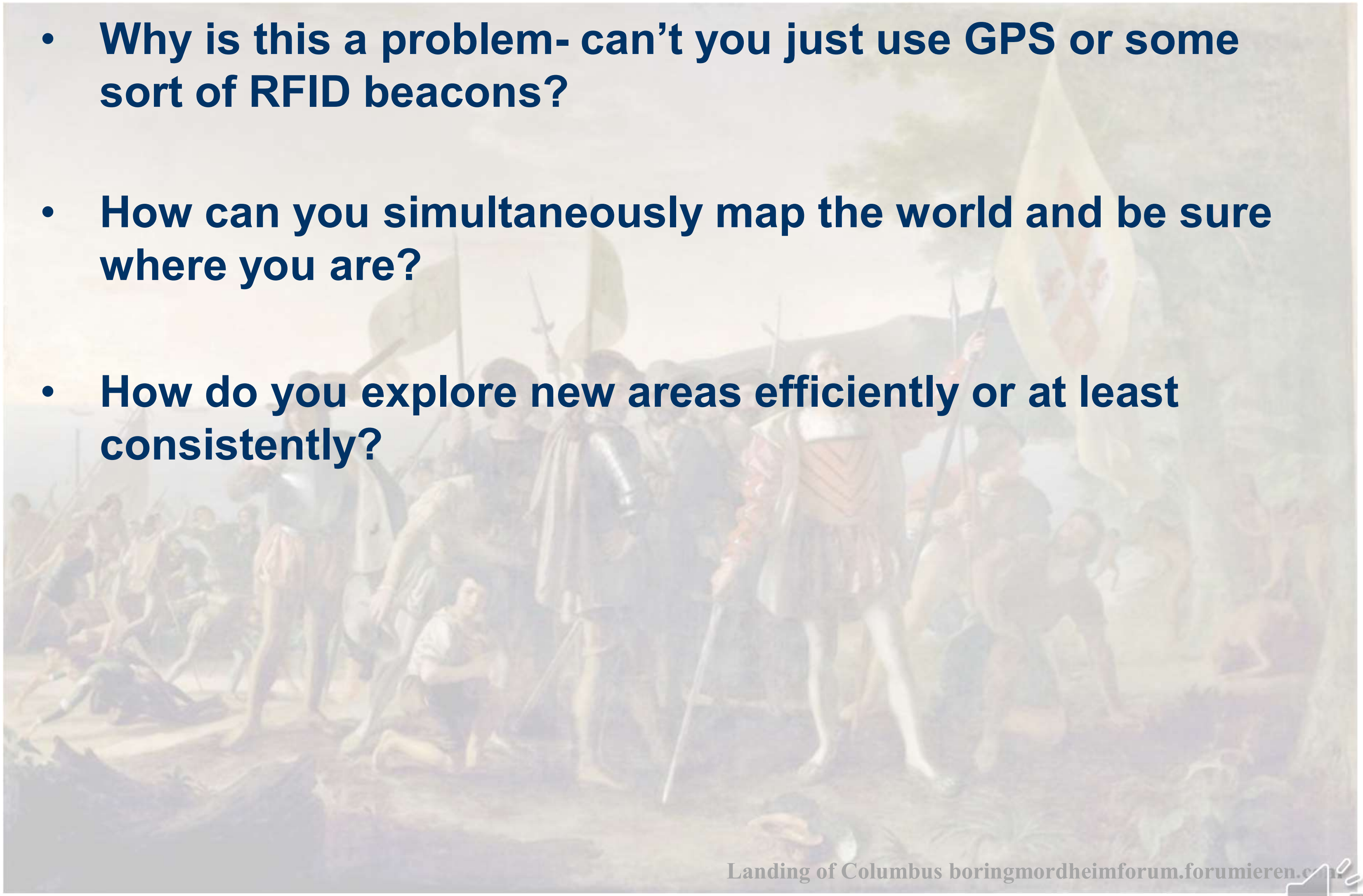
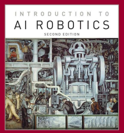
15a Localization, Mapping, Exploration



- Why is this a problem- can't you just use GPS or some sort of RFID beacons?
- How can you simultaneously map the world and be sure where you are?
- How do you explore new areas efficiently or at least consistently?

