اصول علم ربات – اسلاید پانزدهم

Fundamentals of Robotics – Slide 15

Navigation in Presence of Obstacles
Maps, Local Maps

دکتر مهدی جوانمردی

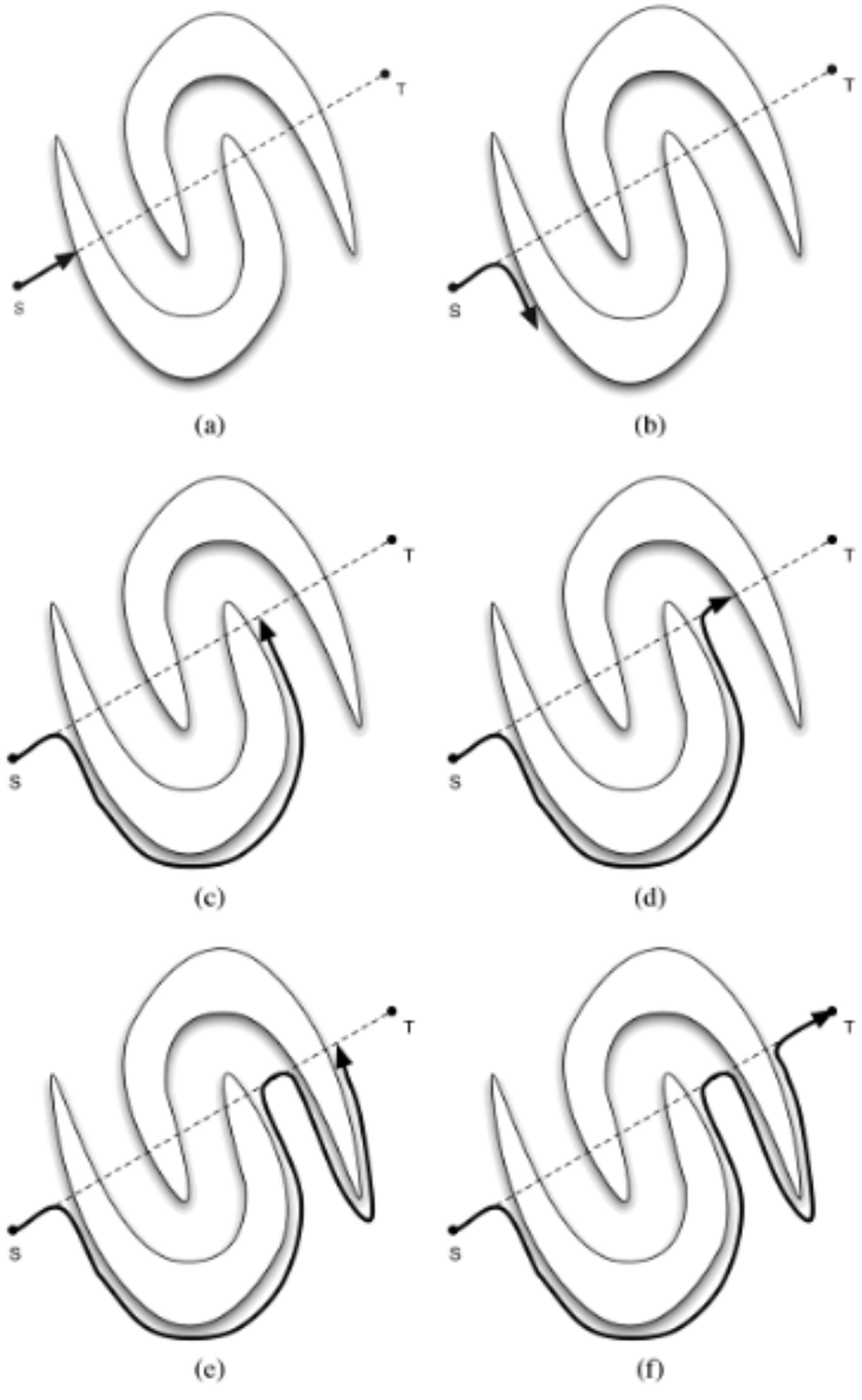
زمستان ۱۴۰۰ – بهار ۱۴۰۱

[slides adapted from Gianni Di Caro, @CMU with permission]

Behaviors for moving to a point in presence of obstacles

Bug 1 and Bug 2
Insect-inspired Algorithms (1987)

- **Local decisions, map-free!**
- **Complete algorithms**: find a path if it exists, otherwise returns with a failure.
- **No guarantees** in terms of finding a good (short) path!
- "Nasty" obstacle structure and placement or "wrong" left/ right turning choices can make things going arbitrarily bad.



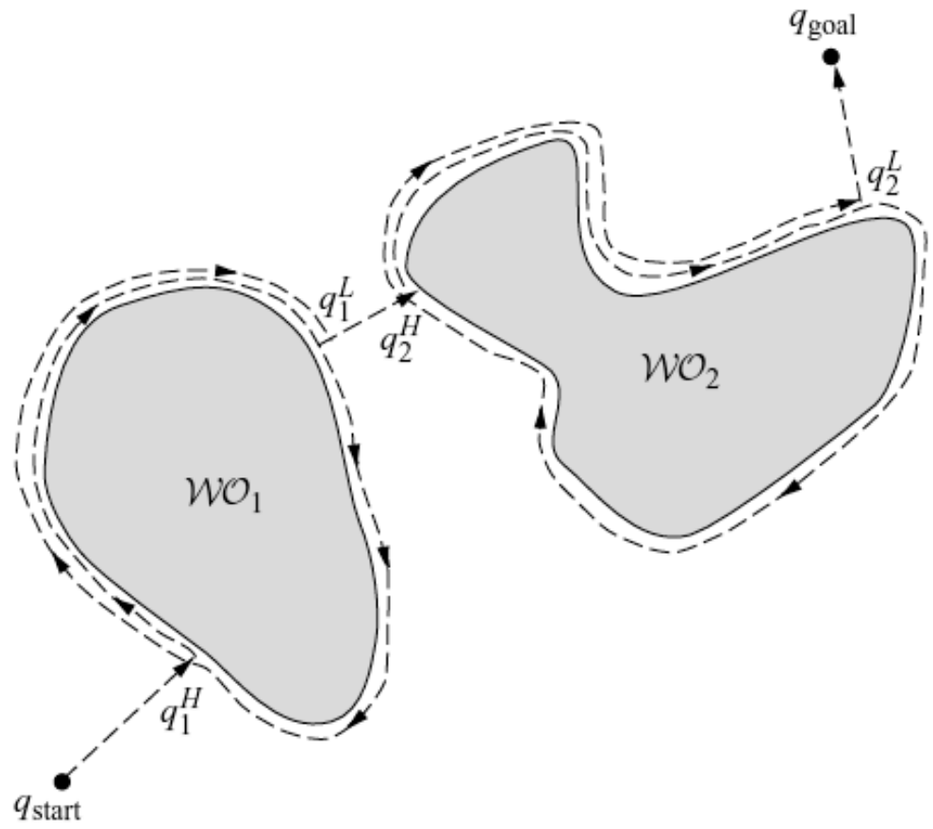
Bug 1: ~Exhaustive Search

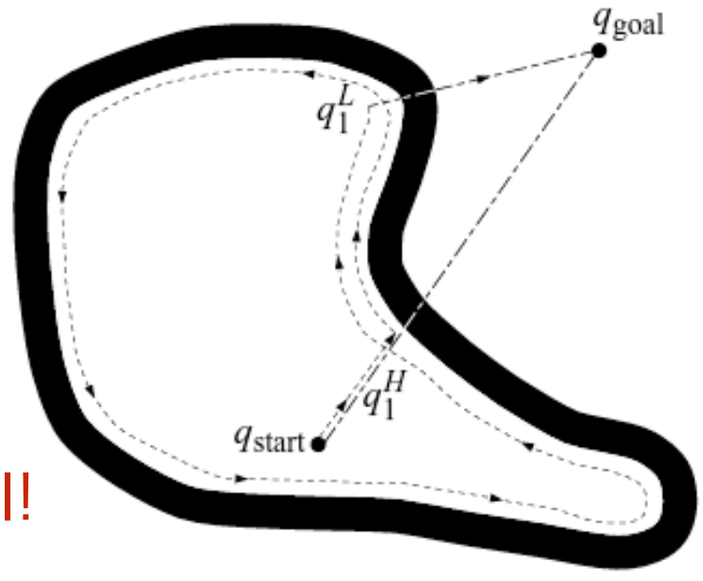
Algorithm 1 Bug1 Algorithm

Input: A point robot with a tactile sensor

Output: A path to the q_{goal} or a conclusion no such path exists

1: **while** Forever **do**
2: **repeat**
3: From q_{i-1}^L, move toward q_{goal}.
4: **until** q_{goal} is reached **or** an obstacle is encountered at q_i^H.
5: **if** Goal is reached **then**
6: Exit.
7: **end if**
8: **repeat**
9: Follow the obstacle boundary.
10: **until** q_{goal} is reached **or** q_i^H is re-encountered.
11: Determine the point q_i^L on the perimeter that has the shortest distance to the goal.
12: Go to q_i^L.
13: **if** the robot cannot move toward the goal **then**
14: Conclude q_{goal} is not reachable and exit.
15: **end if**
16: **end while**





Cannot move to goal!

