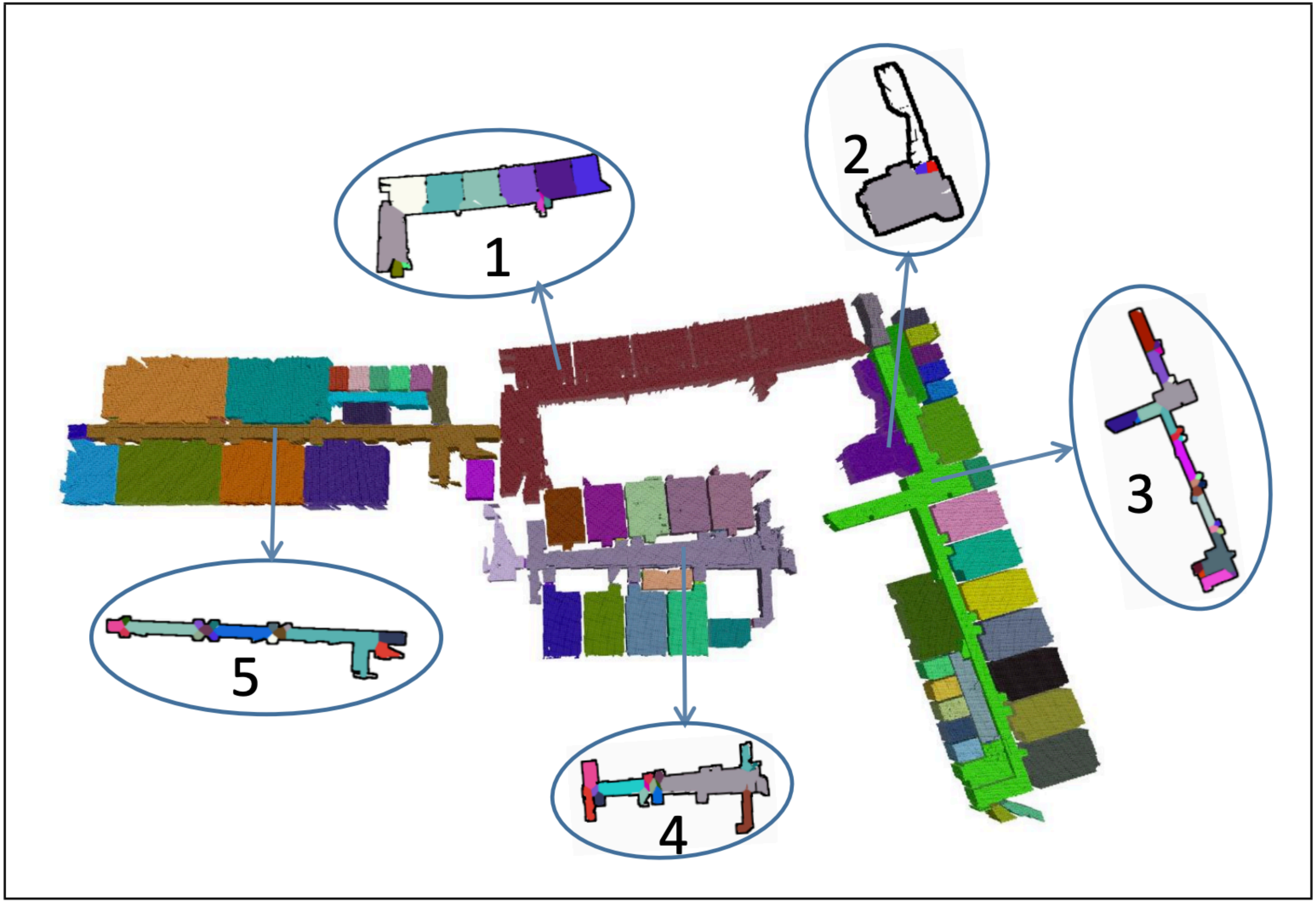


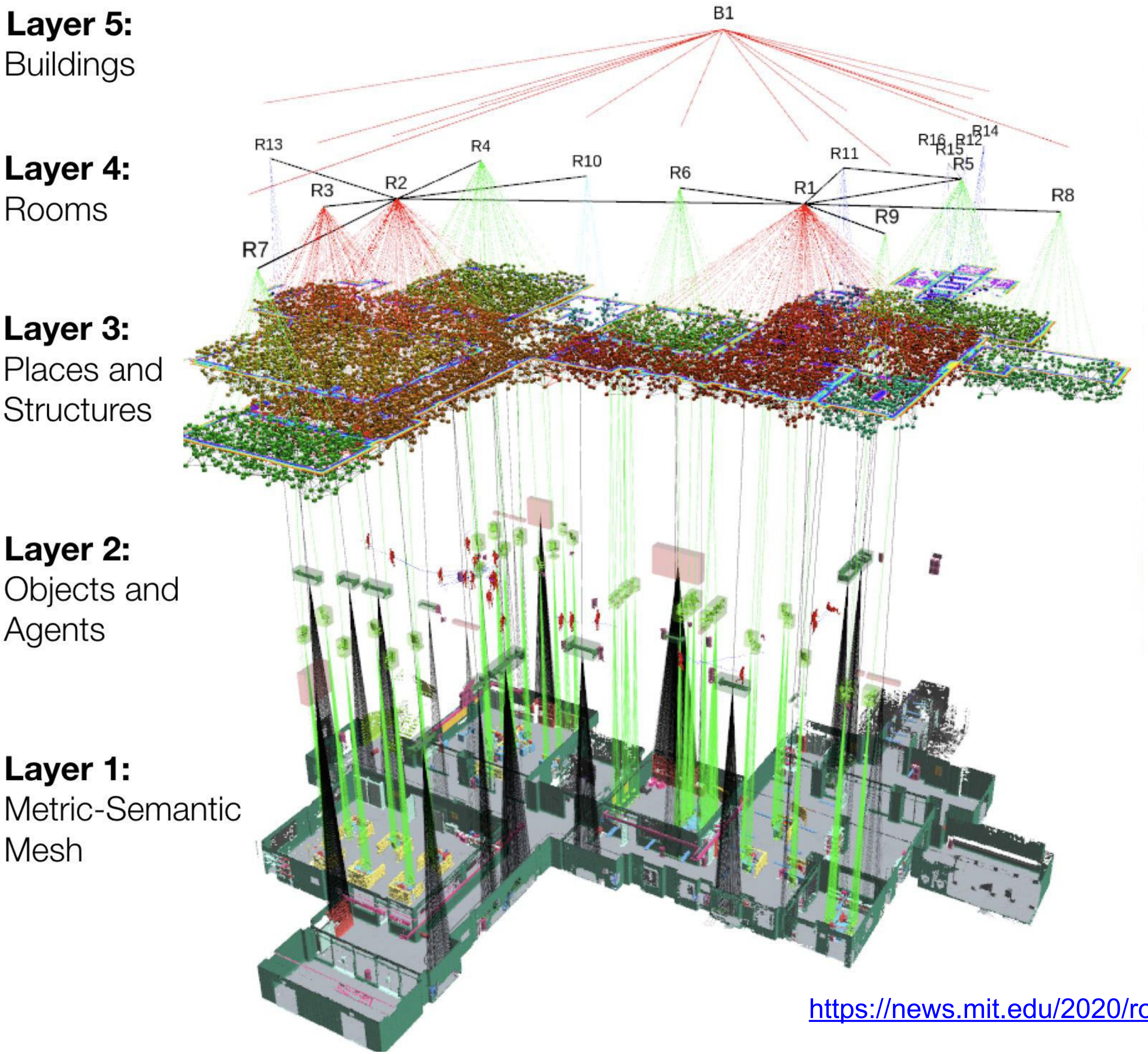
Fig. 1: Input 3D point cloud of 4 floors of a building. Below: results of our algorithm in different dimensions. From top to bottom, left to right are: 0D, 1D, 2D, 3D

Hierarchical Maps



- Higher abstracted maps that contain lower ones with more details
- E.g. Grid map & Topological Map
- Useful for very fast planning; Human Robot Interaction; …
- Another form: image pyramid in GIS (different scale images)

