Mapping

Alfredo Weitzenfeld

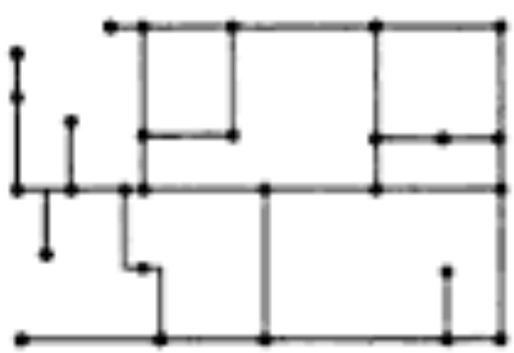
Mapping

- A **map** is a model of the environment used for robot localization and to compute paths. A map is impacted by robot pose representation.
- **Mapping** is the task of generating models of robot environments from sensor data.
- **Map precision** must match robot and application. The higher the map precision the higher the computational complexity.

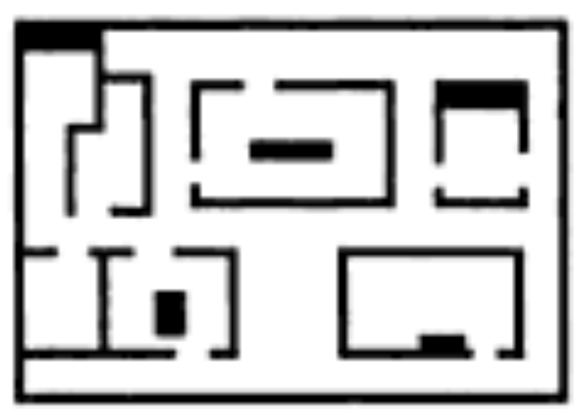
Mapping

Mapping

1. High level features (e.g. landmarks for topological maps, etc.): Low volume, filters out lot of the information
2. Low level features (e.g. lines, etc.): Medium volume, filters out some information
3. Raw sensor data: Large volume, uses all acquired information



1



2

3

Mapping

- **Odometric map**: distances between locations (no landmarks)
- **Landmark-based map**: distances and orientations in relation to external landmarks
- **Topological map**: similar to landmark-based map with nodes and edges representing particular locations (no odometry)
- **Metric map**: combines all previous types of maps with precise measurements between map locations and landmarks

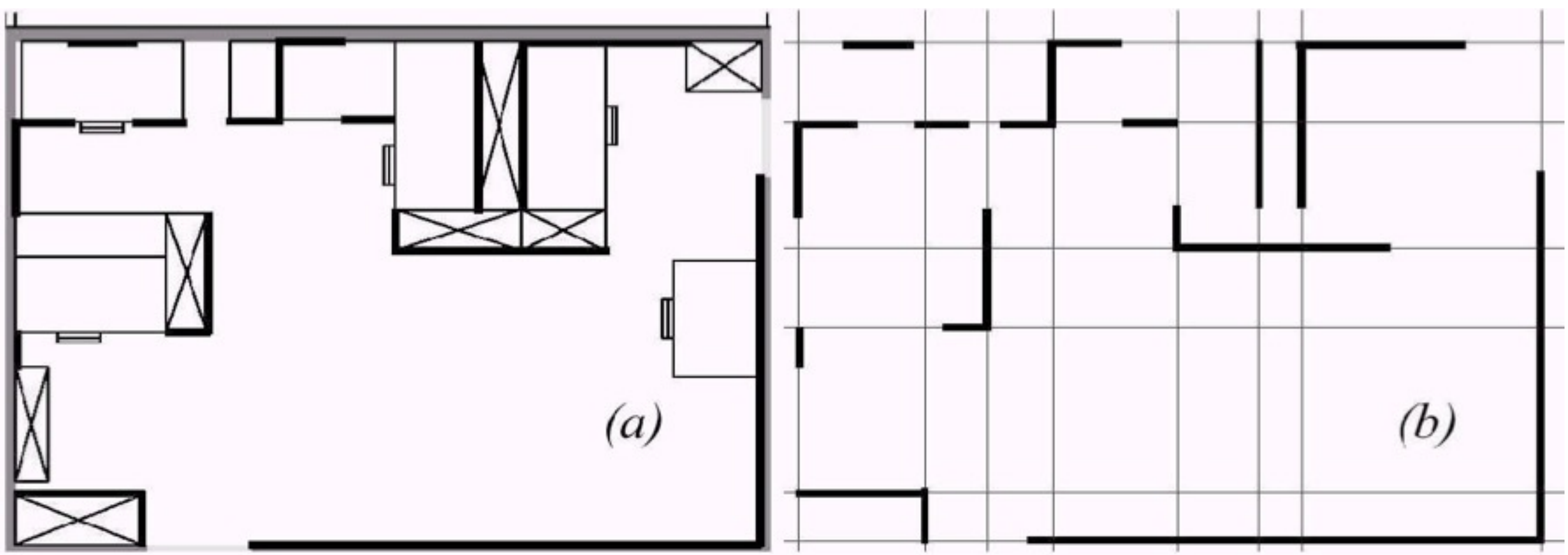
Representation

- **Representation** refers to how information is stored or encoded
- **Robot representation**
  - Represent the robot as a point (e.g. Bug Algorithms)
  - Assume robot is capable of omnidirectional motion
  - Robot in reality is of nonzero size
    - Dilation of obstacles by robot radius
    - Resulting objects are approximations
    - Leads to problems with obstacle avoidance
- **World representation**
  - Continuous
  - Discrete

Continuous Representation

a) High accuracy but can be computationally expensive

b) Map represented as series of infinite lines, e.g. using a laser ranger finder

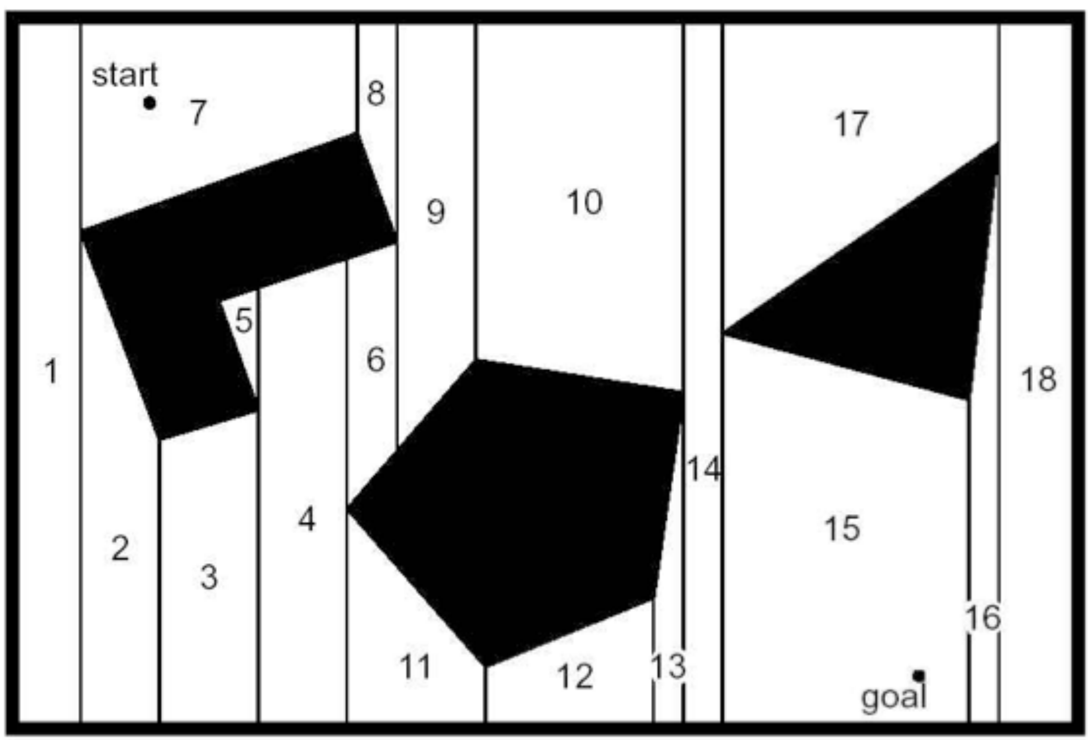


Discrete Representation

- Capture only useful features of world
- Lower accuracy but computationally less expensive
- Computationally better for reasoning, particularly if map is hierarchical
- Discrete Cell Decomposition
  - Exact Cell Decomposition
  - Fixed Cell Decomposition
  - Adaptive Cell Decomposition
- Occupancy Grid
- Topological Maps

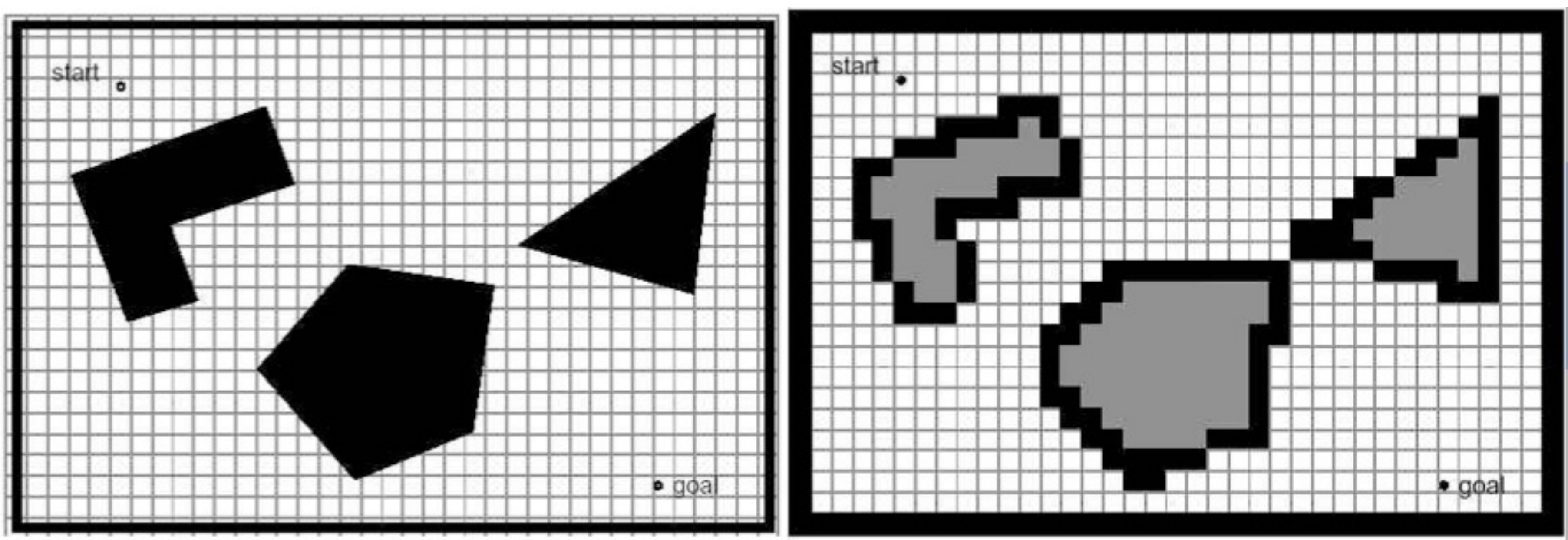
Exact Cell Decomposition

- Free space is represented by the "exact" union of simple trapezoidal regions or cells, while obstacles are represented by polygons.
- Regions or cells can be extremely compact