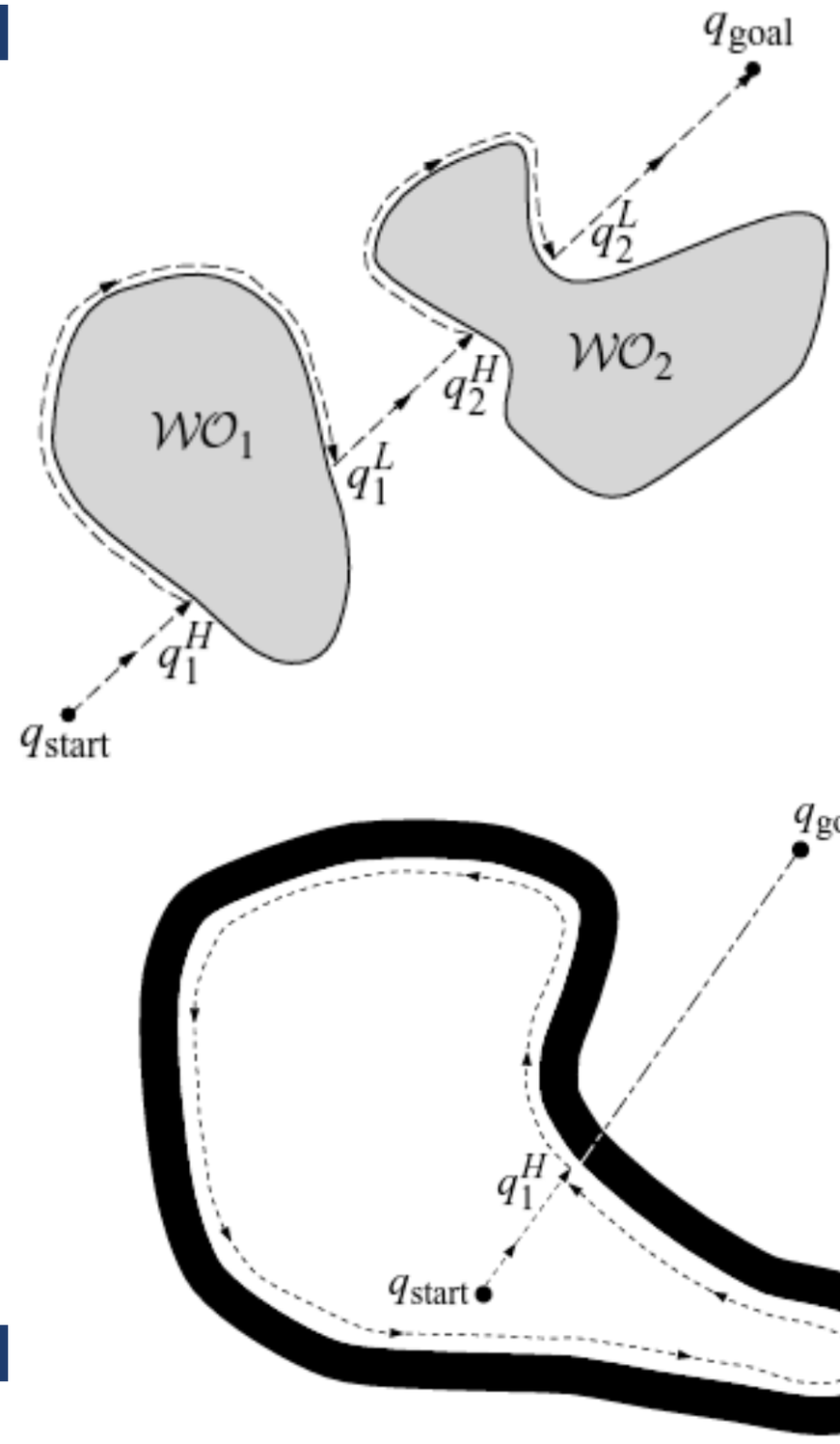
Bug 2: Greedy search

Algorithm 2 Bug2 Algorithm

Input: A point robot with a tactile sensor

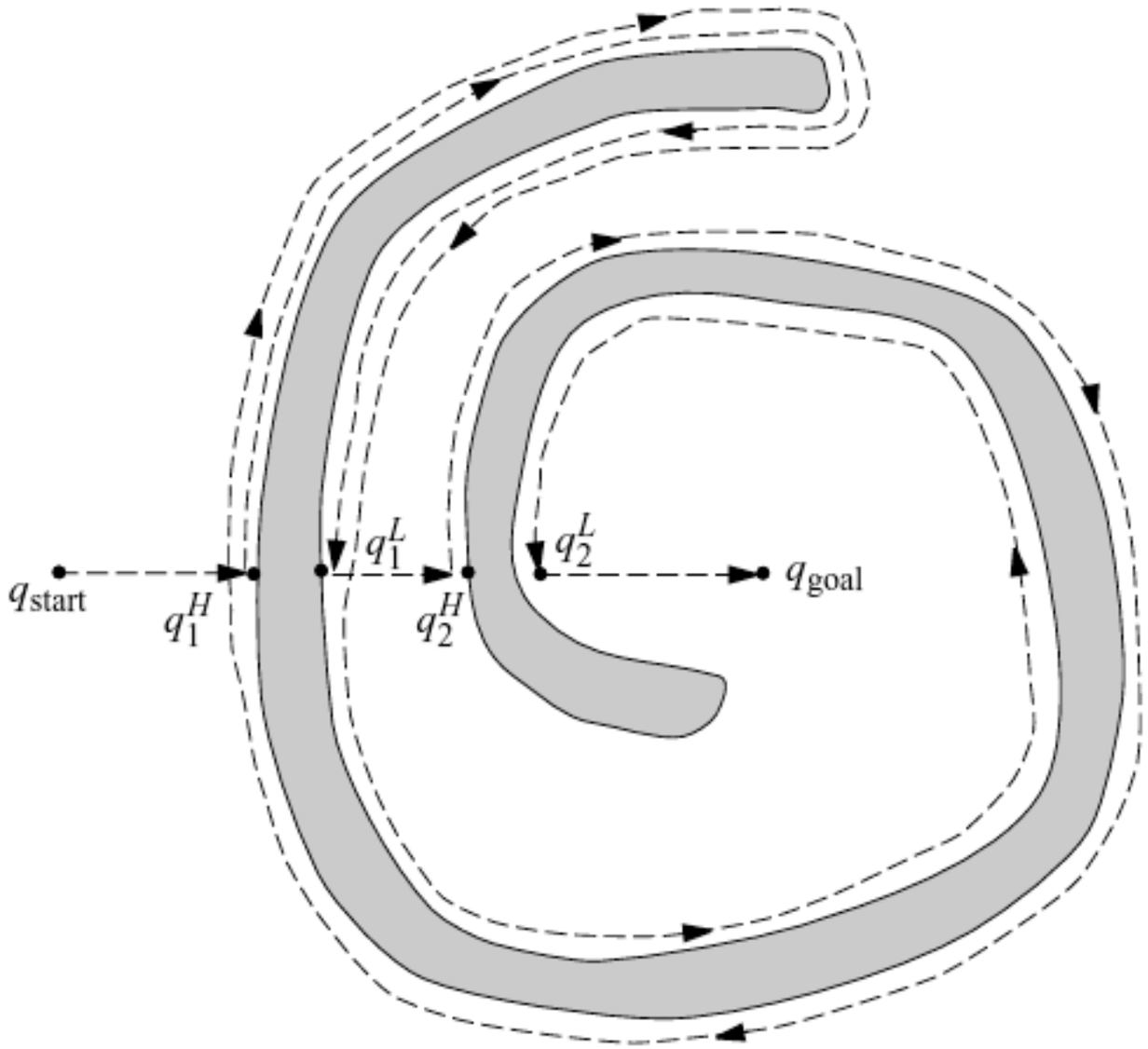
Output: A path to q_{goal} or a conclusion no such path exists

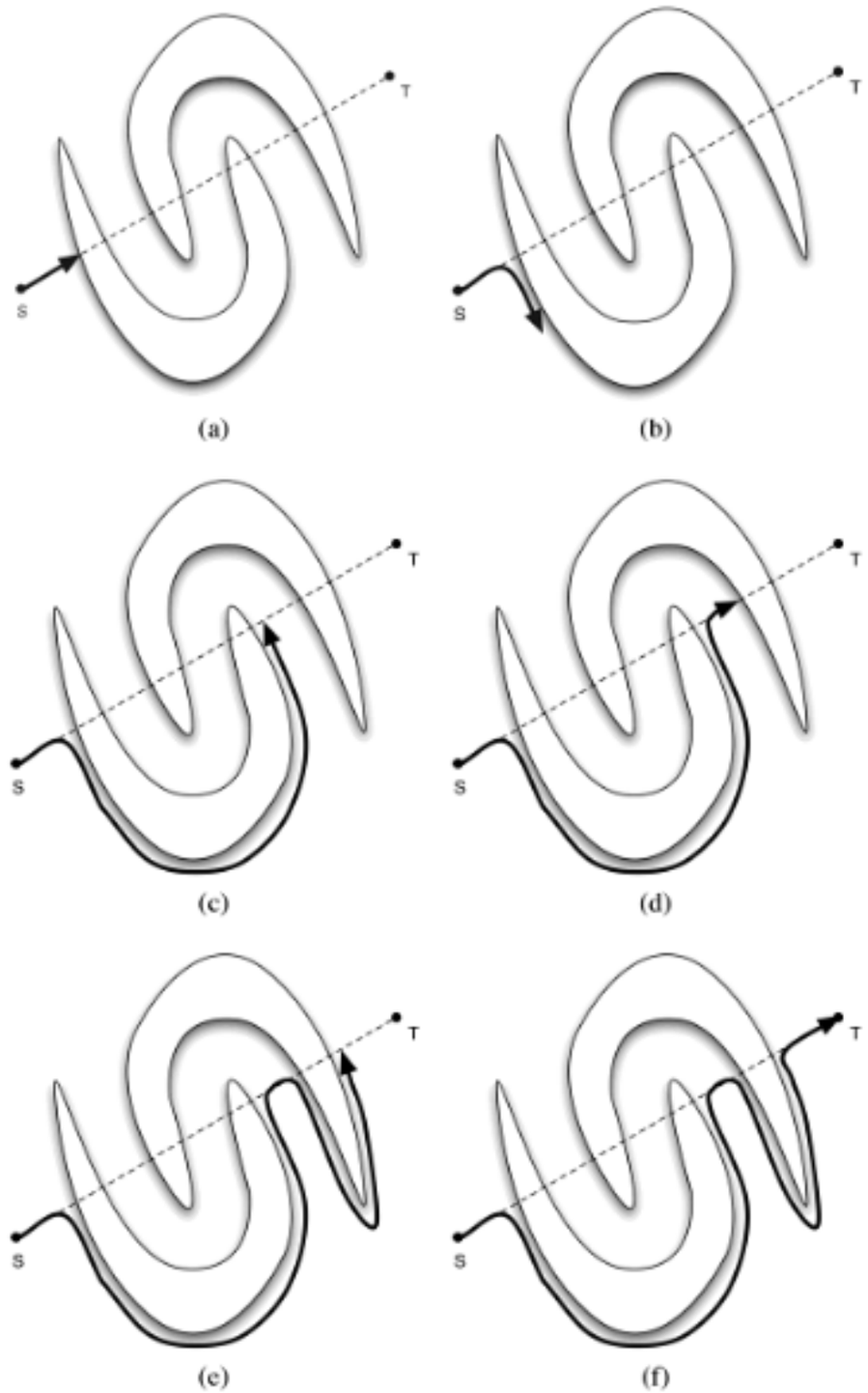
1: **while** True **do**
2: **repeat**
3: From q_{i-1}^L, move toward q_{goal} along m-line.
4: **until**
q_{goal} is reached **or**
an obstacle is encountered at *hit point* q_i^H.
5: Turn left (or right).
6: **repeat**
7: Follow boundary
8: **until**
9: q_{goal} is reached **or**
10: q_i^H is re-encountered **or**
11: m-line is re-encountered at a point m such that
12: $m \neq q_i^H$ (robot did not reach the hit point),
13: $d(m, q_{\text{goal}}) < d(m, q_i^H)$ (robot is closer), and
14: If robot moves toward goal, it would not hit the obstacle
15: Let $q_{i+1}^L = m$
16: Increment i
17: **end while**



