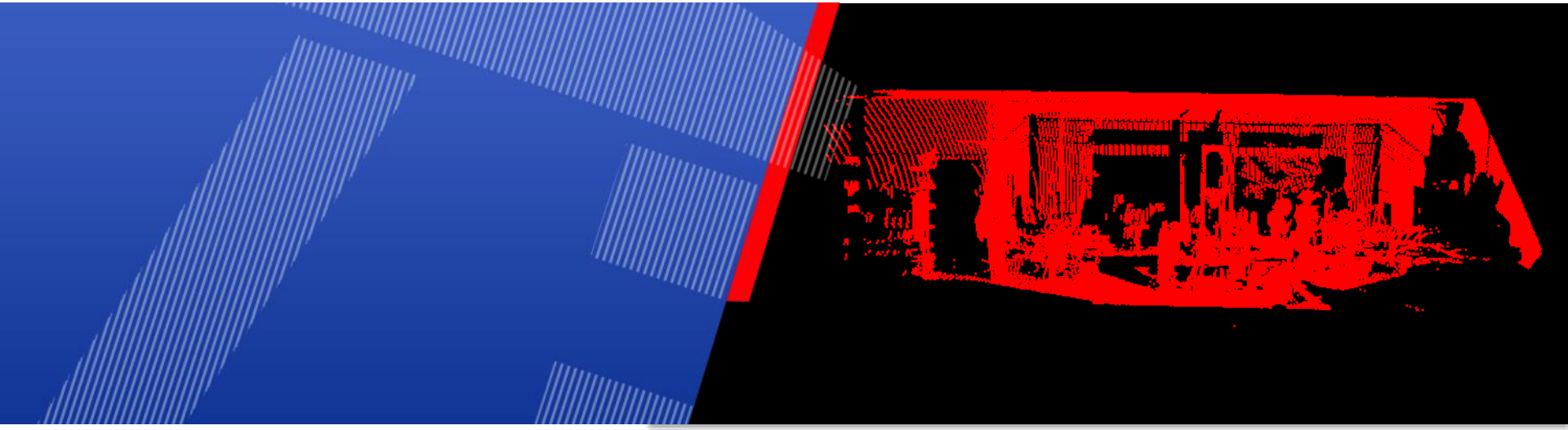
RRLAB
THE ROBOTICS RESEARCH LAB

Autonomous Mobile Robots (AMR)
7. Mapping



Prof. Karsten Berns
Robotics Research Lab
Department of Computer Science
University of Kaiserslautern, Germany

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Contents

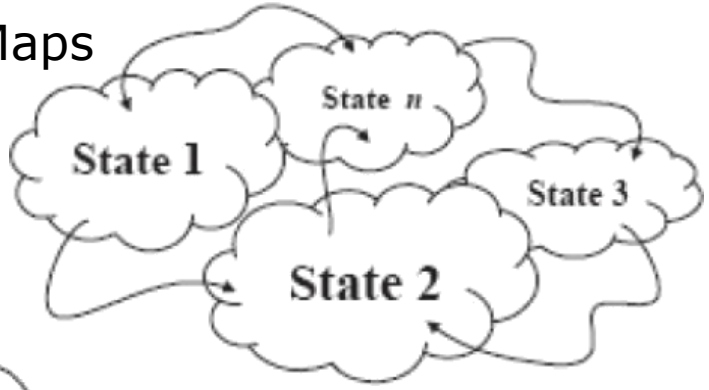
Introduction

Application of Maps

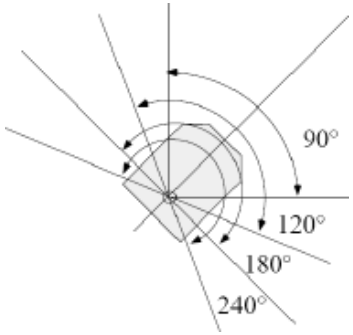
- Store information outside of sensory horizon
- Map can be provided a priori or can be build online
- Dependent on application scenario (indoor/outdoor)
- World-centric maps: navigation, path planning
- Robot-centric maps: pilot tasks (e. g. collision avoidance)
- Problem: inaccuracy due to sensor systems
- Map types
  - Metrical maps
  - Grid maps
  - Topological maps
  - Hybrid maps

Map Types

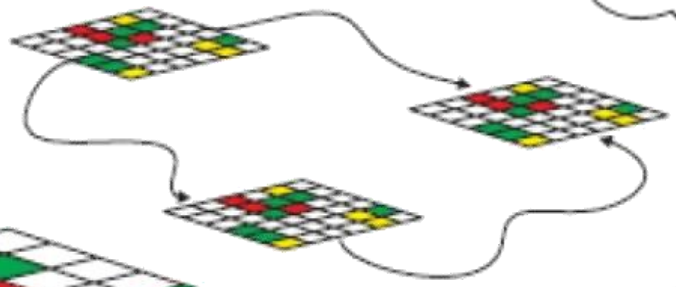
Topological Maps



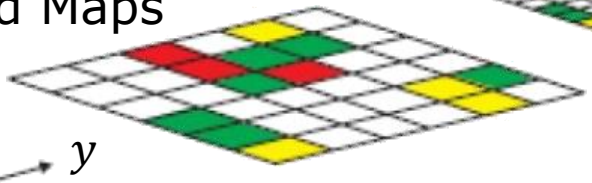
Sector Maps



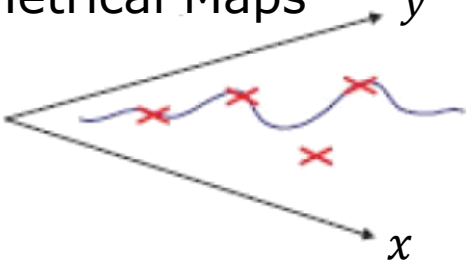
Hybrid Maps



Grid Maps



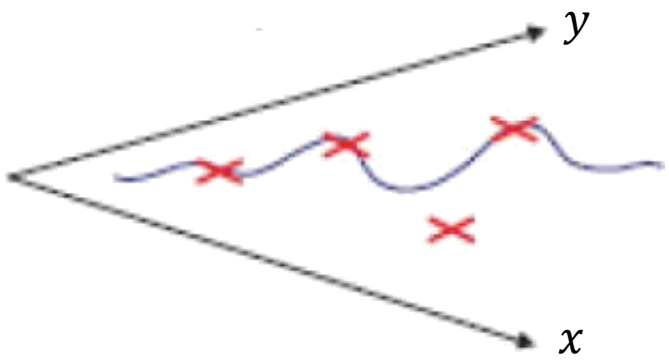
Metrical Maps



Abstraction Level

Metrical Maps

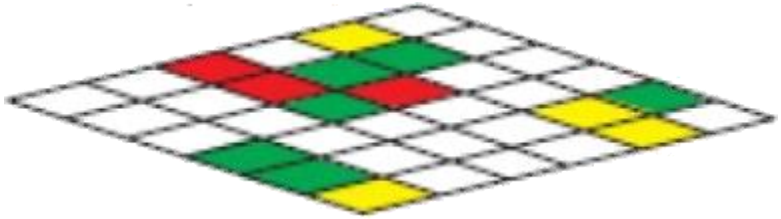
- Global, metrically consistent frame of reference
- Comparable accuracy as available sensor data
- No modeling of special places but equal importance of metrical location
- Stored features
  - Basic 3D points
  - Line features
  - Box features
  - Landmarks



- Problem: Scalability (memory consumption, computational effort for algorithms)

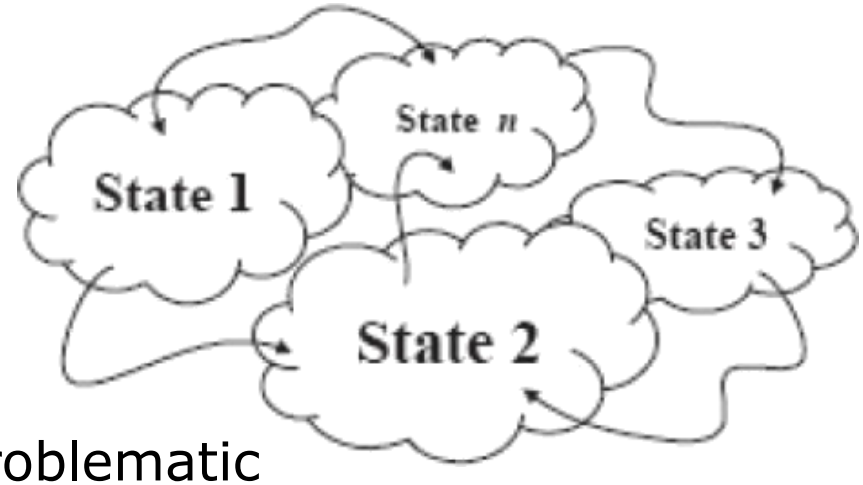
Grid Maps

- Simplicity, intuitive representation
- Divides space into adjacent portion of equal metrical sizes
- 2D map: square grid
- 3D map
  - Rubik's cube
  - Excessive storage requirements
- Variants: occupancy grid, elevation map
- Problem: Scalability (memory consumption, computational effort for algorithms)



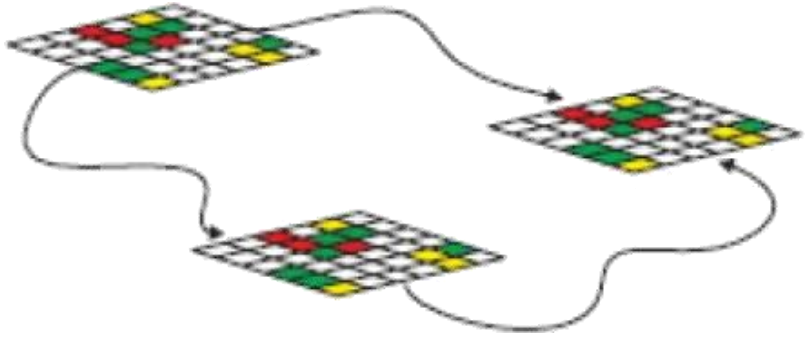
Topological Maps

- Compact, qualitative representation (overcome drawbacks of metrical maps and grid maps)
- Represent navigation-relevant places and their connections on an abstract level
- No exact metrical layout
- Advantages
  - Imprecise sensor data less problematic
  - Faster algorithms due to less data
- Underlying data structure: Graphs
  - Nodes: locations with characteristics
  - Edges: travel between nodes



Hybrid Maps

- Combine metrical and topological methodologies
- Aim: combine high precision with compact data structure



Metrical Maps

Line-based Metrical Maps

- Extract lines of distance measuring sensors
- Represent obstacles as clusters or border lines

Segmentation

- Given: Radar scan $\{r_i, \varphi_i\}\ i = 1, \ldots, n$ at position $Q = (x_0, y_0)$
- A radar point P_i is defined by (r_i, φ_i)
- Wanted: clusters C_j, segments S_k and auxiliary clusters H_m
- Result: k segments and j clusters

Init:

$C_1 := \{P_1\};\ \ i := 1;\ \ j := 1;\ \ k := 1;$

$m := 1;\ \ S_k := \{\emptyset\}; H_m := \{\emptyset\};$

…

Segmentation

…

while $i < n - 1$ do

if $|P_{i+1} - P_i| < d \Rightarrow H_m := H_m \cup P_{i+1}$; $i := i + 1$;

else if $|P_{i+2} - P_i| < d$

$\Rightarrow H_m := H_m \cup P_{i+2}$; $C_j := \{P_{i+1}\}$; $j := j + 1$; $i := i + 1$;

else if $|P_{i+2} - P_{i+1}| < d$

$\Rightarrow m := m + 1$; $H_m := \{P_{i+1}, P_{i+2}\}$; $i := i + 2$;

else $C_j := \{P_{i+1}\}$; $j := j + 1$; $m := m + 1$;

$H_m := \{P_{i+2}\}$; $i := i + 2$;

endwhile;

…

Segmentation

…

while $m > 0$ do

if $\#(H_m) \leq c \Rightarrow C_j := H_m;\ j := j + 1;\ m := m - 1;$

// $c \rightarrow \max\#$ in cluster

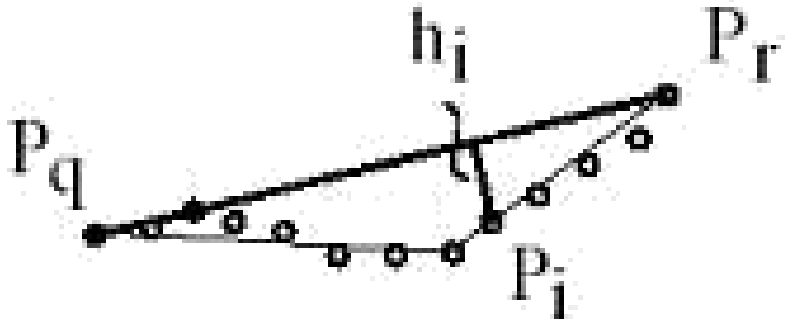
else $S_k := H_m;\ k := k + 1;\ m := m + 1;$

endwhile;

Result: k segments and j clusters

Line Generation

- Given segment S_k with nodes $P_q, \dots, P_r$
- Calculate distances $h_{r+1}, \dots, h_{q-1}$
- Assume $h_j = \max_{r<i<q} h_i > \varepsilon$ (fuzziness)
- For new line (P_r, P_j) and (P_j, P_q) else: (P_q, P_r)
- Costs: $\#lines \cdot (q - r)$



Line generation

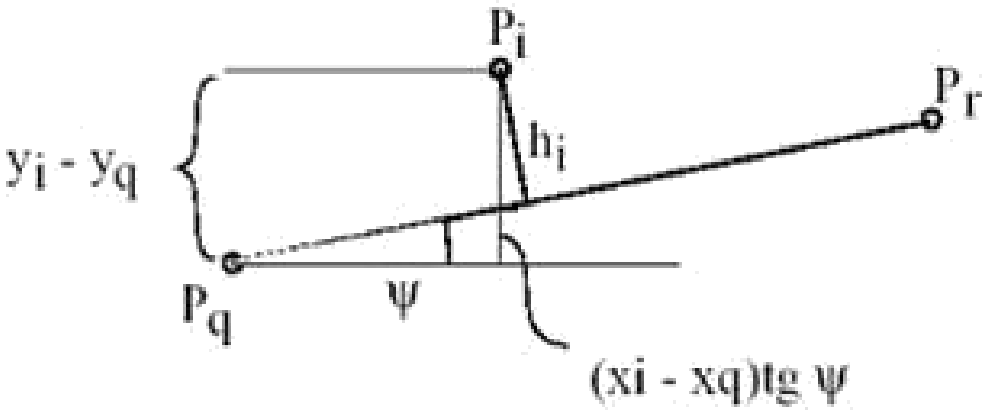
Line Generation

$$P_i = (x_i, y_i)$$

$$\tan\psi = \frac{y_q - y_r}{x_q - x_r}$$

$$h_i = \left[(y_i - y_q) - (x_i - x_q)\tan\psi\right]\cos\psi$$

$$= (y_i - y_q)\cos\psi - (x_i - x_q)\sin\psi$$



Line generation

Best Fit Straight Line

Find straight line through $P_q, \ldots, P_r$ with the min sum of squared distances:

$$h_i = (y_i - y_q)\cos\psi - (x_i - x_q)\sin\psi \text{ line } P_q, P_r$$

$$= (y_i - b)\cos\gamma - x_i \sin\gamma \text{ line through } n \text{ points and } (0, b)$$

$$\sum_i h_i^2 = \sum_i \{(y_i - b)^2 \cos^2\gamma - 2(y_i - b)x_i \sin\gamma\cos\gamma + x_i^2 \sin^2\gamma\}$$

$$\frac{\partial \sum(\ldots)}{\partial b} = \sum_i \{-2(y_i - b)\cos^2\gamma + 2x_i \sin\gamma\cos\gamma\} \overset{!}{=} 0$$

$$0 = \sum_i \{-(y_i - b)\cos\gamma + x_i \sin\gamma\} = \sum_i b - \sum_i y_i + \sum_i x_i \tan\gamma$$

Best Fit Straight Line

$$nb = \sum_i y_i - \tan\gamma \sum_i x_i \Rightarrow y_s = \tan\gamma\, x_s + b$$