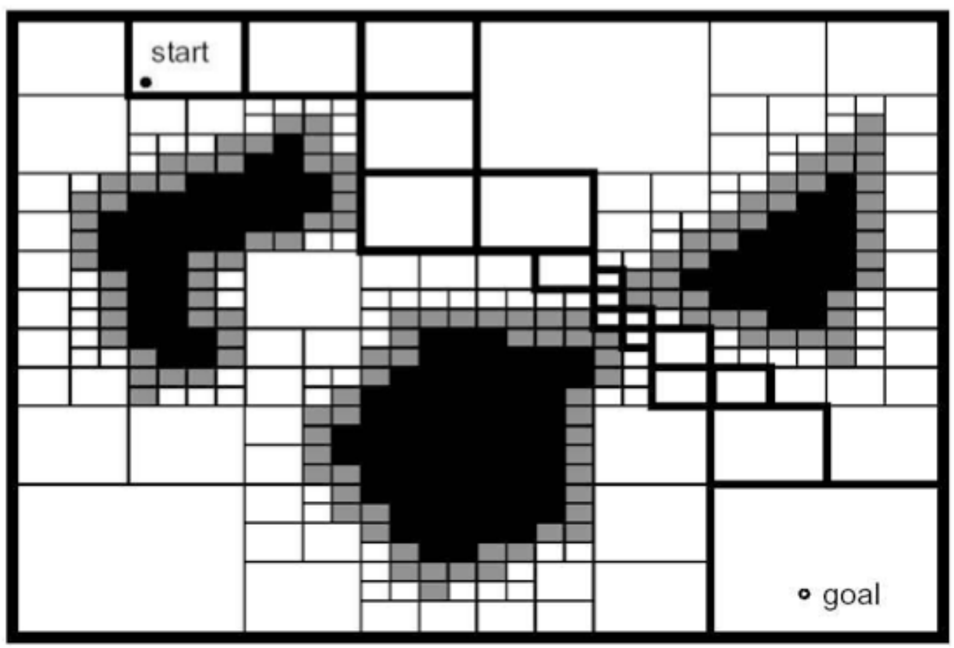
Fixed Cell Decomposition

- Free space is decomposed into cells of a fixed size
- Each cell is either empty or full (there may be loss of information such as loss of the narrow passageway)



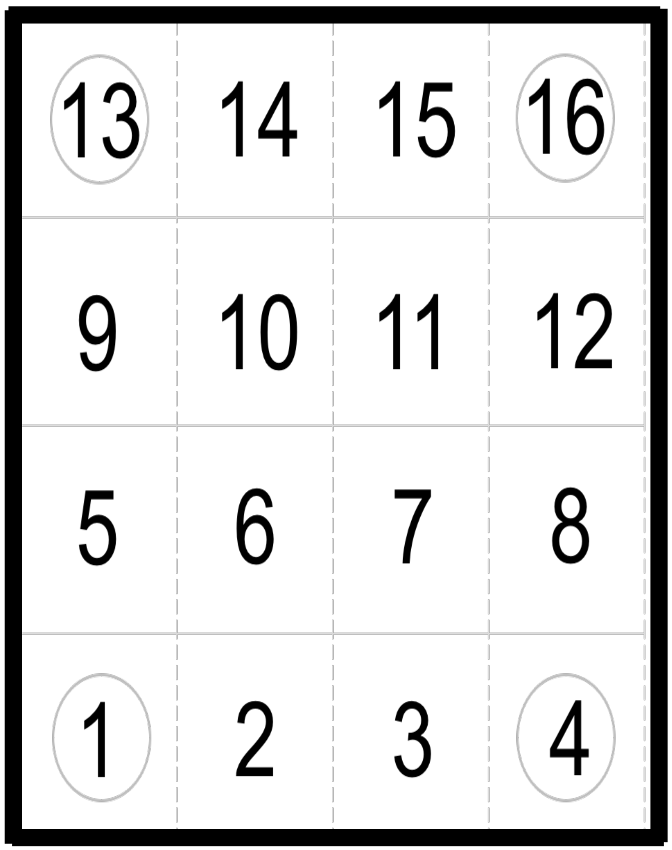
Adaptive Cell Decomposition

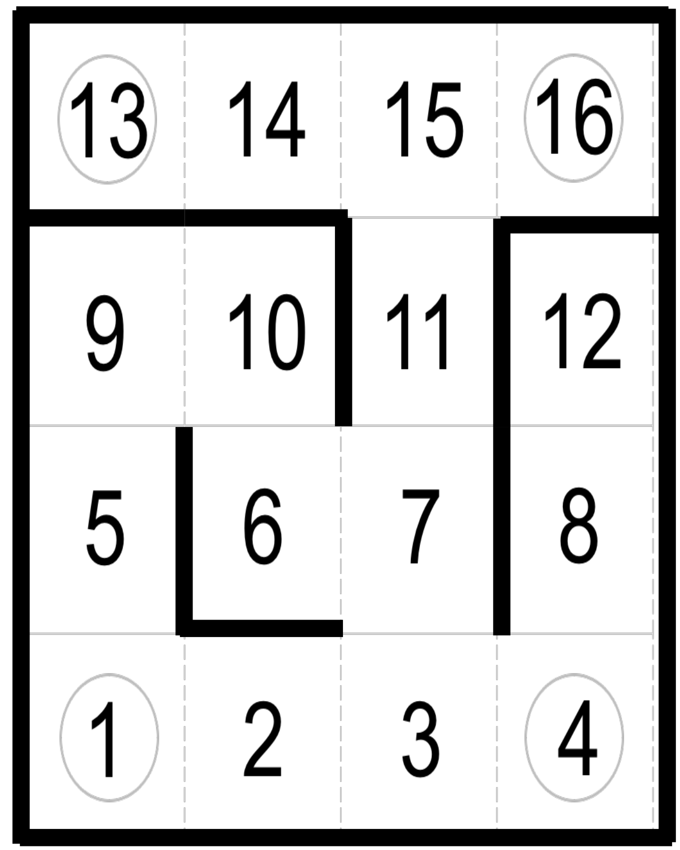
- Multiple types of adaptation: quadtree or other
- Recursively decompose free space until a cell is completely empty or full (there may be loss of information as with fixed cell decomposition)
- Space efficient if compared to fixed cell approach



