上海科技大学
ShanghaiTech University

CS283: **Robotics Spring 2026: Maps**

Sören Schwertfeger / **师泽仁**

ShanghaiTech University

ADMIN

Project Proposal & Presentations

- See the website for instructions!

- Project selection: Due Tue, March 24 (send email to TA)
- Project Proposal
  Due Friday, March 31, 23:59

- Paper Presentation
  Due Monday, April 20, 23:59

- Check out https://robotics.shanghaitech.edu.cn/teaching/robotics2026

Paper Presentation

- Choose one paper from ICRA or IROS which is relevant to your project!
  - ICRA: https://ieeexplore.ieee.org/xpl/conhome/1000639/all-proceedings (any year!)
  - IROS: https://ieeexplore.ieee.org/xpl/conhome/1000393/all-proceedings (any year!)
  - Only full papers (6 or more pages) are allowed; no workshop papers

- **Send TA an email with your paper – no double papers are allowed – first come first serve! Latest on April 10!**

- Present the paper as if it were your own work!
- Front page: Name of the Paper; Full citation of the paper; **Your name in Pinyin; Your email address**
- Last slide: **ONE** slide about how this paper is relevant to your project.
- Your presentation has to be professional – not cute…
- Submit pdf or ppt to the project group paper repository (inside /doc) till Monday to repo! Late submissions (or if you come with the ppt/ pdf to the presentation time) will receive a flat 33% loss of points!
- No videos in the slides (very short animations are OK)
- Put slide numbers into presentation!
- Missed presentation slot (randomized) -> 33% score loss
- Presentations in 3 slots – during lecture hours!
- 10 minute presentation plus 1 minute project relevance plus 2 minutes questions
  - Do not rush your presentation! Better present less items more slowly!
  - 10 minute presentation => 8 – max. 12 slides
  - Maybe have a slide towards the end that you can skip if you run out of time.
  - Give a test presentation to your friends beforehand!
- Finish early for practicing – don’t learn by heart.

Grading of the Presentation

- 10 %: Your basic understanding/ knowledge about the paper you present
- 20 %: Presentation timing (plus or minus one minute is ok) – no rushing – good speed!
- 10 %: Correct written English in presentation:
  - No complete sentences, no grammatical or spelling mistakes
- 10 %: Good structure of presentation:
  - Depends on the type of paper, how much time you have, how long you need to present the main achievement.
  - For example: outline, introduction/ motivation, **problem statement**, state of the art, **approach**, experiments, **results**, **conclusion**, outlook
- 20 %: Clarity of written presentation
- 10 %: Good presentation style:
  - Interact with audience: look at the whole room (not just your slides, notes, or the back of the room)
  - Present the paper – do not read (or repeat the learned) speech from a prepared text
  - Use the presentation as visual aid – not as your tele-prompter to read from
  - Move your body – do not stand frozen at one place
- 10 %: Answering the questions
  - Questions have to be asked and answered in English – Chinese can be used for clarification
- 10 %: Asking questions to other students!
- **Not** scored: Your English skill

Project Proposal (1)

- **Title:** Find a nice, catchy title for your project
- **Abstract**: A short abstract/ summary what the project is about
- **Introduction**: general description & Motivation
- **State of the Art**: Literature & open-source-ROS packages
- Per team member:
  - present and cite three papers with just three or four sentences
  - present in more detail one further paper relevant to your project. Describe it with at least 1/3rd of a page.
  - present in detail one open source ROS package relevant to your project. At least 1/3rd of a page
  - => about one page per team member – => 3 pages for 3 person team

Project Proposal (2)

- **System Description**
- **System Evaluation**: Describe how you want to test your system.
  - Experiments & how to measure their success
- **Work Plan**: Define some mile stones.
  - Possible phases: Algorithm design, implementation, testing, evaluation, documentation – some of those things can also happen in a loop (iteration).
  - Deliverables of Project:
    - Proposal (this document)
    - Intermediate Report
    - Final demo
    - Final Report
    - Website
- **Conclusions**: Short summary and conclusions

Project Proposal (3)

- Important dates:
- **March 31, 23:59: due date for the proposal**

- Parts of proposal go into the intermediate and final project report.
- Please don’t forget to take **pictures and videos** when testing your system!

- In English! Using LaTeX!
- Put sources and PDF in git.

- **Additional task:** In glit/ gitlab: “Readme.txt” with:
  - Team Name and Members; email addresses
  - Documentation and how to’s regarding your project.

Project Safety!

- You work with real robotic hardware => can be dangerous!

- Don’t stick your fingers where they don’t belong!
- Be aware of the moving robot!
- Be careful when handling/ charging the batteries! If you are unsure, ask one of my students for help!
- Be careful with electric components.

- Be nice to the robot!

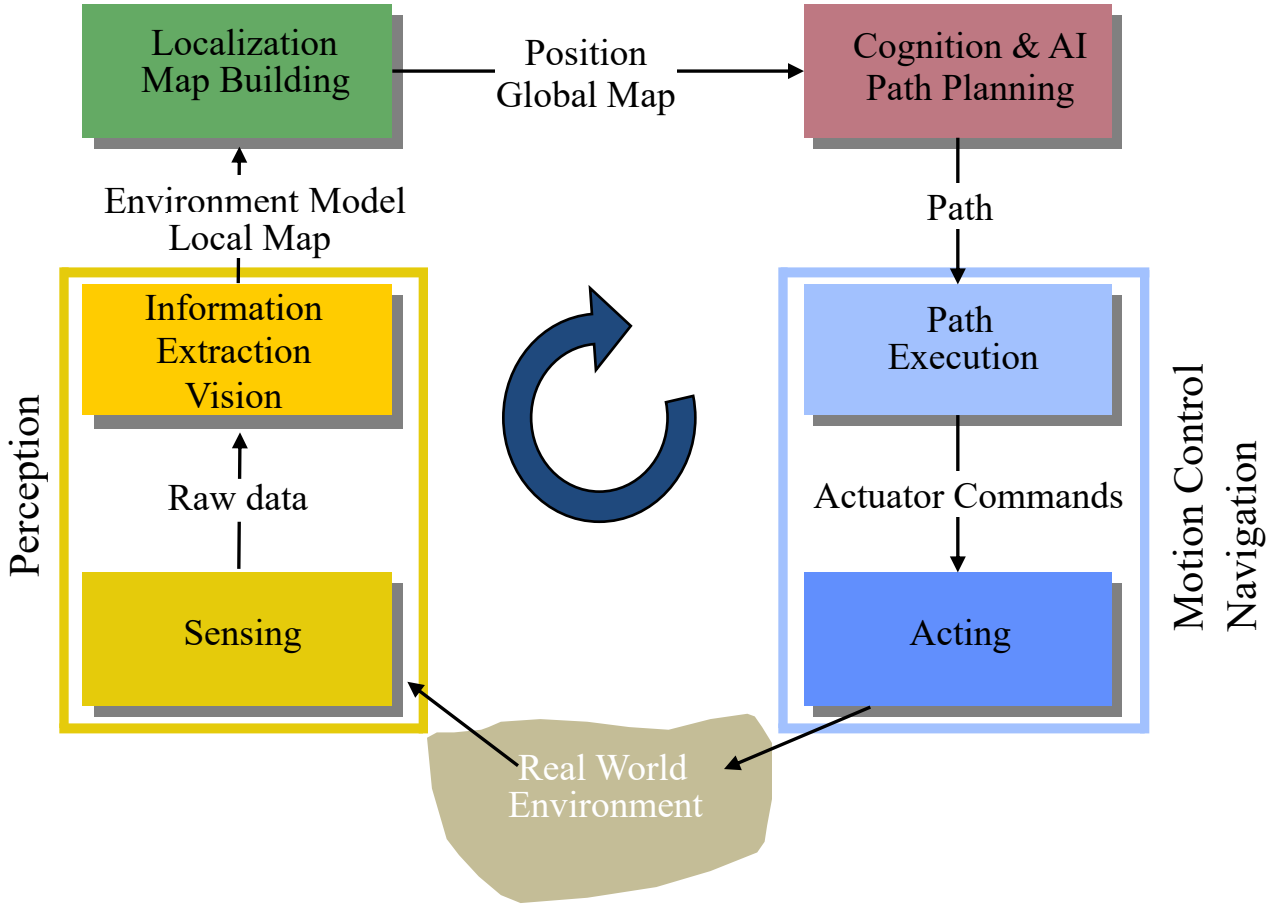
HW2

- Has been posted on https://robotics.shanghaitech.edu.cn/teaching/robotics2026

- Due date: April 3rd

MAP REPRESENTATION

General Control Scheme for Mobile Robot Systems

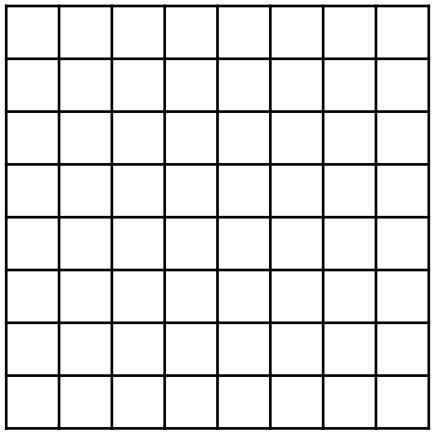


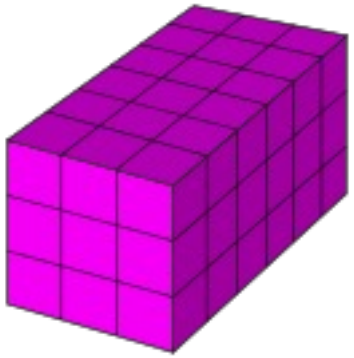
With material from Roland Siegwart and Davide Scaramuzza, ETH Zurich

Map Representation: what is "saved" in the map

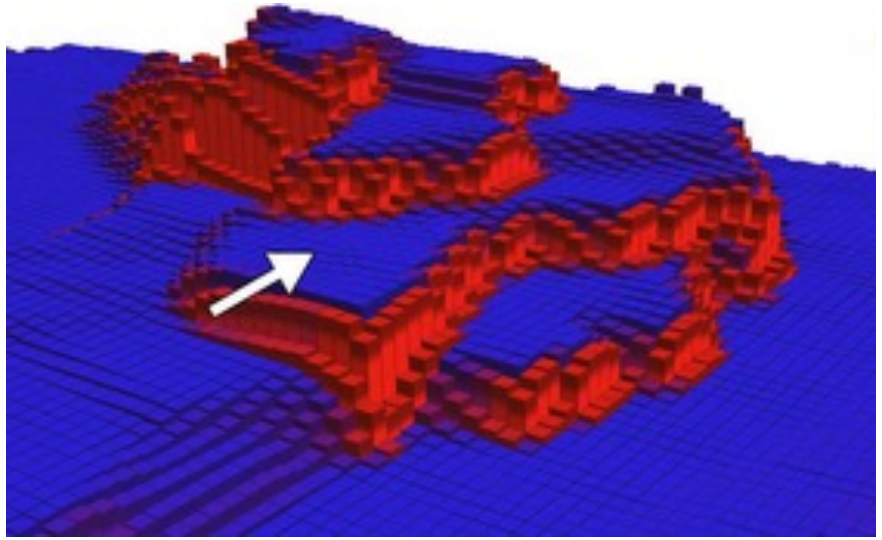
- Points (surface of objects, buildings): 2D or 3D
  - What: x,y or x,y,z coordinates;
    Optional: intensity; maybe RGB; maybe descriptor; temperature; …
  - From range sensors (laser, ultrasound, stereo, RGB-D): dense
  - From cameras (structure from motion; feature points): sparse
  - Variant: kd-tree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

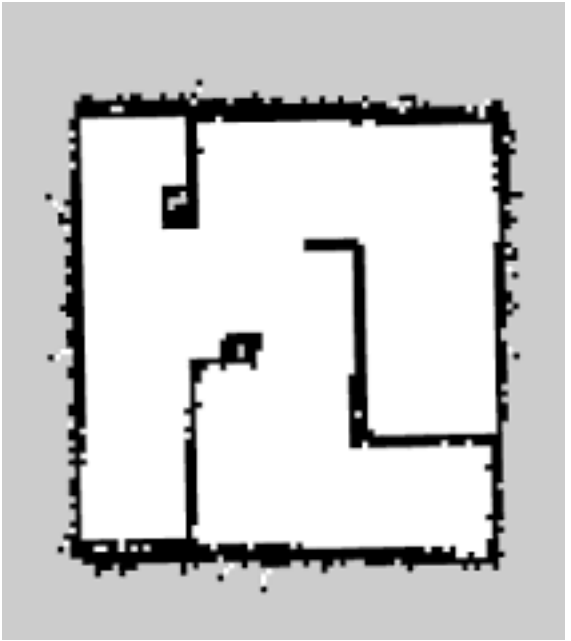
Grid Maps





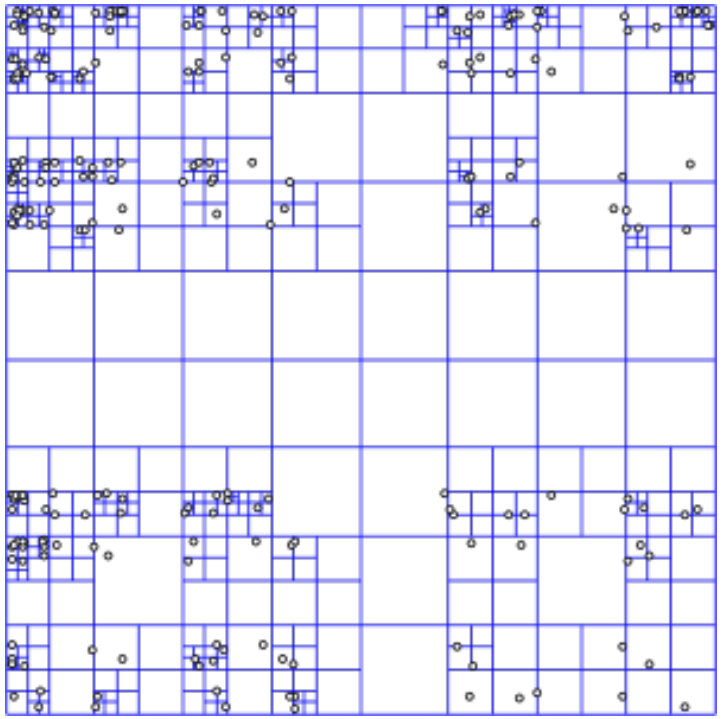
- Grid-map: 2D or 3D cartesian grid
- What to save in the cell:
  - Binary: Free; Occupied;
  - Colored: +Unknown; +Searched; +Path; …
  - Probability of being occupied 0…1.0
  - Height above ground: Elevation Map
  - Cost map: used for planning (covered later)





