16-311-Q Introduction to Robotics Fall'17

Lecture 17: Obstacle Avoidance, Local Maps

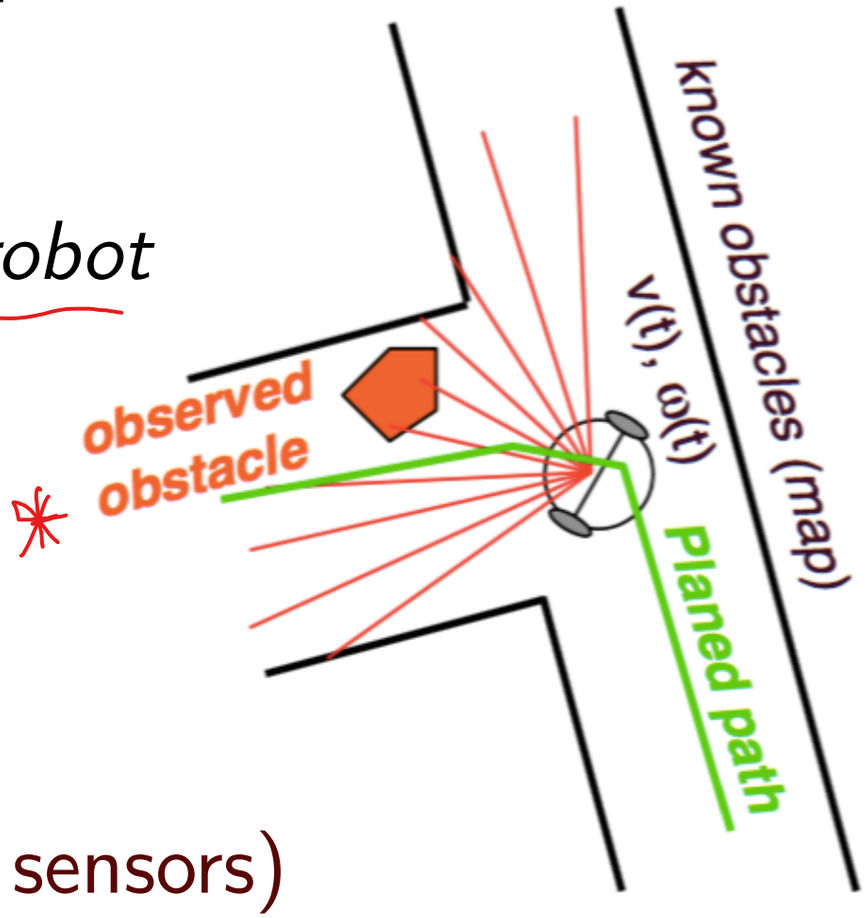
Instructor:
Gianni A. Di Caro

OBSTACLE AVOIDANCE

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

- It is usually based on *local map*
- Implemented as an ~*independent task*

- 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*

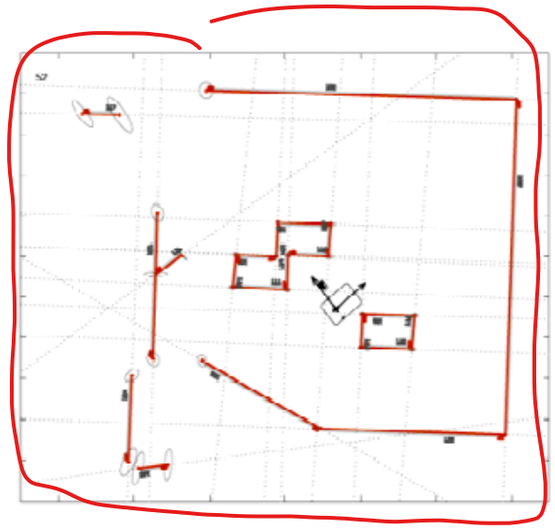


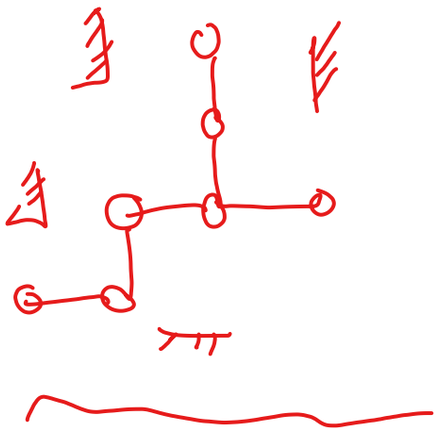
SLAM

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

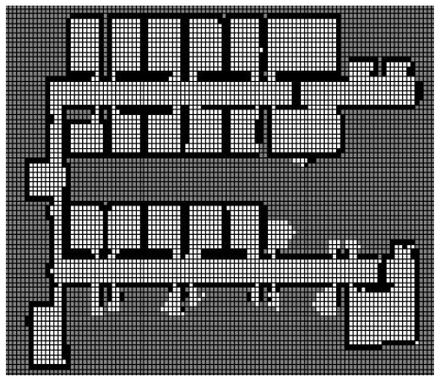
HOW TO BUILD A LOCAL MAP FOR OA?

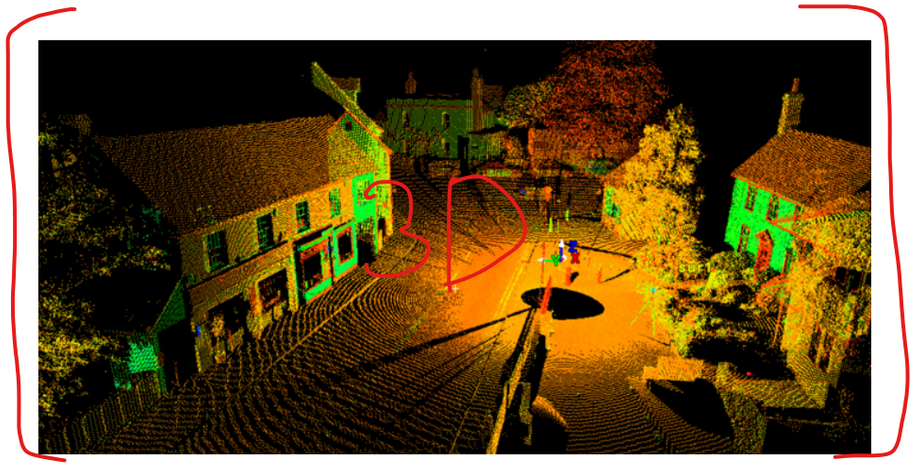
- Metric and/or topological representations of the environment

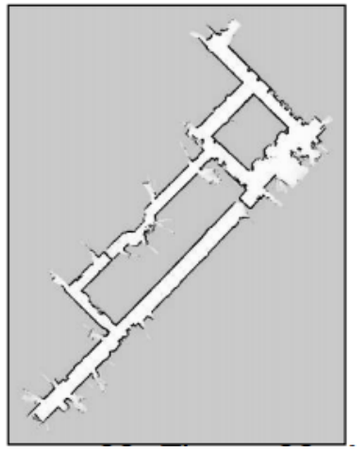


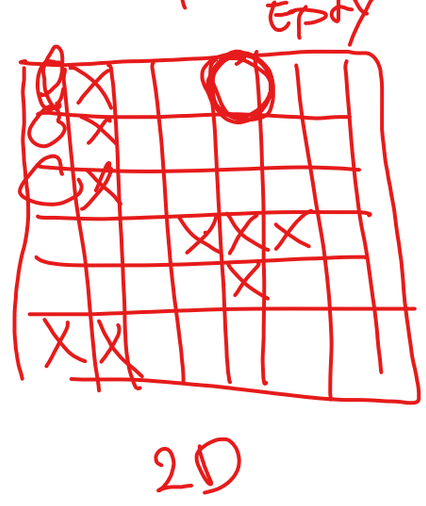


- Grid-based, 2D-3D scan

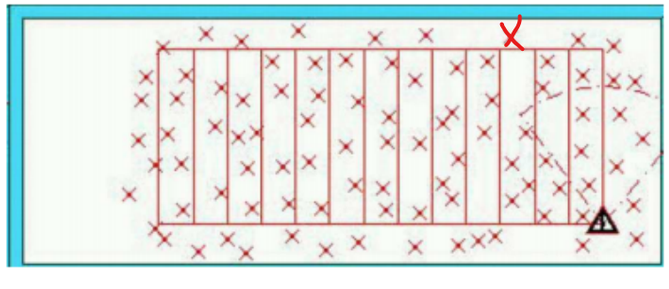


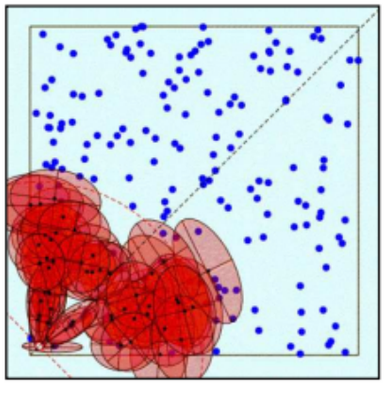






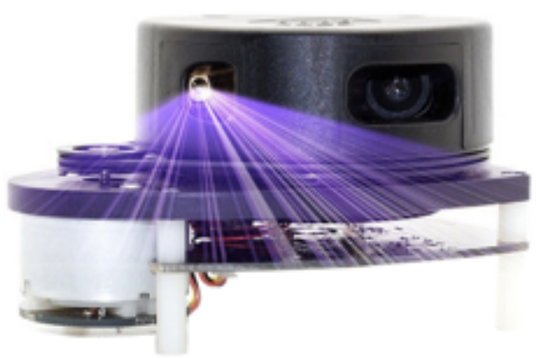
- Landmark-based

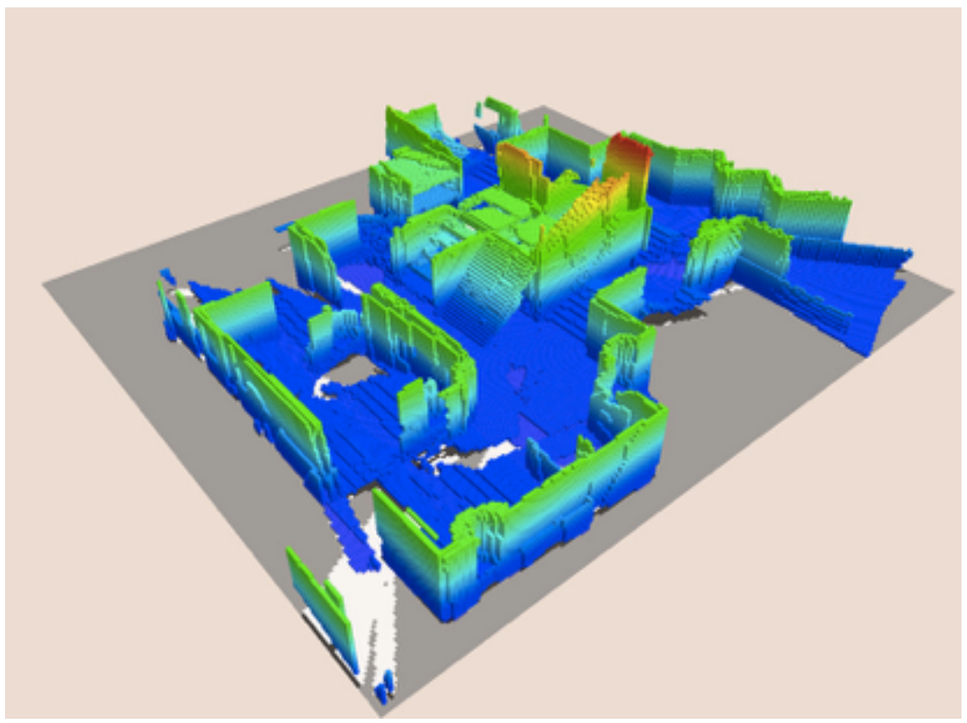




MAP: GIVEN OR CONSTRUCTED?