Line through focal point

$$\sum_{i=1}^{N} h_i^2$$

$$= \sum_{i=1}^{N} \{(y_i - y_s)^2 \cos^2\gamma - 2(y_i - y_s)(x_i - x_s)\sin\gamma\cos\gamma + (x_i - x_s)^2 \sin^2\gamma\}$$

$$\frac{\partial \sum(\dots)}{\partial y}$$

$$= \sum^{N} \{-2(y_i - y_s)^2 \sin\gamma\cos\gamma - 2(y_i - y_s)(x_i - x_s)(-\sin^2\gamma + \cos^2\gamma)$$

Best Fit Straight Line

$$\sum_{i=1}^{N}\{(y_i - y_s)^2 \tan\gamma - (y_i - y_s)(x_i - x_s)(1 - \tan^2\gamma) + (x_i - x_s)^2 \tan\gamma\} = 0$$

$$\sum_{i=1}^{N} -(y_i - y_s)(x_i - x_s)\tan^2\gamma + \sum_{i=1}^{N}((y_i - y_s)^2 + (x_i - x_s)^2)\tan\gamma$$
$$-\sum_{i=1}^{N}(y_i - y_s)(x_i - x_s) = 0$$

$$az^2 + bz - a = 0$$

Best Fit Straight Line

Therefore we get:

$$a = \sum_{i=1}^{N} (y_i - y_s)(x_i - x_s)$$

$$b = \sum_{i=1}^{N} -(y_i - y_s)^2 + (x_i - x_s)^2$$

$$z = \tan\gamma \rightarrow z_{1,2} = -\frac{b}{2a} \pm \sqrt{1 + \frac{b^2}{4a^2}}$$

Straight Line Adjustment

- Assume $\psi \approx \gamma$

$$h_s = (y_s - y_q)\cos\psi - (x_s - x_q)\sin\psi$$

$$\sum (h_i - h_s) = \sum (y_i - y_s)\cos\psi - (x_i - x_s)\sin\psi$$

$$0 = \cos\psi\left(\sum y_i - N\sum y_s\right) - \sin\psi\left(\sum x_i - N\sum x_s\right)$$

- Line adjustment h_s

$$P_q \rightarrow Q = (x_q + \Delta x, y_q + \Delta y)\Delta x = h_s \sin\psi$$
$$P_r \rightarrow R = (x_r + \Delta x, y_r + \Delta y)\Delta x = h_s \cos\psi$$

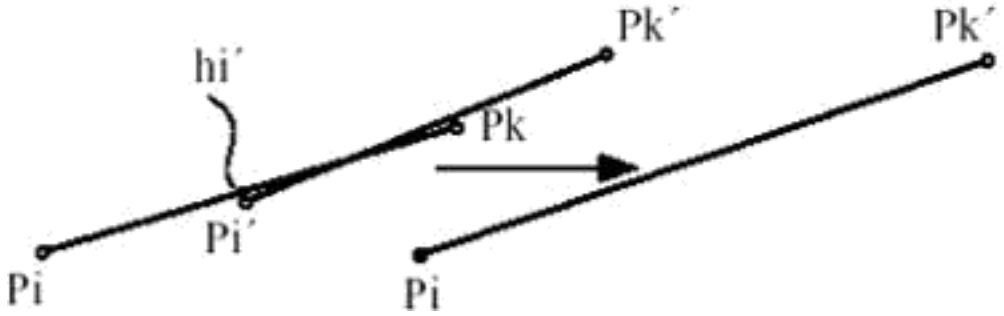
- The points P, Q and the focal point (x_s, y_s) are element of the found best fit line

Line Merging

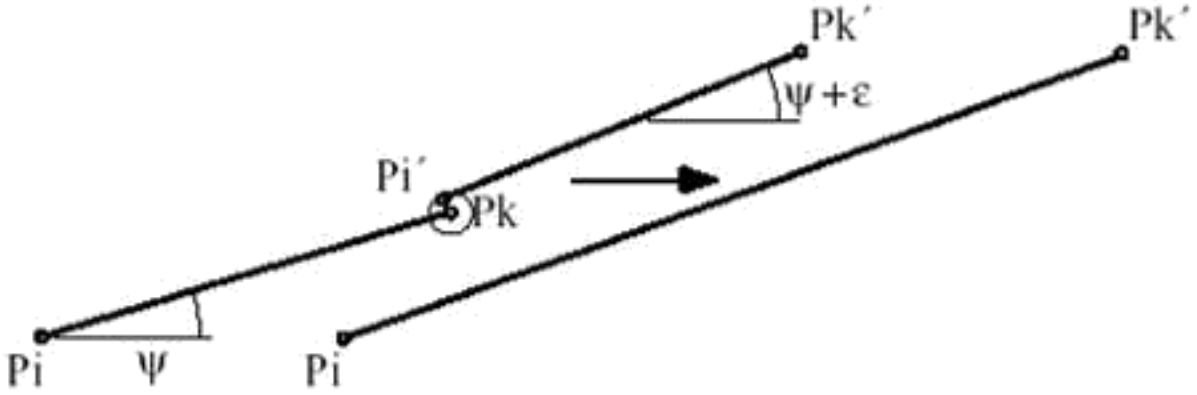
- Merging of lines from different scans to one new line
- Given: R_1 and R_2 two radar scans at positions P_0 and $P_{0'}$
- Assume: line (P_i, P_k) from R_1 and line $(P_{i'}, P_{k'})$ from R_2 with $\psi' = \psi + \varepsilon$

Line Merging

- If $h_i' < \delta$ for P_i' in between P_i and P_k and $|P_i', P_k'| > |P_i', P_k|$
  $\Rightarrow$ merge to one line P_i, P_k'



- If $|P_k, P_i'| < d \Rightarrow$ merge to one line P_i, P_k'



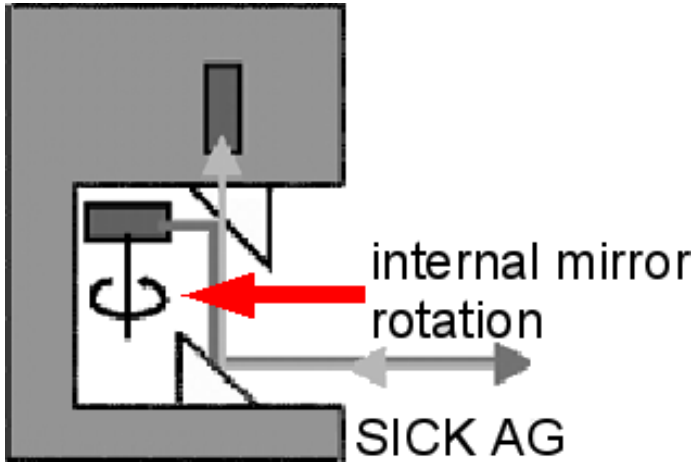
Plane-based Metrical Maps

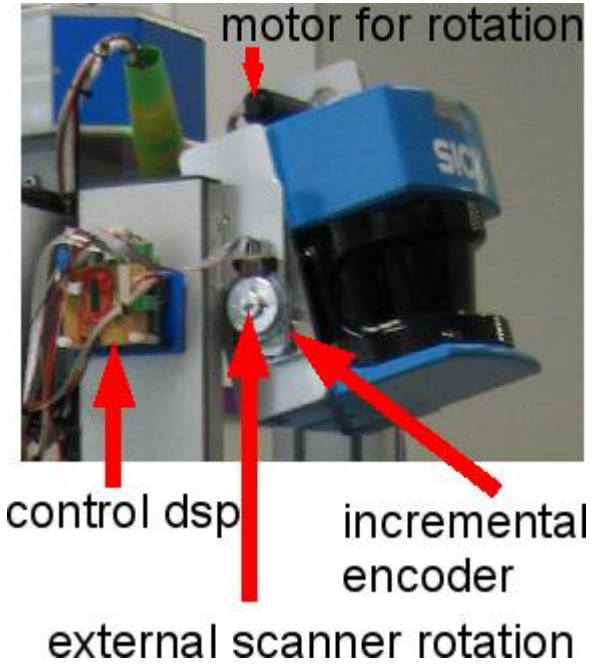
Scenario:

- Autonomous exploration of indoor environments
- Map building
- Detecting objects of daily use
- Extraction of room and furniture primitives from 3D distance measurements
- Matching plane patches to point clouds
- Landmarks for re-localization and navigation
- Hints for interesting places during object exploration (table tops, shelves)

3D Scanner Device

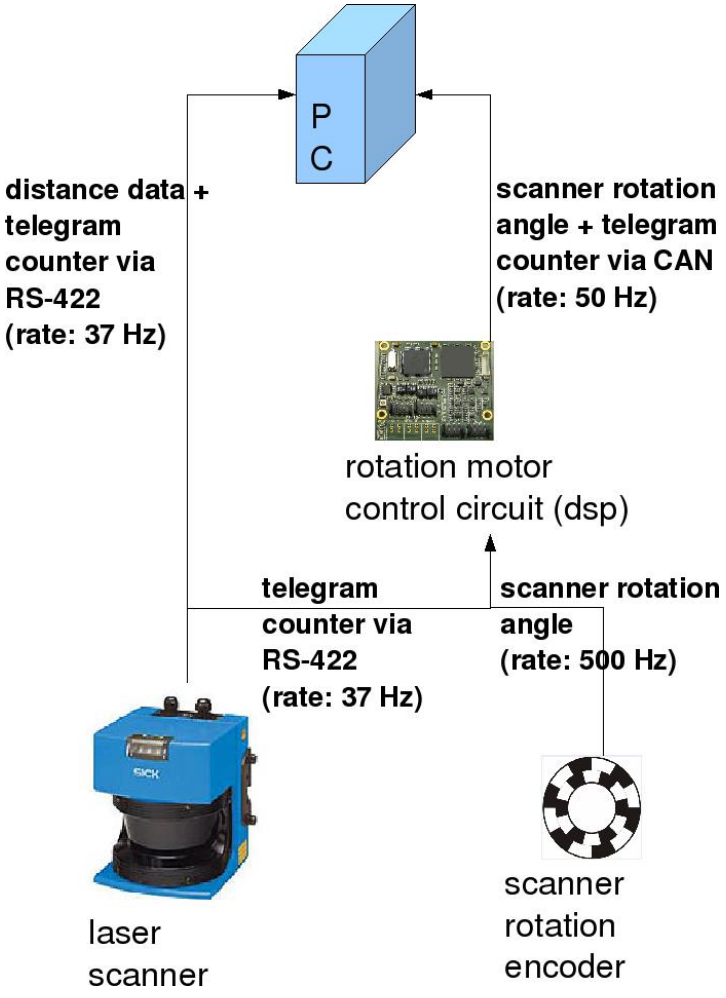
- Planar distance measurements via SICK LMS 200
- 3D information via rotation of whole scanner device





Sensor Data Processing

- Problem: internal mirror and external scanner rotations are independent
- Solution
  - Motion control dsp tags external rotation angles with scan telegram counter
  - Fusion of 2D measurement data and external position information via telegram index in PC

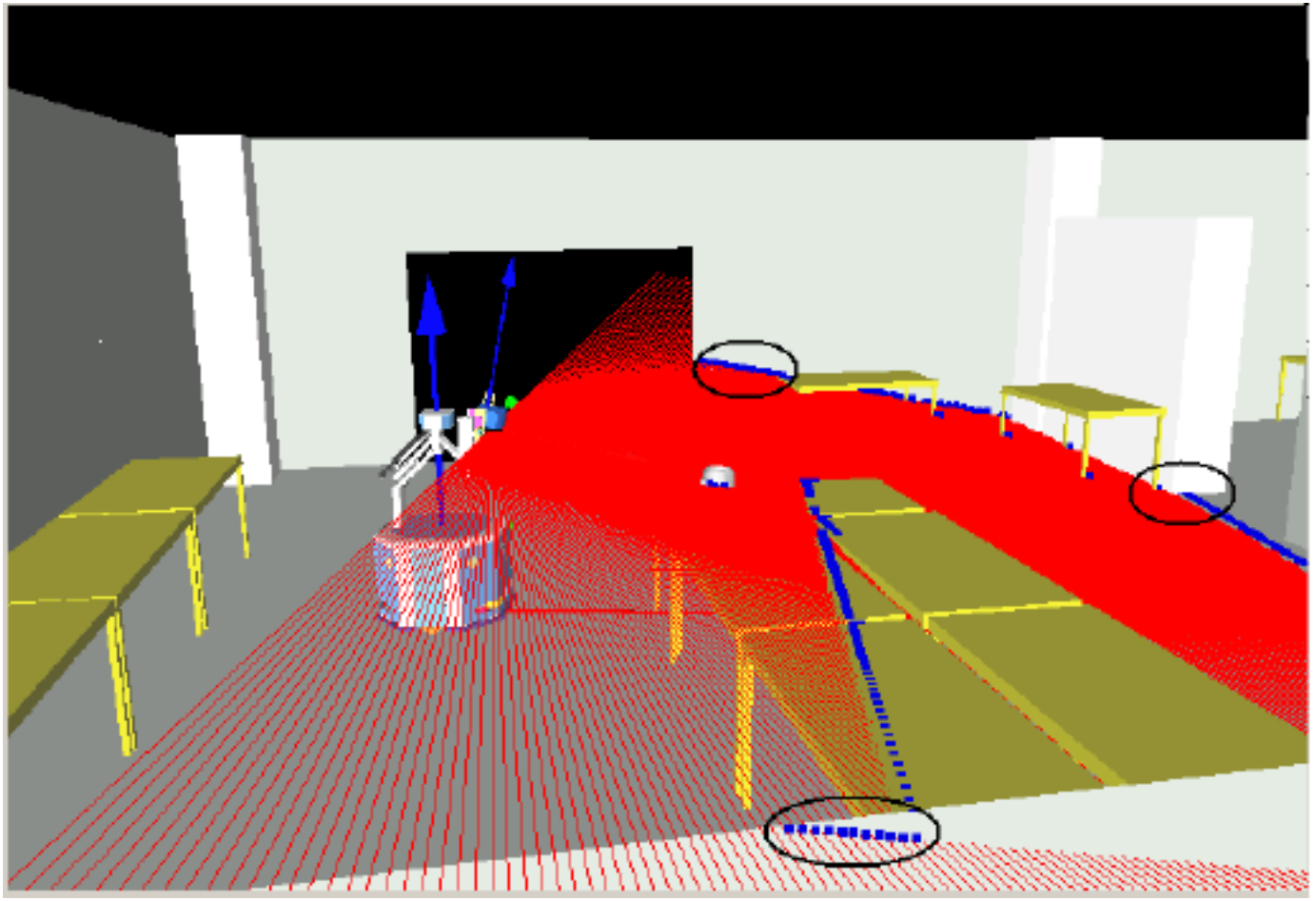


Sensor data processing

Virtual 3D Scanner in SimVis3D

- Goal: test feature extraction strategy offline in realistic 3D simulation with definable measurement conditions
- New virtual sensor device within the SimVis3D framework
  - Distance calculation by ray-tracing too slow
  - Evaluation of depth buffer information from offscreen-rendered perspective camera
  - 2 camera viewpoints to cover $180\,°$ viewing angle of scanner
  - "Real-time" scan rate ($20\,\mathrm{Hz}$)
  - Gaussian noise for realistic distance samples

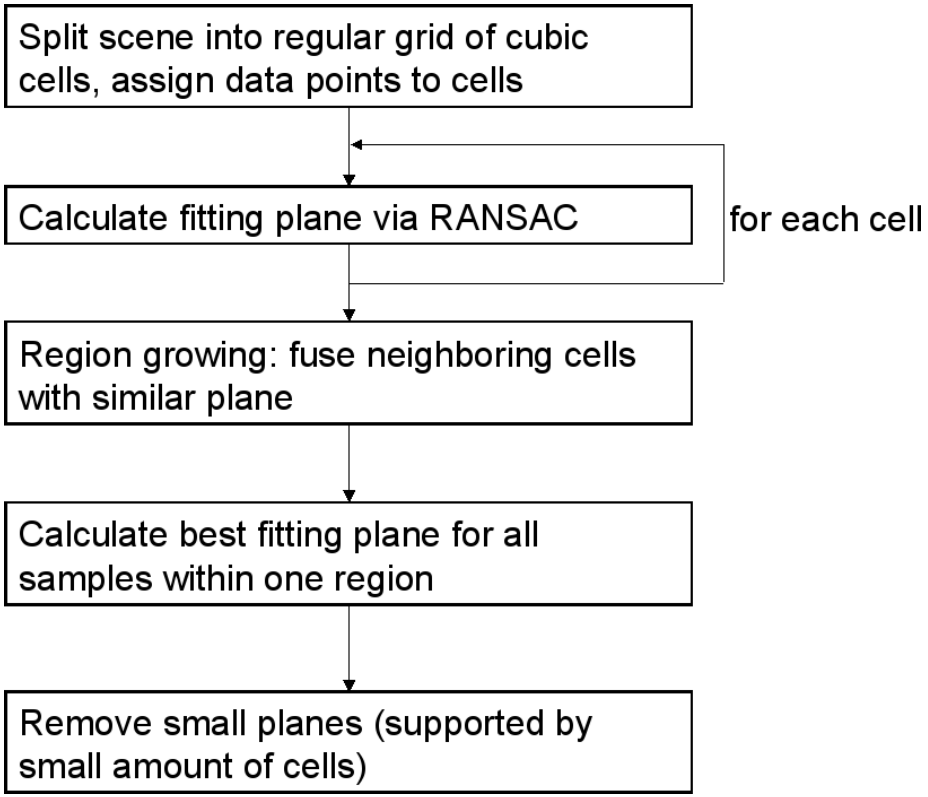
Example



Virtual Indoor Scene

Plane Extraction

- Planes as features: floor, ceiling, walls, doors, etc.
- Main problem: extraction of planes
- Typical algorithm [Weingarten03]: based on 3D point cloud
- Approximate point cloud by a set of planar patches (optimal in respect to least square)
- Reduce large features (e. g. corridor walls) to one planar patch