Landing of Columbus boringmordheimforum.forumieren.com

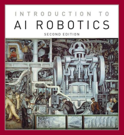


15a Specific Learning Objectives

- Define the pose of a mobile robot.
- Explain the difference between localization, mapping, and exploration.
- Define each term in:

$$bel(x_t)=f(bel(x_{t-1}), u_t, z_t, m).$$

- Describe the difference between localization and simultaneous localization and mapping (SLAM).
- Define the loop closure problem.
- Compare and contrast frontier-based and generalized Voronoi graph (GVG) based exploration

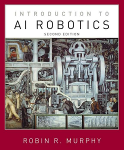


Localization (Position Estimation)

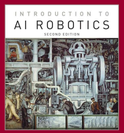
- "Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment"- Thrun, Burgard, Fox
  - It is often called position estimation

- The pose of a robot is its position plus its orientation.
  - For two dimensions, the pose is $\underline{(x, y, \theta)^T}$

- Assumes that there is some sort of *a priori* map and the robot is localizing itself relative to it

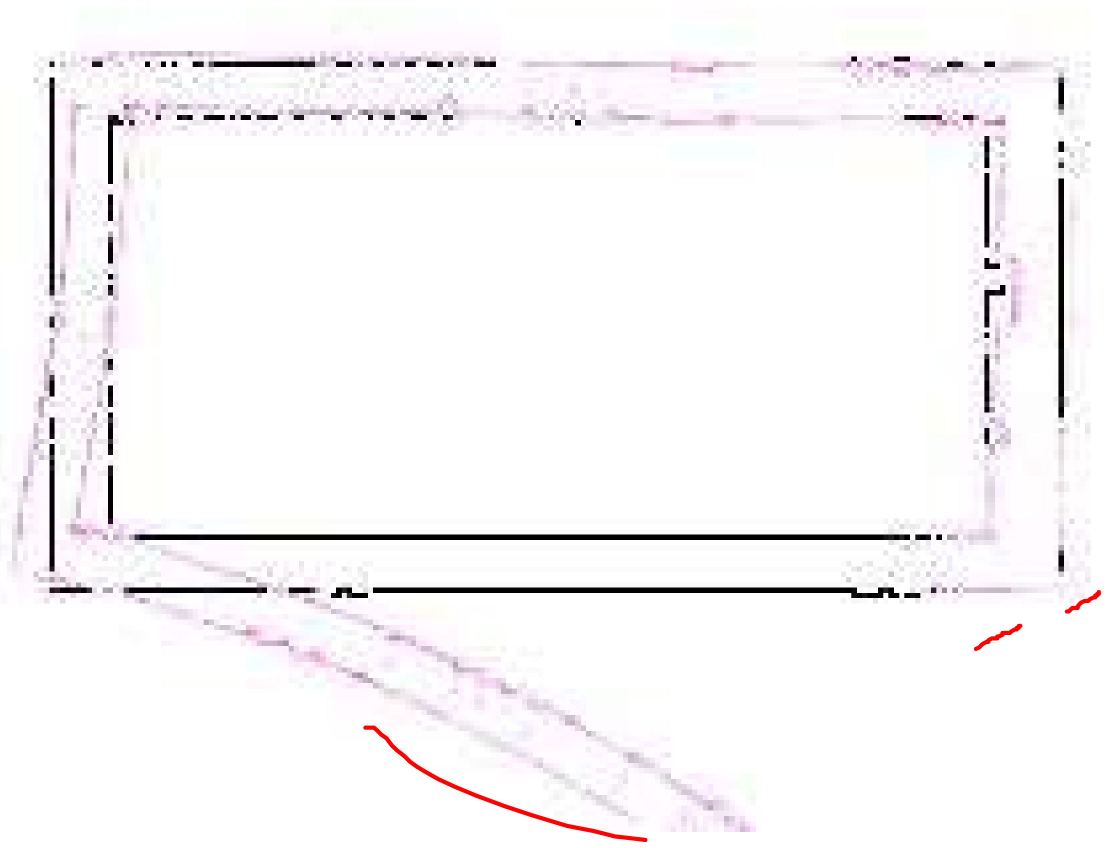


What makes localization hard

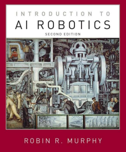


- It requires a model that incorporates the robot's actuators, its sensors, and the accuracy and repeatability of those sensors in different environments.
- There may be sensor noise, either proprioceptive and exteroceptive, which introduces uncertainty.
- Methods for localization are computationally expensive.
- In some localization problems, the robot may not know its initial position.
- The robot may be operating in a dynamic work environment where objects move, causing the robot to have to consider whether sensor readings are a result of the robot moving or Object A moving.
- The path or task may not support localization. For example, if the robot travels across a large, open, featureless warehouse, it may not be able to sense any features to localize against.