https://emanual.robotis.com/docs/en/platform/turtlebot3/slam/

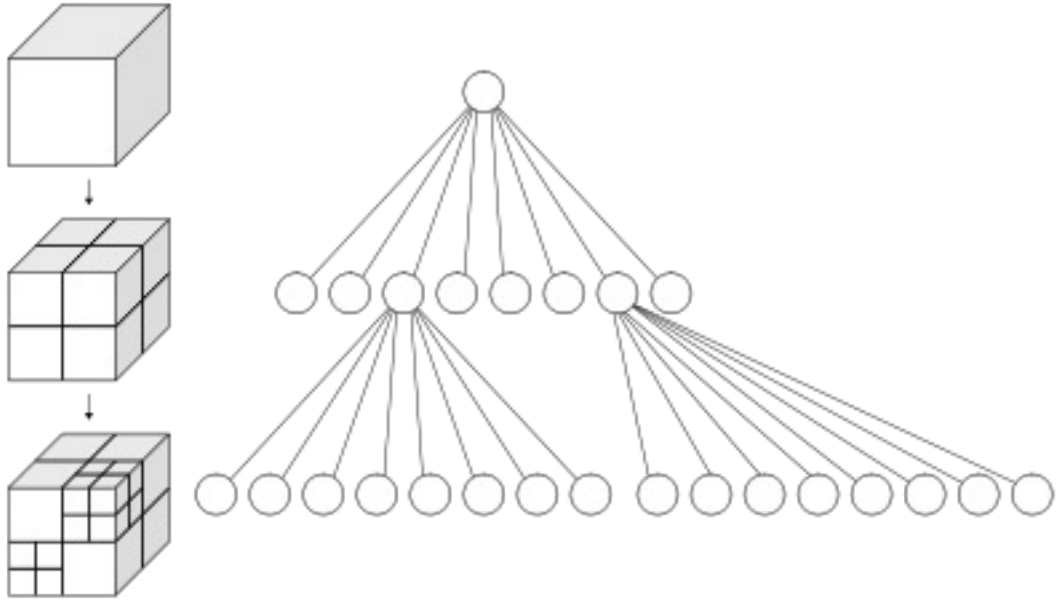
Adaptive Cell Decomposition

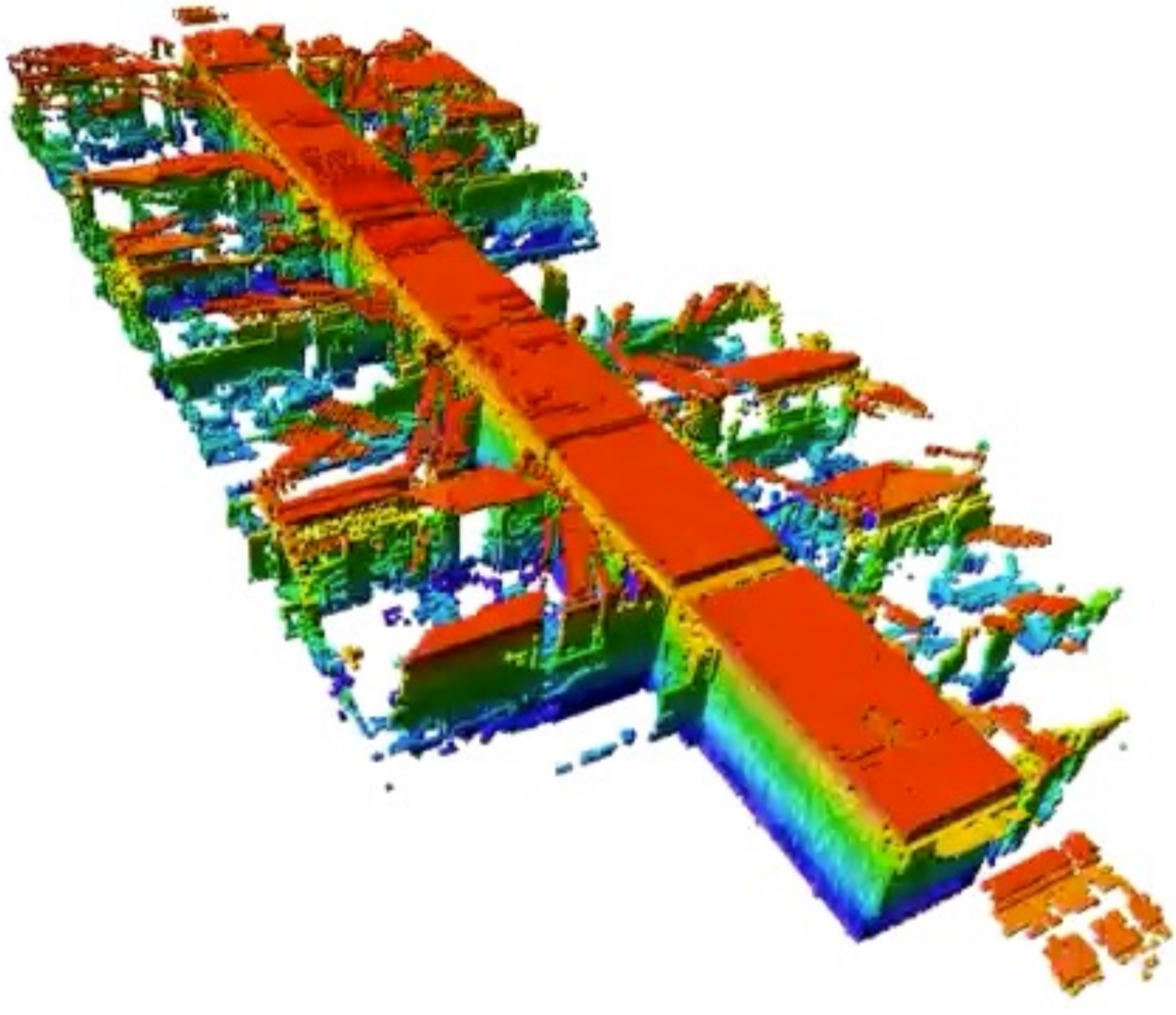


- Quad-tree
  - 2D map/ grid is recursively divided into 4 smaller cells
  - Only cells with different values/ points get divided further =>
  - Compact representation of big space (if many cells stay merged)

- Oct-tree
  - 3D grid divided into 8 smaller cells
  - Very compact! (There is lots of free space!)

- OctoMap: probabilistic oct-tree!
  - http://octomap.github.io
  - Good support in ROS (e.g. MoveIt!)





Multi-level surface maps

- 2D grid map
- Each cell can store multiple levels (saves mean and variance of height plus depth for each level)
- Useful for terrain classification and planning

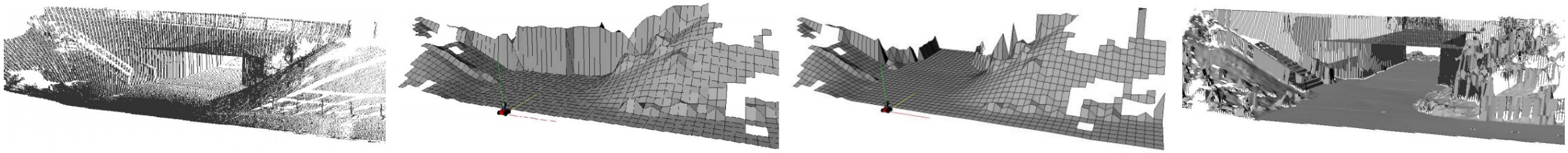


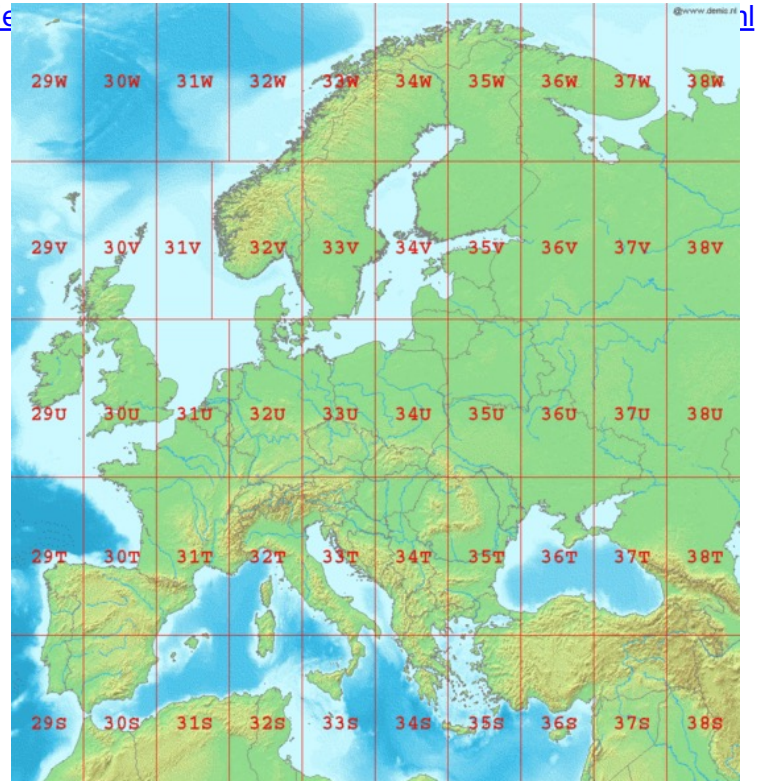
Fig. 1. Scan (point set) of a bridge (a), standard elevation map computed from this data set (b), extended elevation map which correctly represents the underpass under the bridge (c), and multi level surface map that correctly represents the height of the vertical objects (d)

Big Maps/ GPS

- World is NOT flat! =>
- Have to project 2D cartesian map onto a sphere!
- Different Projections, e.g. Mercator (preserves local directions and shapes, but not sizes)
- GPS polar coordinates: longitude & latitude of earth
- Universal Transverse Mercator (UTM): specify 60 planes on the surface of earth; specify location as Cartesian x,y,z coordinates on those planes
- World Geodetic System WGS 84 datum: used by GPS: model lang and lat to ellipsoid (Earth is not a perfect sphere!) from center of gravity …
- China: GCJ-02: WGS 84 plus random offsets (about 300-500m) (for national safety) – makes robotics difficult :/

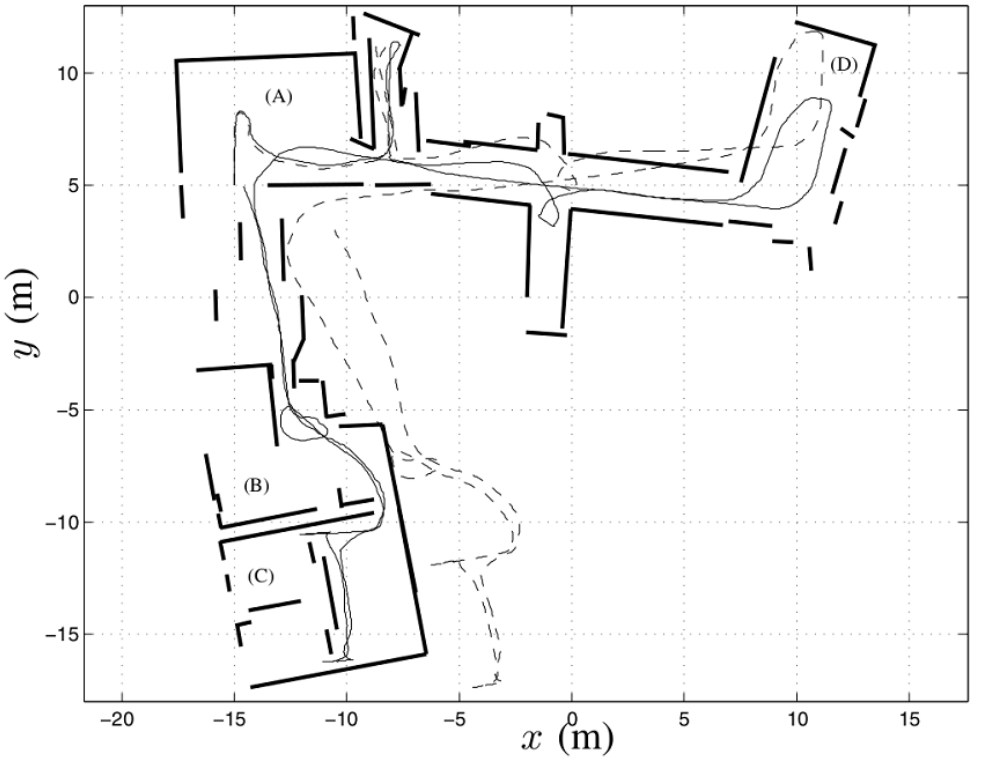
  地形图非线性保密处理算法
- Baidu: BD-09 further offsets (so competitors don't copy)

https://www.livescie… …l

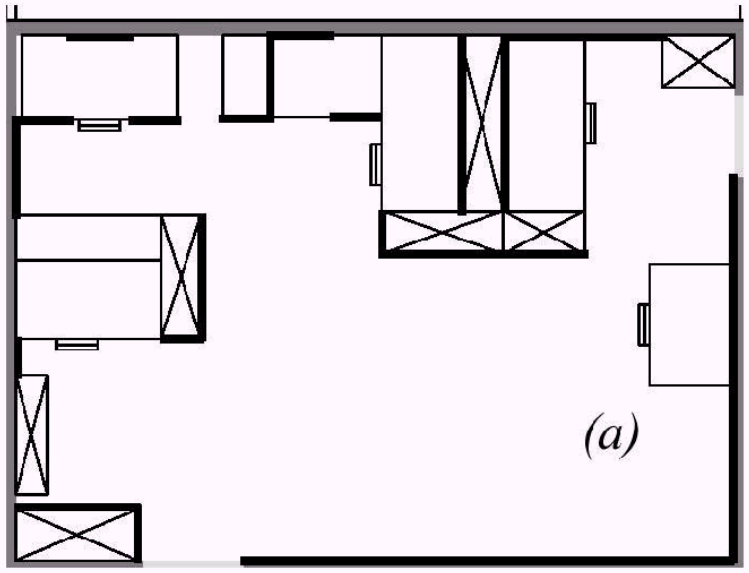


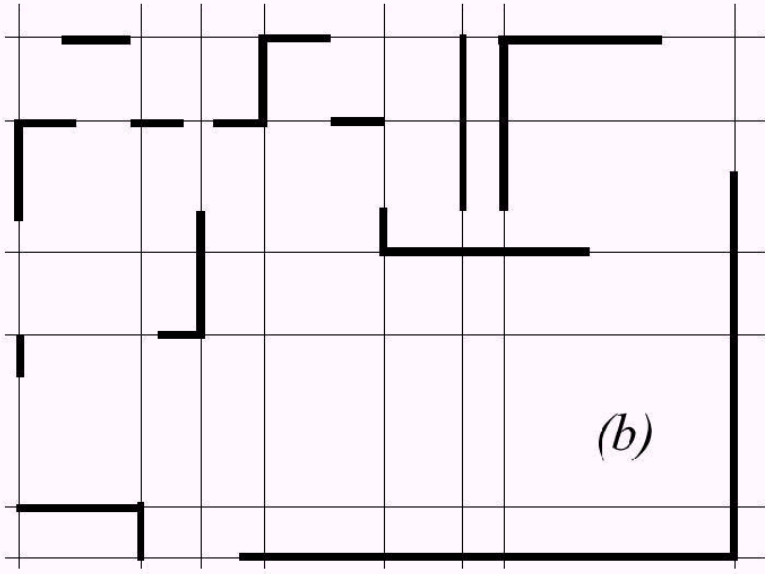
Line Map

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



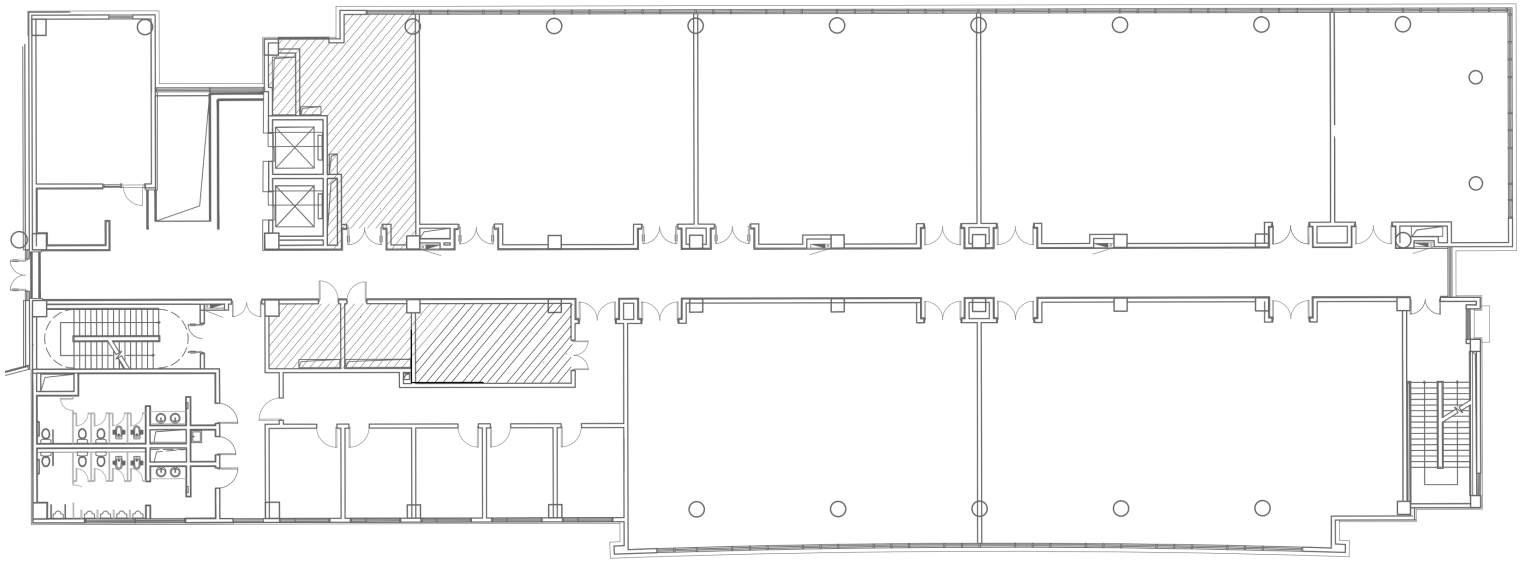
- Finite lines (a)
- Infinite lines (b)



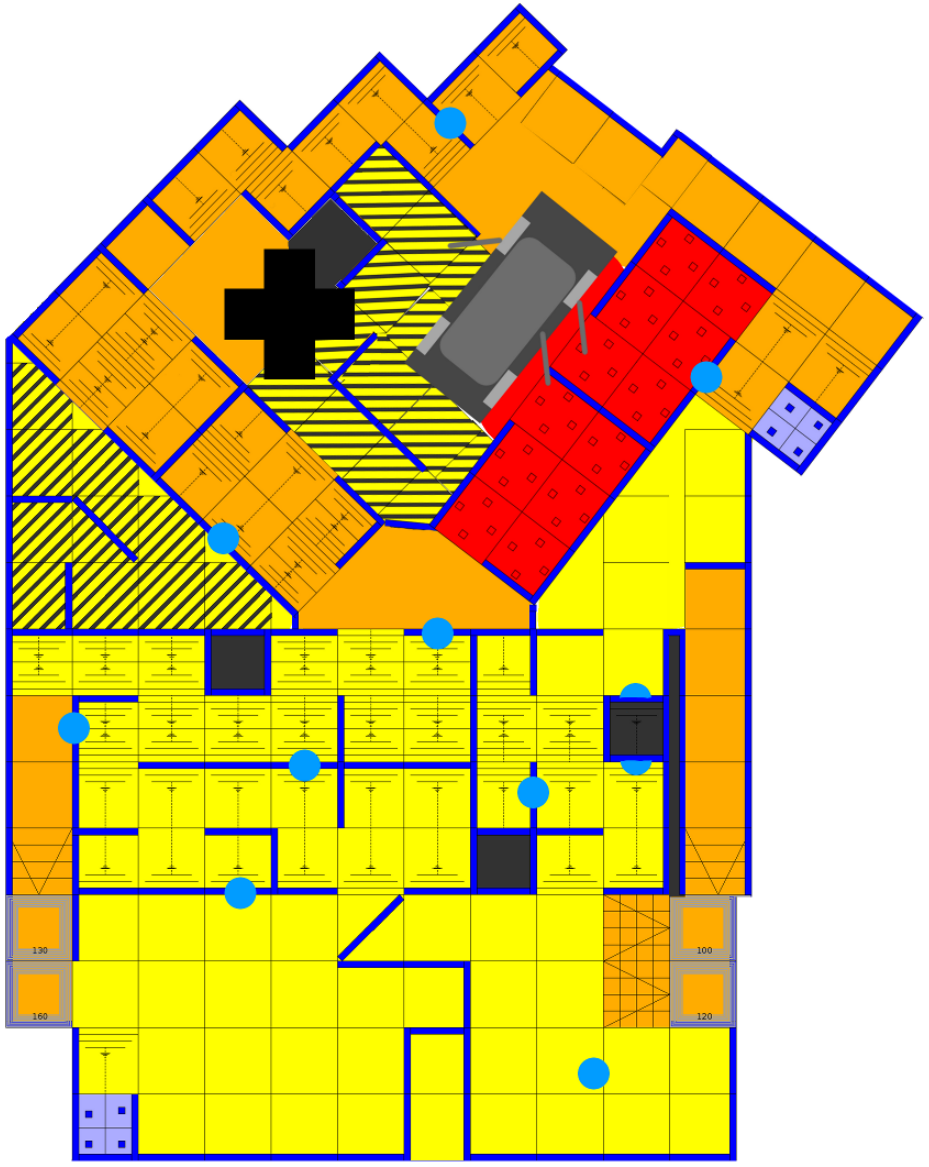


- Very compact
- Can do scan-matching (e.g. against laser scan)

