Robot Localization: Historical Context

- Initially, roboticists thought the world could be modeled exactly
- Path planning and control assumed perfect, exact, deterministic world
- Reactive robotics (behavior based, ala bug algorithms) were developed due to imperfect world models
- But Reactive robotics assumes accurate control and sensing to react – also not realistic
- Reality: imperfect world models, imperfect control, imperfect sensing
- Solution: Probabilistic approach, incorporating model, sensor and control uncertainties into localization and planning
- Reality: these methods work empirically!

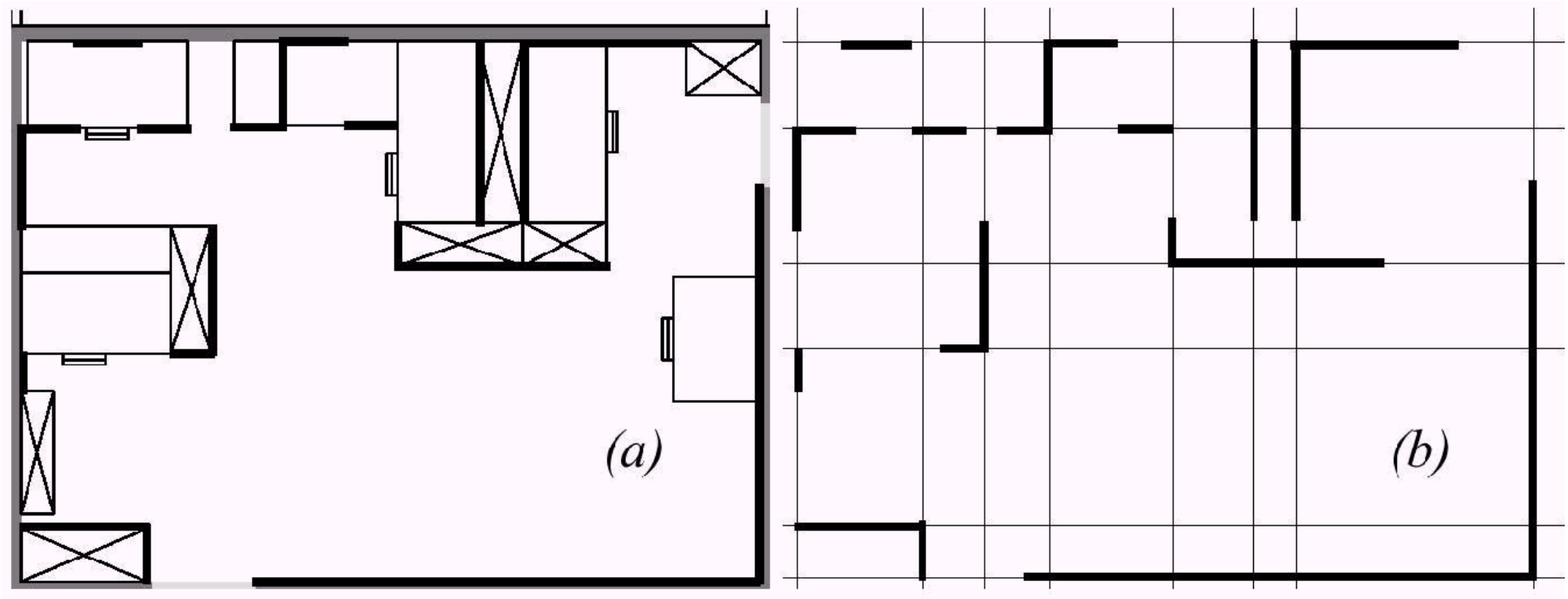
Requirements of a Map Representation for a Mobile Robot

- The precision of the map needs to match the precision with which the robot needs to achieve its goals
- The precision and type of features mapped must match the precision of the robot's sensors
- The complexity of the map has direct impact on computational complexity for localization, navigation and map updating

Map Representation
Continuous Line-Based

a) Architecture map

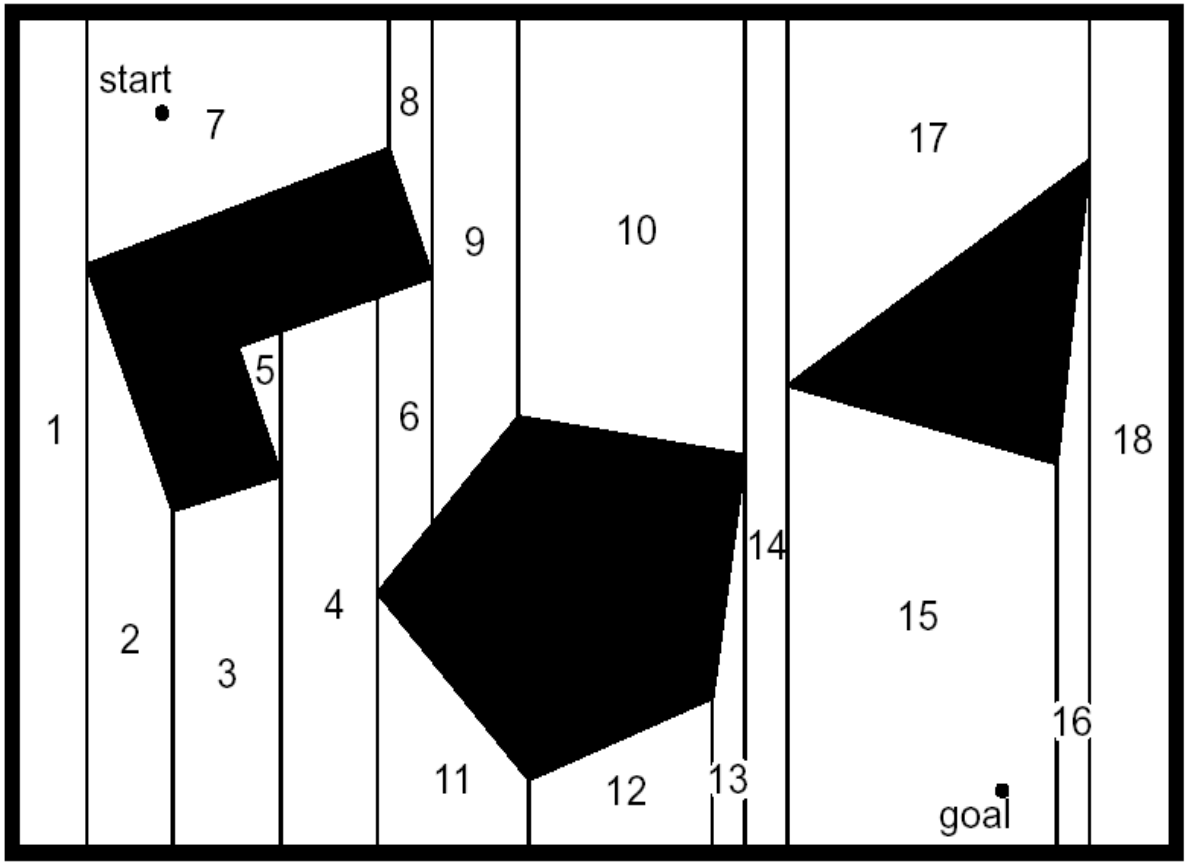
b) Representation with set of finite or infinite lines



