



上海科技大学
ShanghaiTech University

CS283: Robotics Spring 2023: Localization

Sören Schwertfeger / 师泽仁

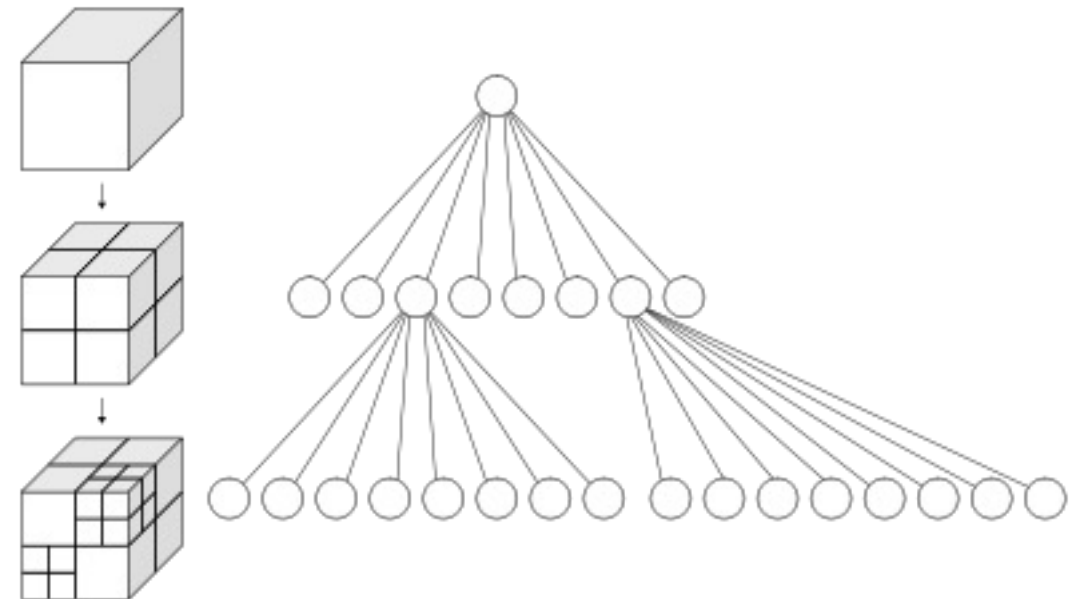
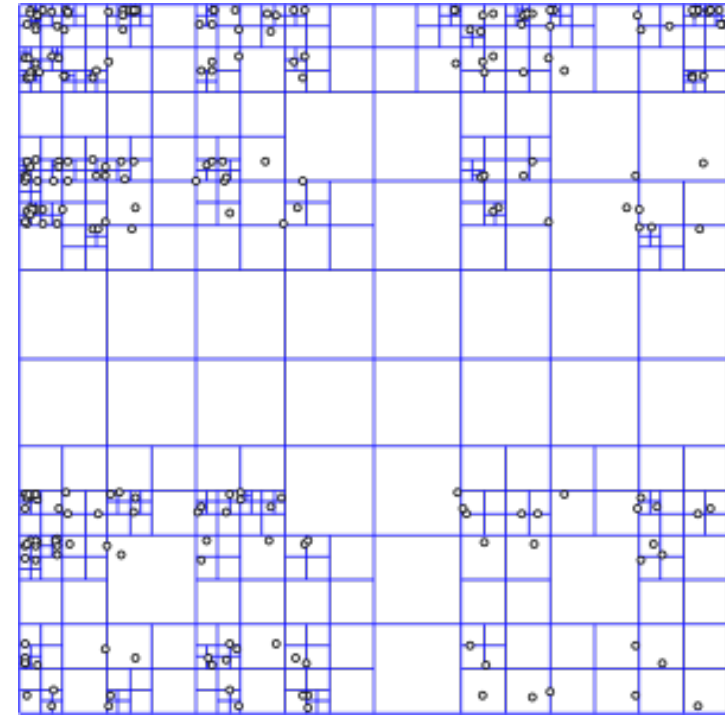
ShanghaiTech University

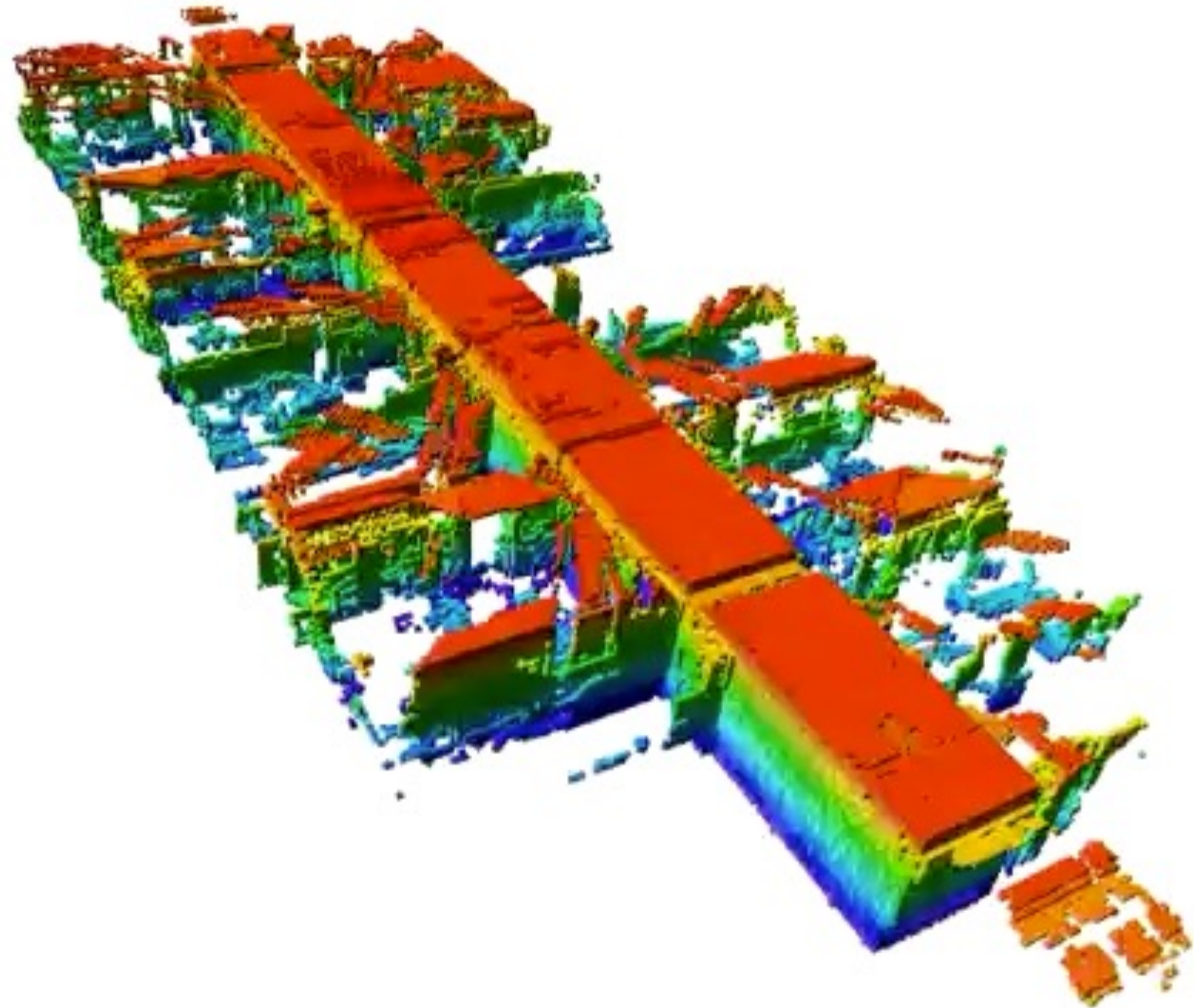
Map Representation: what is “saved” in the map

- 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

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

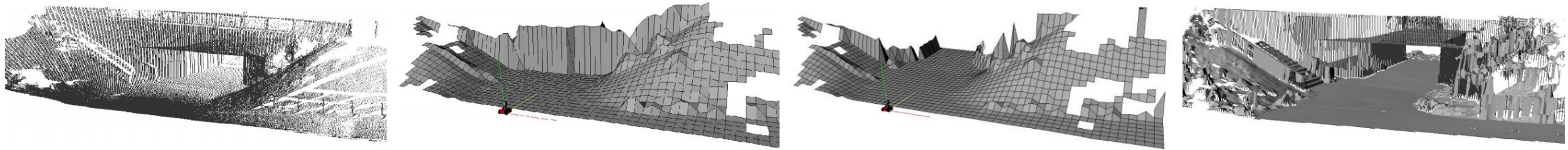


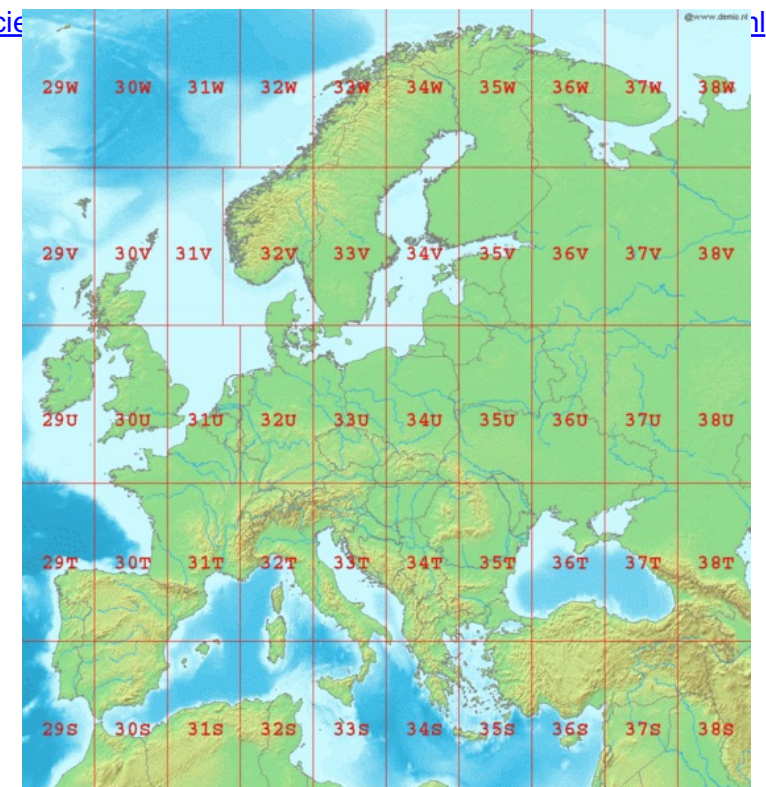
Fig. 1. Scan (point set) of a bridge (a), standard elevation map computed from this data set (b), extended elevation map which correctly represents the underpass under the bridge (c), and multi level surface map that correctly represents the height of the vertical objects (d)

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)

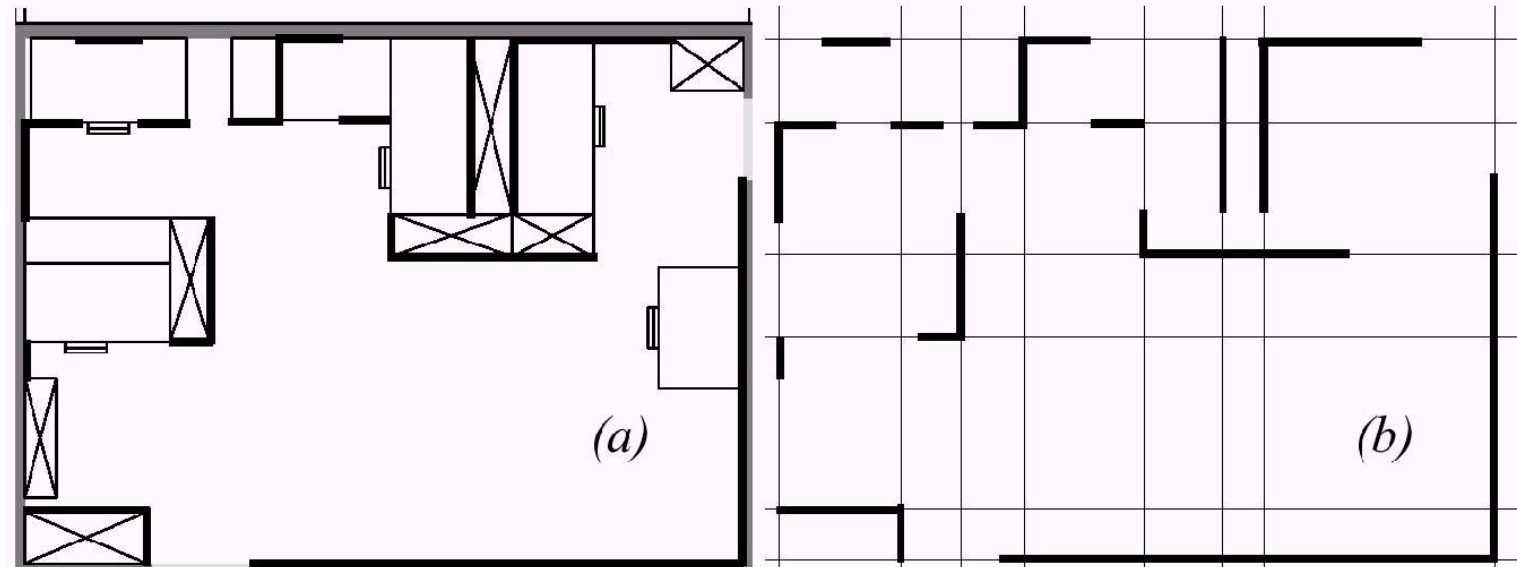
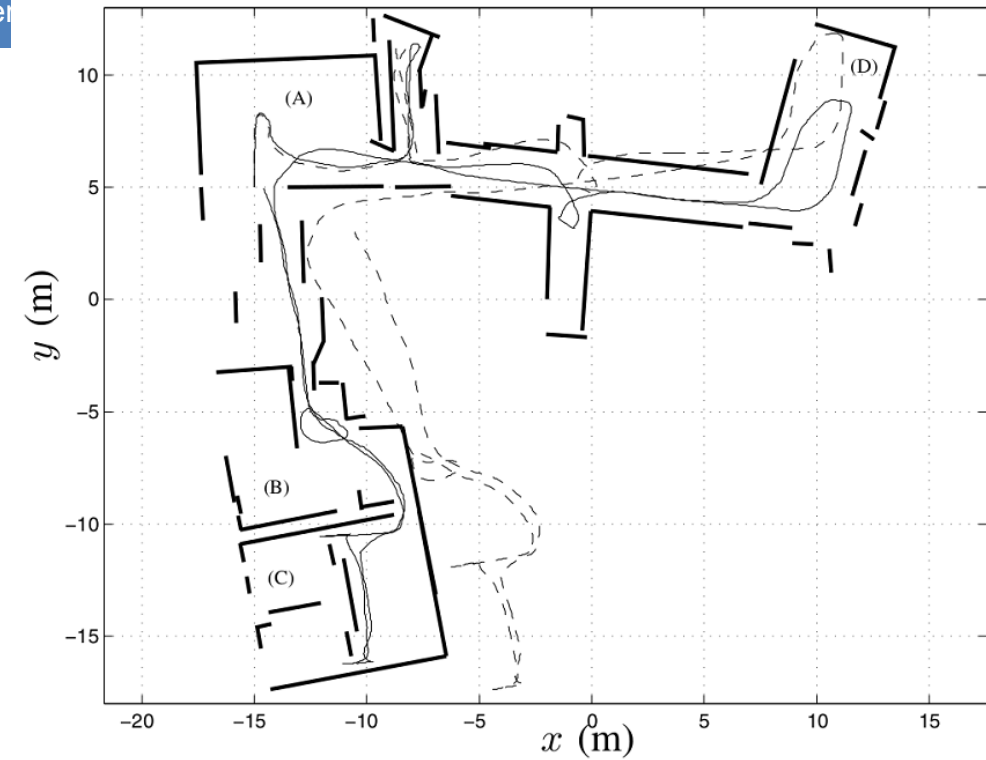