Ground Truth Maps

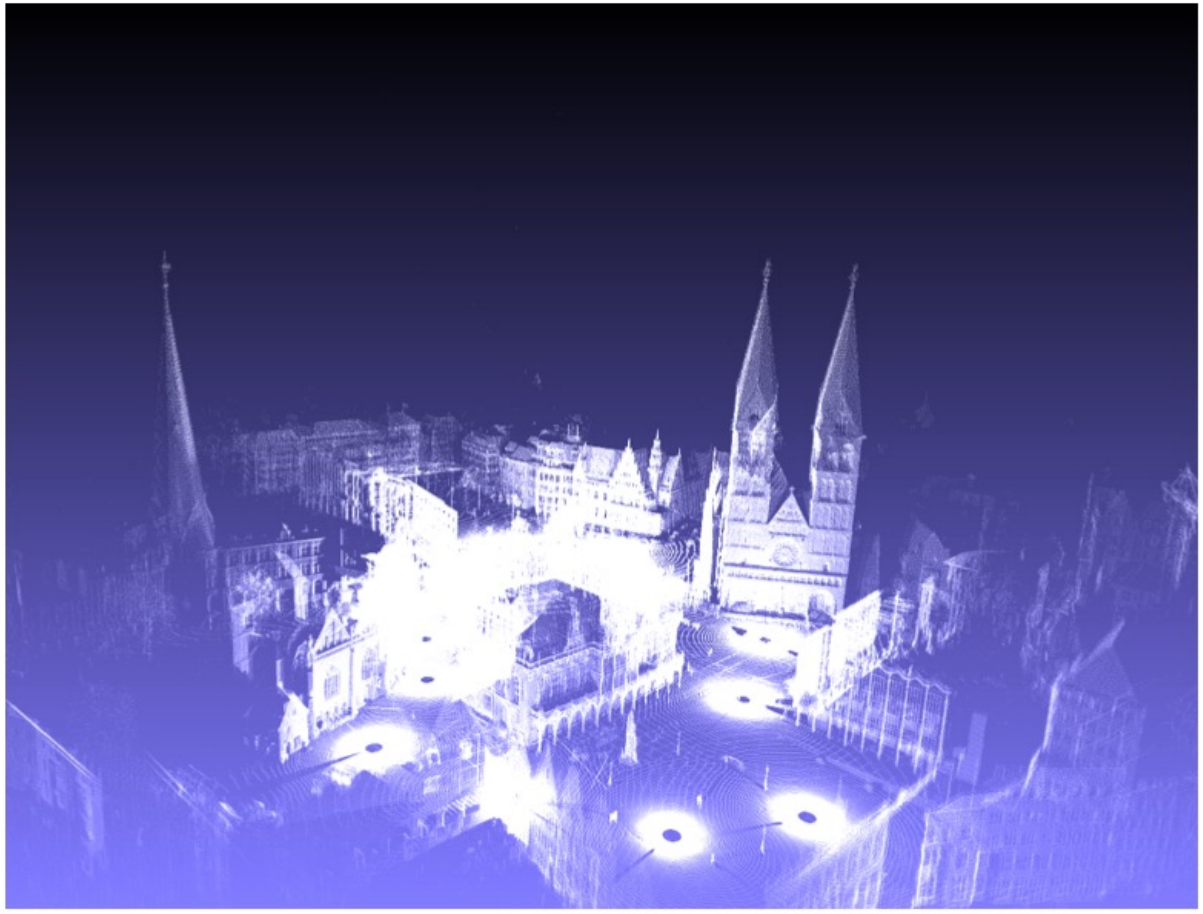


CAD drawing (STAR Center)
Vector format (lines, circles, …)

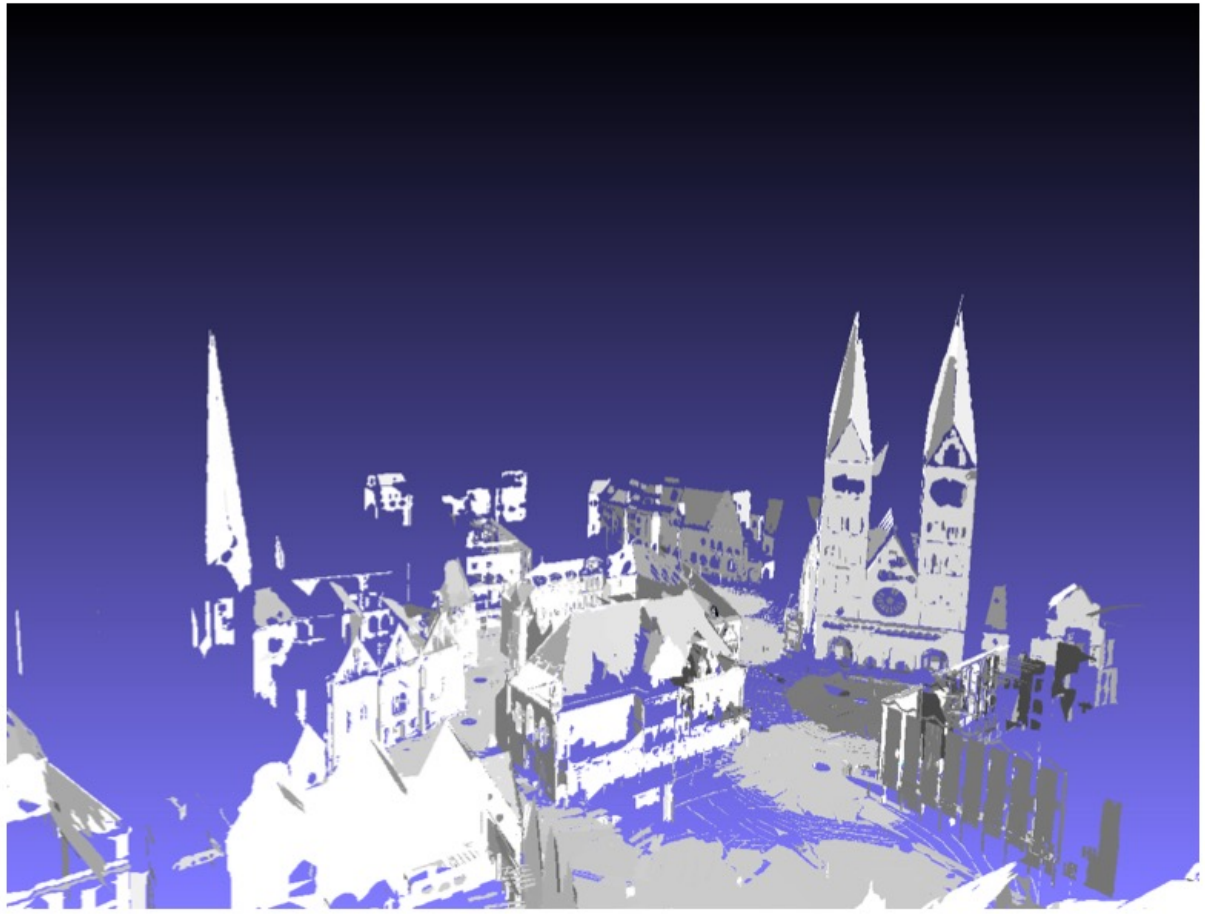


Grid Map (from vector map)

Bremen: 3D Point Cloud & 3D Plane Map

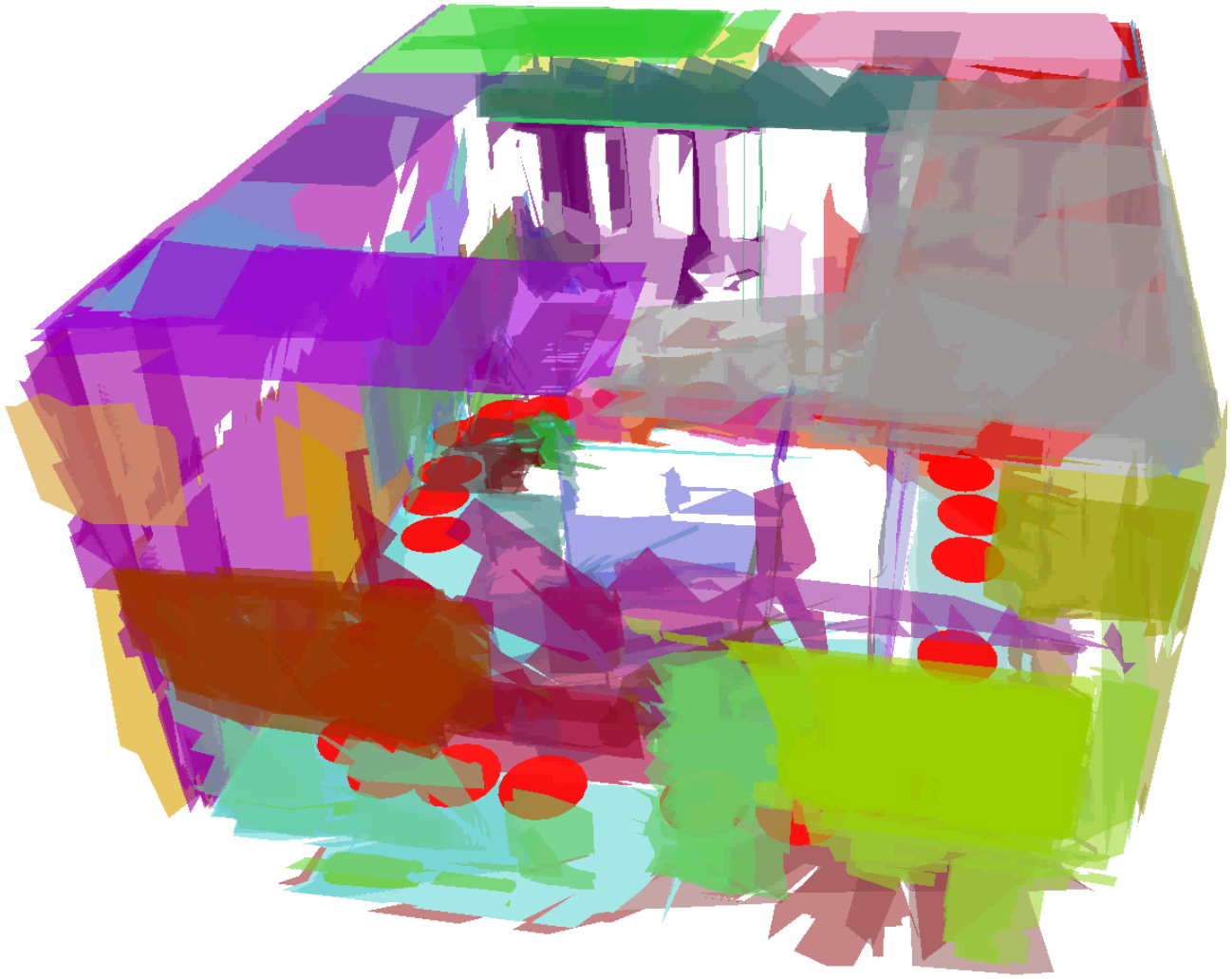


(a) 3D Point Cloud



(b) 3D Plane Map

Plane map: 29 poses; each with several planes



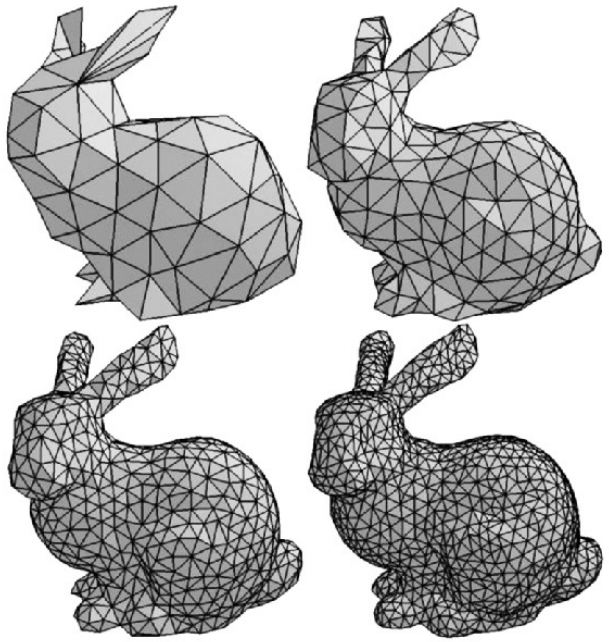
Vector Maps

- 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, …)

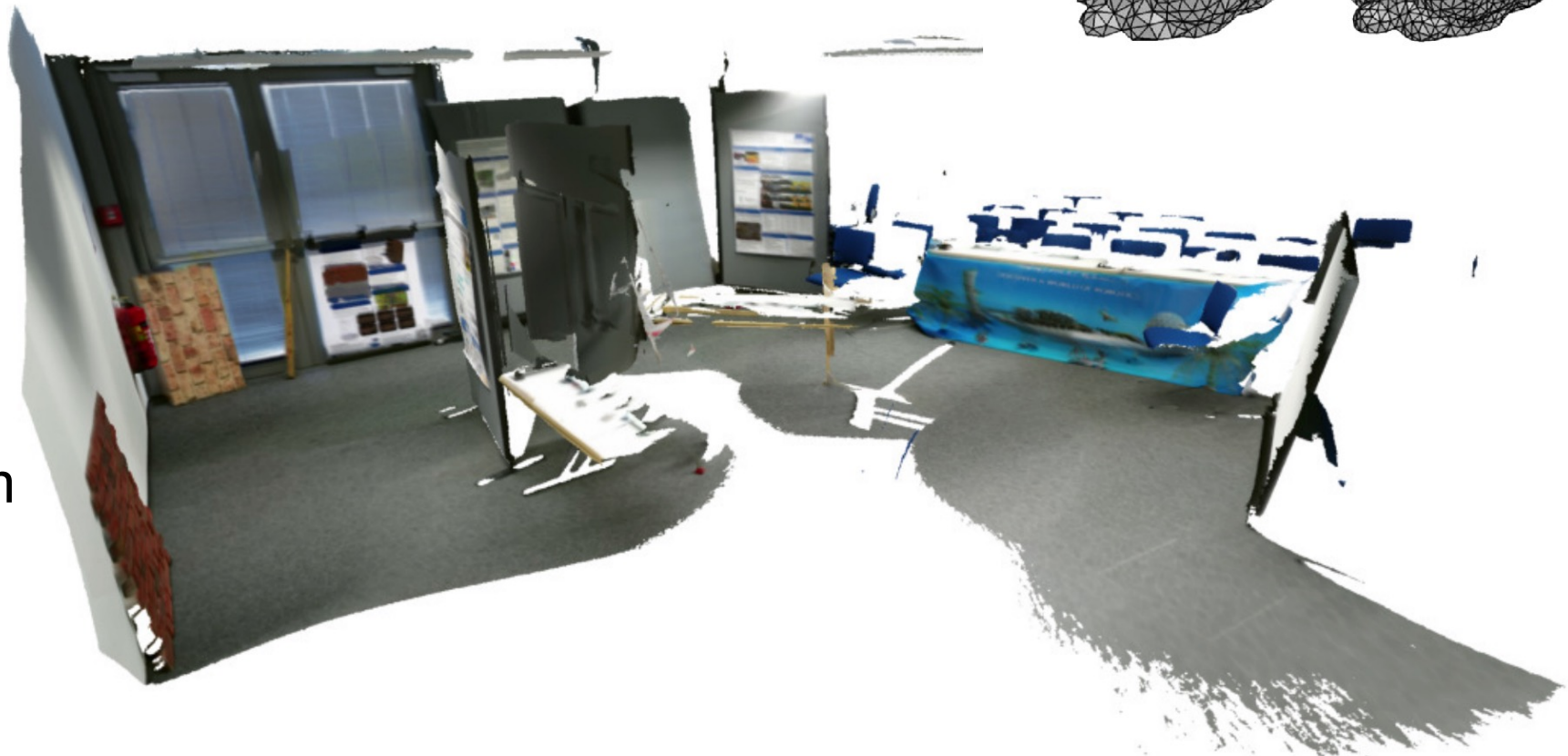
Mesh



- Often build via Signed Distance Field

- Close relation to 3D reconstruction from Computer Vision

- Often with texture (RGB information from camera)