Map Representation
Exact cell decomposition

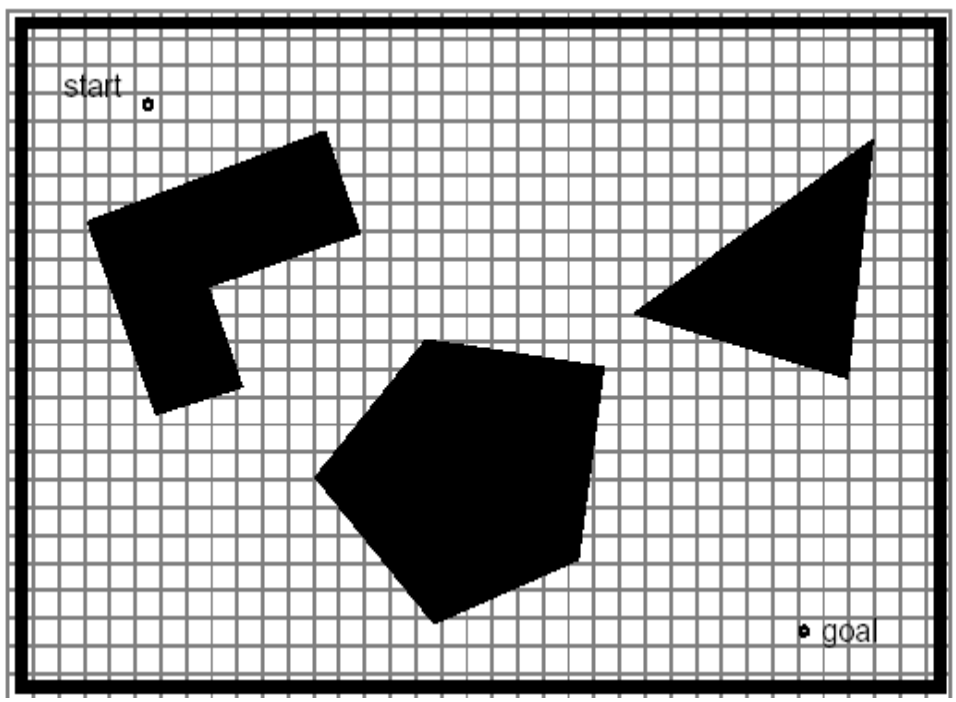
- Exact cell decomposition - Polygons

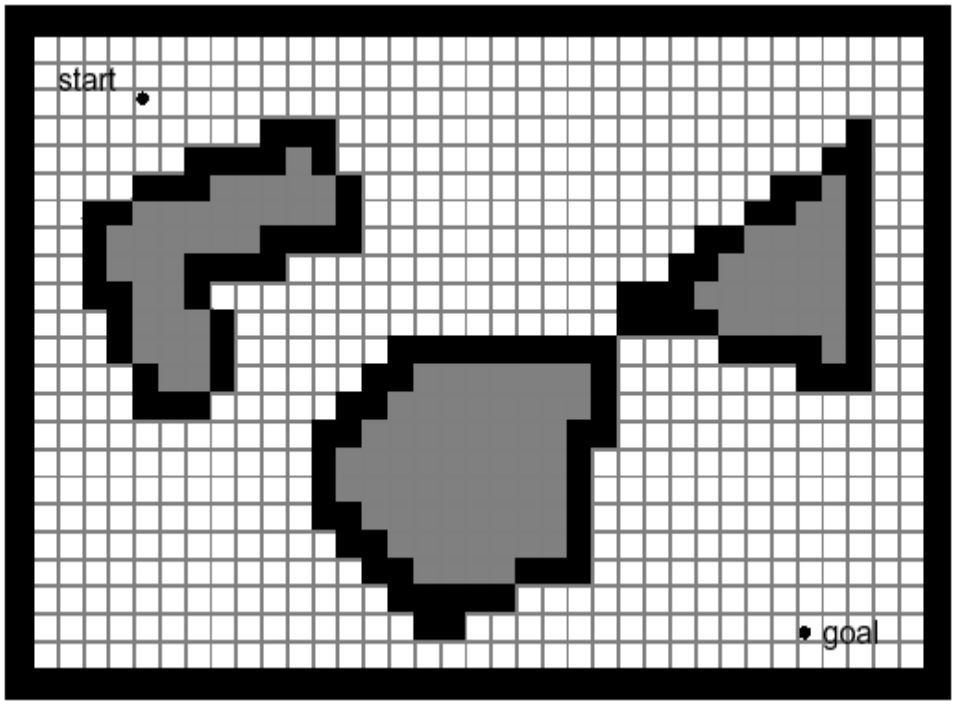
Compact representation - shows adjacency of free space cells - useful for nav.



Map Representation
Approximate cell decomposition

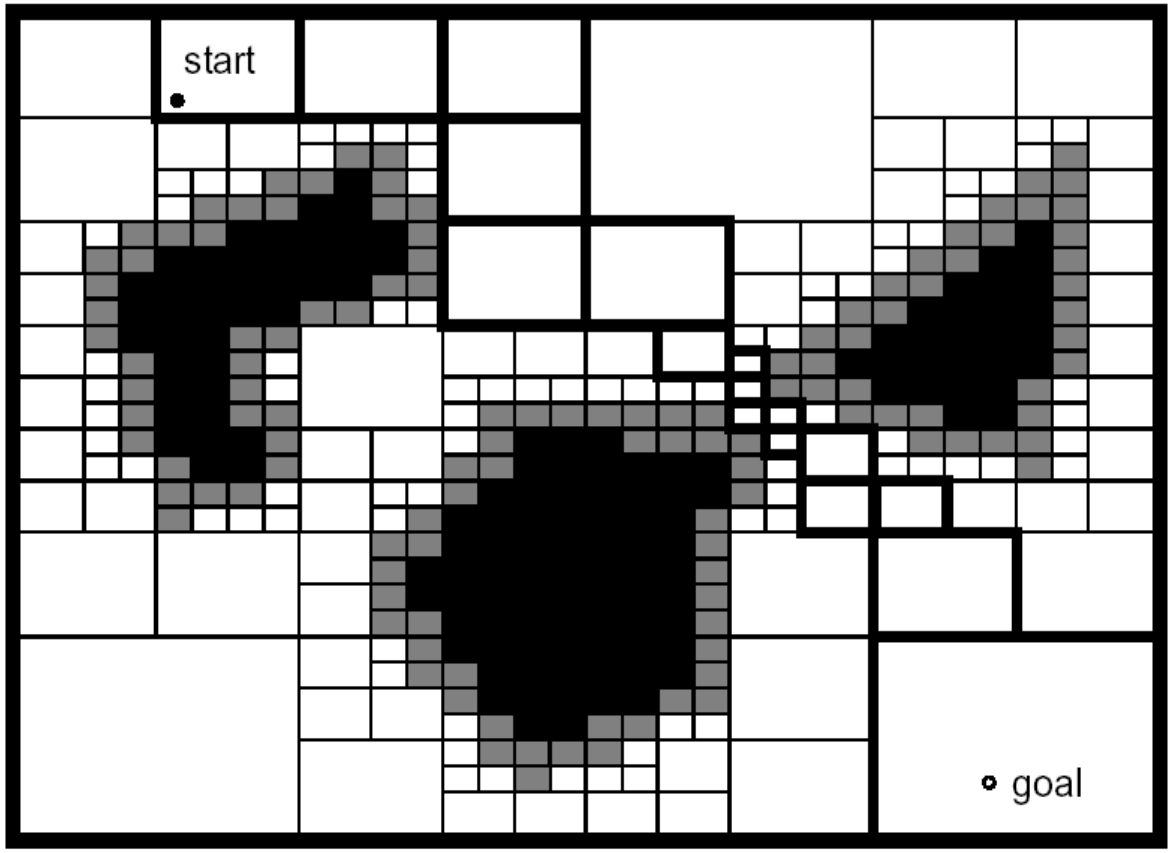
- Fixed cell decomposition - occupancy grids
  - Narrow passages disappear





Map Representation
Adaptive cell decomposition

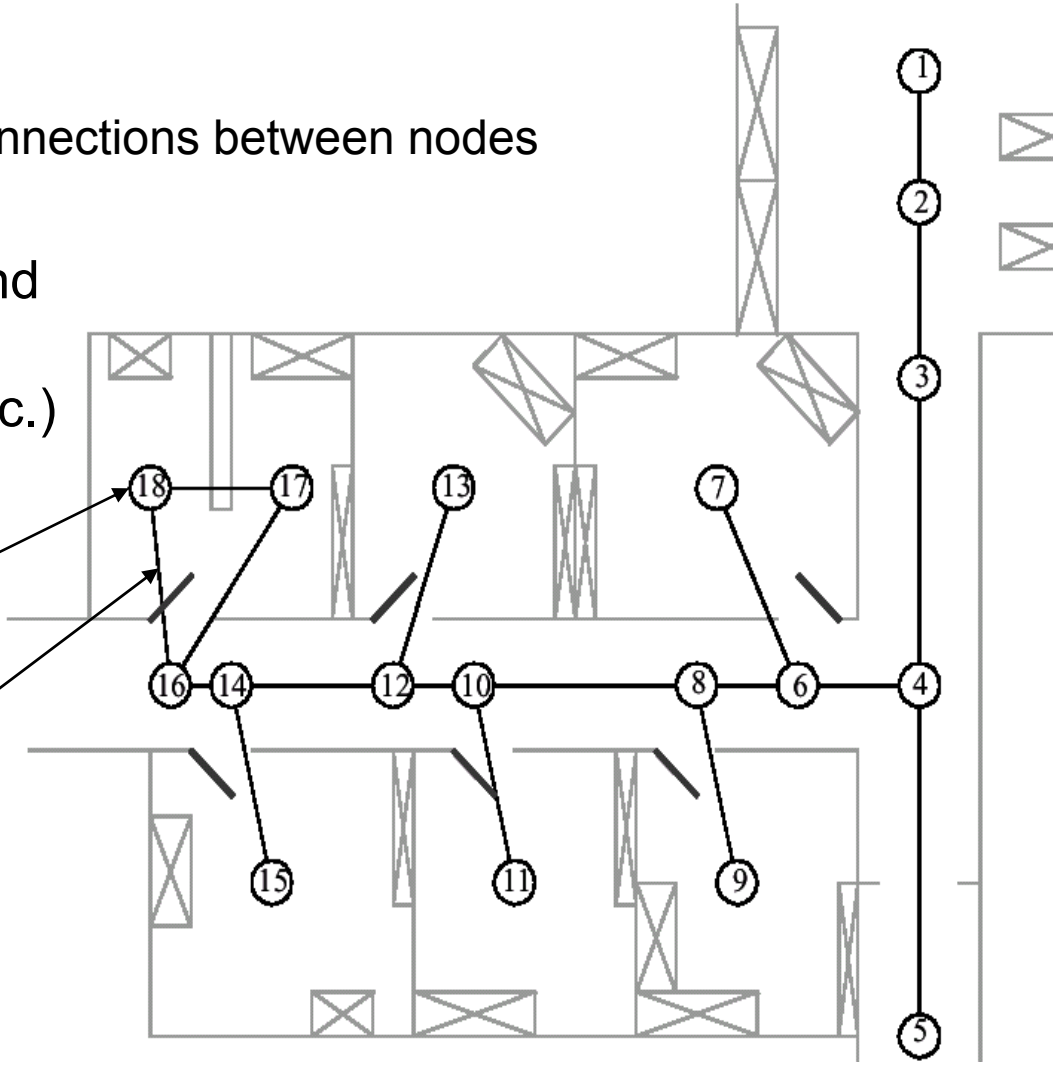
- Example: Hierarchical quadtree decomposition



Map Representation
Topological map

- A topological map represents the environment as a graph with nodes and edges.
  - Nodes correspond to spaces
  - Edge correspond to physical connections between nodes

- Topological maps lack scale and distances, but topological relationships (e.g., left, right, etc.) are mantained