Bug 2: going to kitchen!

Bug 2: issues

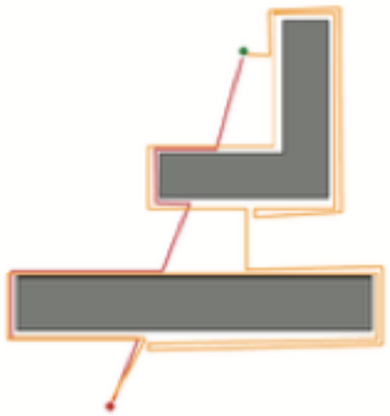




Bug 1 or Bug 2 ?

- At first sight it seems that Bug2 is better than Bug1 (does not involve a complete circumnavigation of obstacles)
- This is not always true, depends on the type of obstacles

Bug2 better than Bug1



Bug1 better than Bug2

Bug 1

- Exhaustive *leave-point* search
- Better on complex obstacles

Bug 2

- Opportunistic *leave-point* search
- Better on simple obstacles

(strong) Assumptions

- **Localization**: The robot must know where it is in the world, and must be able to maintain a (robust) estimate of this while it moves

- **Mapping**: Location of the goal is known within the robot's representation of the world.

- **Mapping/Localization**: The robot must be able to mark a location in its representation of the world and be able to determine that it has returned to it.

- **Sensing:** Obstacles need to be sensed with precision, in order to circumnavigate them.

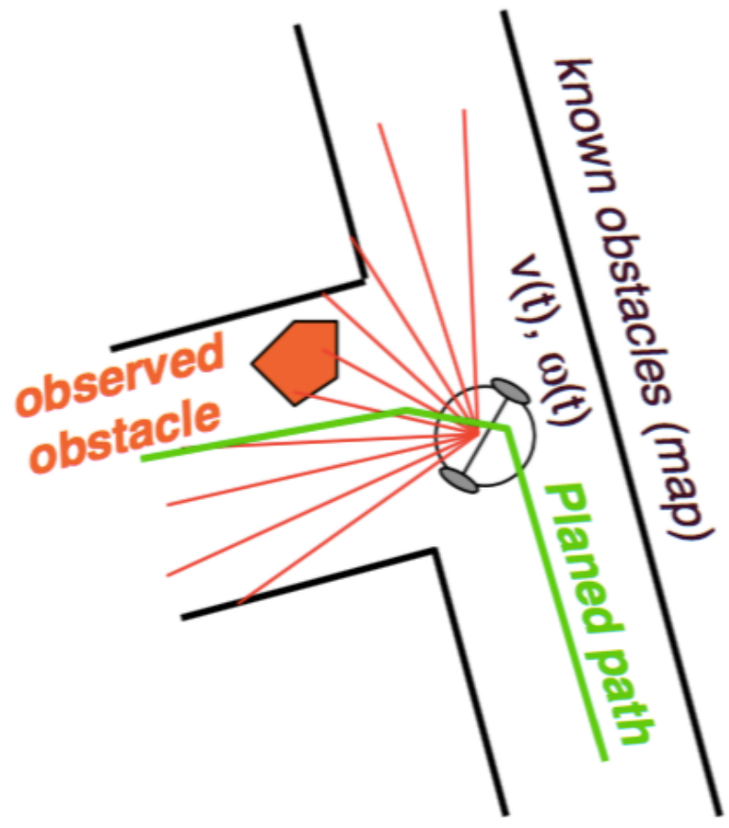
  - Obstacle detection and avoidance

- **Actuation/Control:** The robot needs to be able to smoothly change and adapt its trajectory ... velocities?

Obstacle avoidance

The goal of an obstacle avoidance algorithm is to **avoid collisions** with obstacles

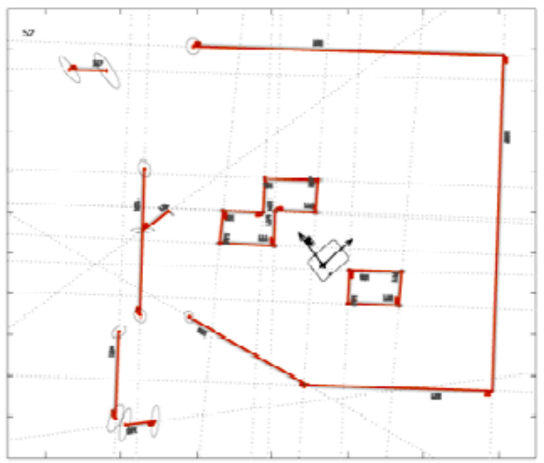
- It is usually based on a **local map**
- Implemented as an independent task / ROS node
- Efficient OA should (try to) be optimal wrt:
  - ✓ the overall goal
  - ✓ the actual speed and kinematics of the robot
  - ✓ the on board sensors
  - ✓ the actual and future risk of collision



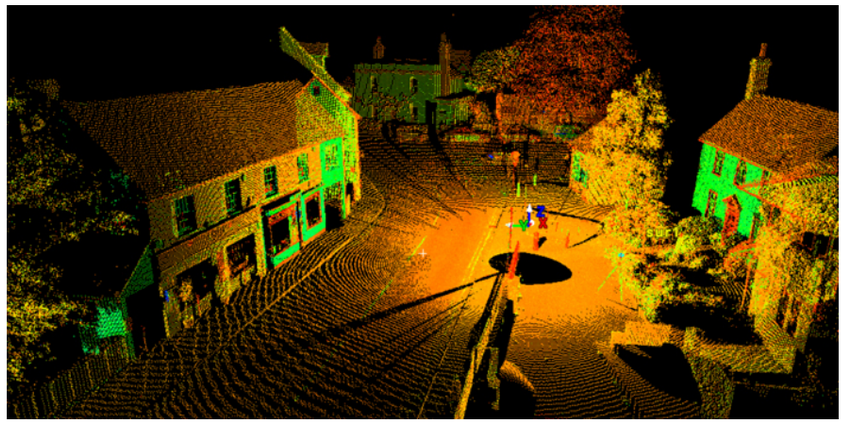
- Given map *and/or*
- Map built using sensor (range sensors)

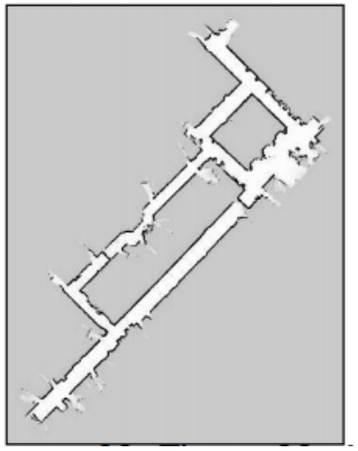
How to build a (local) map (for OA)?