Plane extraction strategy

Plane Extraction Algorithm: RANSAC

Given: A set of 3D distance measurement samples (data points)

Return: A set of planes approximating disjunctive subsets of the input points

Step 1: Split the whole 3D scene into a regular grid of cubic cells

Step 2: Assign the input points to the corresponding cells

Step 3: Calculate fitting plane for each cell:

for every cell **do**

Find approximating plane using RANSAC algorithm (output: best fitting plane after certain number of RANSAC iterations)

Remove outliers with respect to the best RANSAC plane
Calculate least-square fitting plane for inliers

end for

Plane Extraction Algorithm: RANSAC

Step 4: Region growing – fuse matching planes of neighboring cells:

for every cell **do**

compare plane parameters with those of all neighboring cells
if angle between plane normals below angular threshold and
distance of center of gravity of points of neighboring cell to plane
below distance threshold **then**

mark both cells as belonging to same region

end if

end for

for all regions **do**

calculate best fitting plane for all points of cells belonging
to the same region

end for

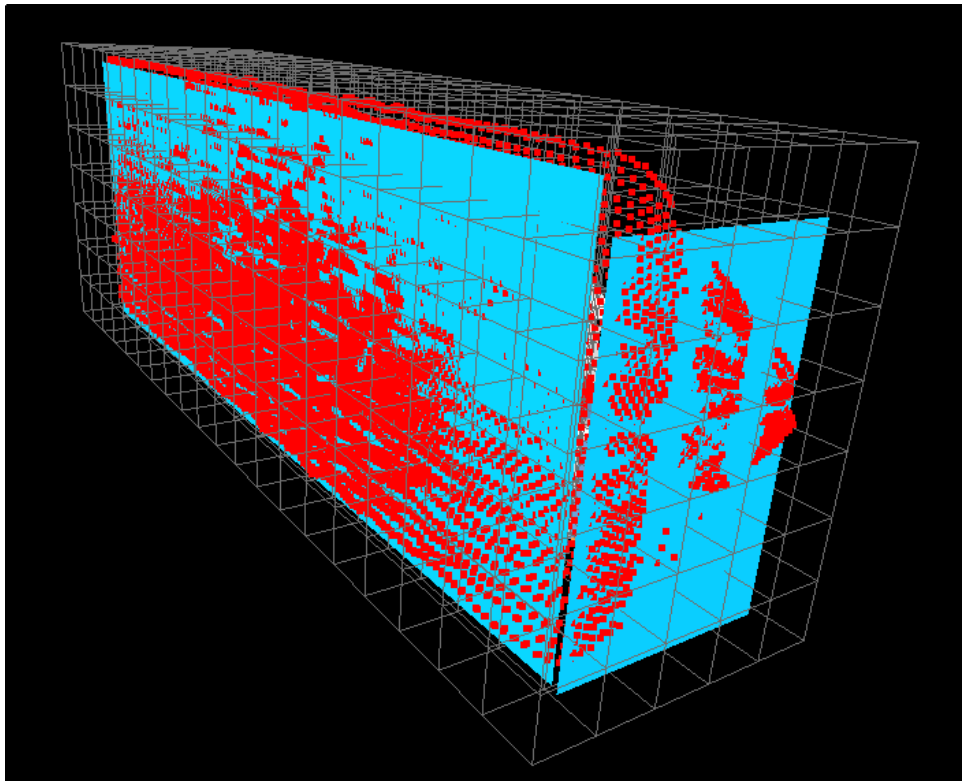
Step 5: remove small planes supported only by a small amount of cells

Plane Extraction Algorithm

- Repeatedly calculates planes approximating local set of points within one cell
- Randomly select 3 points within local set, calculate spanning plane, count the number of points within certain distance to plane
- Calculated plane is used as best plane if number of supporting points is higher than for previous best plane
- Restrict computation time: typically not more than 100 RANSAC (*Ran*dom *Sa*mple *C*onsensus) iterations
- No approximating plane found ⇒ no merging, filtered out

Step 1/2: Split Scene into Cells and Assign Samples

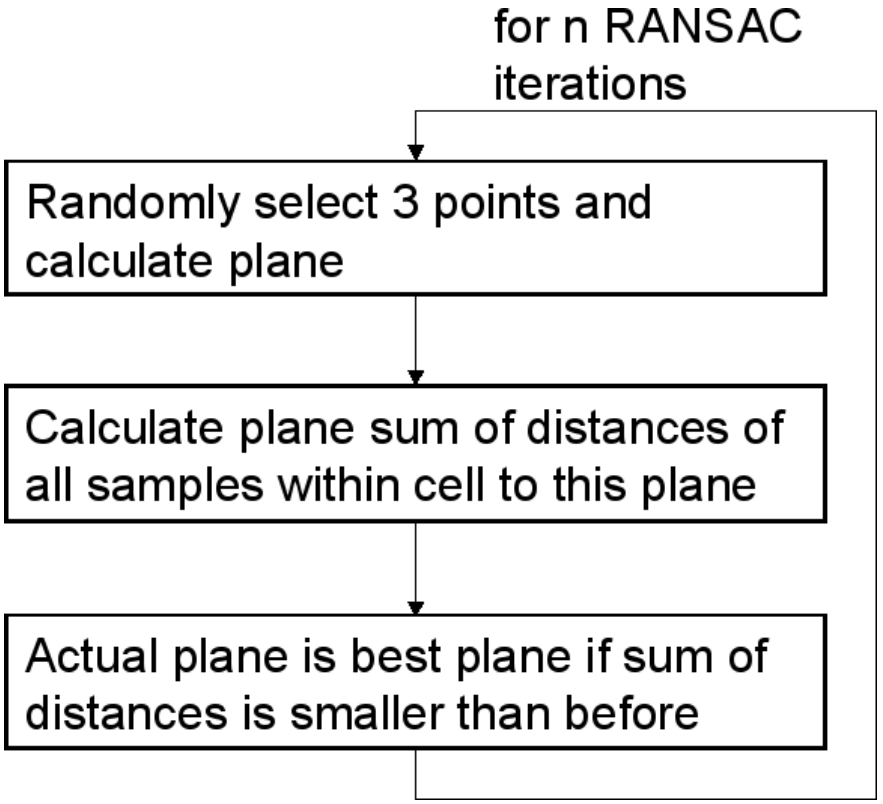
Cells stored in sorted order (octree data structure)



Samples assigned to cubic cells (extracted planes only shown for better visual impression)

Step 3: RANSAC-Based Plane Extraction

- Calculated for each cell
- Can be done in parallel for all cells
- Assumes existence of a plane (convergence)
- Number of iterations depends on point-to-cell ratio (adjustable, via experiments set to 50)
- "Bad" planes are only supported by single cells ⇒ automatically filtered out in step 3 (region growing)



Plane Extraction via RANSAC

Step 3: RANSAC-Based Plane Extraction

Postprocessing:

- For each cell: remove outliers with respect to best RANSAC plane
- Calculate least-square fitting plane for inliers via PCA

Step 3: Calculation of Least Square Fitting Plane

Principal Component Analysis:

- Given: set of samples
- Search for linear transform (new coordinate system) such that greatest variance comes to lie on first (principal) axis, the second greatest on the second, …
- Applied to range measurement samples (3D points) of a plane: first two axes lie in the plane, third one is the plane normal (direction of smallest variance)

Principal Component Analysis

- Plane described as $\langle \vec{n}, \vec{x} \rangle = d$
- 4 parameters
  - Normal $\vec{n} = (n_x, n_y, n_z)$
  - Distance from origin d
- Regression problem with N samples (3D points) $\vec{x}_i = (x_i, y_i, z_i)$:

$$R(n_x, n_y, n_z, d) = \sum_{i=1}^{N} w_i (n_x x_i + n_y y_i + n_z z_i - d)^2$$

  with weights w_i representing measurement error of each sample (set to 1 in the following)
- Due to $\langle \vec{n}, \vec{x} \rangle = d$, (n_x, n_y, n_z, d) have to be chosen such that R is minimized

Principal Component Analysis

Derivation wrt. d yields:

$$\partial_d R(n_x, n_y, n_z, d) = 0$$

$$\Leftrightarrow 2 \sum_{i=1}^{N} w_i (n_x x_i + n_y y_i + n_z z_i - d)(-1) = 0$$

$$\Leftrightarrow \sum_{i=1}^{N} w_i (n_x x_i + n_y y_i + n_z z_i) = \sum_{i=1}^{N} w_i d$$

$$\Leftrightarrow \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i (n_x x_i + n_y y_i + n_z z_i) = d$$

Principal Component Analysis

$$\Leftrightarrow \frac{1}{\sum_{i=1}^{N} w_i} \begin{bmatrix} \sum_{i=1}^{N} w_i x_i \\ \sum_{i=1}^{N} w_i y_i \\ \sum_{i=1}^{N} w_i z_i \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = d$$

$$\Leftrightarrow o \cdot n = d$$

$\Rightarrow$ best fitting plane passes through the center of gravity $o = (o_x, o_y, o_z)^T$ of the samples

In case of $w_i = 1$ o is simplified to $o = \frac{1}{N} \cdot \begin{bmatrix} \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} z_i \end{bmatrix}$

Principal Component Analysis

- Translating the samples by $-o$ into the origin yields

$$S(n_x, n_y, n_z) = \sum_{i=1}^{N} w_i \left(n_x \hat{x}_i + n_y \hat{y}_i + n_z \hat{z}_i\right)^2$$

- With input samples centered around cog (mean value)

$$\hat{x}_i = x_i^{raw} - o_x \qquad \hat{y}_i = y_i^{raw} - o_y \qquad \hat{z}_i = z_i^{raw} - o_z$$

- Calculation of normal $\vec{n}$ via PCA

Principal Component Analysis

- Covariance matrix of transformed samples

$$A = \begin{pmatrix} \sum_{i=0}^{N} w_i \hat{x}_i^2 & \sum_{i=0}^{N} w_i \hat{x}_i \hat{y}_i & \sum_{i=0}^{N} w_i \hat{x}_i \hat{z}_i \\ \sum_{i=0}^{N} w_i \hat{x}_i \hat{y}_i & \sum_{i=0}^{N} w_i \hat{y}_i^2 & \sum_{i=0}^{N} w_i \hat{y}_i \hat{z}_i \\ \sum_{i=0}^{N} w_i \hat{x}_i \hat{z}_i & \sum_{i=0}^{N} w_i \hat{y}_i \hat{z}_i & \sum_{i=0}^{N} w_i \hat{z}_i^2 \end{pmatrix}$$

- A is symmetric and positive definite ⇒ can be diagonalized
- Eigenvectors of A represent principal axes
- $\vec{n}$ = Eigenvector of smallest eigenvalue of covariance matrix (direction of smallest variance)

Results of Step 3

- For each cell containing enough samples:
- Best fitting plane through all inliers
- Inliers calculated with best RANSAC plane

Step 4: Region Growing, Merged Plane Fitting, Filtering

- Comparison of neighboring planes based on distance to origin and normal vector
- All cells with similar planes belong to same region
- Within each region: PCA for calculating least square fitting plane for all samples in this region
- Omit planes supported by small amount of cells (depends on focused features, e. g. room walls, chairs, screens)

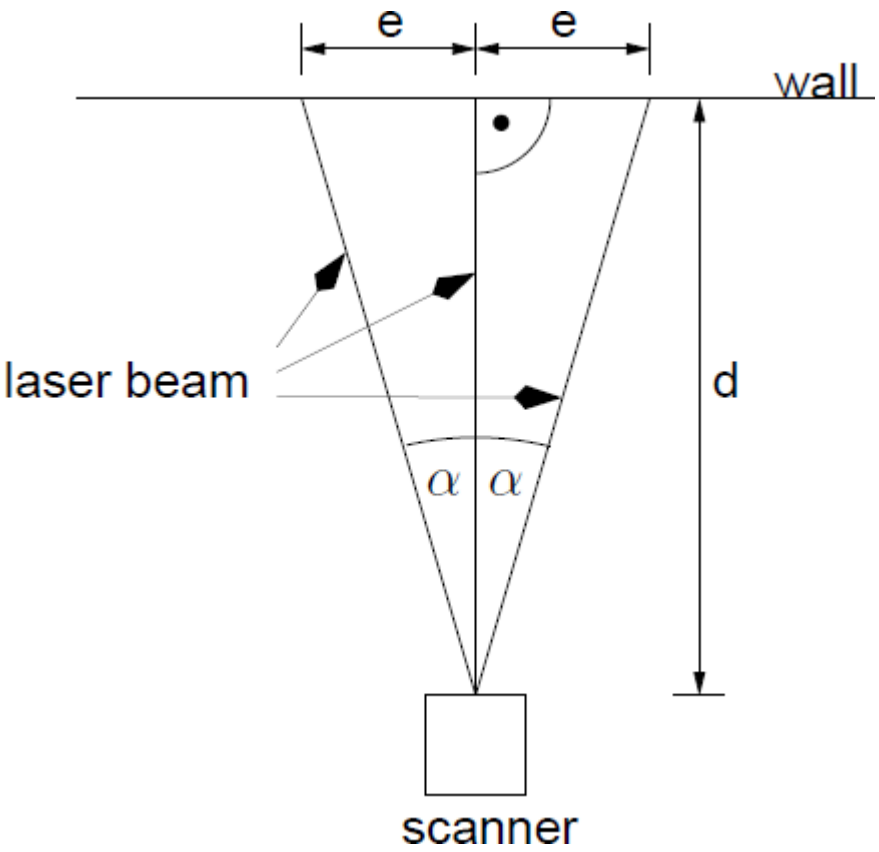
Parameter Summary

Step	Parameter	Value
1	Cell size	200 mm
3	Min. amount of points in a cell to start RANSAC algorithm	10
	Number of RANSAC iterations	50
	Max. point-to-plane distance for inliers	50 mm
4	Angular threshold for normals of neighboring planes	15 °
	Distance threshold for cog of one plane to neighboring plane	50 mm
5	Min. amount of supporting cells for plane filtering	10