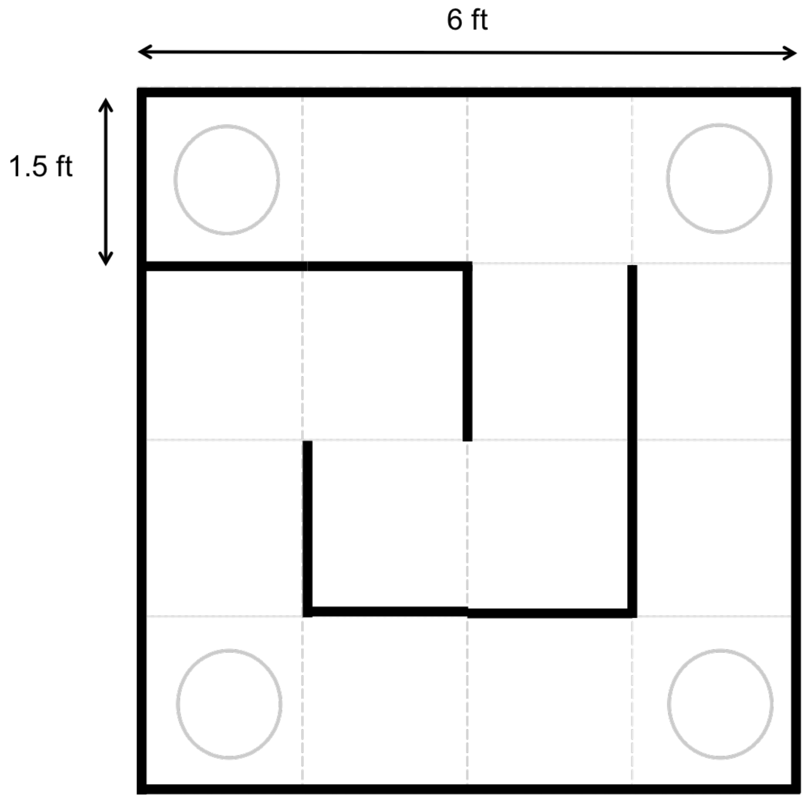
Fixed Cell Decomposition Example

- 16 fixed size cells
- No obstacles but walls separating cells





Fixed Cell Decomposition Example

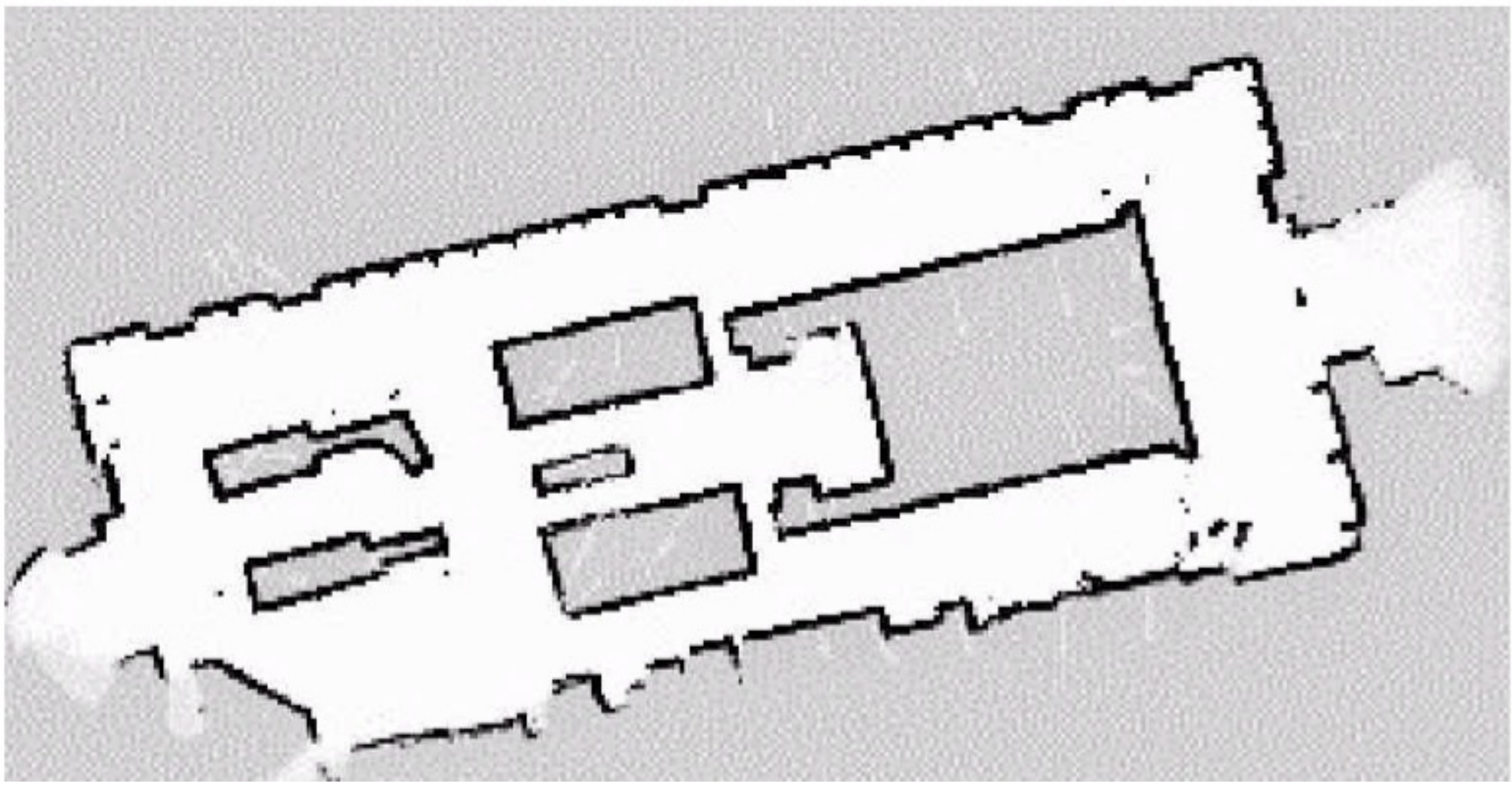


W	W	W	W	W	W	W	W	W
W	1		2		3		4	W
W	W	W	W	W		W		W
W	5		6	W	7	W	8	W
W		W		W		W		W
W	9	W	10		11	W	12	W
W		W	W	W	W	W		W
W	13		14		15		16	W
W	W	W	W	W	W	W	W	W

- *W*: Walls and Wall Corners
- *1-16*: Grid cell locations where robot may be found in the maze
- Empty cells are either possible locations of wall or robot

Occupancy Grid Maps

- Darkness of cell proportional to cell counter value

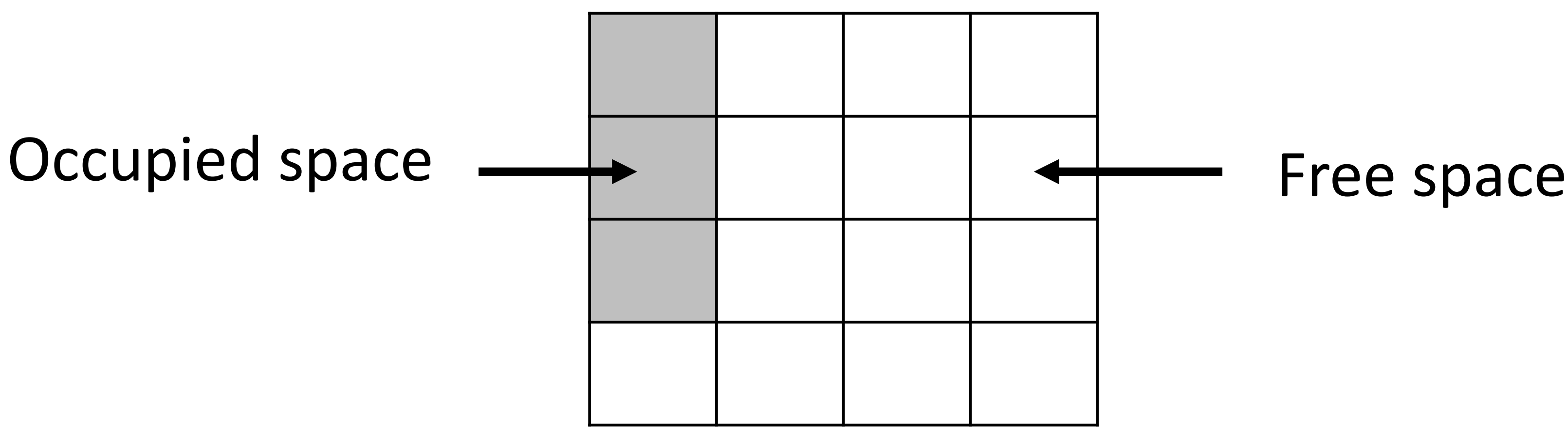


Occupancy Grid Maps

- Each cell indicates probability of being free or occupied:
  - Requires known robot pose
  - Grid cell probability distribution
  - Each variable is binary or a probability, corresponding to the degree of occupancy of the location it covers
- Particularly useful with range-based sensors
  - If sensor strikes something in a cell, higher probability
  - If sensor goes over cell and strikes something else, lower probability (presuming it is free space)
- Disadvantages
  - Map size is a function of size of environment and size of cell

Occupancy Grid Maps

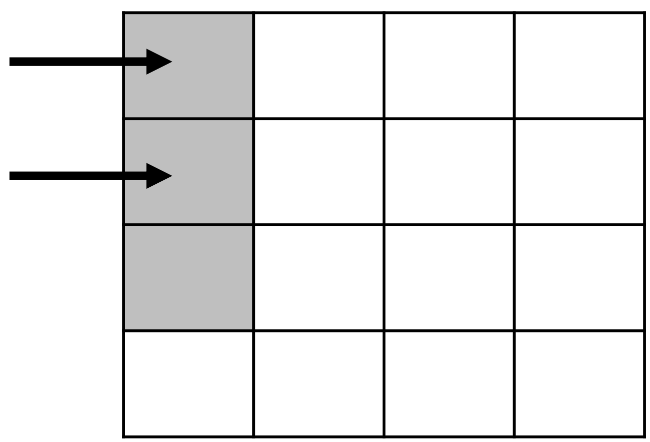
- The area that corresponds to a cell is either completely free or occupied.



Occupancy Grid Maps

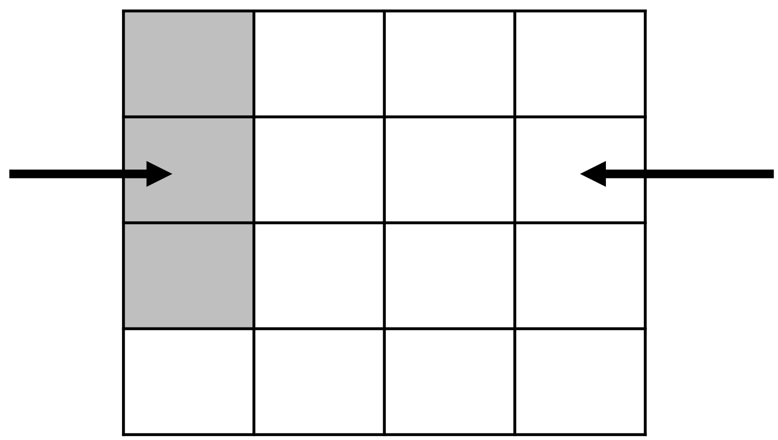
- The cells (the random variables) are independent from each other.

No dependency between the cells



Occupancy Grid Maps

- The probability of cell occupancy, $p(m_i)$, is computed from a binary variable m_i with value 0 or 1:

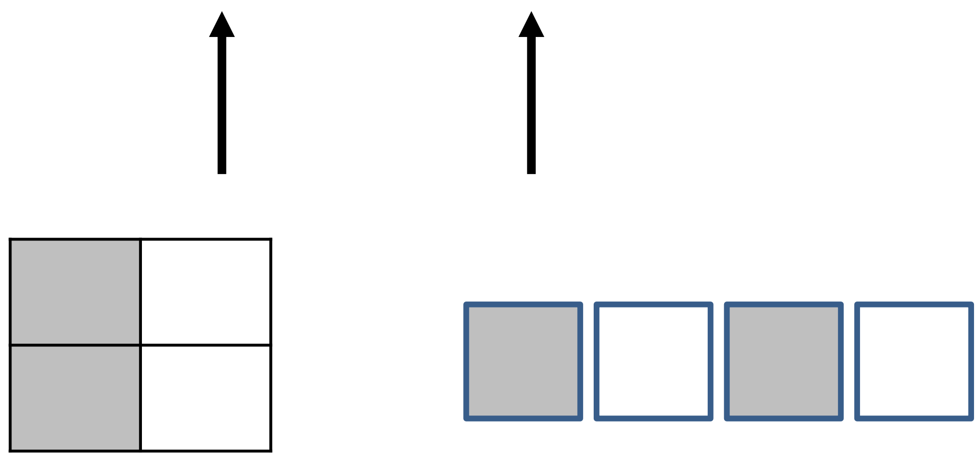


- p_{occupied}: Cell is occupied, $p(m_i = 1) > 0.5$
- p_{empty} or p_{free} or $p_{\text{unoccupied}}$: Cell is not occupied, $p(m_i = 0) < 0.5$
- p_{unknown}: No knowledge, $p(m_i) = 0.5$

Occupancy Grid Maps

- The probability distribution of the map is given by the product over the maps.

$$p(m) = \prod_i p(m_i)$$



Example map
(4-dimension vector)

4 individual cells

Bayes Filter with Static States

- **State Belief**: If we assume a static environment, i.e. only static objects <u>not affected by robot control</u>, the belief is a function only of the measurement:

  $$\mathrm{bel}_t(s) = \mathrm{p}(s \mid z_{1:t}, u_{1:t}) = \mathrm{p}(s \mid z_{1:t})$$

Binary Bayes Filter with Static States

- **Binary State Belief**: The belief is defined as a binary state, i.e. occupied or not occupied:

$$\mathrm{bel}_t(\neg s) = 1 - \mathrm{bel}_t(s)$$

Occupancy Grid Maps

- Given sensor data $z_{1:t}$ and the poses $s_{1:t}$ of the sensor, the occupancy grid map m is estimated by:

$$p(m|z_{1:t}, s_{1:t})$$

- The controls $u_{1:t}$ play no role in the occupancy grid map, since the path is already known.