- Metric and/or topological representations of the environment

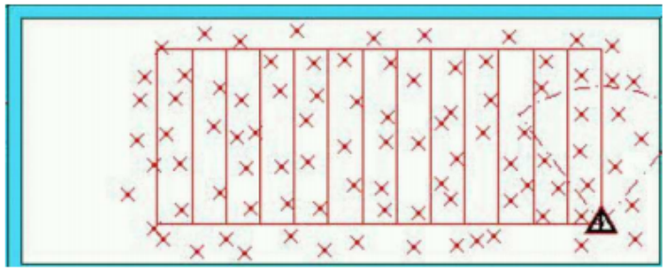


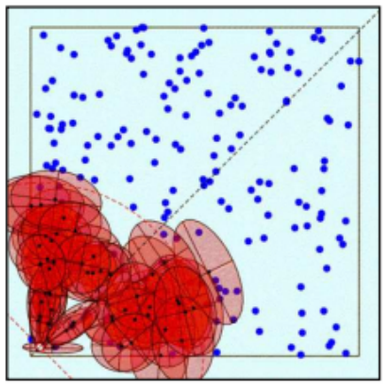
- Grid-based, 2D-3D scan





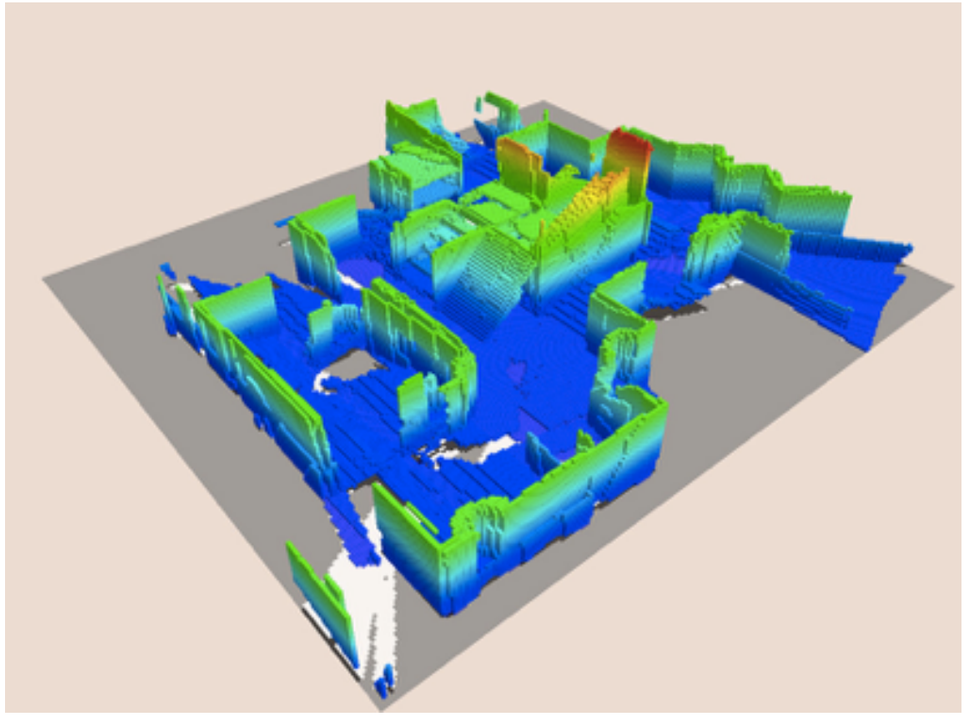
- Landmark-based





MAP: given or constructed?

Hand-made / Made using external tools



Made by the robot using exteroceptive sensors

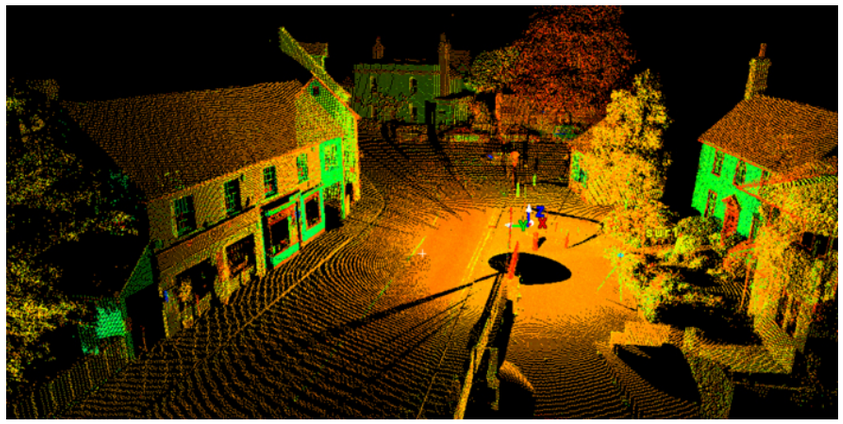
A map classification

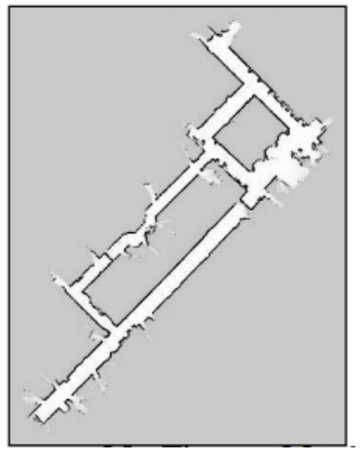
- A map M of the environment is a list of objects in the environment and along with their properties:

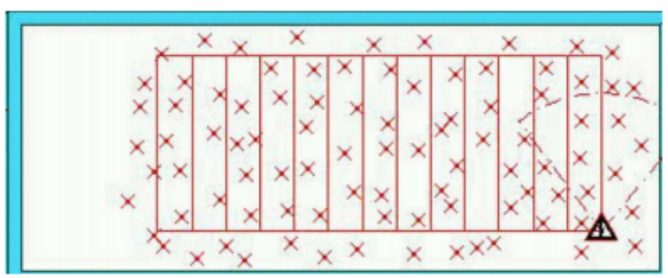
$$M = \{m_1, m_2, \ldots, m_N\}$$

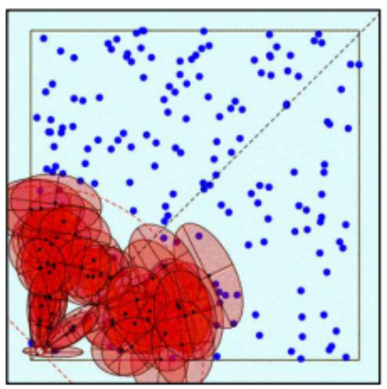
where N is total number of objects used to represent the environment, and each $m_i, i =, 1 \ldots, N$, specifies a *property* that characterizes the i-th map object.

- Maps are usually *indexed* in one of two ways:







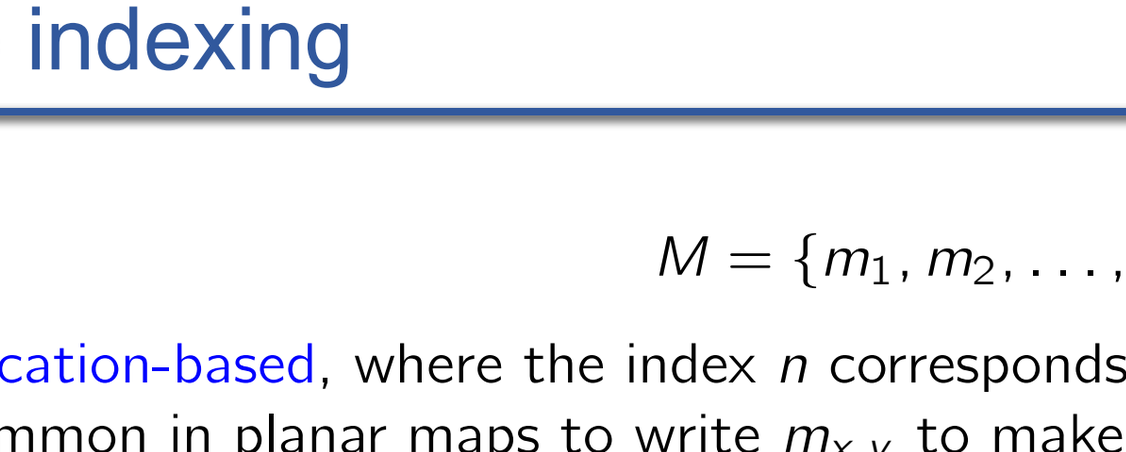


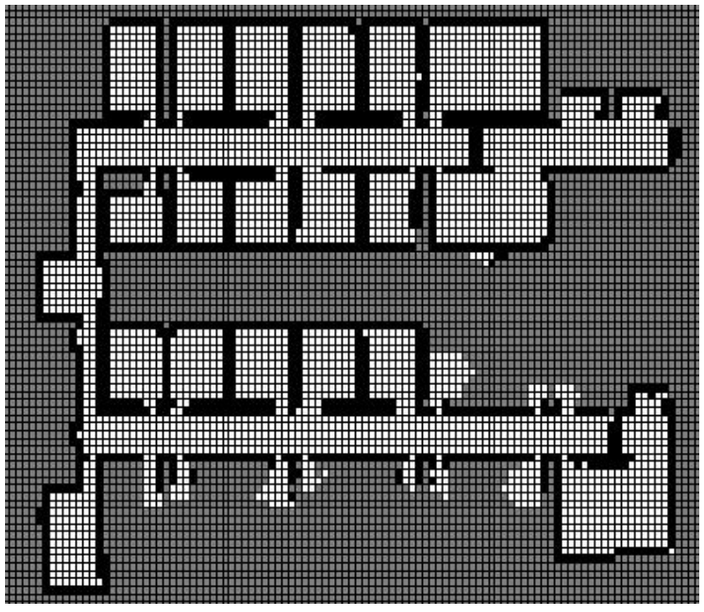
Map indexing

$$M = \{m_1, m_2, \ldots, m_N\}$$

- Location-based, where the index n corresponds to a specific location, such that it is common in planar maps to write $m_{x,y}$ to make explicit that m is the property of a specific world coordinate (x, y)

  Location-based maps are *volumetric*, in that they offer a label for **any location in the world**: they contain information not only about objects in the environment, but also about the absence of objects at each world location