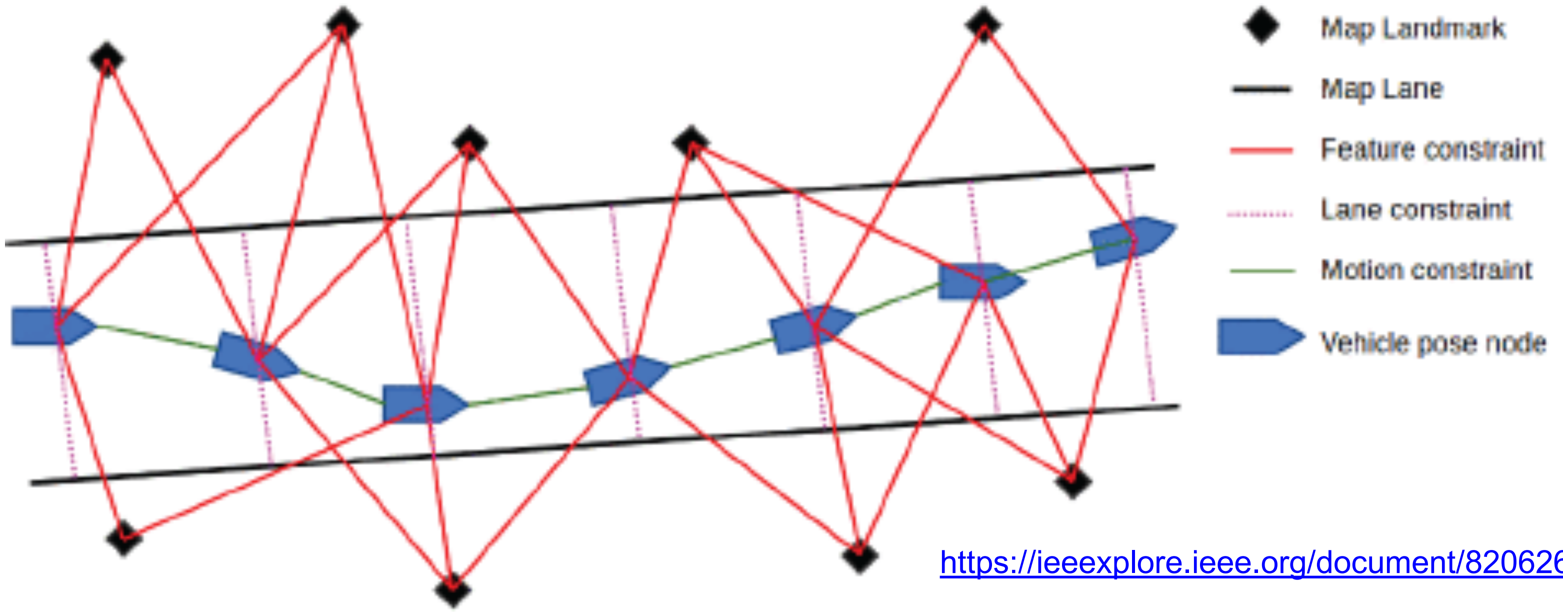

1
2
3
4
5



https://news.mit.edu/2020/robots-spatial-perception-0715

Pose Graph

- Graph structure
- Nodes are:
  - Robot
  - Landmarks/ observations



https://ieeexplore.ieee.org/document/8206264

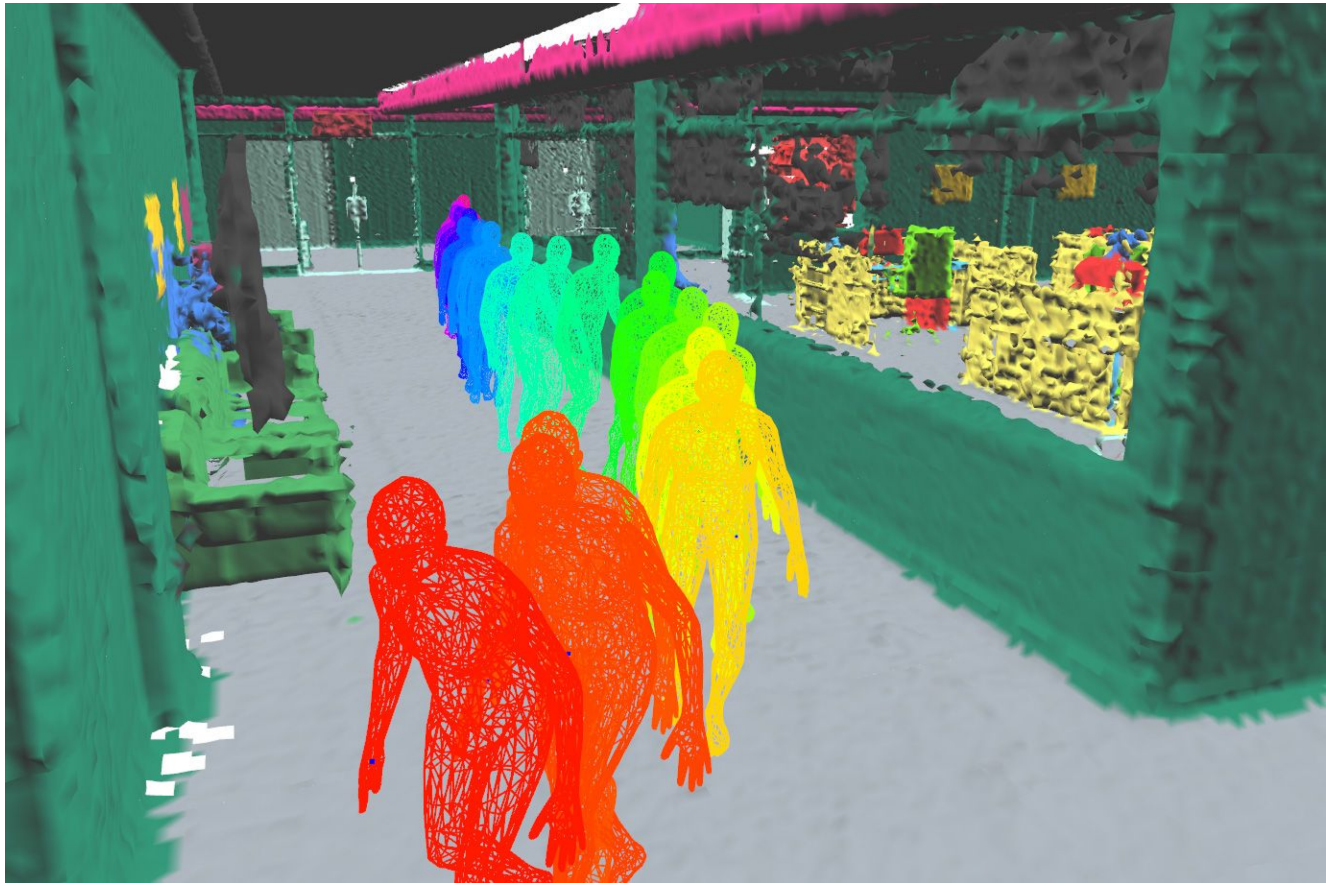
- Used for Simultaneous Localization and Mapping (more details later in course)

- Typically saves (raw) sensor data in robot nodes =>
- For most applications: needs to be rendered before using it:
  put all sensor data in common frame in a point cloud or grid map or plane map or ...

Dynamic Map

- All map representations above assumed a static environment: nothing moves
- Dynamic Map: capture moving objects (e.g. cars, humans)

- E.g: 3D Dynamic Scene Graphs:
- enables a robot to quickly generate a 3D map of its surroundings that also includes objects and their semantic labels
- Some can be dynamic (can move)



https://news.mit.edu/2020/robots-spatial-perception-0715

Other Models of the Environment/ Map Representations

- Many different possibilities:
- A set of images
- All kinds of sensor data (e.g. smoke map; noise map; radiation map)
- Heat map (infrared readings)
- …

MAPPING

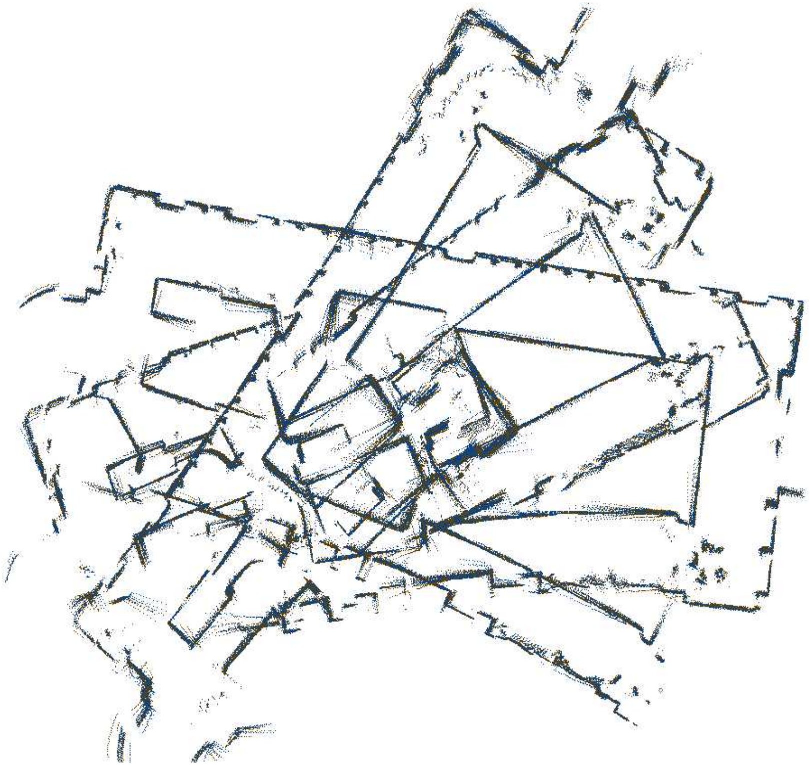
Mapping

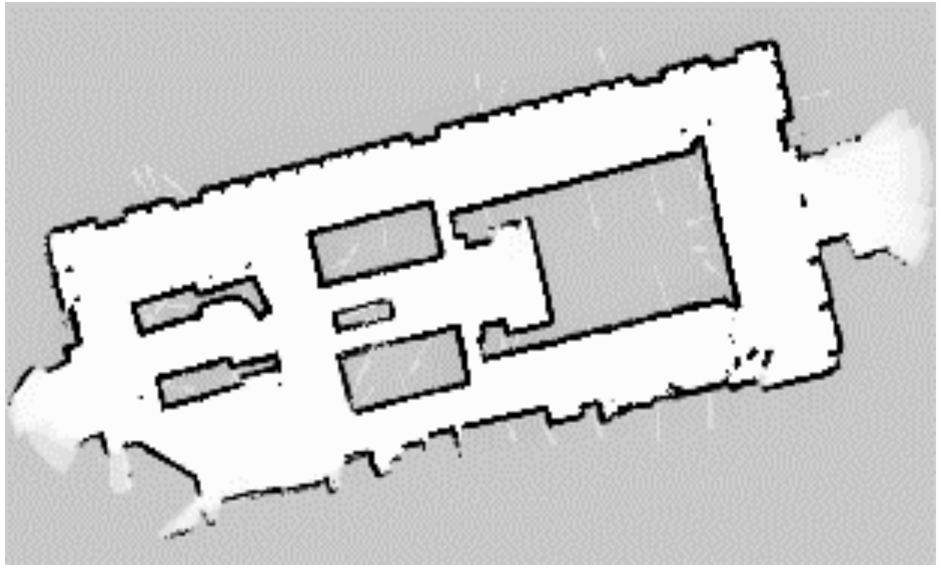
- Process of building a map
- Basic principle:
  1. Initialize the map with unknown or free
  2. Take a sensor scan
  3. Maybe pre-process it (e.g. plane detection)
  4. Localize the robot w.r.t. the map frame (maybe difficult!)
  5. Transform the (processed) sensor scan to the global frame
  6. “Merge” 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