Occupancy Grid Maps
Estimating a Map from Data

- Given sensor data $z_{1:t}$ and poses $s_{1:t}$ of the sensor, estimate the map.

$$p(m|z_{1:t}, s_{1:t}) = \prod_i p(m_i|z_{1:t}, s_{1:t})$$

Binary Bayes filter for a static state, i.e. map cell state is either "Occupied" or "Empty"

Occupancy Grid Maps

- The occupancy grid algorithm breaks down the problem of estimating the map into a collection of separate problems of estimating each grid cell m_i:

$$p(m_i|z_{1:t}, s_{1:t})$$

- Each estimation is a binary problem with static states, and the complete map is approximated as the products of individual grid cells:

$$p(m|z_{1:t}, s_{1:t}) = \prod_i p(m_i|z_{1:t}, s_{1:t})$$

- This equation is equivalent to:

$$\underset{\substack{\uparrow \\ \text{map}}}{p(m)} = \prod_i \underset{\substack{\uparrow \\ \text{cell}}}{p(m_i)}$$

Occupancy Grid Maps

- Let m_i denote the grid cell with index i. The occupancy grid map partitions the space into finitely many grid cells:

$$m = \{m_i\}$$

- Each m_i has attached to it a binary occupancy value, either free ("0") or occupied ("1").
- The probability of the cell being occupied is given by

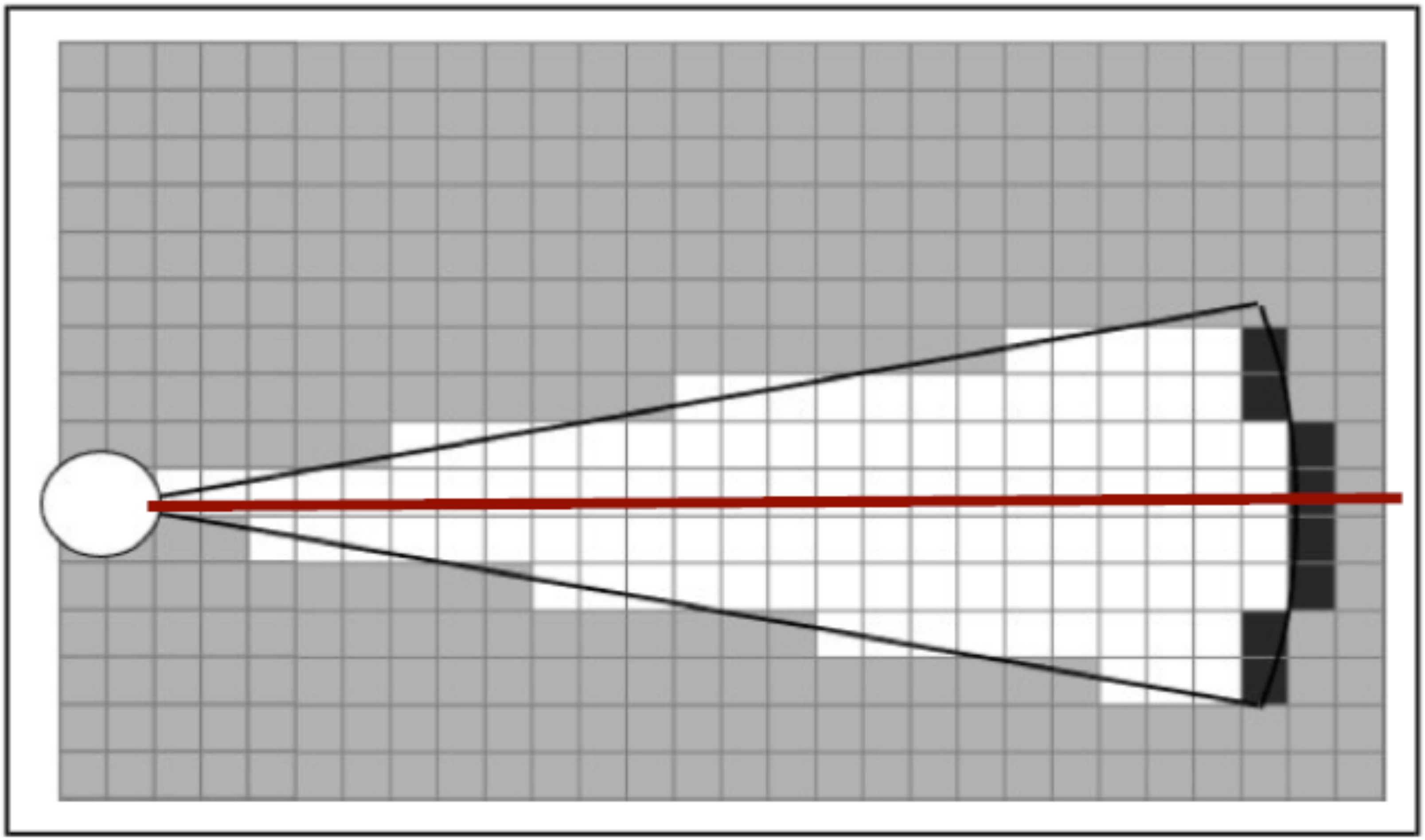
$$p(m_i = 1) = p(m_i)$$

- The probability of the cell being free is given by

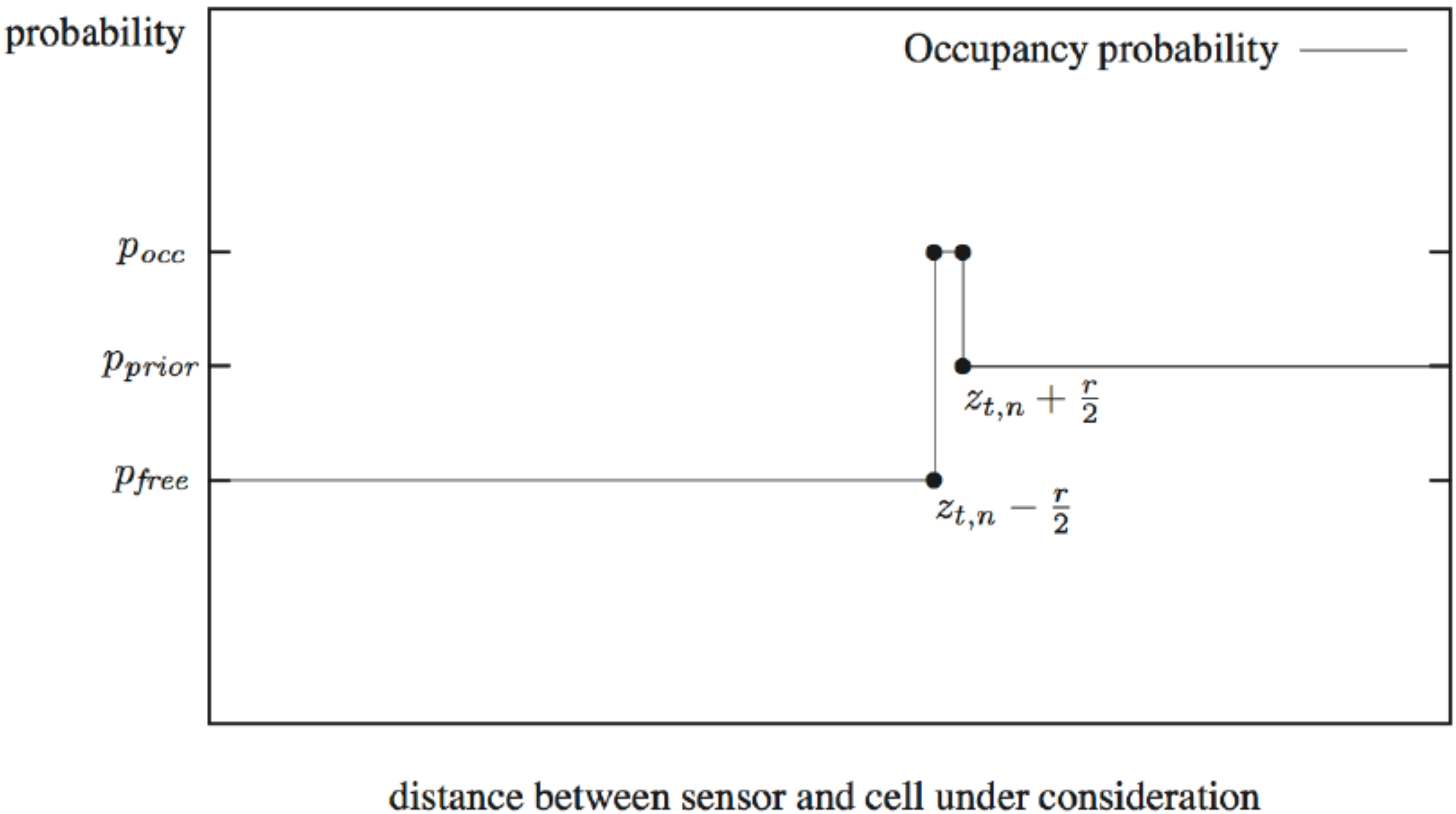
$$p(m_i = 0) = 1 - p(m_i)$$

Occupancy Grid Maps

- Consider the cells along the optical axis (red lines)