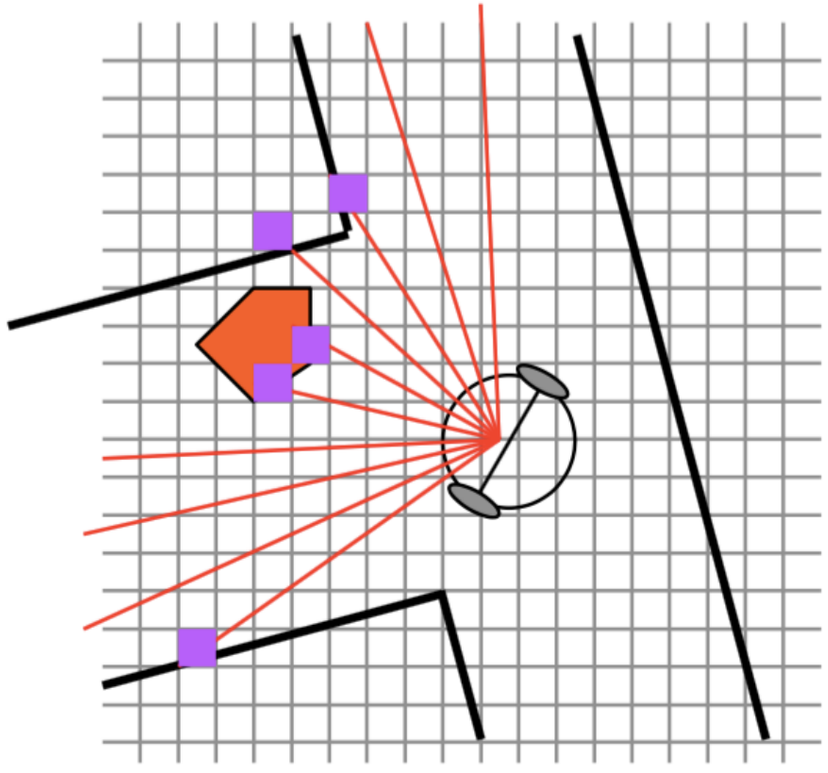


- Feature-based, where i is a generic feature index, such that the value of m_i contains, next to the properties of a feature (e.g., the color), the Cartesian location of the feature.

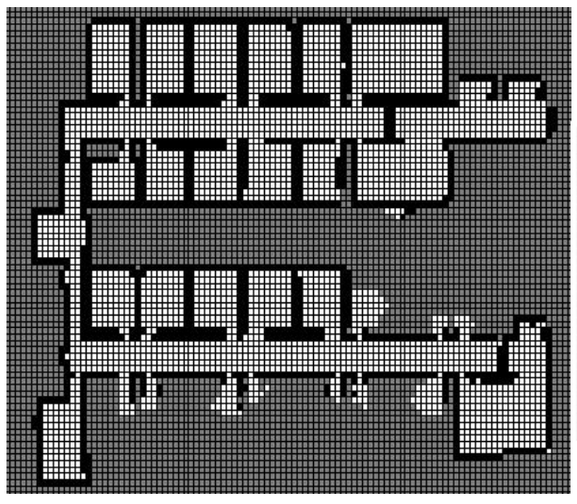
  Feature-based maps only specify the characteristics of the environment at **the specific locations that are indexed in the map**.

Local map for OA?

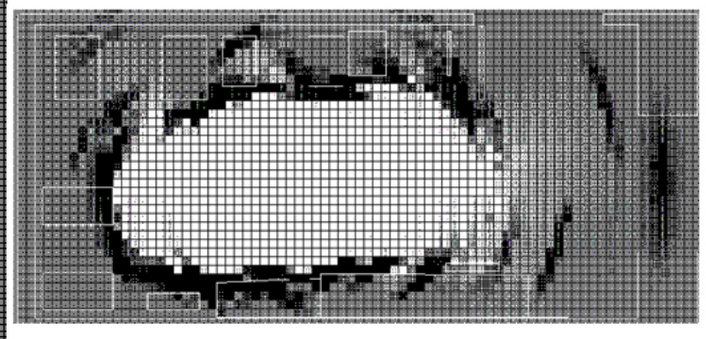
- **Occupancy grids** are a good choice!
- This grid is populated by relatively recent sensor data
- Grid cell values are equivalent to the probability that there is an obstacle in that cell



Local



Global



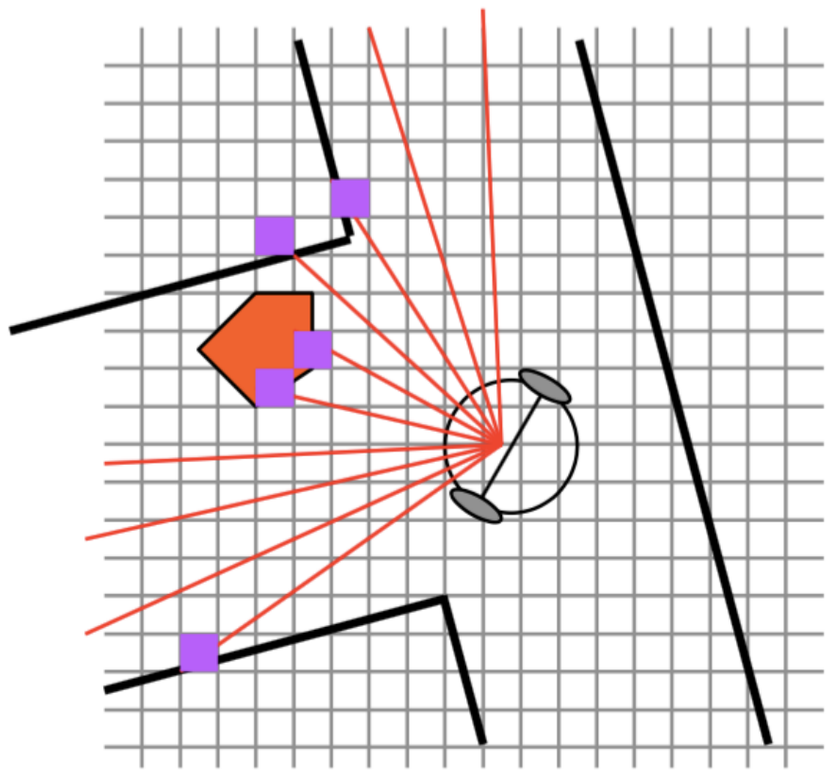
(b)

Occupancy grids

- The more classical (and used, in robotics) *location-based* map representation is known as occupancy grid map: the (continuous) space is partitioned into finitely many grid cells m_i, whose union covers the considered space:

$$M = \sum_{i=1}^{N} m_i$$

each m_i corresponds to a grid cell and has attached a binary occupancy value which specifies whether the cell is occupied or free.



Occupancy grids: probabilistic updates

The gold standard of any *occupancy grid mapping algorithm* is to calculate, at any time k, the **posterior over the map given the observation data**:

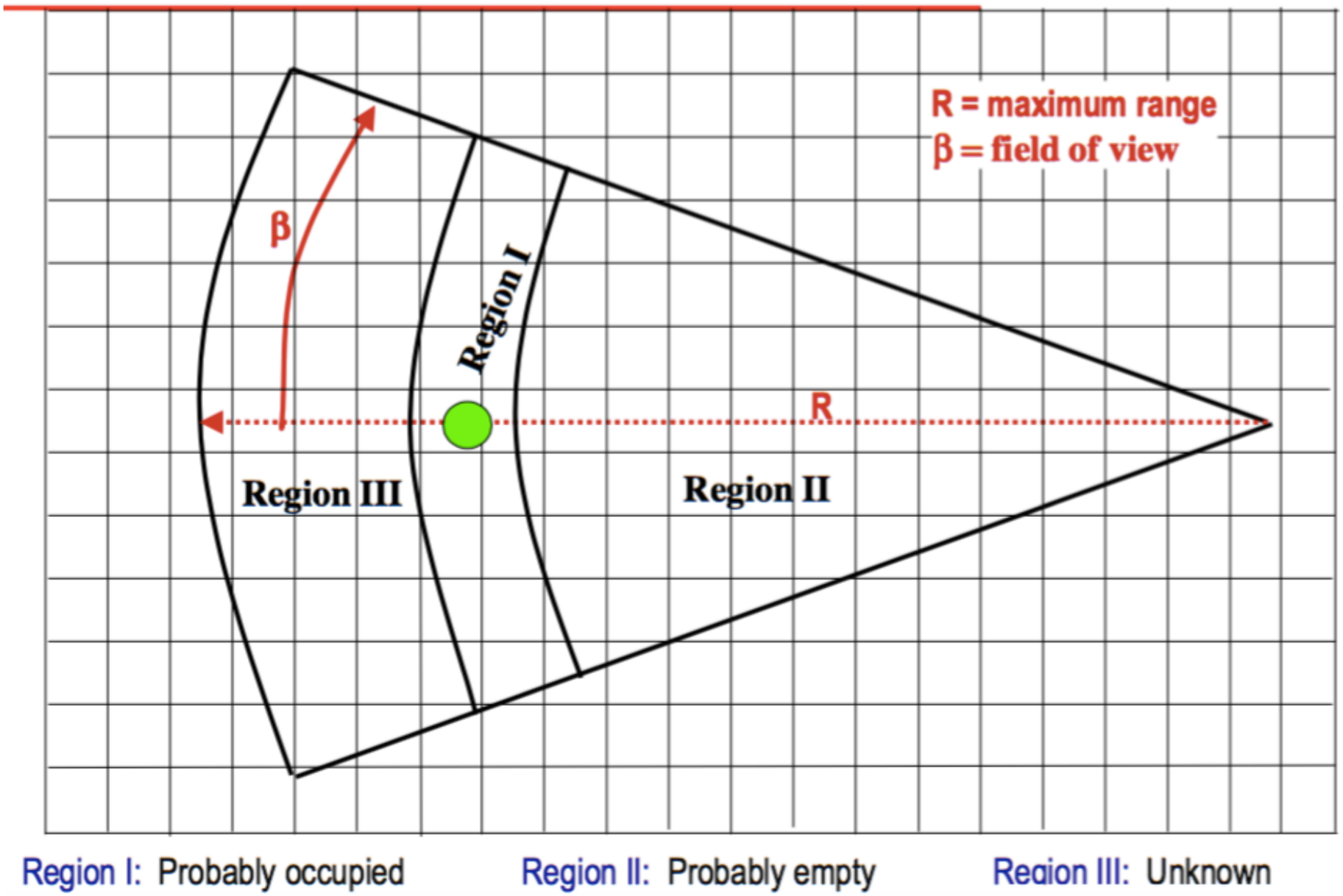
$$p(M \mid z_{1:k}, \boldsymbol{\xi}_{1:k})$$

,
which means estimating the joint probability that each cell is occupied or free.