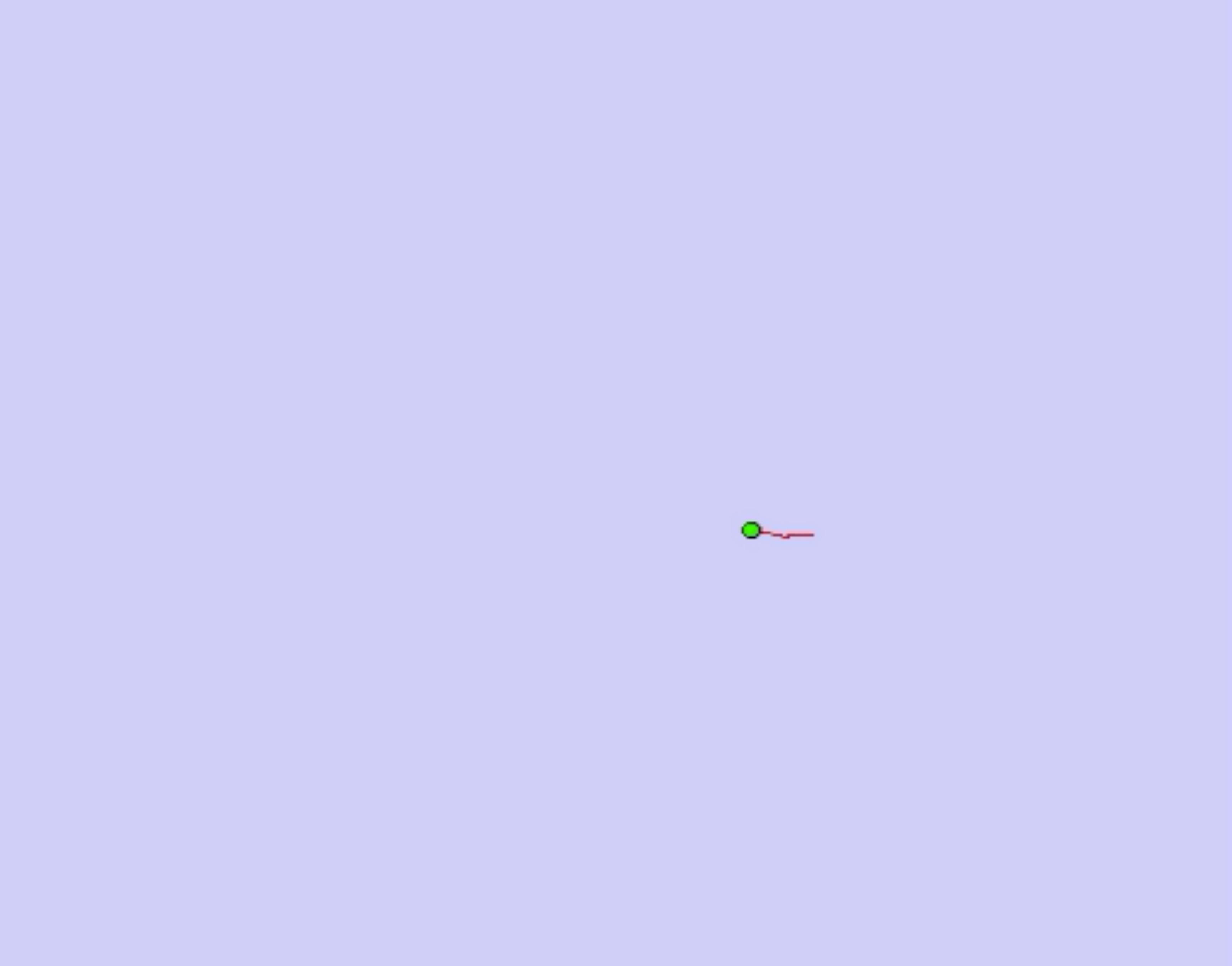


Raw Odometry

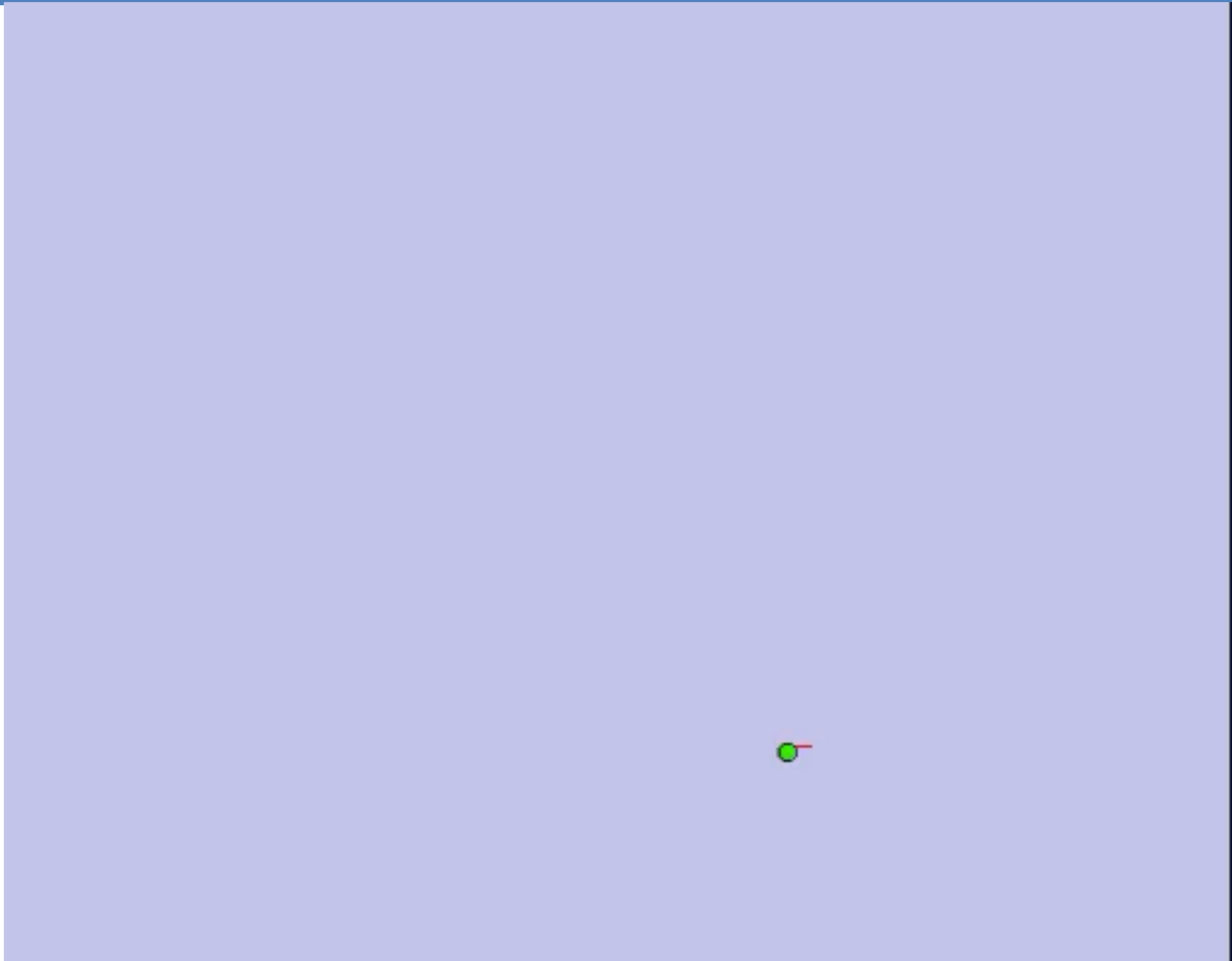
- Famous Intel Research Lab dataset (Seattle) by Dirk Hähnel