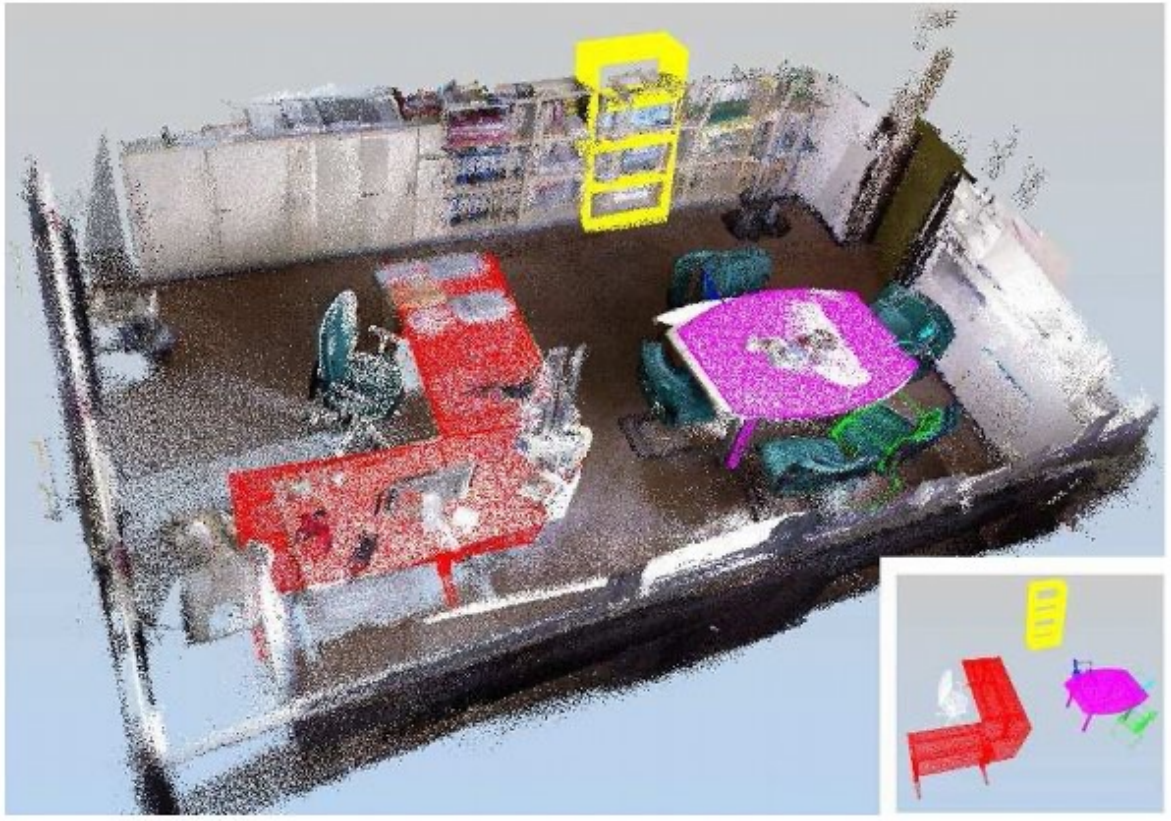


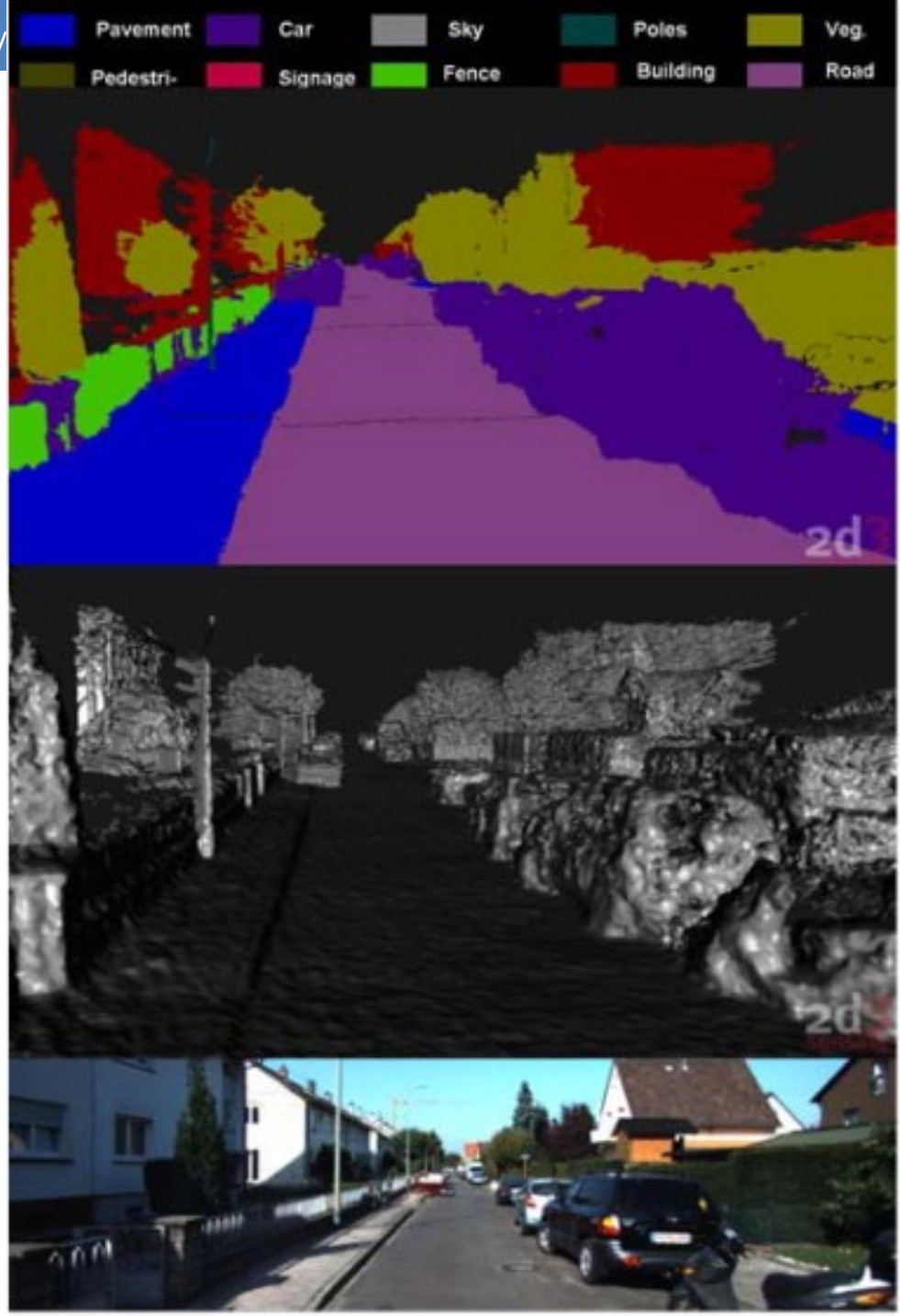
Stereye: Mesh Simplification

- 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

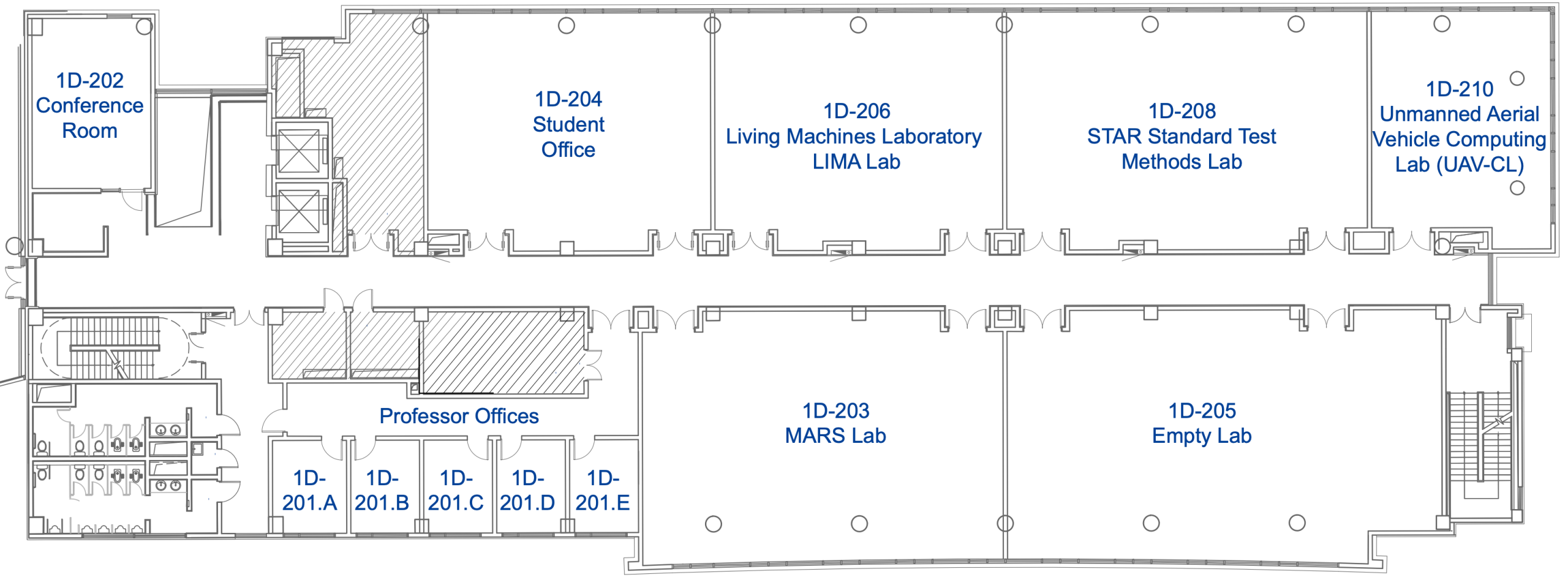
Semantic Map

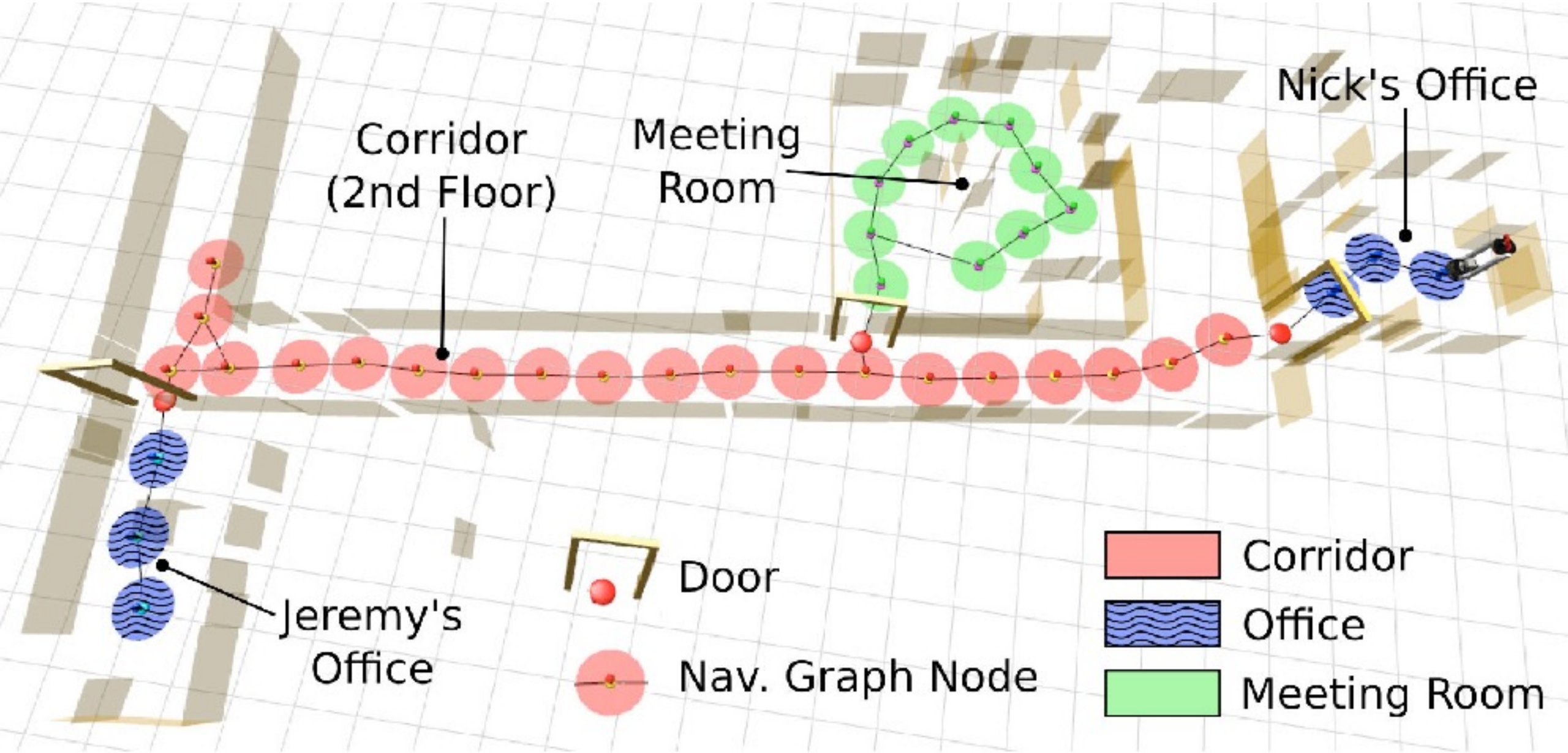
- Semantic Segmentation
  - In room (e.g. detect furniture)
  - Outdoors





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

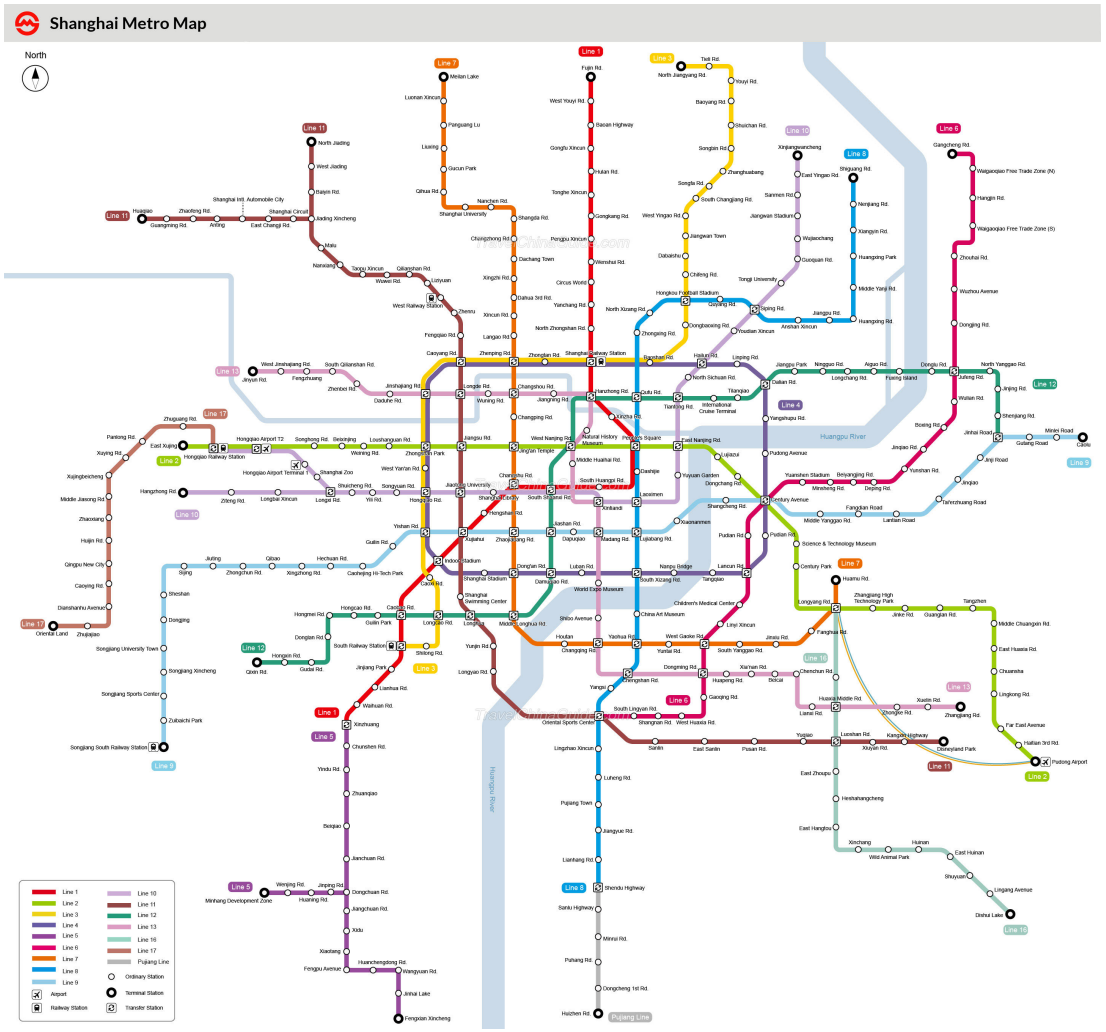
- Assign labels to data
- Segmentation: automatically group data (e.g. points) according to their semantic class