Example

- Scanner in front of wall with distance d
- Mid laser beam perpendicular to wall
- Scan resolution $\alpha = 0.5\,°$
- Cell size $20 \times 20 \times 20\ \mathrm{cm}^3$
- $N = 9$ samples per cell (3×3 grid) $\Rightarrow e = 10\ \mathrm{cm}$

$$d = \frac{e}{\tan \alpha} \Rightarrow d \approx 11.46\ \mathrm{m}$$



Scan resolution

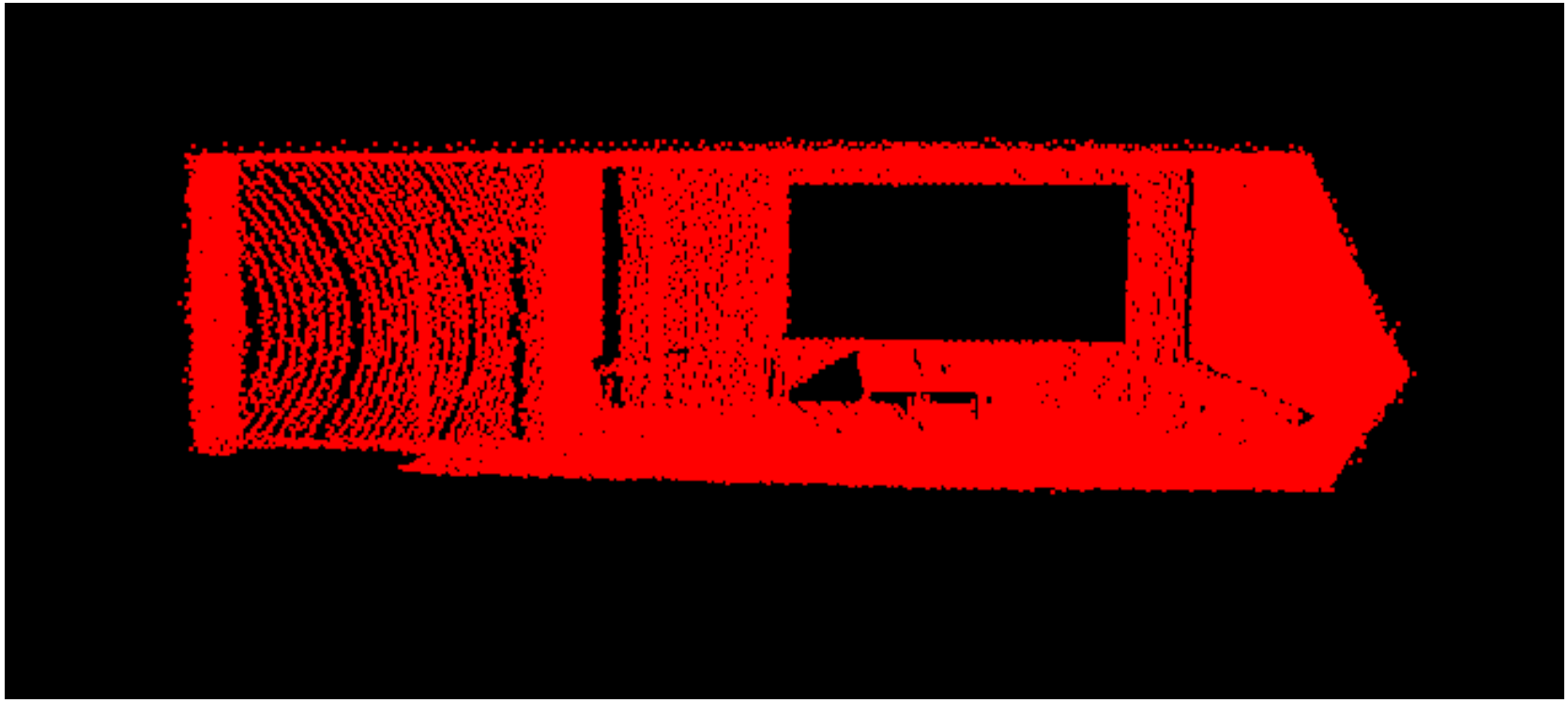
Complexity Analysis

- Step 1/2
  - Space decomposition into k cubic cells
  - Distribution of n samples within these cells
  - Cells are stored in sorted order (octree) $O(\log k)$
  - $\Rightarrow O(n \cdot \log k)$
- Step 3
  - RANSAC-based plane extraction in each cell with n_j samples $\Rightarrow O\left(k \cdot n_j^2\right)$
  - Can be reduced to $O(k)$ if number of samples n_j in one cell is assumed constant
  - PCA is executed in $O(n)$
- Step 4: Region growing executed in $O(n)$ (sorted cells)
- Total: $O(n \cdot \log k)$

Experiments: Setup

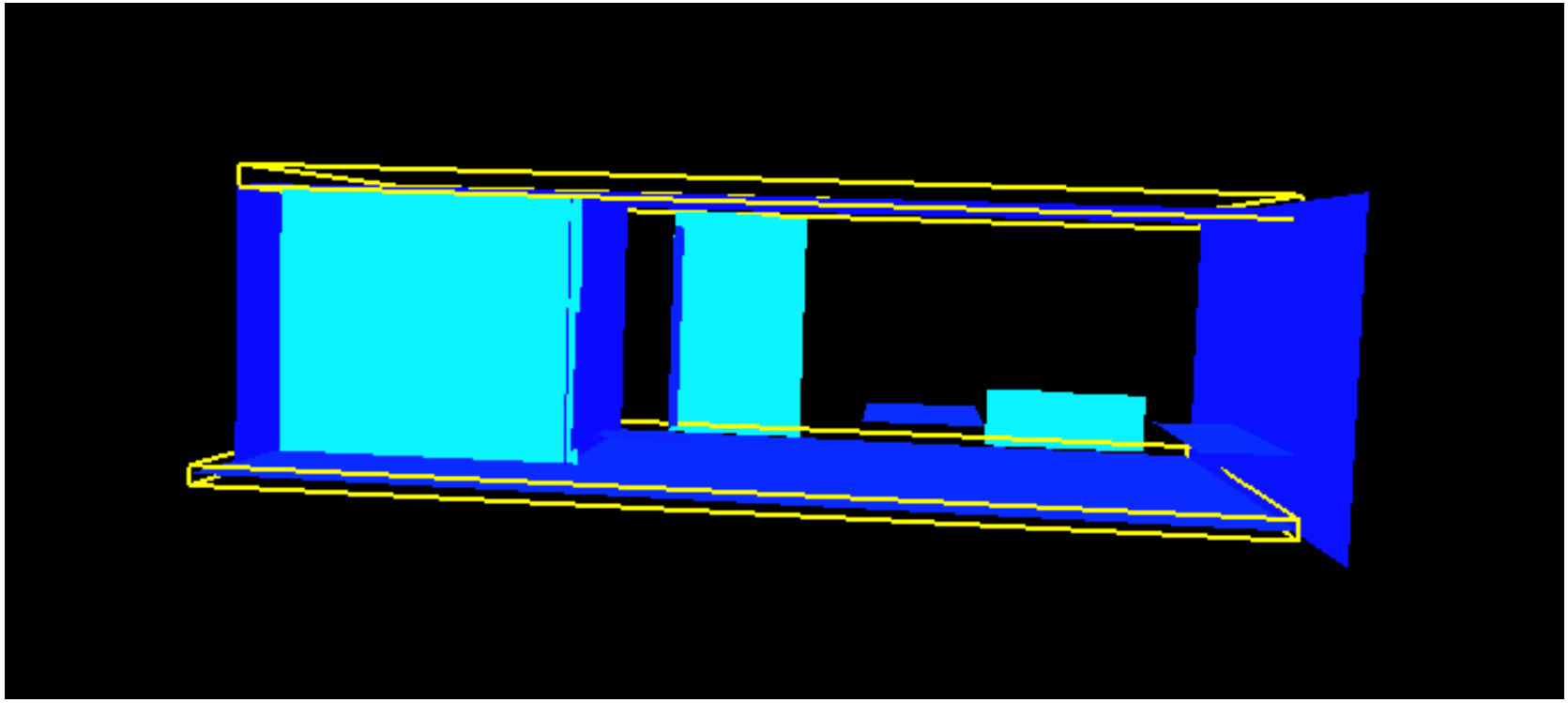
- Input data
  - 3D point cloud from one scan sweep from $-40° \dots 40°$
  - Scan resolution: $0.5°$ horizontal, $0.3°$ vertical
  - ~ 92000 samples
- Scenarios
  - Virtual 3D scene of Robotics Research Lab (simulation, less clutter)
  - Real Robotics Research Lab (typical indoor scenario)
- Time for plane extraction: $< 2\,s$
- Clipping of final planes by bounding boxes of supporting points

Experimental Result: Simulation



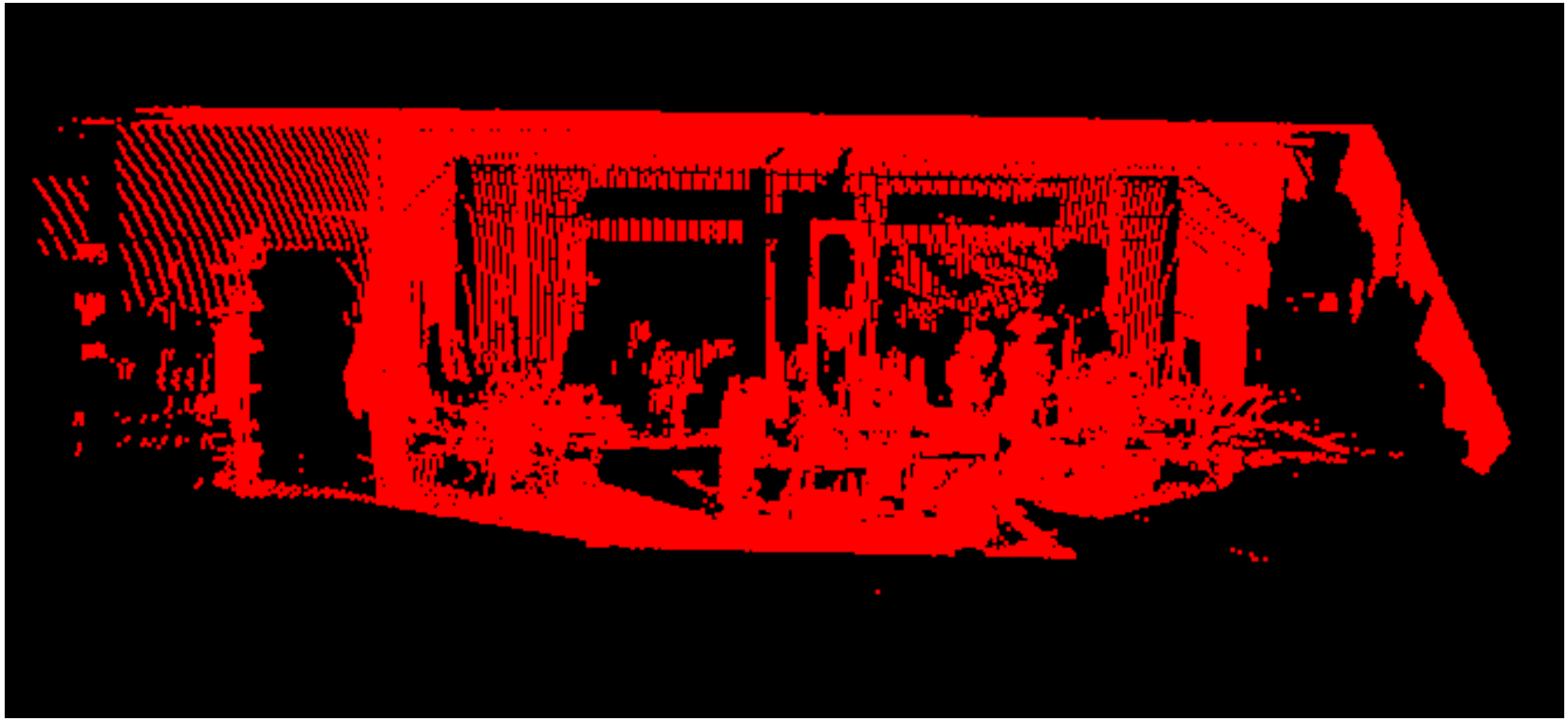
3D Point Cloud

Experimental Result: Simulation



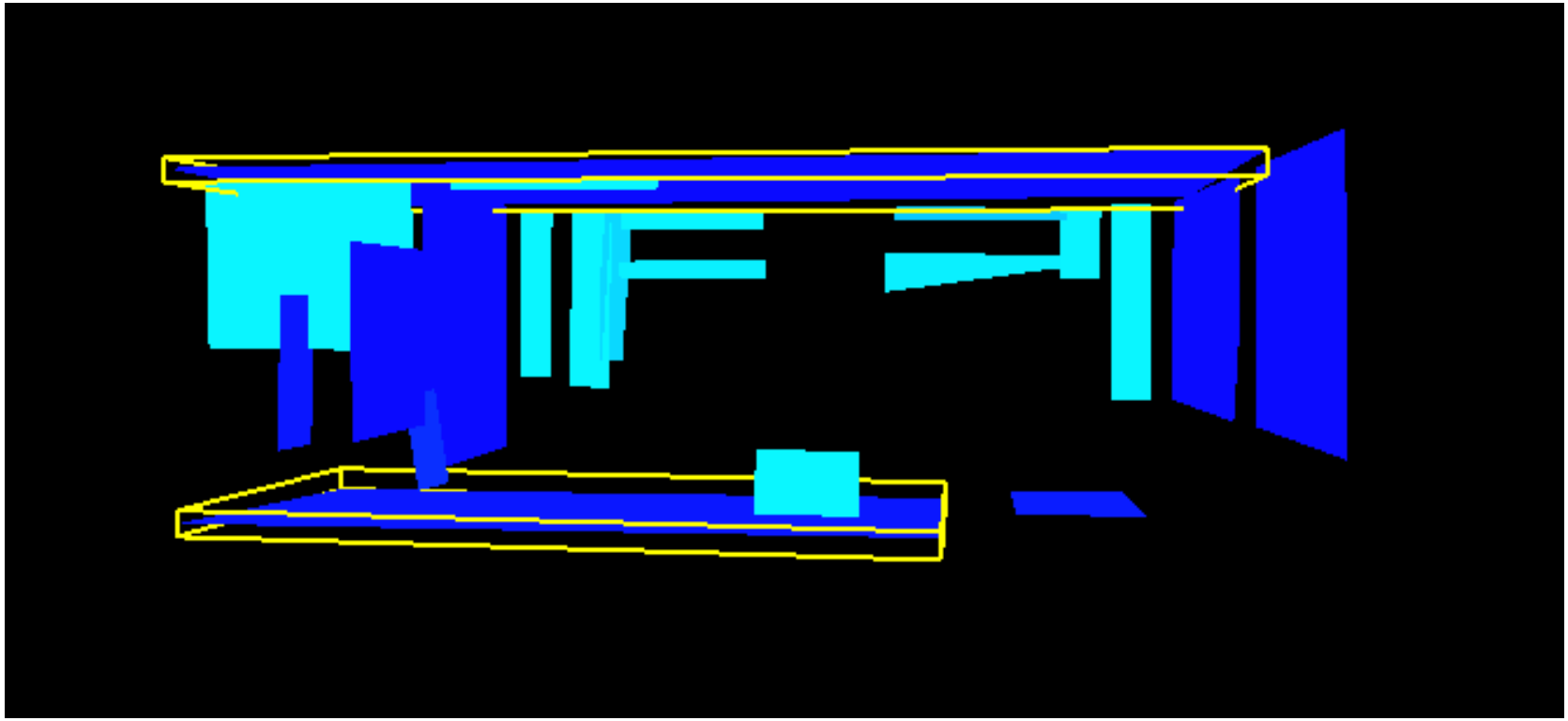
Extracted Planes (floor and ceiling marked)

Experimental Result: Robotics Research Lab



3D Point Cloud

Experimental Result: Robotics Research Lab



Extracted Planes (floor and ceiling marked)

Extraction of Room Walls from 3D Scans

- Goal: reliable detection of room walls in real world scenarios for topological mapping
- Motivation: wall detection with 2D scanner fails in highly cluttered areas
- Strategy: extract features from planes based on their orientation, size and relation to each other

Extraction of Room Walls from 3D Scans

1) Group planes into horizontal ones, vertical ones and others
   criterion: direction of plane normal

2) Lowest horizontal plane: floor, highest horizontal plane: ceiling (choose biggest one if there are several distinct plane patches with similar distance from origin) criterion: plane distance from origin

3) Group vertical planes into walls, wall candidates and others
   criterion: distance of vertical plane from floor and ceiling