Occupancy Grid Maps



Logs Odd Notation

- **Odd Ratio** defines the ratio of the probability of an event divided by the probability of its negate:

$$\frac{p(s)}{p(\neg s)} = \frac{p(s)}{1 - p(s)}$$

- **Logs Odd** notation is defined as ("log" is the natural logarithm):

$$l(s) = \log\frac{p(s)}{1-p(s)}$$

- Retrieving $p(s)$:

$$\text{bel}_t(s) = p(s) = 1 - \frac{1}{1+\exp l(s)}$$

Logs Odd Notation

- **Logs Odd Ratio** defines the ratio of the probability of an event divided by the probability of its negate:

$$\frac{p(m_i|z_{1:t}, s_{1:t})}{p(\neg m_i|z_{1:t}, s_{1:t})} = \frac{p(m_i|z_{1:t}, s_{1:t})}{1 - p(m_i|z_{1:t}, s_{1:t})}$$

- **Logs Odd** notation is defined as:

$$l(m_i|z_{1:t}, s_{1:t}) = \log\frac{p(m_i|z_{1:t}, s_{1:t})}{1-p(m_i|z_{1:t}, s_{1:t})}$$

- Retrieving $p(m_i|z_{1:t}, s_{1:t})$:

$$p(m_i|z_{1:t}, s_{1:t}) = 1 - \frac{1}{1+\exp l(m_i|z_{1:t}, s_{1:t})}$$

Logs Odd Notation

- Computing the ratio of both probabilities (occupied and empty), by combining Bayes Rule with Markov Assumption:

$$\frac{p(m_i|z_{1:t},s_{1:t})}{1-p(m_i|z_{1:t},s_{1:t})} = \underbrace{\frac{p(m_i|z_t,s_t)}{1-p(m_i|z_t,s_t)}}_{\text{latest probability computation}} \underbrace{\frac{p(m_i|z_{1:t-1},s_{1:t-1})}{1-p(m_i|z_{1:t-1},s_{1:t-1})}}_{\text{recursive term}} \underbrace{\frac{1-p(m_i)}{p(m_i)}}_{\text{prior}}$$

- The prior defines the initial belief before processing any sensor measurements.

Logs Odd Notation

- Apply the logs odd ratio, and the product turns into a sum:

$$l(m_i|z_{1:t}, s_{1:t}) = l(m_i|z_t, s_t) + l(m_i|z_{1:t-1}, s_{1:t-1}) - l(m_i)$$

inverse sensor model　　　recursive term　　　prior

- The equation is rewritten to:

$$l_{t,i} = \text{inverse\_sensor\_model}(m_i, s_t, z_t) + l_{t-1,i} - l_0$$

$$\text{inverse\_sensor\_model}(m_i, s_t, z_t) = \log \frac{p(m_i|z_t,s_t)}{1-p(m_i|z_t,s_t)}$$

Logs Odd Notation

- The prior defines the logs odd of the initial belief before processing any sensor measurements.

- l_0 is the prior or initial logs odd,before processing any sensor measurements:

$$l_0 = \log \frac{p(m_i=1)}{p(m_i=0)} = \log \frac{p(m_i)}{1-p(m_i)}$$

- if p_{prior}= 0.5, $l_0 = \log \frac{0.5}{0.5} = \log 1 = 0$

Occupancy Grid Maps

occupancy_grid_mapping($\{l_{t-1,i}\}$, s_t, z_t):

1. For all cells m_i do
2.   if m_i in perceptual field of z_t then
3. $l_{t,i}$ = $l_{t-1,i}$ + inverse_sensor_model(m_i, s_t, z_t) - l_0
4.   else
5. $l_{t,i}$ = $l_{t-1,i}$
6.   endif
7. endfor
8. return $\{l_{t,i}\}$

Notes

- Line 3 uses additions, no multiplications
- The computation is based on the inverse sensor model, p(s_t l z_t), instead of the forward model p(z_t l s_t). The inverse sensor model specifies a distribution over the (binary) state variable as a function of the measurement z_t.

Inverse Sensor Model

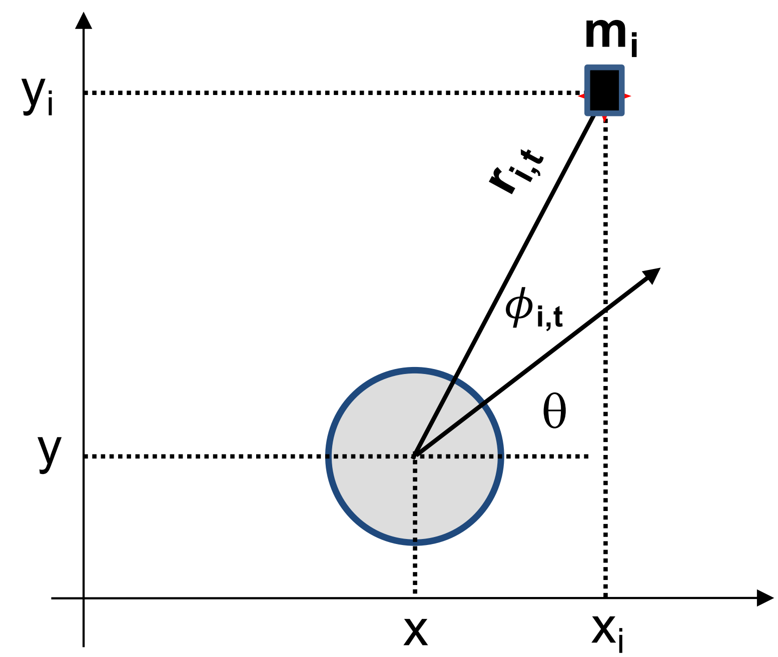
Assume:

- s_t= $[x, y, \theta]^T$ is the robot pose at time t
- m_i = (x_i, y_i) is the location of the cell (can also be applied to landmarks)
- θ is the robot orientation
- $\phi_{i,t}$ is the relative orientation of grid cell m_i
- $r_{i,t}$ is the relative distance of grid cell m_i
- $z_{i,t}$ is the measurement from grid cell m_i

$$z_{i,t} = (r_{i,t}, \phi_{i,t})$$

where

$$r_{i,t} = \sqrt{(x_i - x)^2 + (y_i - y)^2}$$
$$\phi_{i,t} = atan2\big((y_i - y), (x_i - x)\big) - \theta$$



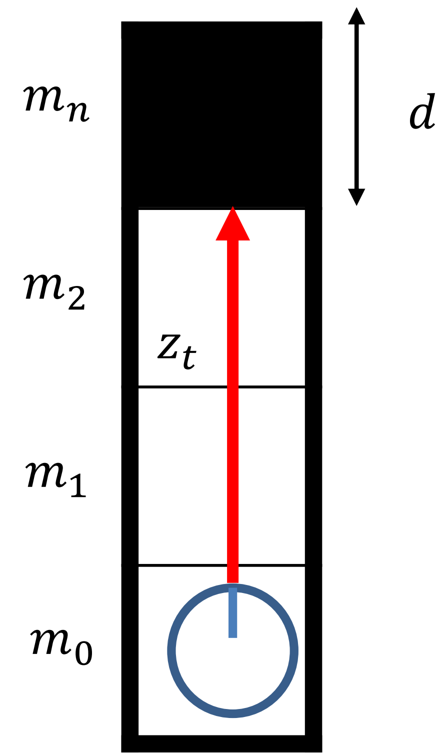
- The forward sensor model $p\left(z_{i,t} \,\middle|\, m_i, s_t\right)$ computes the measurements from the state s_t.
- The inverse sensor model $p(m_i | z_{i,t}, s_t)$ specifies a distribution over the (binary) state variable as a function of the measurement $z_{i,t}$

Occupancy Mapping Example

Build an occupancy grid map (cells $m_0, \ldots, m_n$) of a simple one-dimensional environment using a sequence of measurements from a range sensor.

Assume a very simple sensor model:

- Every grid cell with a distance (based on its coordinate) smaller than the measured distance is assumed to be occupied with p = 0.3.
- Every cell behind the measured distance is occupied with p = 0.6.
- Every cell located more than d=20cm behind the measured distance should not be updated.
- Robot starting position m_0 heading north.



Occupancy Mapping Example

- Using a log-odds equations:

$$l(m_i|z_{1:t}, s_{1:t}) = l(m_i|z_t, s_t) + l(m_i|z_{1:t-1}, s_{1:t-1}) - l(m_i)$$

$$p(m_i) = 0.5 \Rightarrow l(m_i) = \log \frac{p(m_i)}{1-p(m_i)} = 0$$

- Let $p(m_i|z_t, s_t)$ be the inverse sensor model:

$$p(m_i|z_t, s_t) = \begin{cases} 0.3 & \text{if position}(m_i) \leq z_t \\ 0.6 & \text{if position}(m_i) > z_t \wedge \text{position}(m_i) \leq z_t + d \\ 0.5 \text{ (unused)} & \text{if position}(m_i) > z_t + d \end{cases}$$

- Let $l(m_i|z_t, s_t)$ be the log odds inverse sensor model:

$$l(m_i|z_t, s_t) = \log \frac{p(m_i|z_t, s_t)}{1 - p(m_i|z_t, s_t)}$$

Occupancy Mapping Example

The log-odds ratio should be applied to this function to obtain $p(m_i|z_t, s_t)$. Note that unused in this context means we should not update the corresponding m_i cells.

Doing an update with $p(m_i|z_t, s_t) = 0.5$ would be equivalent, since $l(0.5) = 0$, but computationally more expensive.

The solution will involve applying for each measurement and for each cell the log-odds update formula. Once done we convert from log-odds to probability and display the output.