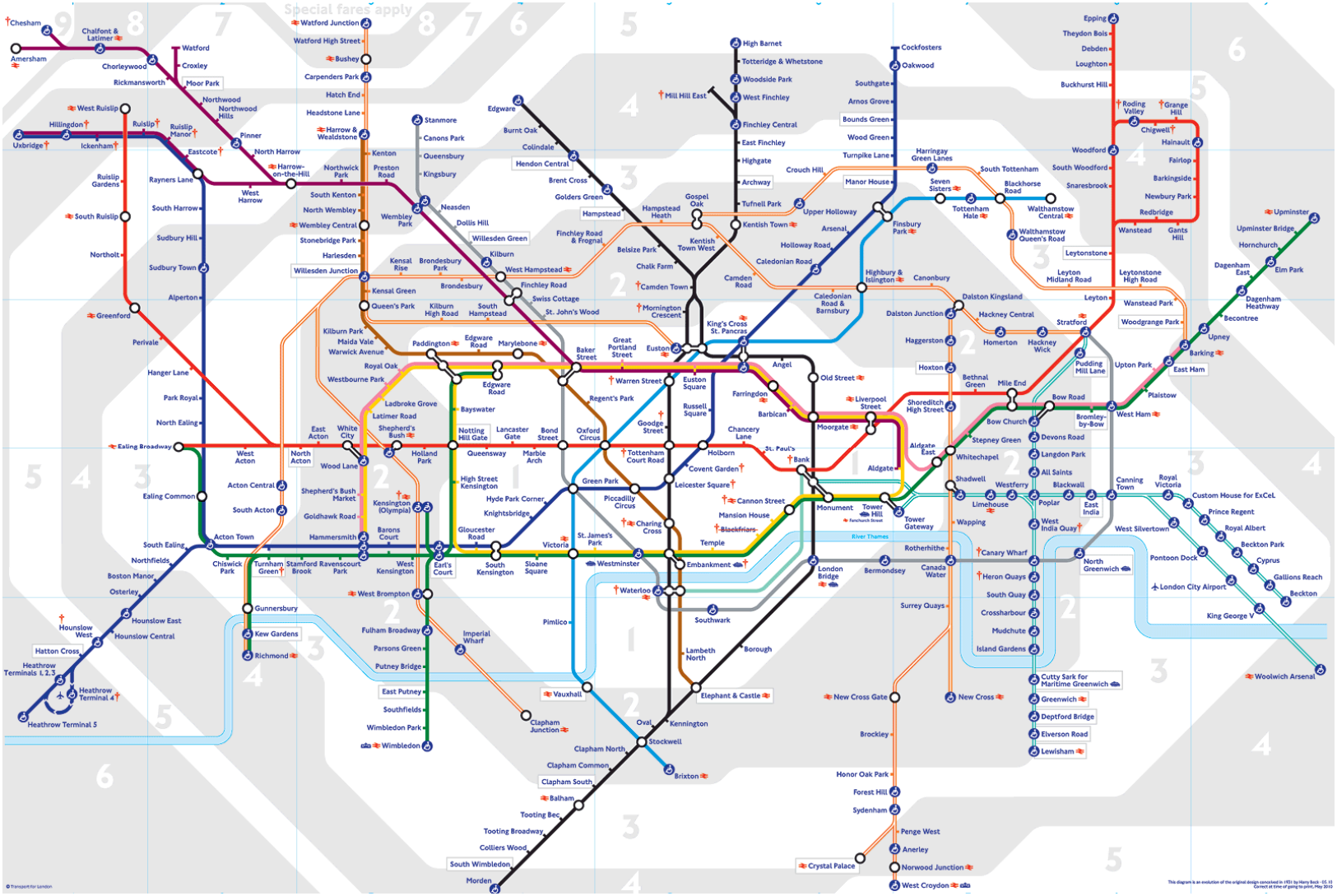


node
(location)

edge
(connectivity)

Map Representation
Topological map

- London underground map

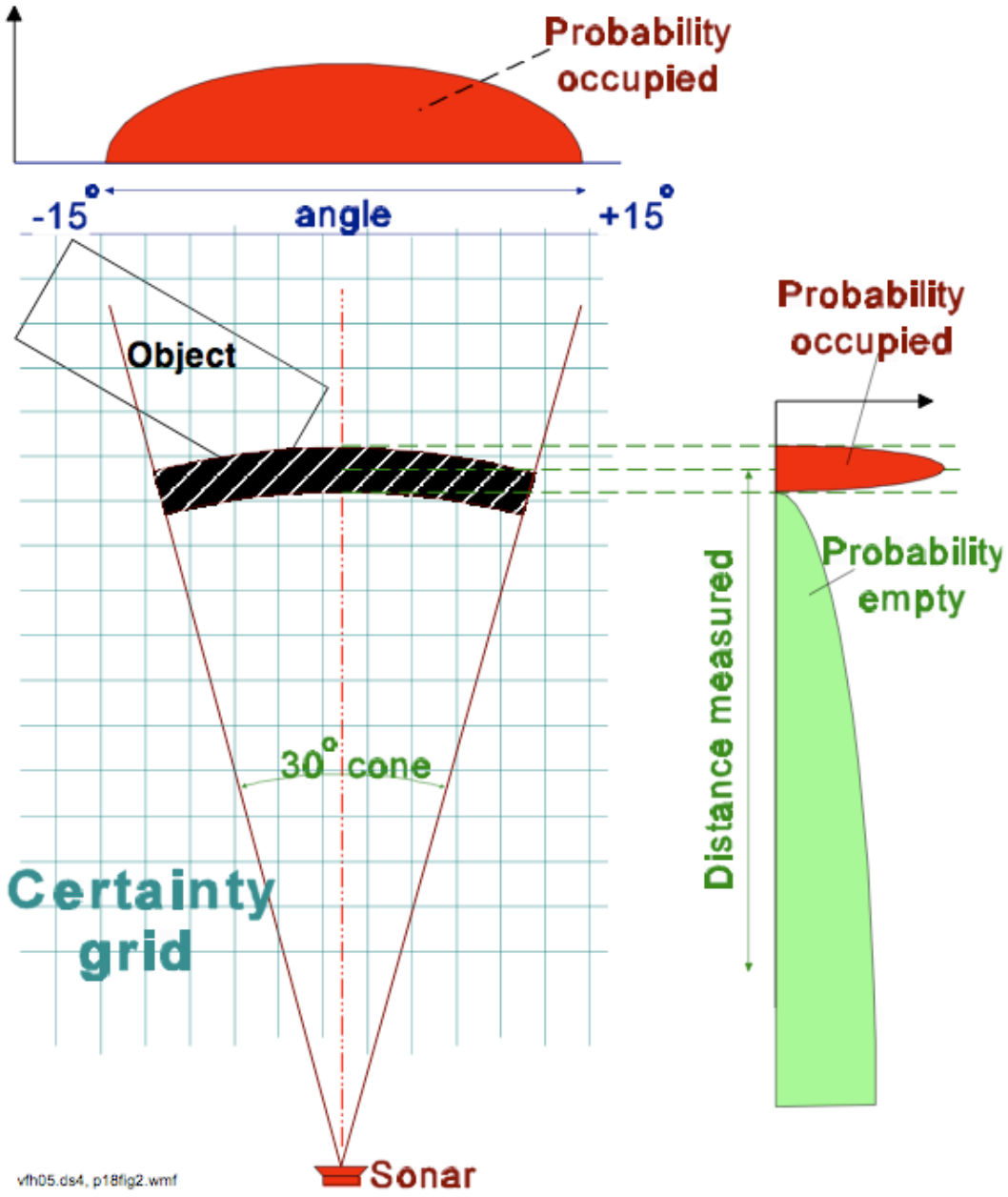


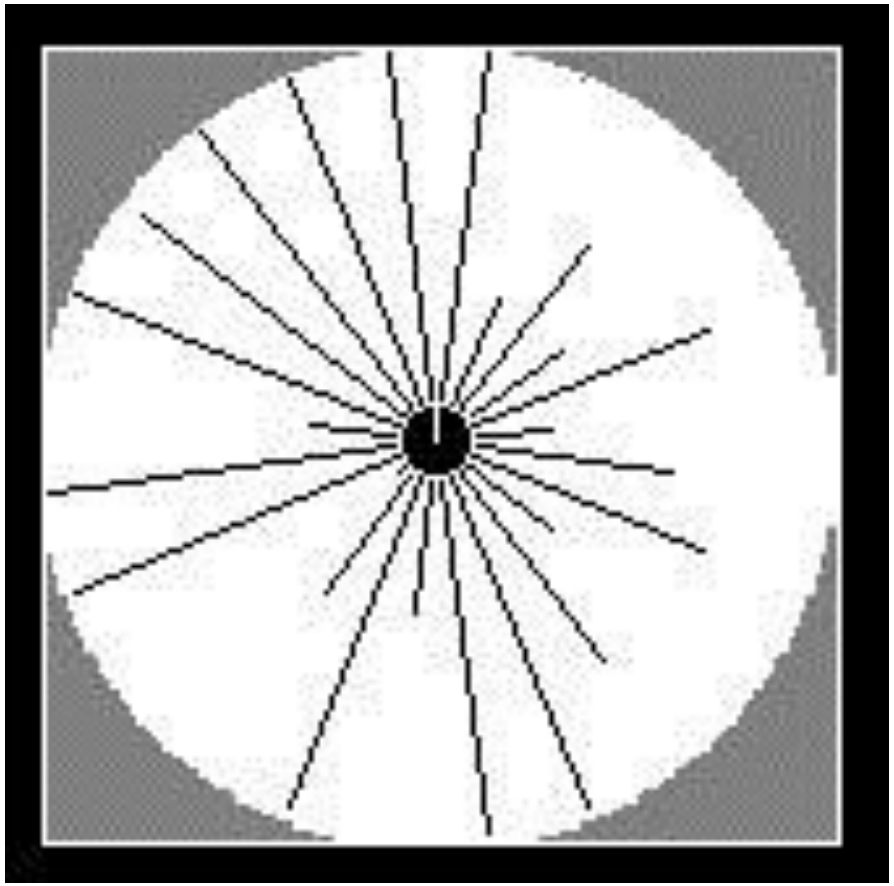
Mapping: Occupancy Grids

- 2D metric occupancy grids are used
- Each grid cell has probability of the cell being occupied, Prob(occ(x,y))
- Occupancy grid integrates multiple sensors (e.g. sonar and stereo)