Why Odometry is Insufficient



Black
Is ground
Truth,
Purple is
Measured
Using shaft
Encoders for
Δ

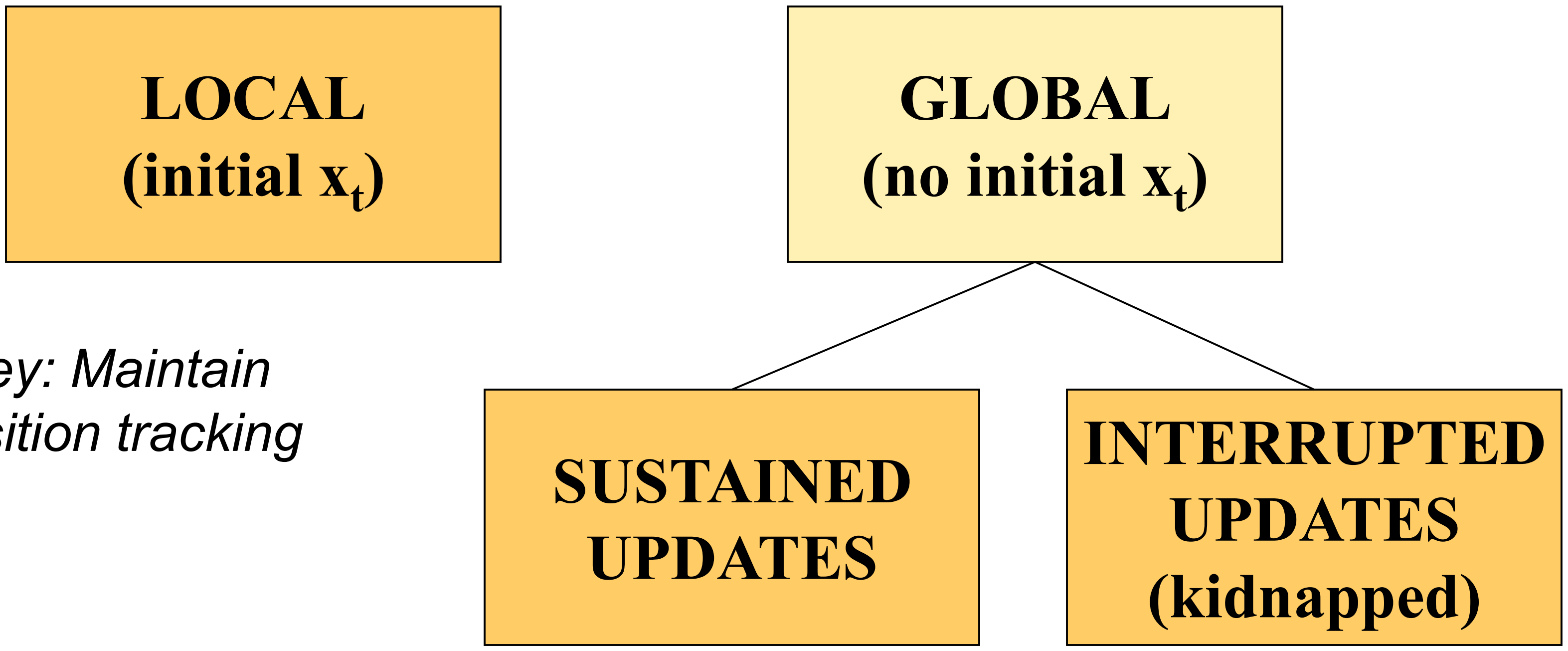


3 Types of Localization

is not the same as localization algorithms

Position Tracking

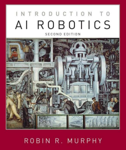
Global localization



Key: Maintain position tracking

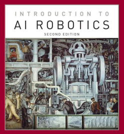
Key: Converge to right place quickly

Key: Notice that world is different



Two Approaches

- **Feature-based**: use features extracted from raw data
  - Label and match corners, walls, whatever
  - Less features, so less computation
  - Hard to extract features reliably
- **Iconic**: use raw (or near raw) sensor readings
  - Match observations to what would expect to see at a location
  - Computationally intensive