Hand-made / Made using external tools





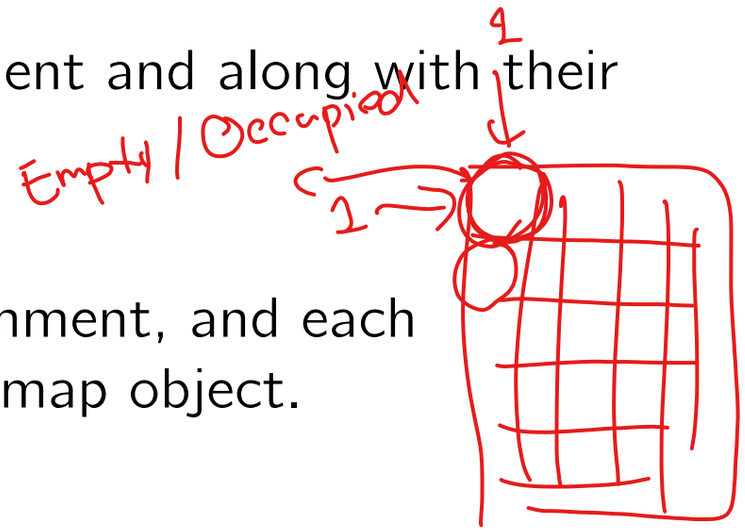
Made by the robot using exteroceptive sensors

A MAP CLASSIFICATION

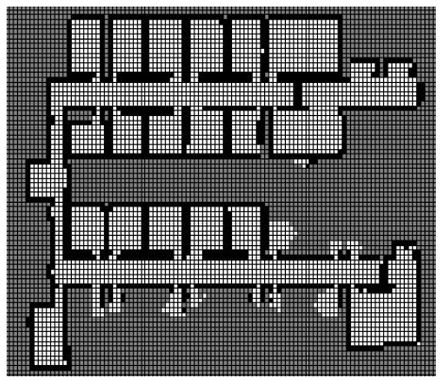
- A map M of the environment is a list of objects in the environment and along with their properties:

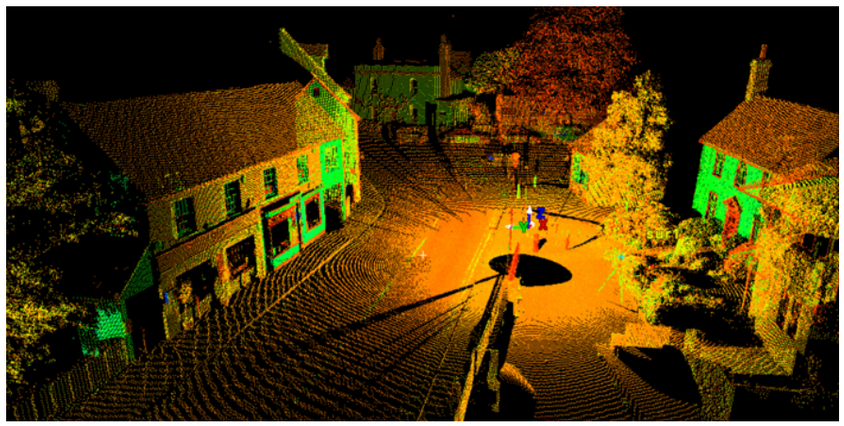
$$M = \{m_1, m_2, \ldots, m_N\}$$

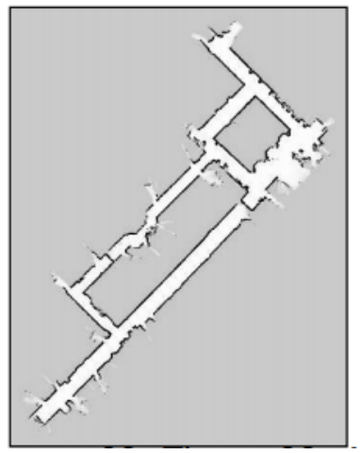
where N is total number of objects used to represent the environment, and each $m_i, i =, 1 \ldots, N$, specifies a *property* that characterizes the i-th map object.



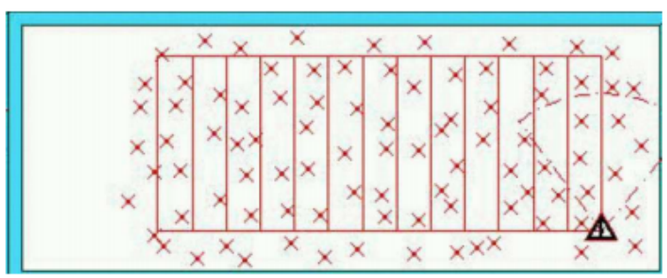
- Maps are usually *indexed* in one of two ways:

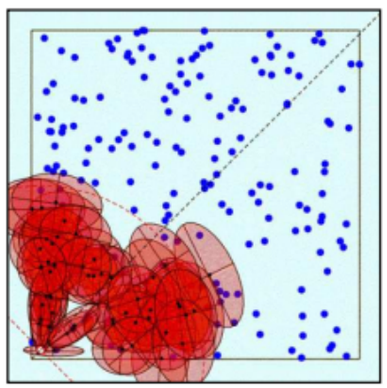






(id, location, type)