<https://www.livesci>

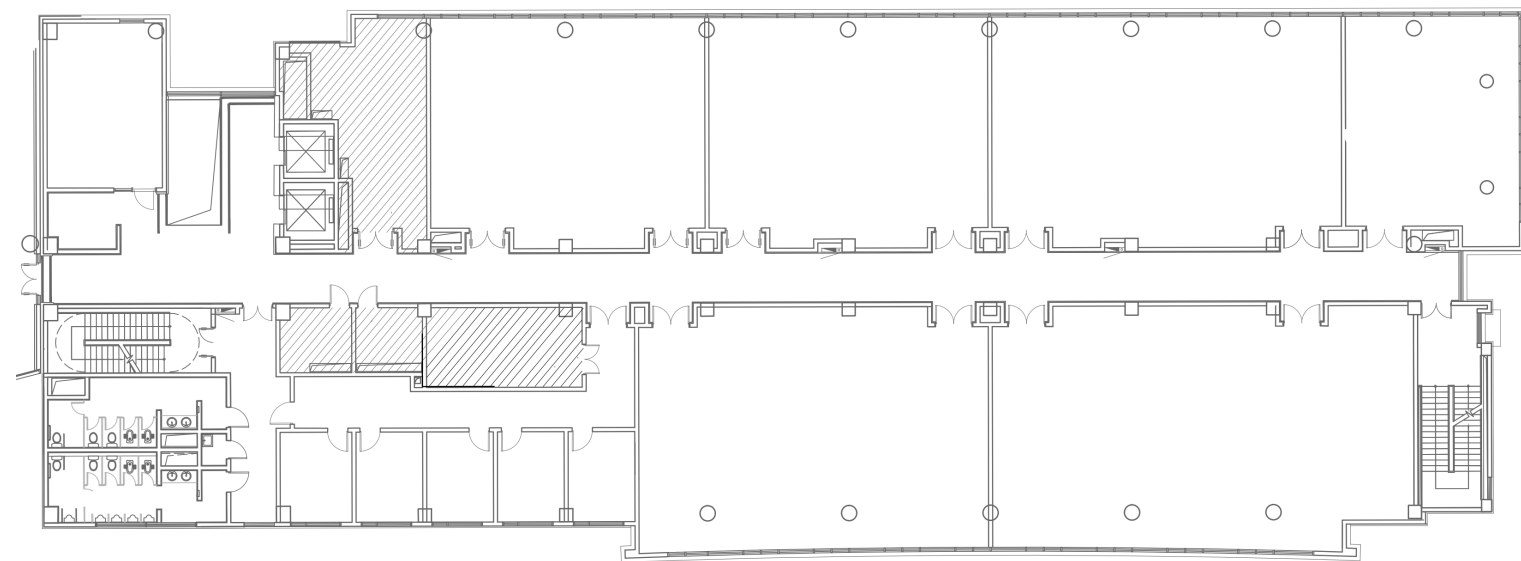


Line Map

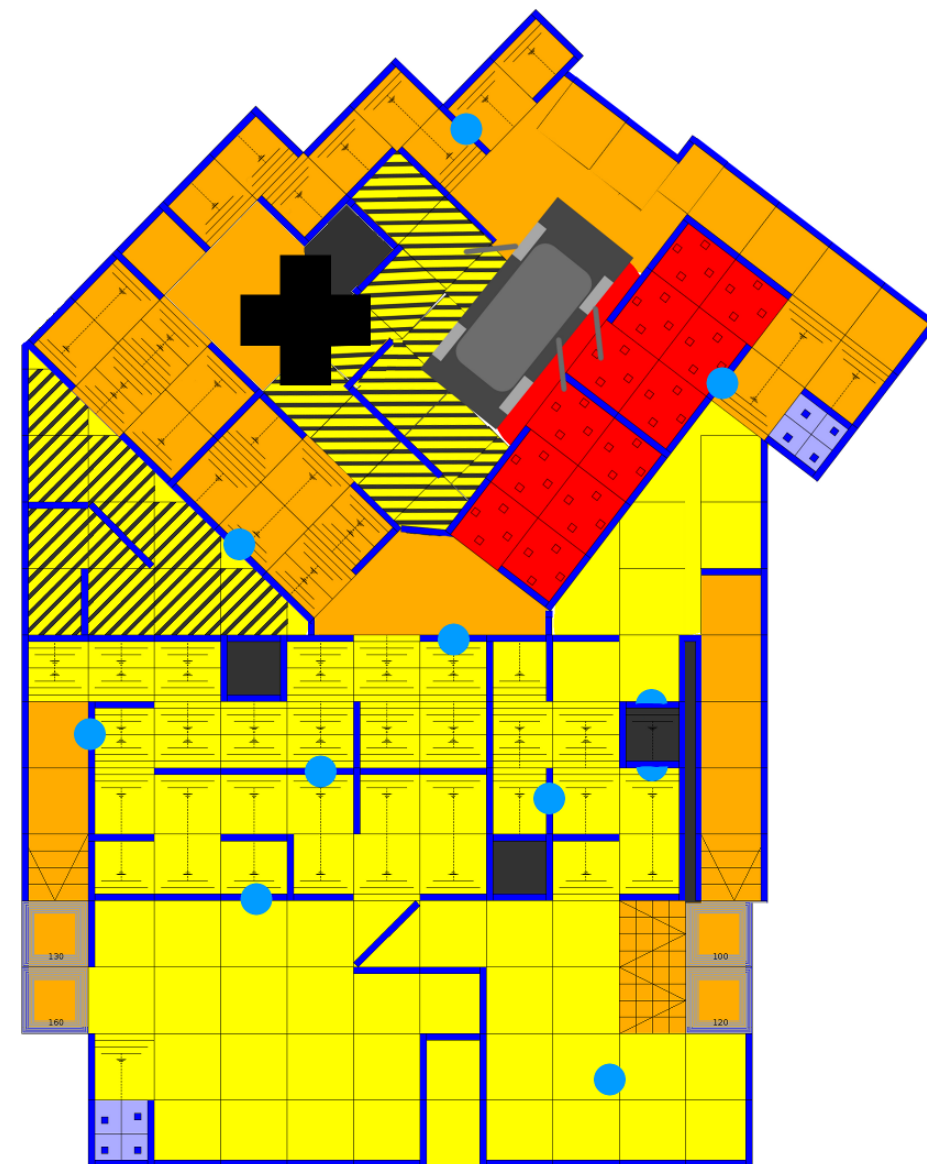
- Abstract from points =>
 - Extract features (e.g. lines, planes)
 - E.g. using RANSAC, Hough Transform
 - E.g. region growing, e.g. via normals
-
- Finite lines (a)
 - Infinite lines (b)
-
- Very compact
 - Can do scan-matching (e.g. against laser scan)



Ground Truth Maps

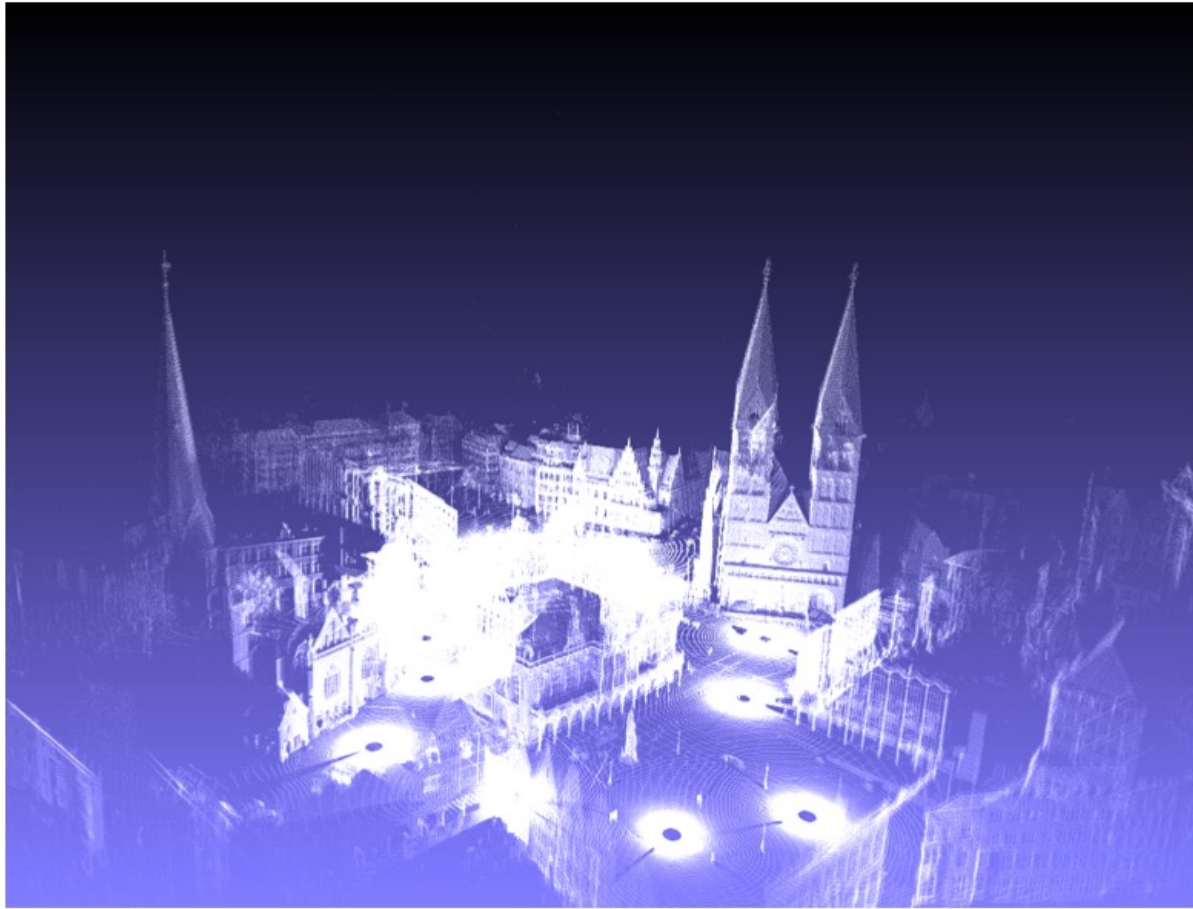


CAD drawing (STAR Center)
Vector format (lines, circles, ...)

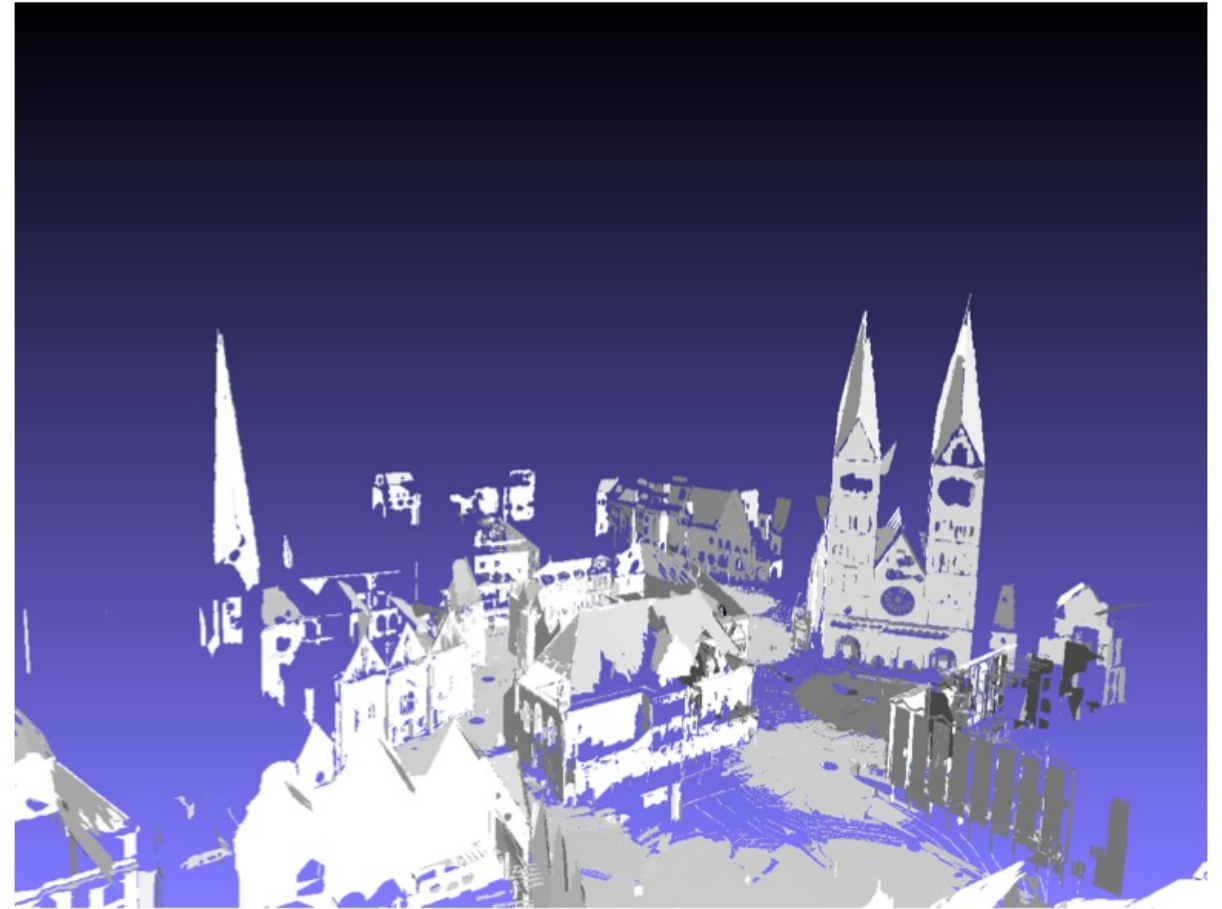


Grid Map (from vector map)