Probabilistic, Map-Based Localization

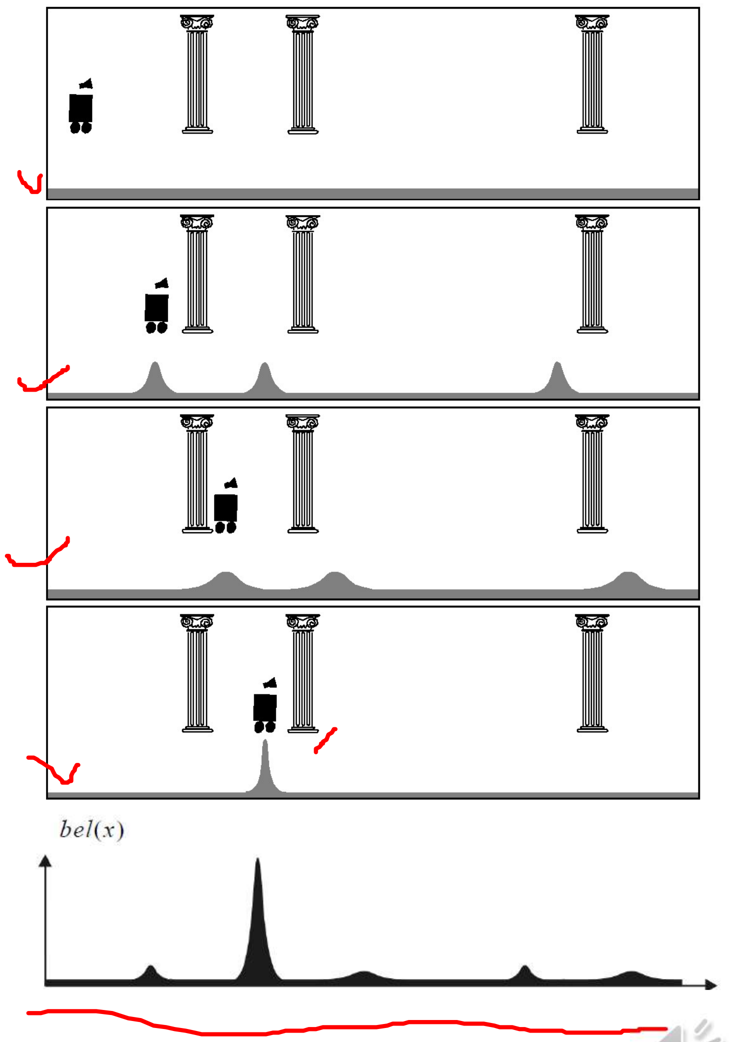
- The key idea in probabilistic robotics is to represent uncertainty using probability theory: instead of giving a single best estimate of the current robot configuration, probabilistic robotics represents the robot configuration as a probability distribution over the all possible robot poses. This probability is called belief.

The probabilistic localization problem

- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.
- Due to odometry uncertainty, after some movement the robot will become very uncertain about its position.
- To keep position uncertainty from growing unbounded, the robot must localize itself in relation to its environment map. To localize, the robot uses its on-board exteroceptive sensors (e.g. ultrasonic, laser, vision sensors) to make observations of its environment
- The robot updates its position based on the observation. Its uncertainty shrinks.

Working Principle: Improving belief state by moving

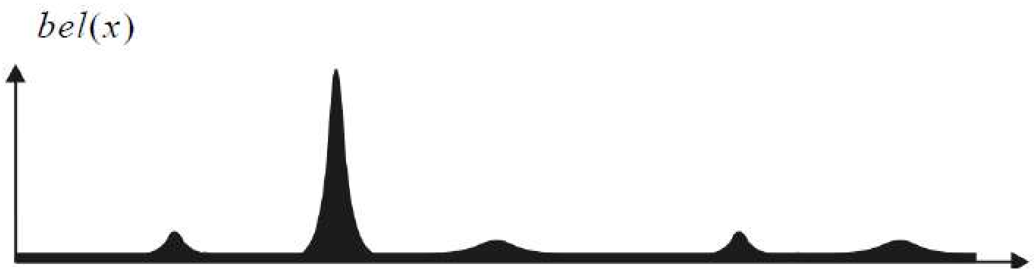
- The robot is placed somewhere in the environment but is not told its location
- The robot queries its sensors and finds it is next to a pillar
- The robot moves one meter forward. To account for inherent noise in robot motion the new belief is smoother
- The robot queries its sensors and again it finds itself next to a pillar
- Finally, it updates its belief by combining this information with its previous belief



Why a probabilistic approach for mobile robot localization?

- Because the data coming from the robot sensors are affected by measurement errors, we can only compute the probability that the robot is in a given configuration.