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

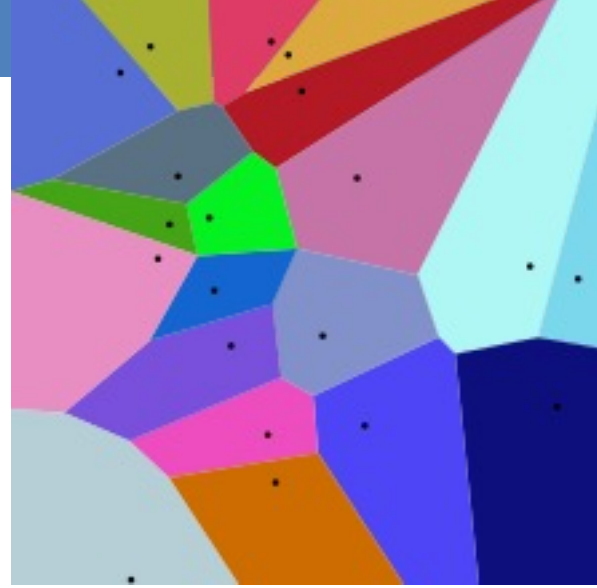
- Applications:
  - Human Robot Interaction (“go to kitchen”)
  - Scene understanding
  - Navigation (detect road; detect door)
  - Localization
  - …

Topologic Map

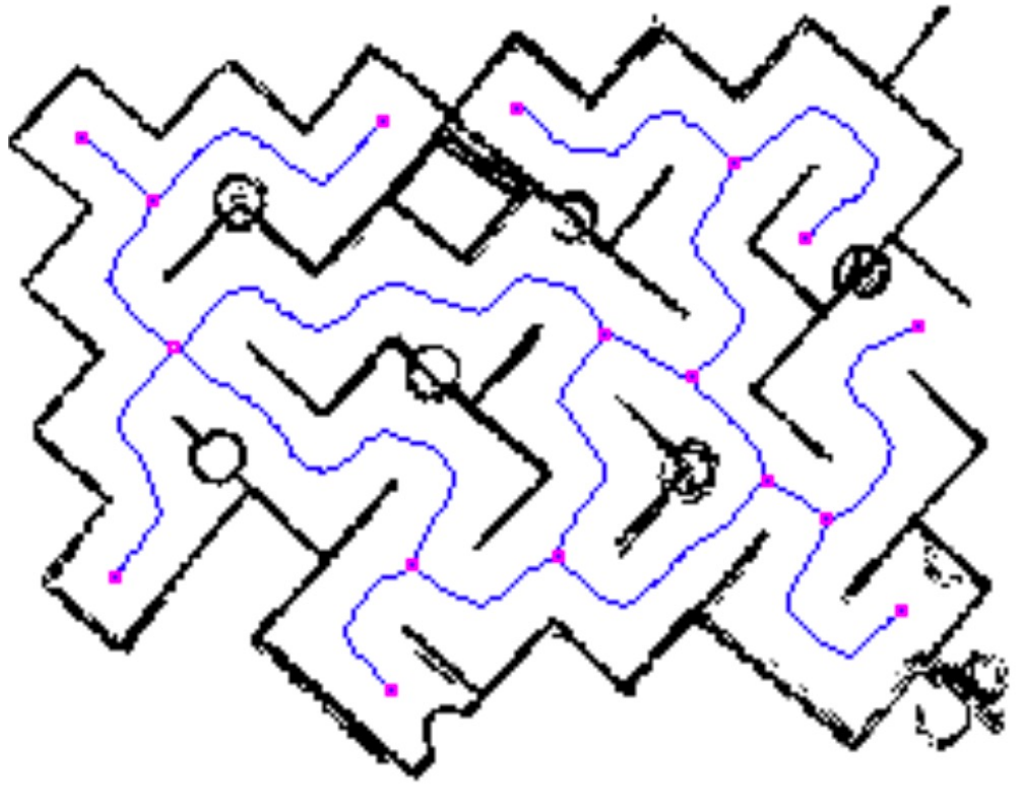
- A (undirected) graph
- Places (vertices) and their connections (edges)
- E.g. subway map of Shanghai: stations (vertices) and lines (edges)
- Do not have coordinates
- Topometric map: vertices and/ or edges are attributed with coordinates
- Very abstract – e.g. no obstacles anymore



Voronoi Diagram (-> Topology Graph)

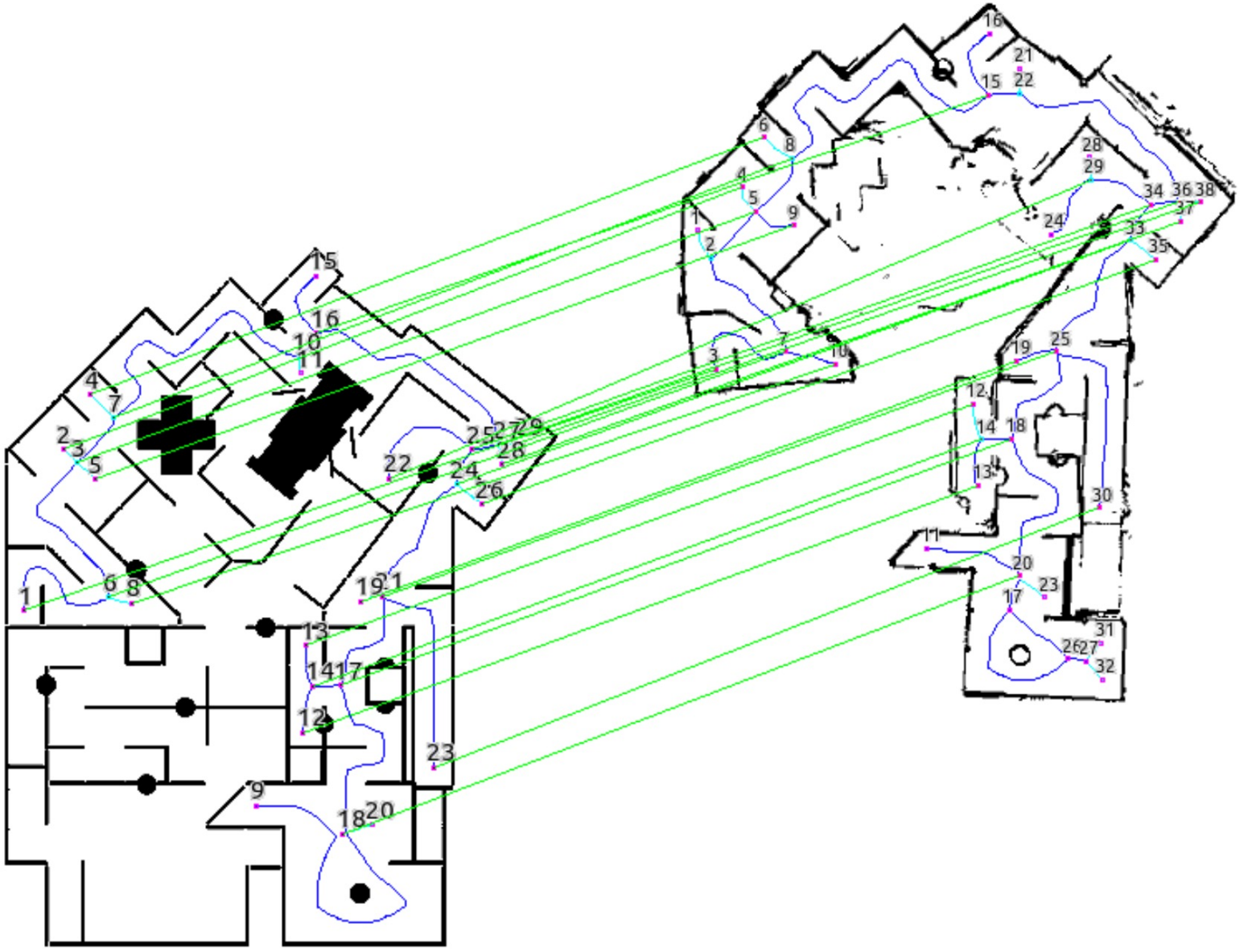


- Voronoi Diagram (VD): partition space such that edge is always equidistant to 2 closest obstacles.
- Topology Graph: vertices at junctions and dead ends
- Applications:
  - Very fast path planning
  - Human robot interaction (follow corridor, then go left)
  - Map matching

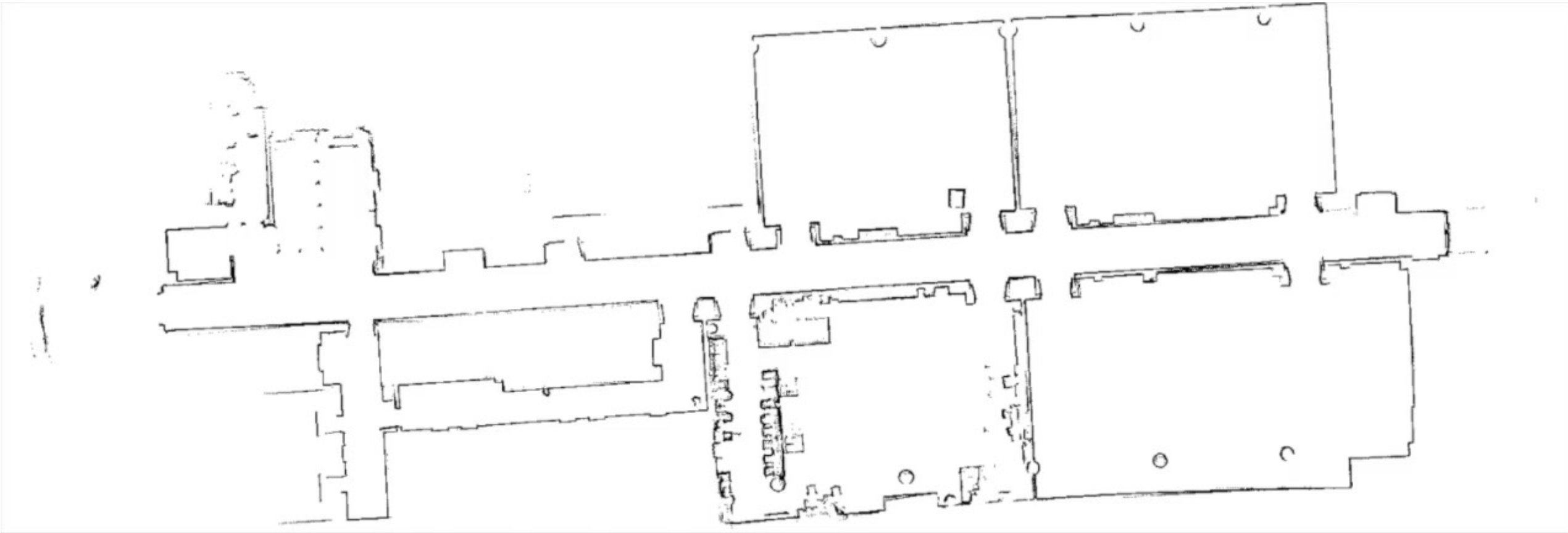


Map Matching

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

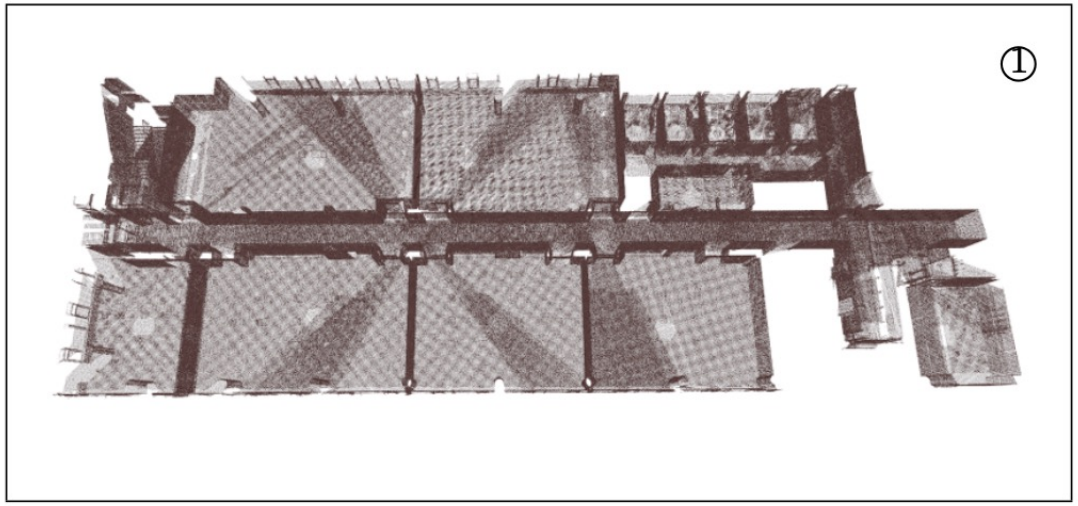


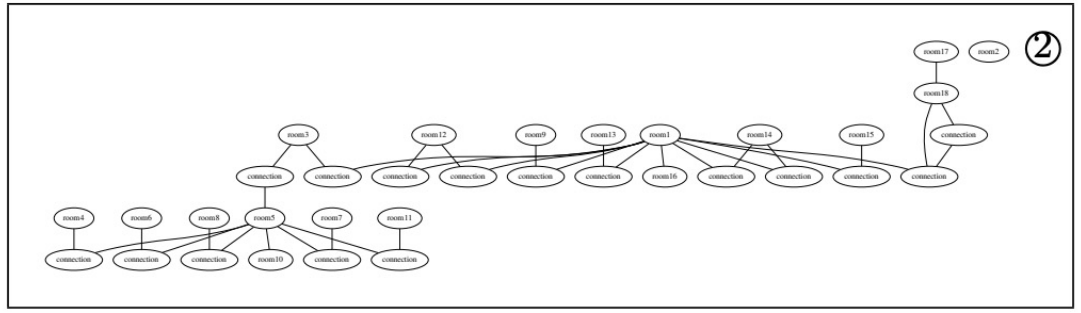
Area Graph: from 2D Grid Map to Topology Graph