Bremen: 3D Point Cloud & 3D Plane Map

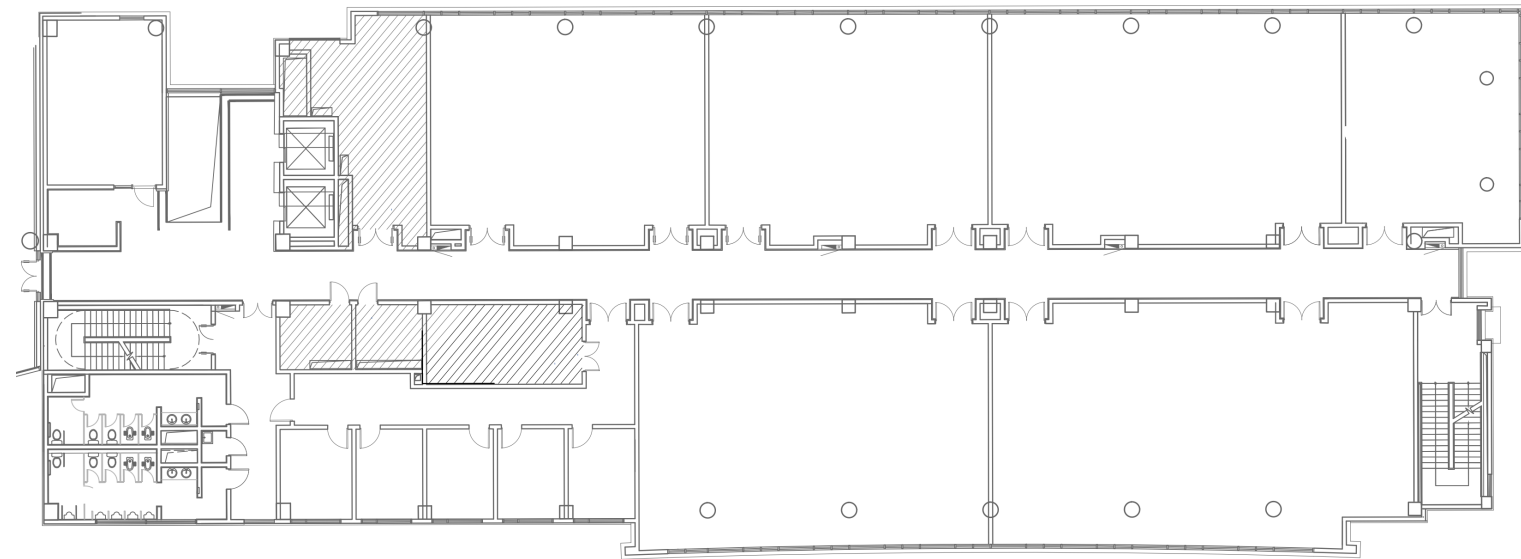


(a) 3D Point Cloud

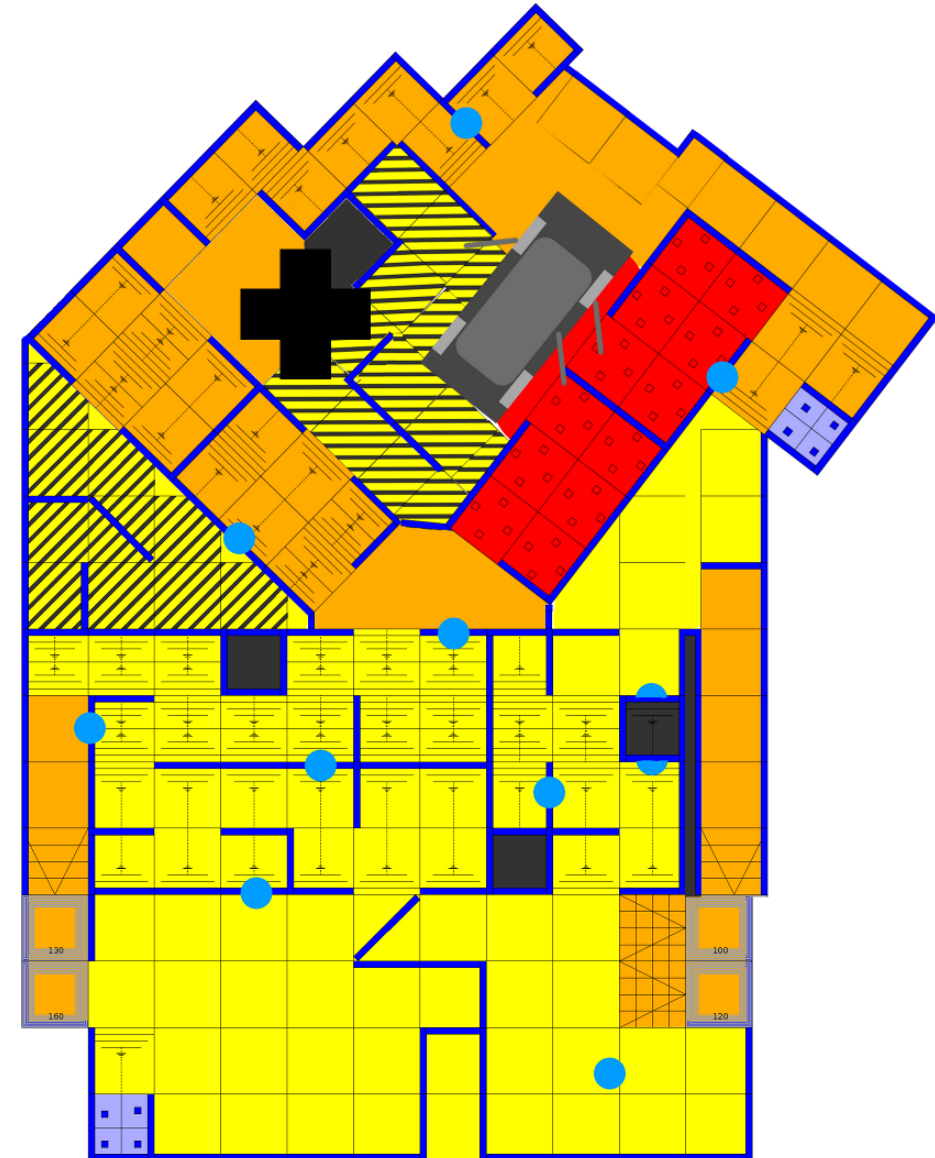


(b) 3D Plane Map

Ground Truth Maps

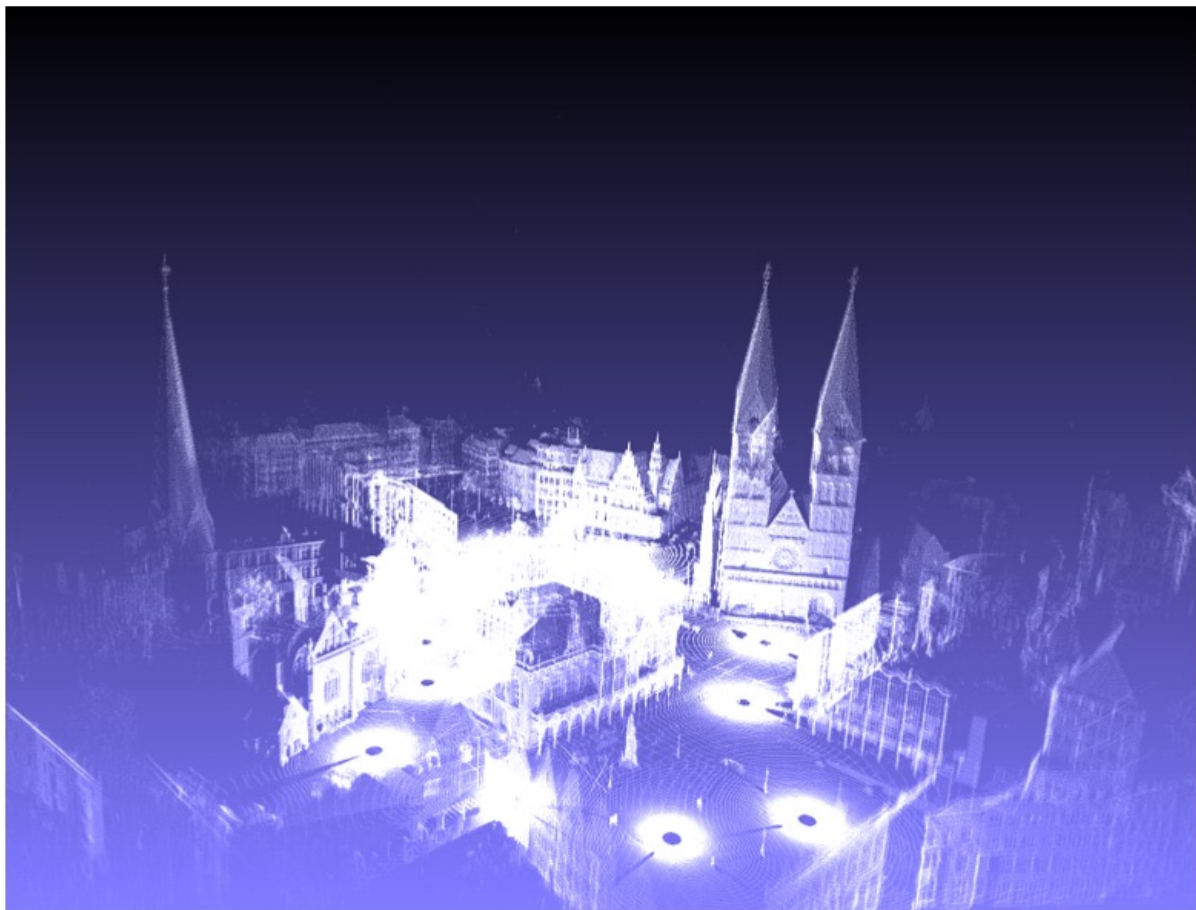


CAD drawing (STAR Center)
Vector format (lines, circles, ...)

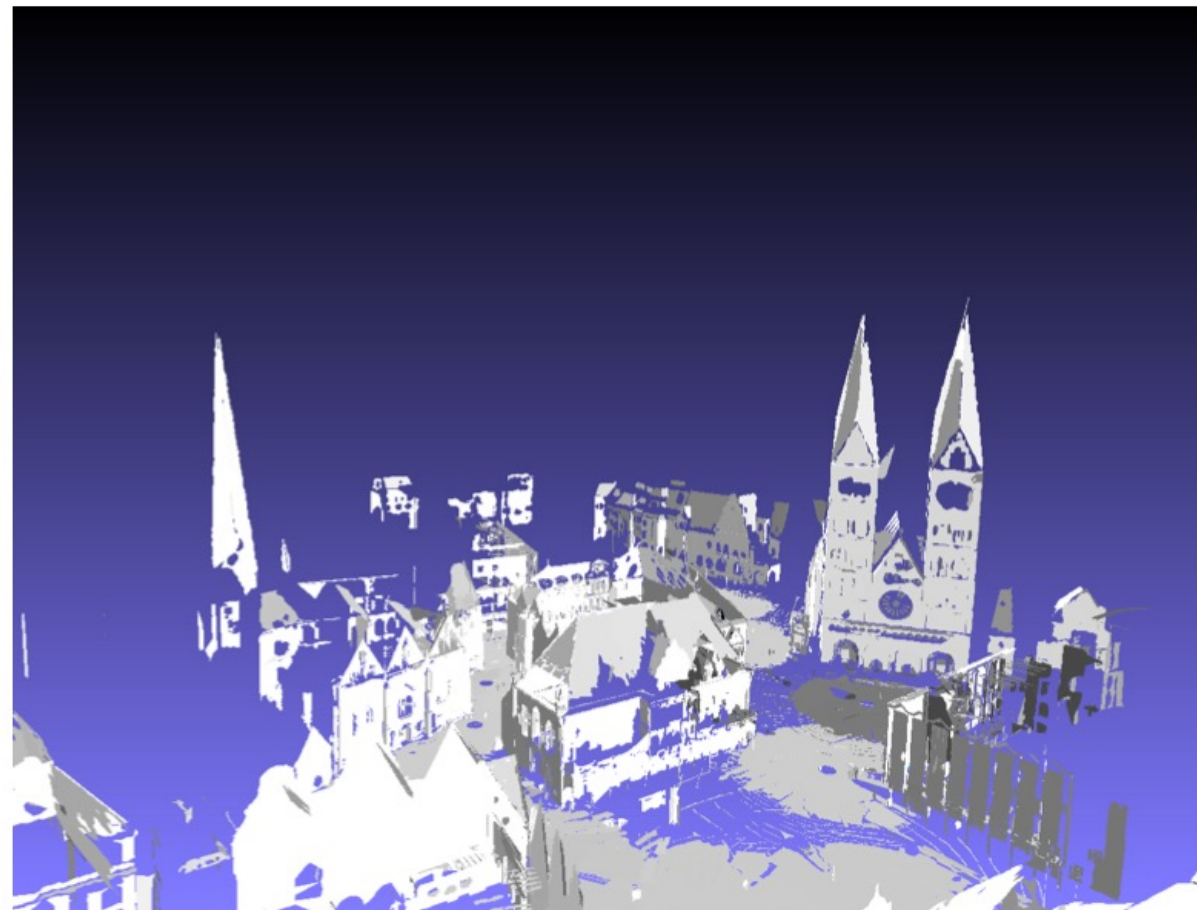


Grid Map (from vector map)