- The key idea in probabilistic robotics is to represent uncertainty using probability theory: instead of giving a single best estimate of the current robot configuration, probabilistic robotics represents the robot configuration as a probability distribution over the all possible robot poses. This probability is called *belief.*



The two update steps in robot localization

1. ACTION (or prediction) update: the robot moves and estimates its position through its proprioceptive sensors. During this step, the robot uncertainty grows. This step uses the Total Probability formula to update belief

$$\underline{\overline{bel}(x_t)} = \underline{p(\underline{x_t} \mid \underline{u_t}, \underline{x_{t-1}})} \underline{*} \underline{bel(x_{t-1})}$$

$$\boxed{p(x) = \int_y p(x|y)p(y)dy}$$

2. PERCEPTION (or measurement) update: the robot makes an observation using its exteroceptive sensors and correct its position by opportunely combining its belief before the observation with the probability of making exactly that observation. During this step, the robot uncertainty shrinks. This step uses the Bayes Rule to update belief.

$$\underline{bel(x_t)} = \underline{\eta} \cdot \underline{p(z_t \mid x_t)} \cdot \underline{\overline{bel}(x_t)}$$

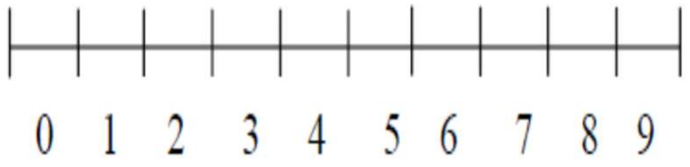
$$\boxed{p(x|y) = \frac{\boxed{p(y|x)}\,p(x)}{\underline{p(y)}}}$$

Markov versus Kalman localization

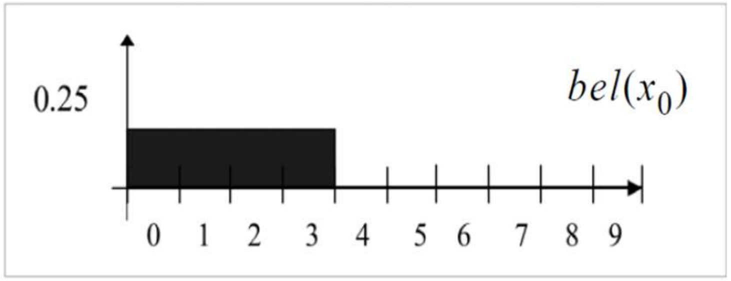
- Two approaches exist to represent the probability distribution and to compute the Total Probability and Bayes Rule during the Action and Perception phases
- Markov approach:
  - The configuration space is divided into many cells. Each cell contains the probability of the robot to be in that cell.
  - The probability distribution of the sensors model is also discrete.
  - During Action and Perception, all the cells are updated. Therefore, the computational cost is very high.
- Kalman approach:
  - The probability distribution of both the robot configuration and the sensor model is assumed to be continuous and Gaussian!
  - Need only to update mean value μ and covariance Σ. Therefore the computational cost is very low!

Markov localization example

- Consider a robot moving in a one dimensional environment (e.g. moving on a rail). Therefore the robot configuration space can be represented through the sole variable x.
- Let us discretize the configuration space into 10 cells



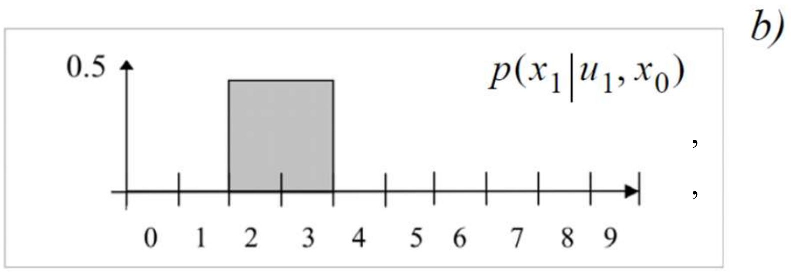
- Initial belief distribution



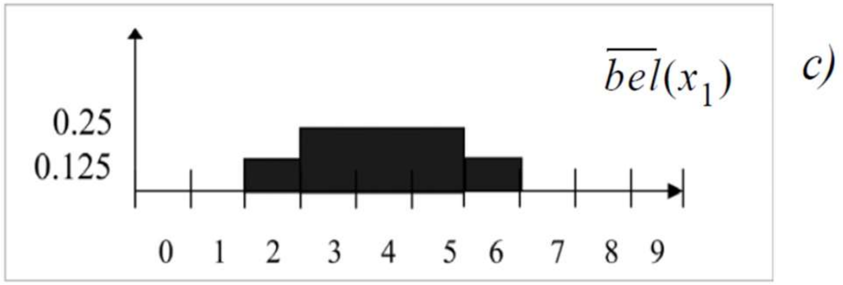
a)

the Action update

- Let us assume that the robot moves forward with the following statistical model

