Mapping

Alfredo Weitzenfeld

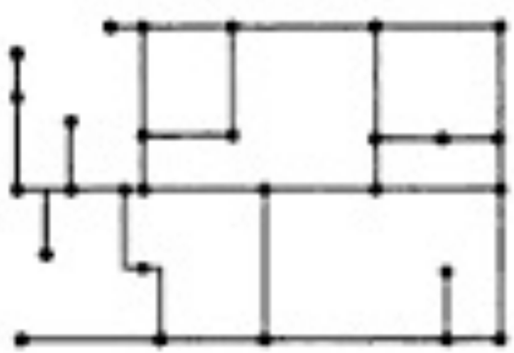
Mapping

- 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.

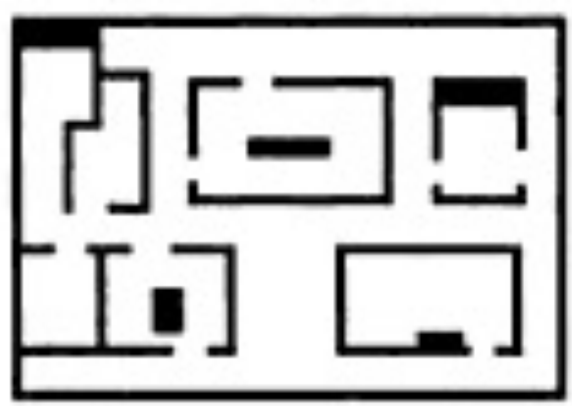
Mapping

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

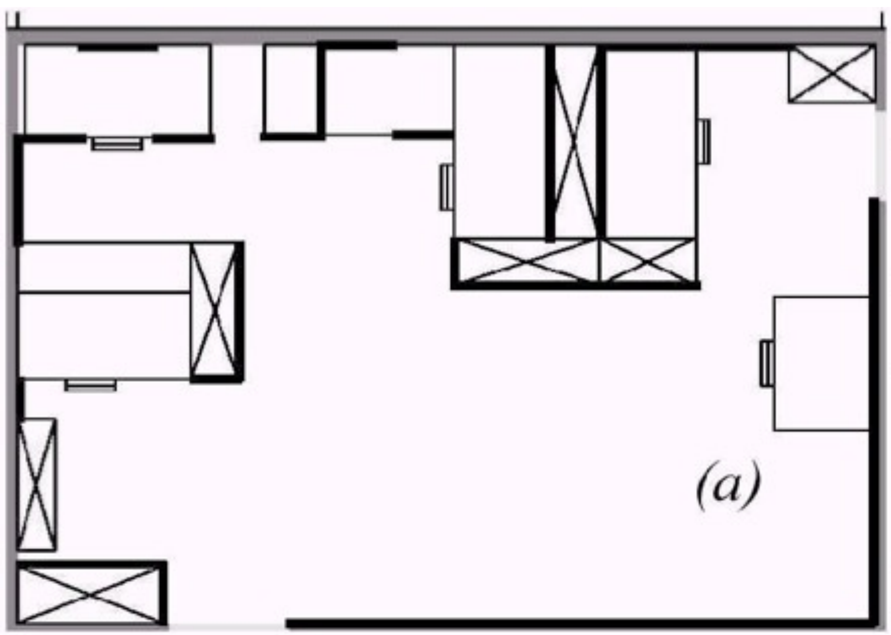
Representation

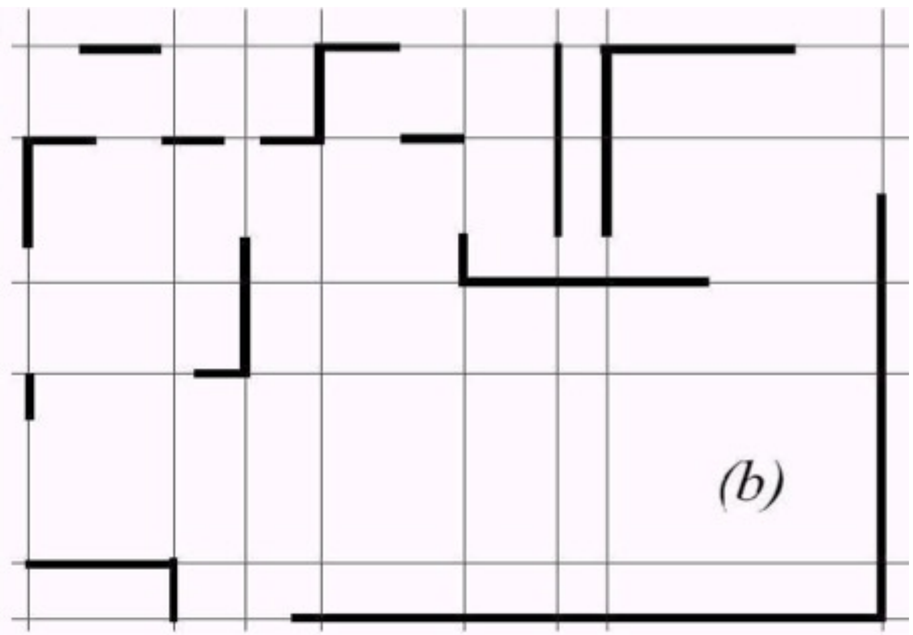
- **Representation** refers to how information is stored or encoded
- **Robot representation**
  - Represent the robot as a point (e.g. Bug Algorithms)
  - Assume robot is capable of omnidirectional motion
  - Robot in reality is of nonzero size
    - Dilation of obstacles by robot radius
    - Resulting 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


(a)

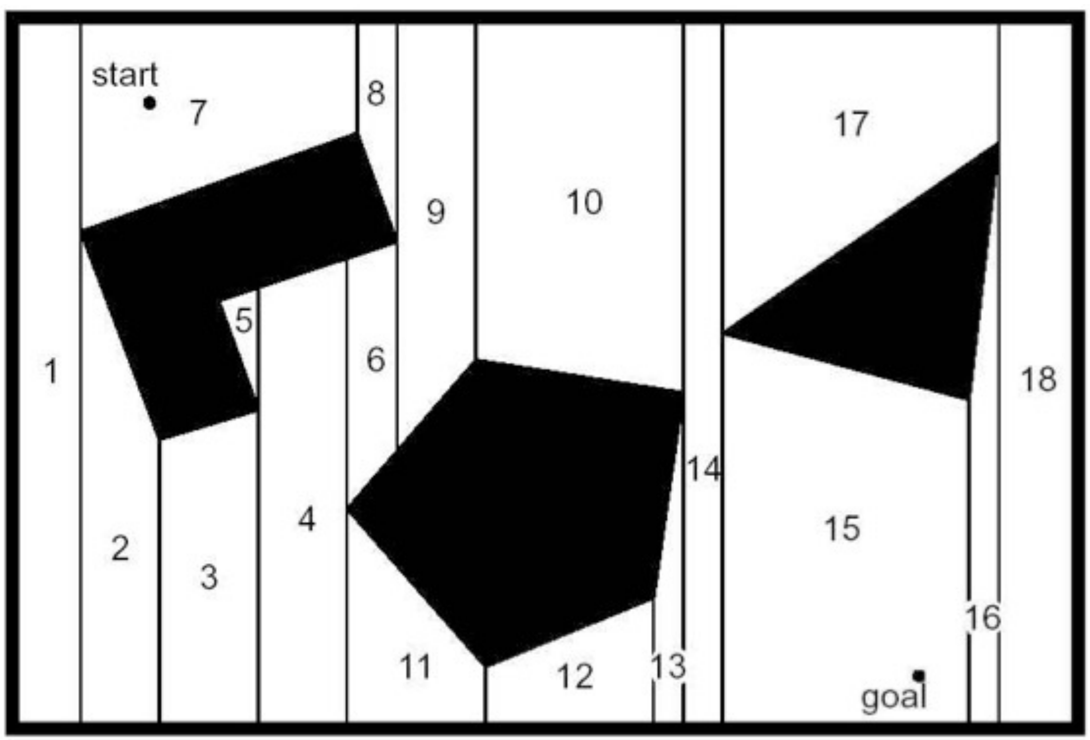

(b)