Bremen: 3D Point Cloud & 3D Plane Map

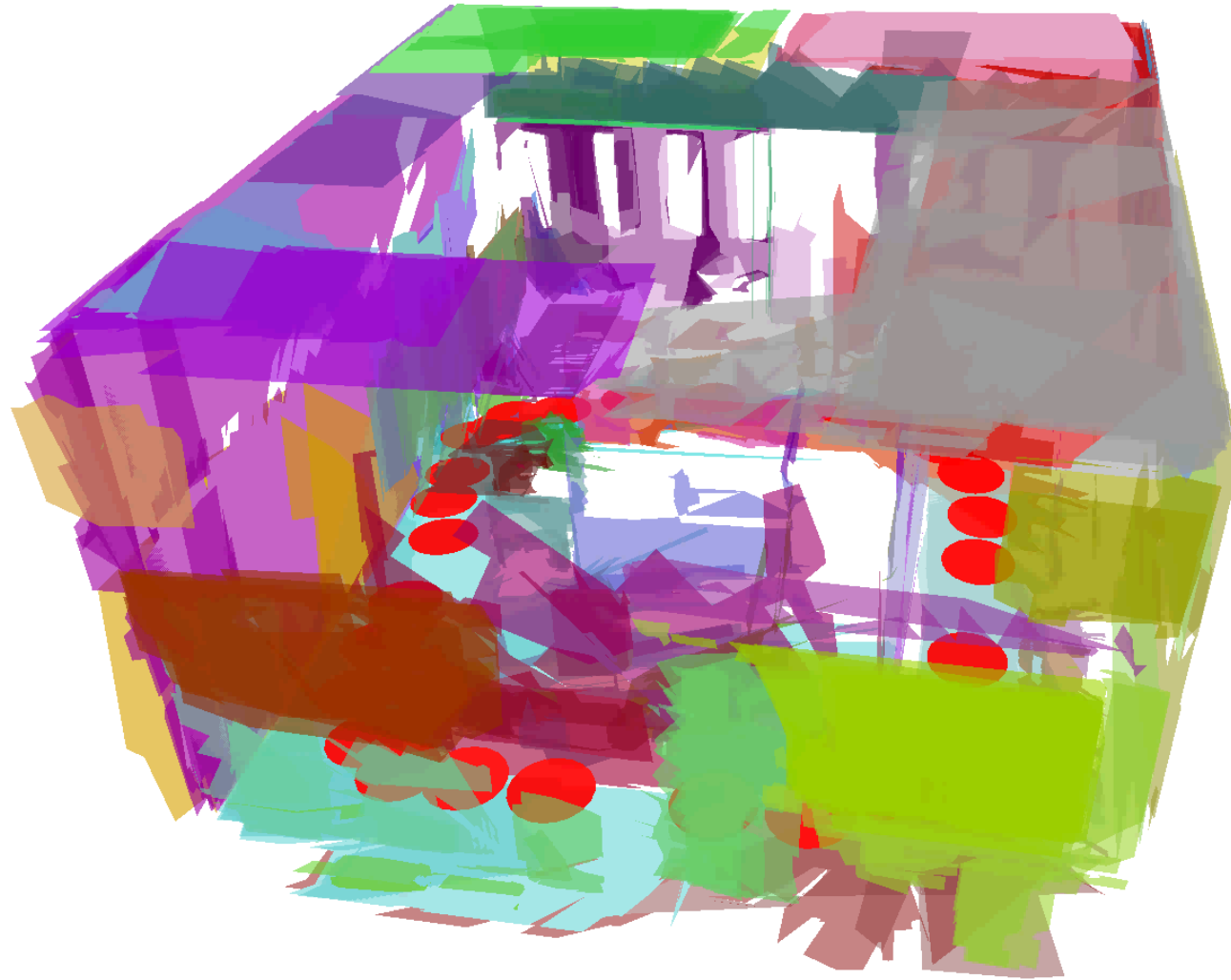


(a) 3D Point Cloud



(b) 3D Plane Map

Plane map: 29 poses; each with several planes



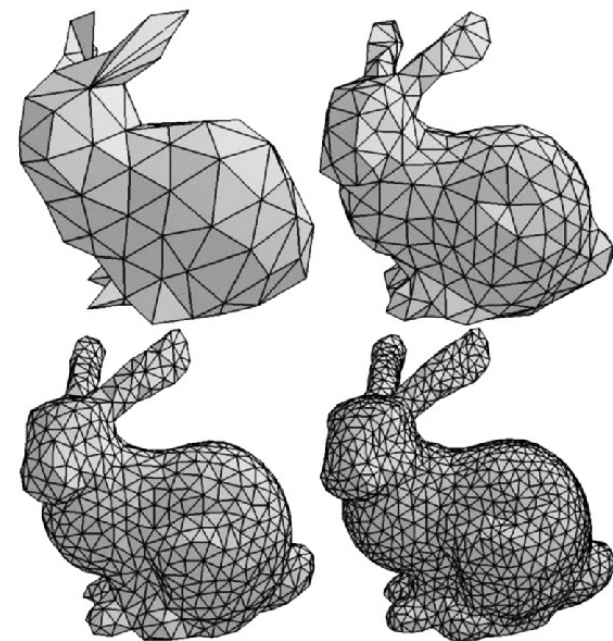
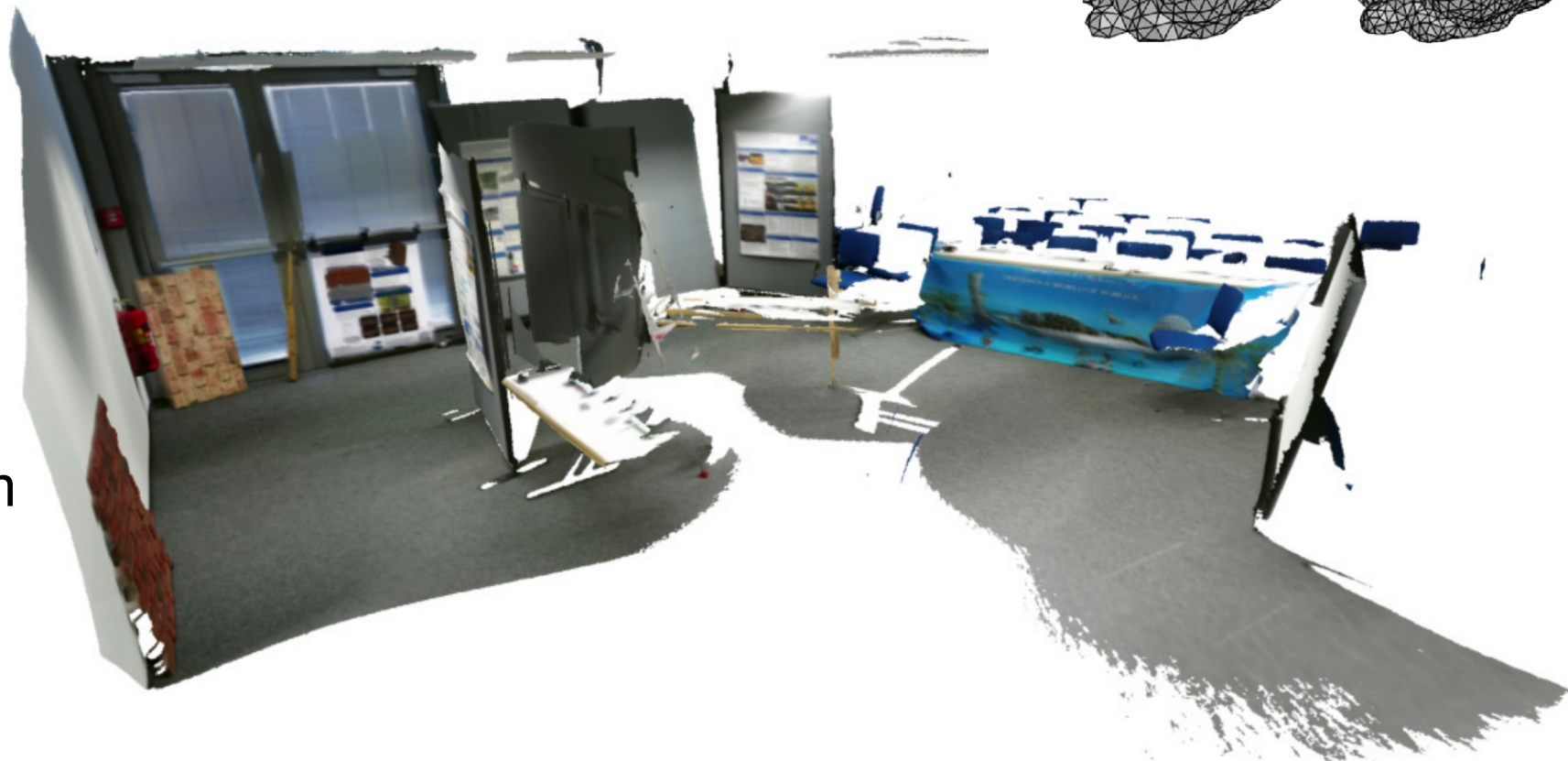
Vector Maps

- 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)