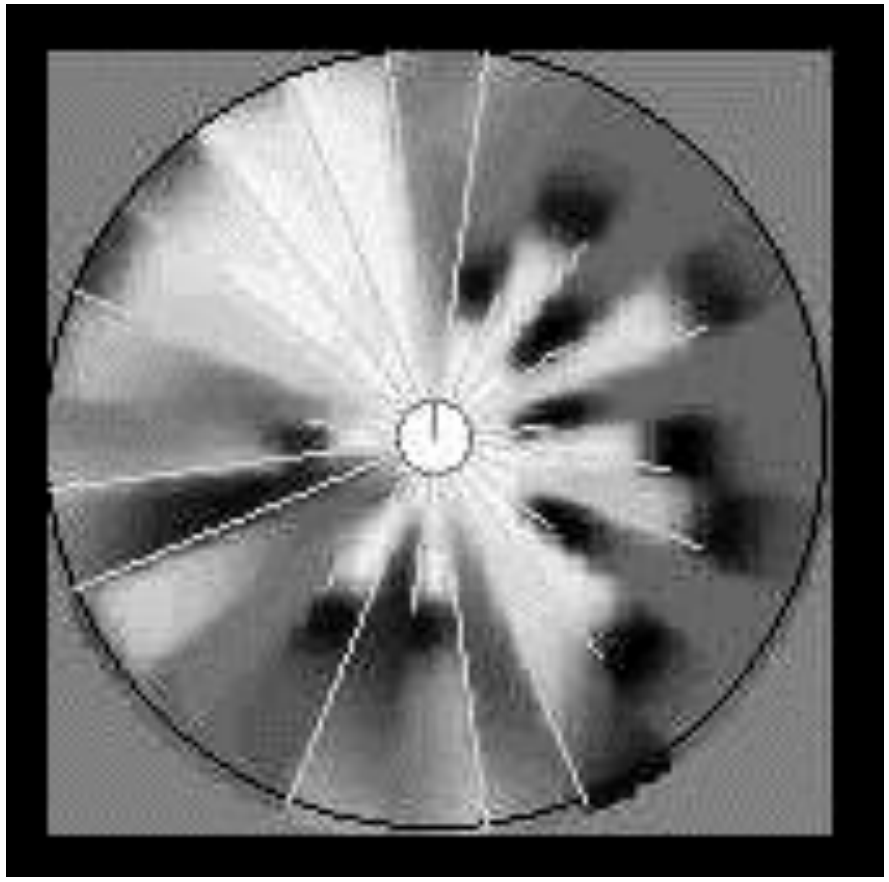
Sonar Sweeps a Wide Cone

- Obstacle could be anywhere on the arc at distance D.
- The space closer than D is likely to be free.





Sonar scan



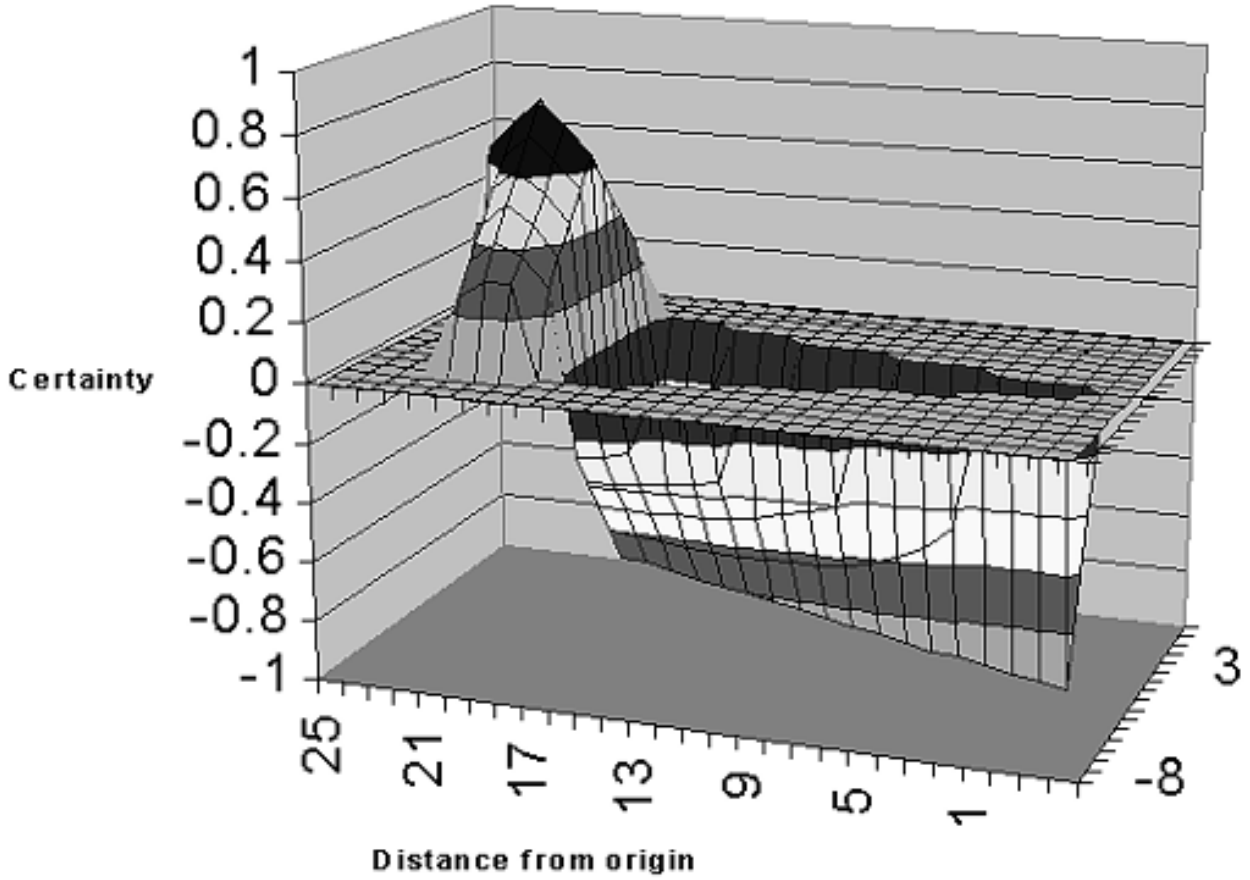
Probabilistic Occupancy Grid

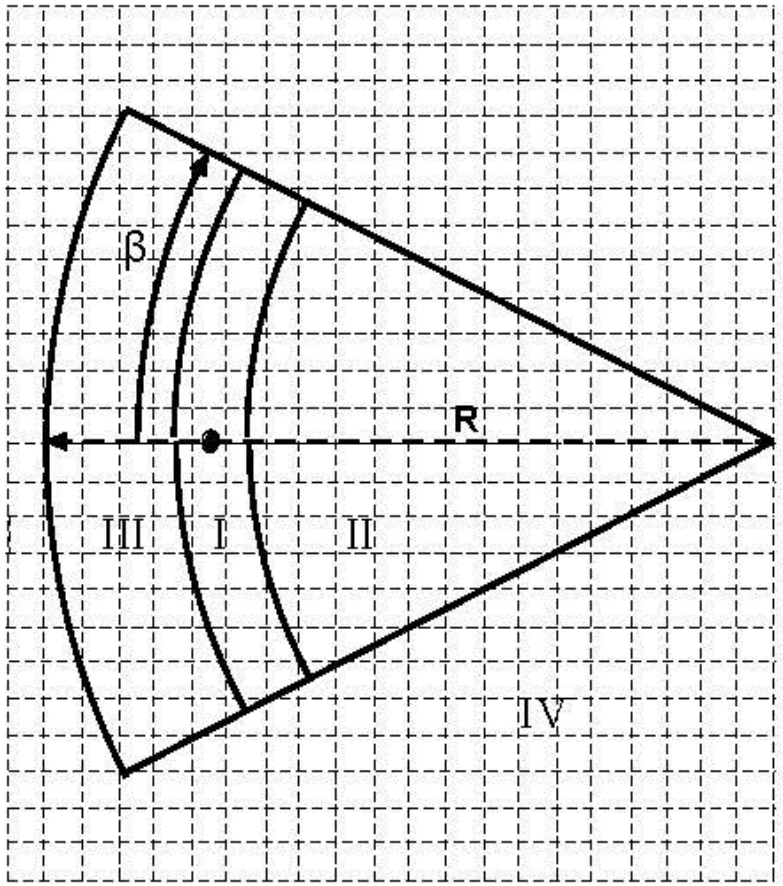
Updating Occupancy Grids Using Bayesian Estimation

Reference:

An Introduction to AI Robotics by R. Murphy, MIT Press, chapter 11

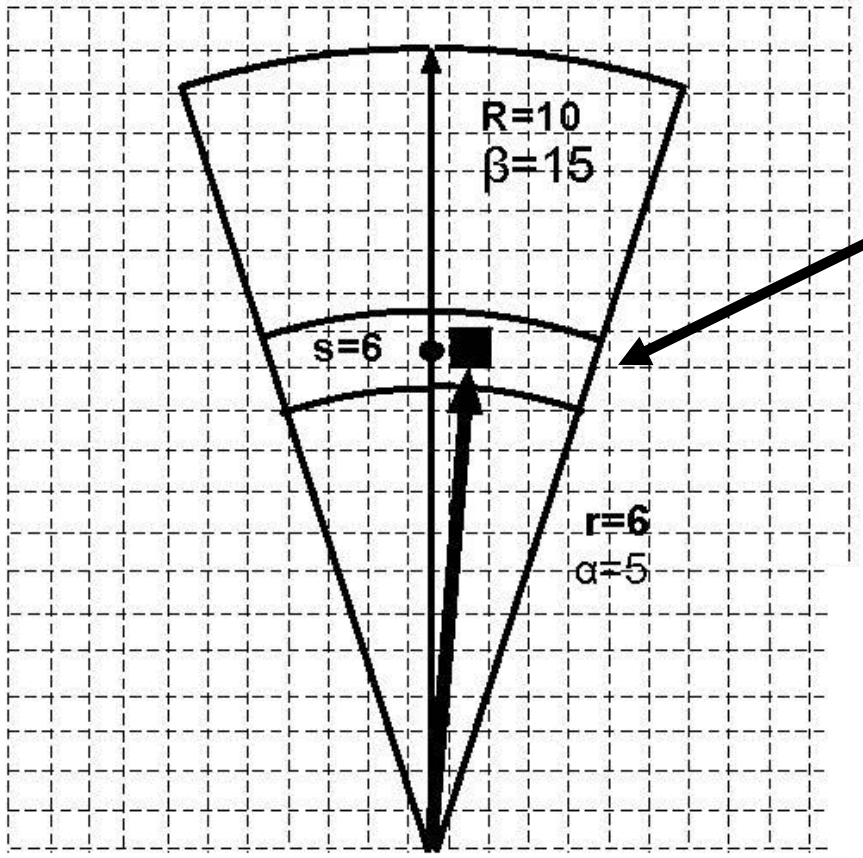
Sonar Sensor Model





Region I: Probable obstacle Region II: Probable free-space Region III: unknown

Mapping Sonar values to Occ grid values: Region I



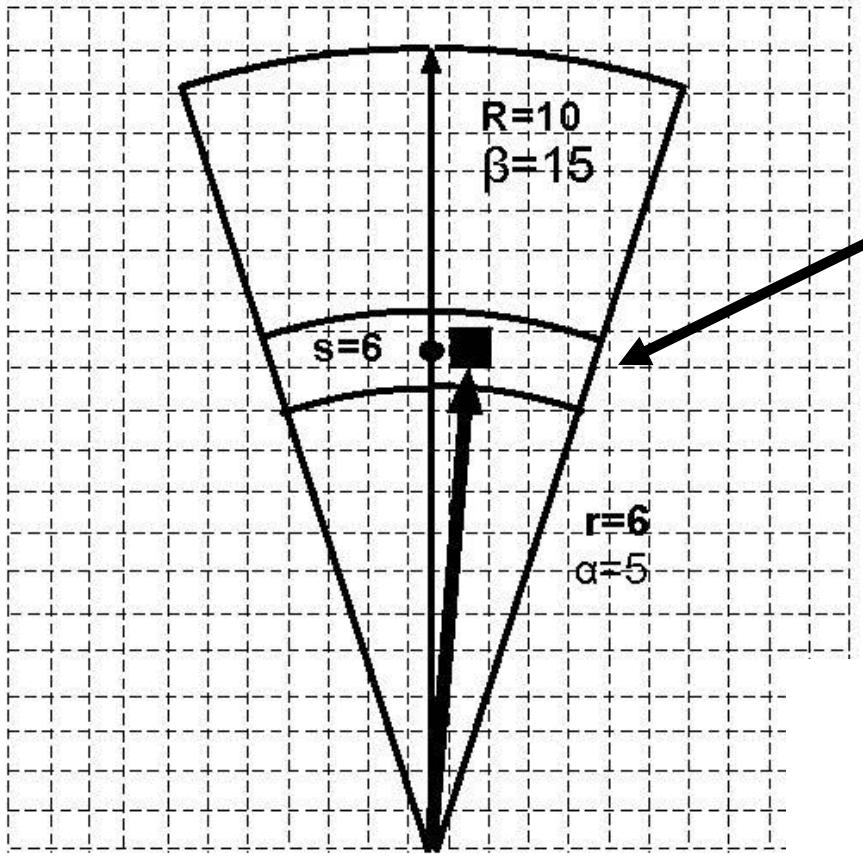
Region I: probable obstacle
R = max sensor range
β = Beam width (half-angle)
(r,α) = polar coordinates of grid point measured from sonar
s = sensor distance reading
ε = tolerance band for distance reading
Max_occupied = Maximum certainty of obstacle (0.0 – 1.0)

If Grid[i][j] iff: (r,a) within cone of uncertainty and

$$s - \varepsilon < r < s + \varepsilon$$

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

Mapping Sonar values to Occ. grid values: Region II



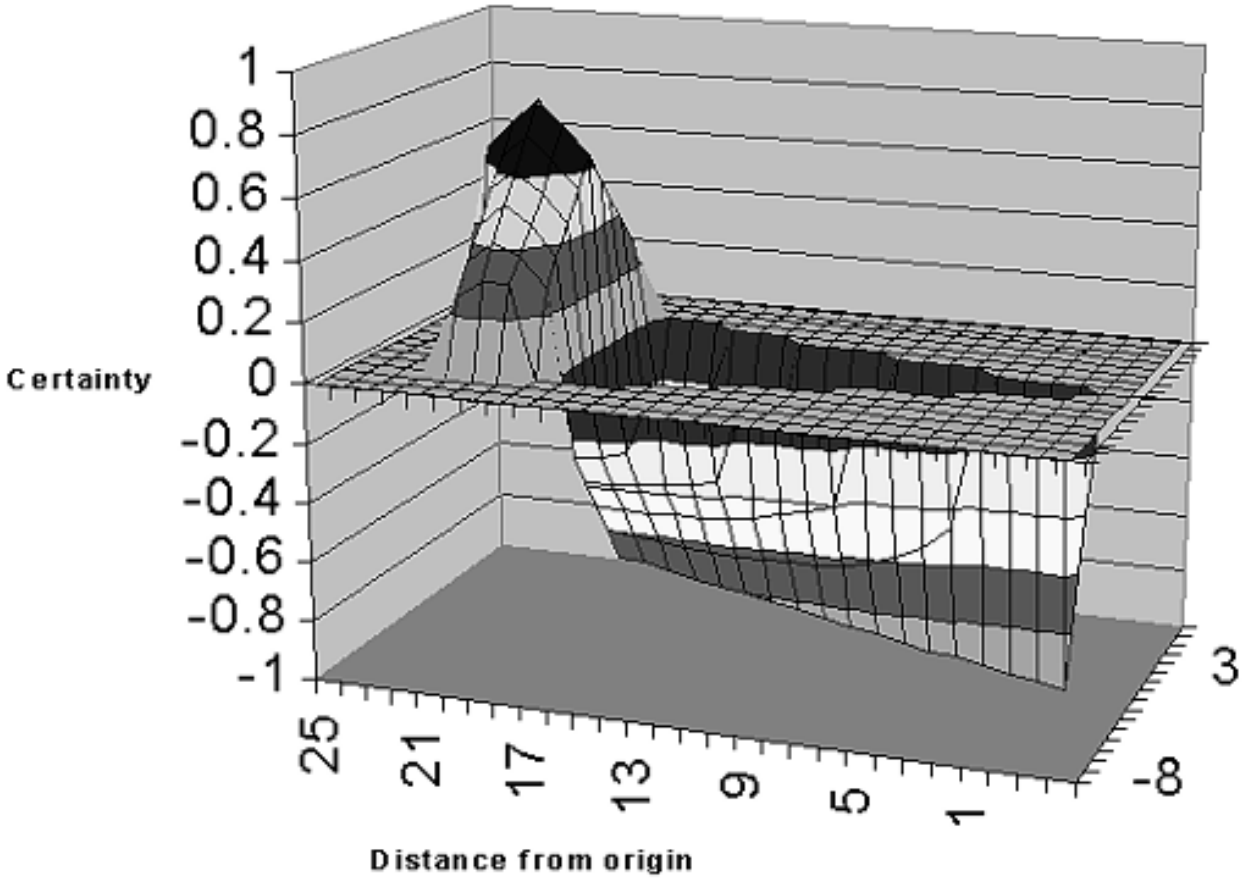
Region II: probable free-space
R = max sensor range
β = Beam width (half-angle)
(r,α) = polar coordinates of grid point measured from sonar
s = sensor distance reading
ε = tolerance band for distance reading
Max_occupied = Maximum certainty of obstacle (0.0 – 1.0)

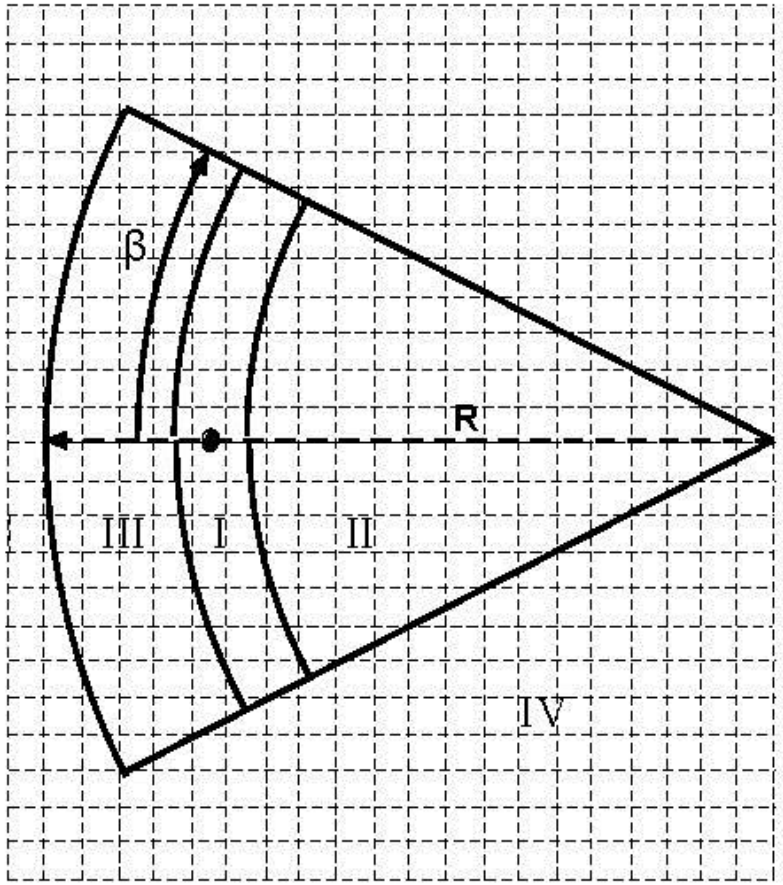
Update Grid[i][j] iff: (r,a) within cone of uncertainty and

$$r < s - \varepsilon$$

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

Region III: Unknown, don't update these cells!