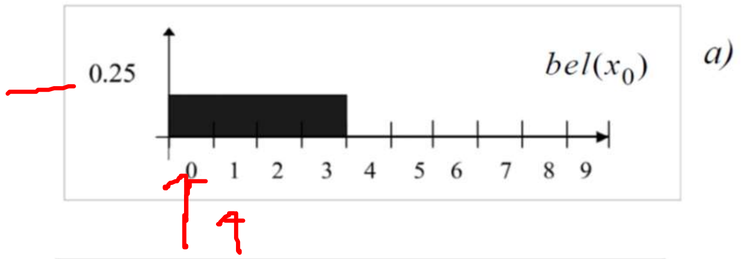


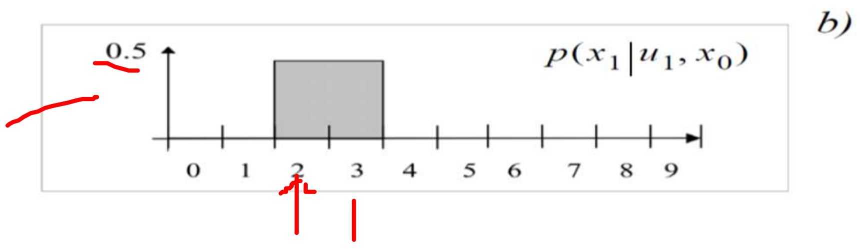
- This means that we have 50% probability that the robot moved 2 or 3 cells forward
- The probability (belief) be after the motion (the Action update)

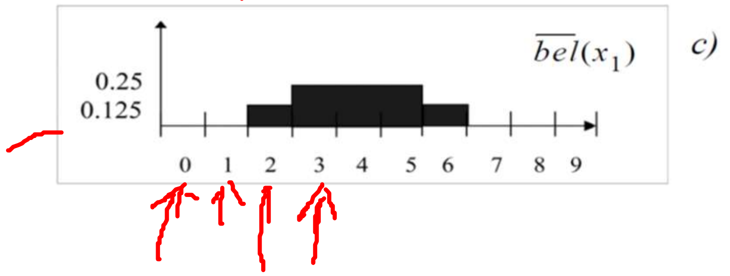


$$\overline{Bel}(x_1 = 0) = 0$$
$$\overline{Bel}(x_1 = 1) = 0$$
$$\overline{Bel}(x_1 = 2) = .125$$
$$\overline{Bel}(x_1 = 3) = 0.25$$

the Action update: Computing details







$$\overline{Bel}(x_1 = 2) = p(x_0 = 0)p(u_1 = 2)$$
$$= \frac{1}{4} \times \frac{1}{2} = \frac{1}{8} = 0.125$$

$$\overline{Bel}(x_1 = 3) = p(x_0 = 0)p(u_1 = 3) + p(x_0 = 1)p(u_1 = 2) = \frac{1}{4} \times \frac{1}{2} + \frac{1}{4} \times \frac{1}{2} = 0.25$$

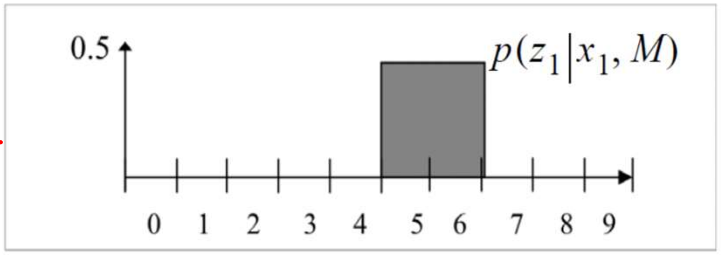
$$\overline{Bel}(x_1 = 4) = p(x_0 = 1)p(u_1 = 3) + p(x_0 = 2)p(u_1 = 2) = \frac{1}{4} \times \frac{1}{2} + \frac{1}{4} \times \frac{1}{2} = 0.25$$

$$\overline{Bel}(x_1 = 5) = p(x_0 = 2)p(u_1 = 3) + p(x_0 = 3)p(u_1 = 2) = \frac{1}{4} \times \frac{1}{2} + \frac{1}{4} \times \frac{1}{2} = 0.25$$