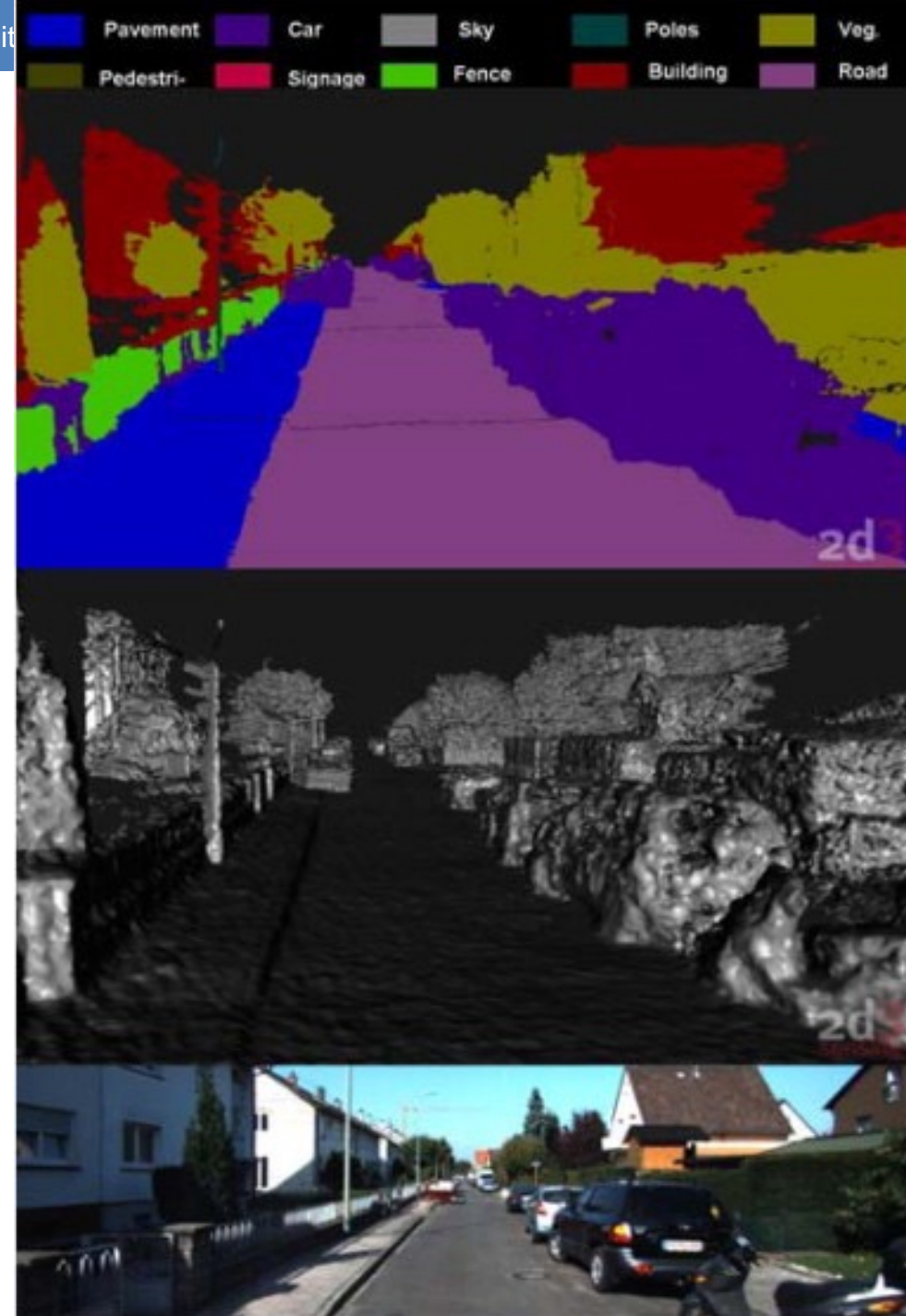
StereEye: Mesh Simplification

- 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

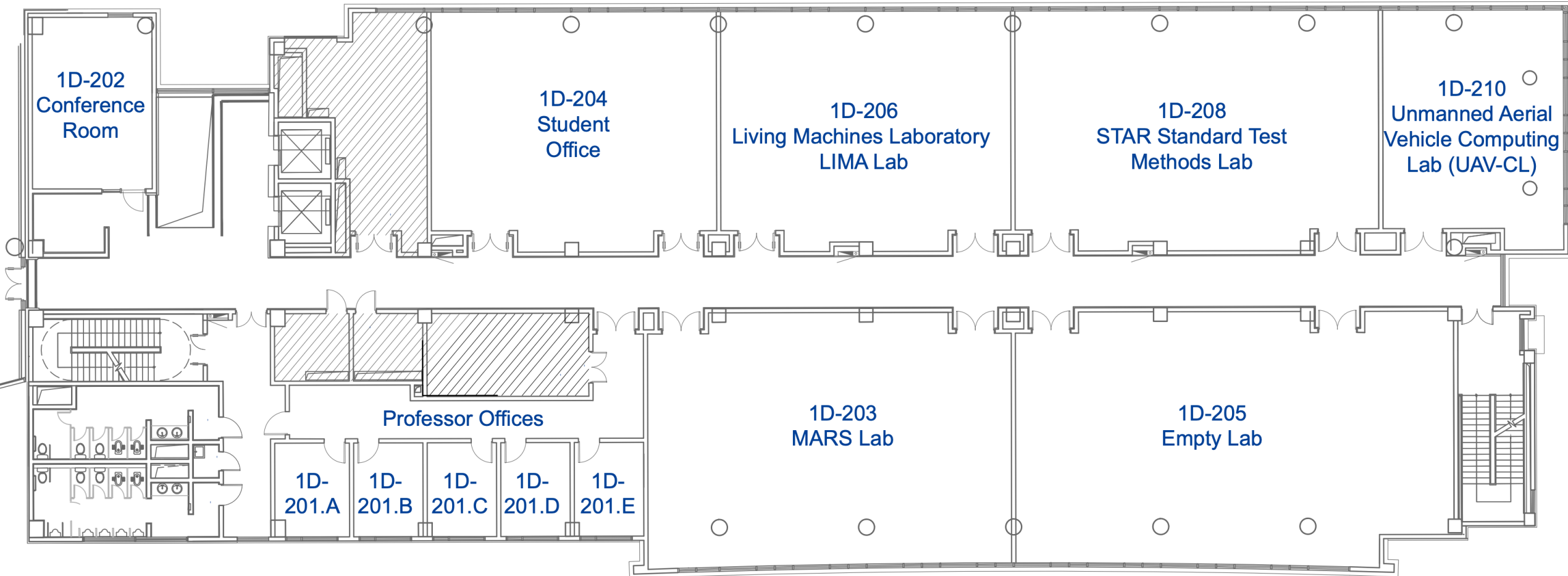


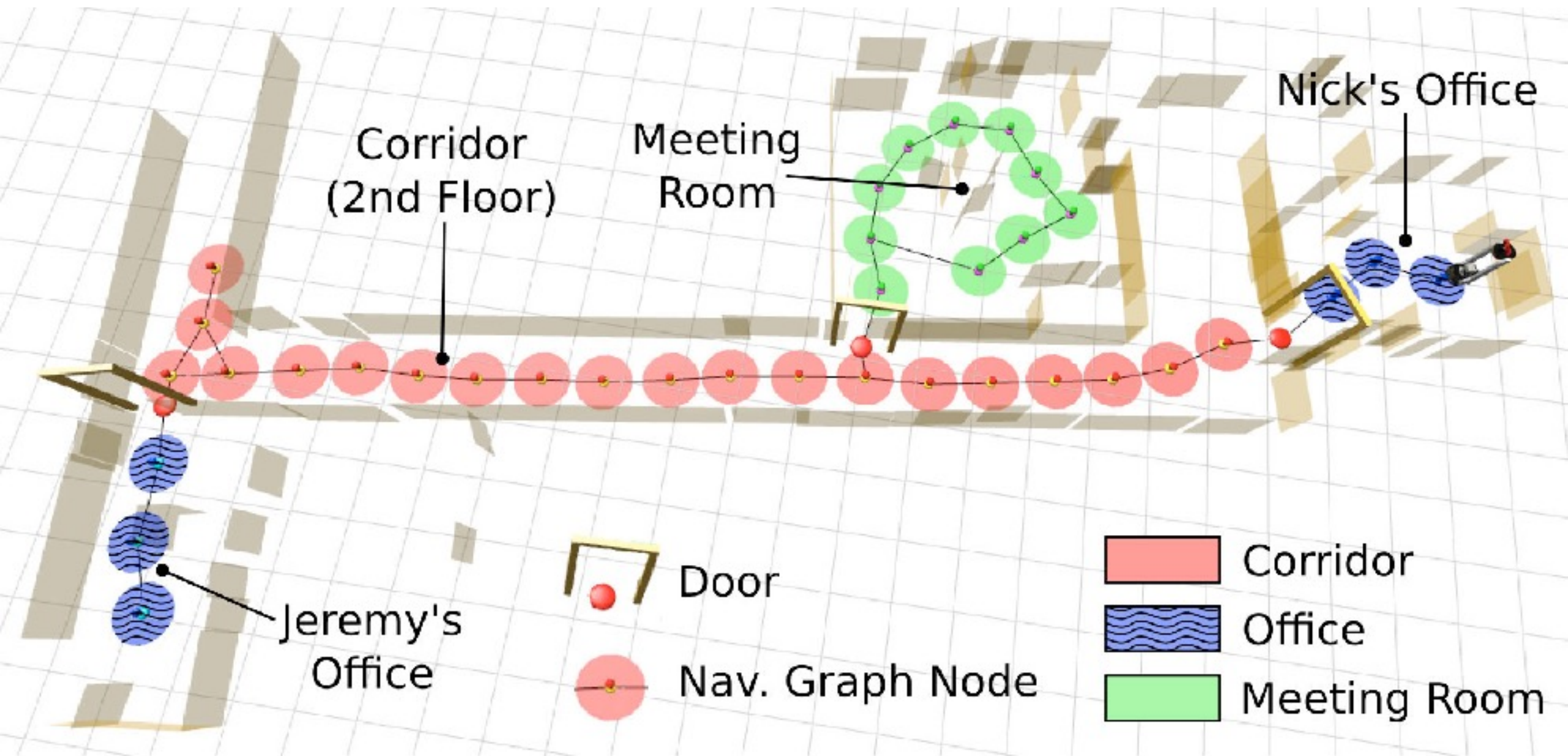
Semantic Map

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



Semantic Annotation: Room Names



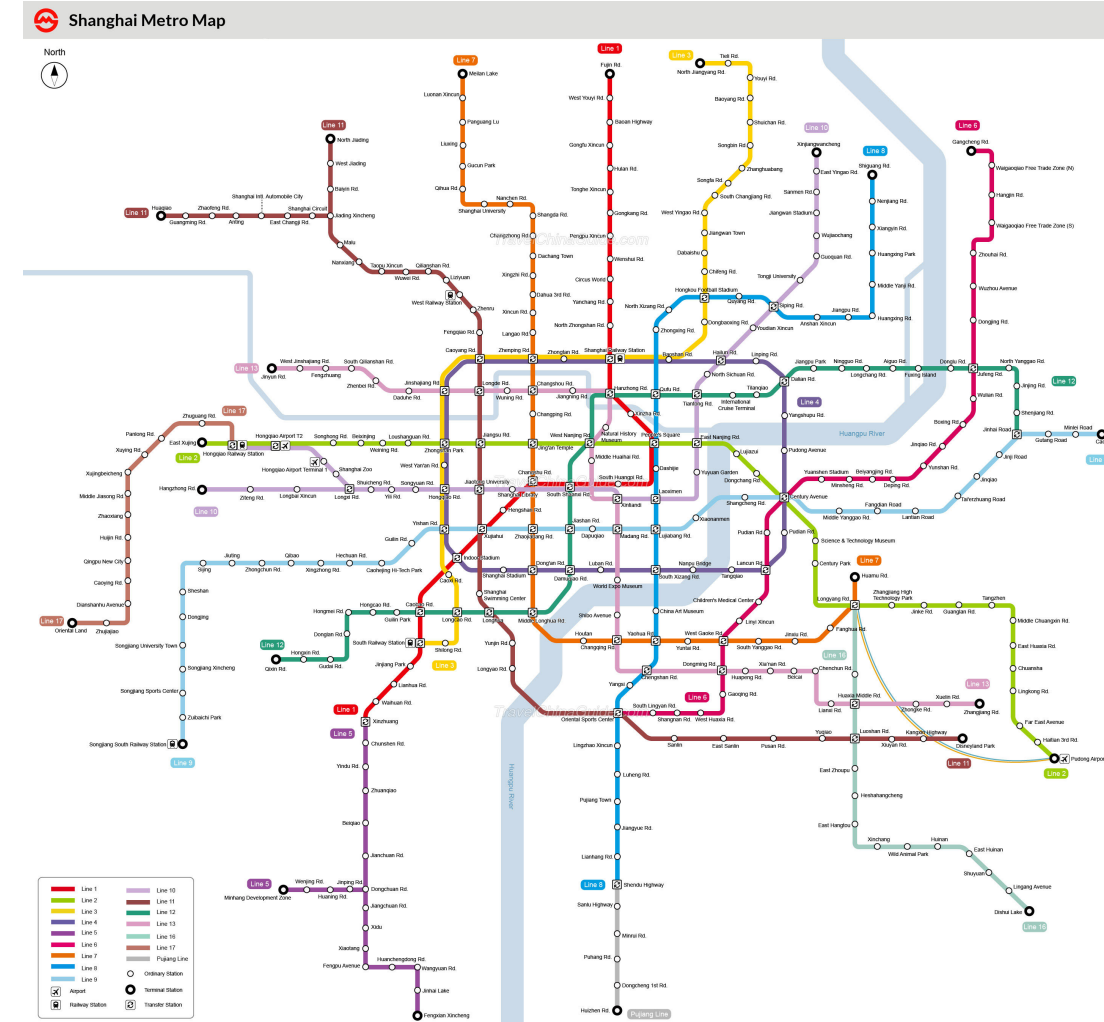


Semantic Information

- 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
 - ...

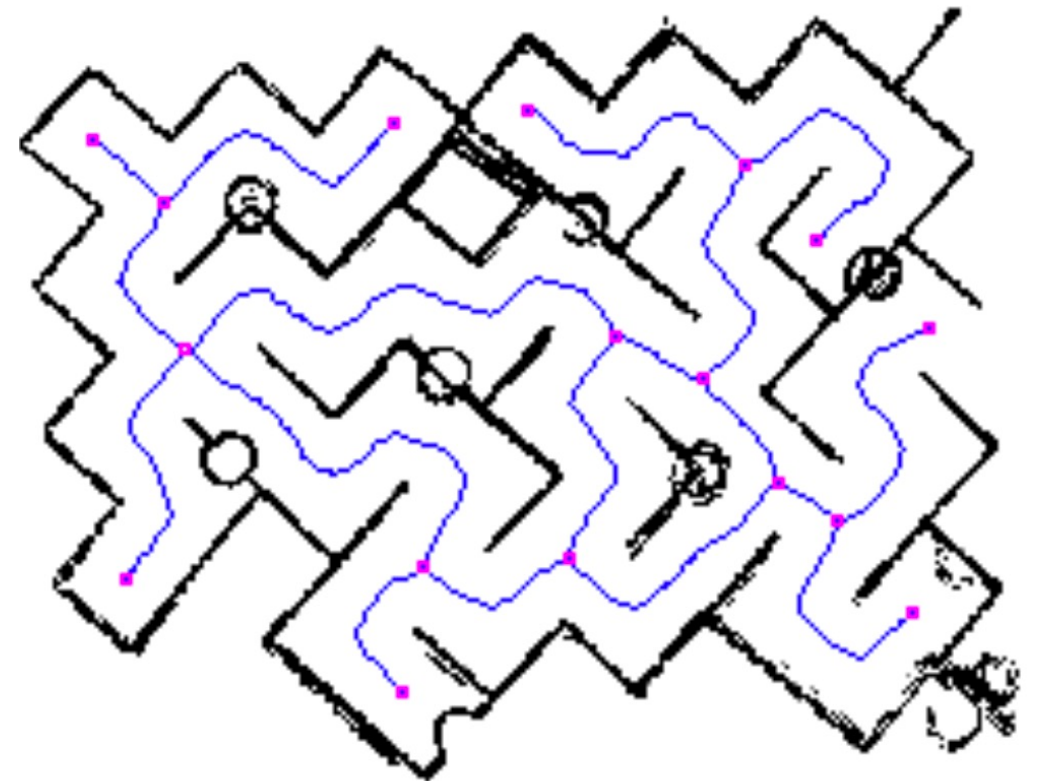
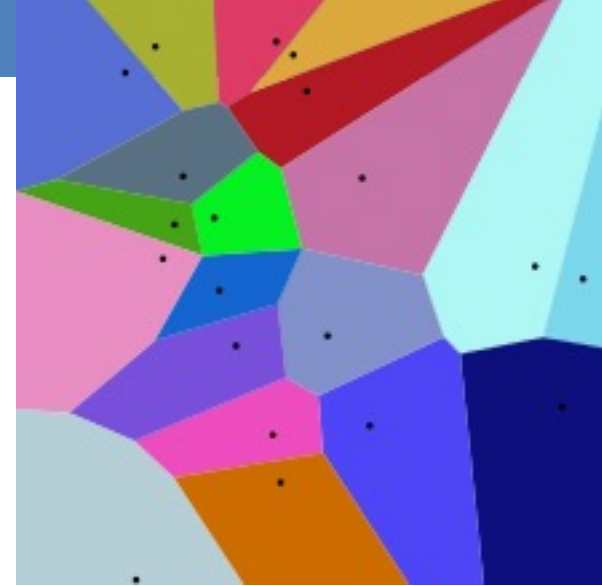
Topologic Map

- 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



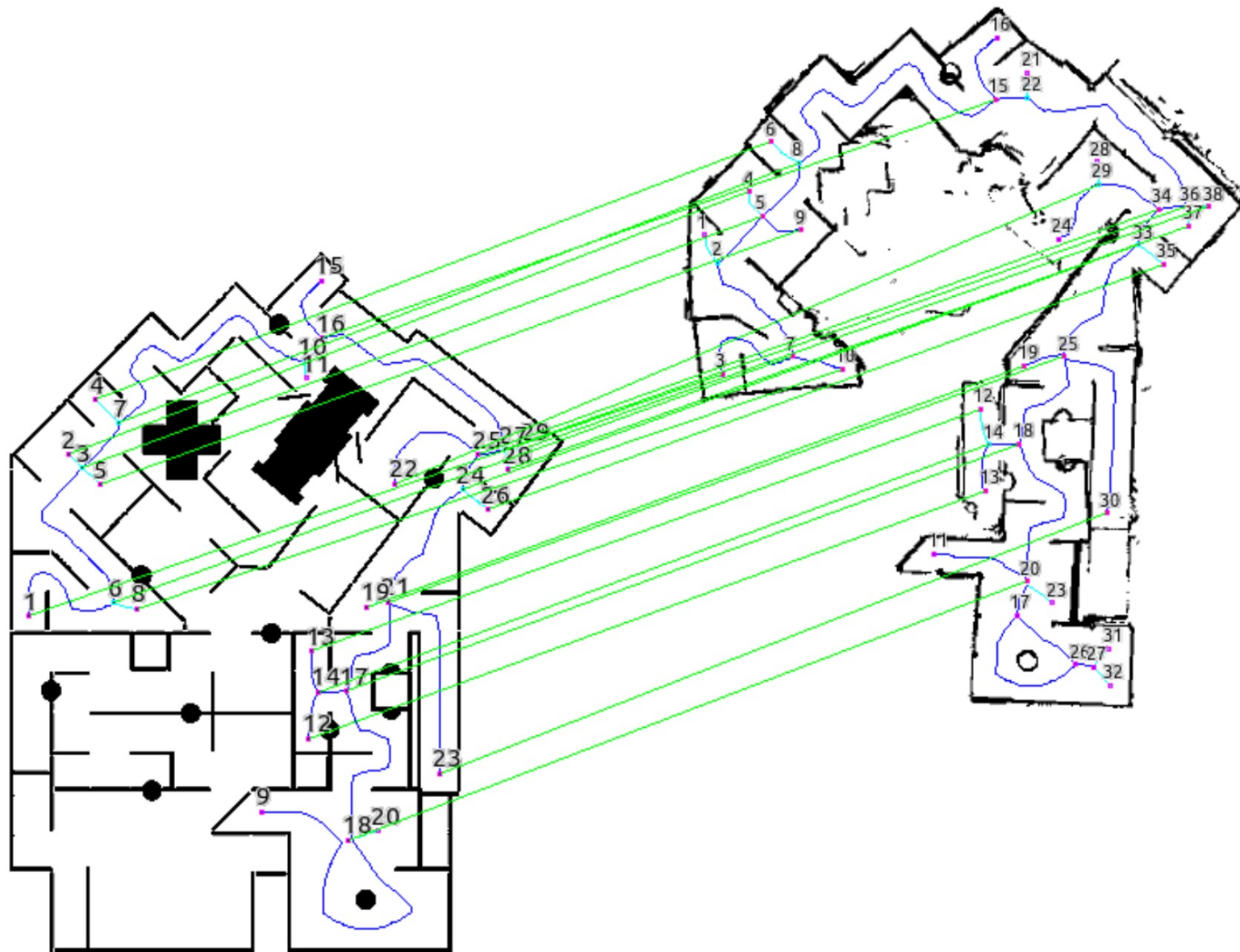
Voronoi Diagram (-> Topology Graph)