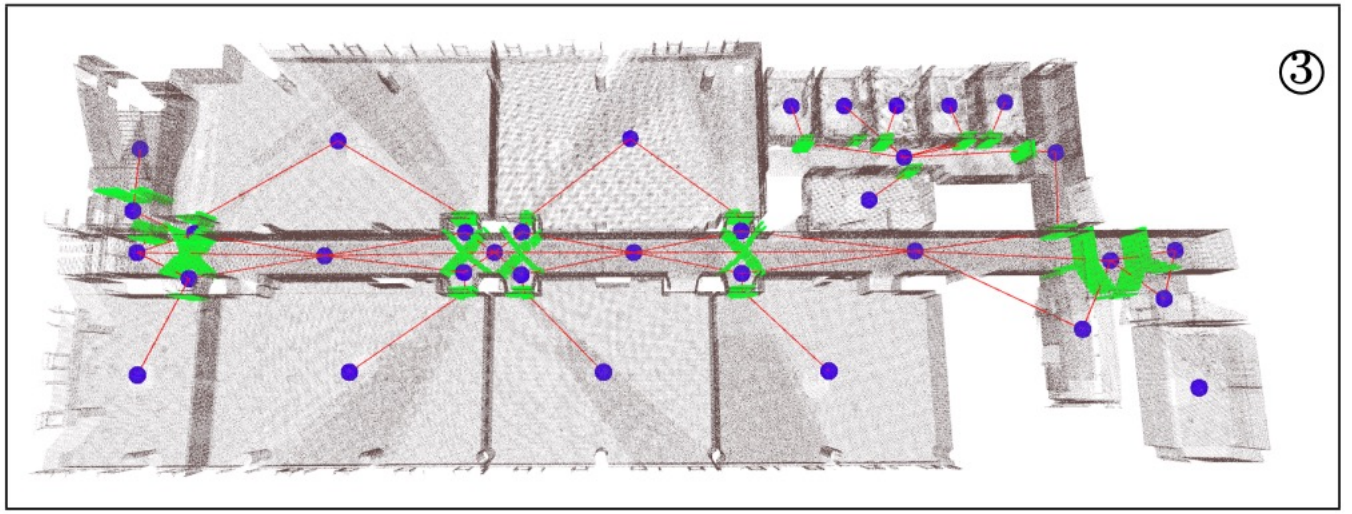


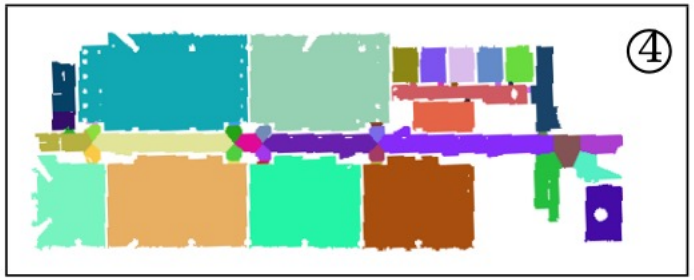
Topological Map in different Dimensions

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









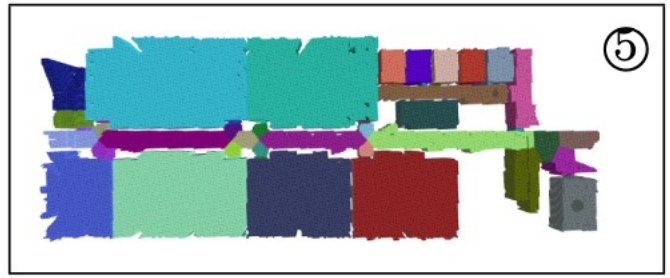
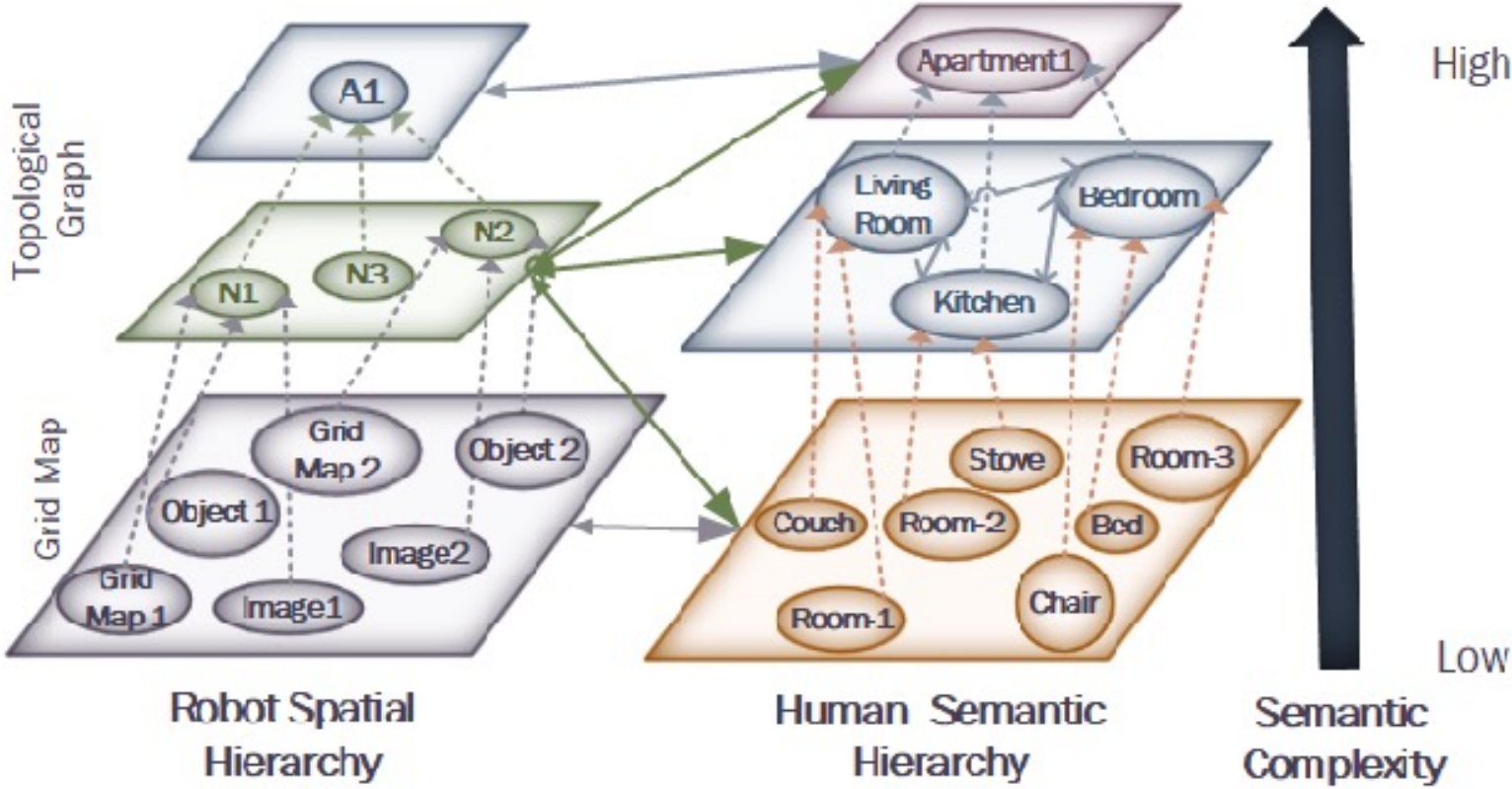


Fig. 1: Input 3D point cloud of 4 floors of a building. Below: results of our algorithm in different dimensions. From top to bottom, left to right are: 0D, 1D, 2D, 3D

Hierarchical Maps

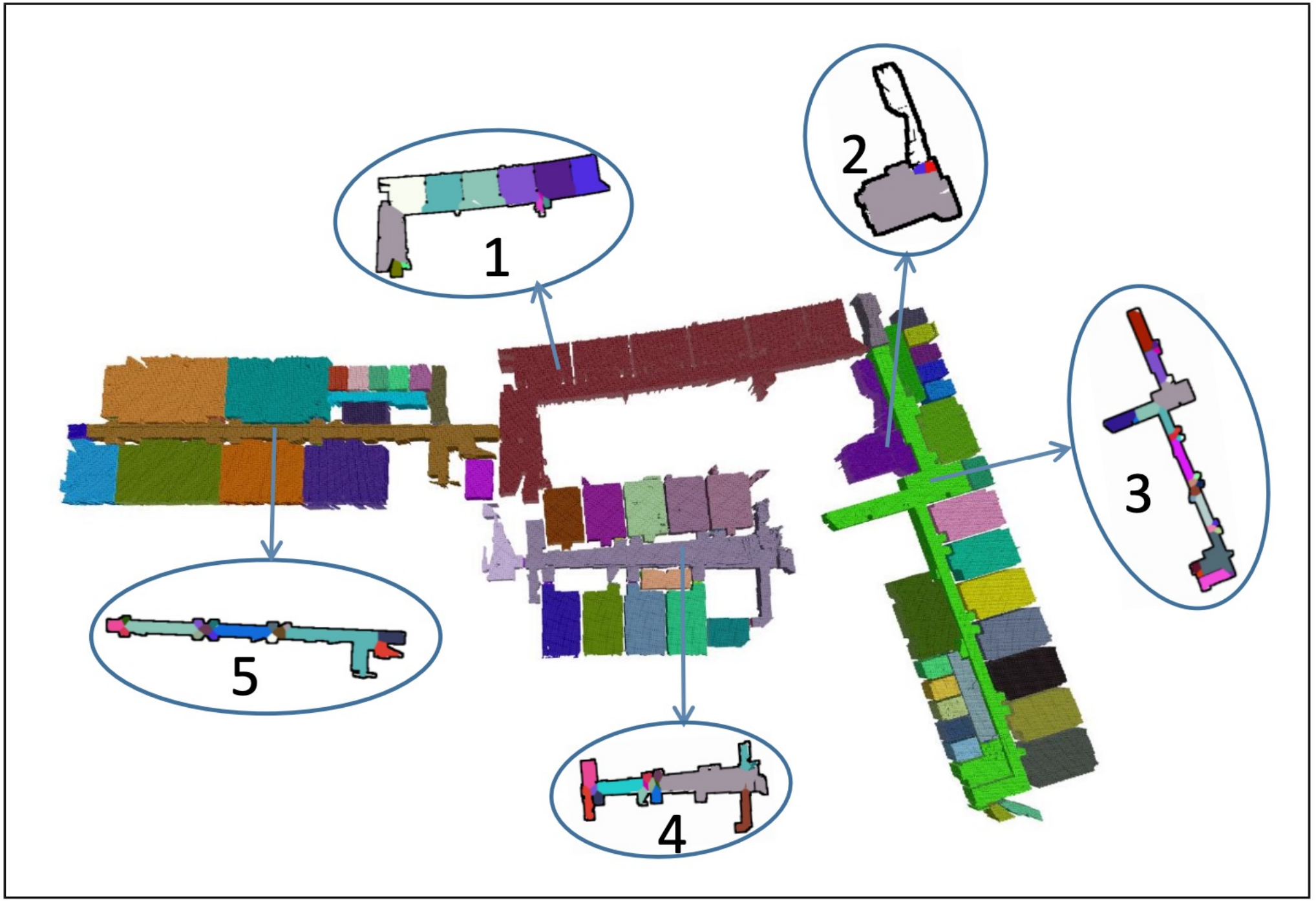
- Higher abstracted maps that contain lower ones with more details



- E.g. Grid map & Topological

- Useful for very fast planning; Human Robot Interaction; …

- Another form: image pyramid in GIS (different scale images)


1
2
3
4
5

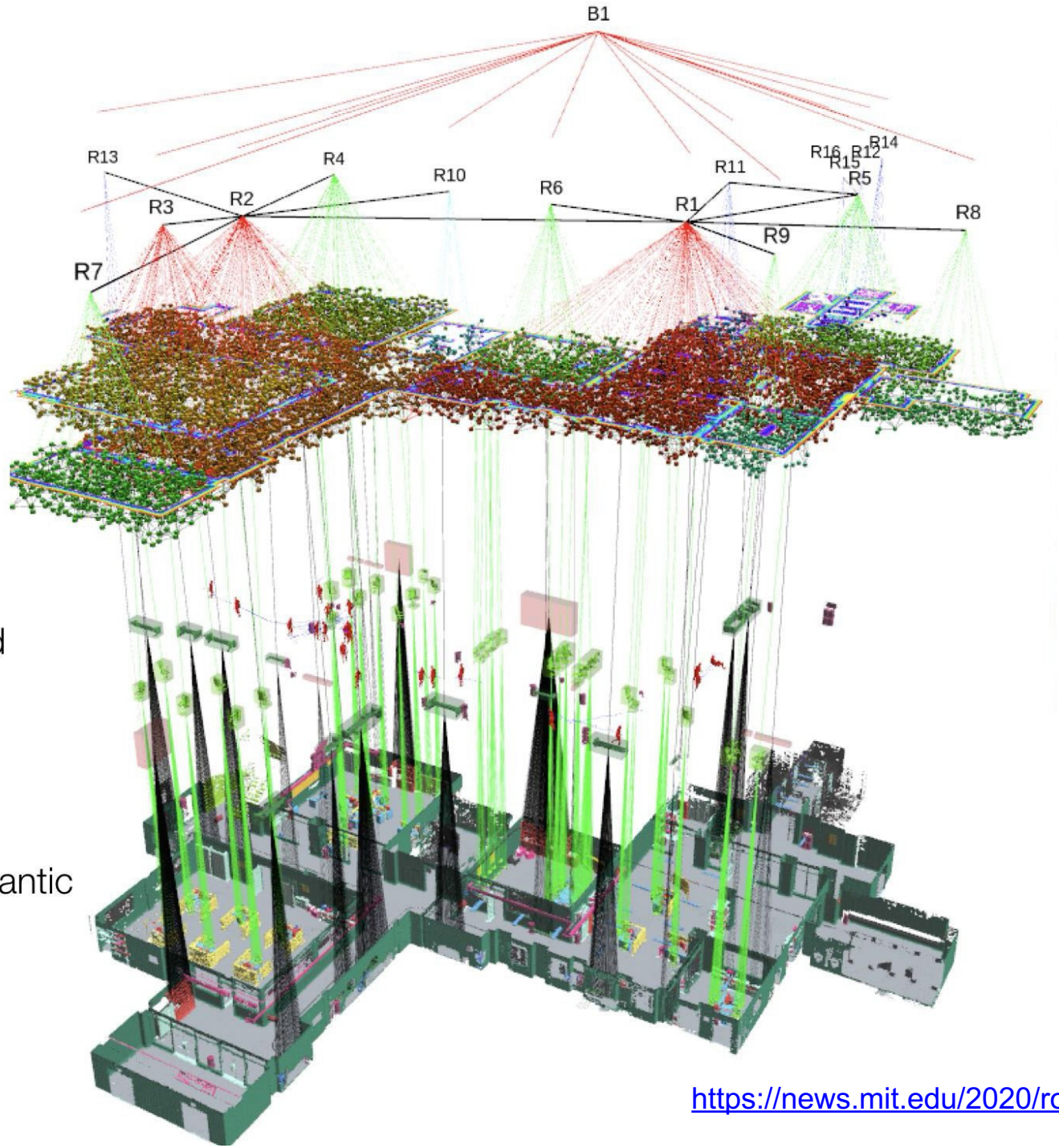
Layer 5:
Buildings

Layer 4:
Rooms

Layer 3:
Places and Structures

Layer 2:
Objects and Agents

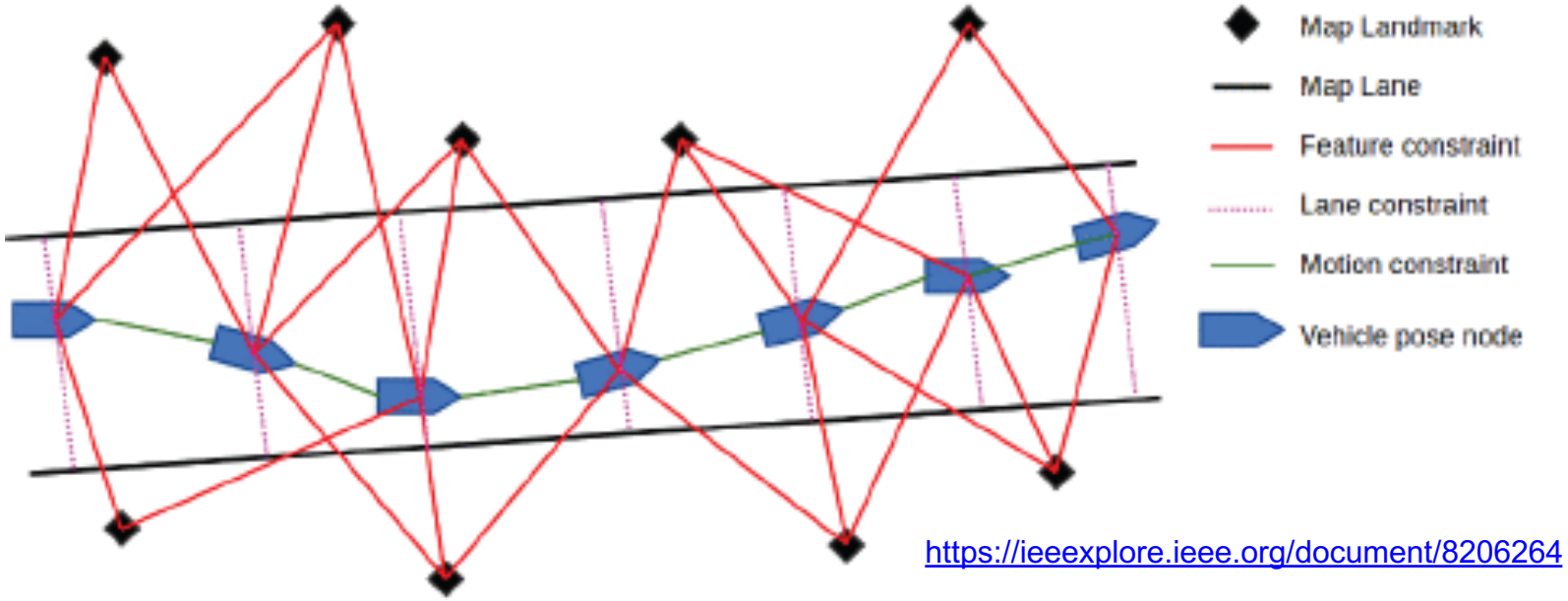
Layer 1:
Metric-Semantic Mesh



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

Pose Graph

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



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

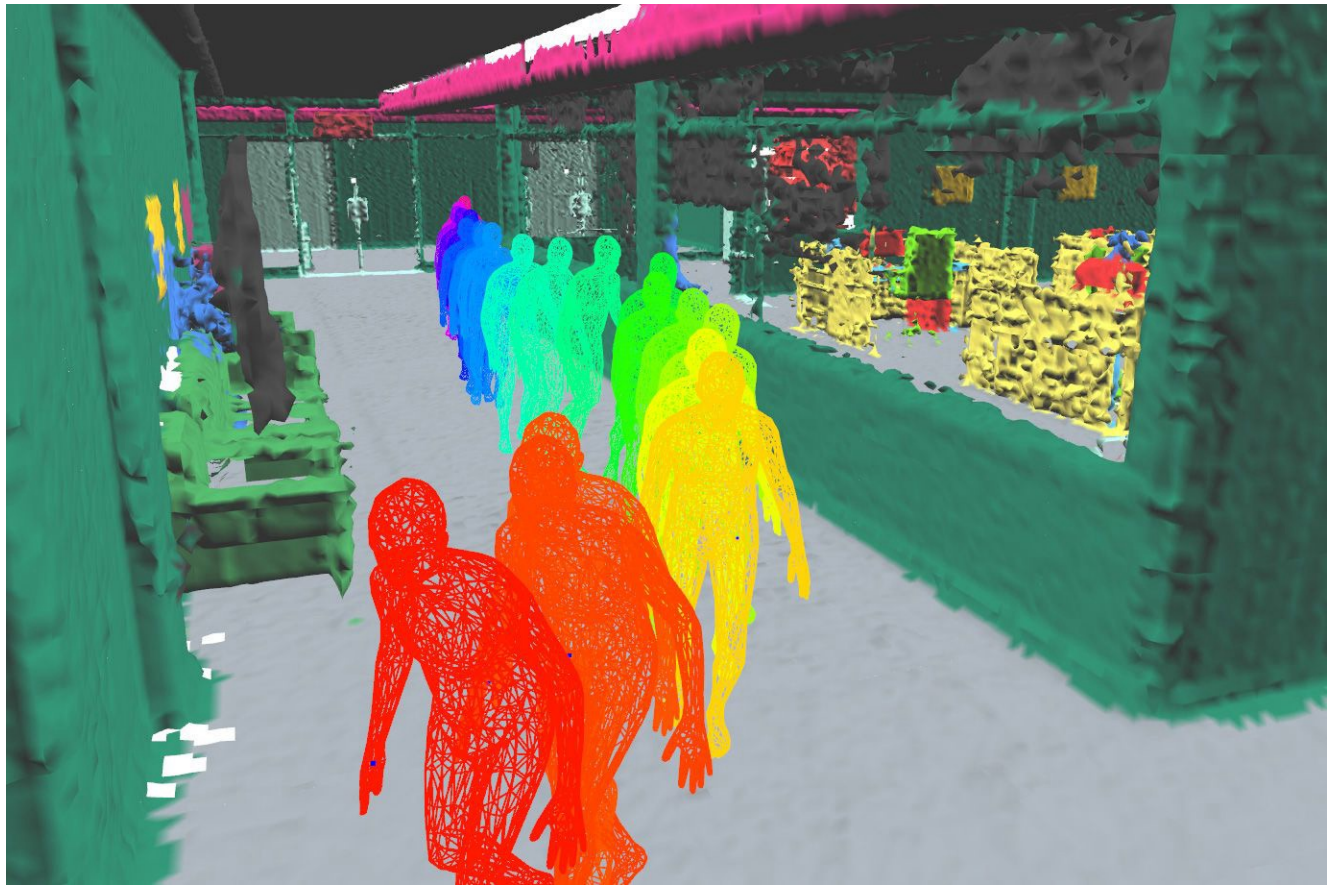
- Used for Simultaneous Localization and Mapping (more details later in course)