Courtesy of S. Thrun

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

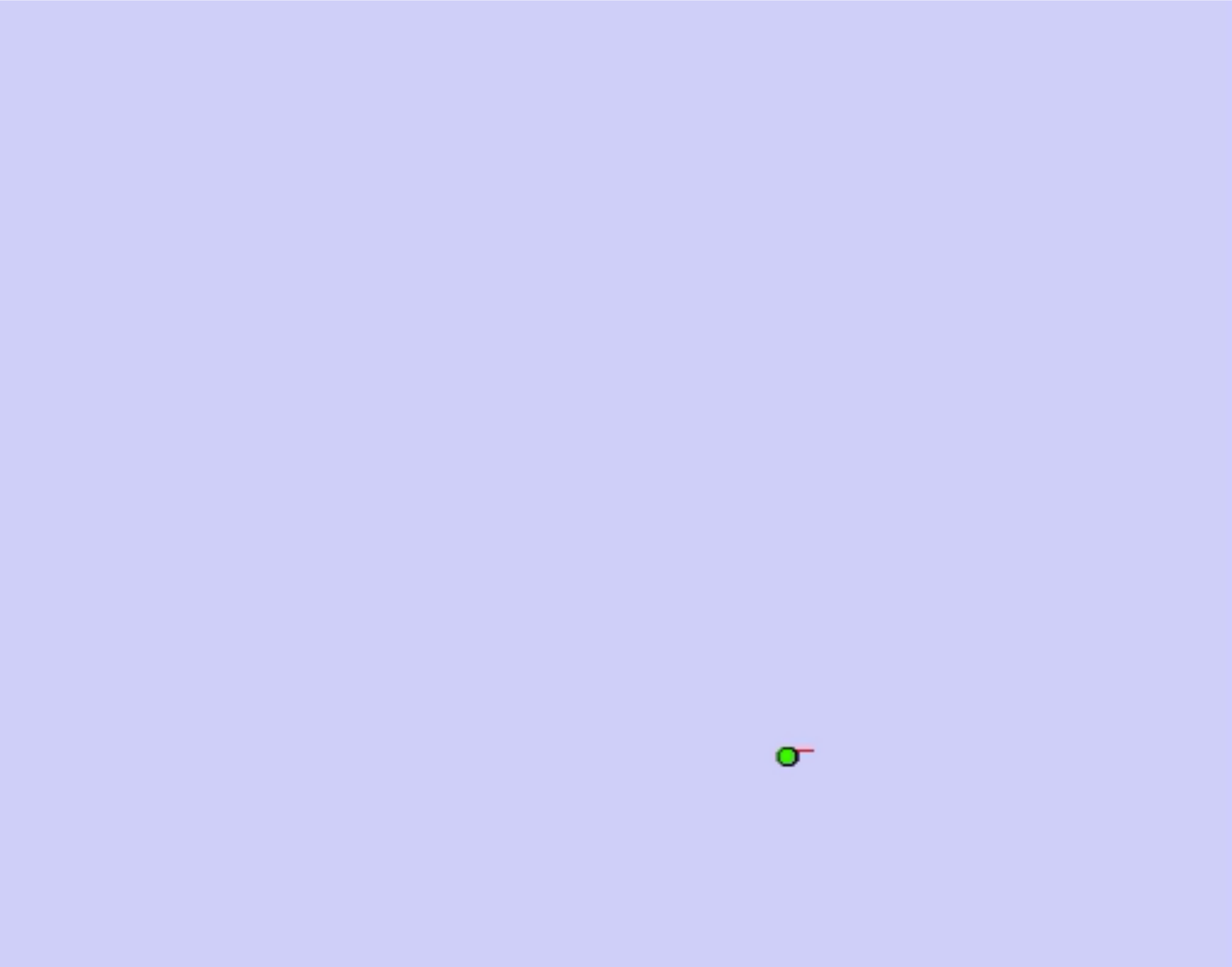


Scan Matching: compare to sensor data from previous scan

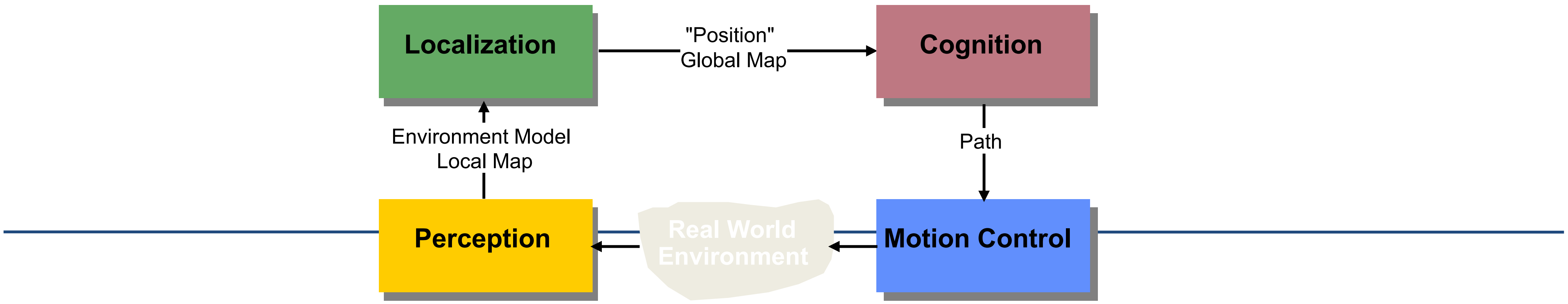


Courtesy of S. Thrun

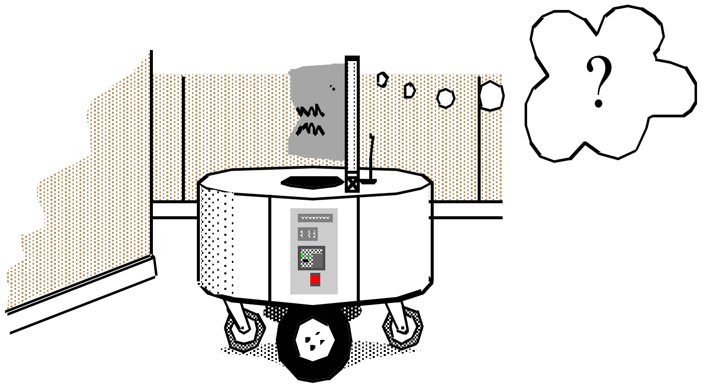
FastSLAM: Particle-Filter SLAM



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

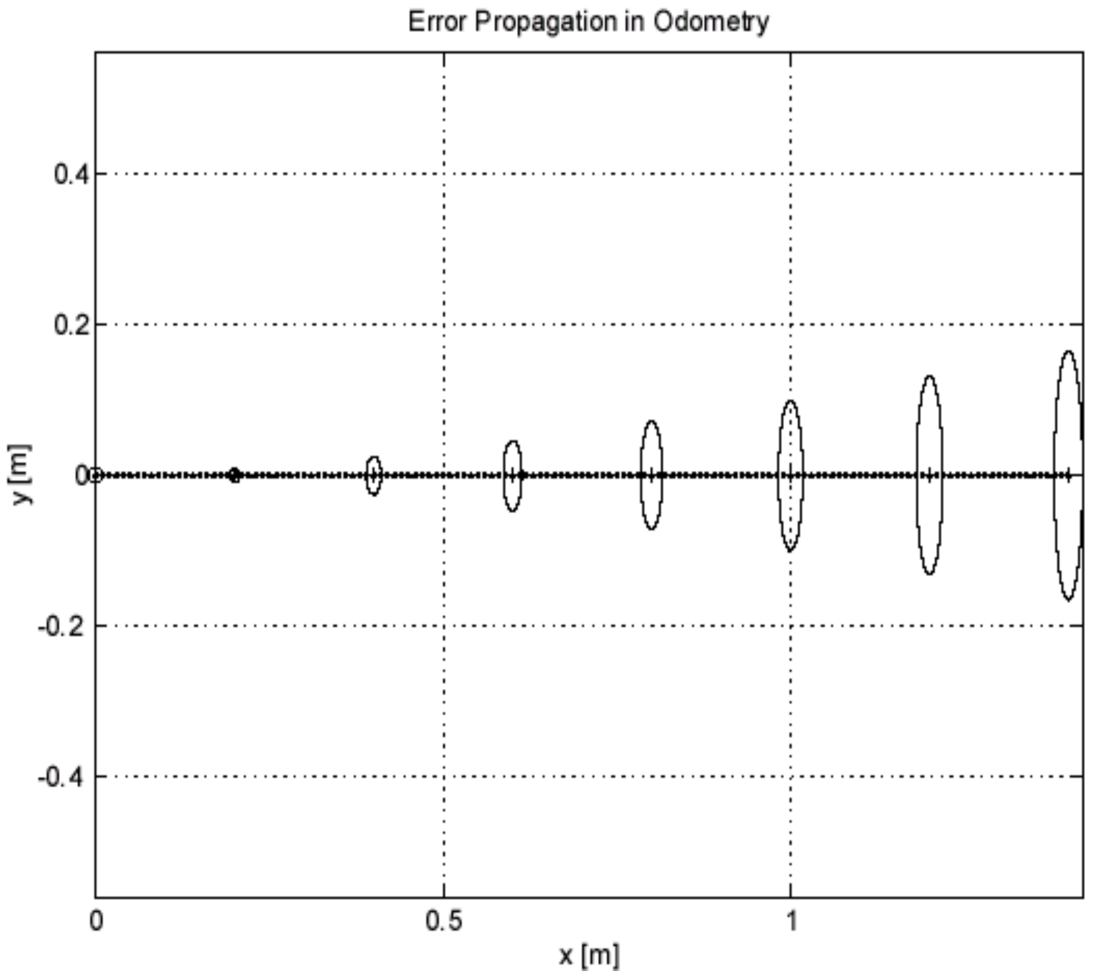
- Odometry and dead reckoning:
  Position update is based on proprioceptive sensors
  - Odometry: wheel sensors only
  - Dead reckoning: also heading sensors

- 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.

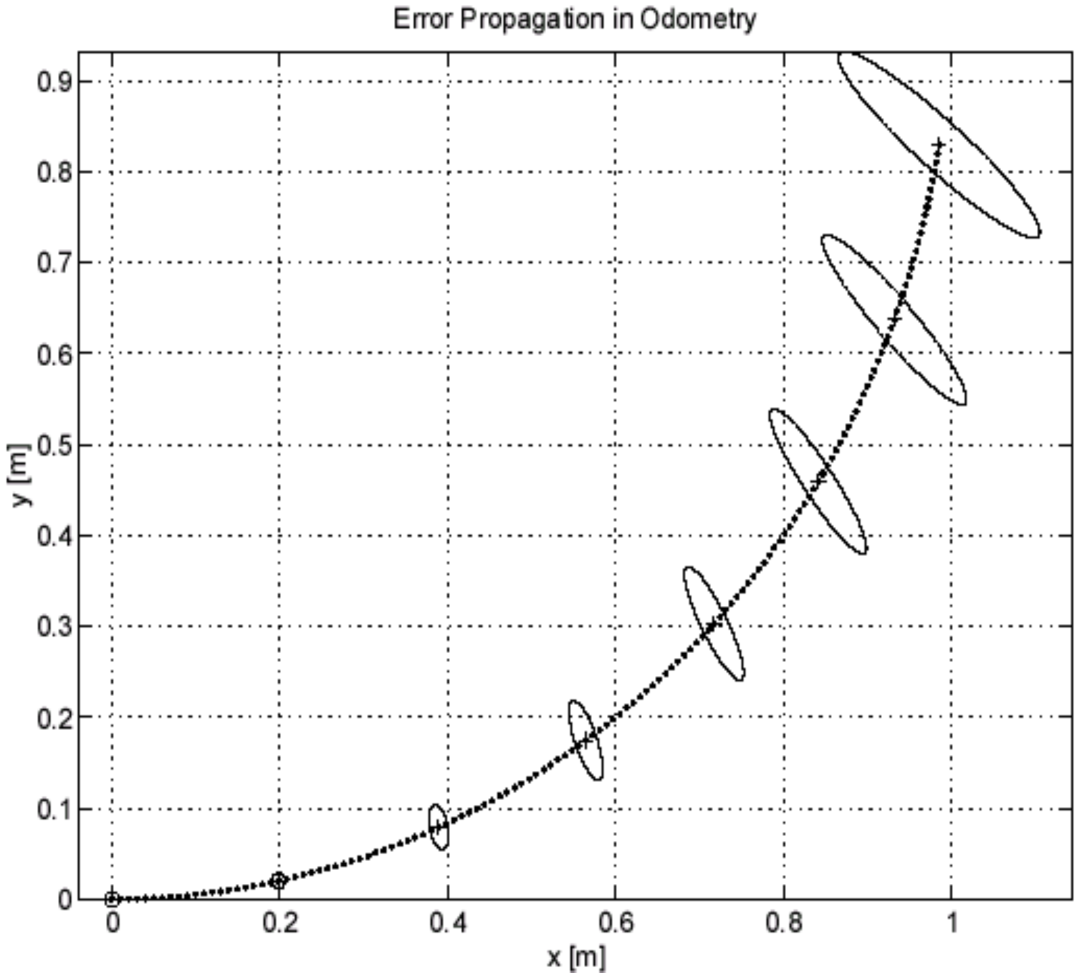
Odometry: Growth of Pose uncertainty for Straight Line Movement