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

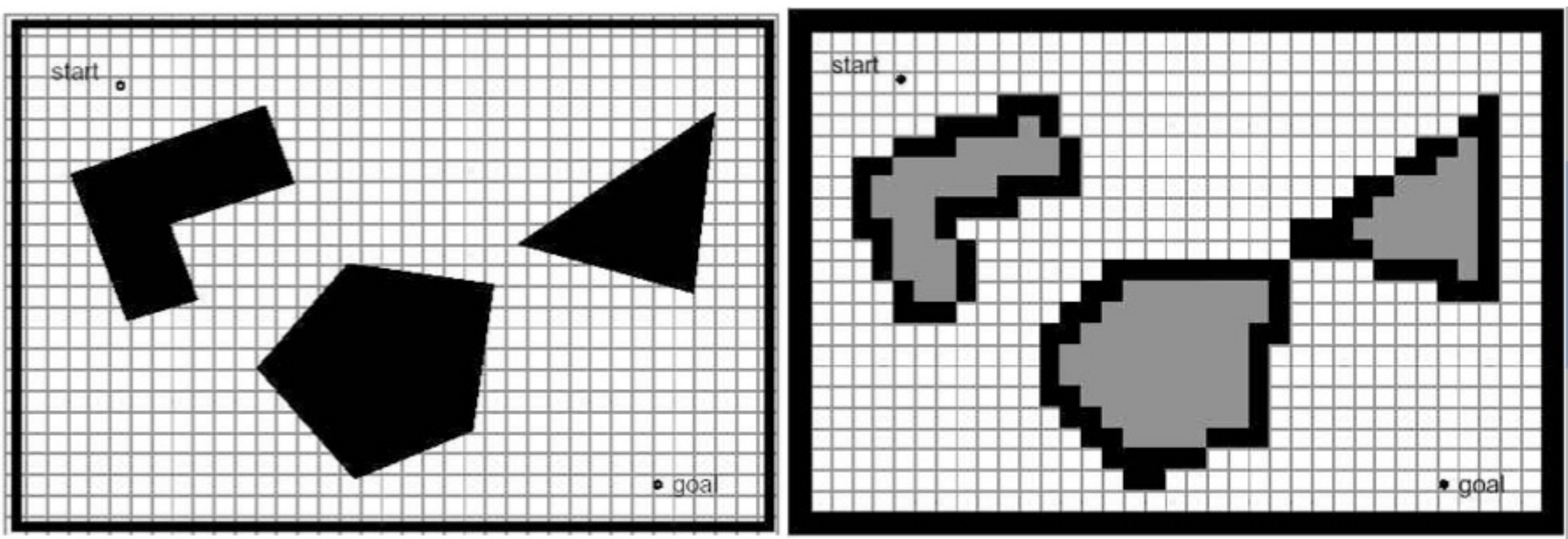
Exact Cell Decomposition

- 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



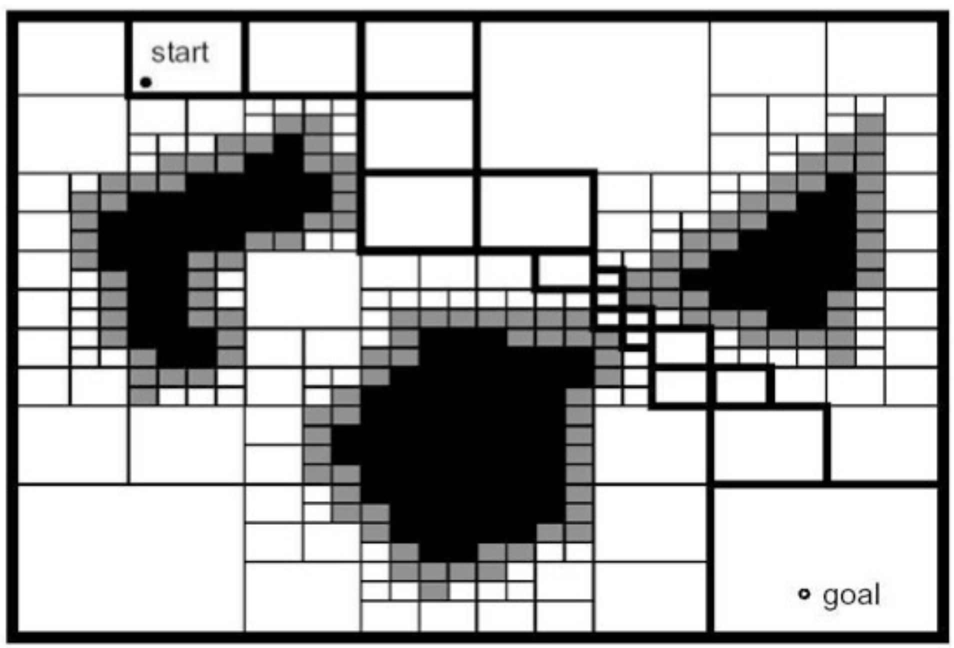
Fixed Cell Decomposition

- 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)



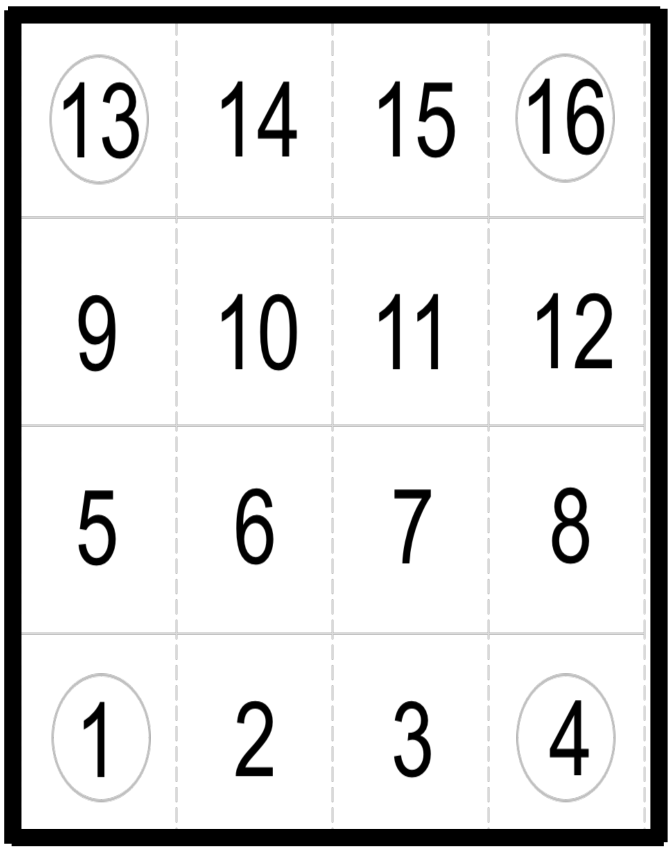
Adaptive Cell Decomposition

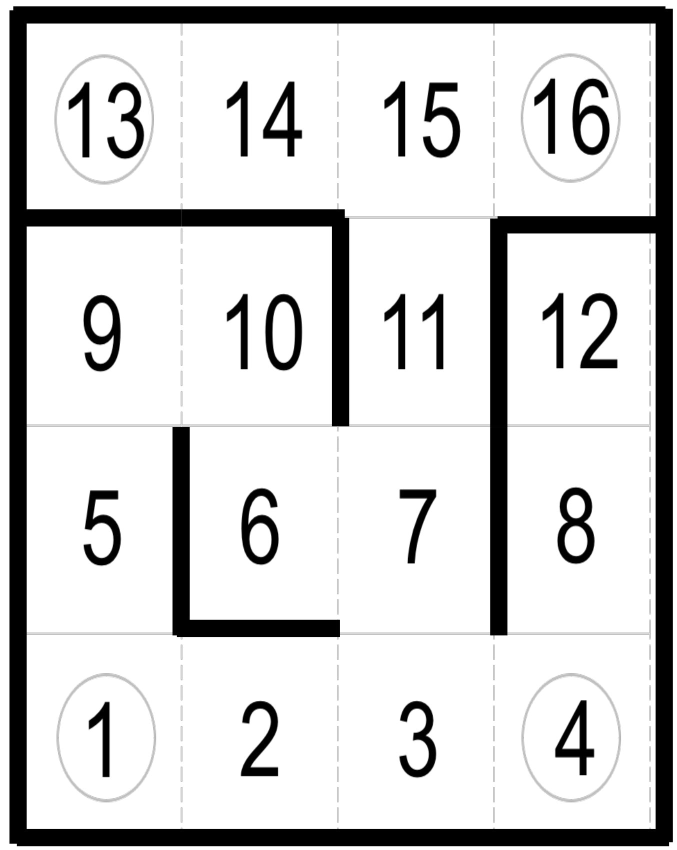
- Multiple types of adaptation: quadtree or other
- Recursively decompose free space until a cell is completely empty or full (there may be loss of information as with fixed cell decomposition)
- Space efficient if compared to fixed cell approach