- 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

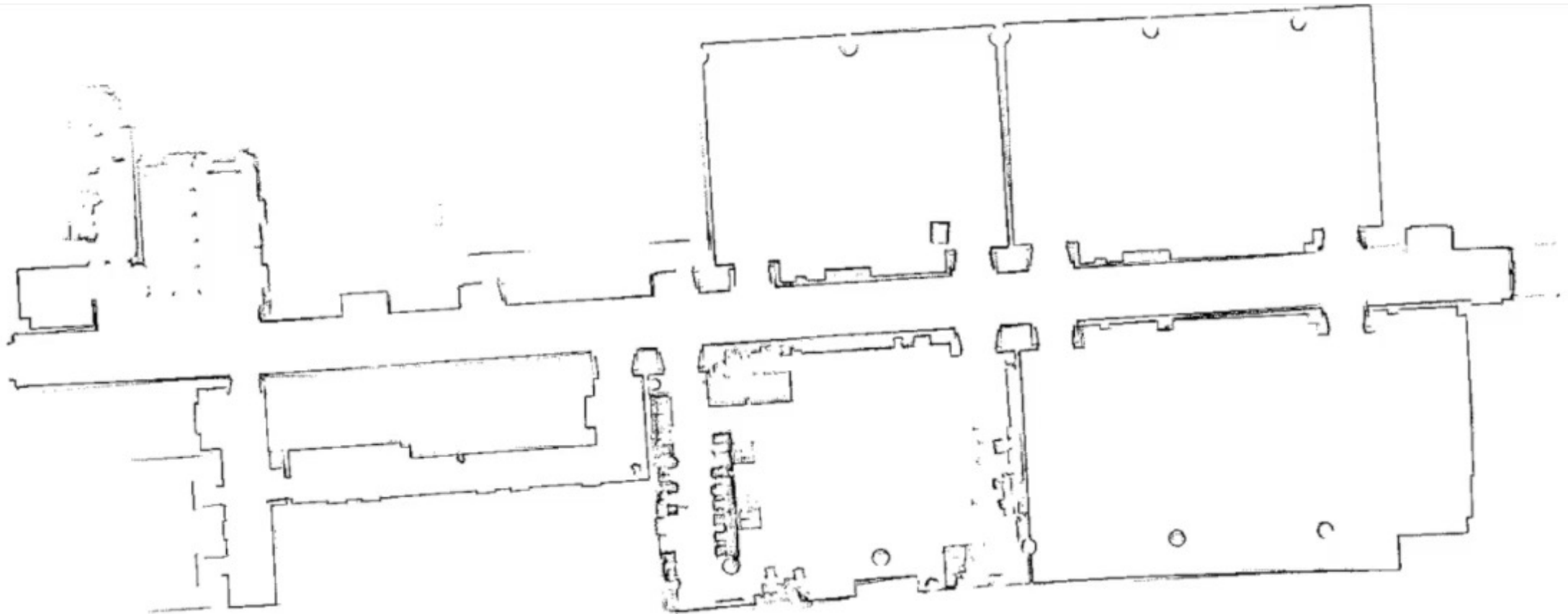


Map Matching

- 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

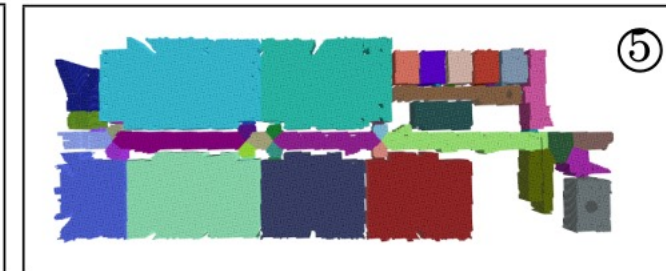
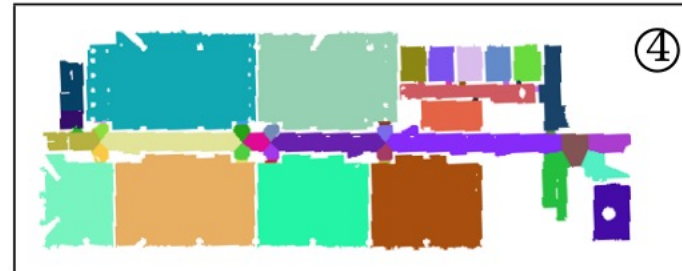
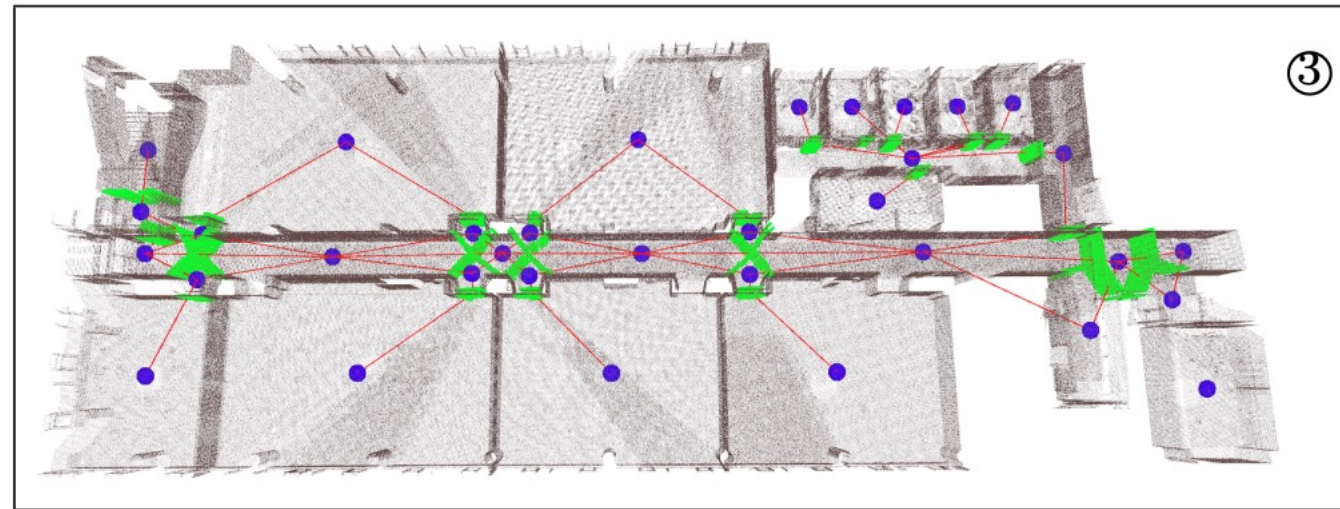
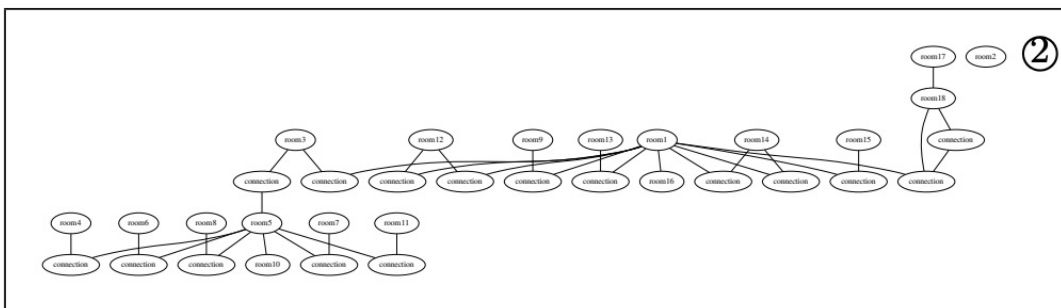
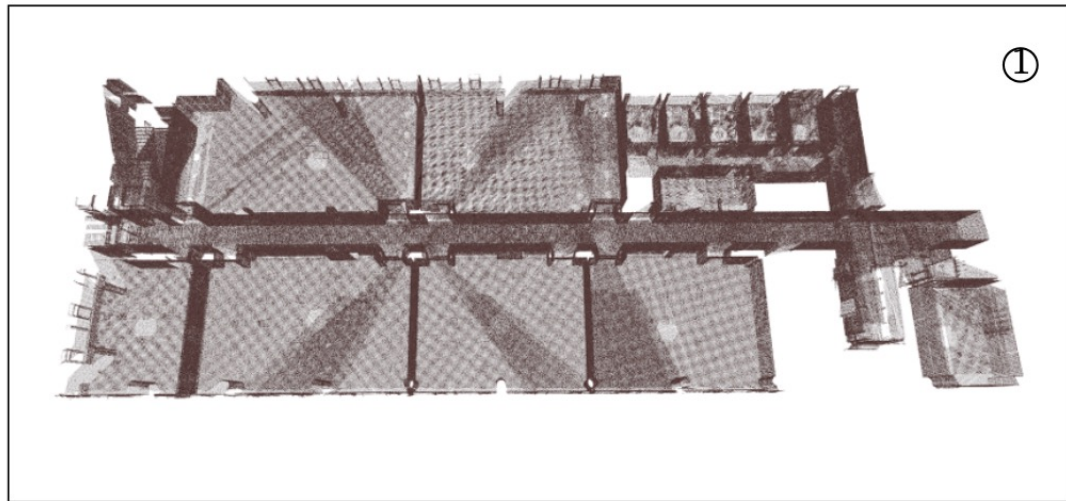
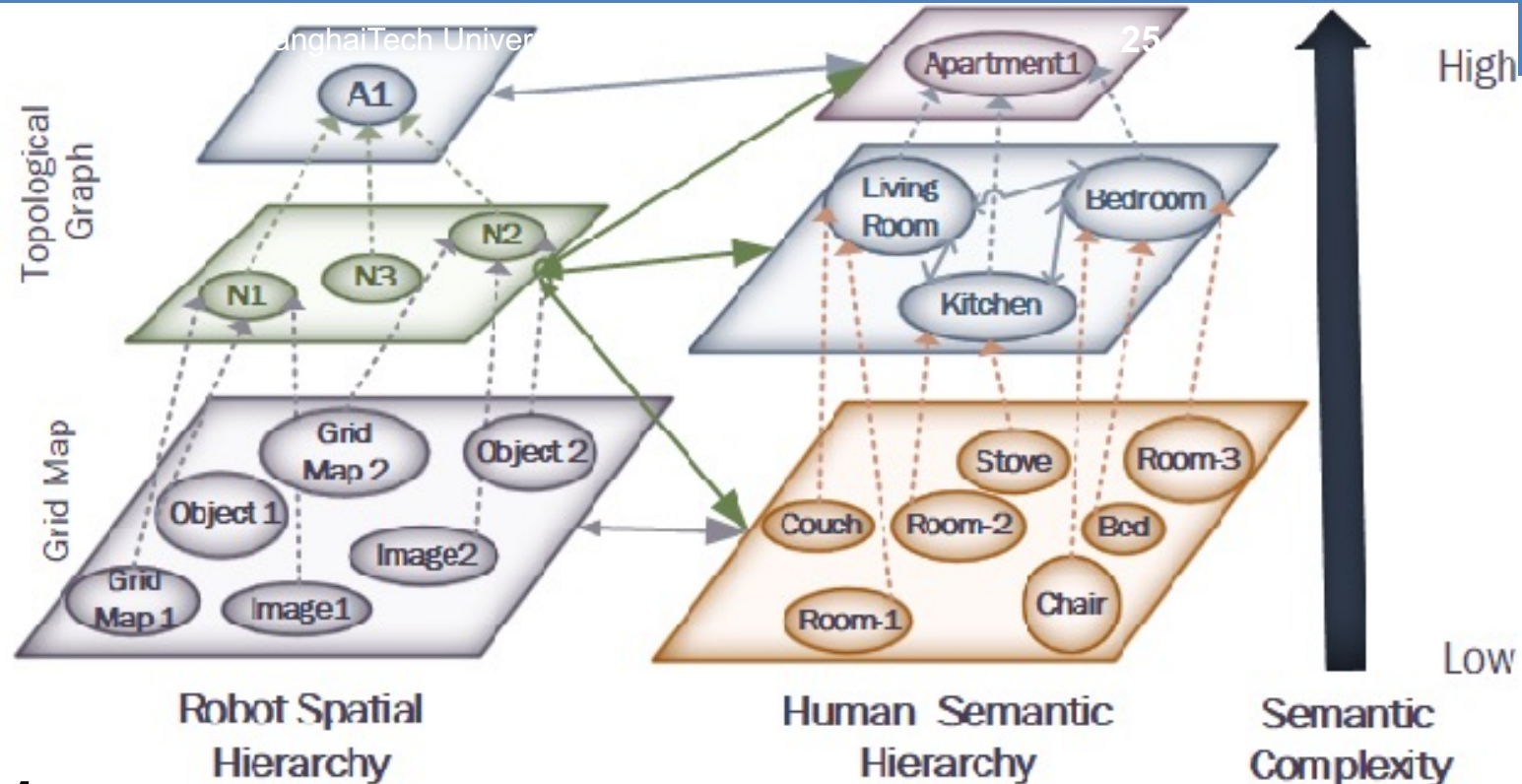
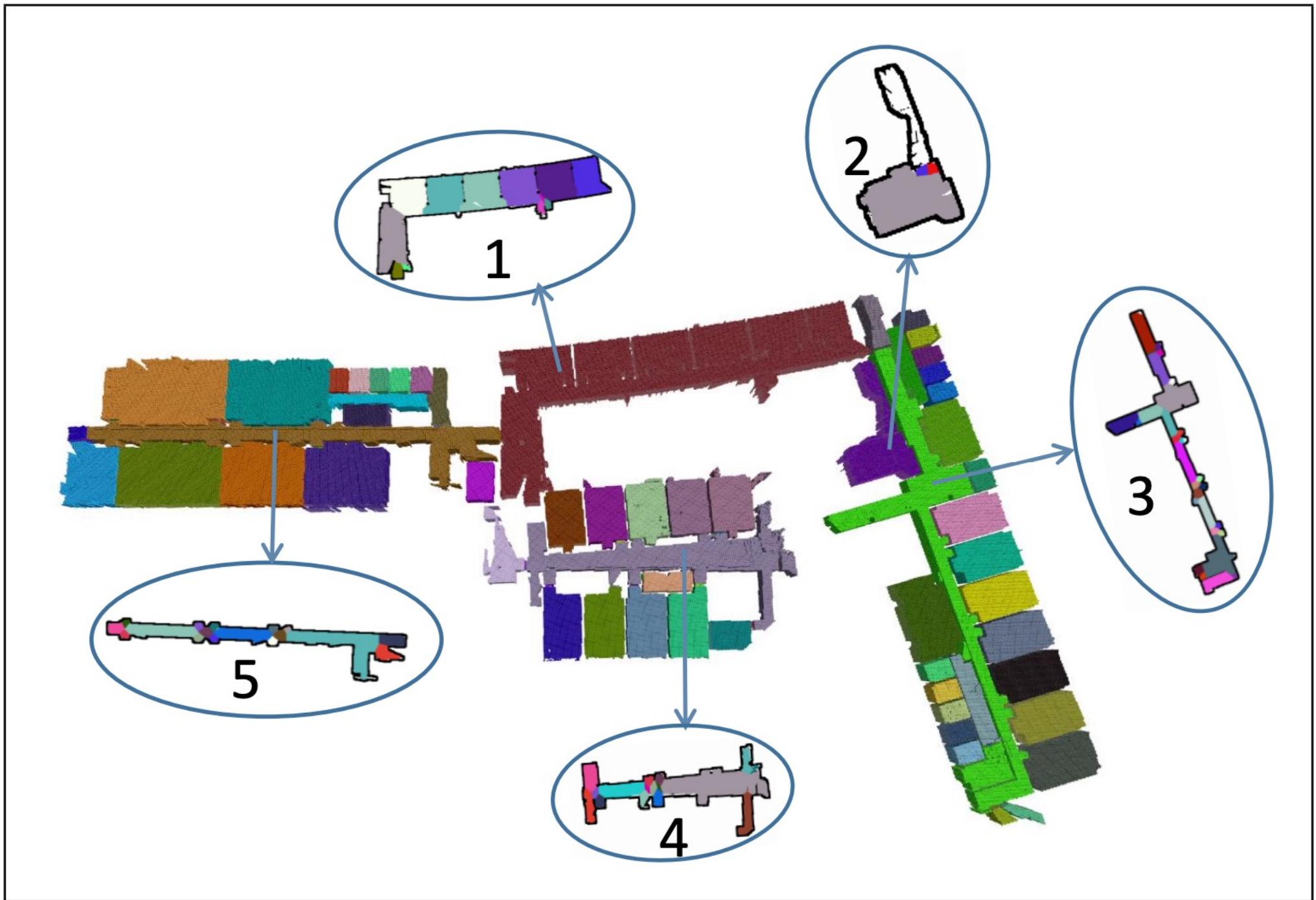


Fig. 1: Input 3D point cloud of 4 floors of a building. Below: results of our algorithm in different dimensions. From top to bottom, left to right are: 0D, 1D, 2D, 3D

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)