Extraction of Room Walls from 3D Scans

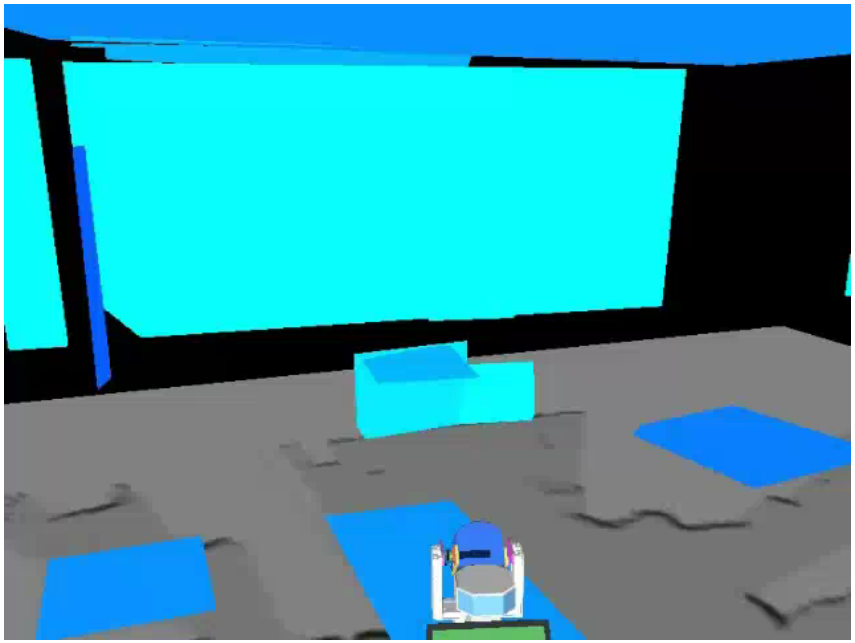
- Vertical planes connected to floor and ceiling are regarded as walls
- Vertical planes connected to floor or ceiling are regarded as wall candidates
- Vertical planes without connection to floor or ceiling: parts of chairs, screens
- Determination of “connectivity” based on distance threshold

Extraction of Room Walls from 3D Scans

- Walls and wall candidates are used as room wall features for topological map building
- Motivation: due to occlusions via furniture objects most stable features are vertical walls connected to ceiling
- Example
  - Experiment in real lab scenario
  - Robot takes 4 consecutive $3D$ scans from fixed position in 4 directions with $90°$ orientation offset each
  - Plane extraction and feature detection with complete point cloud

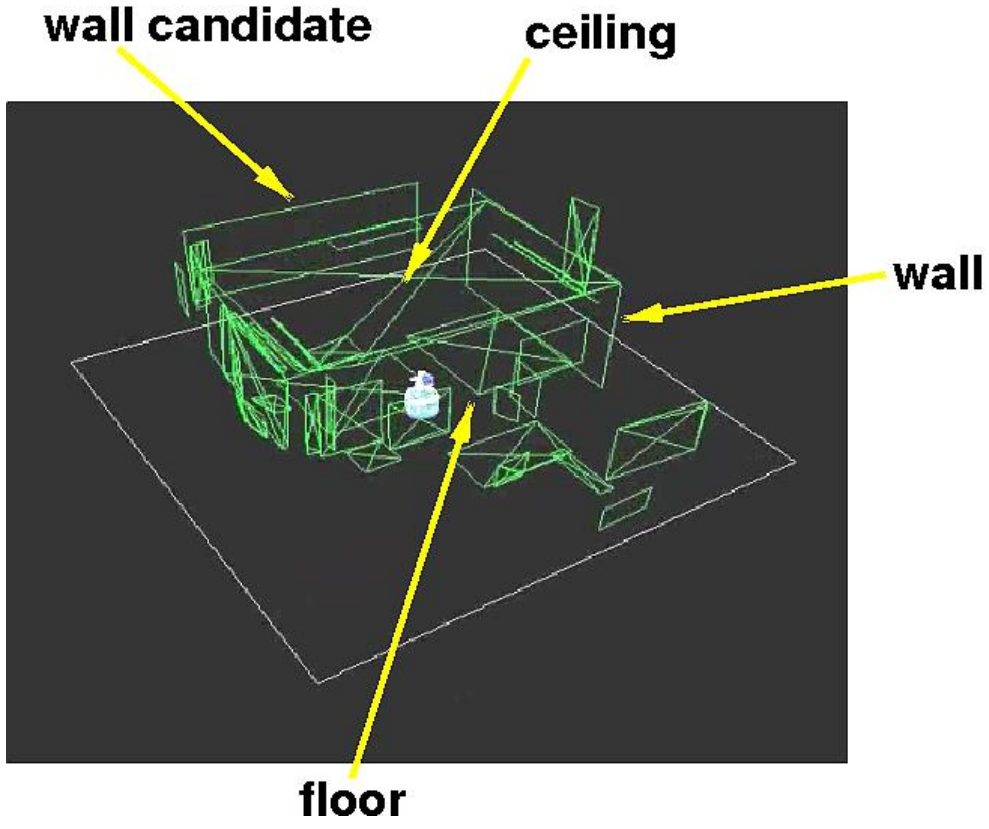
Extraction of Room Walls

Robot camera image during panorama 3D scan



Extracted planes in 3D visualization

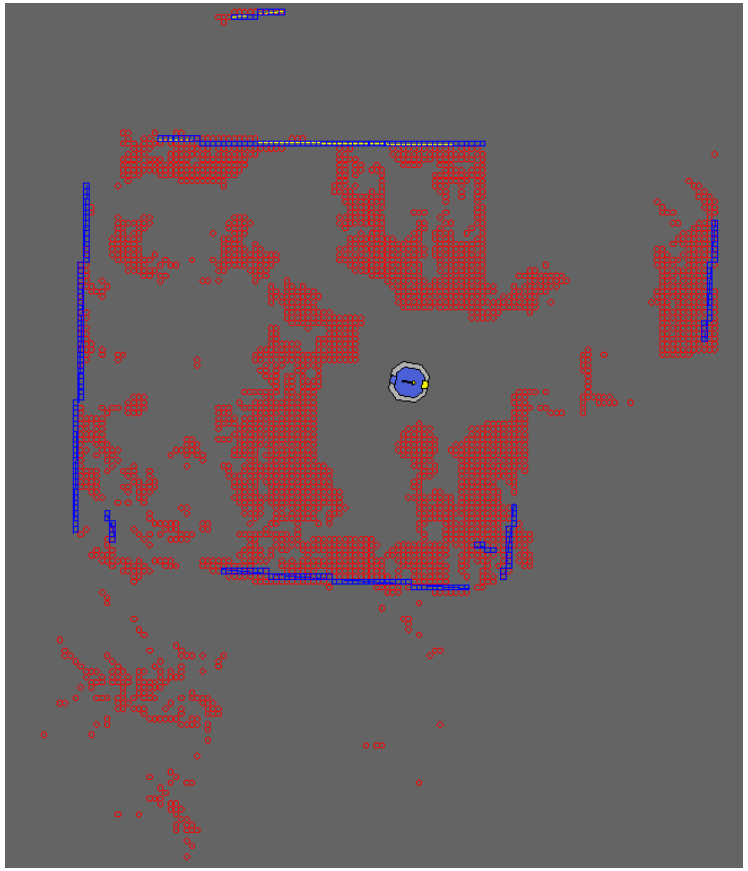
Extraction of Room Walls



Extracted features (example)

Extraction of Room Walls

- 2D projection of 3D scan
- Red cells: occupied due to projected 3D samples
- Blue cells: wall/wall candidates projected onto grids
- Occupied cells outside walls: bushes/trees in front of lab (seen through windows)



Extracted walls projected in 2D grid map

Grid Maps

Grid Maps

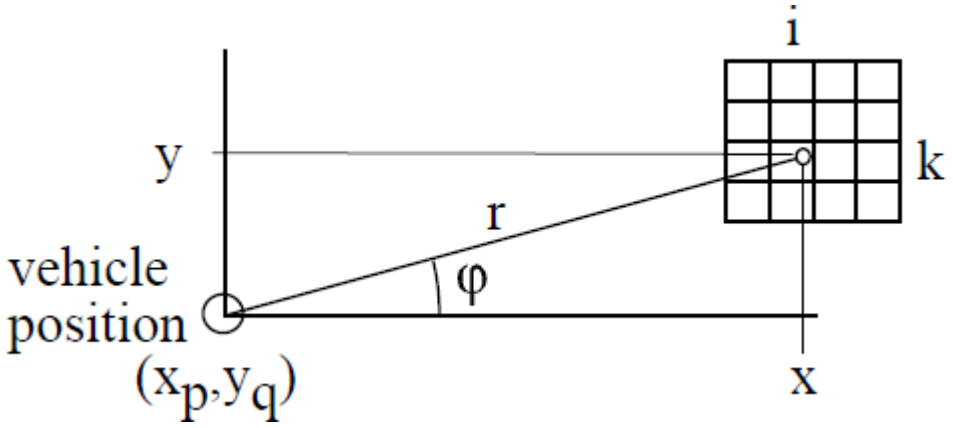
- Regular grid of cells with constant areas
- Precise robot pose required for consistent maps
- Lack of precise knowledge: probability value per cell
- Advantages: robustness, easy implementation
- Disadvantage: exact pose required, scalability

Occupancy Grid Maps

- Rastering with tiles of d cm width (adjusted to application)
- Pair of indices (i, k) for each tile

Building a Grid Map

- Coordinates of the radar point: $x = r \cdot \cos\varphi$, $y = r \cdot \sin\varphi$
- Coordinates of the vehicle: (x_p, y_q)
- Radar cone: a radar point occupies one tile and the cone denotes free space



Building a grid map from
laser radar measurements

Building Up a Grid Map

The corresponding tile with index i, k and center (x_i, y_k) is given by

$$x_i - \frac{d}{2} < x + x_p \leq x_i + \frac{d}{2}$$

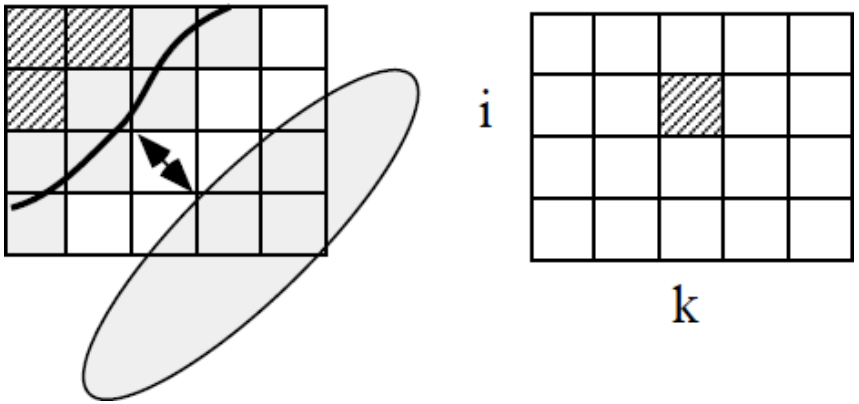
$$x_k - \frac{d}{2} < y + y_q \leq y_k + \frac{d}{2}$$

$$x_i = i \cdot d$$

$$y_k = k \cdot d$$

Marking of Tiles

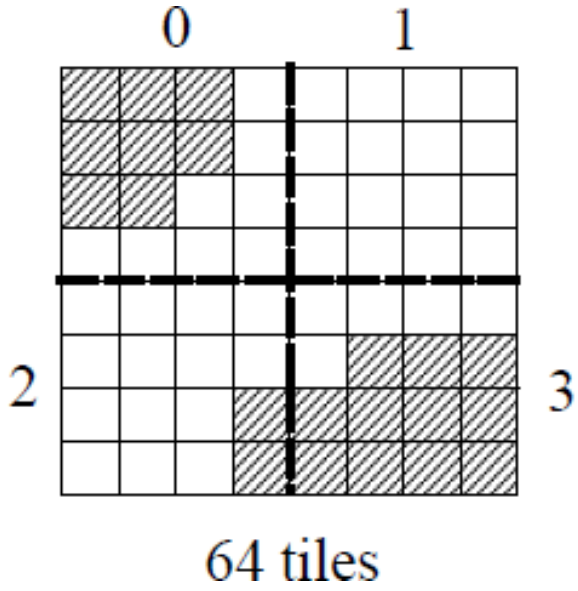
- Mark tiles
  - Free space (white)
  - Obstacle (dashed)
  - Partly free space (gray)
- Safety clearance of n of free space between the vehicle and the occupied region

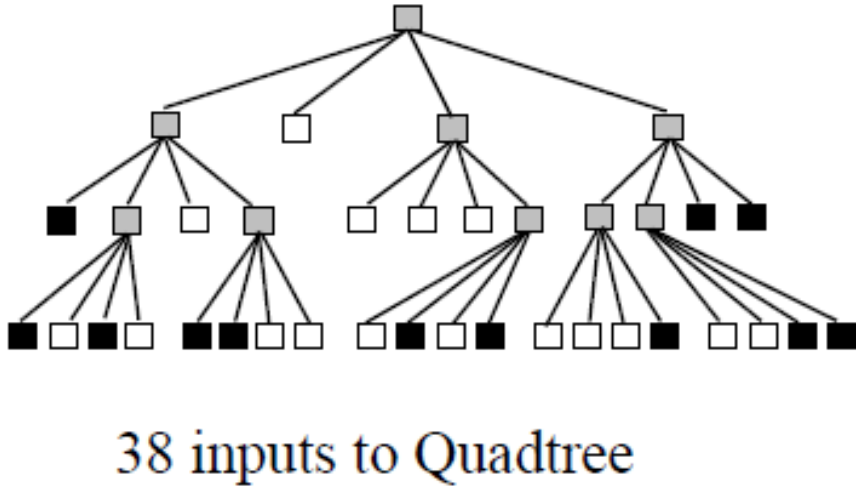


Grid map with marks for free space, obstacles, and mixed spaces

Quadtrees

- Compact description of grid maps
- Split environment recursively into blocks of $2^i \times 2^i$ tiles





Quadtree representation of a grid map with 64 tiles

Quadtrees

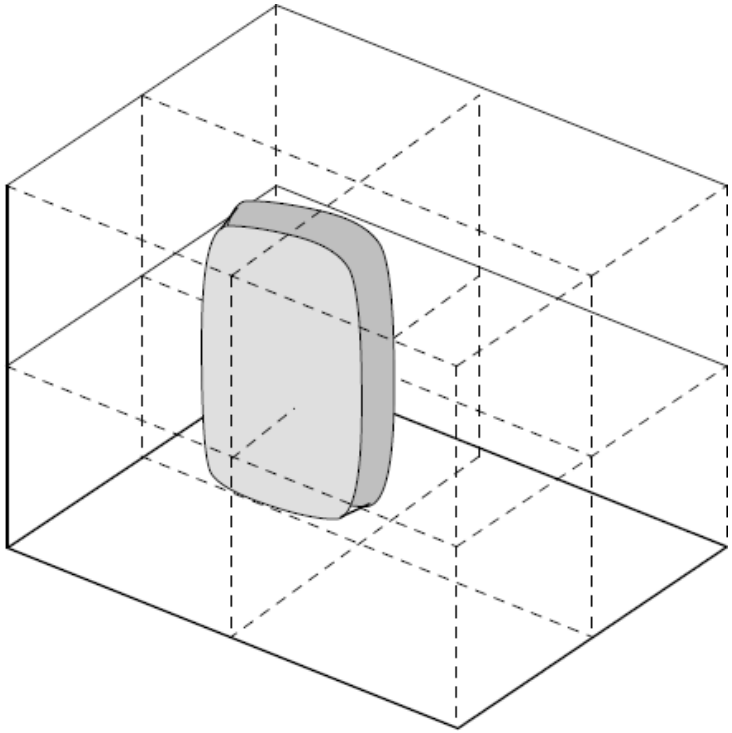
- Node in the tree represents a $2^i \times 2^i$ square part of the environment
- Representation of free space, obstacle, or mixture of both
- If a node is marked as obstacle or free it will not be split any further
- Mixed node: split recursively in 4 equal areas, represented as 4 new nodes of the tree
- Environment made up of $2^k \times 2^k$ tiles ⇒ quadtree representation in general has much less than 2^{2k} nodes

Quadtrees: Representation of a grid map

if $n = 0$ **then**
    it is a leaf node
else
    the map becomes the root node of a quad tree
end if
if the map is not uniformly colored **then**
    split the $2^n \times 2^n$ tiles into four maps of $2^{n-1} \times 2^{n-1}$ tiles each
    handle them the same way
else if a map is uniformly colored **then**
    it is a leaf node of the tree,
    denoting a block of $2^k \times 2^k$ tiles
end if

Octrees

- Extension to three dimensions
- Split a $3\mathrm{D}$-space of $2^n \times 2^n \times 2^n$ voxels recursively into blocks of $2^k \times 2^k \times 2^k$



An object in a $3\mathrm{D}$ grid

Probabilities

- Unreliable sensor measurements ⇒ include obstacle probability into specification of raster points
- Several independent measurements contribute to cell
- For each cell $C(i,j)$ the likelihood of occupancy can be computed as

$$C(i,j) = \frac{\text{hits}(i,j)}{\text{hits}(i,j) + \text{miss}(i,j)}$$

- $\text{hits}(i,j)$: number of scans indicating obstacle
- $\text{miss}(i,j)$: times the cell appeared to be empty
- Simple counter approach: less computational effort than e. g. Bayesian Filters