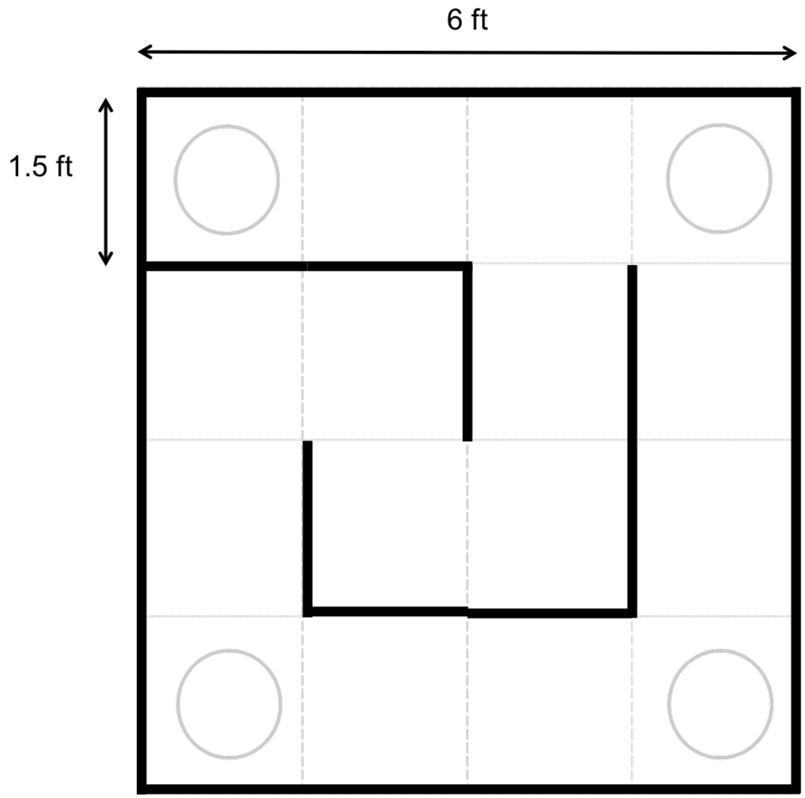
Fixed Cell Decomposition Example

- 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

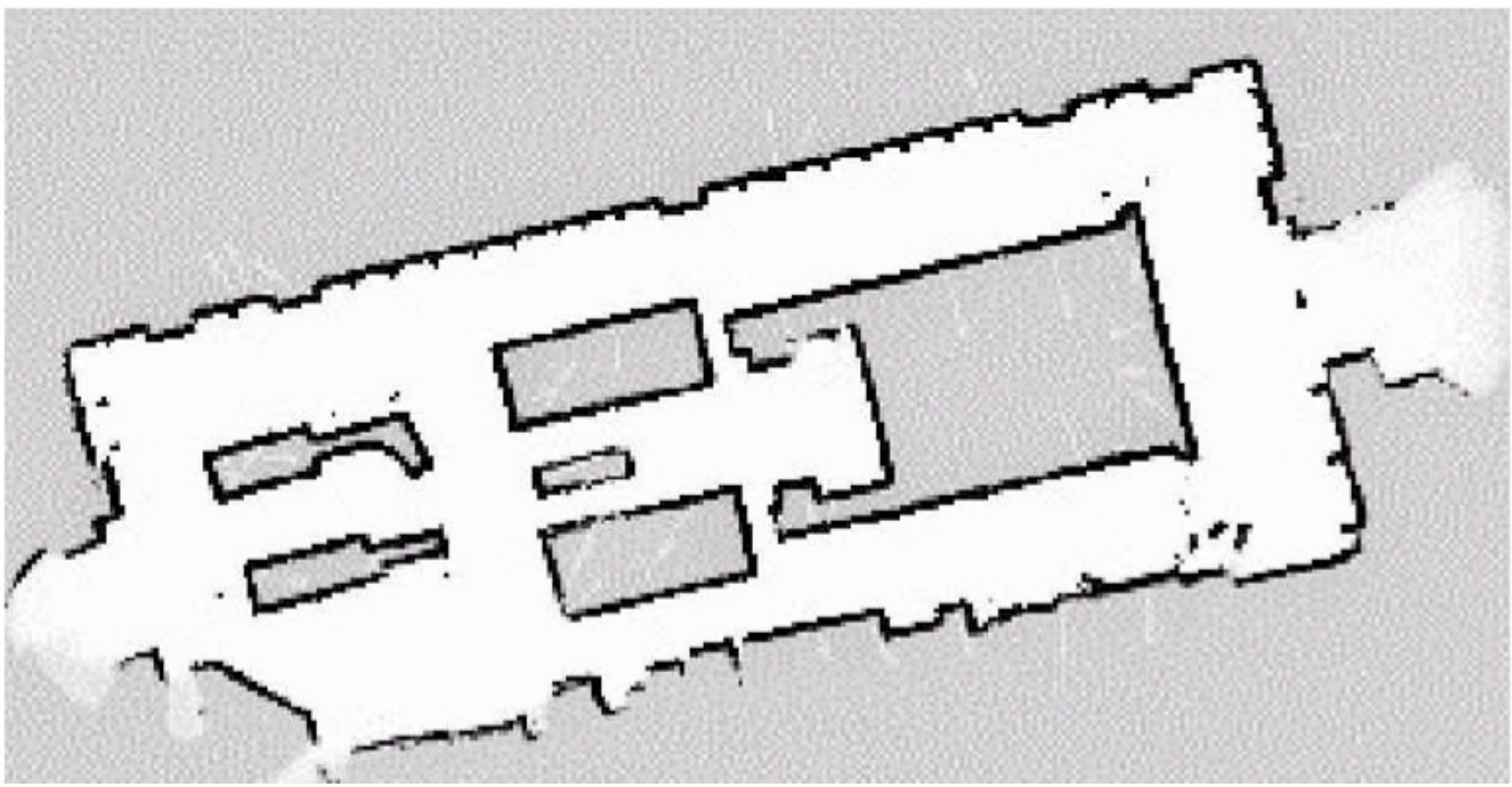
- *W*: Walls and Wall Corners
- *1-16*: Grid cell locations where robot may be found in the maze
- Empty cells are either possible locations of wall or robot

Occupancy Grid Maps

- 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

- Darkness of cell proportional to cell counter value



Believes with Static States

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

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

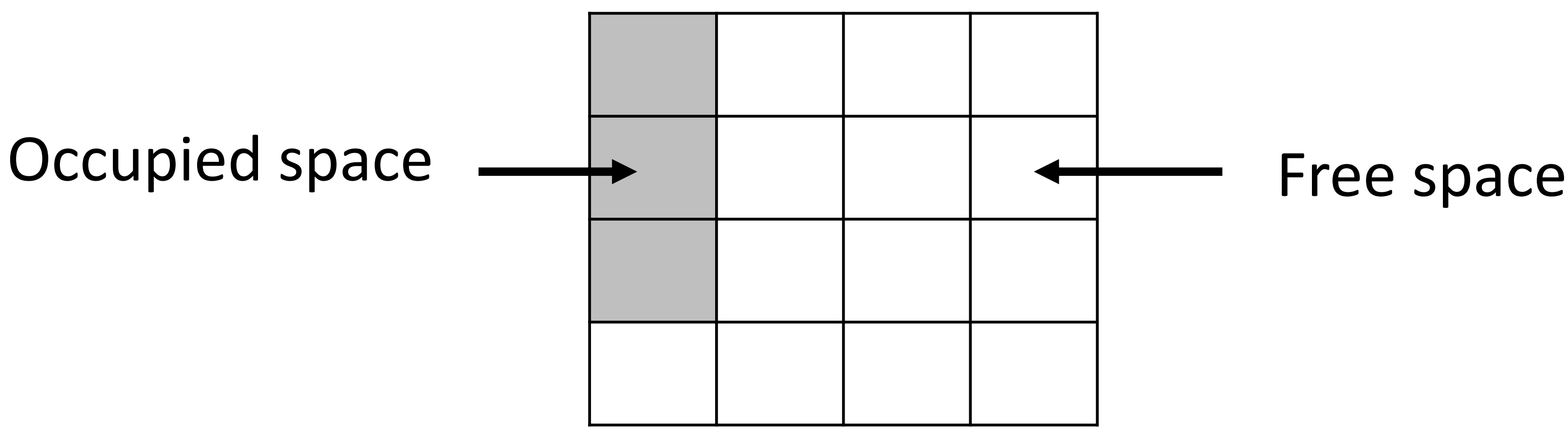
Binary Bayes Filter with Static States

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

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

Occupancy Grid Maps

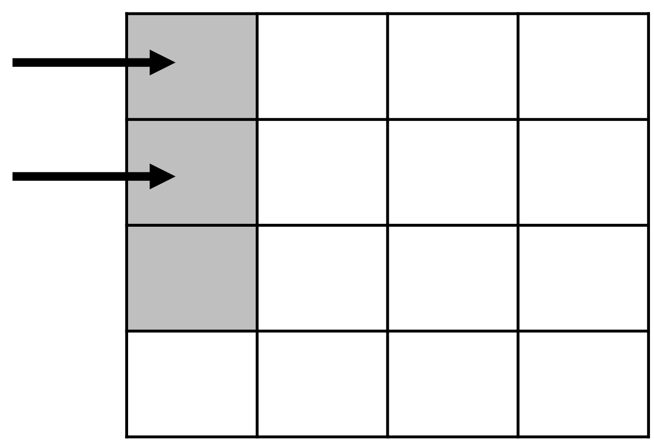
- The area that corresponds to a cell is either completely free or occupied.