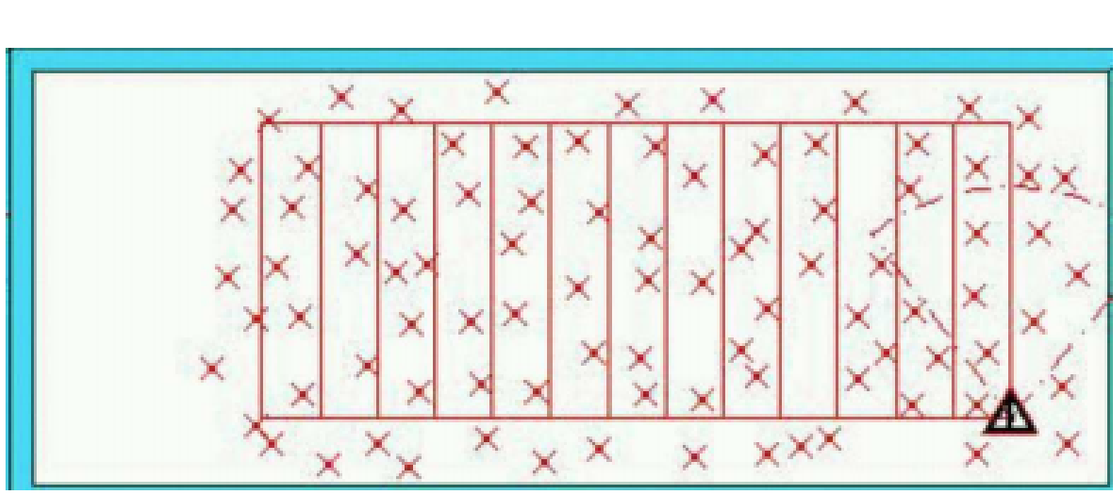


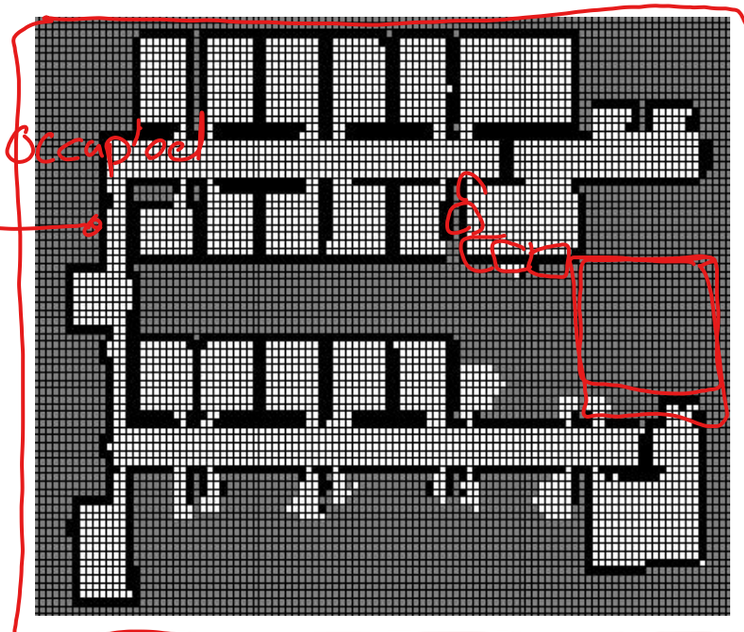
MAP INDEXING

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

- 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



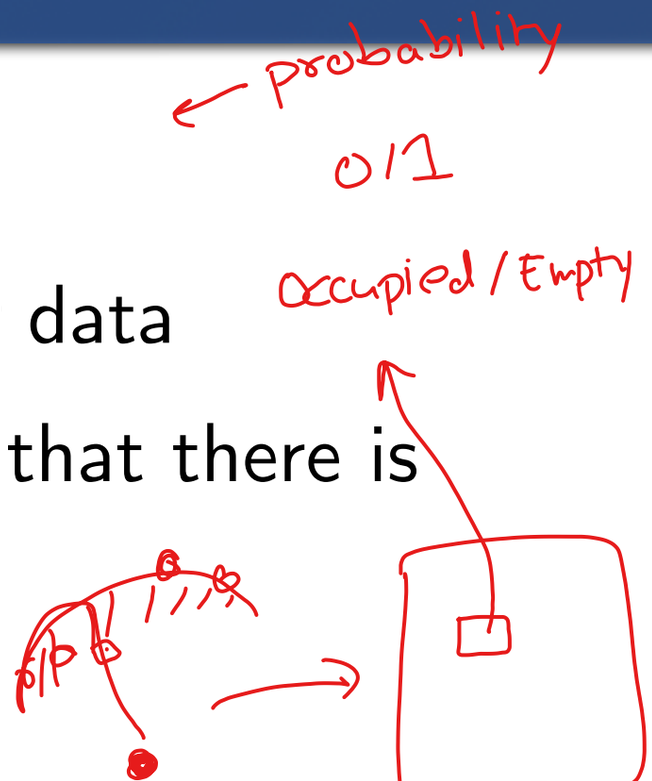


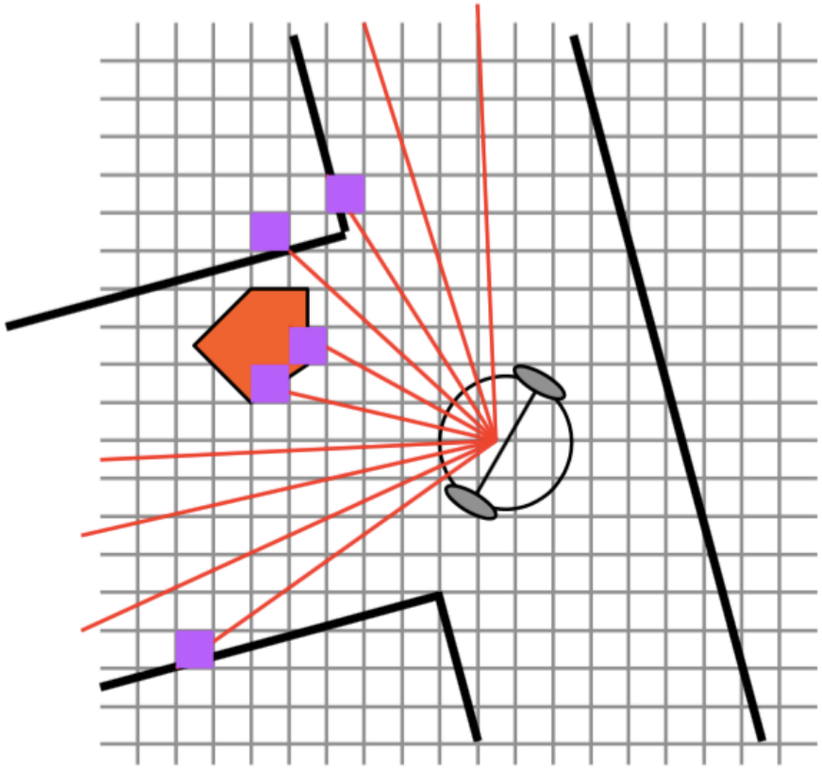
- Feature-based, where i is a generic feature index, such that the value of m_i contains, next to the properties of a feature (e.g., the color), the Cartesian location of the feature.

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

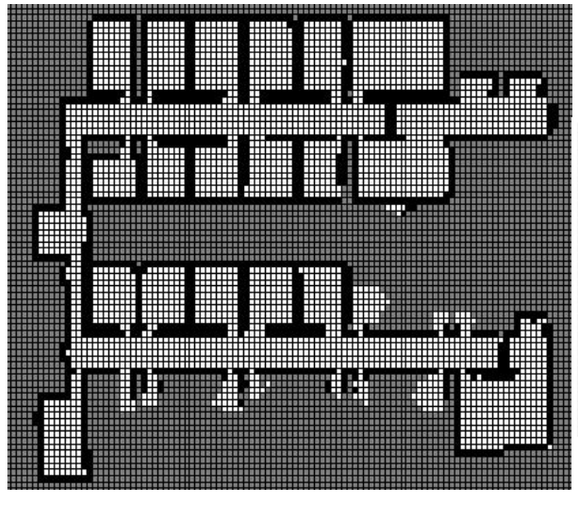
LOCAL MAP FOR OA?

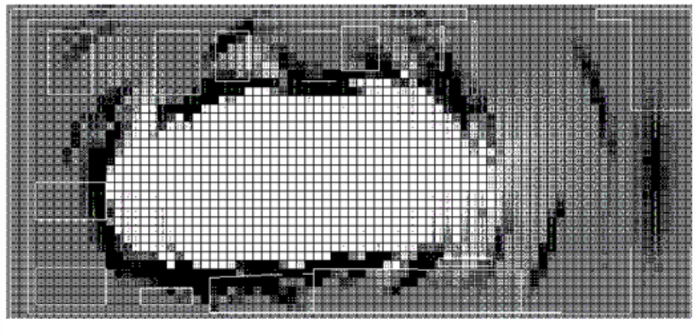
- **Occupancy grids** are a good choice!
- This grid is populated by relatively recent sensor data
- Grid cell values are equivalent to the probability that there is an obstacle in that cell





Local





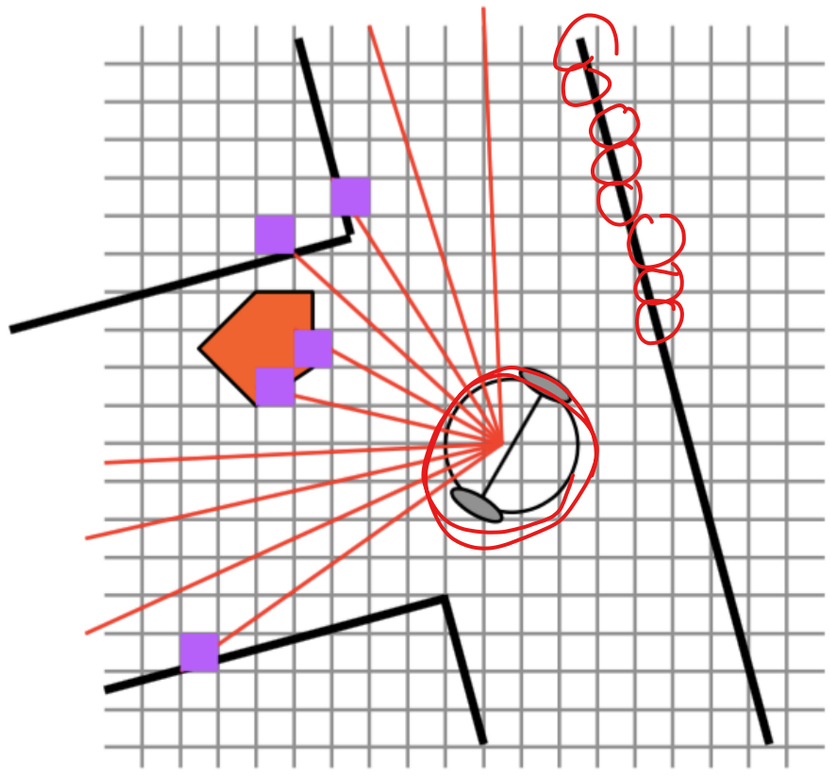
Global

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:

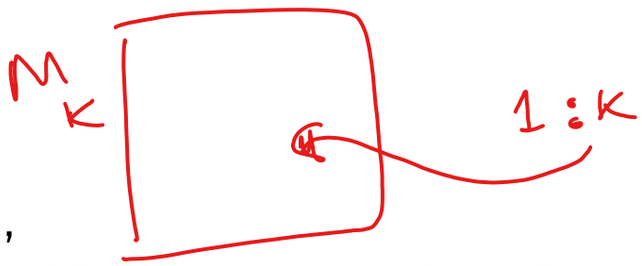
$$M = \sum_{i=1}^{N} m_i$$

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**:

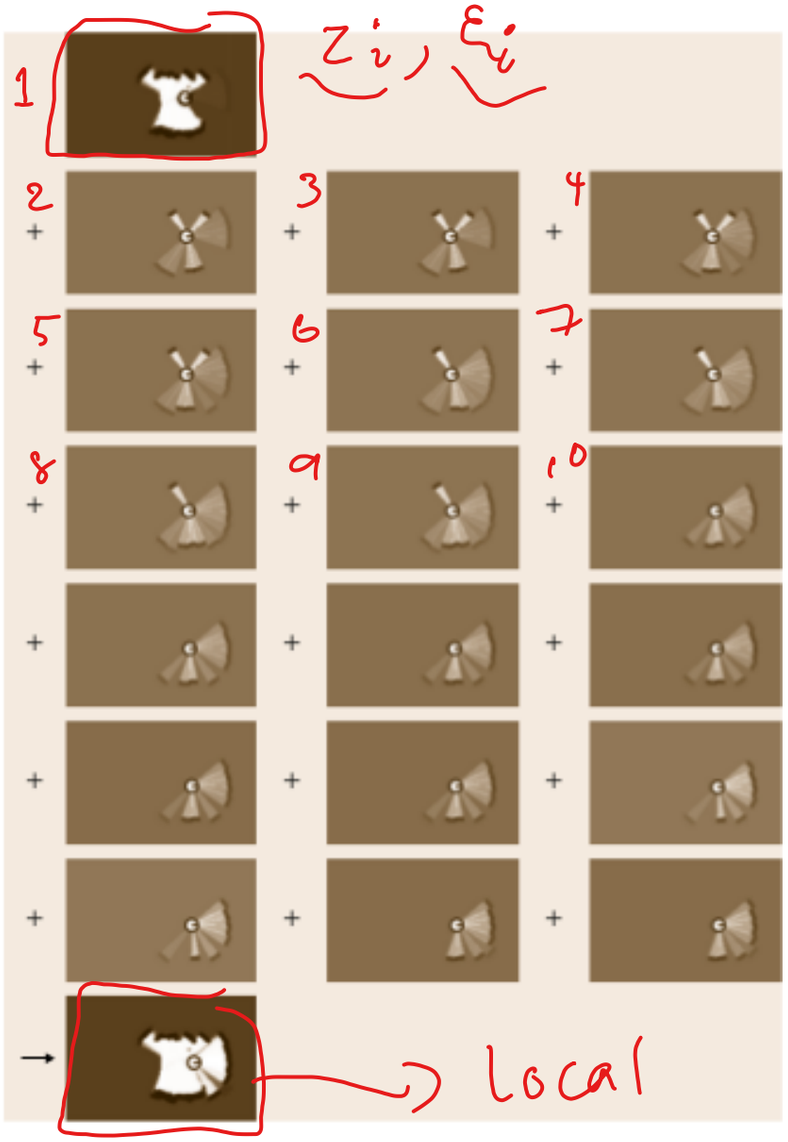


$$p(M \mid z_{1:k}, \xi_{1:k})$$

position / orientation

observations from sensor

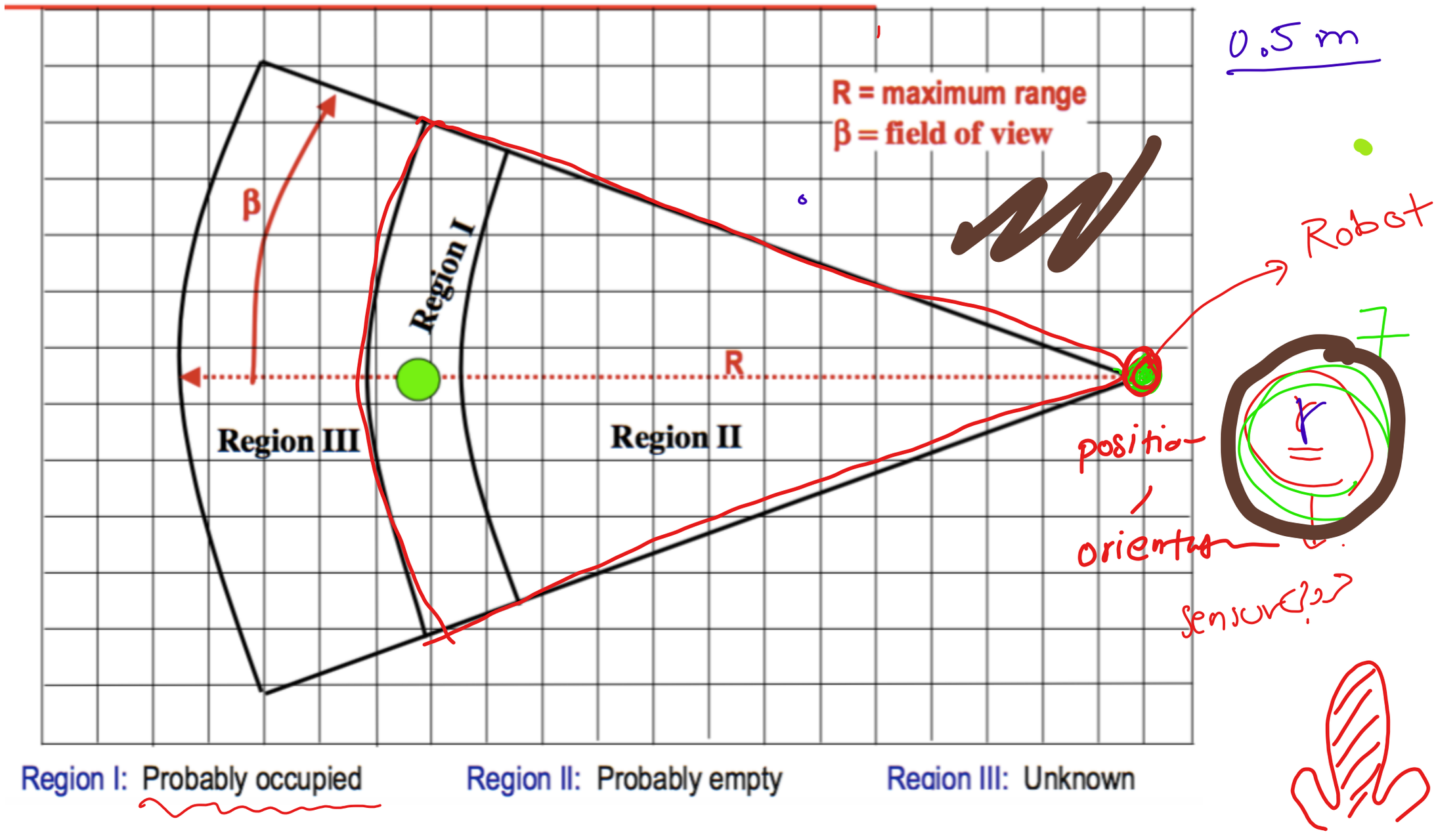
, which means estimating the joint probability that each cell is occupied or free.



How to calculate the joint probability p 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***:

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

Empty / Occupied cell (1,2)

P(A|B)

Sensor reading

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

حسگر sensor

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

دانش پیشین / مقدم

دانش پسین / بروز شده / updated

η

RECAP: 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:

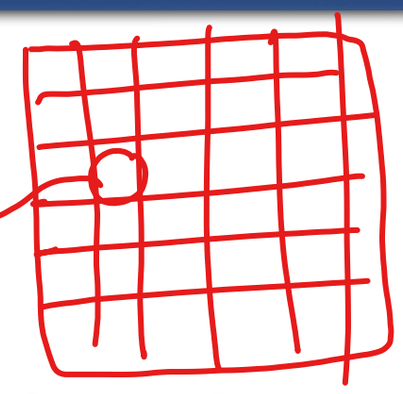
  $0 < P(H) < 1$

- 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*

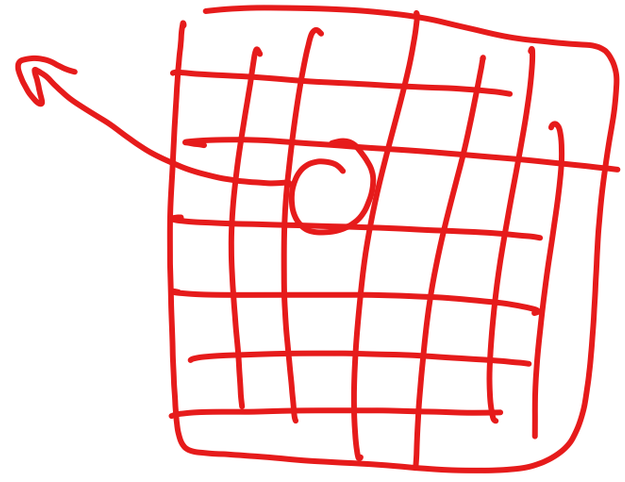


- For robotics, we want: Conditional probability: $P(H \mid s)$
  - *"Probability of* H, *given* s" *(e.g., P(Occupied | s), or P(Empty | s))*
  - *These are based on sensor readings, s*