Region I: Probable obstacle Region II: Probable free-space Region III: unknown

Bayes Rule for Sonar Updates

We want to find out the probability of the cell being occupied, given the new sensor reading s, and also knowing our **prior** probability of the cell being occupied:

Conditional probabilities for $P(H|s)$

The sensor model represents $P(s|H)$: the probability that the sensor would return the value being considered given it was really occupied. Unfortunately, the probability of interest is $P(H|s)$: the probability that the area at $grid[i][j]$ is really occupied given a particular sensor reading. The laws of probability don't permit us to use the two conditionals interchangeably. However, Bayes' rule does specify the relationship between them:

$$P(H|s) = \frac{P(s|H)P(H)}{P(s|H)P(H) + P(s|\neg H)P(\neg H)}$$

Substituting in *Occupied* for H, Eqn. 11.3 becomes:

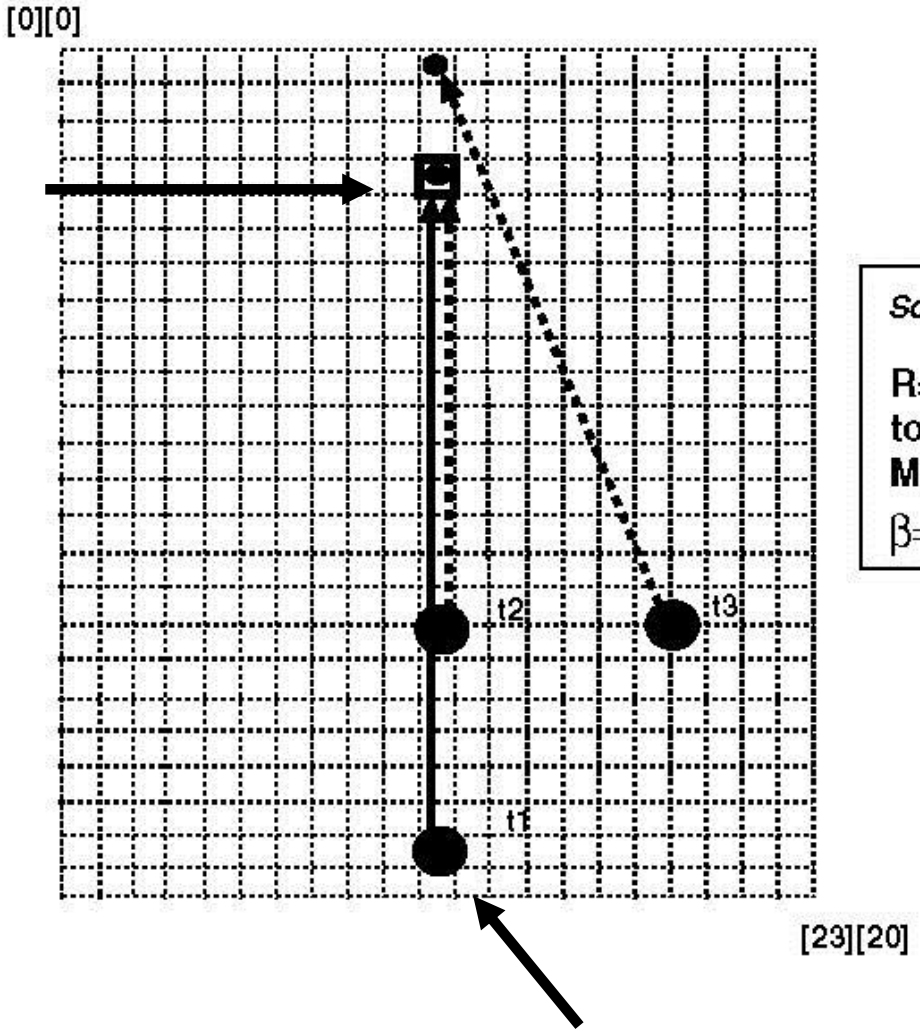
$$P(Occupied|s) = \frac{P(s|Occupied)\boxed{\text{P(Occupied)}}}{P(s|Occupied)\boxed{\text{P(Occupied)}} + P(s|Empty)\boxed{\text{P(Empty)}}}$$

$P(s|Occupied)$ and $P(s|Empty)$ are known from the sensor model. The other terms, $P(Occupied)$ and $P(Empty)$, are the unconditional probabilities, or *prior probabilities* sometimes called *priors*. The priors are shown in the boxes

Example: Initial Grid at t_0

- Occ. Grid of 24 x 21
- Each cell is 0.5 units square
- Robot at [21][10] at time t_1
- ε = 0.5 = tolerance
- Max_occupied = 0.98
- R =10 units = max sonar range

Obstacle at [3][10]



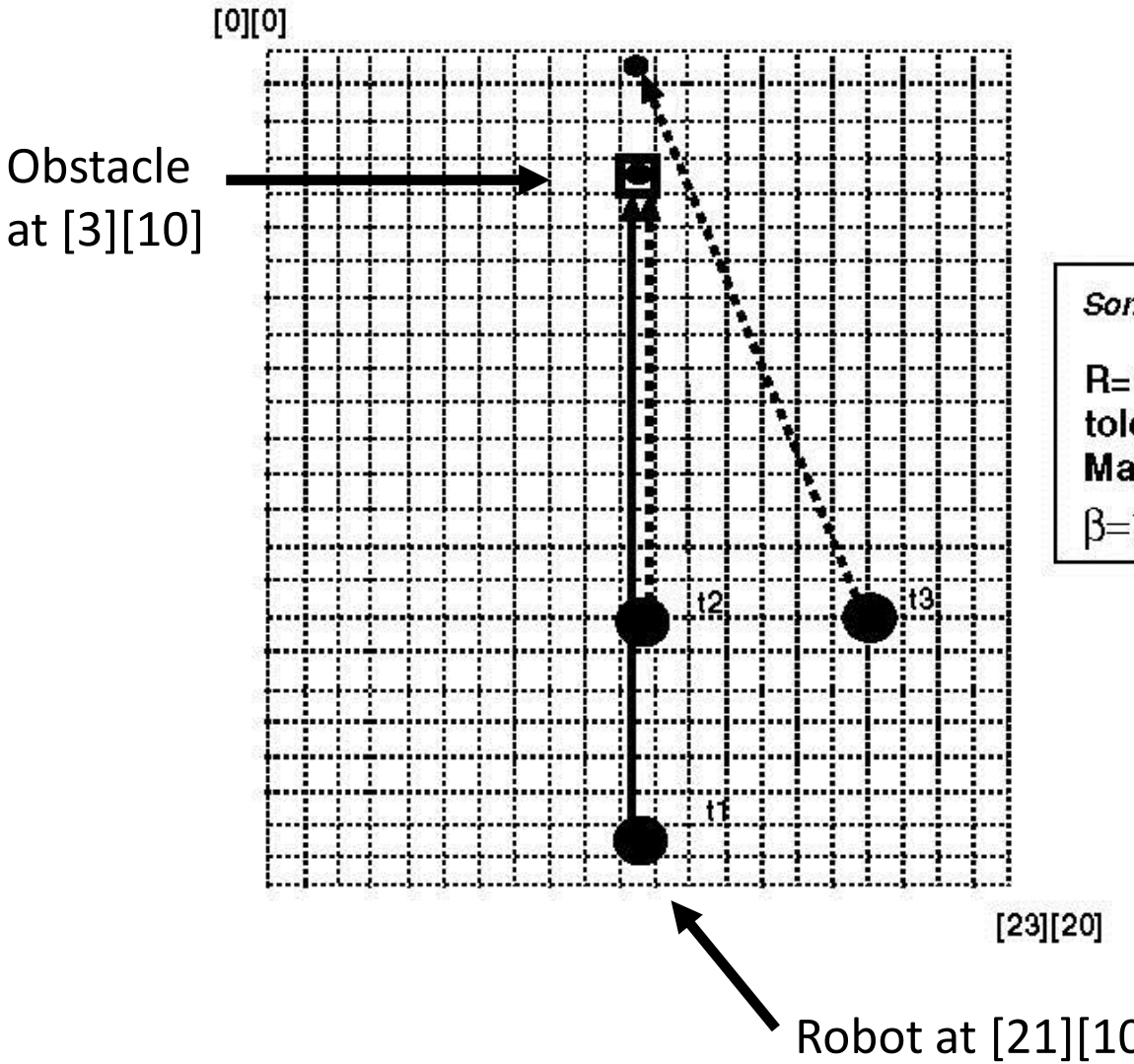
Sonar model parameters:

R=10
tolerance =+/- 0.5
Max_occupied = 0.98
β=15

At t_0, EVERY CELL IS INITIALIZED with P_occ = 0.5

Before we start sensing, every cell is equally Likely to be empty or contain an obstacle

Example: Sensor Reading at t_1



Sonar model parameters:

R=10
tolerance =+/- 0.5
Max_occupied = 0.98
β=15

- Occ. Grid of 24 x 21
- Each cell is 0.5 units square
- Robot at [21][10] at time t_1
- s = 9 units = sonar reading
- ε = 0.5 = tolerance
- Max_occupied = 0.98
- R =10 units = max sonar range
- Cell [3][10] is in region I (obstacle)
- Cell [3][10] is r=9 units from robot at Angle α = 0 – in cone of uncertainty:
- r is in range: s – ε < r < s + ε