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



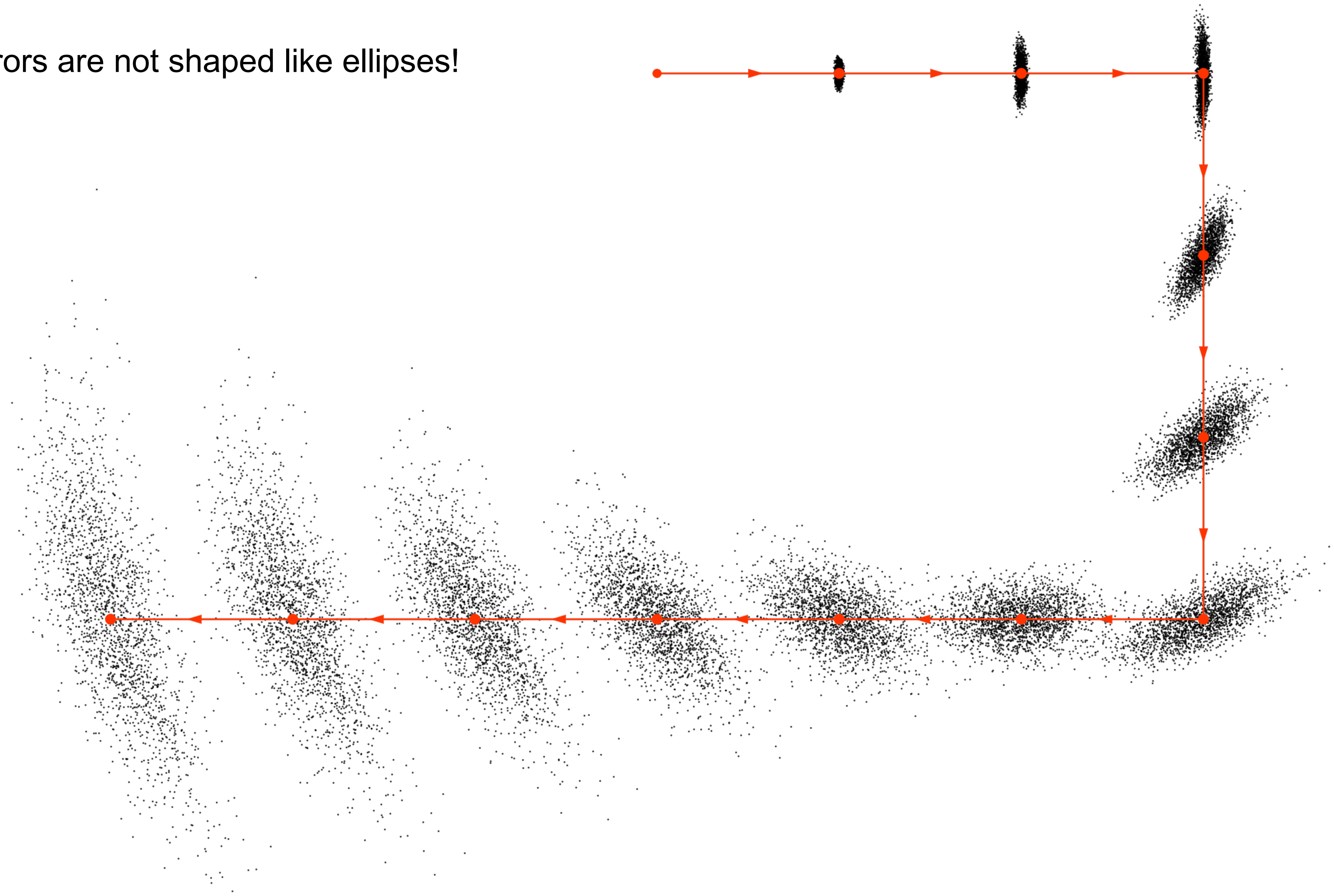
Odometry: Growth of Pose uncertainty for Movement on a Circle

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



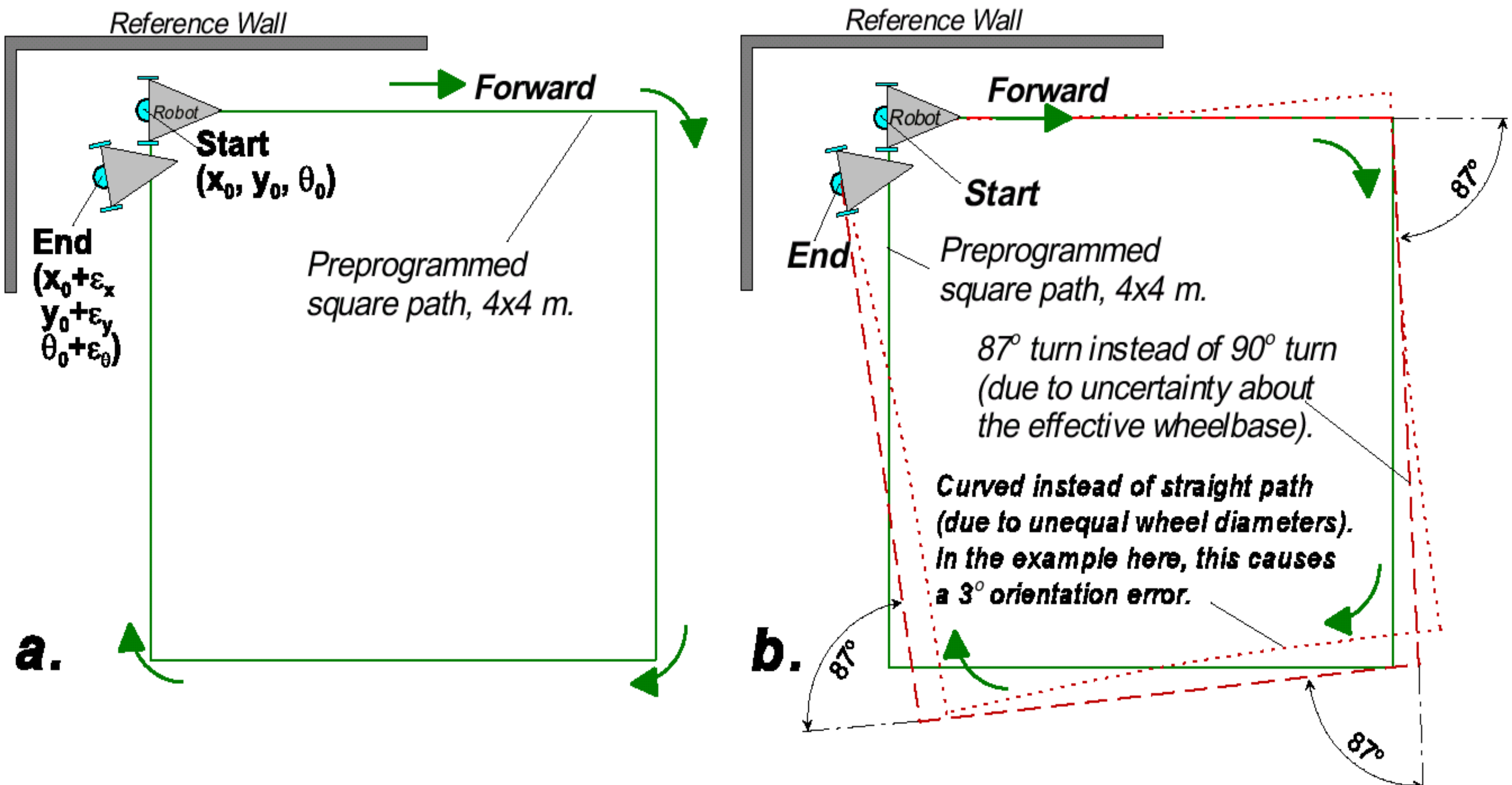
Odometry: example of non-Gaussian error model

- Note: Errors are not shaped like ellipses!



Odometry: Calibration of Errors

- The unidirectional square path experiment



ADMIN

Admin

- Project Proposal due Oct 1
  - Best meet with advisor again!
- HW2 due today!
- Bachelor Thesis?
  - Talk to Prof….

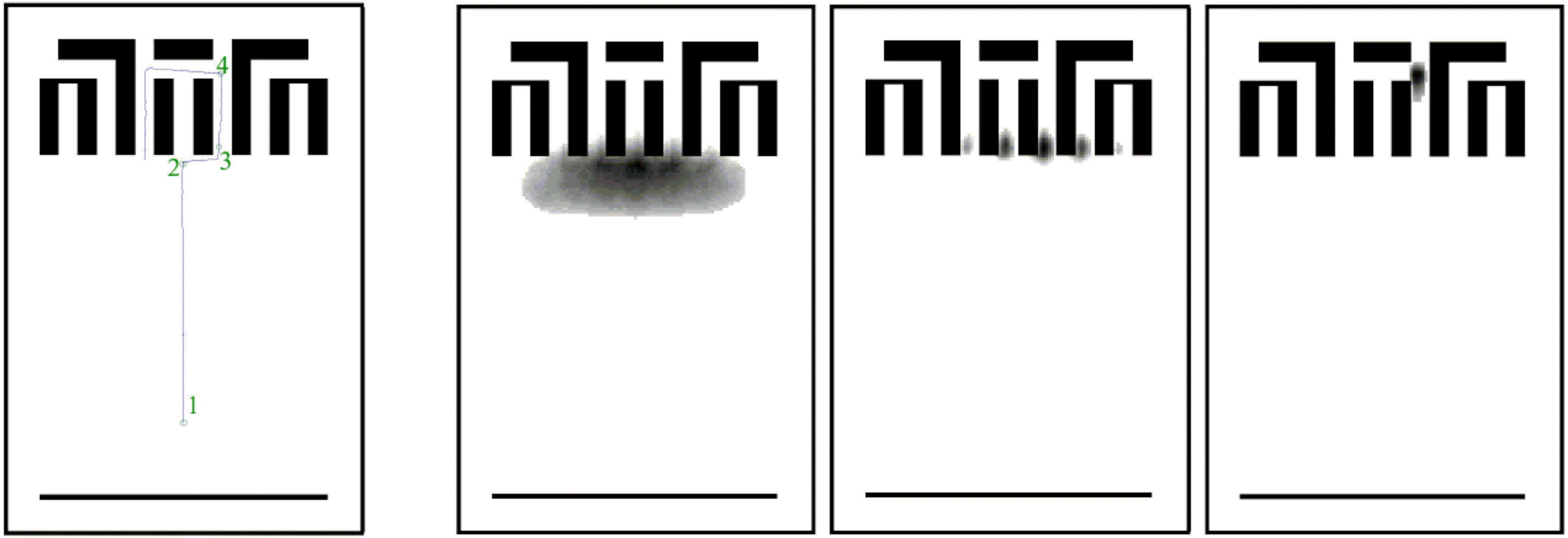
LOCALIZATION METHODS

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

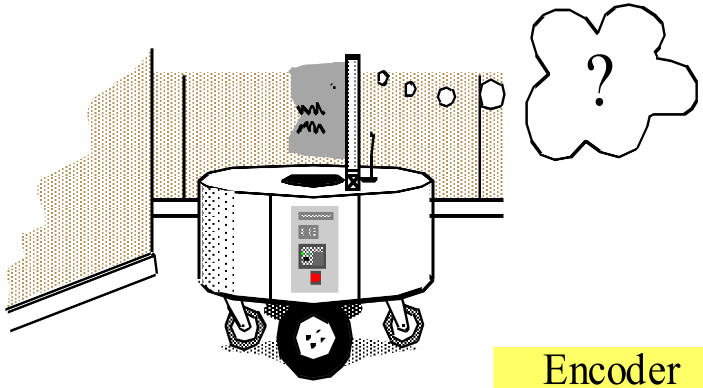
Probability of robot location saved in grid cells – based on combination of:
1) cell values of previous step; 2) odometry; 2) scan matching