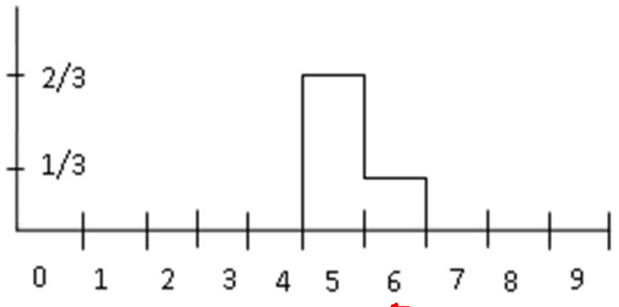
$$\overline{Bel}(x_1 = 6) = p(x_0 = 3)p(u_1 = 3) = \frac{1}{4} \times \frac{1}{2} = \frac{1}{8} = 0.125$$

The Perception (or measurement) update

- Let us now assume that the robot uses its onboard range finder and measures the distance from the origin. Assume that the statistical error model of the sensors is



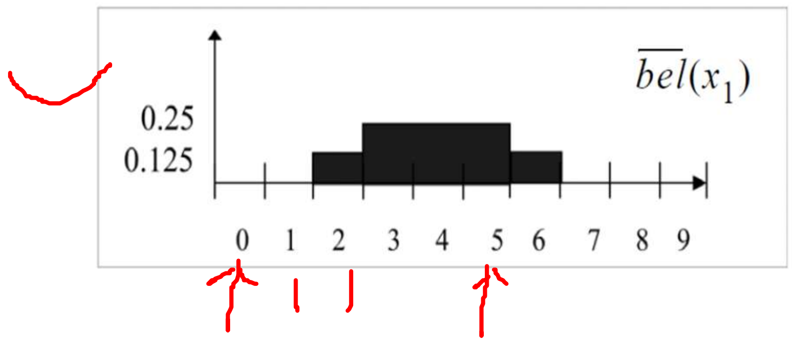
- This plot tells us that the distance of the robot from the origin can be equally 5 or 6 units
- The probability (belief) be after this measurement

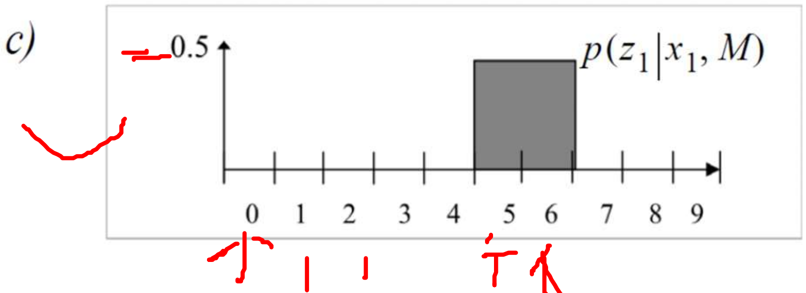


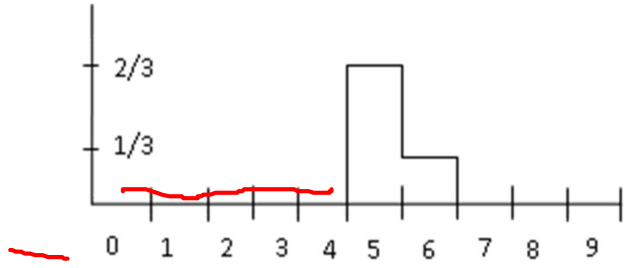
$$\boxed{Bel(x_1 = 5) = {}^{2}/_{3}}$$

$$\boxed{Bel(x_1 = 6) = {}^{1}/_{3}}$$

The Perception update: Computing details







$$\boxed{Bel(x_1 = 5) = {}^2/_3}$$

$$\boxed{Bel(x_1 = 6) = {}^1/_3}$$

$$Bel(x_1 = 5) = \eta \cdot \overline{Bel}(x_1 = 5)p(z_1 = 5) = \eta \cdot \frac{1}{4} \times \frac{1}{2} = \eta \cdot \frac{1}{8}$$