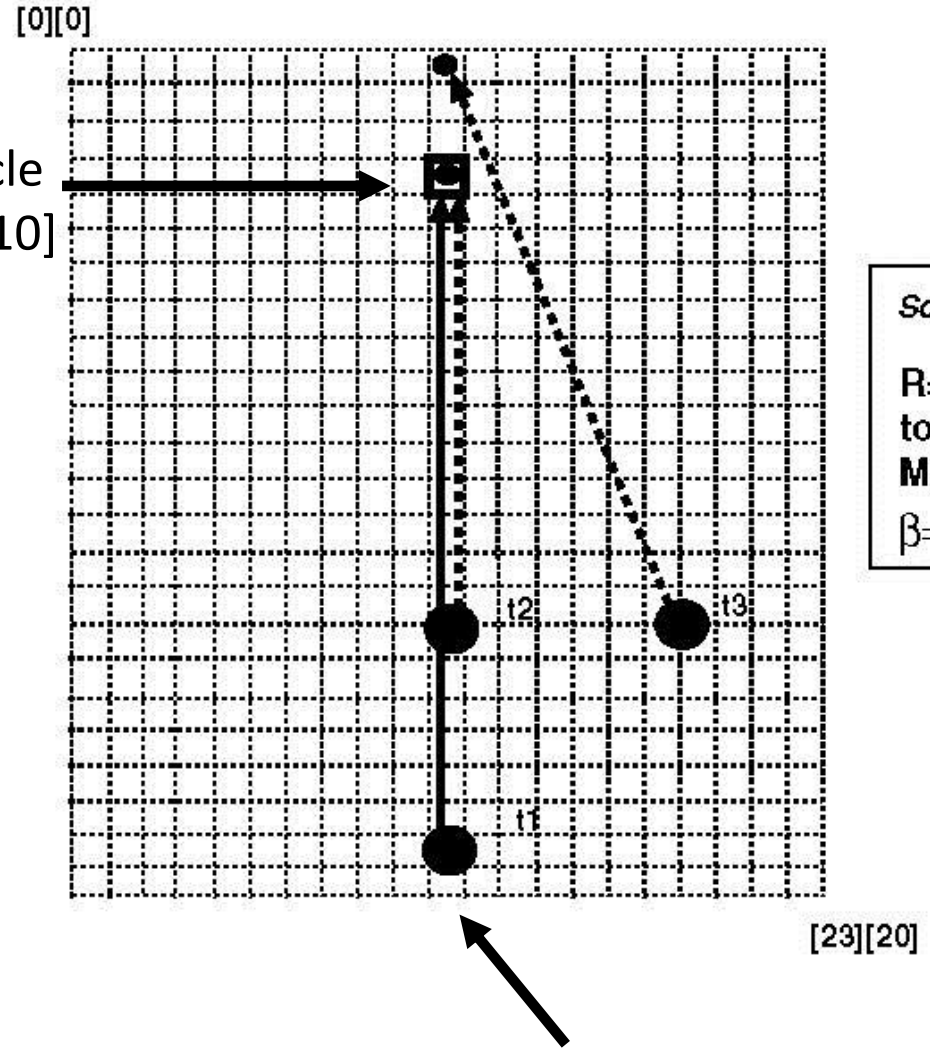
$$P(s|Occupied) = \frac{(\frac{R-r}{R}) + (\frac{\beta-\alpha}{\beta})}{2} \times Max_{occupied}$$

$$= \frac{(\frac{10-9}{10}) + (\frac{15-0}{15})}{2} \times 0.98 = 0.54$$

$$P(s|Empty) = 1.0 - P(s|Occupied)$$

$$= 1.0 - 0.54 = 0.46$$

Example: Bayes Rule Update at t_1 after sensor read



Sonar model parameters:

R=10
tolerance =+/- 0.5
Max_occupied = 0.98
β=15

- Occ. Grid of 24 x 21
- Each cell is 0.5 units square
- Robot at [21][10] at time t_1
- s = 9 units = sonar reading
- ε = 0.5 = tolerance
- Max_occupied = 0.98
- R =10 units = max sonar range

Cell [3][10] is in region I (obstacle)
Cell is 9 units from robot at
Angle α = 0

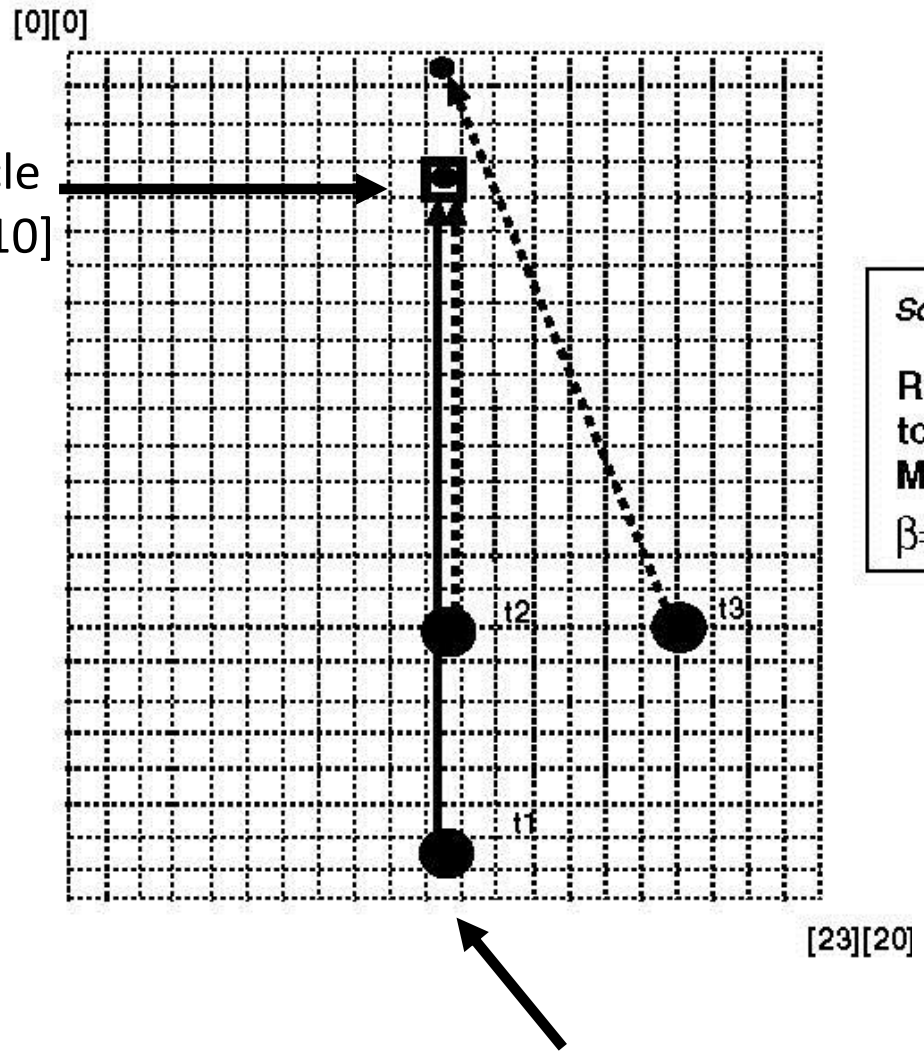
$$P(s_{t_1}|O) = 0.54$$
$$P(s_{t_1}|E) = 0.46$$
$$P(s_{t_0}|O) = 0.50$$
$$P(s_{t_0}|E) = 0.50$$

This yields:

$$P(O|s_{t_1}) = \frac{P(s_{t_1}|O)P(O|s_{t_0})}{P(s_{t_1}|O)P(O|s_{t_0}) + P(s_{t_1}|E)P(E|s_{t_0})} = \frac{(0.54)(0.50)}{(0.54)(0.50) + (0.46)(0.50)} = 0.54$$

$$P(E|s_{t_1}) = 1 - P(O|s_{t_1}) = 0.46$$

Example: Bayes Rule Update at t_2 after 2nd sensor read



Sonar model parameters:

R=10
tolerance =+/- 0.5
Max_occupied = 0.98
β=15

- Robot at [15][10] at time t_2
- s = 6 units = sonar reading
- ε = 0.5 = tolerance
- Max_occupied = 0.98
- R =10 units = max sonar range

Cell [3][10] is in region I (obstacle)
Cell is 6 units from robot at
Angle α = 0

$$P(s_{t2}|Occupied) = 0.69$$
$$P(s_{t2}|Empty) = 0.31$$

$$P(O|s_{t_2}) = \frac{P(s_{t_2}|O)P(O|s_{t_1})}{P(s_{t_2}|O)P(O|s_{t_1}) + P(s_{t_2}|E)P(E|s_{t_1})}$$
$$= \frac{(0.69)(0.54)}{(0.69)(0.54) + (0.31)(0.46)}$$
$$= 0.72$$
$$P(E|s_{t_2}) = 1 - P(O|s_{t_2}) = 0.28$$

Bayes Updating

- Cell [3][10] had P(occ) = 0.5 at t_0
- Cell [3][10] had P(occ) = 0.54 at t_1 after first sensor read
- Cell [3][10] had P(occ) = 0.72 at t_2 after second sensor read
- Successive sensor readings provide confirmation of obstacle
- Note: can use other sensors to update the grid (e.g stereo vision)
- Note: need to update cells in Region II (freespace) as well!