- 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

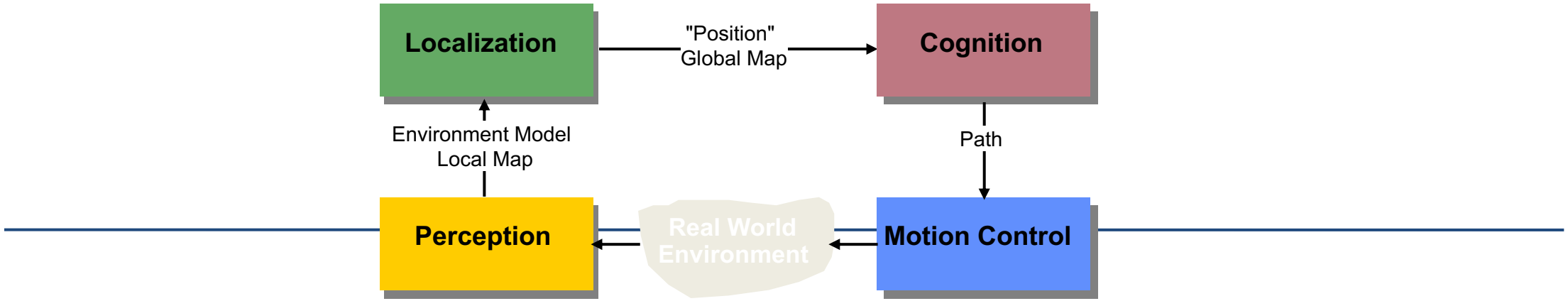
Other Models of the Environment/ Map Representations

- 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

- 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

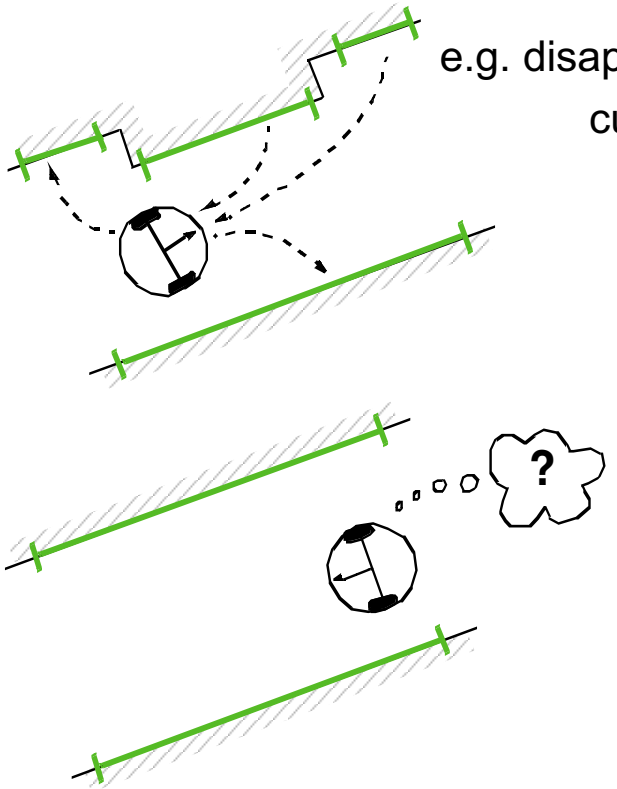
- Localization step may need the map (e.g. matching the scan against the map) => both should be done at the same time =>
- Simultaneous Localization and Mapping : SLAM



SLAM: SIMULTANEOUS LOCALIZATION AND MAPPING

Map Building: The Problems

1. Map Maintaining: Keeping track of changes in the environment

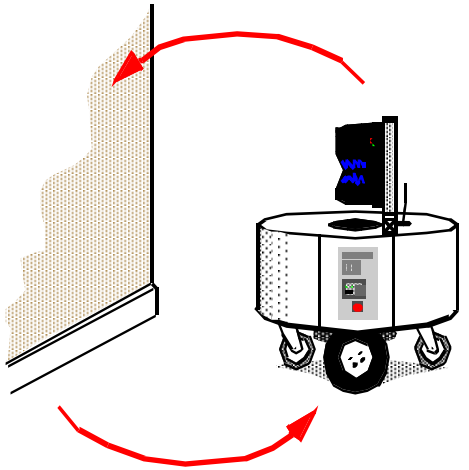


e.g. disappearing cupboard

- e.g. measure of belief of each environment feature

2. Representation and Reduction of Uncertainty

position of robot -> position of wall

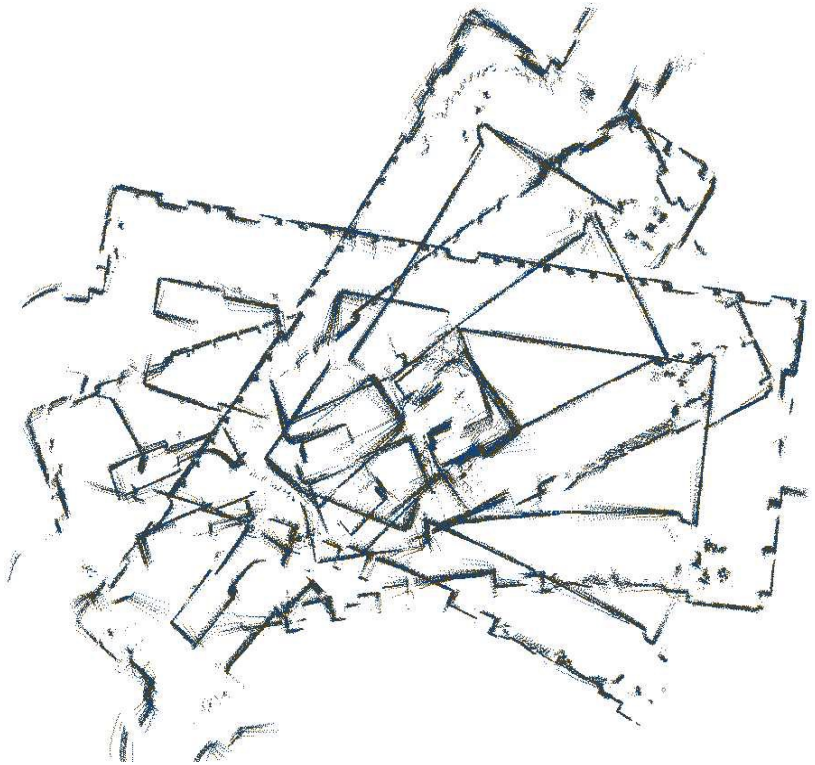


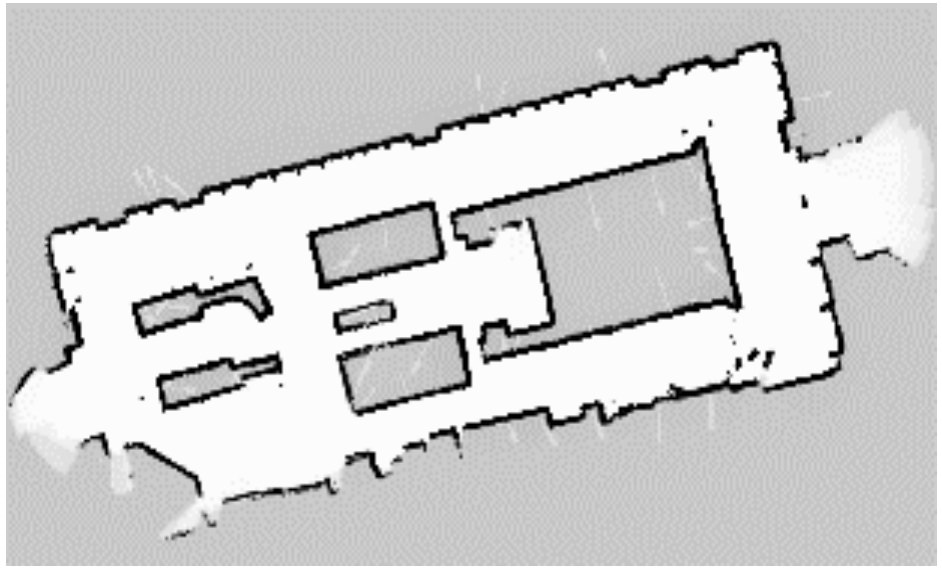
position of wall -> position of robot

- probability densities for feature positions
- Inconsistent map due to motion drift