How to calculate the **joint probability** p *(over multiple observations)* that a cell is occupied?

- Need **sensor model** to deal with uncertainty
- Let's look at the approach for *sonars* ...

Sonar sensor model



- If a sensor reading returns an obstacle at a distance *d*, what does it mean?
- We need to use the sensor model to define a probability value for each cell being either ***occupied*** or ***empty***

Bayesian approach (one evidential method)

- Goal: Convert sensor readings into probabilities

- Combine probabilities using ***Bayes' rule***:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(A \mid B)P(B)}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Normalizing constant}}$$

Basic probability theory

- Probability function:
  - *Gives values from 0 to 1 indicating whether a particular event, H (Hypothesis), has occurred*

- For sonar sensing:
  - *Experiment: Sending out acoustic wave and measuring time of flight*
  - *Outcome: Range reading reporting whether the region being sensed is Occupied or Empty*

- Hypotheses (H) = {Occupied, Empty)

- Probability that H has really occurred:

  $0 < P(H) < 1$

- Probability that H has not occurred:

  $1 - P(H)$

Unconditional and conditional probabilities

- Unconditional probability: $P(H)$
  - *"Probability of* H"
  - *Only provides* a priori *information*
  - *For example, could give the known distribution of rocks in the environment, e.g., "x% of environment is covered by rocks"*
  - *For robotics, unconditional probabilities are not based on sensor readings*

- For robotics, we want: Conditional probability: $P(H \mid s)$
  - *"Probability of* H, *given* s" *(e.g., P(Occupied | s), or P(Empty | s))*
  - *These are based on sensor readings,* s

- Note: $P(H \mid s) + P(\text{not } H \mid s) = 1.0$

$$P(H \mid s) = \frac{P(s \mid H)P(H)}{P(s \mid H)P(H) + P(s \mid notH)P(notH)}$$

Probabilities for occupancy grids

- For each grid element, grid[i][j], store tuple of the two probabilities:

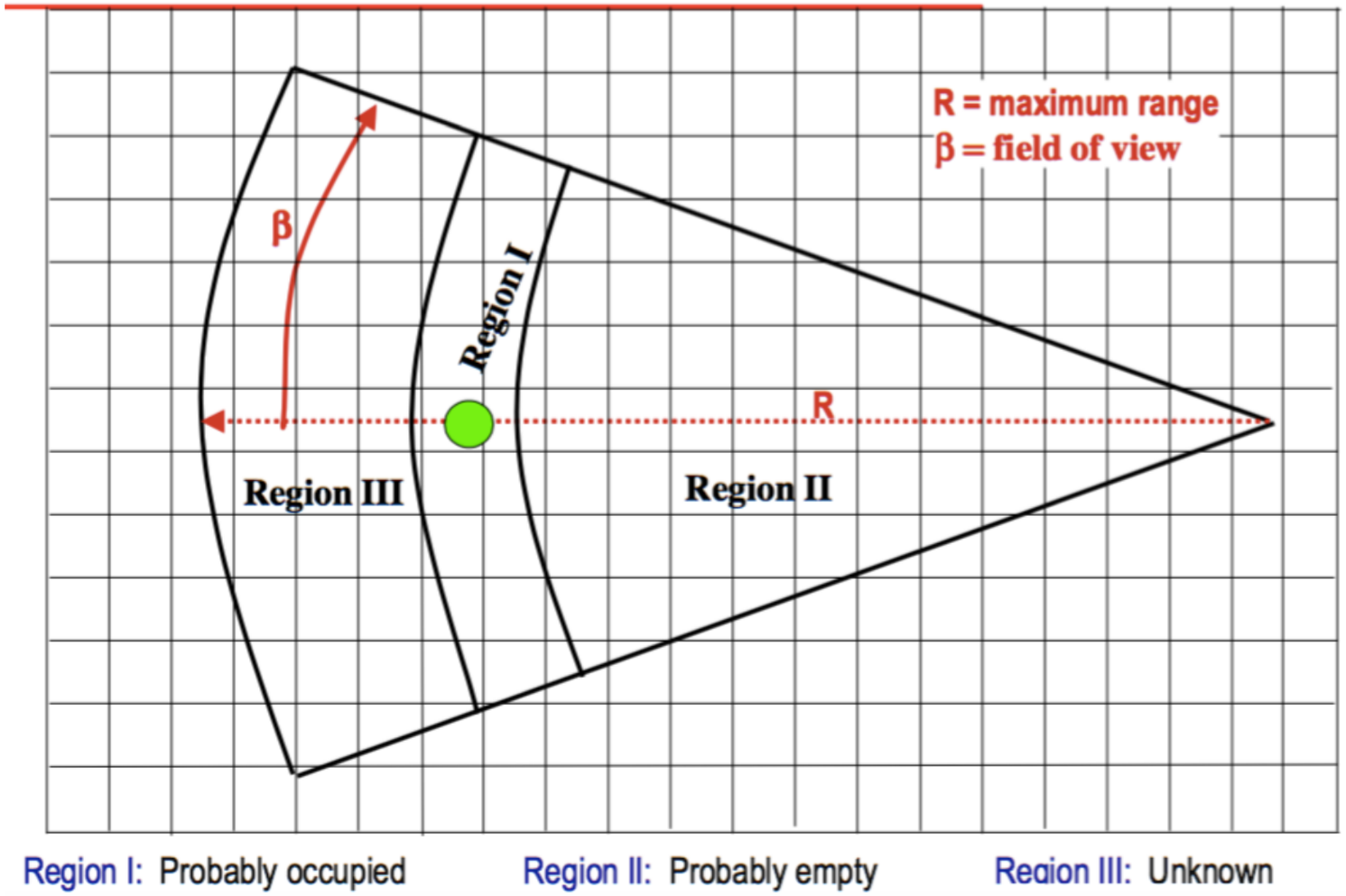
  Compute: P(Occupied | s) and P(Empty | s)

- For each grid element, grid[i][j], store tuple of the two probabilities:

```
typedef struct {
    double occupied;  // i.e., P(occupied | s)
    double empty;     // i.e., P(empty | s)
  } P;

P occupancy_grid[ROWS][COLUMNS];
```

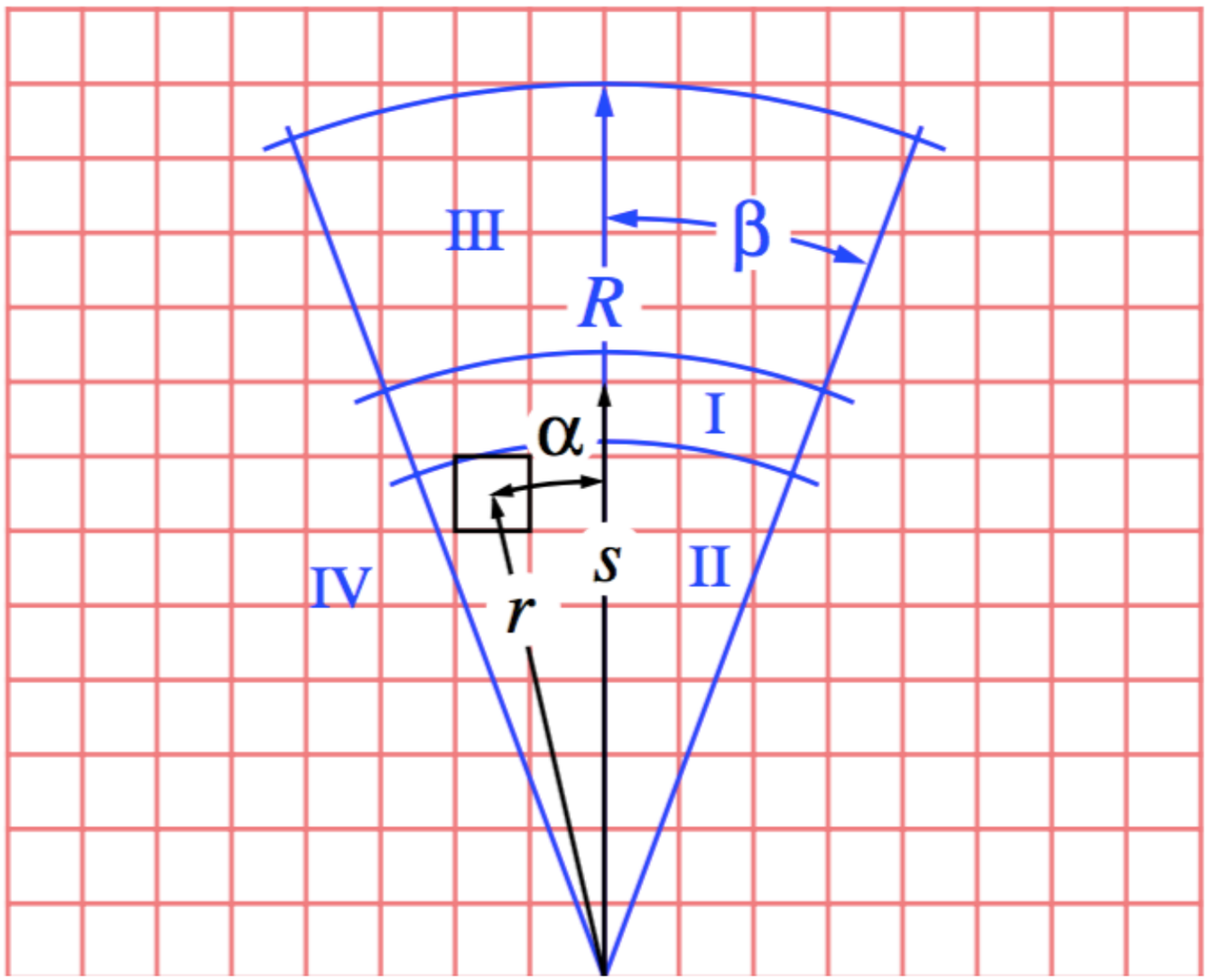
Using sensor model to get P(s|H)



P(*s*|H): given the sensor model, what is the probability of getting reading *s* ?