Occupancy Grid Maps

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

No dependency between the cells



Occupancy Grid Maps

- 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

- 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

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

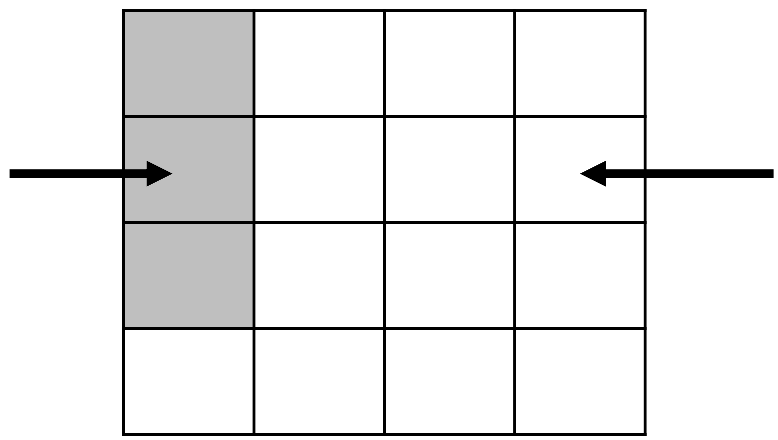
- The probability of the cell being free is given by

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

Occupancy Grid Maps

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

$p(m_i = 1) \rightarrow 1$



$p(m_i = 0) \rightarrow 0$

- 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 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:

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

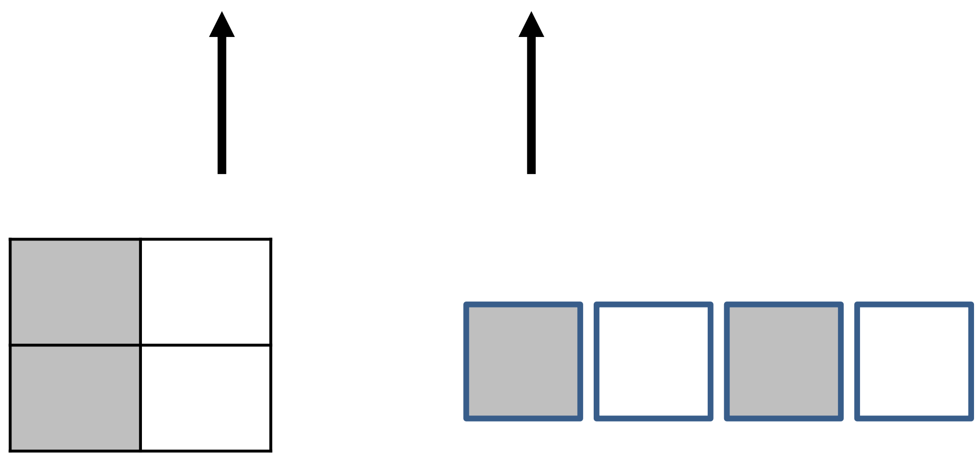
- This equation is equivalent to:

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

Occupancy Grid Maps

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

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



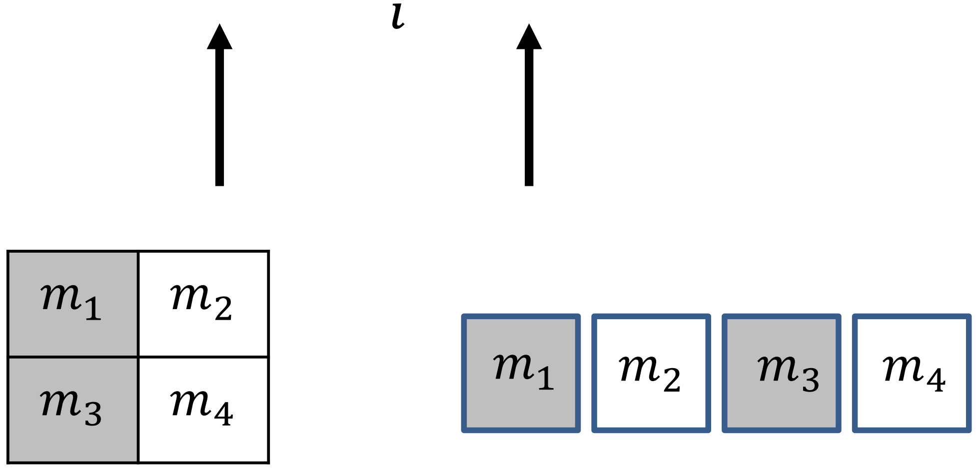
Example map
(4-dimension vector)

4 individual cells

Occupancy Grid Maps

Assume: $p(m_1)$=0.7 $p(m_2)$=0.4 $p(m_3)$=0.9 $p(m_4)$=0.2

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



$$p(m) = p(m_1) * (1 - p(m_2)) * p(m_3) * (1 - p(m_4))$$
$$= 0.7 * (1 - 0.4) * 0.9 * (1 - 0.2)$$

Inverse Sensor Measurement Model

Assume:

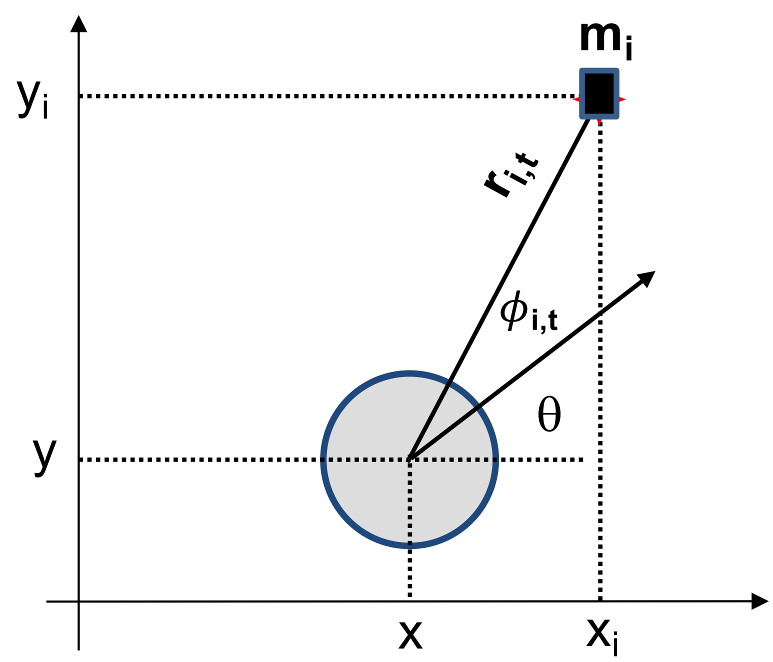
- s_t= $[x, y, \theta]^T$ is the robot pose at time t
- m_i = (x_i, y_i) is the location of the cell (can also be applied to landmarks)
- θ is the robot orientation
- $\phi_{i,t}$ is the relative orientation of grid cell m_i
- $r_{i,t}$ is the relative distance of grid cell m_i
- $z_{i,t}$ is the inverse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