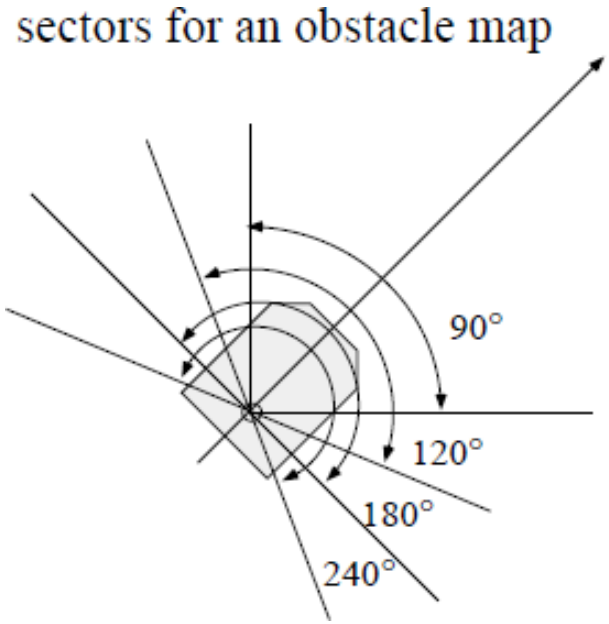
Sector Maps

Sector Maps

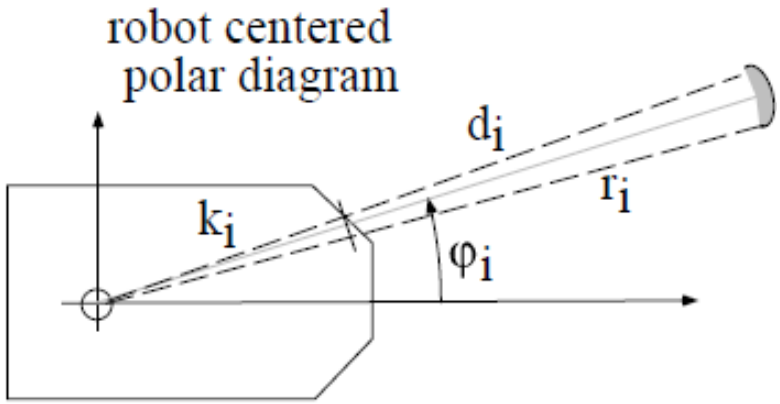
- Partition space in more flexible way than in grid maps
- Most common: uniform polar or rectangular sectors
  - Polar sector: angle and distance to the closest obstacle in the area the sector covers
  - Rectangular sector: x- and y-coordinates of the closest obstacle in the area the sector covers
- Contained information
  - Presence of obstacles
  - Slope of the ground
  - Terrain traversability
  - …

Polar Sector Maps

- Divide space into sectors of some degree, e. g. $\Delta\varphi = 5\,°$ from $-120\,°$ to $+120\,°$
- Distance between vehicle and obstacle: $d_i = r_i - k_i$

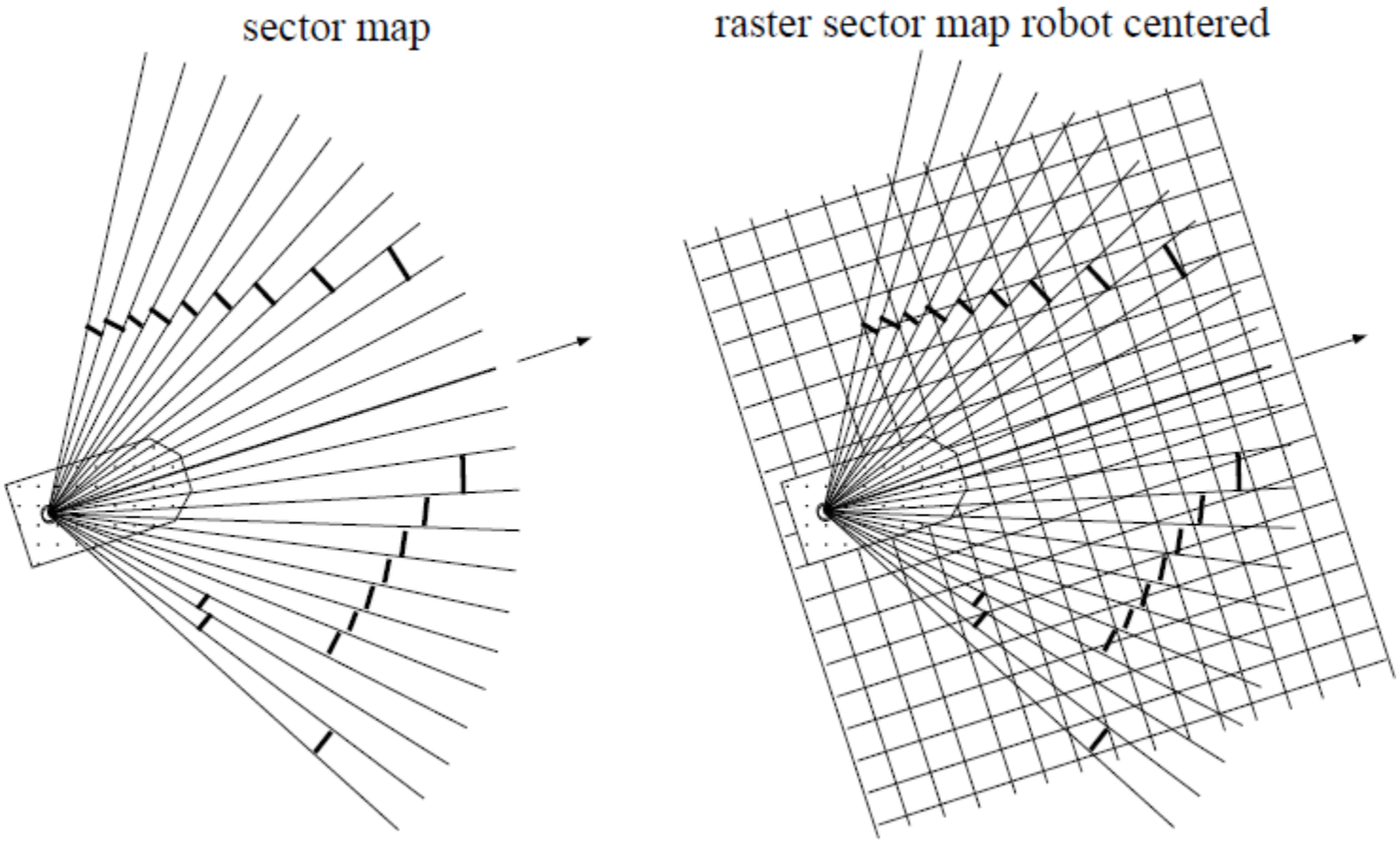


Typical angle range for sector maps



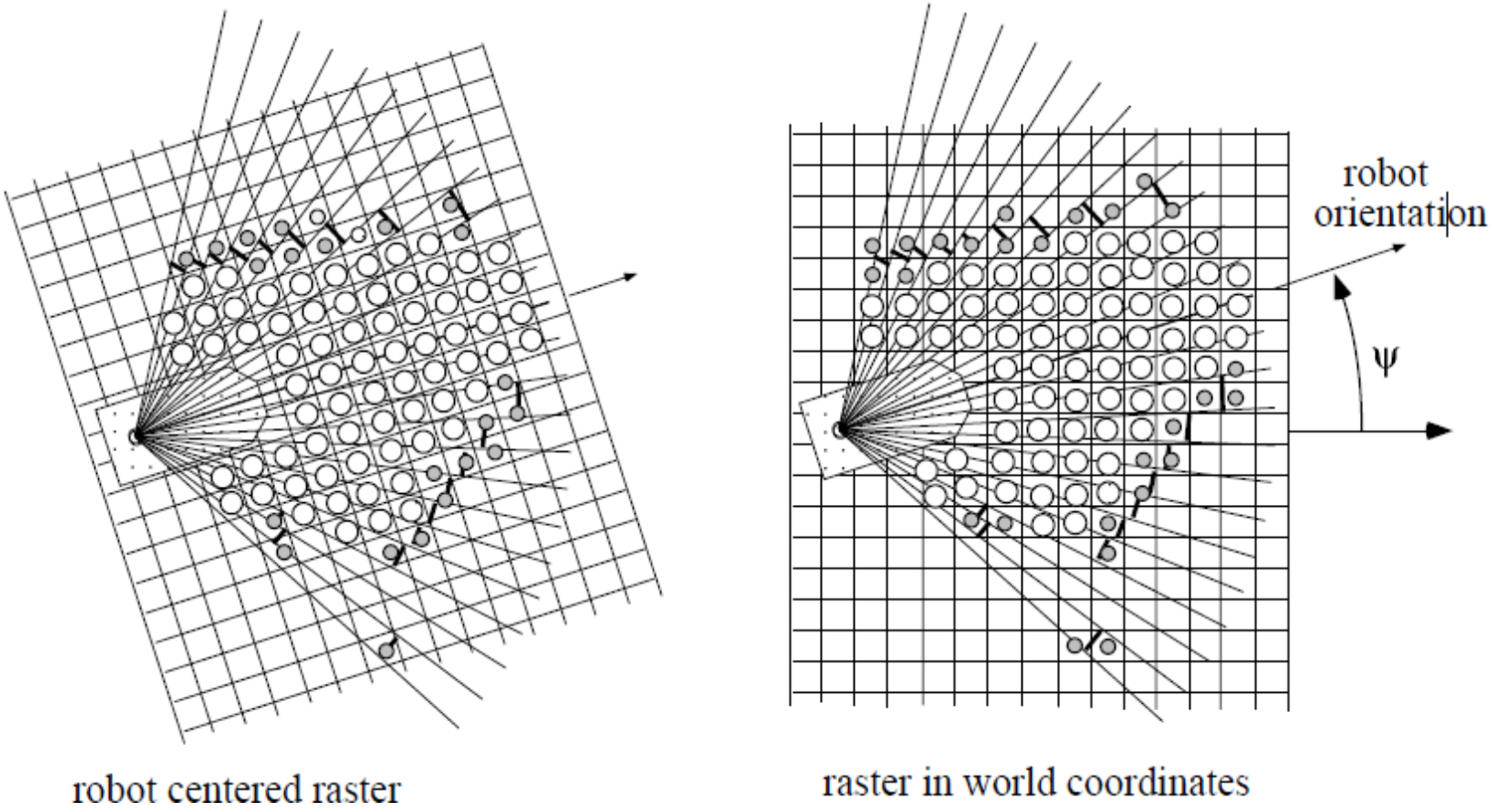
Obstacle representation inside a sector

Sector Map Example



Typical sector map and relation to a grid map

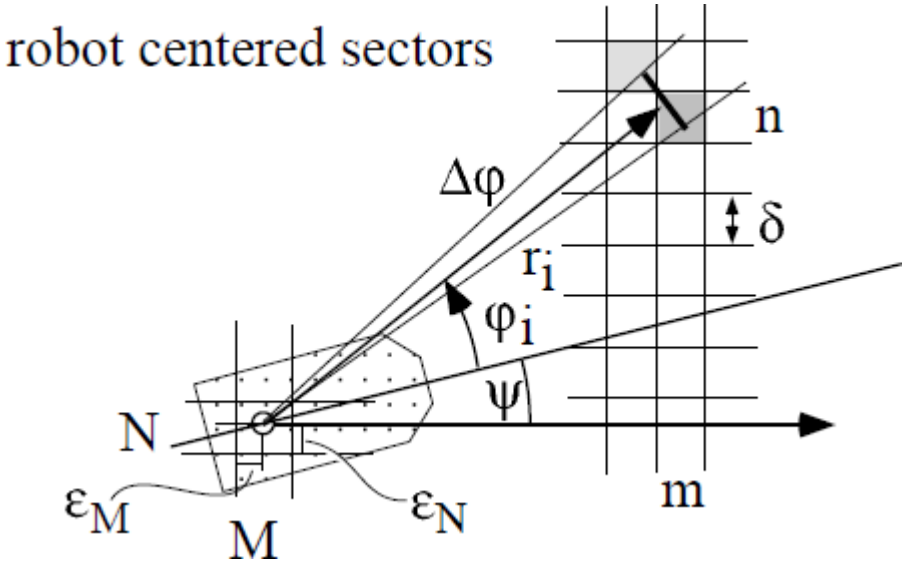
Transfer Sectors to Grid Map



Transformation of a sector map to a
robot-centred/world-centred grid map

Transfer Sectors to Grid Map

- Transfer robot-centered sector map to world-centered grid map
- $(x, y) = (M \cdot \delta + \varepsilon_M, N \cdot \delta + \varepsilon_N)$ and $\psi \neq 0$: position and orientation of the kinematic center in world coordinates
- (M, N): index of the tile in which the kinematic center is posed



Transformation of a robot-centered sector map to a world-centered grid map

Transfer Sectors to Grid Map

- ε_M and ε_N: offset of the kinematic center in x- and y-direction in respect to the origin of the tile
- δ: length of the tiles
- Obstacle at distance r_i in sector i with an opening angle $\Delta\varphi \Rightarrow$ all tiles with index (n, m), which fulfill one of the 3 pairs of inequations listed below have to be marked as obstacles and the space covered by the cone up r_i as free space

Transfer Sectors to Grid Map

$$(M+m)\cdot\delta-\varepsilon_M \le r_i\cdot\cos(\varphi_i+\psi) \le (M+m+1)\cdot\delta-\varepsilon_M$$

$$(N+n)\cdot\delta-\varepsilon_N \le r_i\cdot\sin(\varphi_i+\psi) \le (N+n+1)\cdot\delta-\varepsilon_N$$

$$(M+m)\cdot\delta-\varepsilon_M \le r_i\cdot\cos(\varphi_i+\psi+\Delta\varphi/2) \le (M+m+1)\cdot\delta-\varepsilon_M$$

$$(N+n)\cdot\delta-\varepsilon_N \le r_i\cdot\sin(\varphi_i+\psi+\Delta\varphi/2) \le (N+n+1)\cdot\delta-\varepsilon_N$$

$$(M+m)\cdot\delta-\varepsilon_M \le r_i\cdot\cos(\varphi_i+\psi-\Delta\varphi/2) \le (M+m+1)\cdot\delta-\varepsilon_M$$

$$(N+n)\cdot\delta-\varepsilon_N \le r_i\cdot\sin(\varphi_i+\psi-\Delta\varphi/2) \le (N+n+1)\cdot\delta-\varepsilon_N$$

Topological Maps

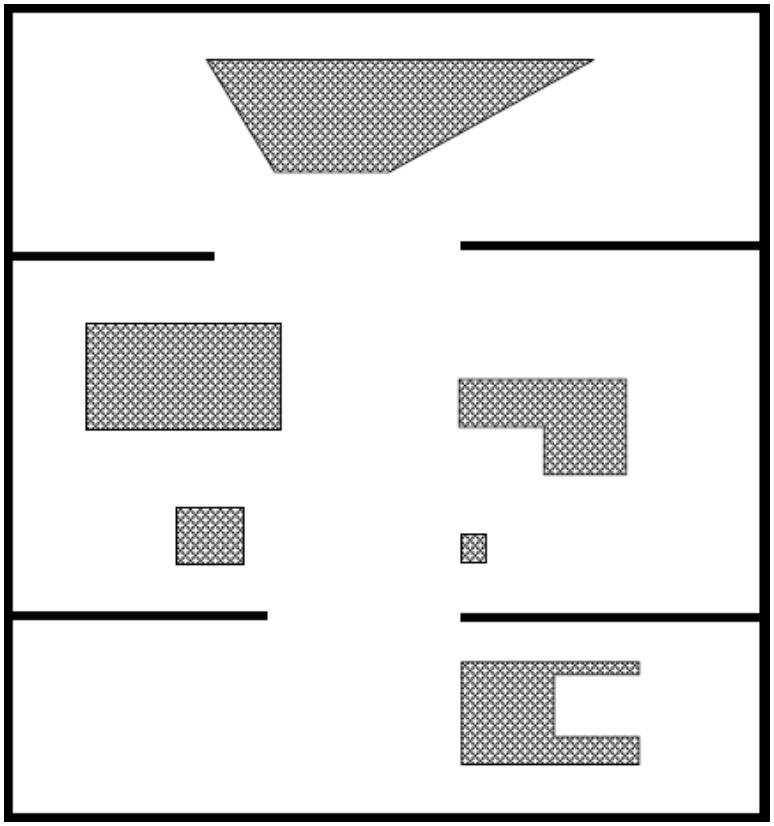
Topological Maps

- Abstract description of large-scale structures
- Describe navigation-relevant places as annotated coordinate tuples linked by paths
- Advantages: formal guarantee for correct map possible, less computational overhead
- Examples
  - Bus lines and bus stops in a town
  - Motor highway net of a country
  - A net with stations and railway lines for a subway or the railway system
  - A grid of the high voltage transmission lines of a country
  - A sewage system of a town