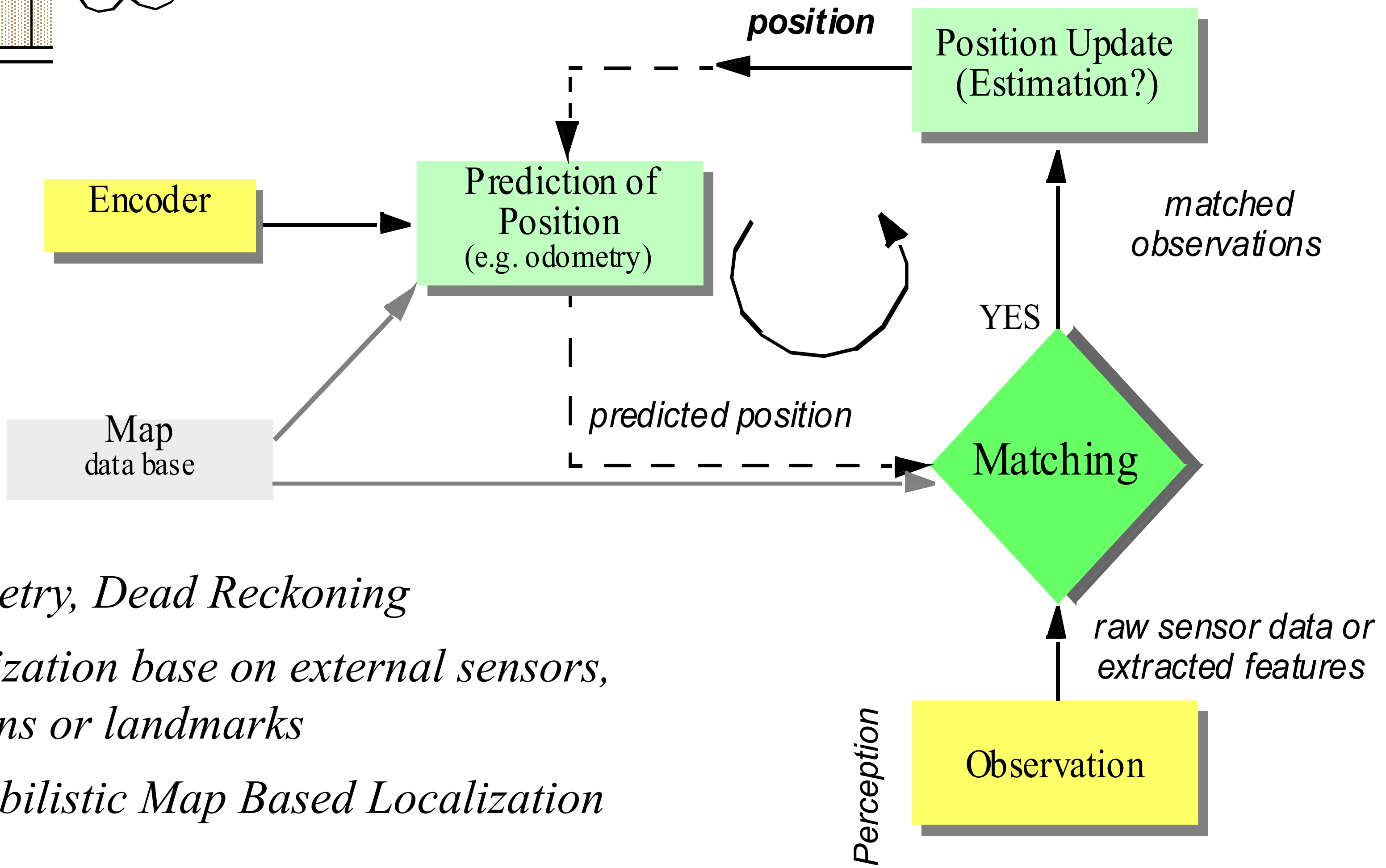


Path of the robot *Belief states at positions 2, 3 and 4*

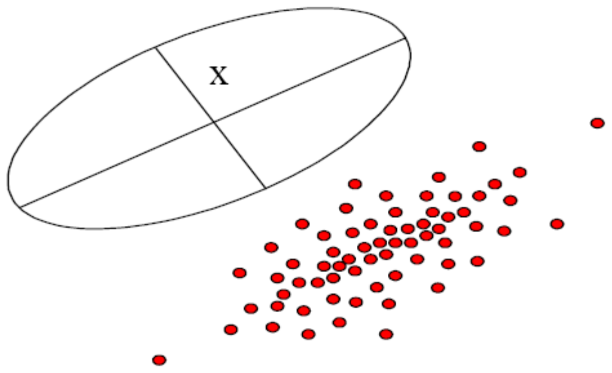
Courtesy of W. Burgard

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

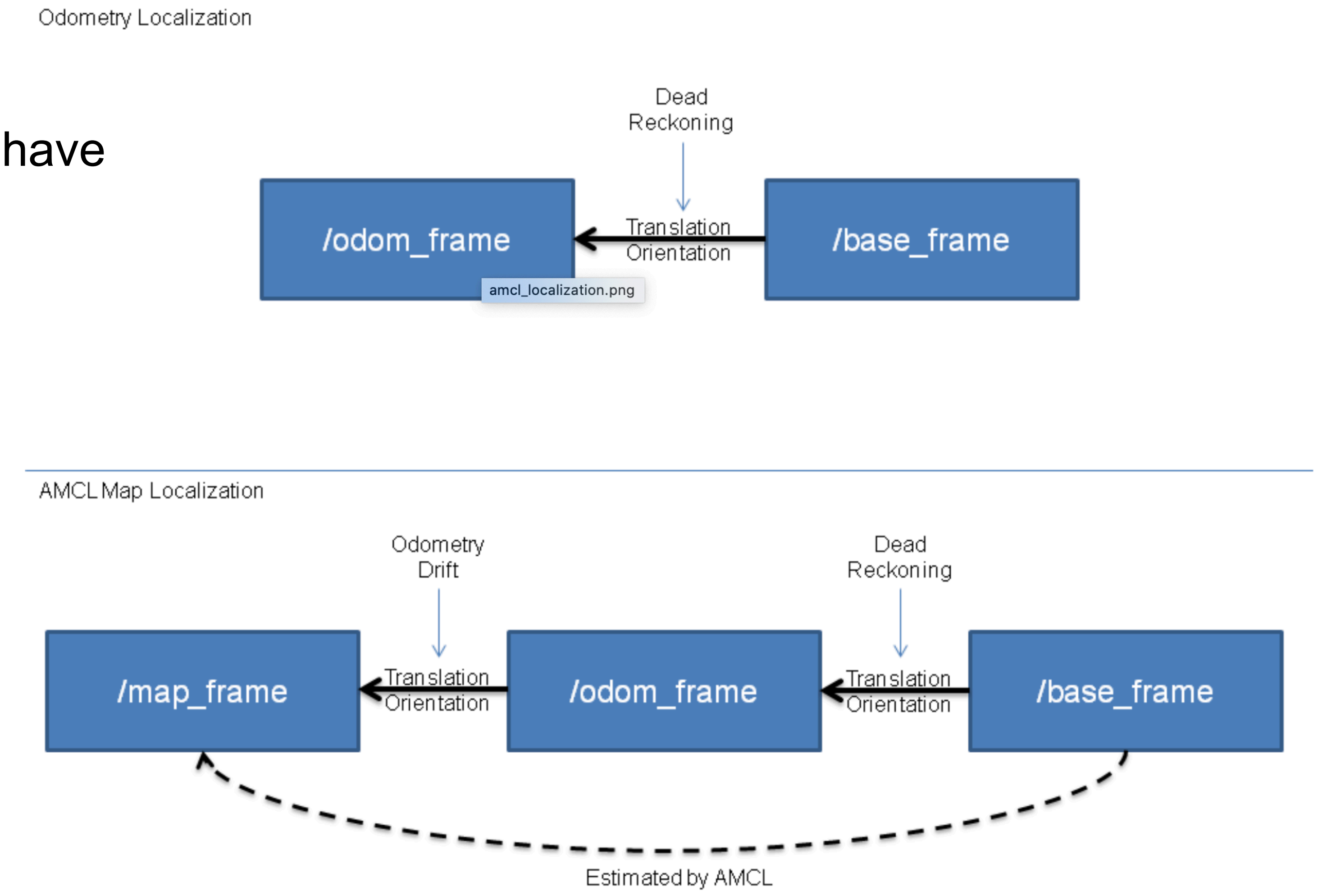
- Algorithm:

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

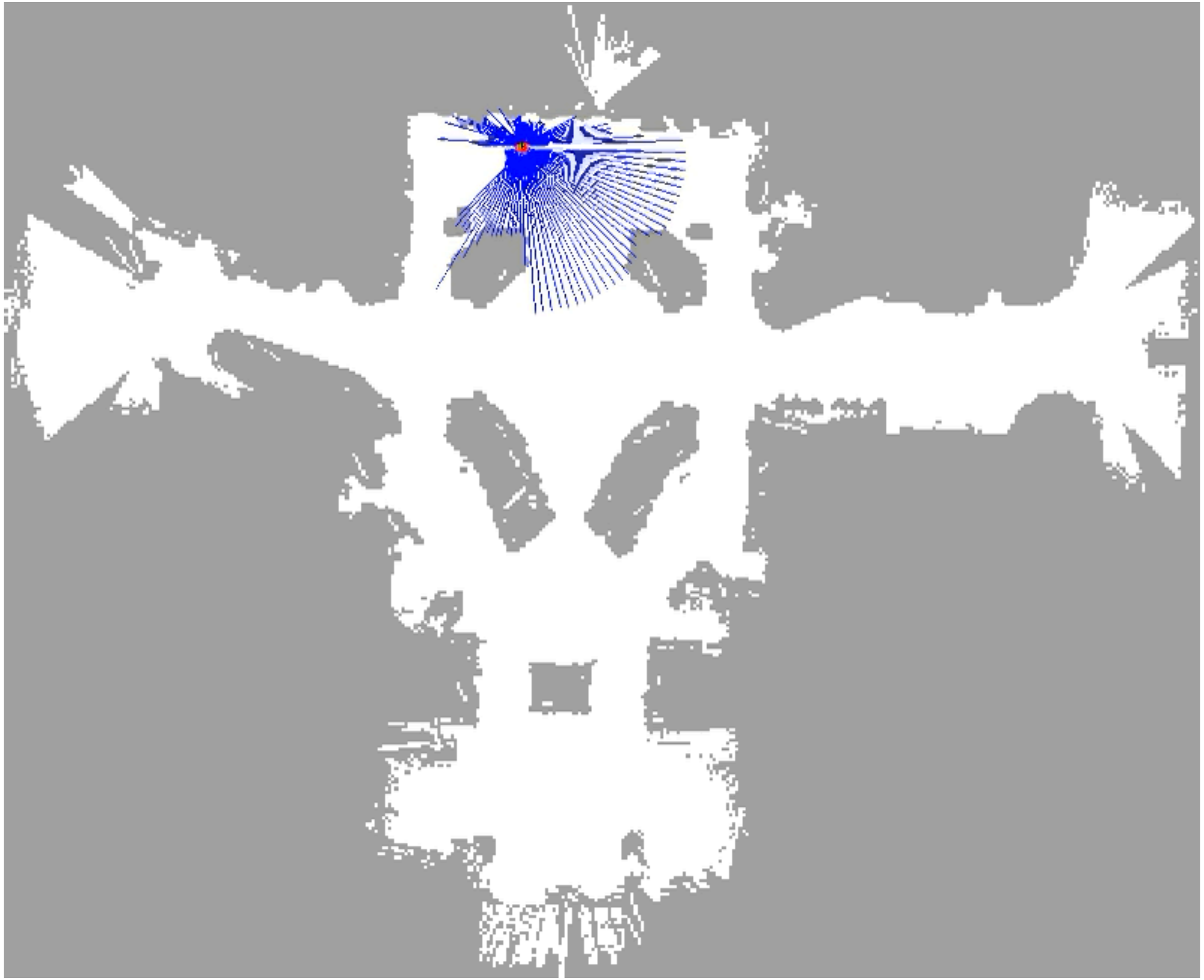
- Can solve the kidnapped robot problem (also wake-up robot problem)
- Problem: Particle of correct pose might not exist…