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





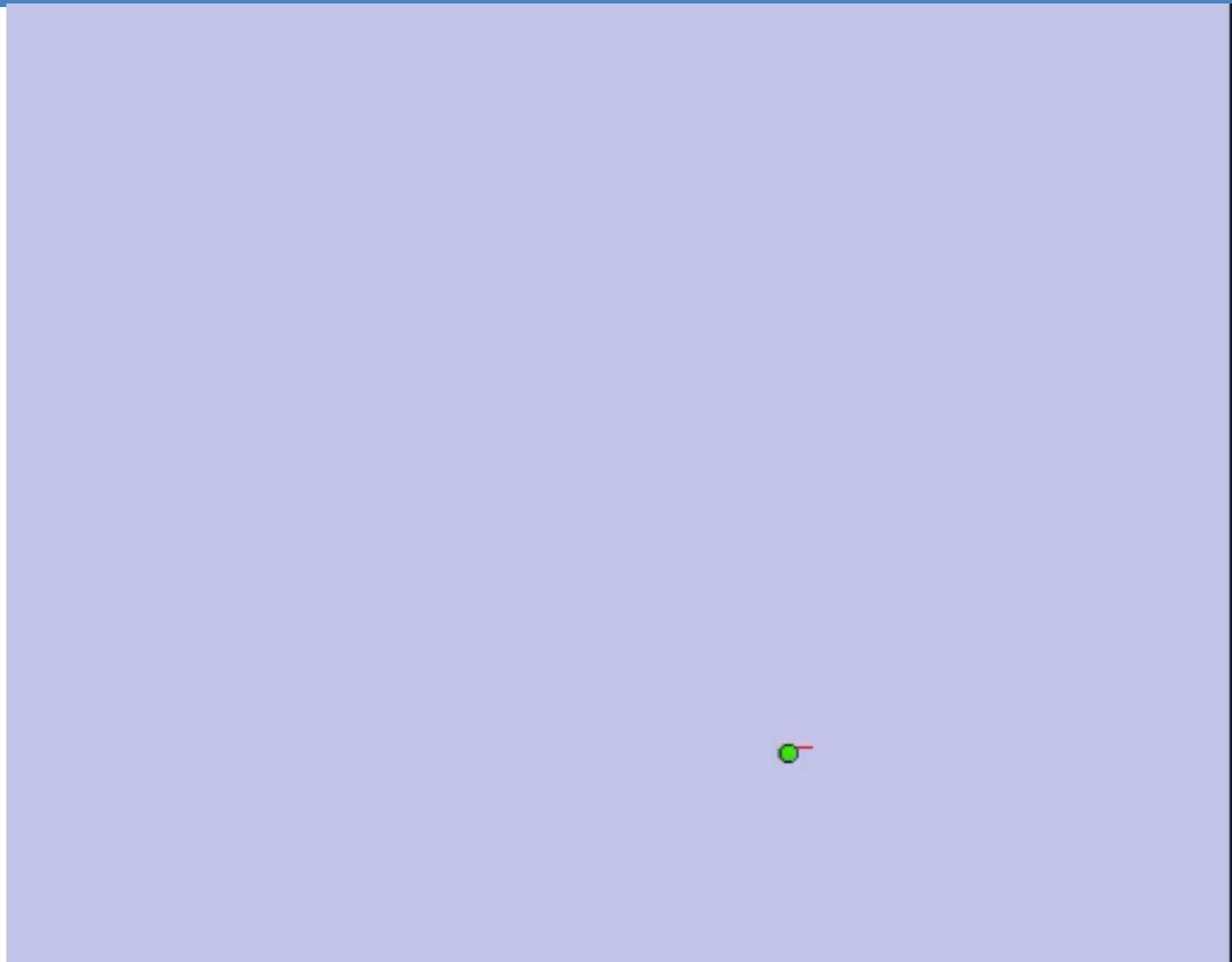
Raw Odometry

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

Courtesy of S. Thrun

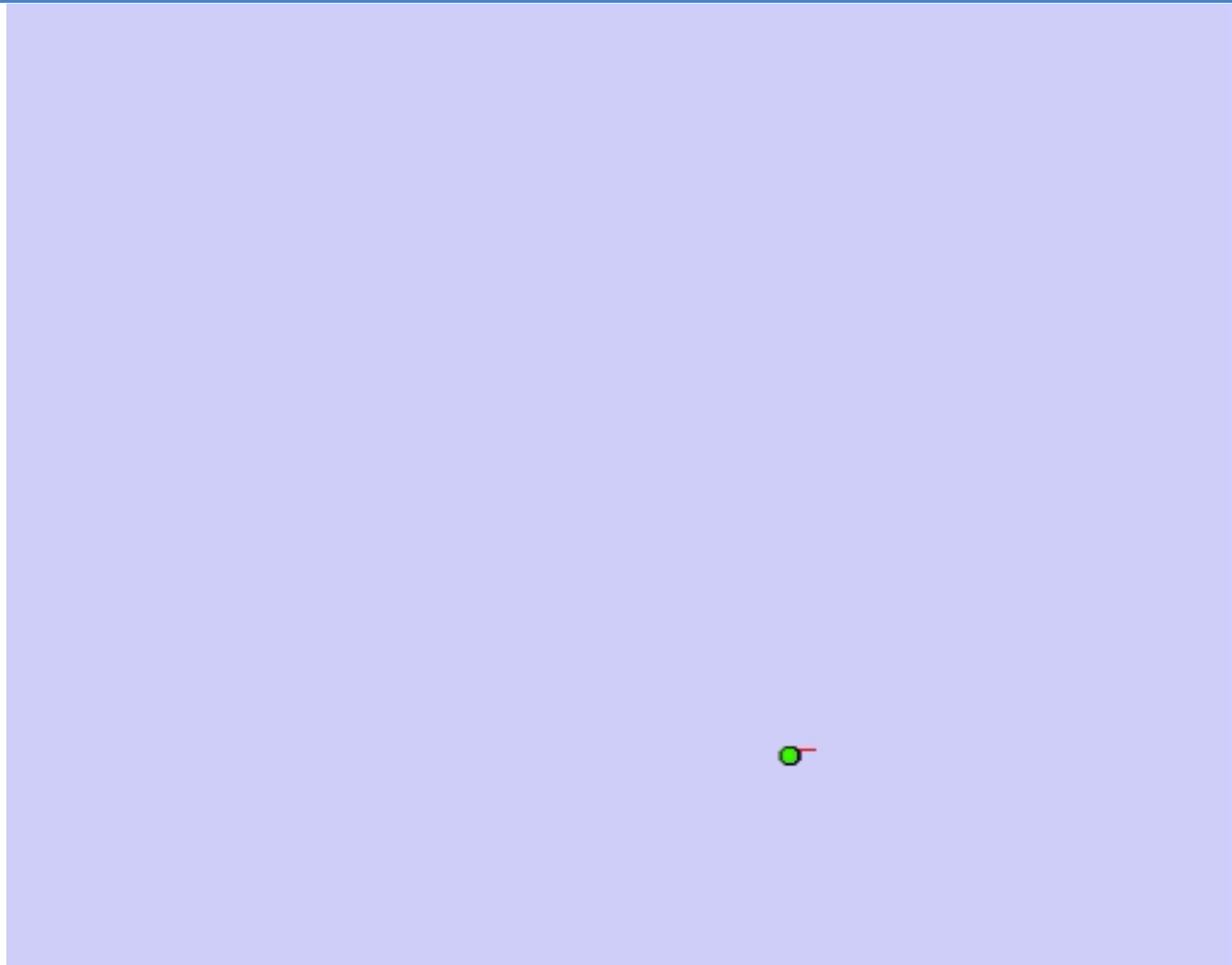
http://robots.stanford.edu/videos.html

Scan Matching: compare to sensor data from previous scan



Courtesy of S. Thrun

FastSLAM: Particle-Filter SLAM



Courtesy of S. Thrun

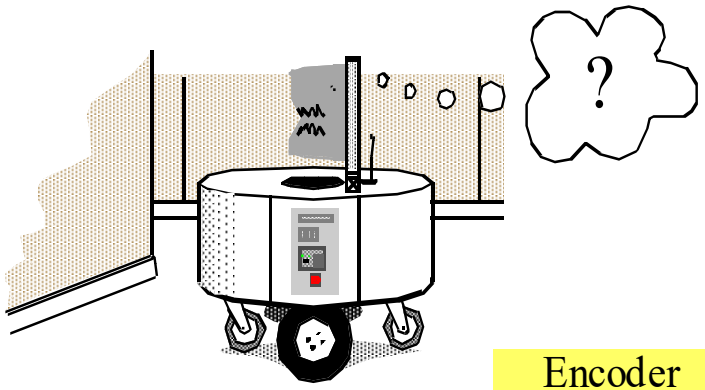
Scan Matching/ Registration

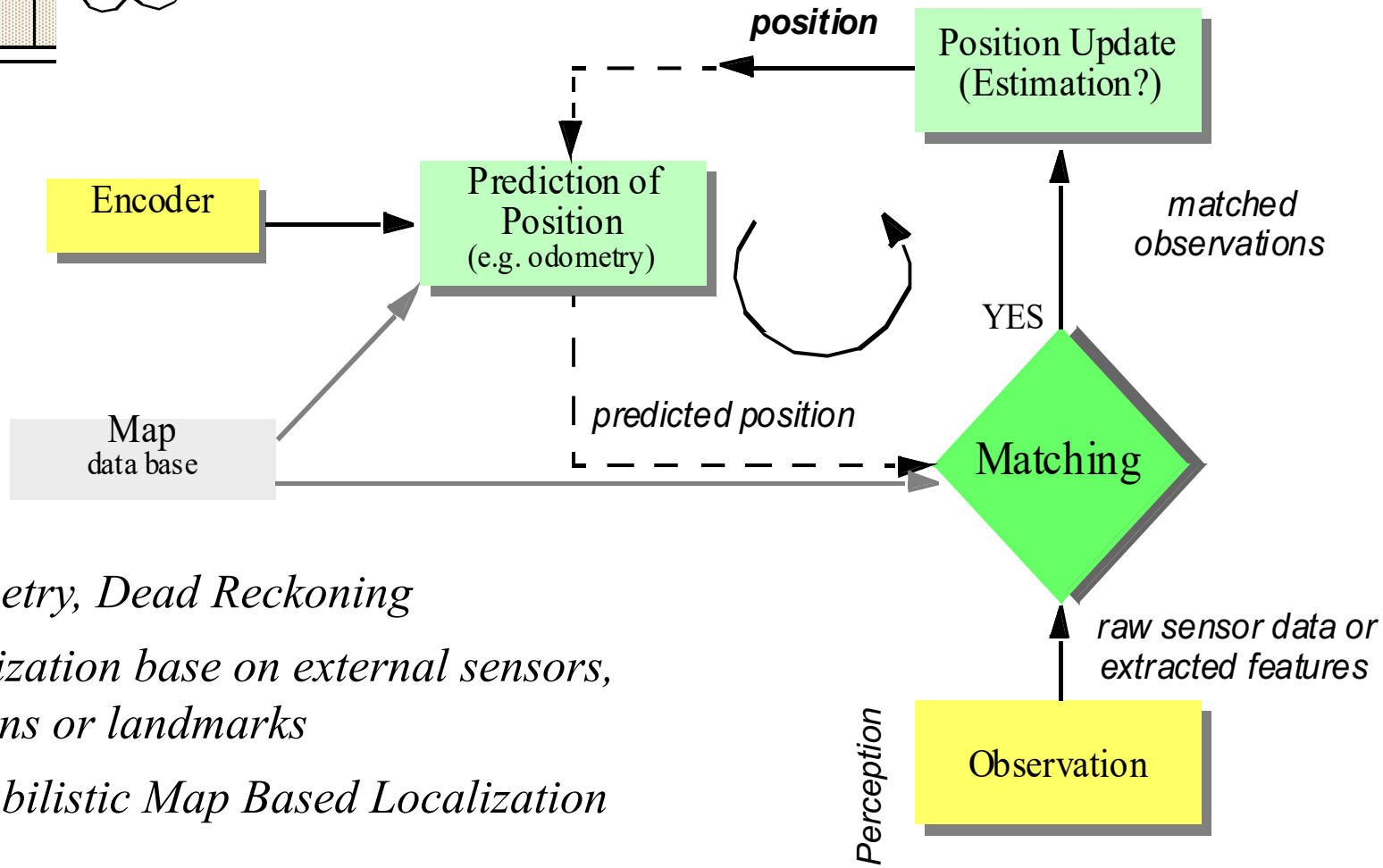
- Take one sensor scan
- Match against:
  - Another sensor scan
  - Against the map

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

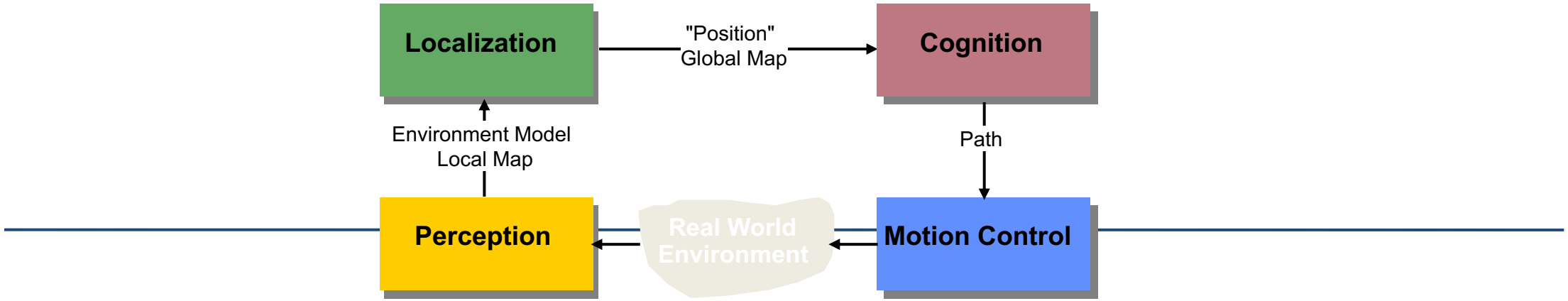
- Used for Localization: Next lecture

Map based localization

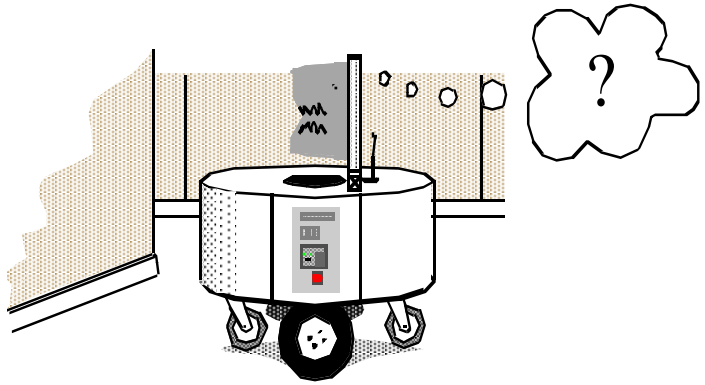




- *Odometry, Dead Reckoning*
- *Localization base on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*



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

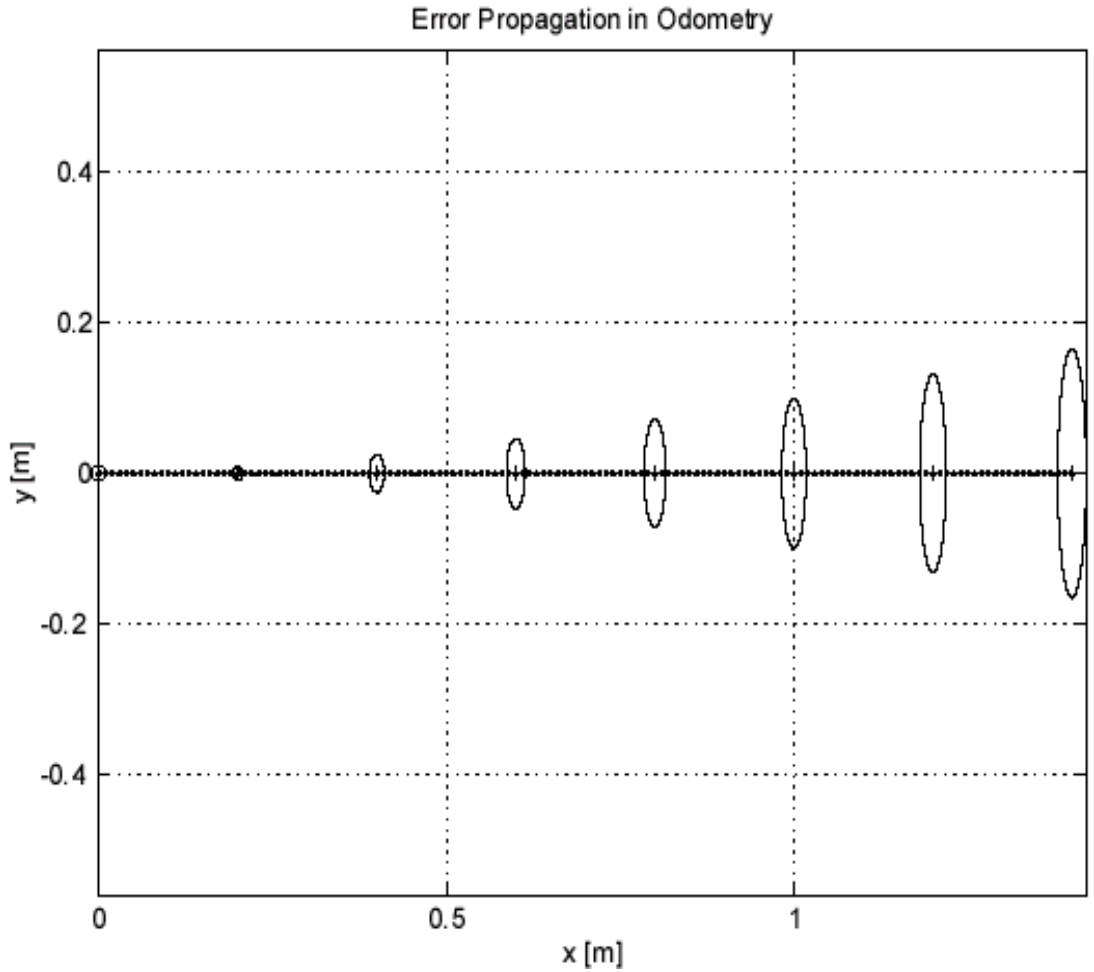
- Odometry and dead reckoning:
  Position update is based on proprioceptive sensors
  - Odometry: wheel sensors only
  - Dead reckoning: also heading sensors

- 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.

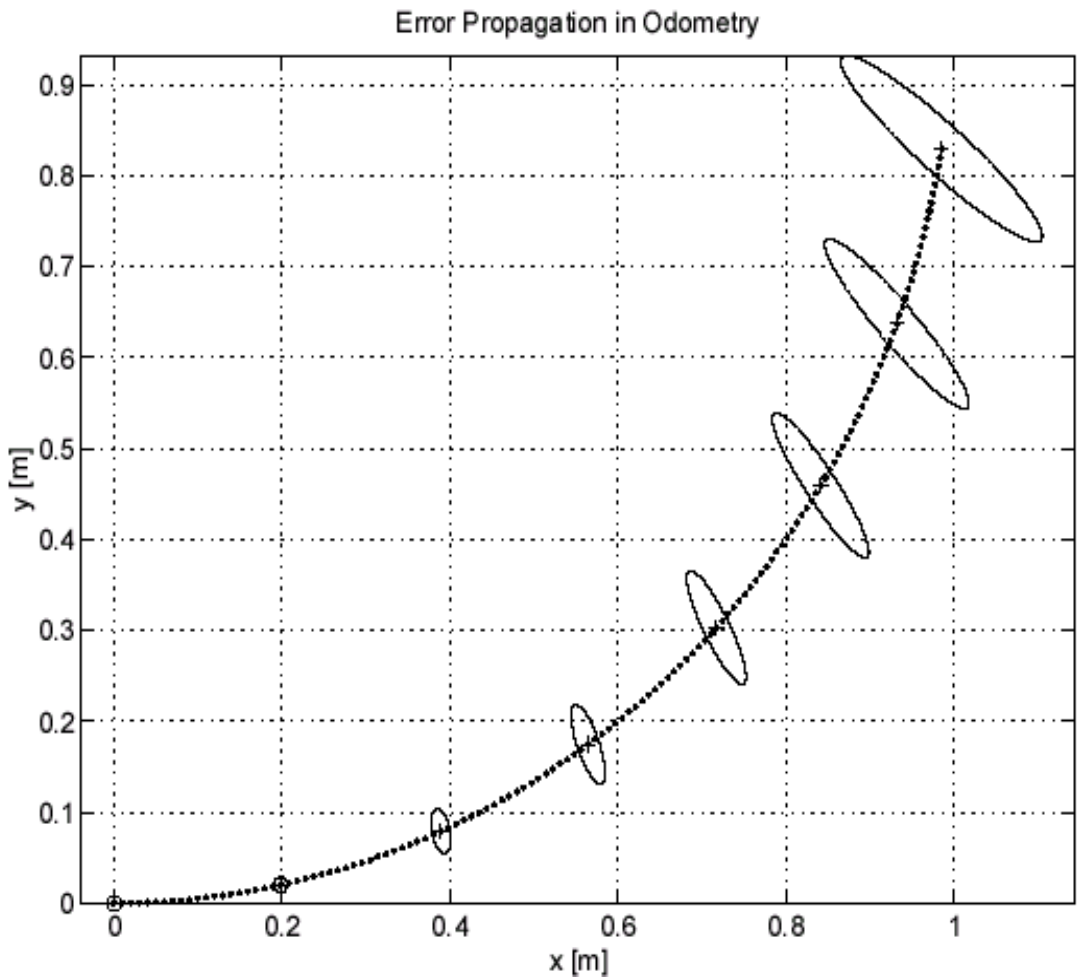
Odometry: Growth of Pose uncertainty for Straight Line Movement

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



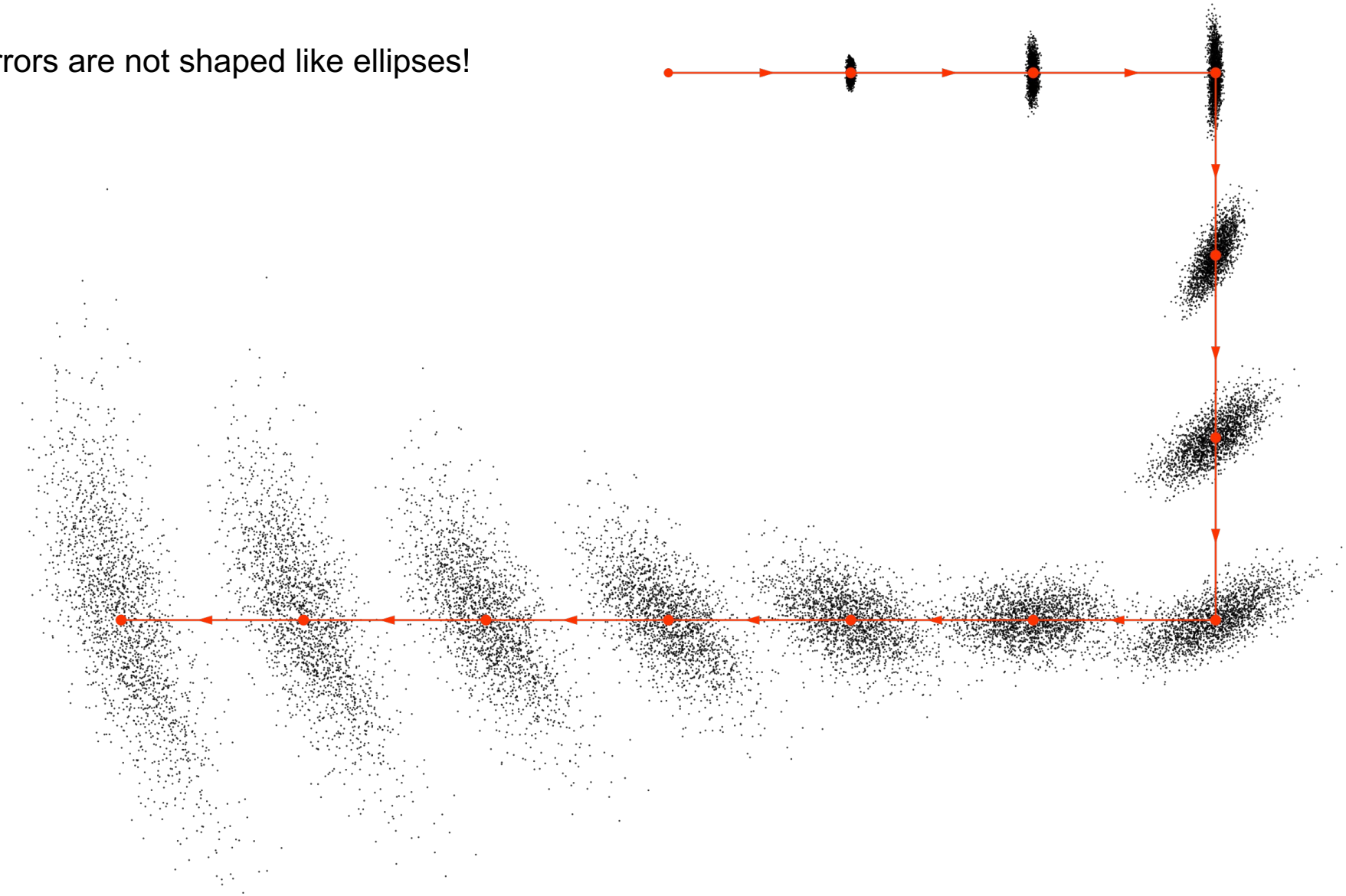
Odometry: Growth of Pose uncertainty for Movement on a Circle

- 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