- 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

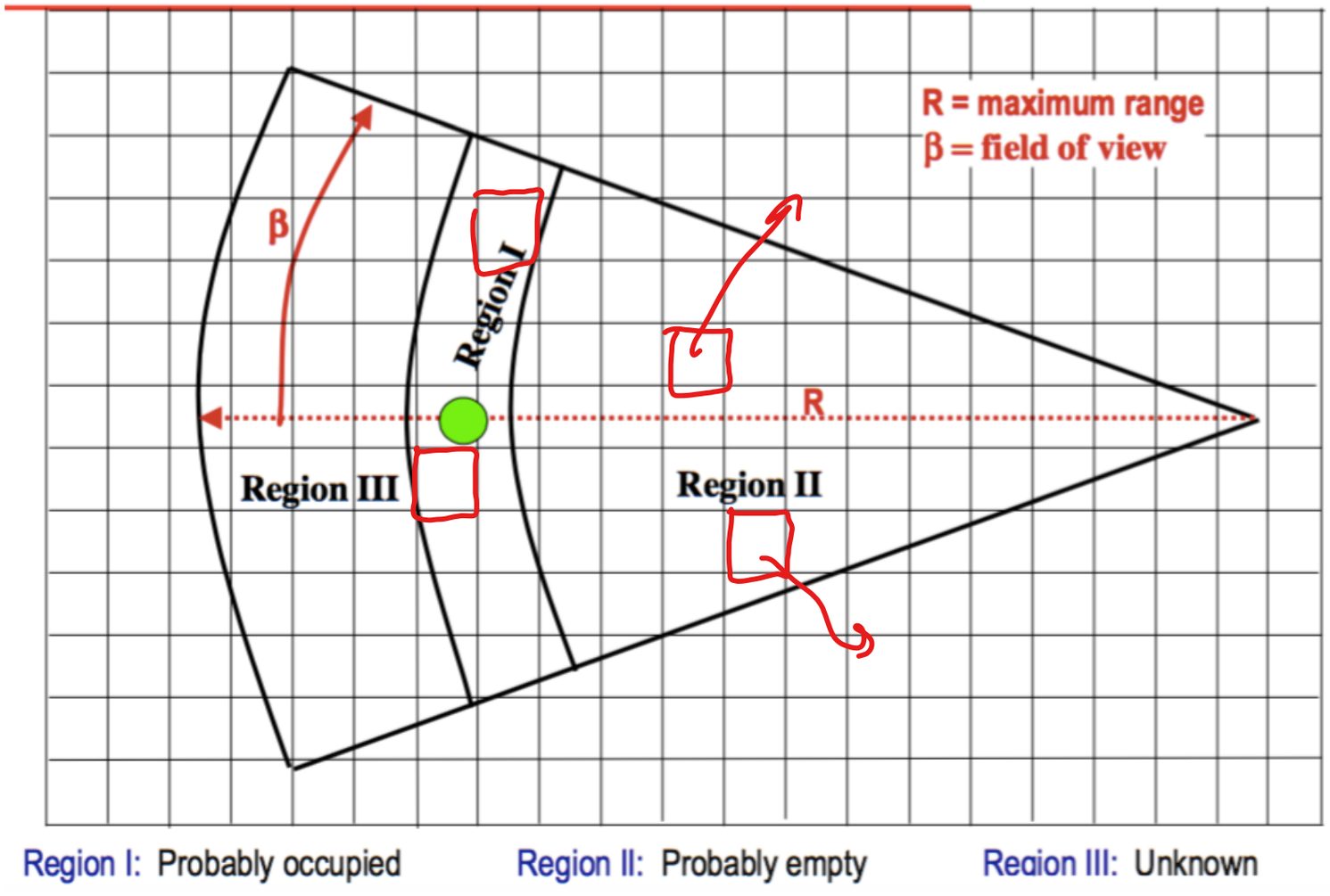


- For each grid[i][j] covered by sensor scan:
  - *Compute* P(Occupied / s) *and* P(Empty / s)
- For each grid element, grid[i][j], store tuple of the two probabilities:

```
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?

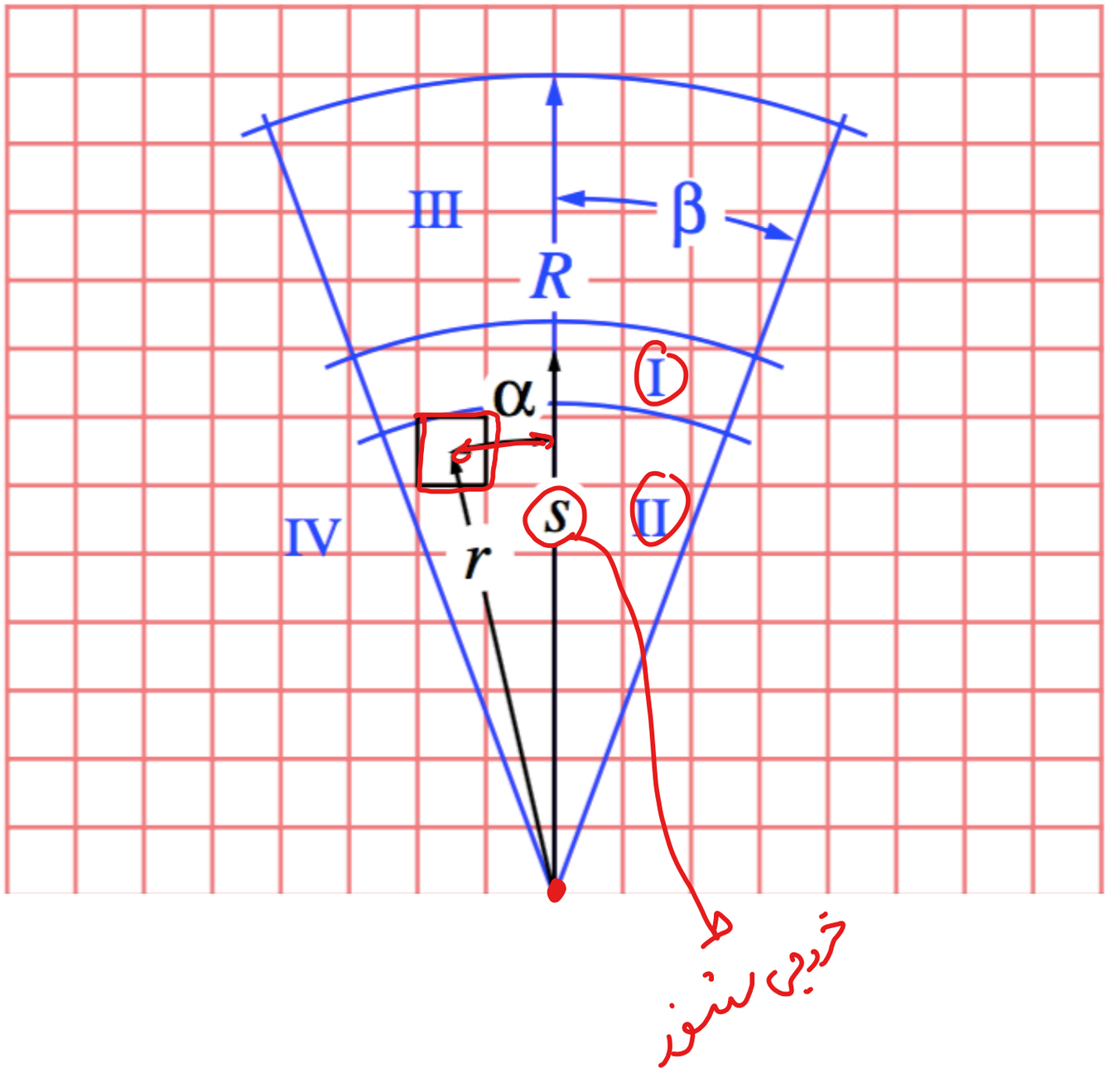
PARAMETERS FOR REASONING ABOUT A CELL

r : فاصله سلول از سنسور

α : زاویه انحراف سلول نسبت به سنسور

R : برد سنسور

β : FoV



خودی نیست

Model for Region I

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

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

(0.98)

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

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

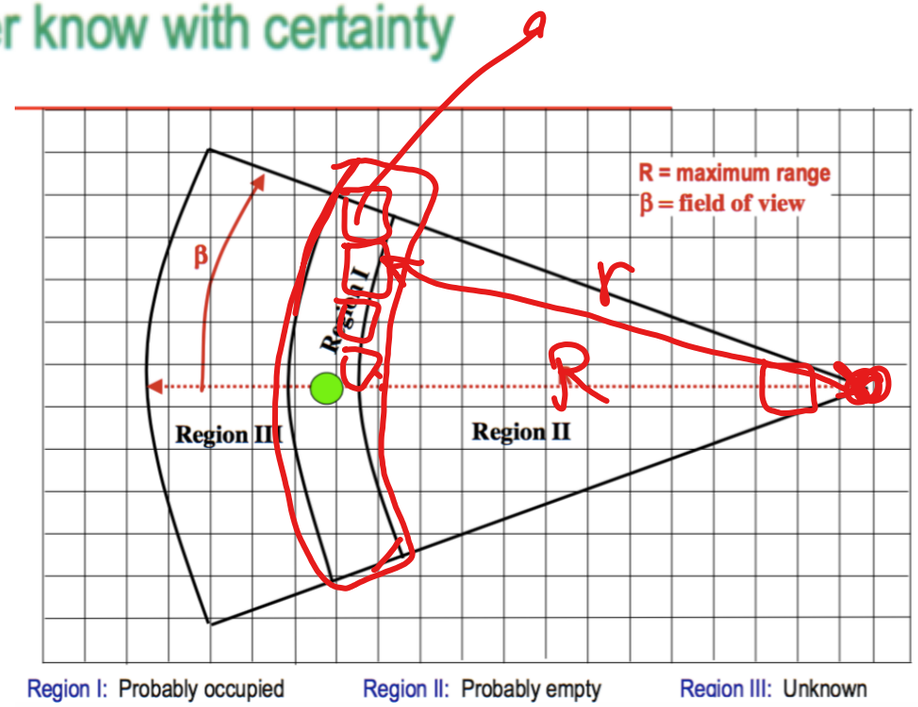
We never know with certainty

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)

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



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

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

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

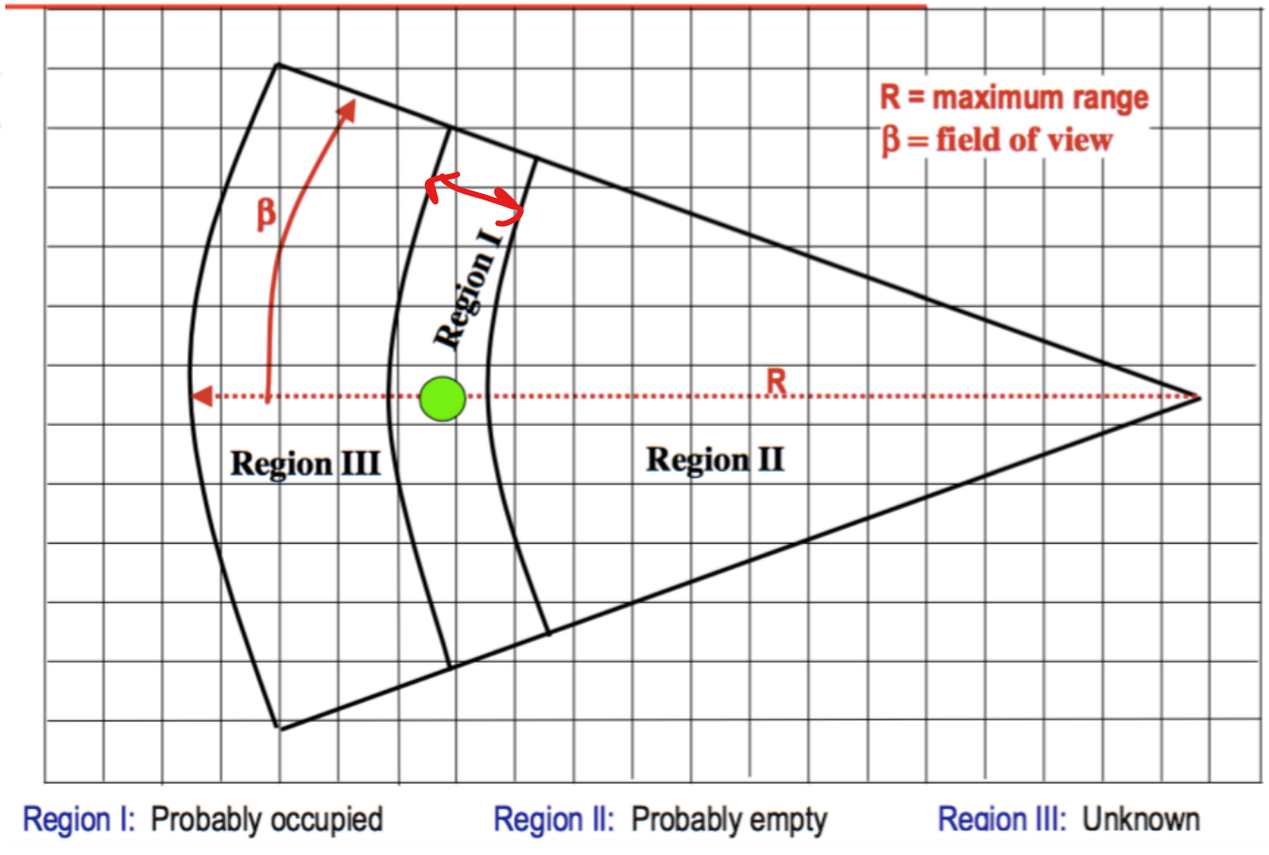
where r *is distance to grid element being updated,*
α is angle to grid element being updated