Note the inverse transformation provides the solution p of the log-odds definition:

$$l = \ln \frac{p}{1-p} \Rightarrow \exp l = \frac{p}{1-p}$$

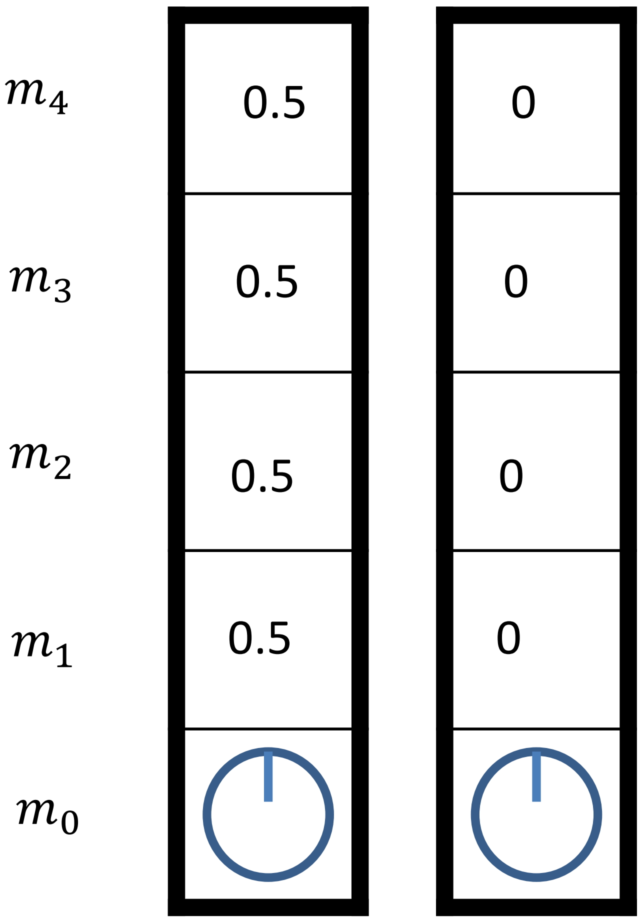
$$\Rightarrow (1-p) \exp l = p \Rightarrow \exp l - p \exp l = p \Rightarrow \exp l = p(1 + \exp l)$$

$$\Rightarrow p = \frac{\exp l}{1+\exp l} \Rightarrow \frac{\exp l + 1 - 1}{1+\exp l} \Rightarrow 1 - \frac{1}{1+\exp l}$$

Occupancy Mapping Example

- Prior (initial) values at $s_{t=0}$

$p(m_i|s_{t=0})$ l_0

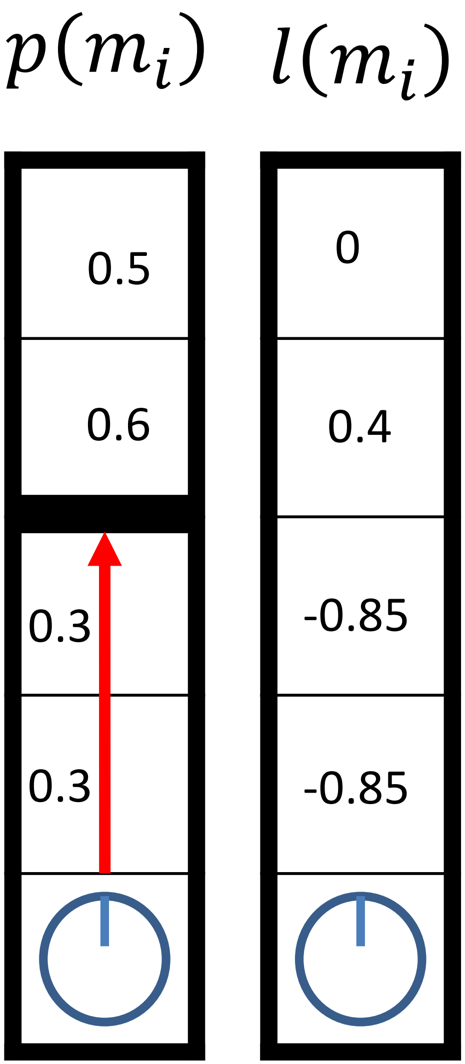


$p(m_i|s_{t=0})$ = 0.5

$l_0 = \log \frac{0.5}{1-0.5} = \log \frac{0.5}{0.5} = 0$

Occupancy Mapping Example

- Distance measurements at $z_{t=1}$ = 20cm (in red):



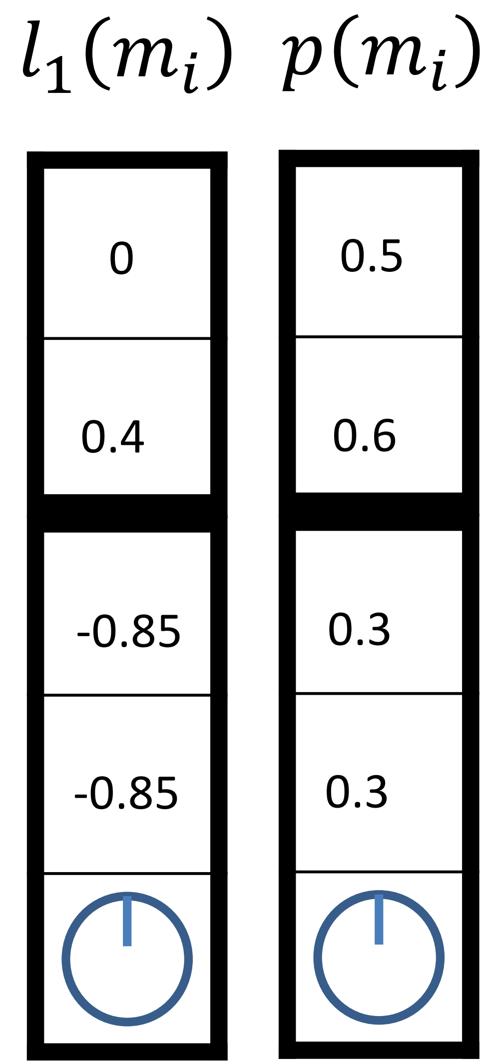
$$l_1(m_4) = l(m_4) - l_0 = 0$$

$$l_1(m_3) = l(m_3) - l_0 = 0.4$$

$$l_1(m_2) = l(m_2) - l_0 = -0.85$$

$$l_1(m_1) = l(m_1) - l_0 = -0.85$$

$$\Rightarrow p = 1 - \frac{1}{1 + \exp l}$$

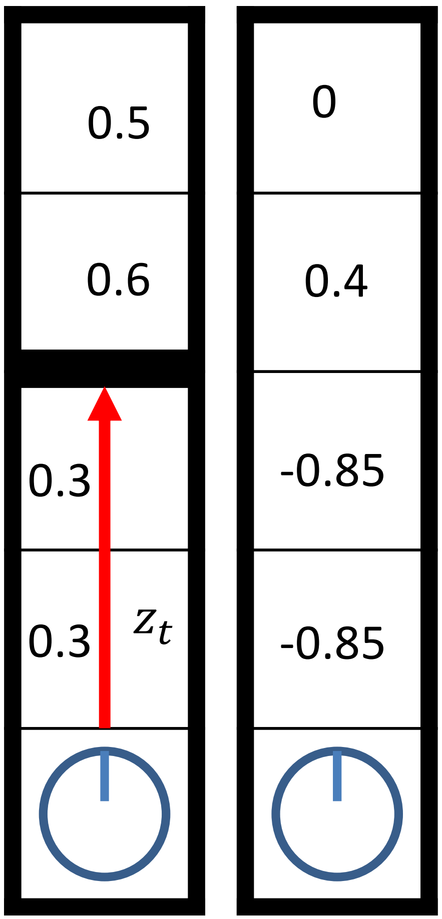


Occupancy Mapping Example

- Distance measurements at $z_{t=2}$= 20cm:

ISM

$p(m_i)$ $l(m_i)$



$$l_2(m_4) = l(m_4) + l_1 - l_0 = 0$$

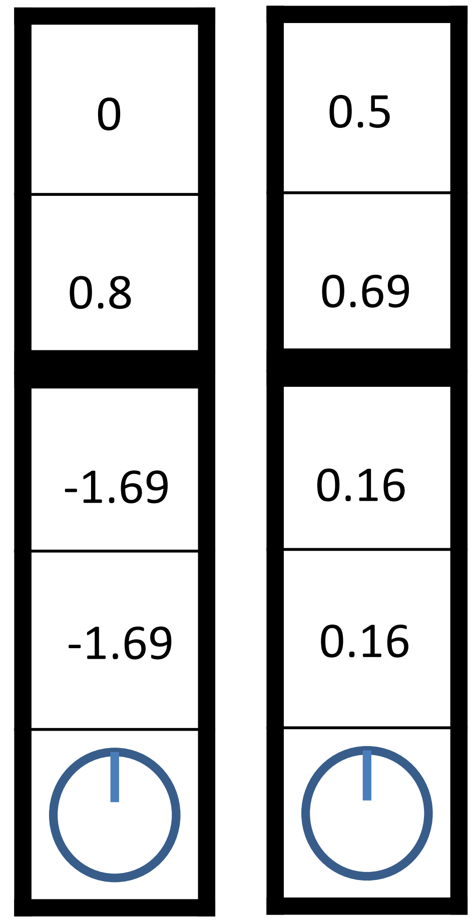
$$l_2(m_3) = l(m_3) + l_1 - l_0 = 0.8$$

$$l_2(m_2) = l(m_2) + l_1 - l_0 = -1.69$$

$$l_2(m_1) = l(m_1) + l_1 - l_0 = -1.69$$

$$\Rightarrow p = 1 - \frac{1}{1 + \exp l}$$

$l_2(m_i)$ $p(m_i)$

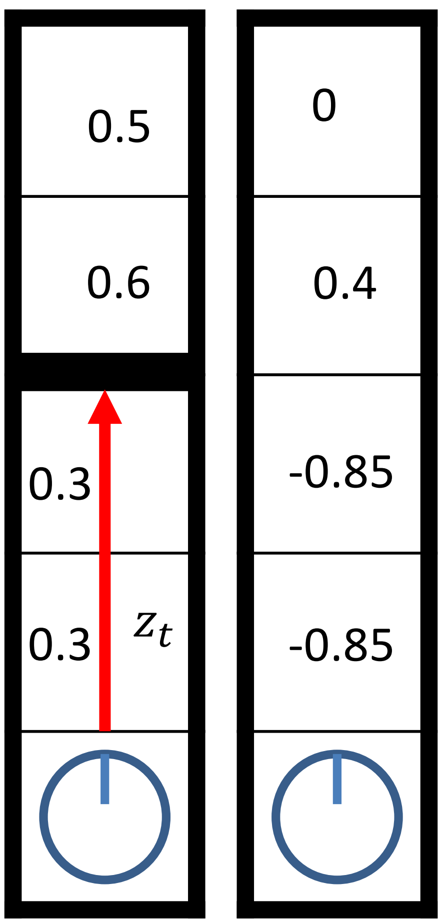


Occupancy Mapping Example

- Distance measurements at $z_{t=3}$= 20cm:

ISM

$p(m_i)$ $l(m_i)$



$$l_3(m_4) = l(m_4) + l_2 - l_0 = 0$$

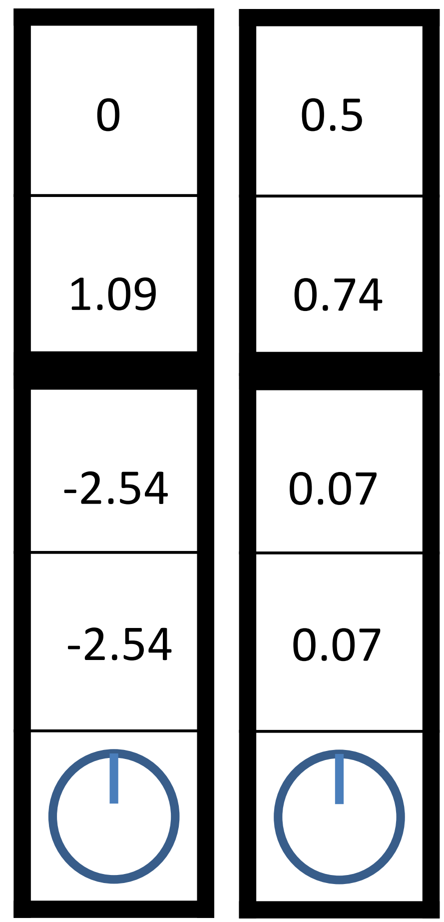
$$l_3(m_3) = l(m_3) + l_2 - l_0 = 1.09$$

$$l_3(m_2) = l(m_2) + l_2 - l_0 = -2.54$$

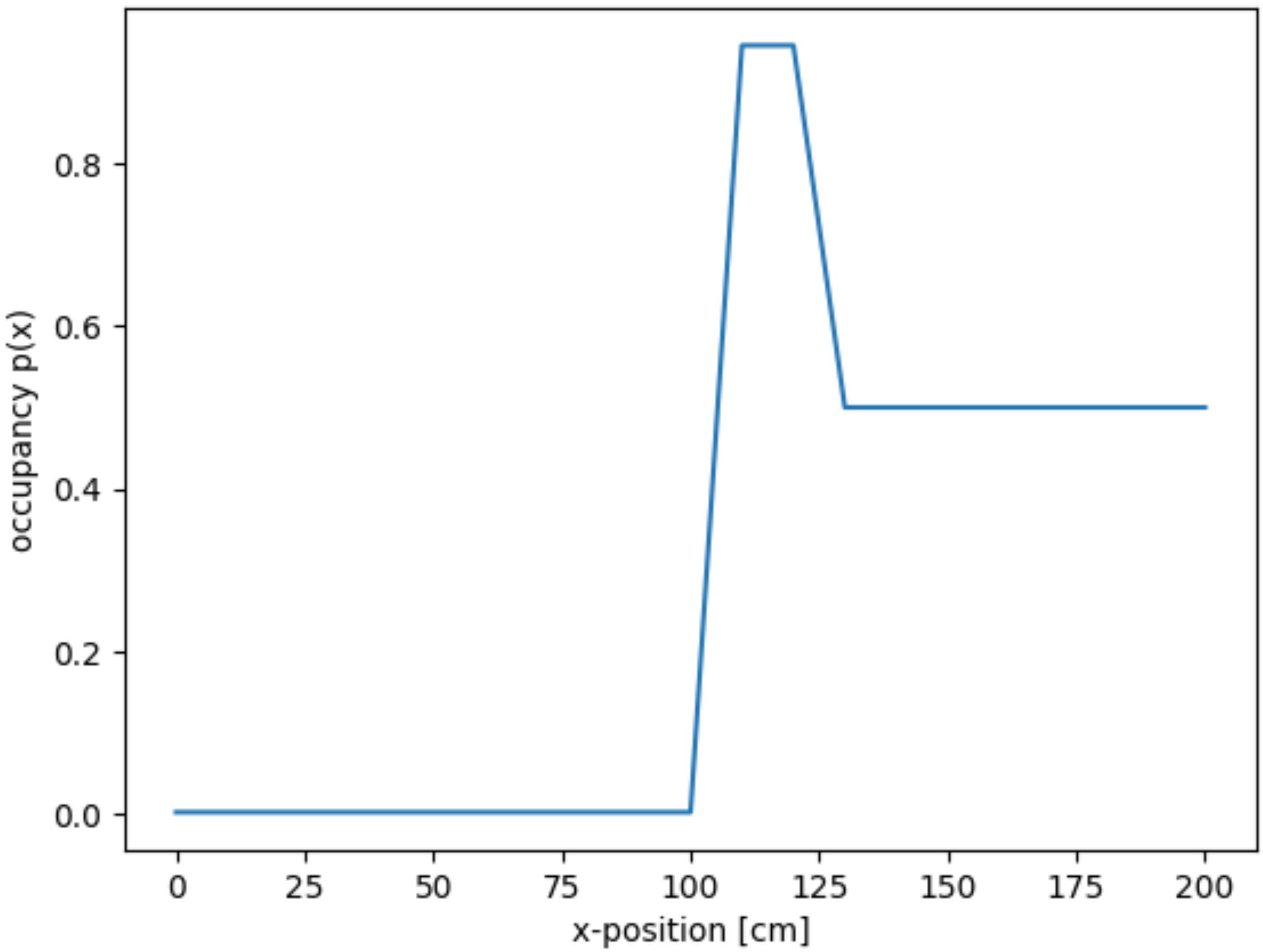
$$l_3(m_1) = l(m_1) + l_2 - l_0 = -2.54$$

$$\Rightarrow p \;= 1 - \frac{1}{1 + \exp l}$$

$l_3(m_i)$ $p(m_i)$

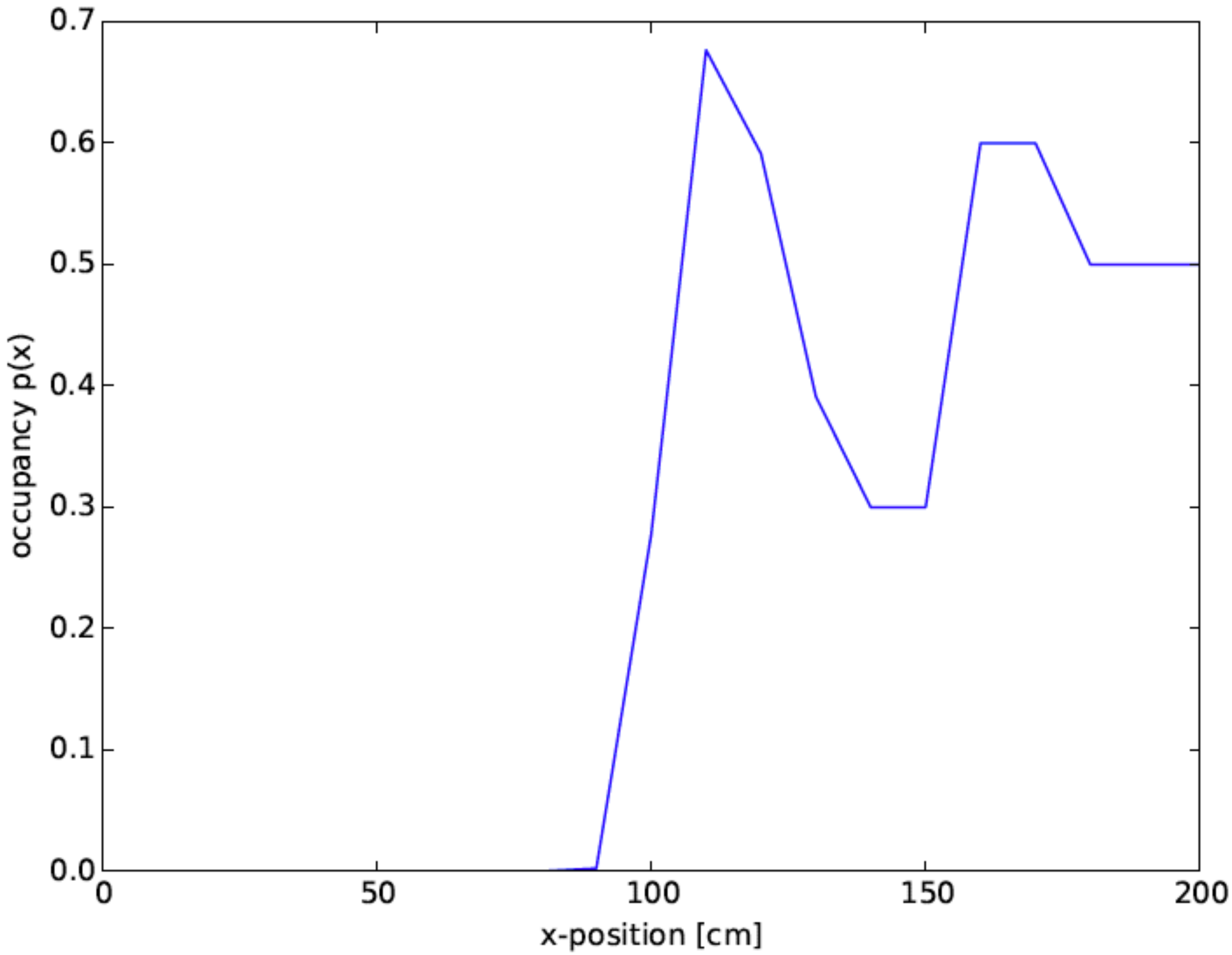


Occupancy Mapping Example



z_t = [100,100,100,100,100,100,100]