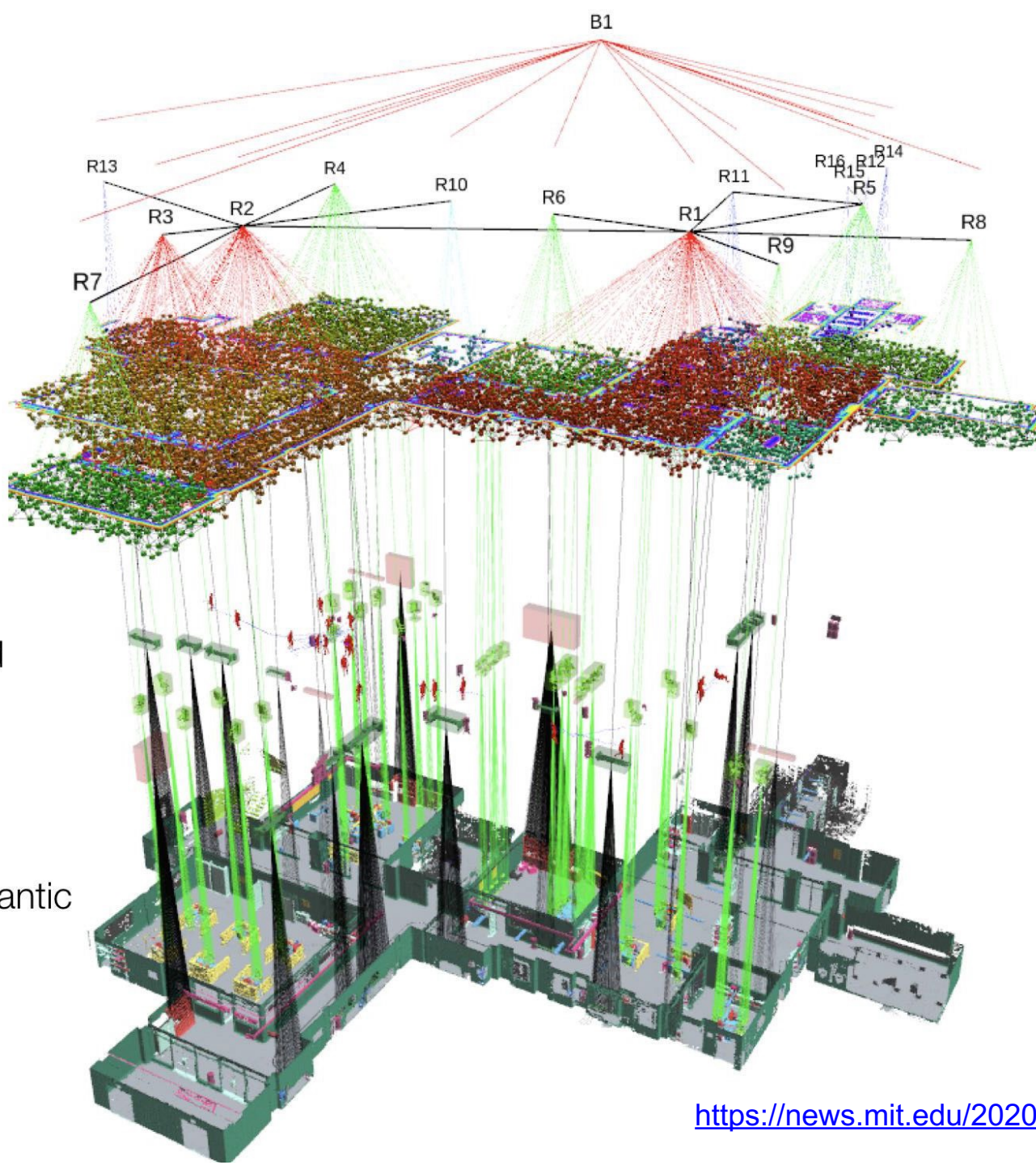
Layer 5:
Buildings

Layer 4:
Rooms

Layer 3:
Places and
Structures

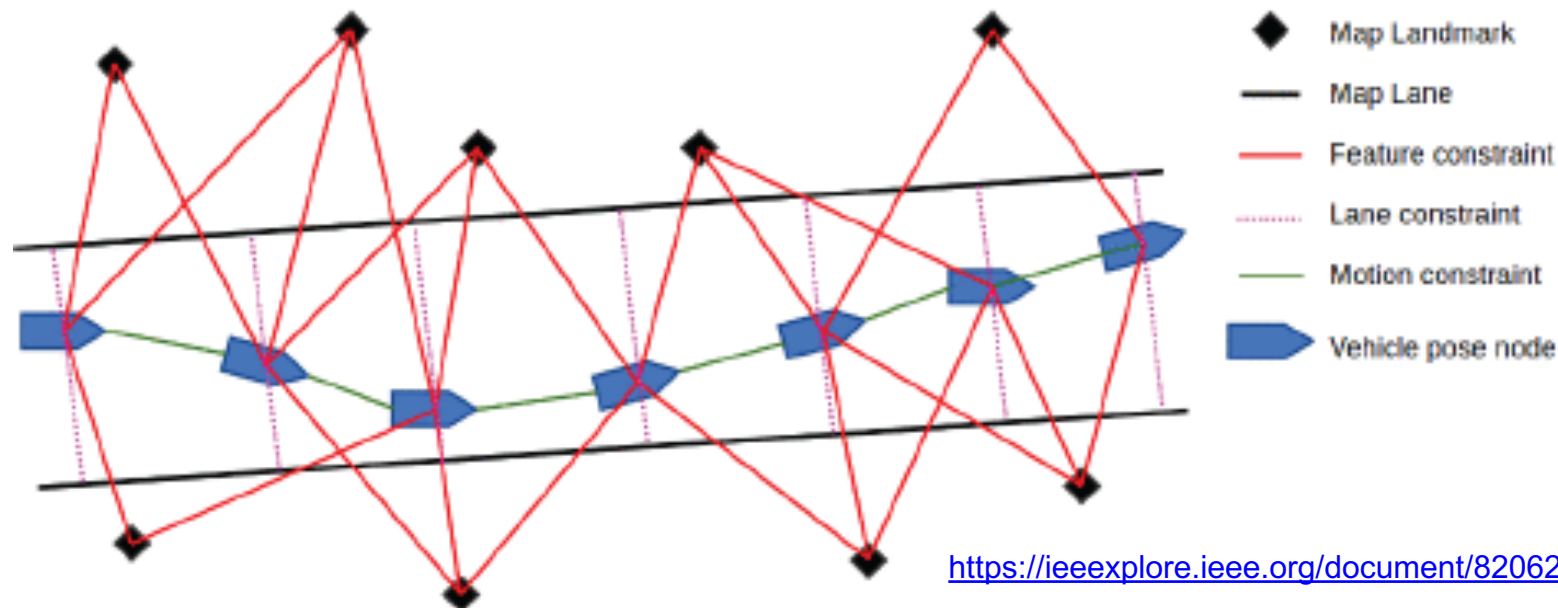
Layer 2:
Objects and
Agents

Layer 1:
Metric-Semantic
Mesh



Pose Graph

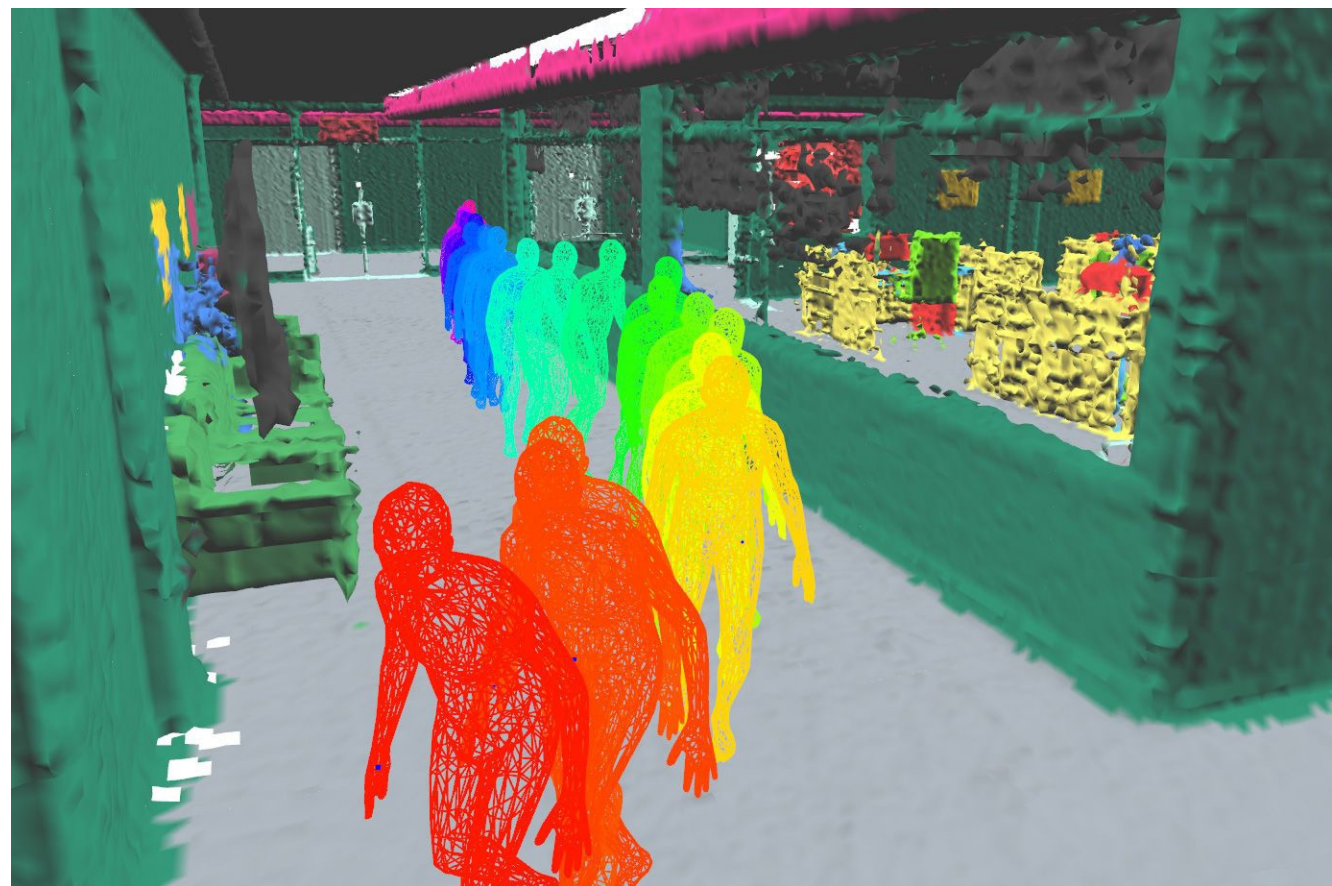
- Graph structure
- Nodes are:
 - Robot
 - Landmarks/ observations



- 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)



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)
- ...

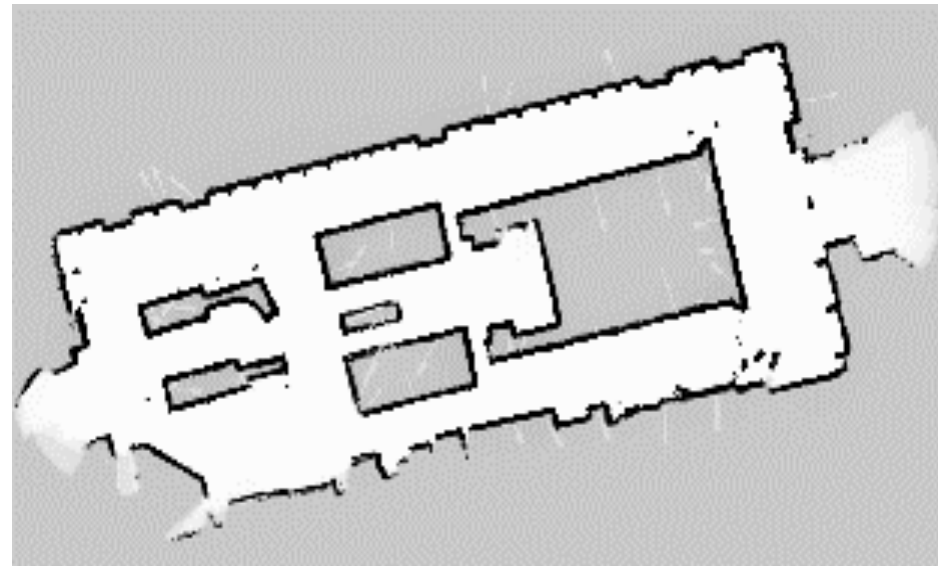
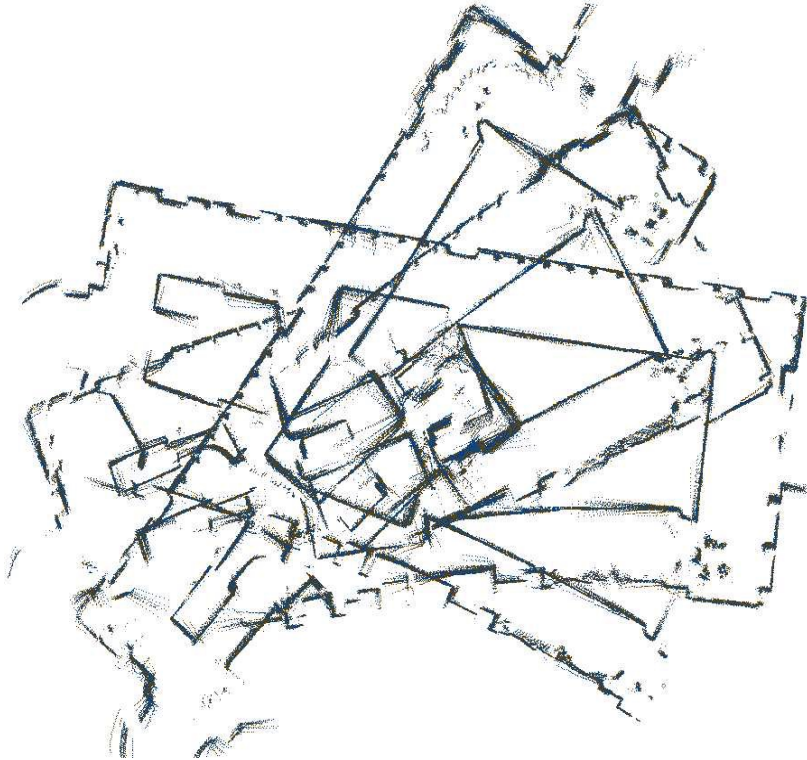
MAPPING

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

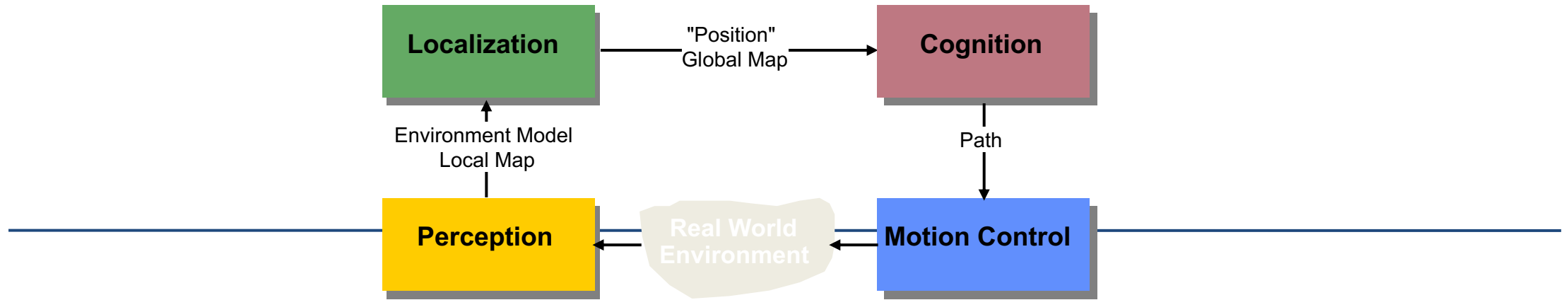
Courtesy of S. Thrun



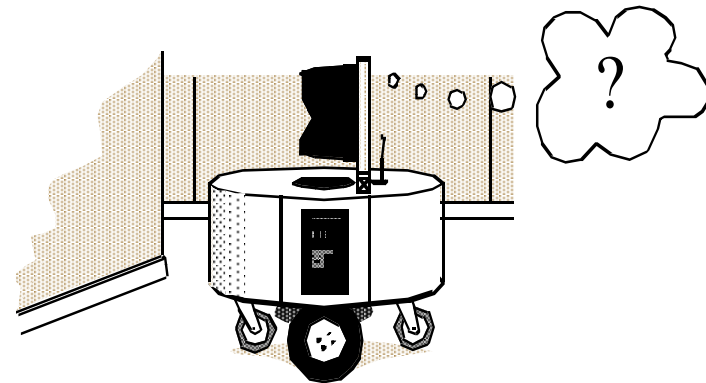
FastSLAM: Particle-Filter SLAM

Courtesy of S. Thrun





LOCALIZATION



Problem: NOISE!