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

where

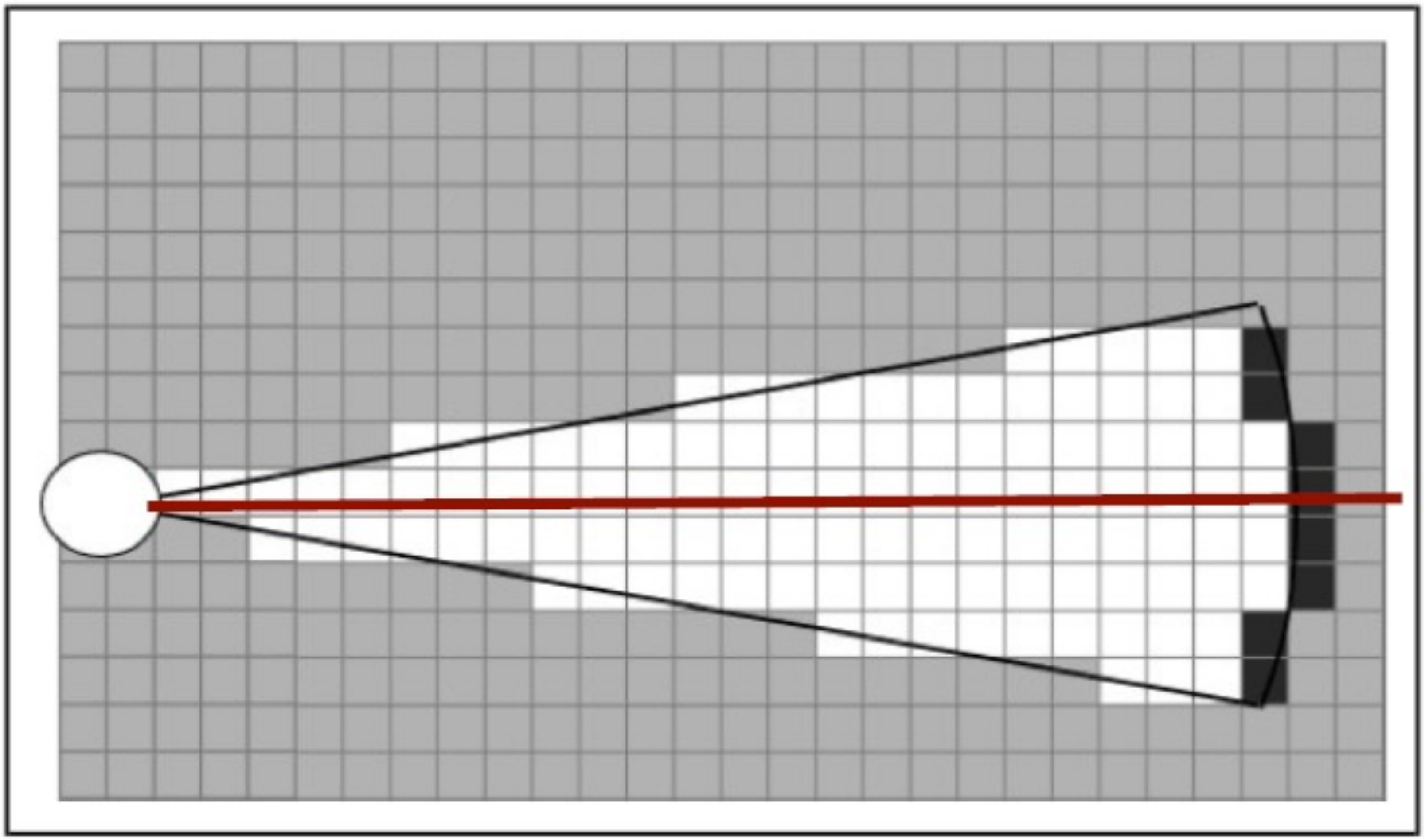
$$r_{i,t} = \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