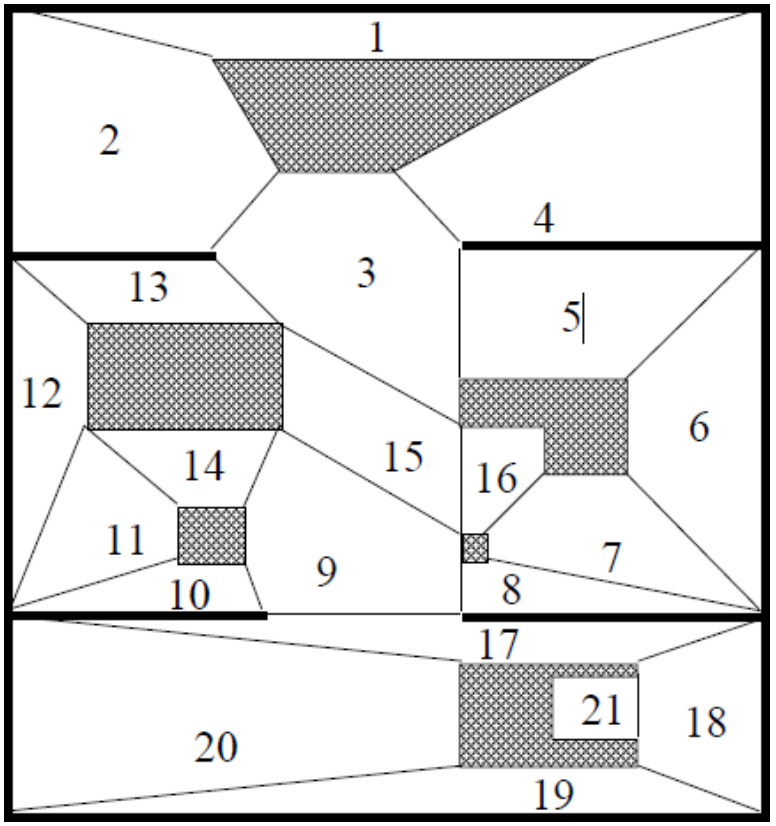
Topological Maps

- Typical questions to be answered with help of a topological map
  - Where to change buses between stations A and B?
  - Can I drive from A to B via C or via D too?
  - How many stops are there on the way from A to B?
  - If one transmission line fails, are there lines to circumvent the failed one?
  - Where is the entrance to a main sewage line?
- These questions are difficult to answer using geometric maps only ⇒ Transformation between geometric and topological maps required (visibility graph algorithms)

Topological Maps



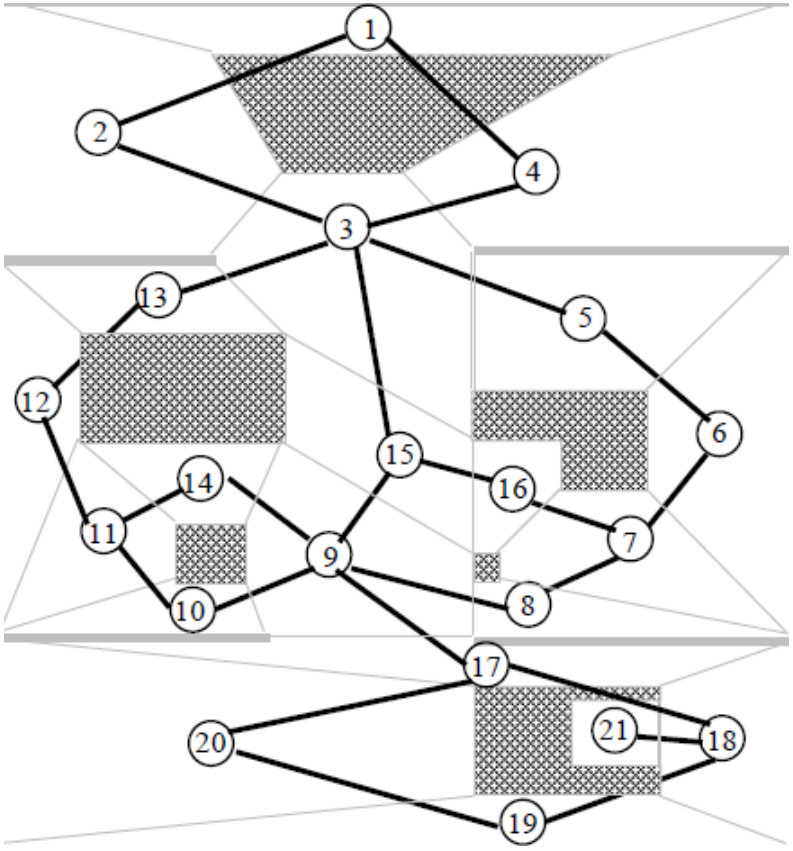
Geometric map

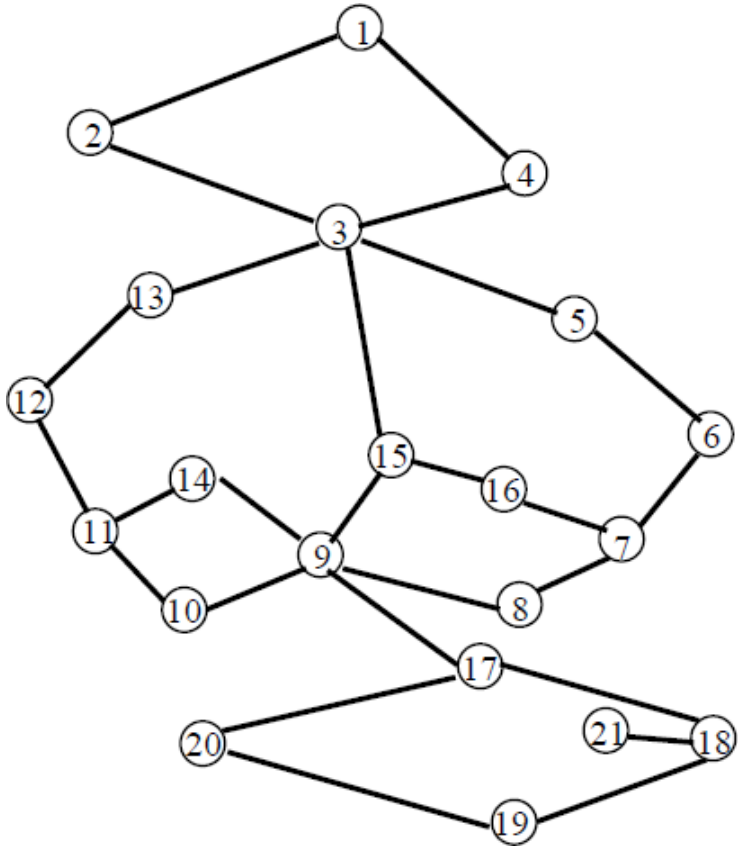


Connecting edges

Connecting corners by edges without crossing objects in a map of 3 rooms

Topological Maps

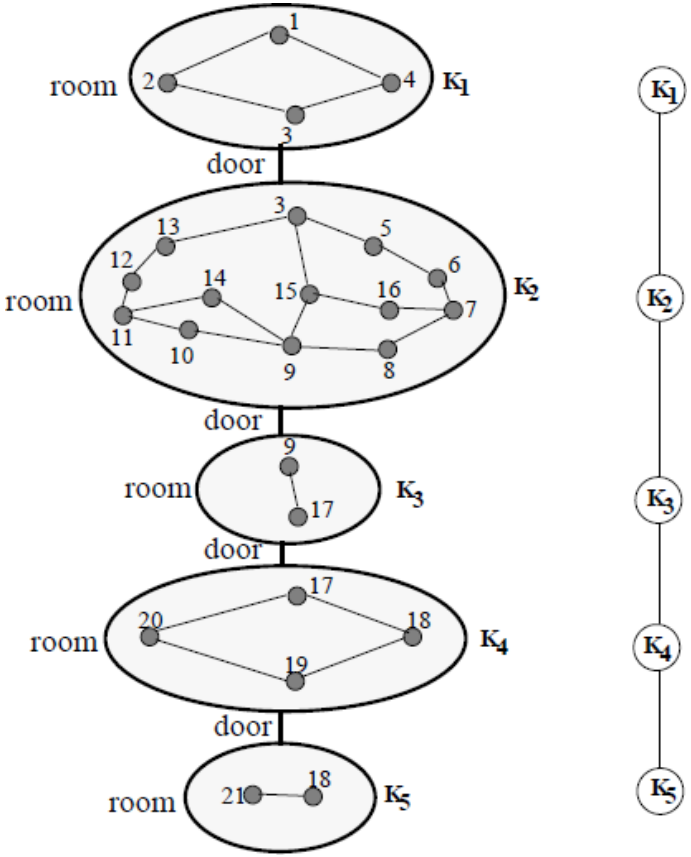




Cell graph

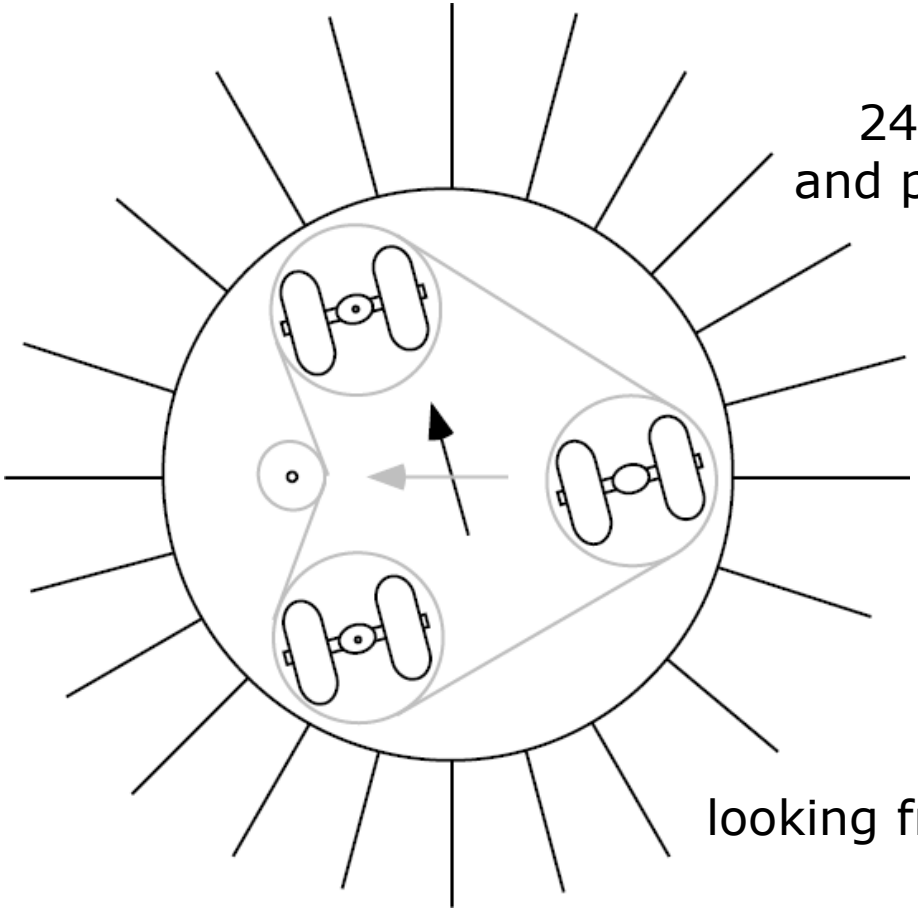
Interconnecting regions and the corresponding graph

Topological Maps



Decomposition tree: subgraphs for rooms, interconnections representing passage ways

Example: ALICE



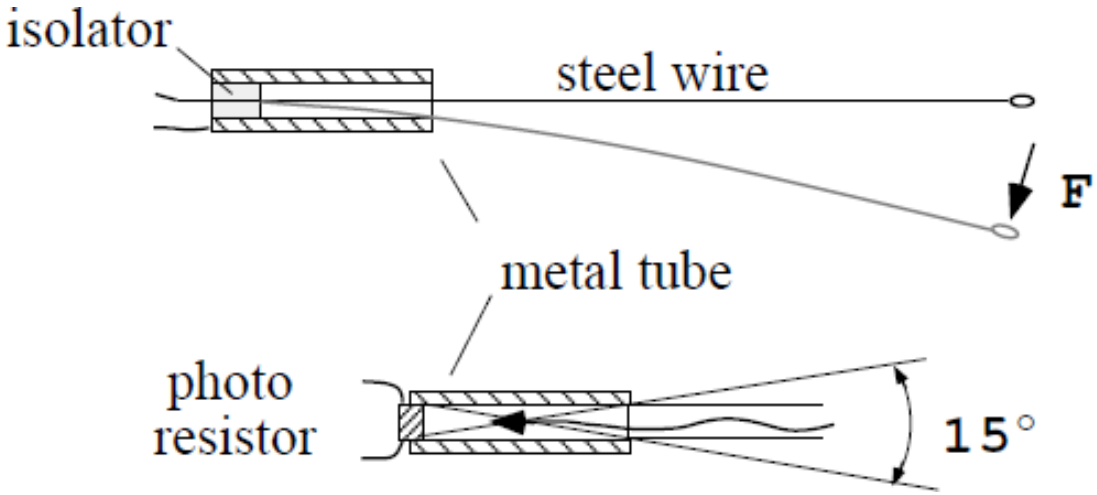
24 whiskers
and photo diodes

looking from downwards

Sketch of ALICE

ALICE: Sensors

- Touch sensor using simple electrical switch.
- Hitting an obstacle bends the wire and closes the contacts of the switch



The touch sensor and the photo sensor of ALICE

ALICE: Sensors

- The measurements of 24 photo sensor $(I_0, \ldots, I_{23})$ are normalized to $\tilde{I}_i = I_i / I_{max} \in [0,1]$
- The 24 touch sensor values $(T_1, \ldots, T_{24})$ are smoothed to cope with the rather large uncertainty of these sensors:

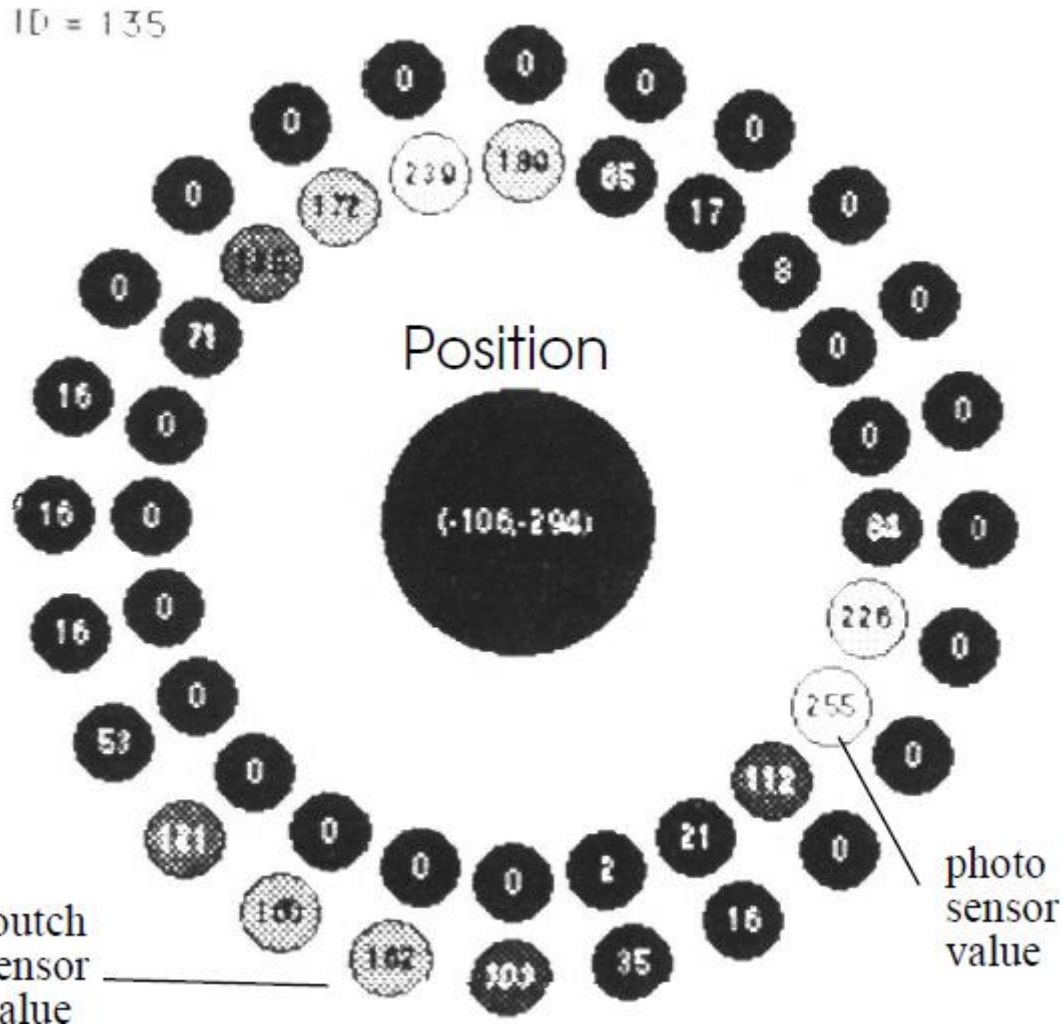
$$\tilde{T}_i = (2T_i + T_{i-1} + T_{i+1})/4, \quad i = i \bmod 24$$