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

SENSOR TOLERANCE

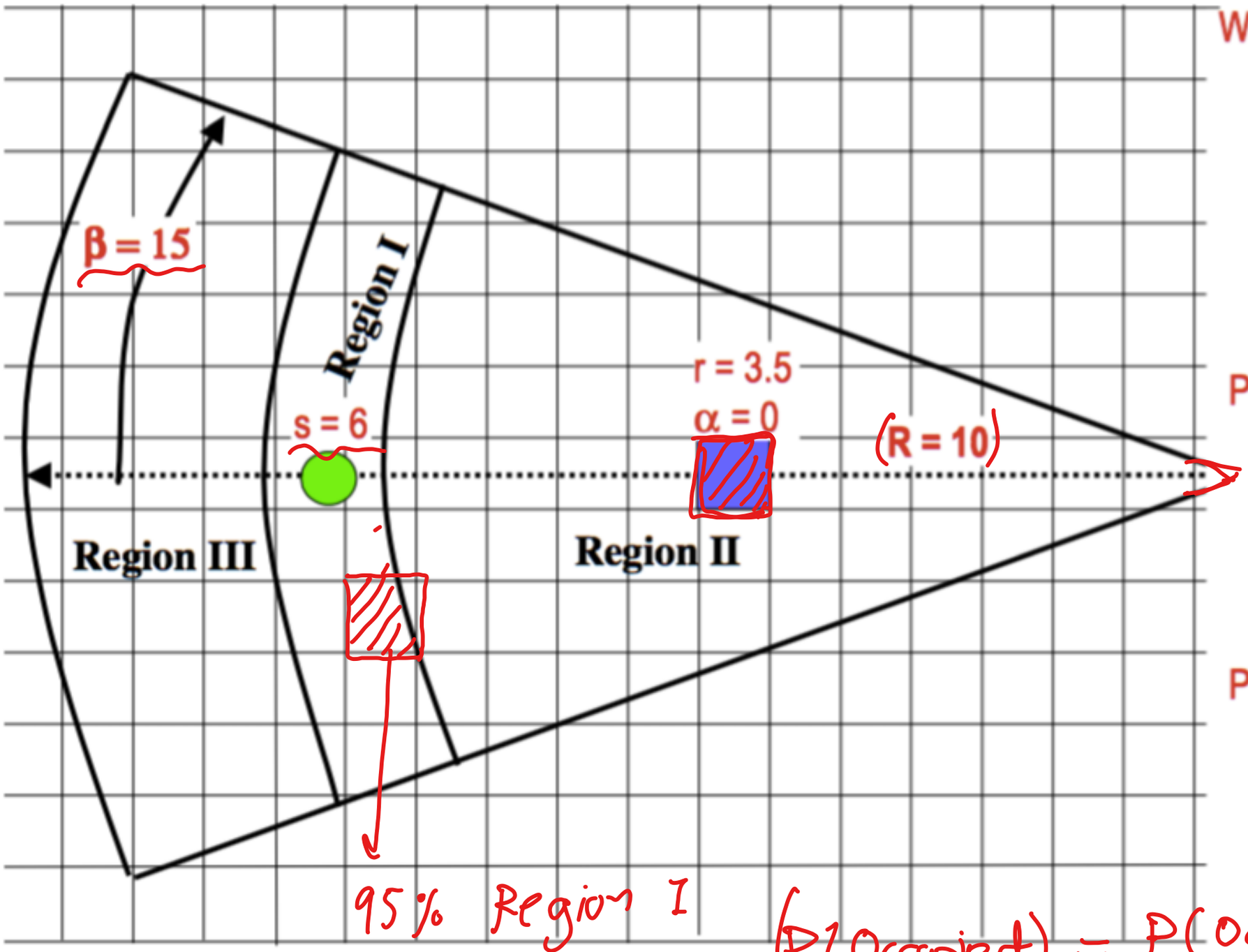
- Sonar range readings have resolution error
- Thus, specific reading might actually indicate range of possible values
- E.g., reading of 0.87 meters actually means within (0.82, 0.92) meters
  - *Therefore, tolerance in this case is 0.05 meters.*

- Tolerance gives width of Region I



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

95% Region I

5% Region II

(P(Occupied) = P(Occupied ~ I) x 0.95 + P(~ ~ II) x 0.05)

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* | *Occupied*) and *P*(*s* | *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  …

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

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

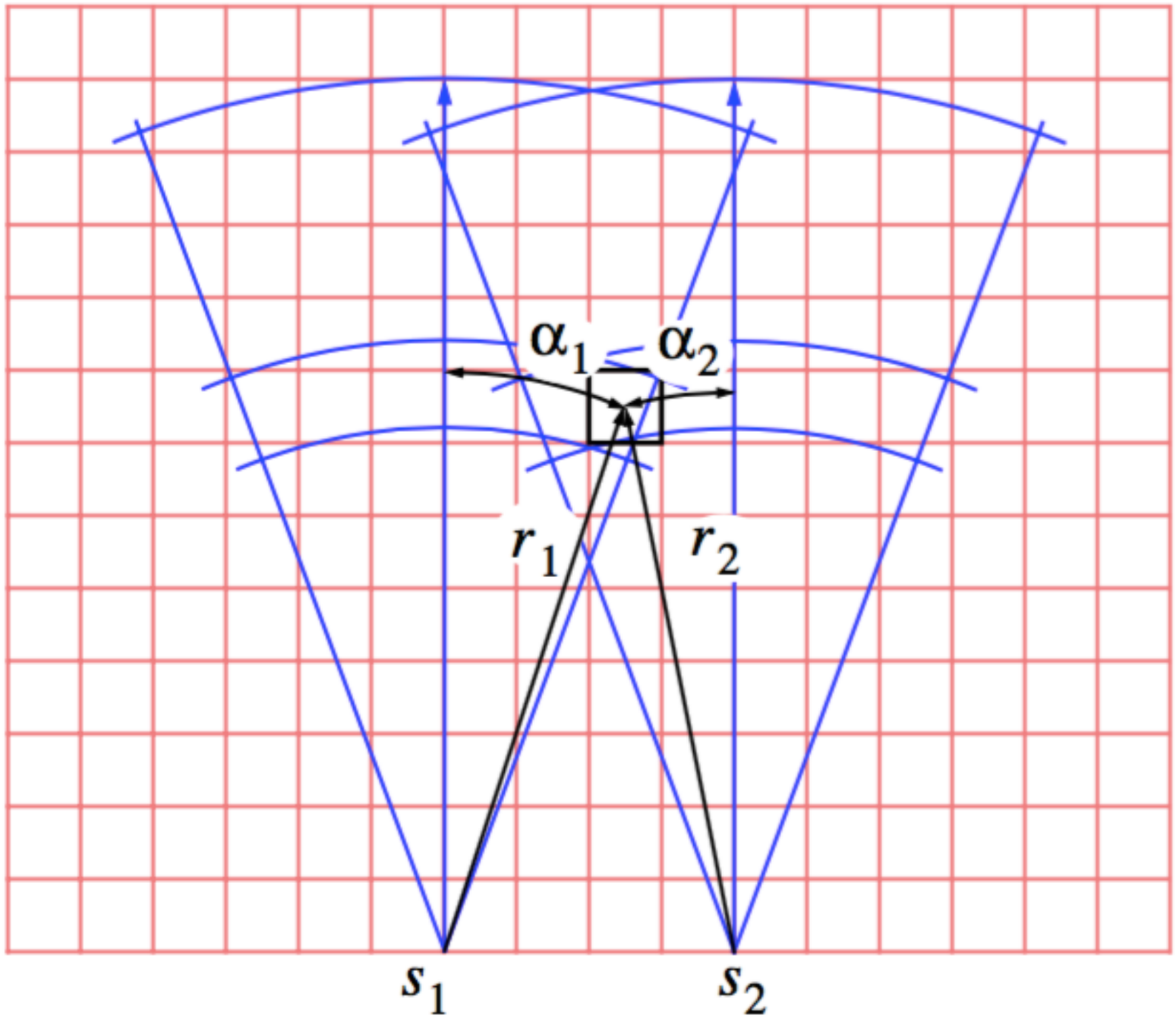
$$= 0.62$$

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

- *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:*

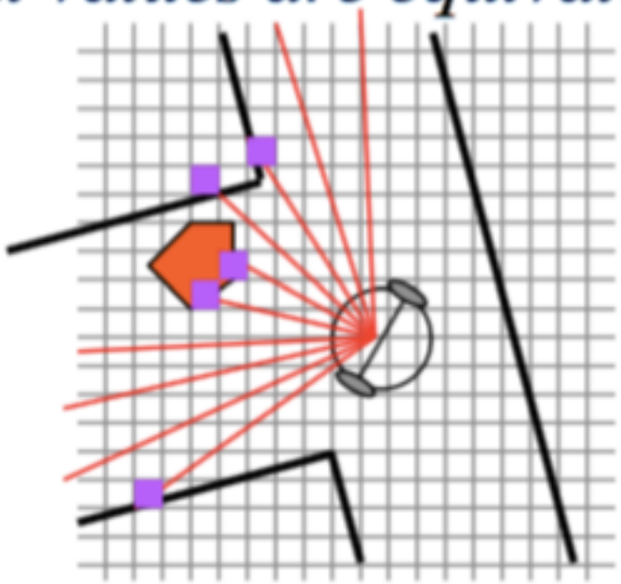
$$P(H \mid s_n) = \frac{P(s_n \mid H)P(H \mid s_{n-1})}{P(s_n \mid H)P(H \mid s_{n-1}) + P(s_n \mid \neg H)P(\neg H \mid s_{n-1})}$$

VECTOR FIELD HISTOGRAM (VHF)

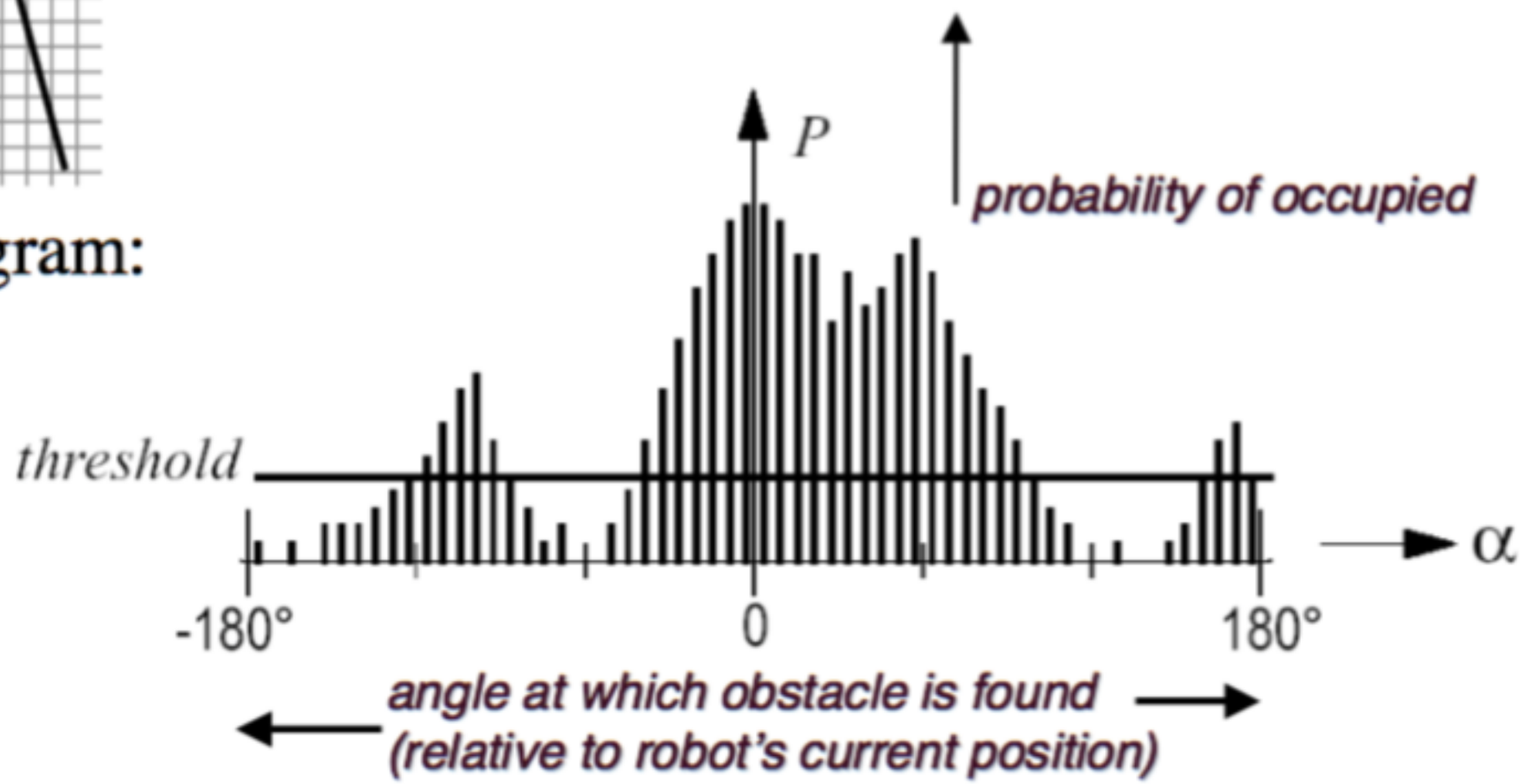
- Environment represented in a grid (2 DOF)

Koren & Borenstein, ICRA 1990