$$\phi_{i,t} = atan2\big((y_i - y), (x_i - x)\big) - \theta$$



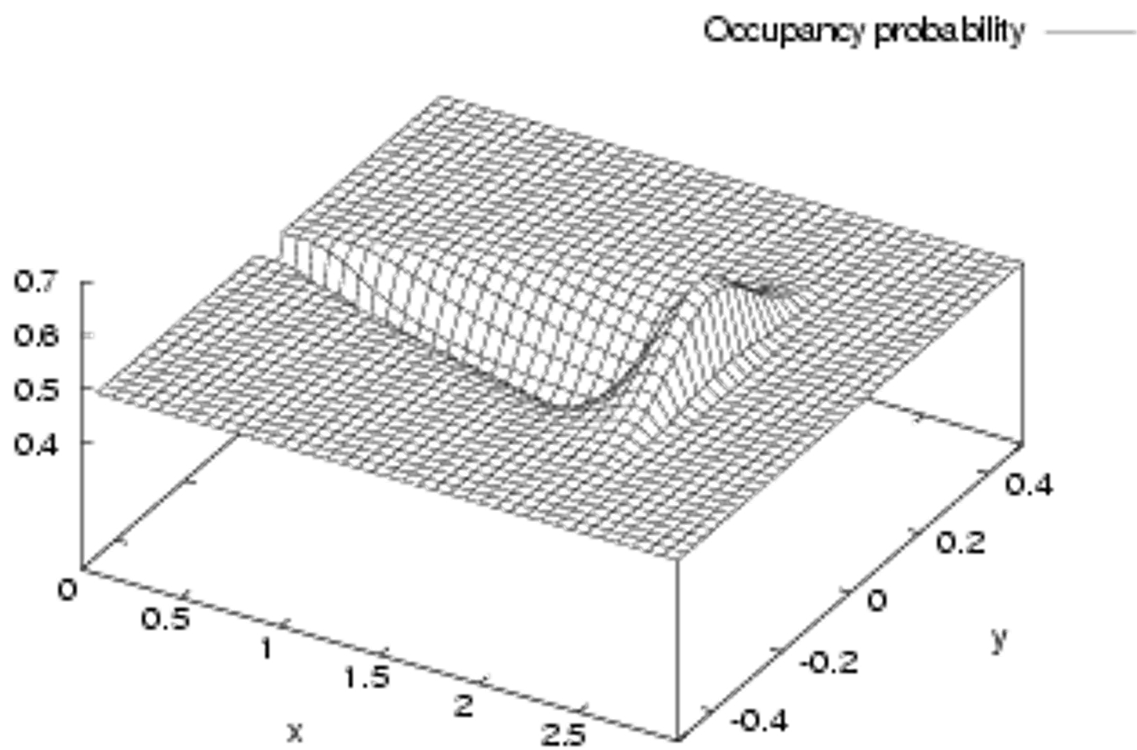
Occupancy Grid Maps

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

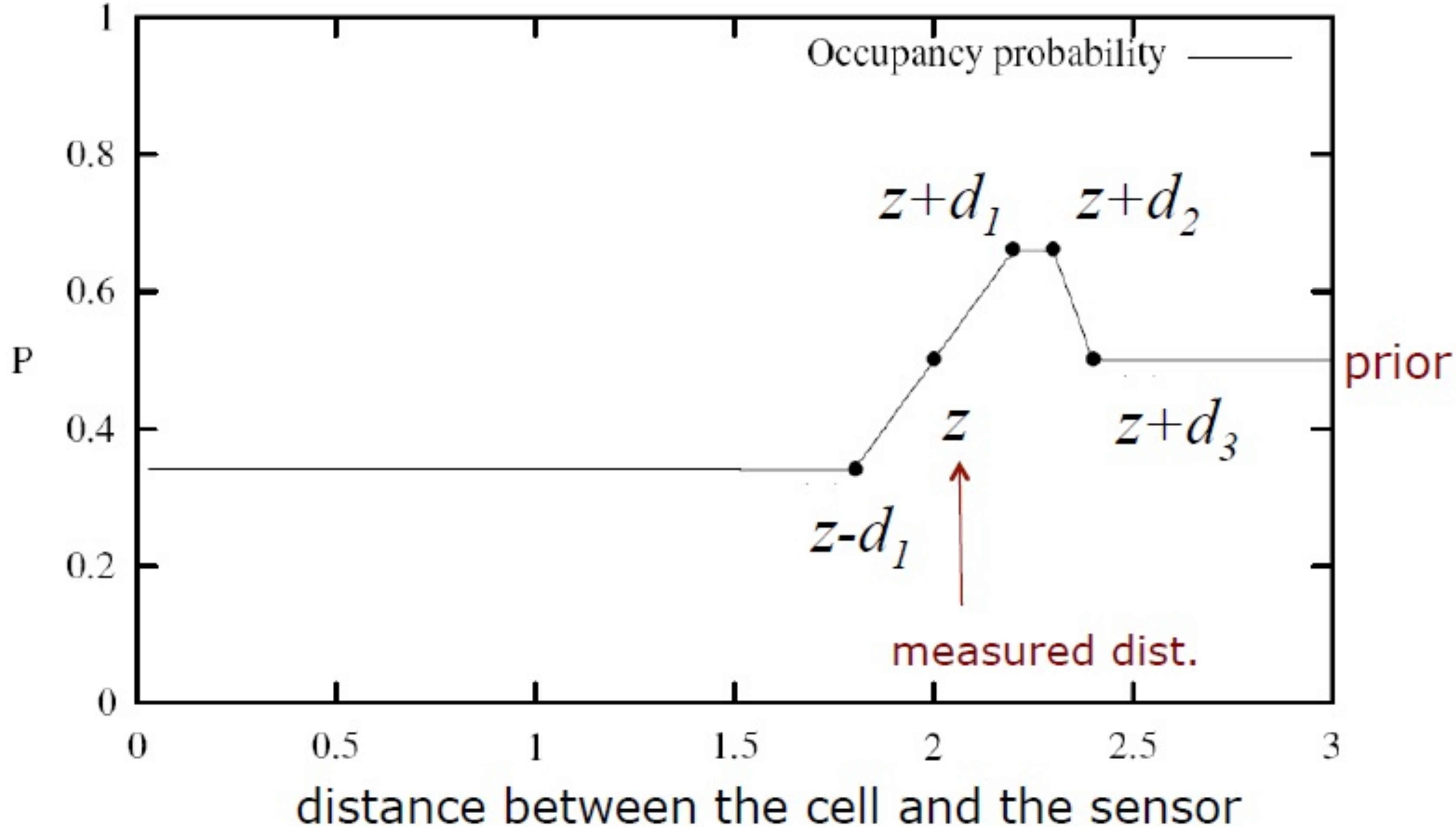


Occupancy Grid Maps

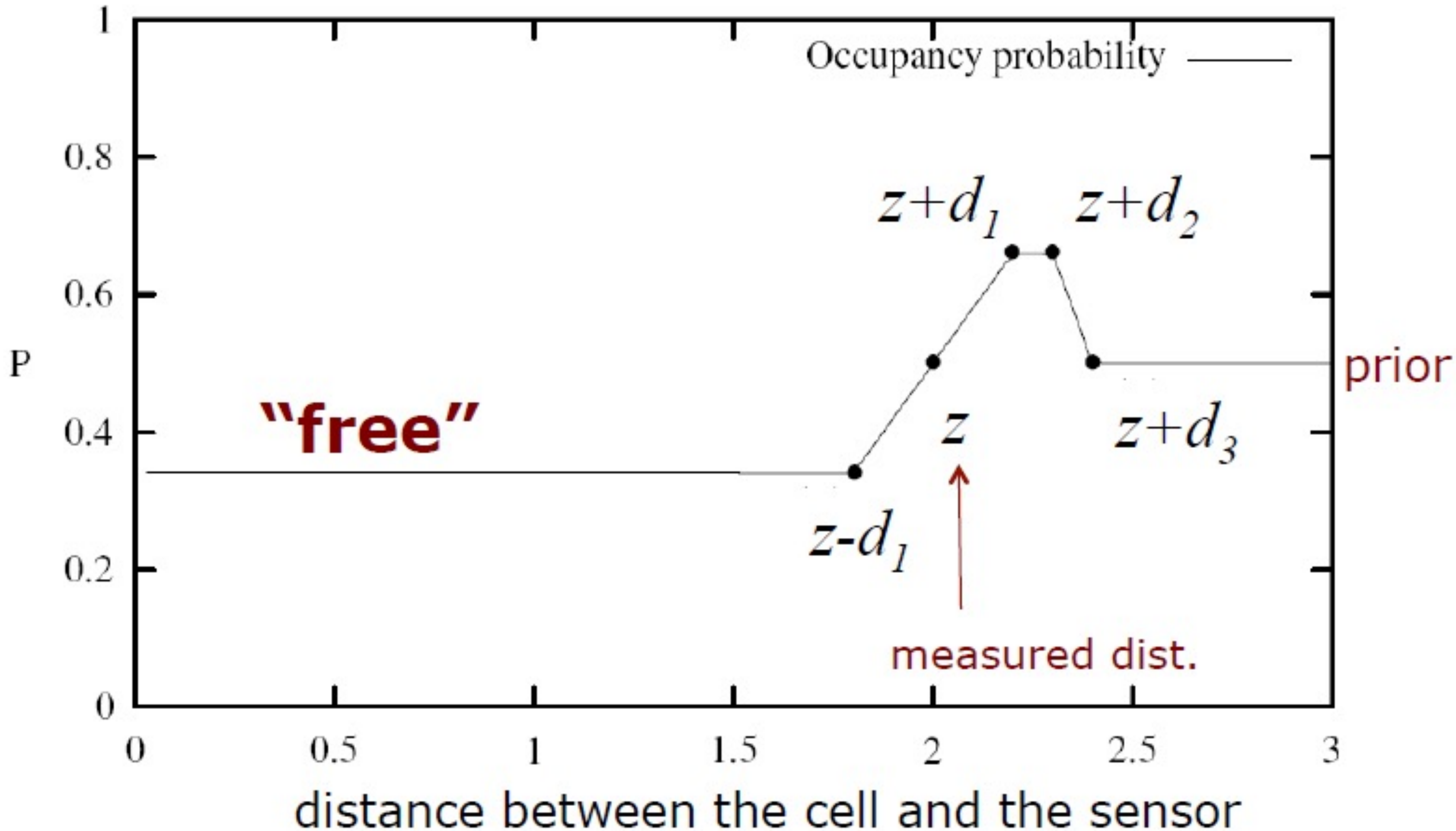
- Typical occupancy sensor model in 3D graph
- Combination of a linear function and a Gaussian



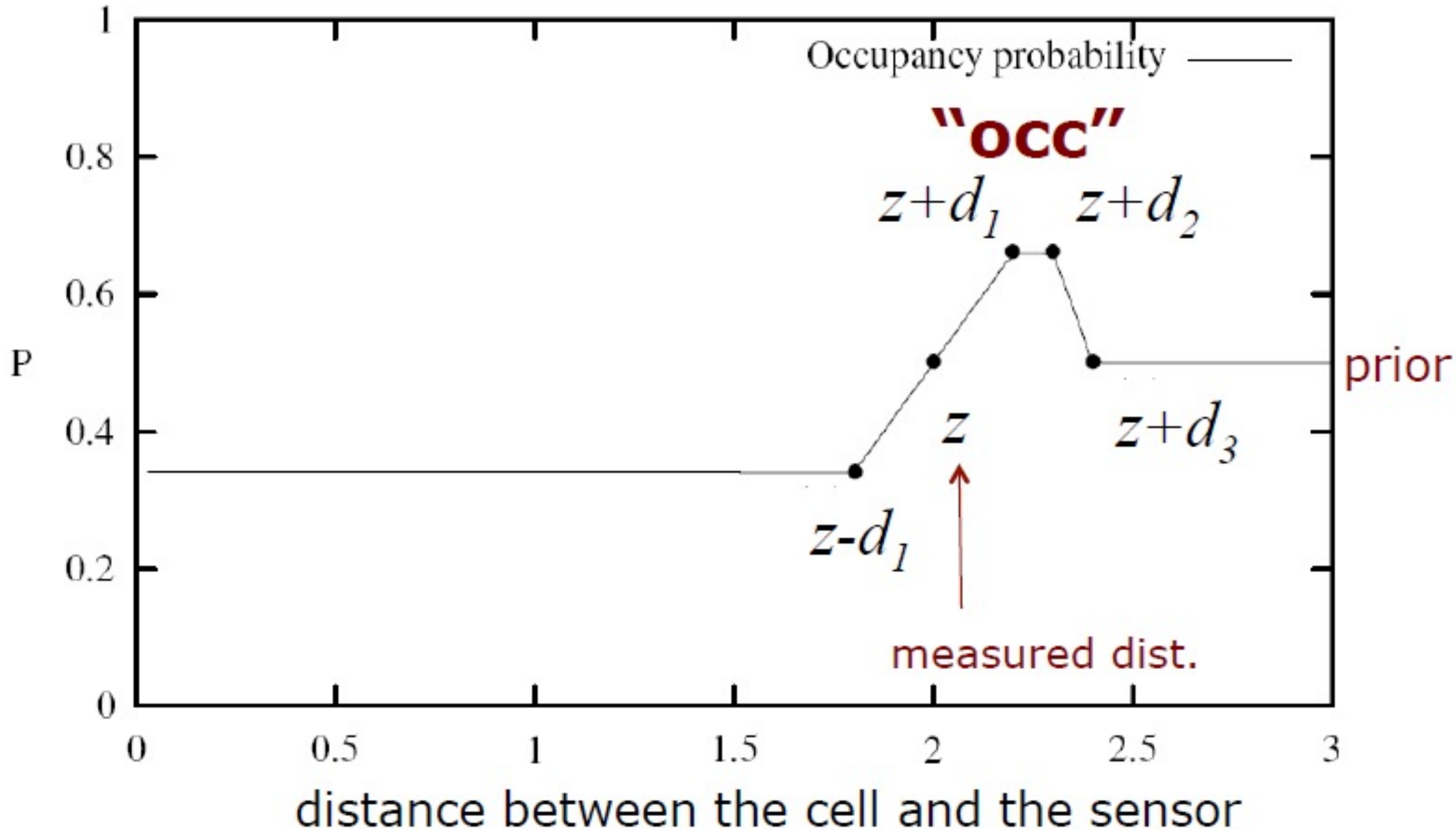
Occupancy Grid Maps
Value Depending on Measured Distance