Map Learning and High Speed Navigation in RHINO

Sebastian Thrun, Arno B¨ucken Wolfram Burgard Dieter Fox,Thorsten Fr¨ohlinghaus Daniel Hennig Thomas Hofmann Michael Krell Timo Schmidt

An indoor mobile robot that uses sonar and vision to map its environment in real-time

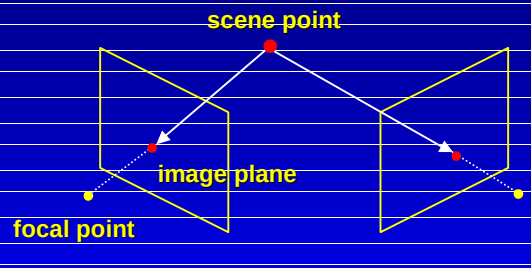
Case Study: Map Learning and High Speed Navigation in RHINO

- Control is distributed and decentralized. Onboard and offboard machines are dedicated to several subproblems of modeling and control. Communication between modules is asynchronous - no central clock, and no central process controlller.
- Whenever possible, anytime algorithms are employed to ensure that the robot operates in realtime.
- Hybrid architecture. Fast, reactive mechanisms are integrated with computationally intense, deliberative modules.
- Models, such as the two dimensional maps described below, are used at all levels of architecture.
- Whenever possible, models are learned from data.
- Machine learning algorithms are employed to increase the flexibility and the robustness of the system. Learning has proven most useful close to the sensory side of the system, where algorithms such as artificial neural networks interpret the robot's sensors.
- Software is modular. A plug and play architecture allows us to quickly reconfigure the system, depending on the particular configuration and application.
- Sensor fusion. To maximize the robustness of the approach, most of the techniques described here rely on more than just a single type of sensor.

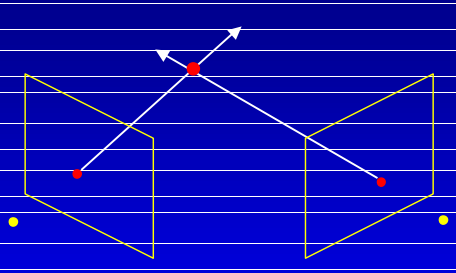
Sonar Data Interpretation

- Need to translate sonar distances into occupancy values: Prob(occ(x,y))
- Method: Neural Net, trained on sonar responses
- RHINO uses a 360^{o} ring of sonars
- Input to net: 4 readings nearest (x,y) – encoded as polar coordinates
- Output: Prob(occ(x,y))
- Training data: train with physical robot on real known environments or use robot simulator
- May need to train anew in different environments –wall textures etc.
- Key point: multiple spatial readings needed to overcome noise and sonar effects

Stereo



Stereo



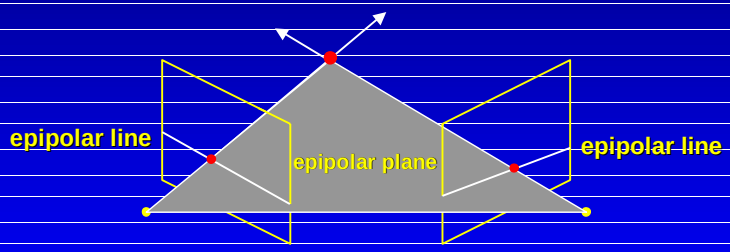
Basic Principle: Triangulation

- Gives reconstruction as intersection of two rays
- Requires ***point correspondence***

Stereo Correspondence

Determine Pixel Correspondence

- **Pairs of points that correspond to same scene point**



Epipolar Constraint

- **Reduces correspondence problem to 1D search along *conjugate epipolar lines***

Stereo Matching Algorithms

Match Pixels in Conjugate Epipolar Lines

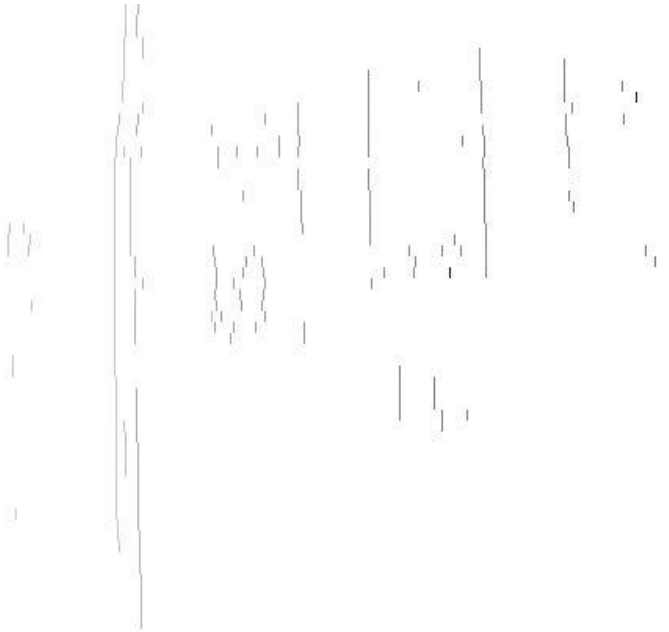
- **Assume color of point does not change**
- **Pitfalls**
  - > specularities
  - > low-contrast regions
  - > occlusions
  - > image error
  - > camera calibration error
- **Numerous approaches**
  - > dynamic programming [Baker 81,Ohta 85]
  - > smoothness functionals
  - > more images (trinocular, N-ocular) [Okutomi 93]
  - > graph cuts [Boykov 00]

Stereo Data Interpretation

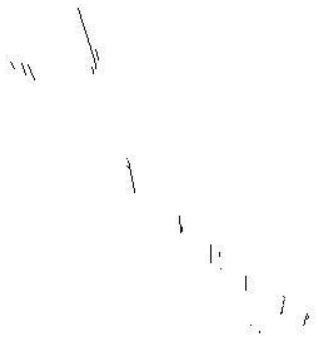
- Vertical edges (doorways, vertical corners, obstacles) are found in each image and triangulated for 3D depth.
- 3D edge points are projected onto the occupancy grid after being enlarged by the robot radius
- Enlargement allows robot to navigate without hitting corners
- Stereo can miss featureless, homogeneous areas like blank walls
- Integration with sonar can improve mapping accuracy

image

Vertical edges



Vertical edge projection



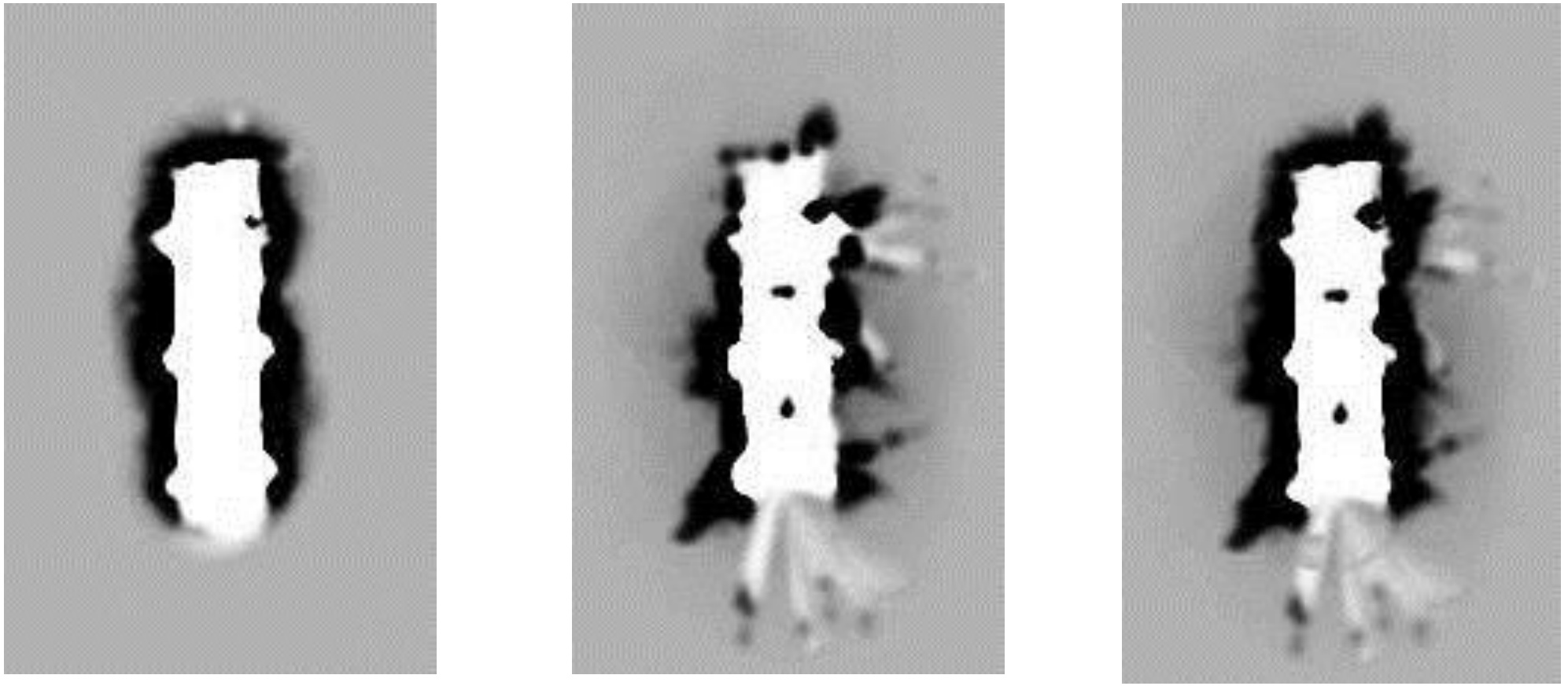
Occupancy grid

Results from Stereo matching

Updating over time

- Mobile robot is moving and making multiple measurements at each sensing time step
- Need to integrate the new values from the sensors with the current occupancy grid values
- These are probabilistic measures, so a Bayes rule update is used to find new probability of occupancy (more on this later….)

Integration results



Maps built in a single run (a) using only sonar sensors, (b) using only stereo information, and (c) integrating both. Notice that sonar models the world more consistently, but misses the two sonar absorbing Chairs which are found using stereo vision.

Topological Maps

- Compact representation, usually as a graph
- Nodes are distinct places, arcs (edges) represent adjacency
- Regions of free-space are nodes, and edges represent connections for adjacency
- Method:
  - Create Voronoi diagram
  - Find critical points – bottlenecks or choke points in the Voronoi diagram
  - Formally, a threshold *epsilon* for minimum distance to obstacle locally
  - Find critical lines: partitions between regions at bottlenecks
  - Partitions are used to form a graph. Nodes are regions, arcs are adjacent regions separated by critical lines