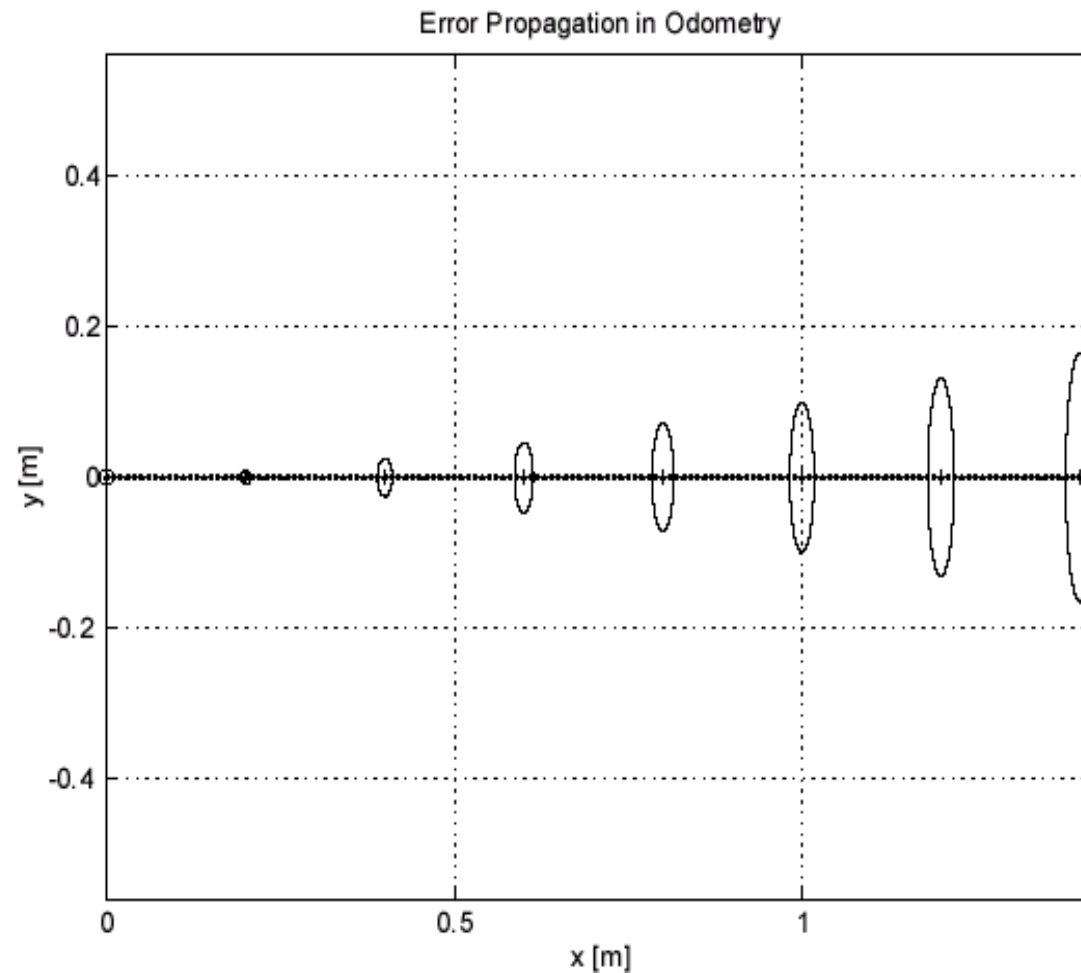
- Exteroceptive Sensor Noise
 - Sensor noise is mainly influenced by environment
e.g. surface, illumination ...
 - and by the measurement principle itself
e.g. interference two Kinects
 - Sensor noise drastically reduces the useful information of sensor readings.
The solution is:
 - to model sensor noise 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.

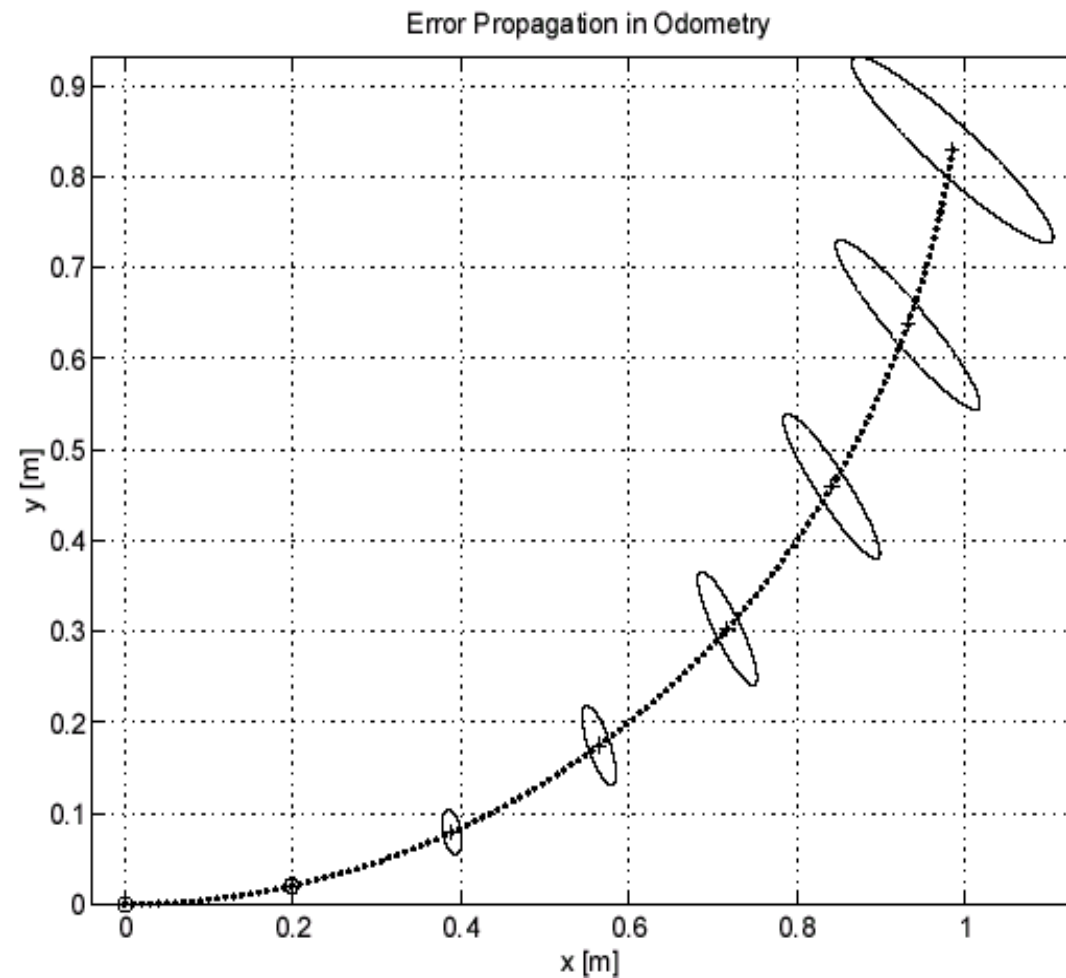
Odometry: Growth of Pose uncertainty for Straight Line Movement

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



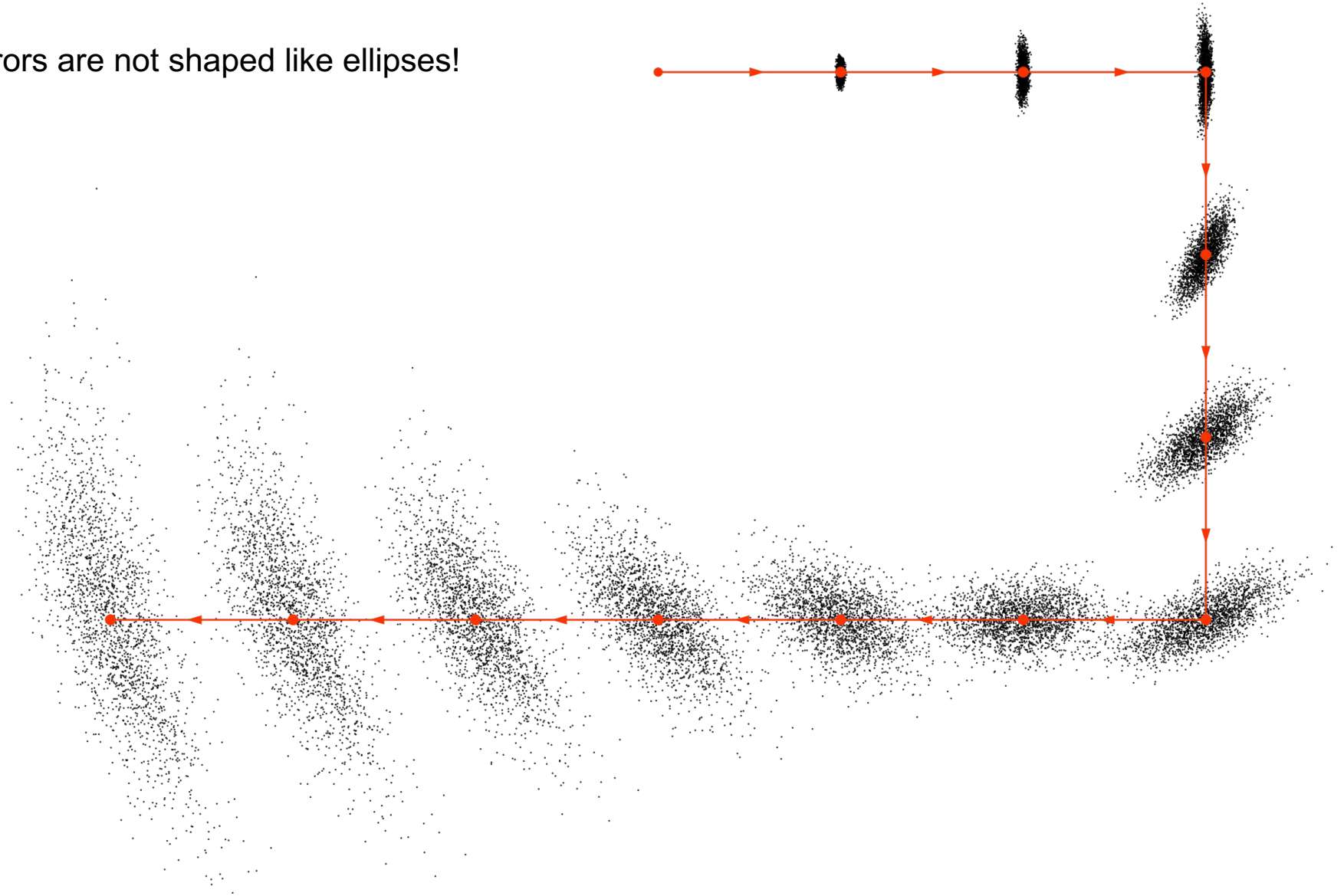
Odometry: Growth of Pose uncertainty for Movement on a Circle

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



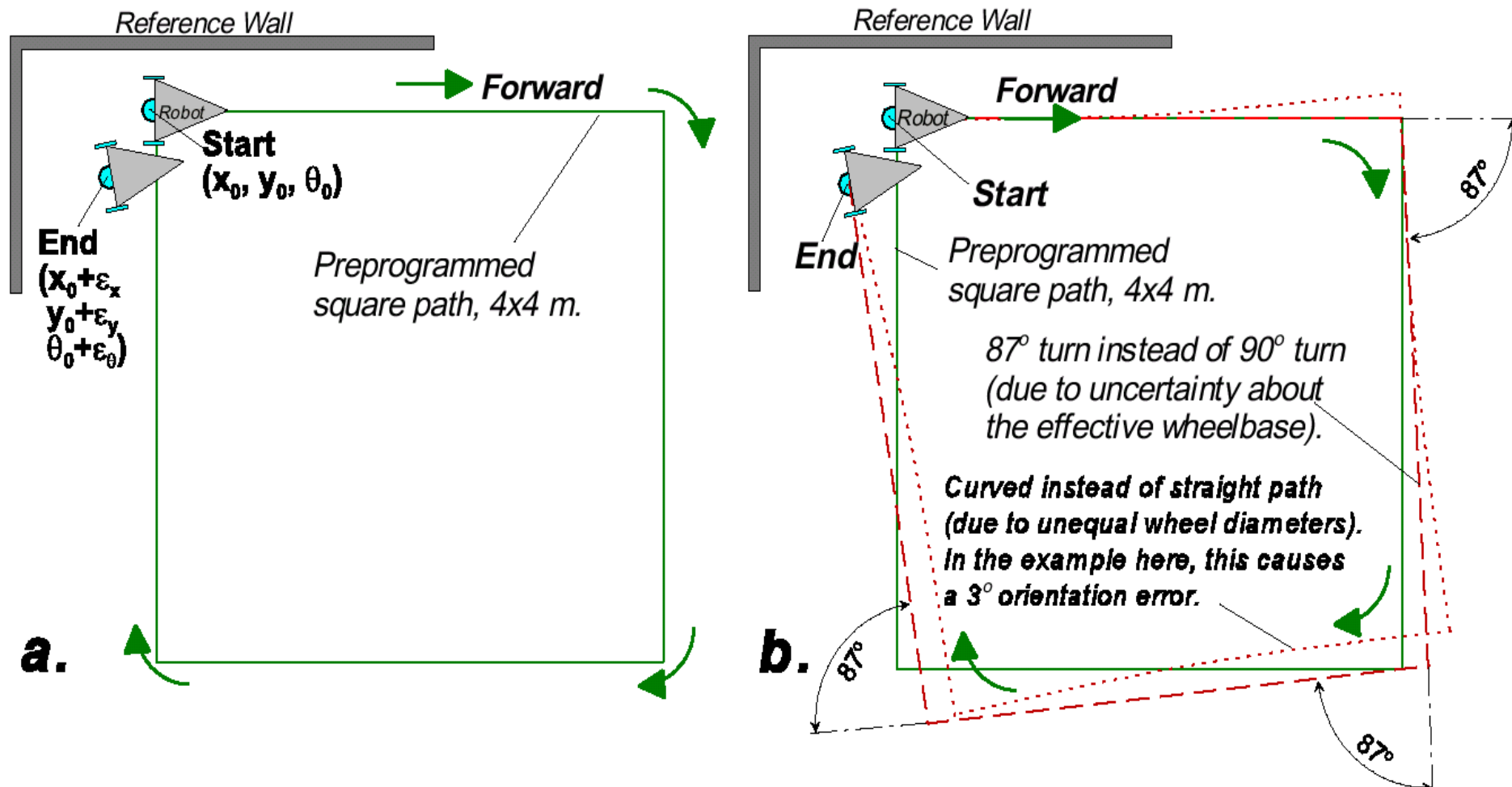
Odometry: example of non-Gaussian error model

- Note: Errors are not shaped like ellipses!



Odometry: Calibration of Errors

- The unidirectional square path experiment



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
 - Adaptive 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)