  - *cell values are equivalent to the probability that there is an obstacle*



- Generate polar histogram:

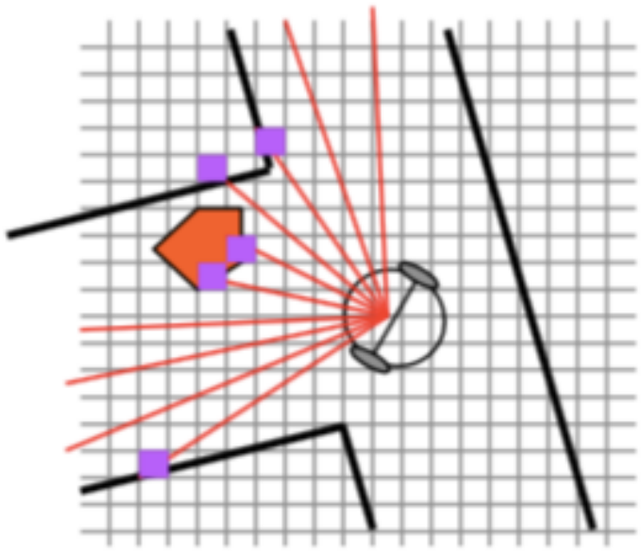


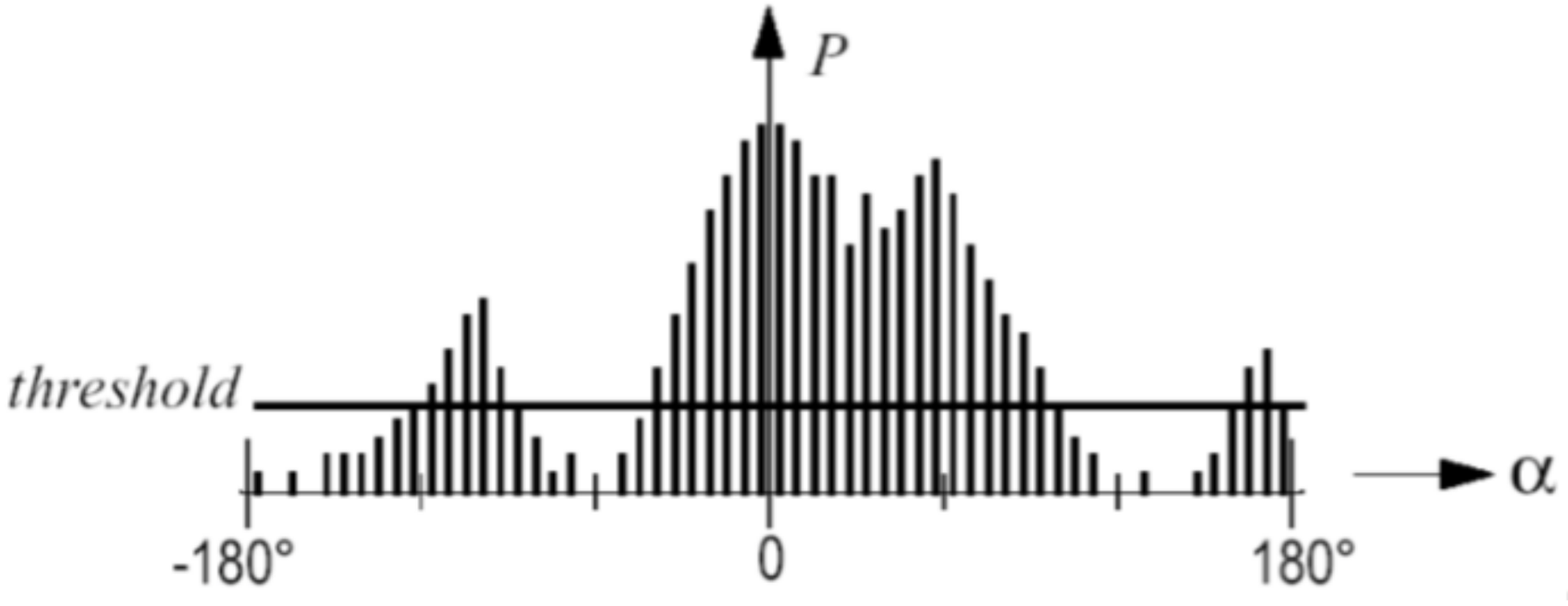
VECTOR FIELD HISTOGRAM (VHF)

- From histogram, calculate steering direction: *Koren & Borenstein, ICRA 1990*
  - *Find all openings large enough for the robot to pass through*
  - *Apply cost function G to each opening*

    $G = a \cdot \text{target\_direction} + b \cdot \text{wheel\_orientation} + c \cdot \text{previous\_direction}$

    where:
    - *target_direction = alignment of robot path with goal*
    - *wheel_orientation = difference between new direction and current wheel orientation*
    - *previous_direction = difference between previously selected direction and new direction*
  - *Choose the opening with lowest cost function value*

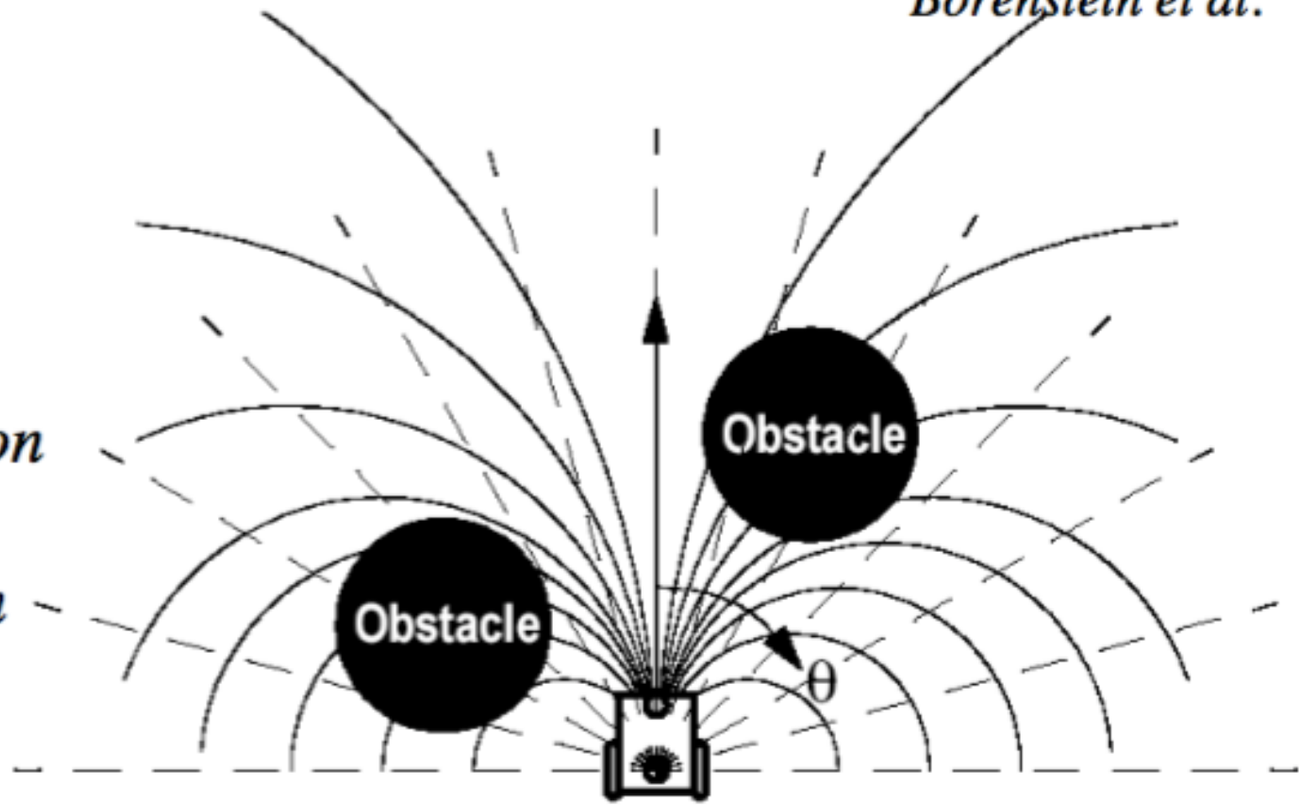


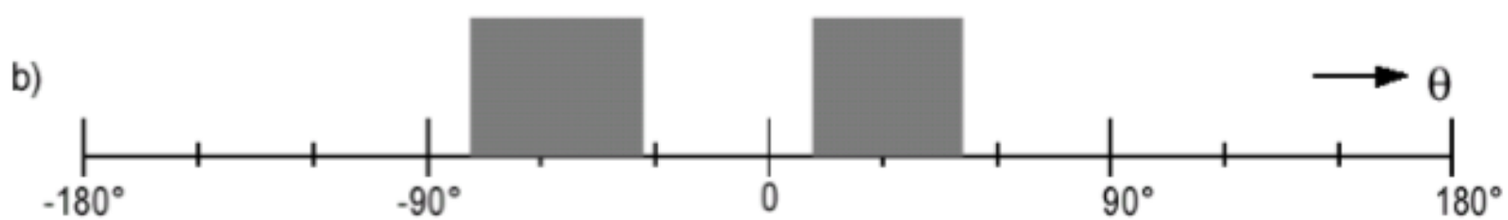


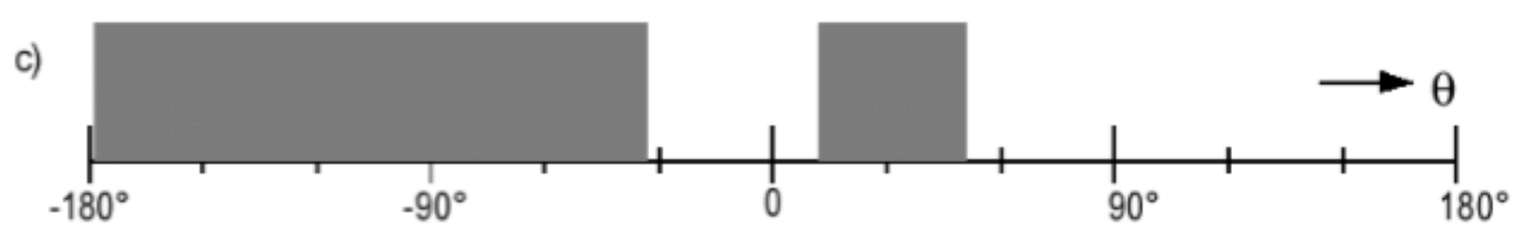
Obstacle Avoidance: **Vector Field Histogram + (VFH+)**

- Accounts also in a very simplified way for the moving trajectories
  - *robot can move on arcs*
  - *arcs take into account kinematics*
  - *obstacles blocking a given direction also block all the trajectories (arcs) going through this direction*