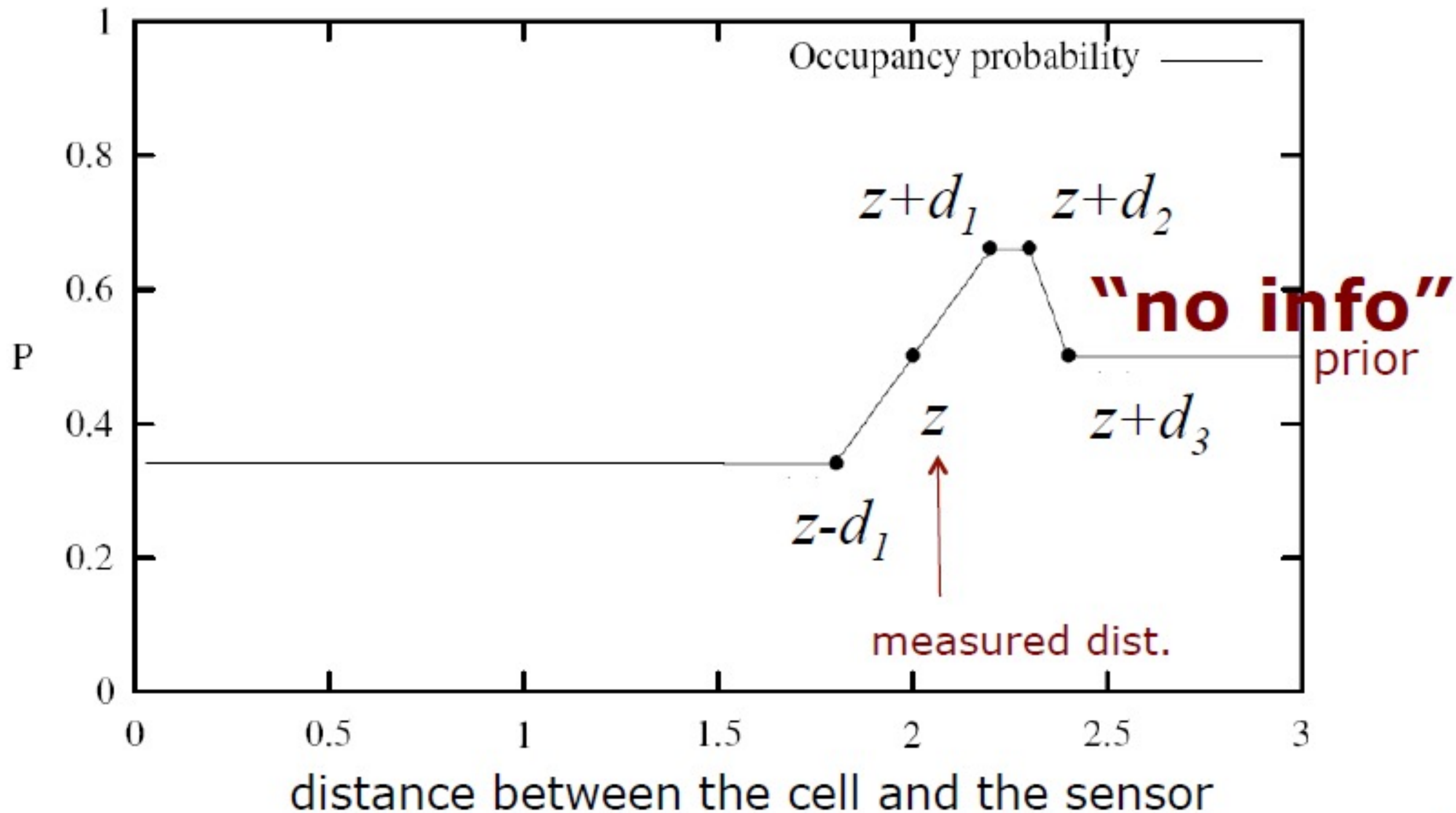
Occupancy Grid Maps
Value Depending on Measured Distance