Adaptive Monte Carlo Localization (AMCL)

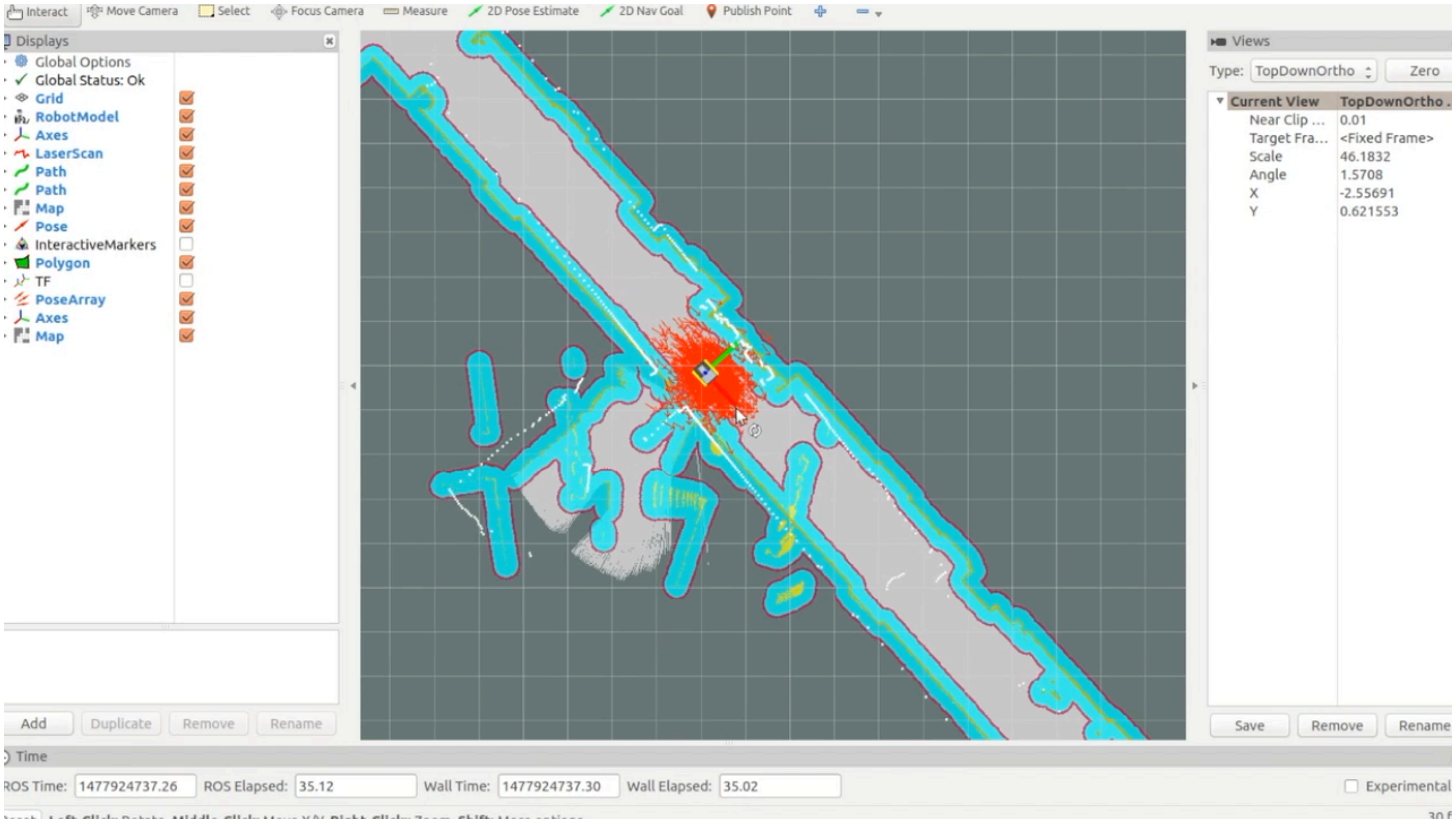
- Sample particles adaptively
  - Based on error estimate
  - Kullback-Leibler divergence (KLD)
  - => when particles have converged, have a fewer number of particles
  - Sample size is re-calculated each iteration
- http://wiki.ros.org/amcl
- Used by the ROS Navigation stack
- Used in MoManTu -> HW 3
- Implement QR-Code based MCL by hand: HW 4



MCL & Robot Kidnapping



AMCL in ROS – play with it in MoManTu!

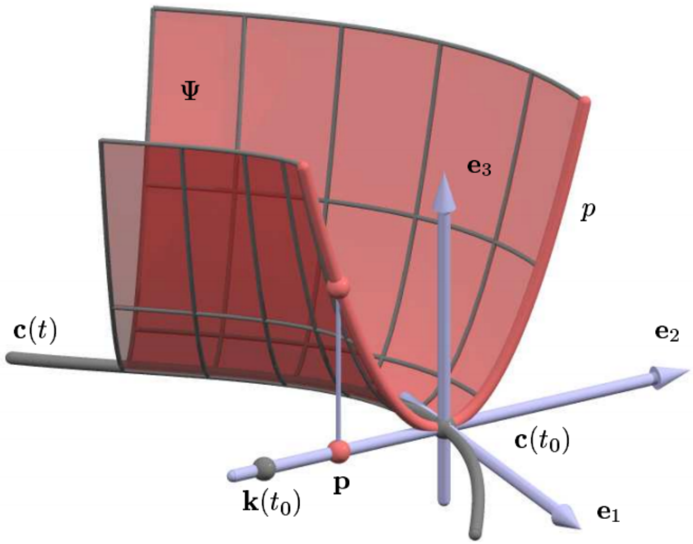


Scan Matching/ Registration

- 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)

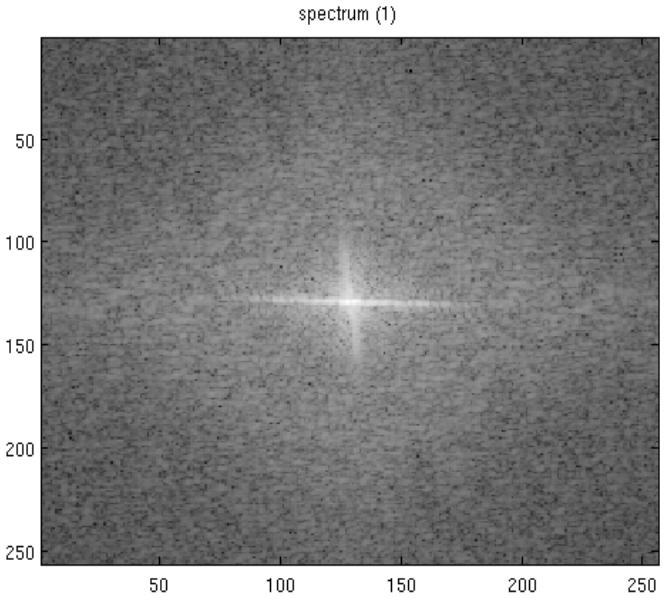


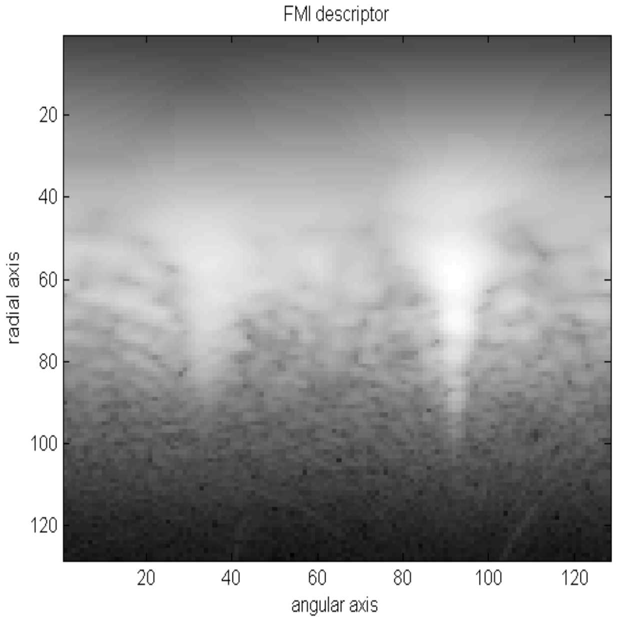
- Feature extracting methods (also for images)
  - Corners in point clouds
  - Lines
  - Planes
  - Feature Descriptors/ also via Deep Learning

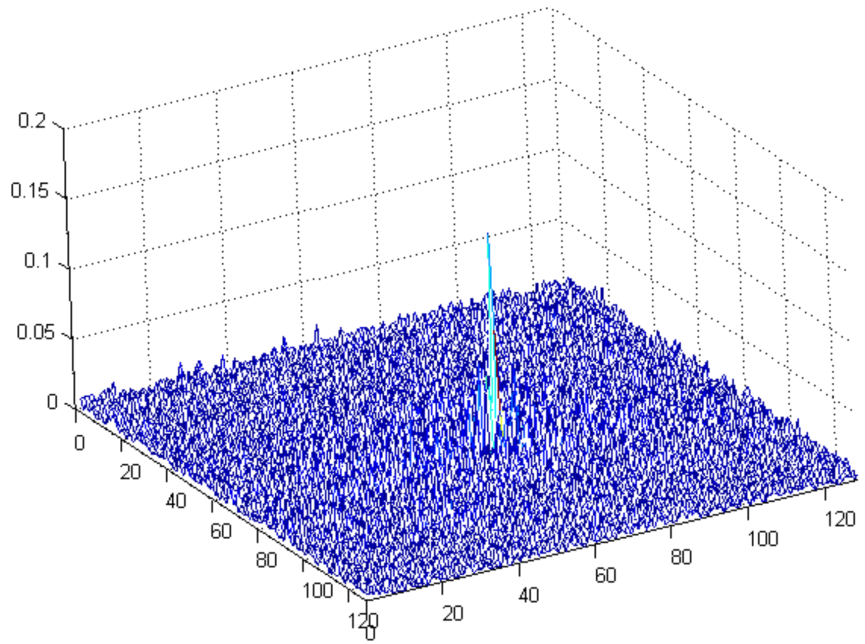
- Spectral methods (also for images)