Borenstein et al.





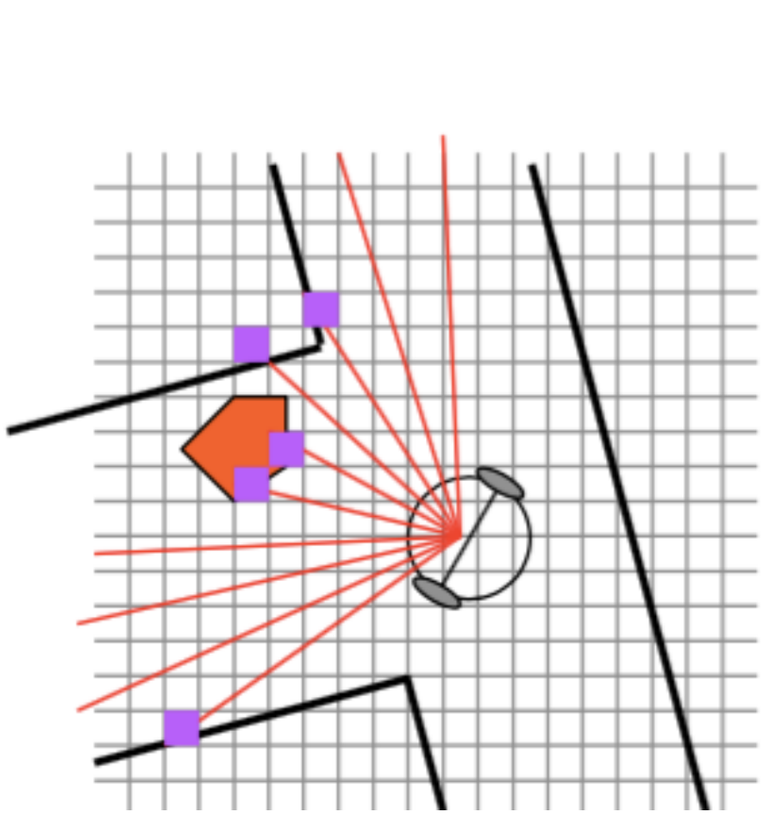


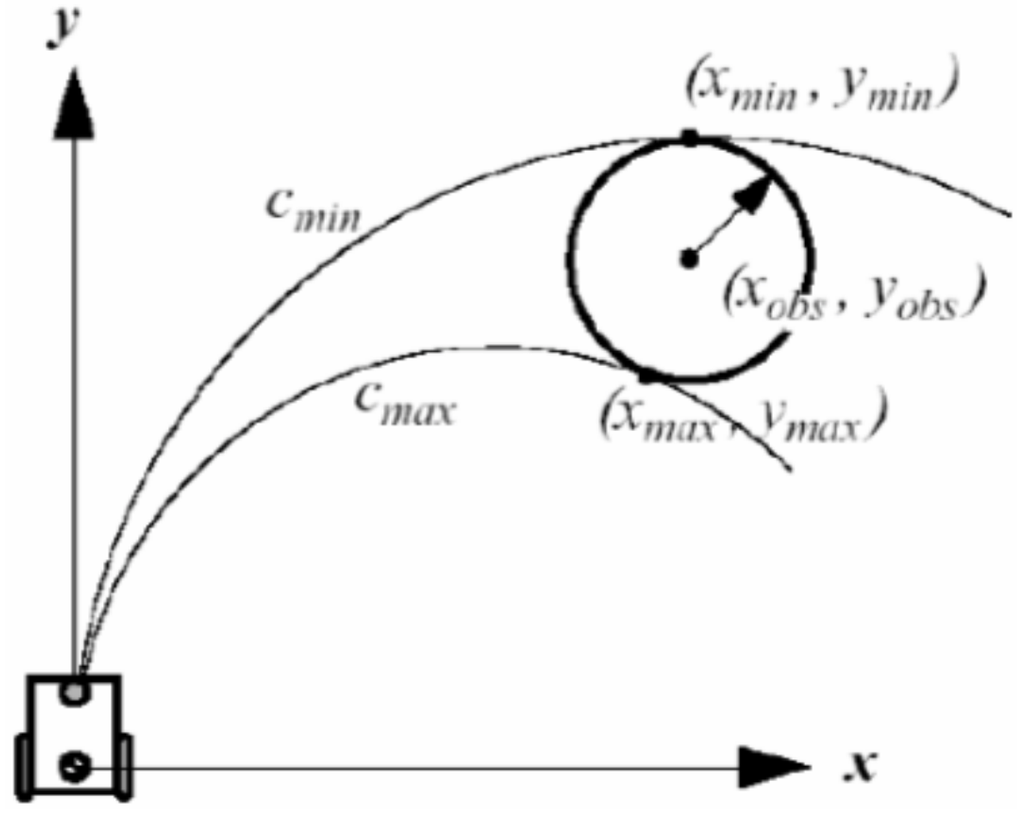
Caprari et al. 2002

Obstacle Avoidance: Basic Curvature Velocity Methods (CVM)

Simmons et al.

- Adding *physical constraints* from the robot and the environment on the *velocity space* (v, ω) of the robot
  - *Assumption that robot is traveling on arcs* $(c = \omega / v)$
  - *Constraints:* $-v_{max} < v < v_{max}$ $\quad -\omega_{max} < \omega < \omega_{max}$
  - *Obstacle constraints: Obstacles are transformed in velocity space*
  - *Objective function used to select the optimal speed*

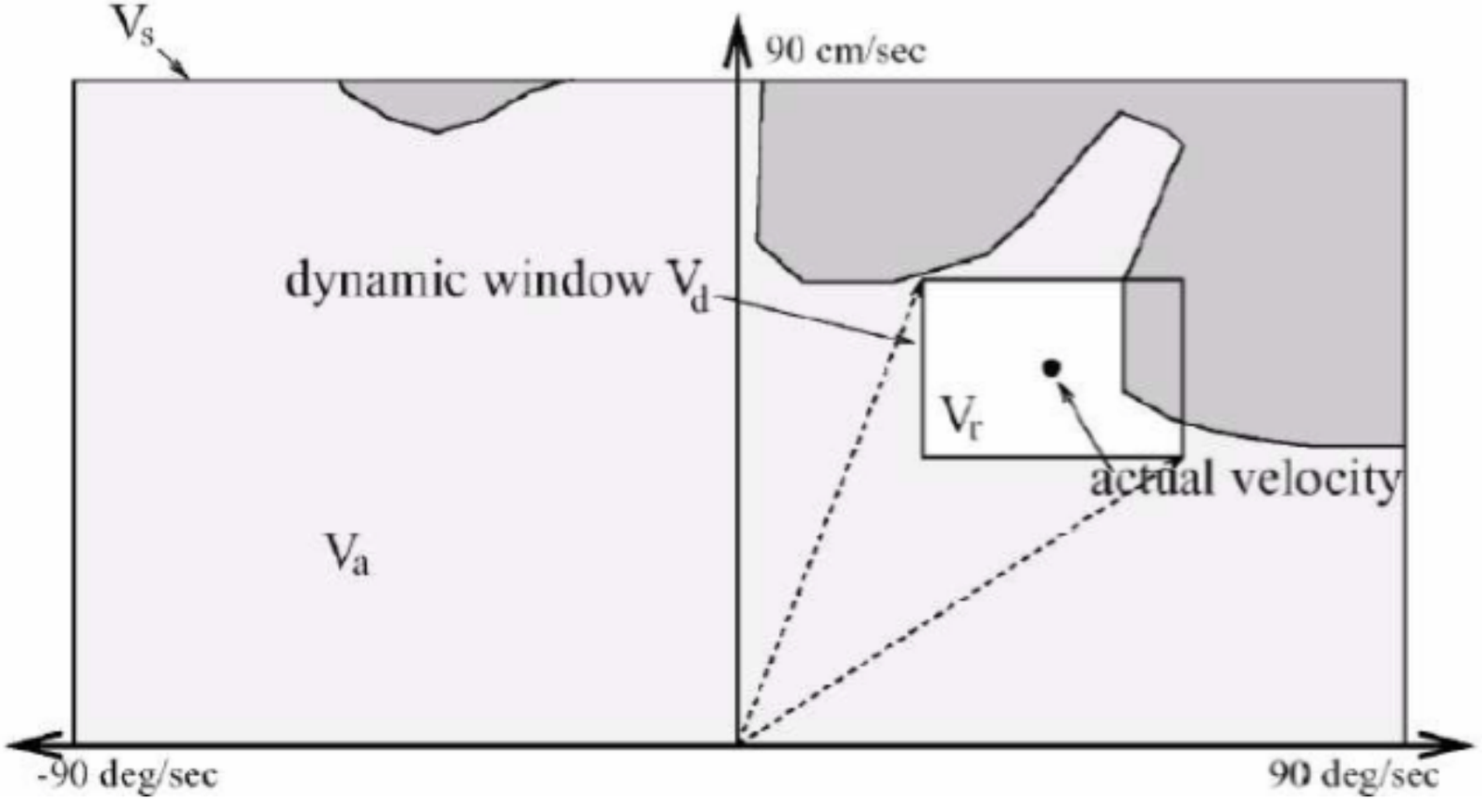




DYNAMIC WINDOW APPROACH (ALREADY SEEN)

- The kinematics of the robot is considered by searching a well chosen velocity space
  - *velocity space -> some sort of configuration space*
  - *robot is assumed to move on arcs*
  - *ensures that the robot comes to stop before hitting an obstacle*
  - *objective function is chosen to select the optimal velocity*

$$O = a \cdot heading(v, \omega) + b \cdot velocity(v, \omega) + c \cdot dist(v, \omega)$$



- *heading* = progress toward goal
- *velocity* = forward velocity of robot (encourages fast movements)
- *dist* = distance to closest obstacle in trajectory