Occupancy Grid Maps
Value Depending on Measured Distance

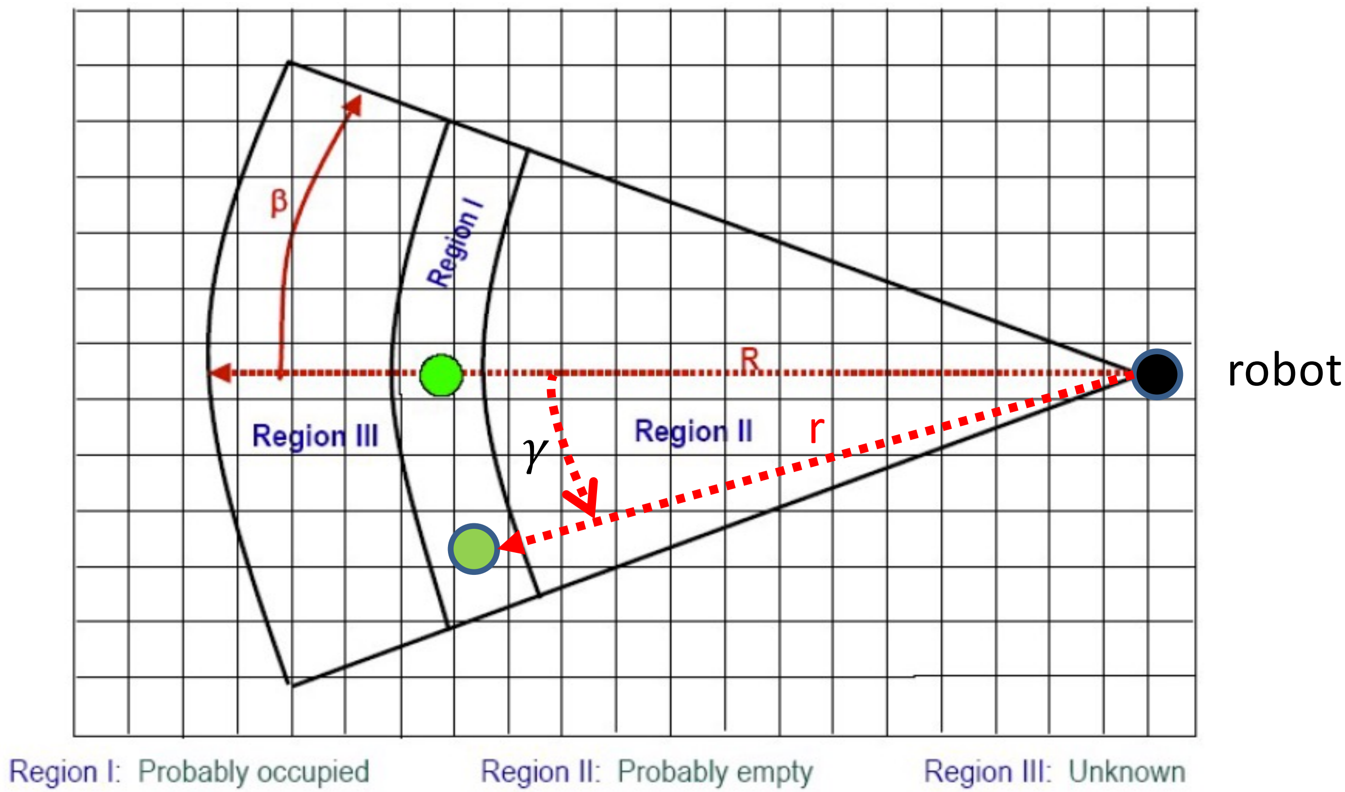


Occupancy Grid Maps
Value Depending on Measured Distance



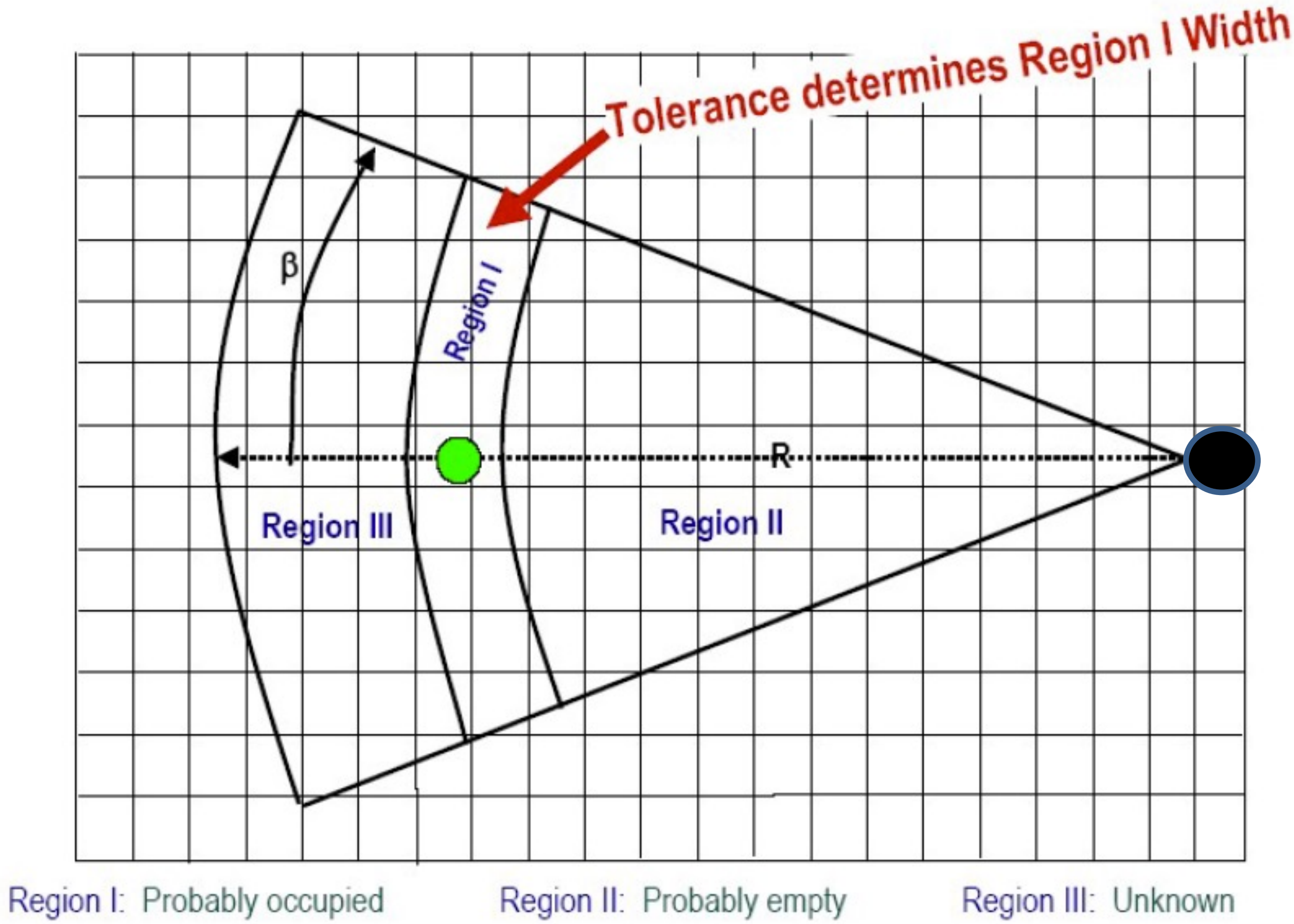
Occupancy Grid Maps

- 3 regions in relation to obstacle: (I) probably occupied by obstacle, (II) probably empty, (III) unknown



- r is the distance to the grid cell
- γ is the angle to the grid cell
- $R = z_{max}$ is the maximum range the sensor can detect
- β is the width of the sensor beam
- α is the thickness of the obstacle (Region I).

Occupancy Grid Maps



- Range readings have resolution error.
- Reading indicates range of possible values, e.g., reading of 0.87 meters with tolerance +/- 0.05 meters, is within (0.82, 0.92) meters
- Tolerance α gives width of Region I.

Occupancy Grid Maps

<u>Region I</u>:

The nearer to the sensor origin, the higher the belief

The closer to the main axis, the higher the belief

$$p_{occupied} = \frac{\frac{R-r}{R} + \frac{\beta-\gamma}{\beta}}{2}$$

$$p_{empty} = 1.0 - p_{occupied}$$

- r is the distance to the grid cell
- γ is the angle to the grid cell
- R is the maximum range the sensor can detect
- β is the width of the sensor beam

Occupancy Grid Maps

Region II:

The nearer to the sensor origin, the higher the belief

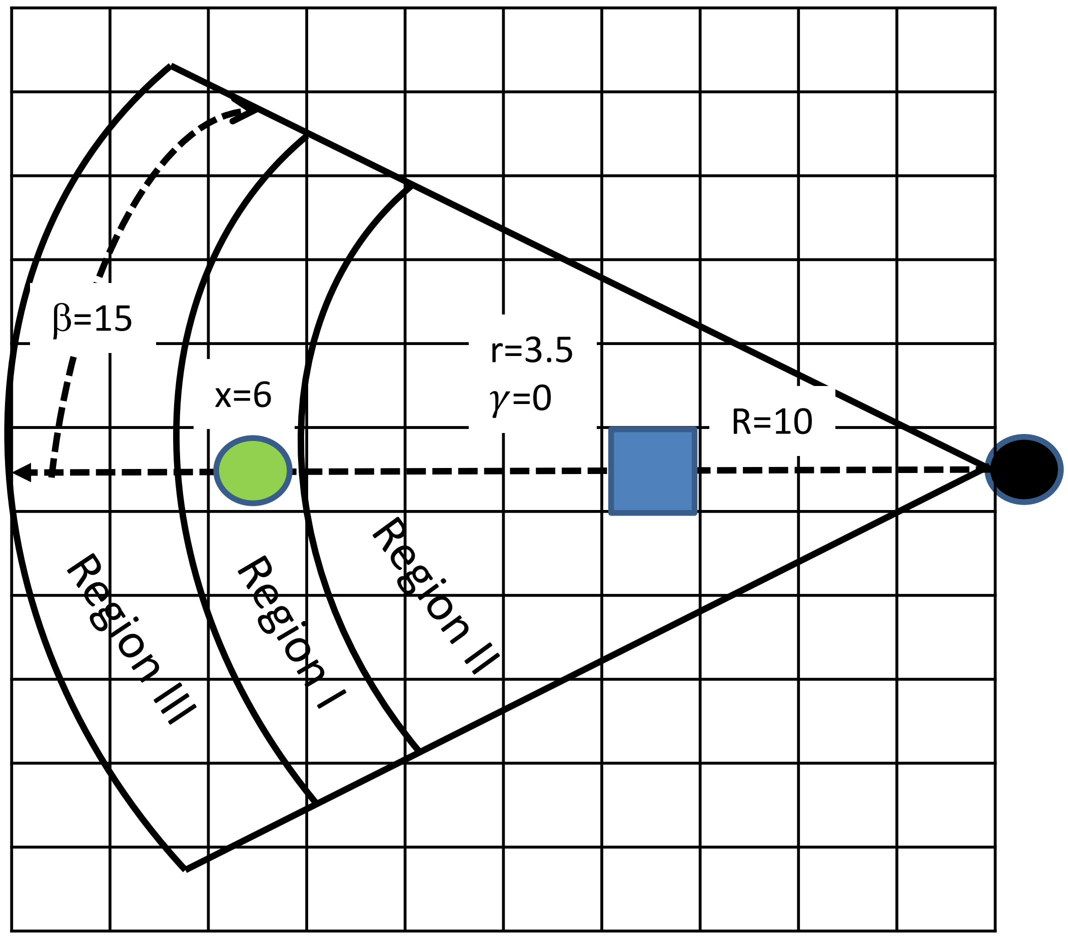
The closer to the main axis, the higher the belief

$$p_{empty} = \frac{\frac{R-r}{R} + \frac{\beta-\gamma}{\beta}}{2}$$

$$p_{occupied} = 1.0 - p_{empty}$$

- r is the distance to the grid cell
- γ is the angle to the grid cell
- R is the maximum range the sensor can detect
- β is the width of the sensor beam

Occupancy Grid Maps



Which region?
3.5 < (6.0-0.5) → Region II

$$p_{empty} = \frac{\frac{R-r}{R} + \frac{\beta-\gamma}{\beta}}{2}$$

$$= \frac{\frac{10-3.5}{10} + \frac{15-0}{15}}{2} = 0.83$$

$$p_{occupied} = 1.0 - p_{empty}$$
= (1.0-0.83) = 0.17

- x = 6 represents the distance read by the sensor to the obstacle.
- r = 3.5 represents the grid cell location where the probability is being computed
- Each square is 0.5 meters in length.

Inverse Sensor Measurement Model

inverse_sensor_model(m_i, s_t, z_t):

1. Let x_i, y_i be the center-of-mass of grid cell m_i
2. Let s_t = (x, y, θ)
3. $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$
4. $\phi = atan2\big((y_i - y), (x_i - x)\big) - \theta$
5. k = arg min$_j$ | $\phi - \theta_j$ |
6. if $r >$ min(z_{max},z_t^k+ $\alpha/2$) or | $\phi - \theta_j$ | > $\beta/2$ then
7. // no new information obtained about m_i
8. else if $z_t^k < z_{max}$ and | r - z_t^k | < $\alpha/2$
9. // measure m_i as occupied
10. else if $r \leq z_t^k$
11. // measure m_i as free

- k is the beam index.

Logs Odd Notation

- **Logs 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:

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

- Retrieving $p(s)$:

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

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:

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

latest probability computation　　recursive term　　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 only, no multiplications
- The computation is based on the inverse sensor model, p(s_t | z_t), instead of the forward model p(z_t | s_t). The inverse sensor model specifies a distribution over the (binary) state variable as a function of the measurement z_t.