Occupancy Mapping Example



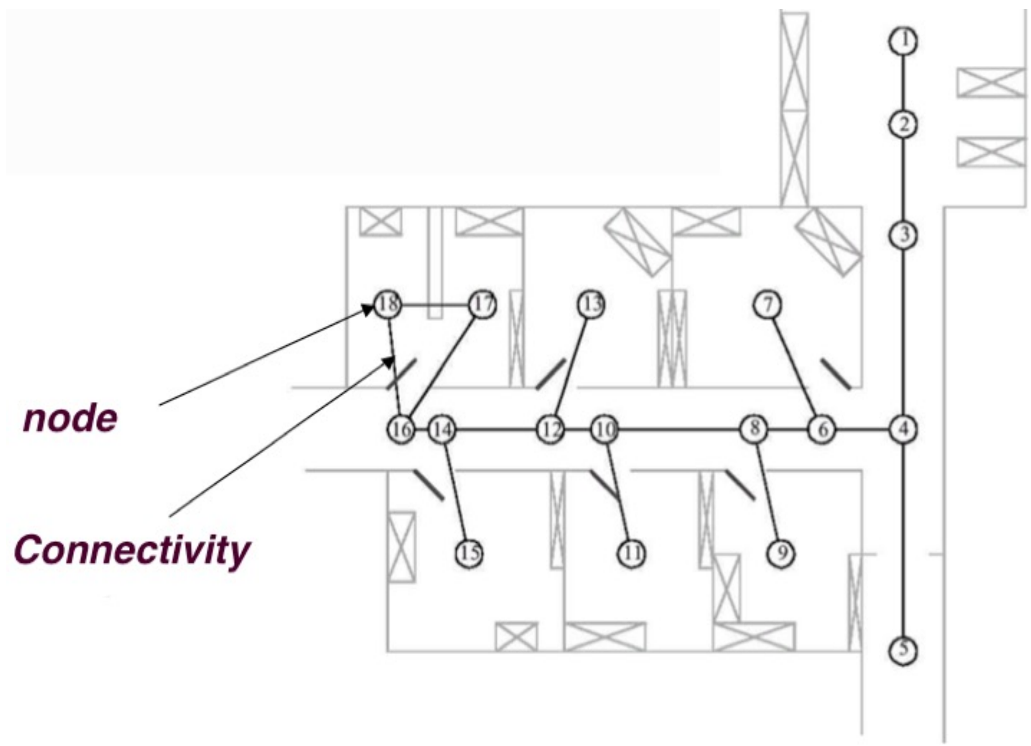
z_t = [101, 82, 91, 112, 99, 151, 96, 85, 99, 105]

Topological Maps

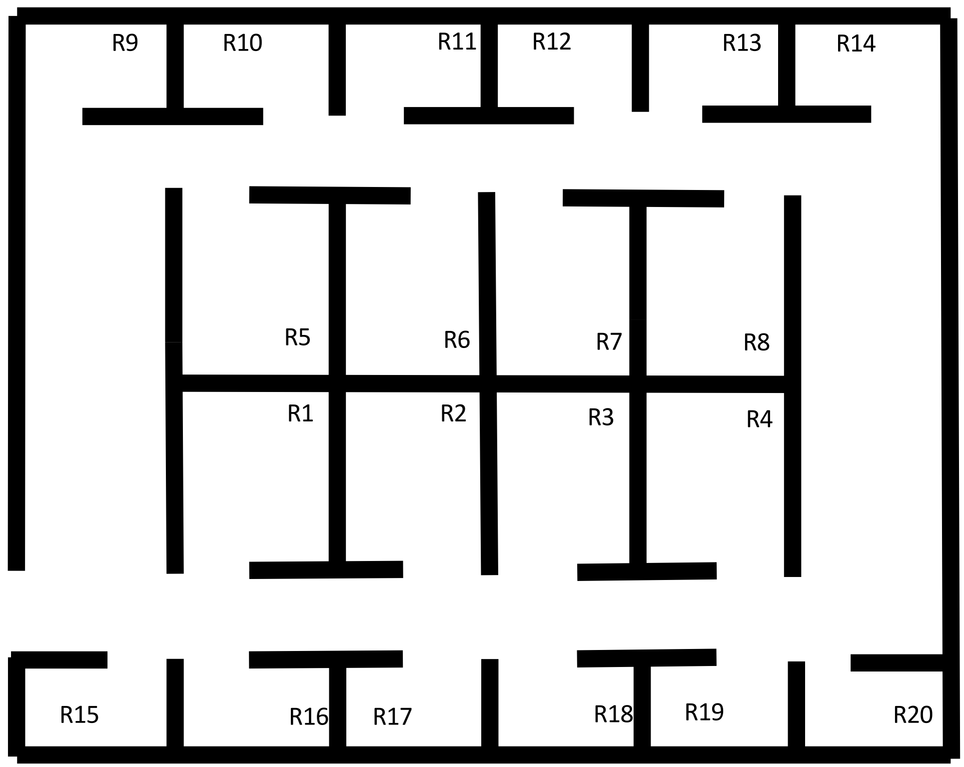
- Use environment features, i.e. landmarks, most useful to robots.
- Navigation is relational between points of interest, e.g. “Go past the corner and enter the second doorway on the left”
- Precise metric information not used
- Approaches are usually based upon graph representations
- A graph specifying nodes and the connectivity between them
  - Nodes are not of fixed size nor specify free space
  - A node is an area the robot can recognize entry and exit
- To robustly navigate with a topological map a robot
  - Must be able to localize relative to nodes
  - Must be able to travel between nodes
  - Robot sensors must be tuned to the particular topological decomposition
- Major advantage is ability to model non-geometric features (like artificial landmarks) that benefit localization

Topological Map Example

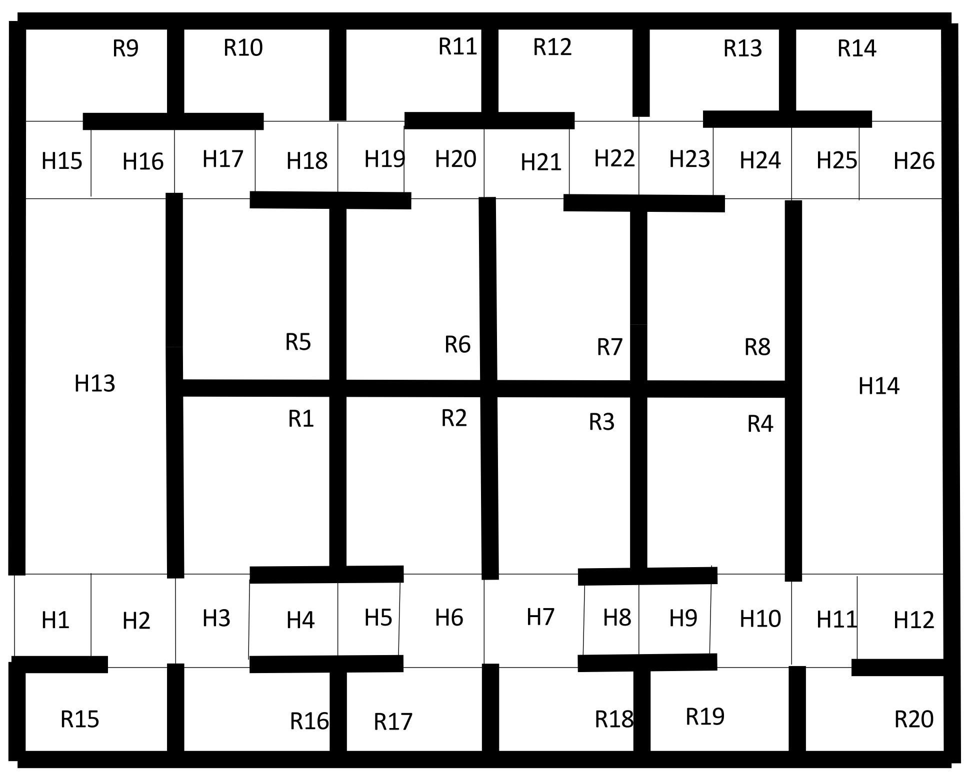
- For example, robot must be able to detect intersections between halls, and between halls and rooms.



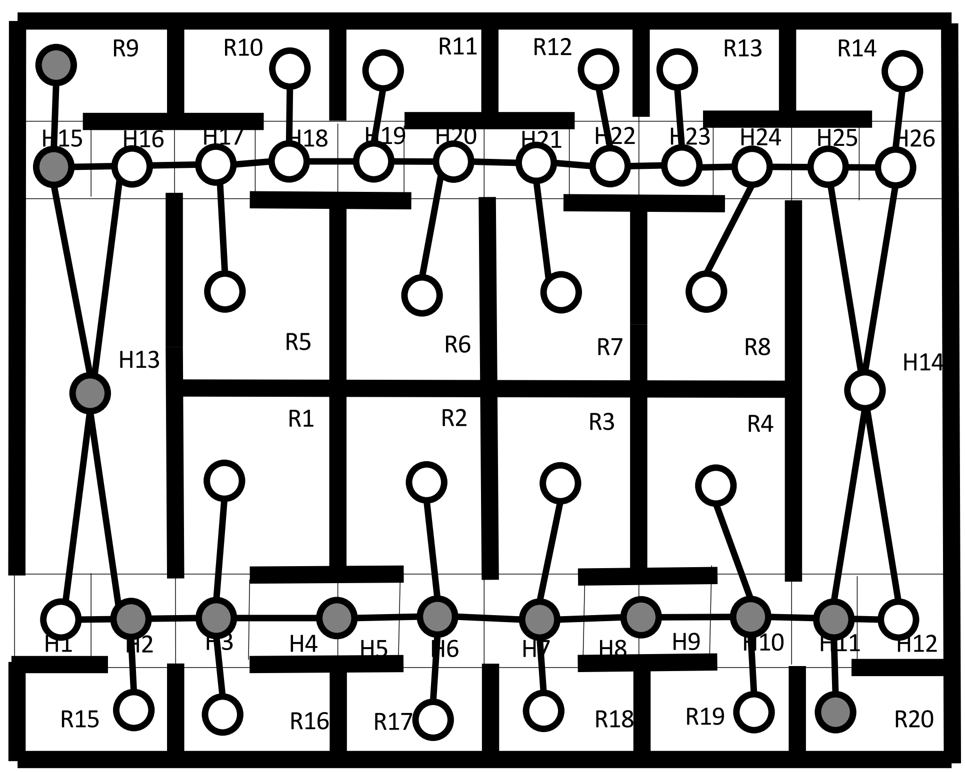
Topological Maps



Topological Maps



Topological Maps



Topological Map Example

- Topology depends on specific map configuration and must distinguish between different node locations
- Different topological map representations:
  - Corners (two walls), single walls, no walls
  - Nodes depending on walls (4x4 grid cells)