- Given the evidence that cell (x,y) belongs to Region I (probably occupied), what is the probability of a sensor reading *s*?
- Given the evidence that cell (x,y) belongs to Region II (probably empty), what is the probability of a sensor reading s?

Parameters for reasoning about a cell



Model for Region I

$$\Pr(C) = \frac{\left(\frac{R-r}{R}\right) + \left(\frac{\beta-\alpha}{\beta}\right)}{2} \times M$$

$$\Pr(\bar{C}) = 1 - \Pr(C)$$

Model for Region II

$$\Pr(C) = 1 - \Pr(\bar{C})$$

$$\Pr(C) = \frac{\left(\frac{R-r}{R}\right) + \left(\frac{\beta-\alpha}{\beta}\right)}{2}$$

Converting sonar readings to probabilities: Region I

Based on the sensor model, if a cell C belongs to Region I, P(Occupied) is the probability that the cell is occupied, that depends on its range r and bearing α

- **Region I:**

The nearer the grid element to the origin of the sonar beam, the higher the belief

The closer to the acoustic axis, the higher the belief

$$\text{P(Occupied)} = \frac{\frac{R-r}{R} + \frac{\beta-\alpha}{\beta}}{2} \; x\, Max_{occupied}$$

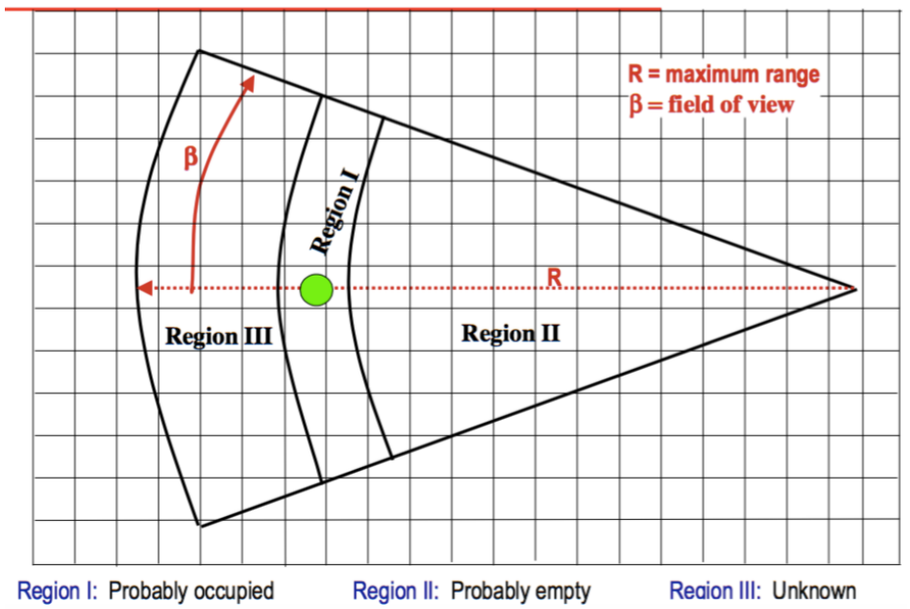
We never know with certainty

where r is distance to grid element that is being updated

α is angle to grid element that is being updated

$Max_{occupied}$ = highest probability possible (e.g., 0.98)

$$\text{P(Empty)} = 1.0 - \text{P(Occupied)}$$



Converting sonar readings to probabilities: Region II

Based on the sensor model, if a cell C belongs to Region I, $P(\text{Empty})$ is the probability that the cell is empty, that depends on its range r and bearing α

- **Region II:**

The nearer the grid element to the origin of the sonar beam, the higher the belief

The closer to the acoustic axis, the higher the belief

$$\text{P(Empty)} = \frac{\frac{R-r}{R} + \frac{\beta-\alpha}{\beta}}{2}$$

$$\text{P(Occupied)} = 1.0 - \text{P(Empty)}$$

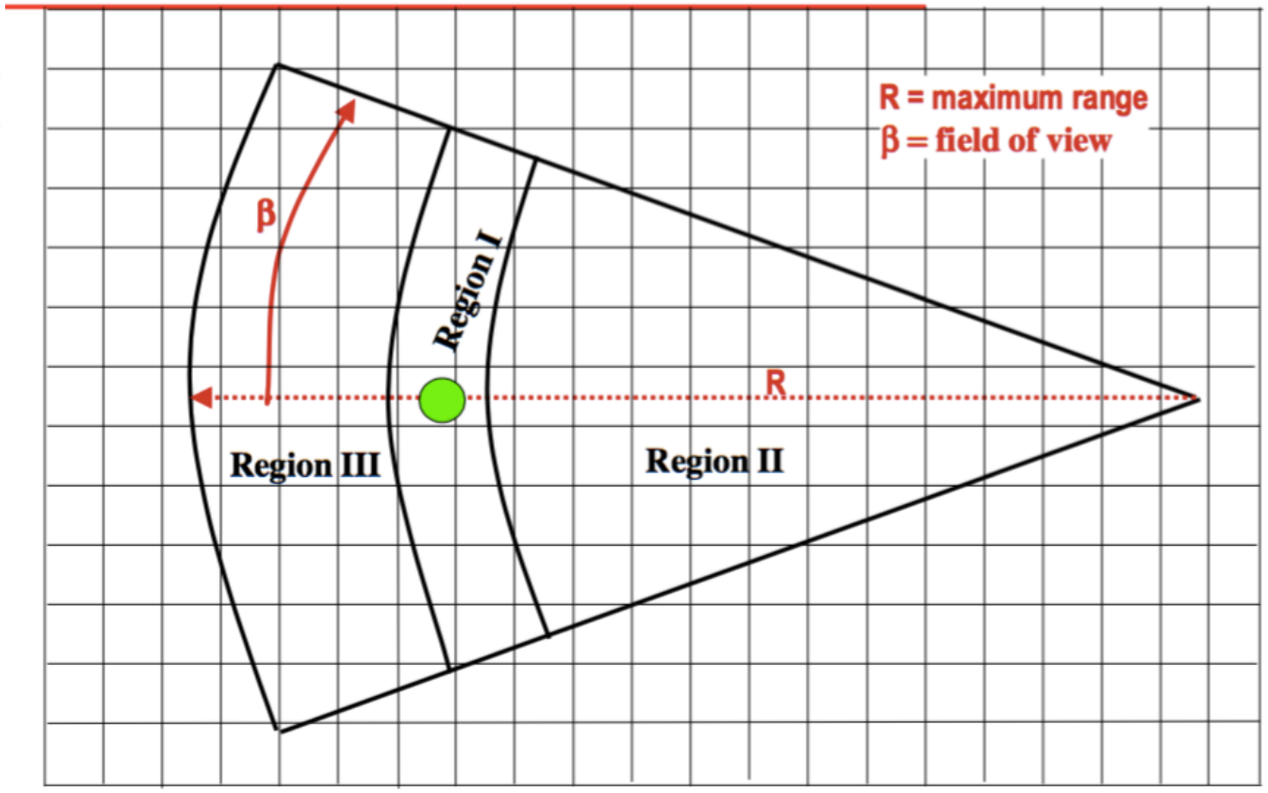
where r *is distance to grid element being updated,*
α is angle to grid element being updated

Note that here, we allow probability of being empty to equal 1.0

Sensor tolerance

- Sonar range readings have resolution error
- Thus, specific reading might actually indicate range of possible values
- E.g., reading of 0.87 meters actually means within (0.82, 0.92) meters
  - *Therefore, tolerance in this case is 0.05 meters.*