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





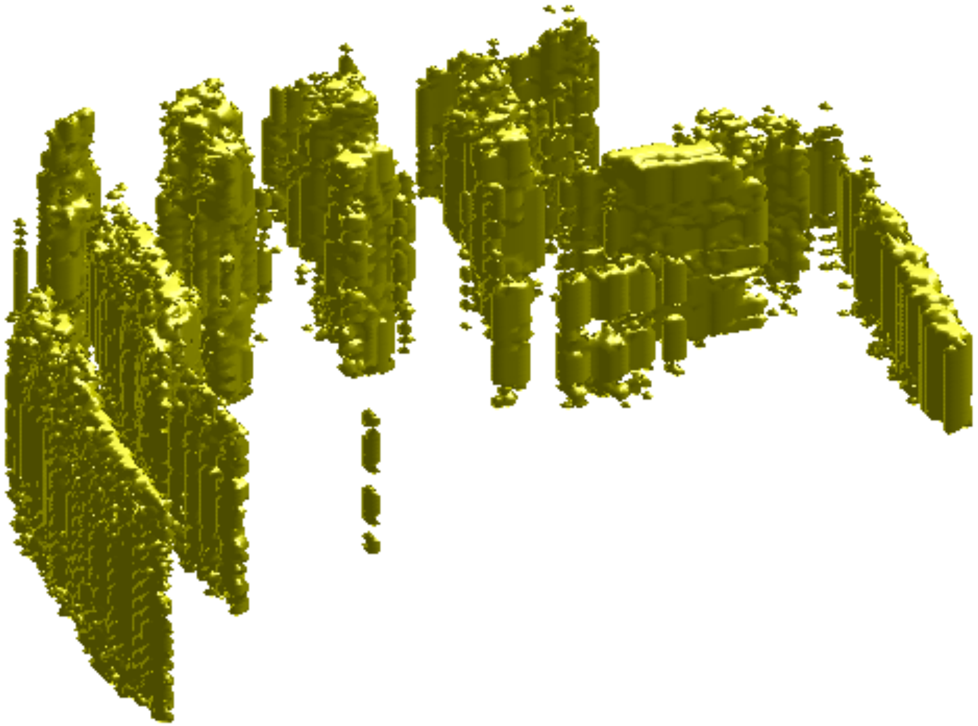


Aerial Map (Mosaic)

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

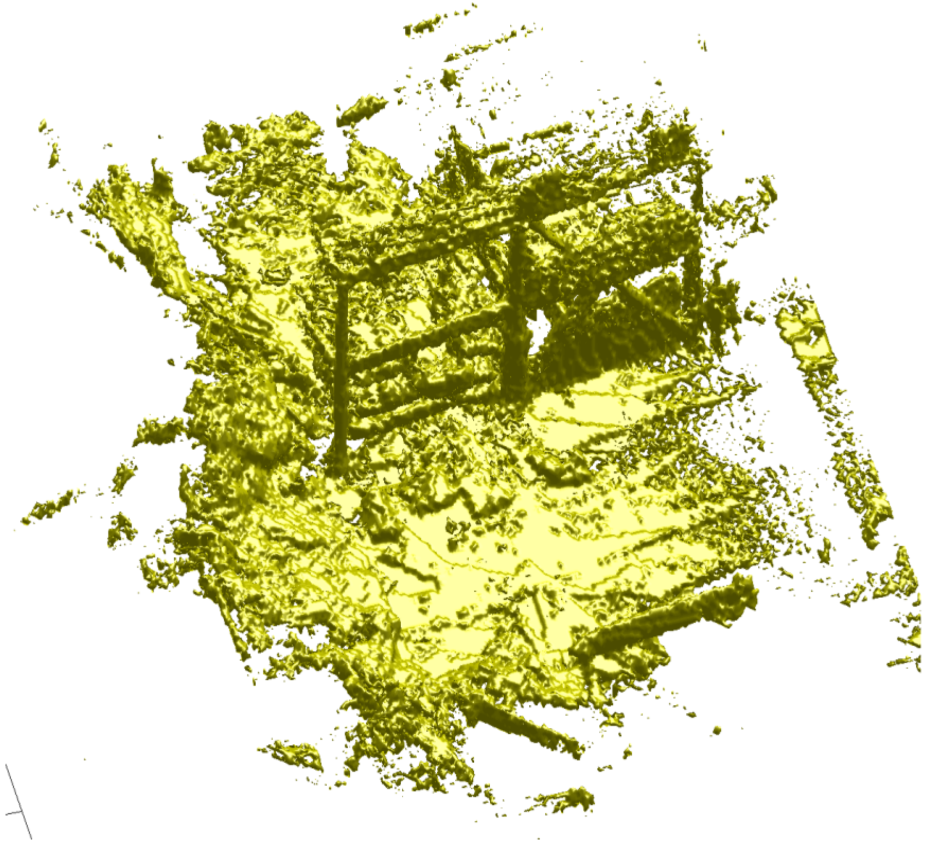
3D Scan matching using Spectral Method

Flood Gate (sonar)

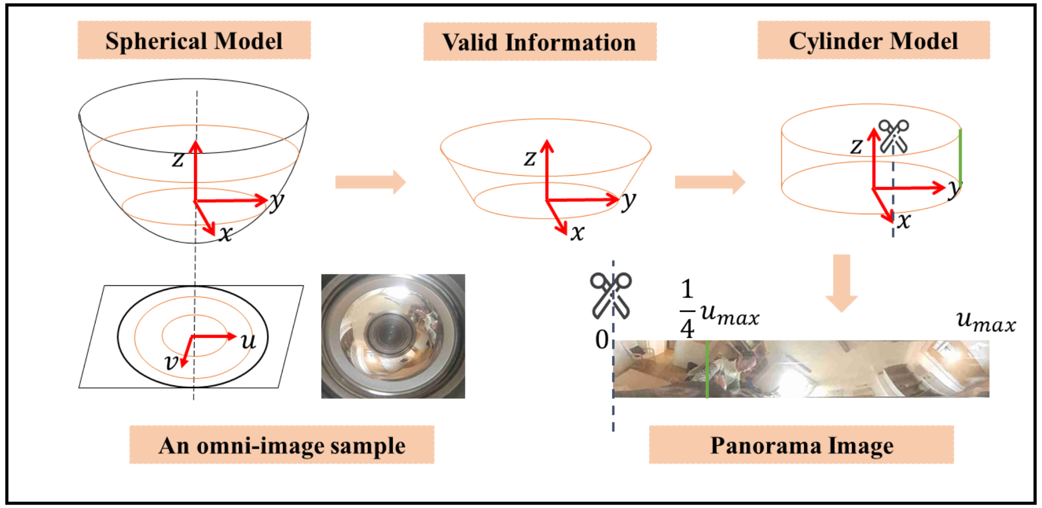


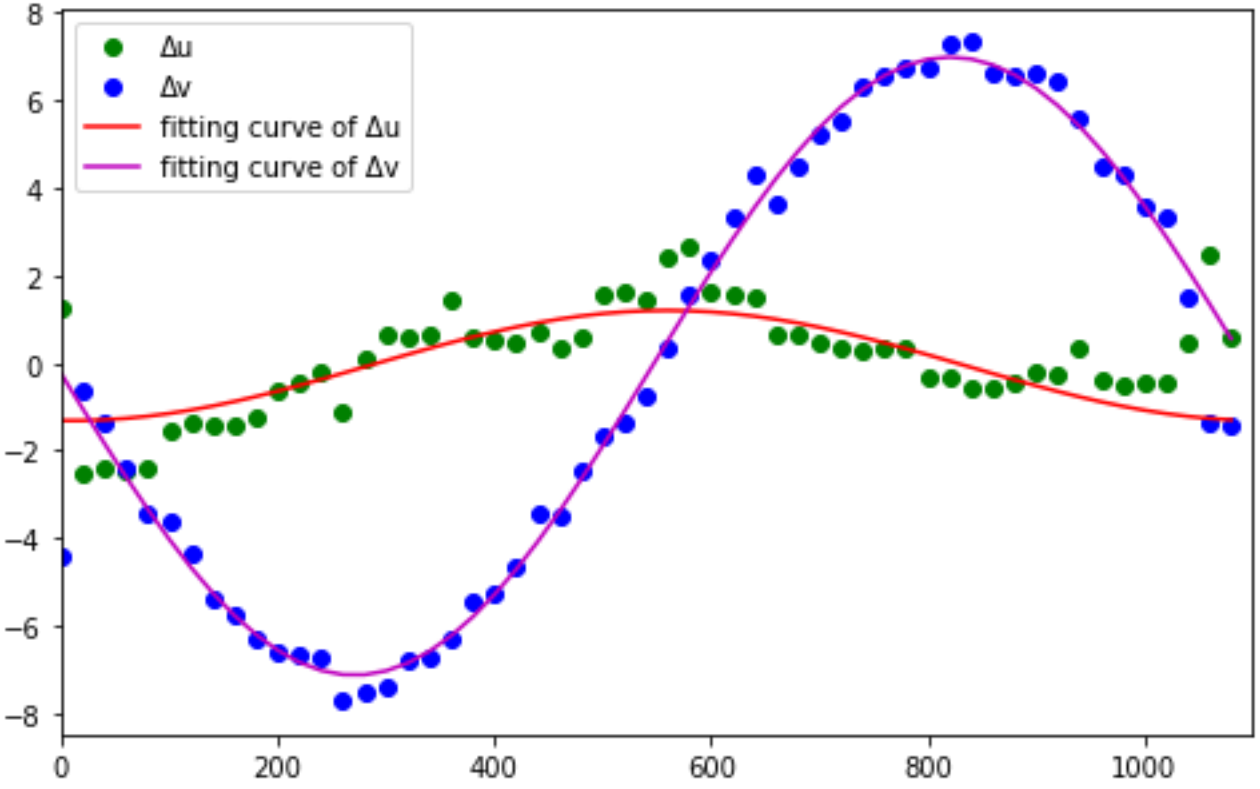
Crashed car park

Disaster City (3D LRF)



Pose Estimation for Omni-directional Cameras using Sinusoid Fitting





$$y = B + A\sin(\omega x + \phi) \qquad (1a)$$

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

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

SLAM using corner structures

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