- Only these 48 measurements describe the environment around ALICE
- For the calculation of the position and orientation of the vehicle the movement of the chain of the synchro drive is detected by light barriers and the wheel velocity measured by optical encoders

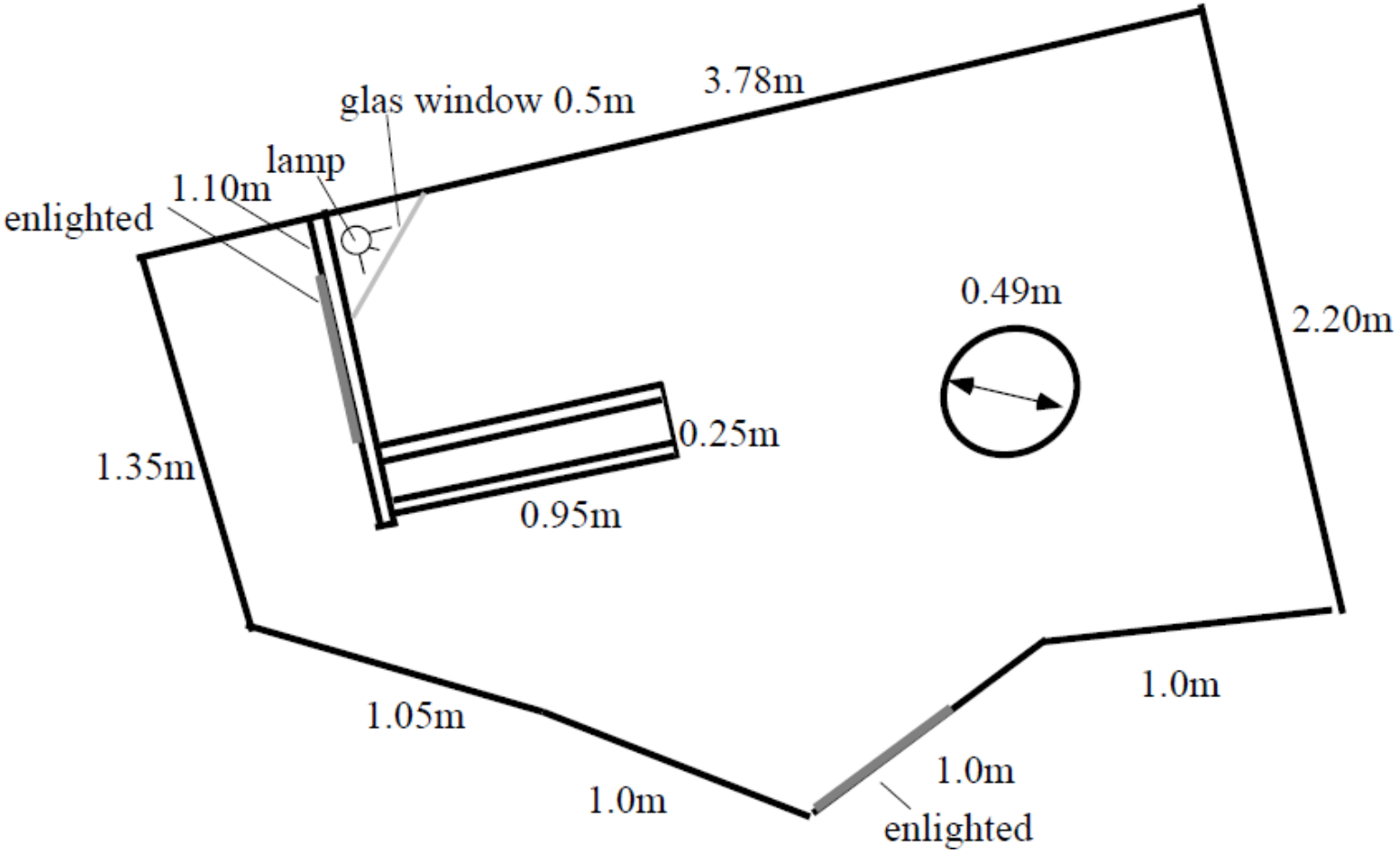
ALICE: Scene Vector

- The ALICE scene vector has 48 components used as input to a neural gas net
- Use position information to build a graph: from one best matching unit to the next one the vehicle actually has been driven, so there is a line drawn. The node is annotated with the position
- Visiting counter for lines used to cope with wrong measurements (delete seldomly traversed lines)
- Correction of drift in direction using light sensors

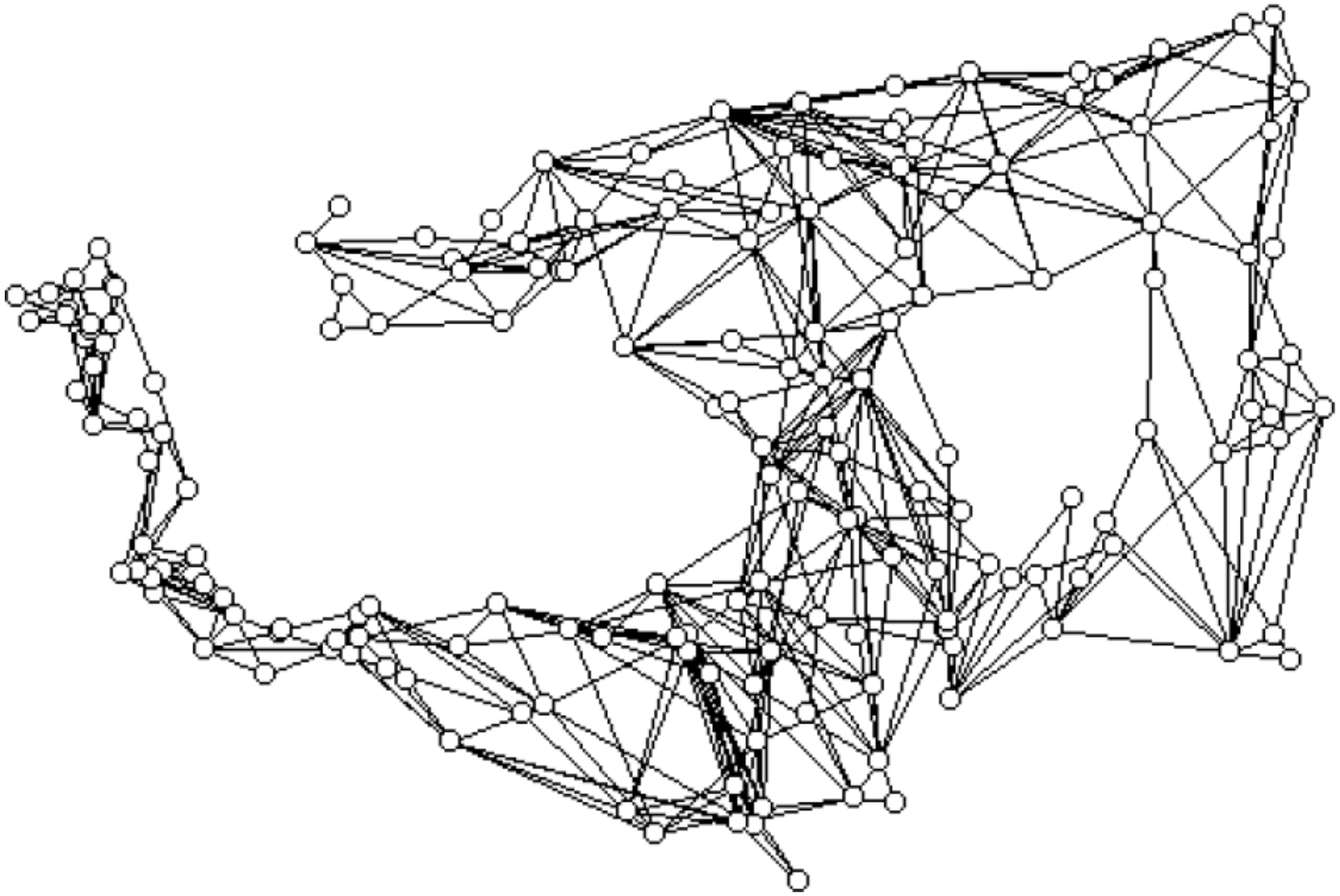
ALICE: Typical Sensor Situation



ALICE: Test environment



ALICE: Topological Graph



Hybrid Maps

Map Type Characteristics

	Topological maps	**Metrical maps**
Scale	Large-scale space	Small-scale space
Sensor inputs	Abstracts sensor inputs	Stores sensor inputs
Computational power	Low	High
Memory consumption	Low	High
Sensitive to noise	Less	More
Real-time mapping	Yes	Depends on computational power

⇒ Hybrid approaches try to combine both types:
high precision with compact representation

Classification of Hybrid Maps

- Abstraction-based hybrid maps
  - Metrical map as basis
  - Abstraction of map as topological representations
  - Efficient planning of approx. path to a given goal location
  - Metrical map for relocalization and obstacle avoidance
- Hierarchical hybrid maps
  - Several local metrical maps with limited scale tyed together in a global topological map
  - Divide-and-conquer approach: tackle scalability
  - Prevents errors from spreading over entire mapped area
  - Disadvantage: partially overlapping local maps cannot be used to enforce global consistency

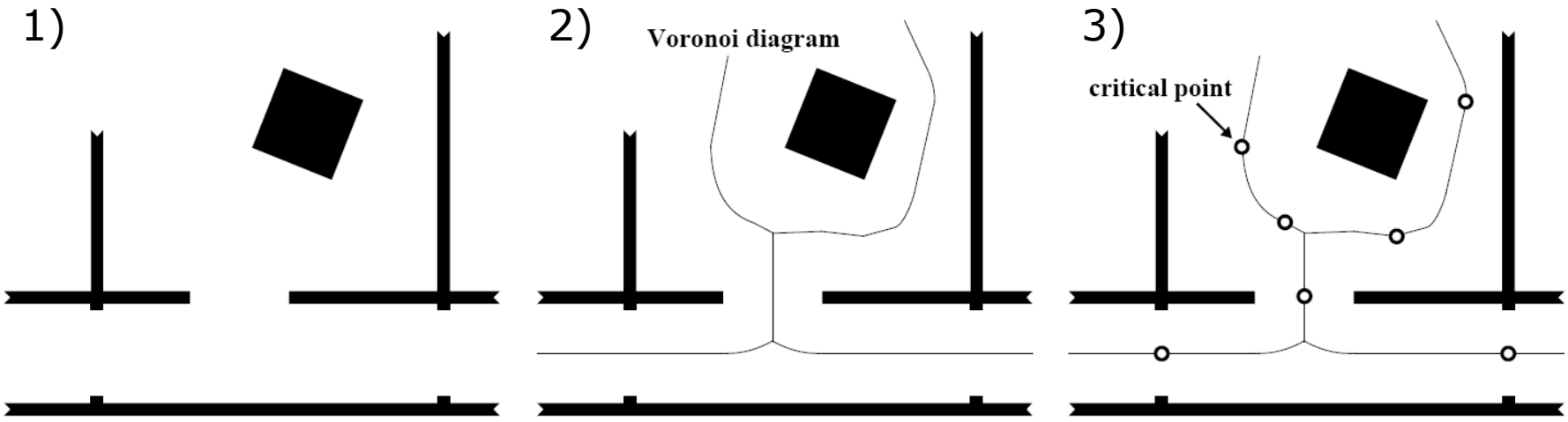
Example: Abstraction-based Hybrid Maps

- Voronoi-graph based topological map through abstraction of a long-lived, complete metrical map
- Topological map building initiated by thresholding an occupancy grid map
- Build Voronoi diagram by selecting all free grid cells whose two nearest occupied grid cells (the base points) are equidistant
- Critical points: base point distance is local minimum
- Lines between critical points: divide space into separate topological regions at narrow passages

Example: Abstraction-based Hybrid Maps

Steps in the topological abstraction scheme of [Thrun98a]:

1) Original thresholded occupancy map
2) Overlaid Voronoi diagram
3) Critical points

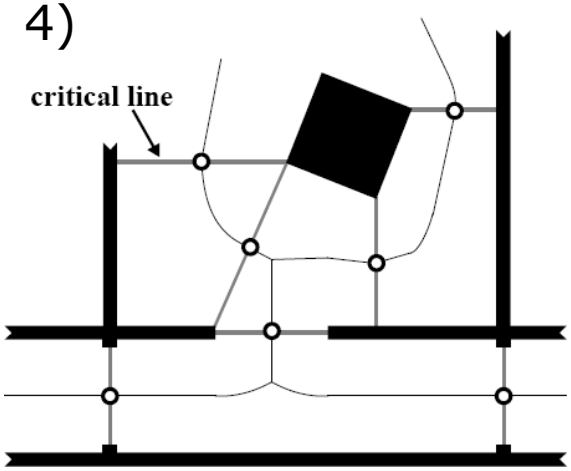


Example: Abstraction-based Hybrid Maps

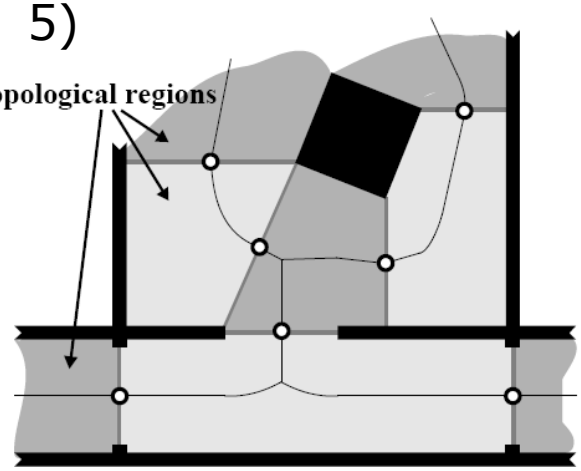
Steps in the topological abstraction scheme of [Thrun98a]:

4) Critical lines
5) Topological regions
6) Resulting topological graph

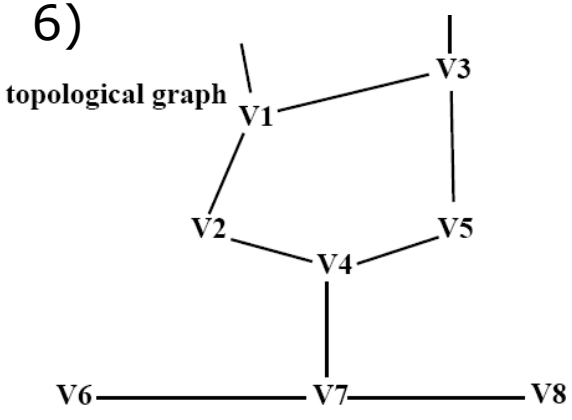
4)



5)



6)



Example: Abstraction-based Hybrid Maps

- Advantage: Fast path planning possible without resorting the detailed metrical mapping

- Disadvantage: Topological nodes only characterized by spatial position ⇒ Arrival at topological node can only be detected using localization techniques on detailed map

Coming Next

Simultaneous Localization and Mapping