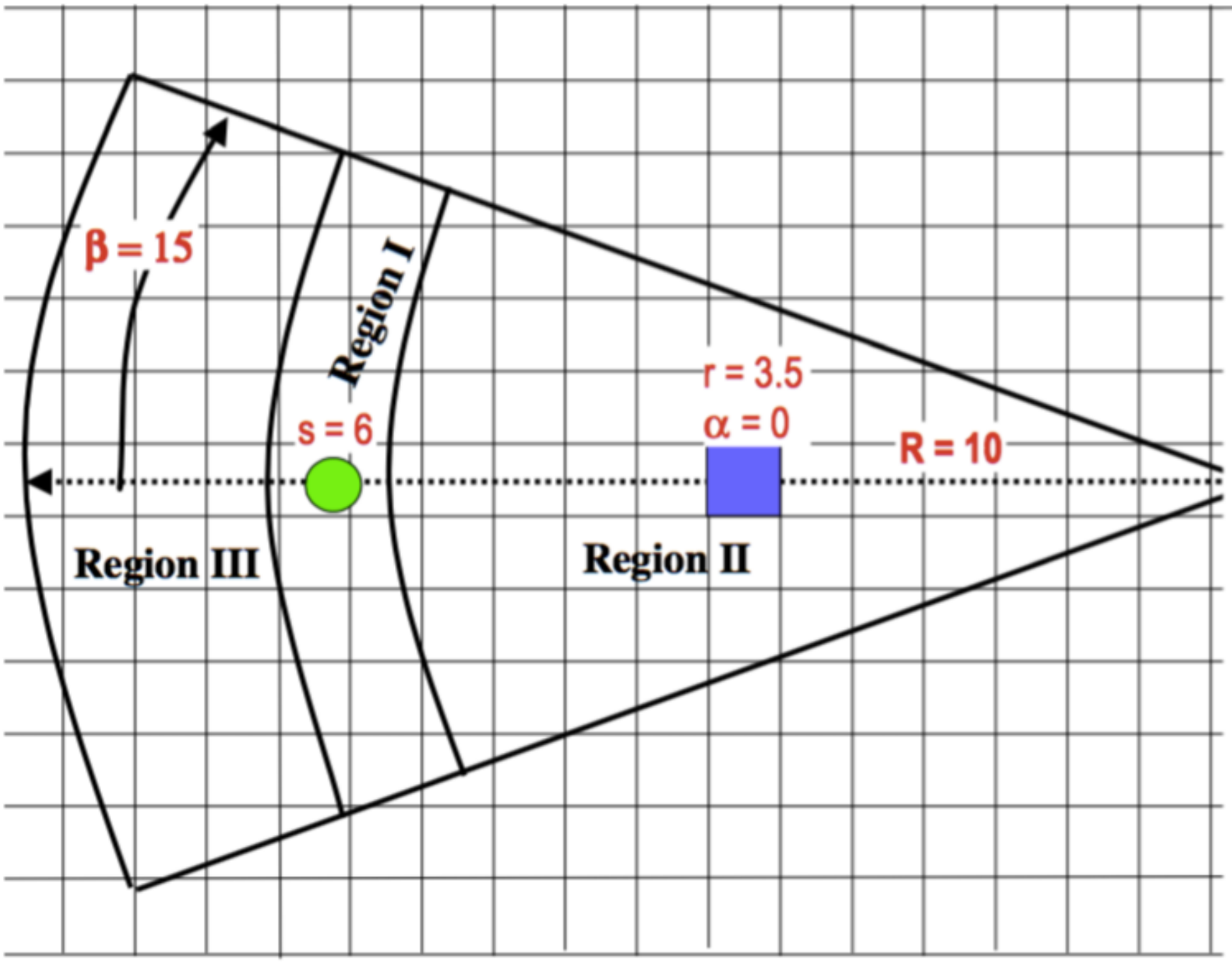
- Tolerance gives width of Region I



Region I: Probably occupied Region II: Probably empty Region III: Unknown

Value of a grid cell

Example: What is value of grid cell ■ ? (assume tolerance = 0.5)



Which region?

3.5 < (6.0 – 0.5) ➔ Region II

$$P(\text{Empty}) = \frac{\frac{10-3.5}{10} + \frac{15-0}{15}}{2}$$

$$= 0.83$$

P(Occupied) = (1 – 0.83) = 0.17

We have P(s | H) but we need P(H | s)

- Note that previous calculations were only based on the sensor model: the probability that a cell C is occupied or empty based on the sensor model, that is, based on its position in the model regions
- These probabilities provide $P(\text{s} \mid \text{H} = \{\text{C Occupied, C Empty}\})$
- In the model a cell C is identified by (r,α), therefore, given the evidence of a cell at $s = (r,\alpha)$ being either empty or occupied, previous probabilities correspond to the conditional probability of getting such a reading *s:* $P(s \mid H)$
- For instance, if $s = (r,\alpha) = (6,5)$ and we have the evidence that the cell is in Region I, the conditional probability $\text{P}(s|\text{H})$ of obtaining a reading s is:

$$P(s \mid H = \text{Occupied}) = \frac{\left(\frac{10-6}{10}\right) + \left(\frac{15-5}{15}\right)}{2} \cdot 0.98 = 0.52$$

We want to compute P(H | s)

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

$$P(H \mid s) = \frac{P(s \mid H)P(H)}{P(s \mid H)P(H) + P(s \mid notH)P(notH)}$$

$$P(H \mid s) = \frac{P(s \mid Empty)P(Empty)}{P(s \mid Empty)P(Empty) + P(s \mid Occupied)P(Occupied)}$$

- $P(s \mid Occupied)$ and $P(s \mid Empty)$ are known from sensor model
- $P(Occupied)$ and $P(Empty)$ are unconditional, prior probabilities (which may or may not be known)
  - *If not known, okay to assume* P(Occupied) = P(Empty) = *0.5*

Back to the example

- Let's assume we're on Mars, and we know that $P(Occupied) = 0.75$
- Continuing same example for cell  …

- $$P(Empty \mid s{=}6) = \frac{P(s \mid Empty)\, P(Empty)}{P(S \mid Empty)\, P(Empty) + P(s \mid Occupied)\, P(Occupied)}$$

$$= \frac{0.83 \times 0.25}{0.83 \times 0.25 + 0.17 \times 0.75}$$

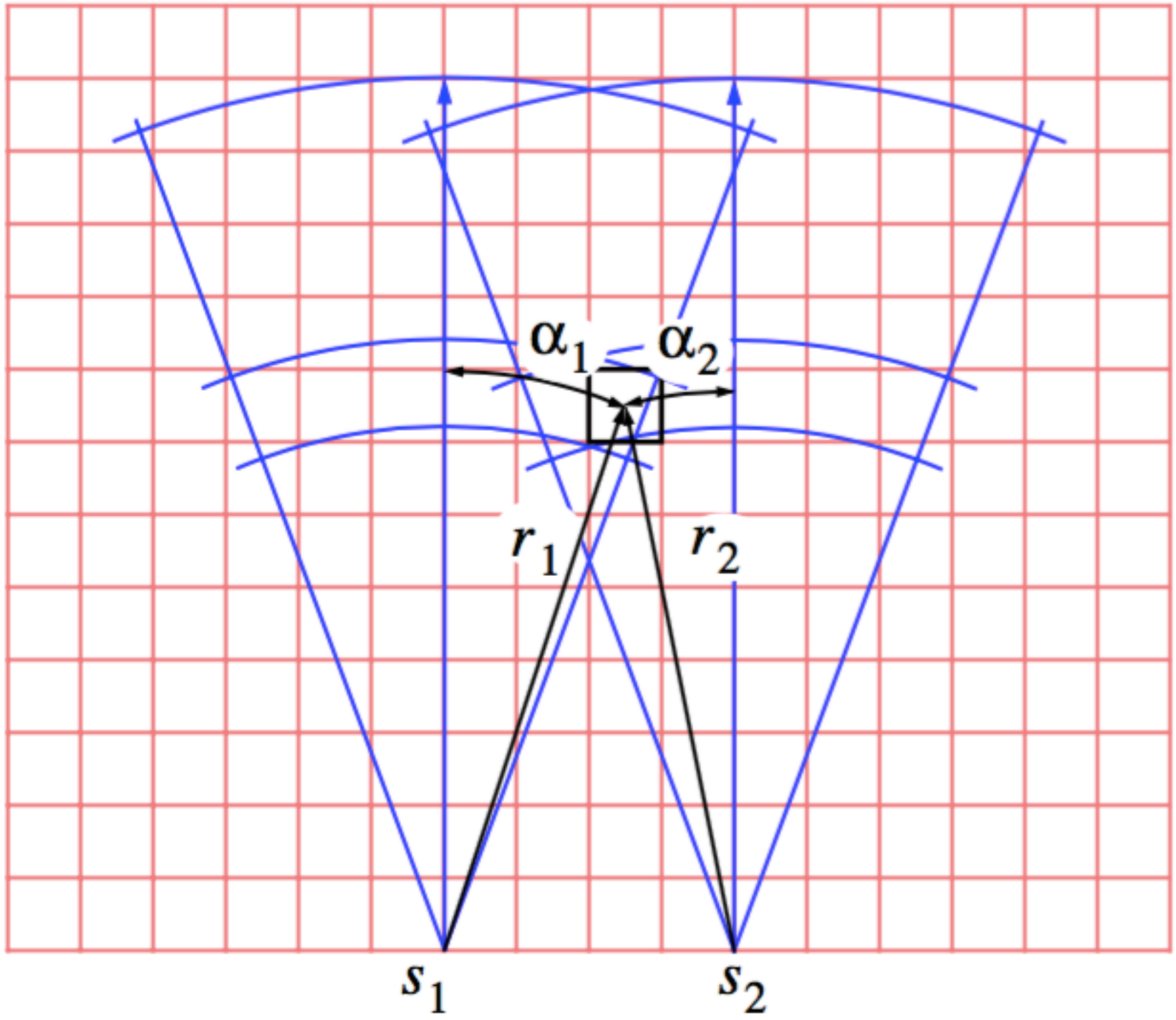
$$= 0.62$$

- $P(Occupied \mid s{=}6) = 1 - P(Empty \mid s{=}6) = 0.38$

- *These are the values we store in our grid cell representation*

Updating with bayes' rule: information fusion

- How to fuse multiple readings obtained over time?



Updating with Bayes' rule: Information fusion

- How to fuse multiple readings obtained over time?
- First time:
  - *Each element of grid initialized with* a priori *probability of being occupied or empty*
- Subsequently:
  - *Use Bayes' rule iteratively*
  - *Probability at time* t_{n-1} *becomes prior and is combined with current observation at* t_n *using recursive version of Bayes rule*:

$$P(H \mid s_n) = \frac{P(s_n \mid H)P(H \mid s_{n-1})}{P(s_n \mid H)P(H \mid s_{n-1}) + P(s_n \mid \neg H)P(\neg H \mid s_{n-1})}$$