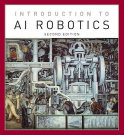
$$Bel(x_1 = 6) = \eta \cdot \overline{Bel}(x_1 = 6)p(z_1 = 6) = \eta \cdot \frac{1}{8} \times \frac{1}{2} = \eta \cdot \frac{1}{16}$$

$$Bel(x_1 = 5) + Bel(x_1 = 6) = 1 \Rightarrow \eta = 16/3$$

15a

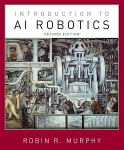
Feature-Based Algorithms

Motivation
Local
-Markov
--EKF
Global
-Grid
-MCL
Dynamic
Summary



- **Markov Localization** covers all three problems, though usually used for local localization (position tracking)
- $\text{bel}(x_t) = f(\text{bel}(x_{t-1}), u_t, z_t, m)$, where
  - **bel(x)** : belief that the robot is at pose **x**
    - **Remember pose for 2D is: $\mathbf{x}=(x\ y\ \theta)^T$**
  - $\mathbf{u_t}$: control actions or *what you told the robot to do*
  - $\mathbf{z_t}$: measurements or *what the robot observed*
  - **m** : map

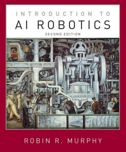
Two Iconic Methods (1)



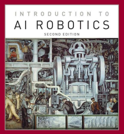
- Grid-based localization
  - Uses a grid as a tessellation of convex polygon
    - (Other algorithms use regular grid as way to store data)
  - Computes the likelihood of all possible poses within that polygon given the observation and puts in a histogram
- The advantage
  - The grid acts to reduce the computational complexity by discretizing the space.
  - In practice, grid-based localization algorithms often further reduce complexity by restricting matching to a local “sub-map”
- The disadvantage
  - Sub-map approach works only for local localization

Two Iconic Methods (2)

- Grid-based localization
- Monte Carlo Localization (MCL)
  - Technically it is not restricted to raw sensor observations, but is typically used this way
  - Scatters “particles” throughout space, then computes the belief that the robot is at the pose given the observation
  - Adds more particles at the next update, while some particles “die” if they have a low probability
  - Fun fact: noise actually helps this method!



15a Static v. Dynamic Environments

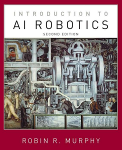


- People, changes introduce a "hidden state" in the model, therefore the algorithms break

- Two fundamental techniques
  - **State augmentation**:
    - Try to make the hidden state unhidden
    - Have to estimate impact of people on observations
  - **Outlier rejection**
    - Use knowledge about the sensors and the environment to eliminate suspicious readings
    - Ex. Eliminate surprisingly short readings
    - Depends on sensors, environment

15a

Summary of Localization

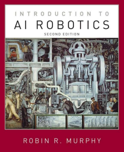


- Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment
- Localization algorithms are variants of Bayes Filter algorithms, where the belief in a pose is a function of the last pose and a motion model
  - Motion model must contain at a minimum pose, control actions, map
- Iconic localization is more common than feature-based due to problems in reliably extracting features
- In practice, Monte Carlo methods (MCL) dominate localization, both for iconic and feature-based
- Localization works very well for static environments with good models, but dynamic environments are difficult. State augmentation is incomplete and outlier detection hard to quantify
- The results rely heavily on the ability to accurately sense the environment
  - Math is there, but beyond Sick laser is problematic

15a

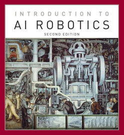
SLAM (Mapping)

- Simultaneous localization and mapping
- Recall that localization requires an a priori map
- Mapping is when you don't have a map, but you probably don't have localization either…
- Thus, in practice, mapping means that the robot has to build a map and simultaneously localize itself to that map as it goes
- The new map represents the maximum likelihood of the world, that is, the map represents the highest probability of what the world really is
- Rao-Blackwellized Filtering is a dominant method in SLAM
  - YouTube: https://www.youtube.com/watch?v=ZO83858FyaE



Rao-Blackwellized Filtering

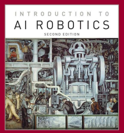
- If you know the path that the robot has taken, computing the maximum likelihood map is easy
  - It's the map where the observations are consistent
  - But you don't know the path, because that would be localization…
- Let each particle contain a path and a local map (instead of only one version of the map)
  - After each observation, update only the sensed area of the maps
  - Use a tree to save all the particles forming the history of the current particles
  - Each particle is computed independently of the others
- Will get pretty good map as go along, though probably several possible paths
  - When "close the loop" will get a better path estimate (closure will propagate backwards)



15a

The loop closure problem

- The mechanism by which a robot notices that it has returned to a previously visited place is called loop closure
- Four main approaches to resolving the loop closure problem
  - Feature matching: Match the geometric features within the map.
  - Sensor scan matching: Does the sensor scan match a complete profile of the area?
  - Hybrid feature-scan matching: Do visible features and the range readings have the same profile?
  - Expectation matching: Is it statistically likely that the robot has returned to the starting location?

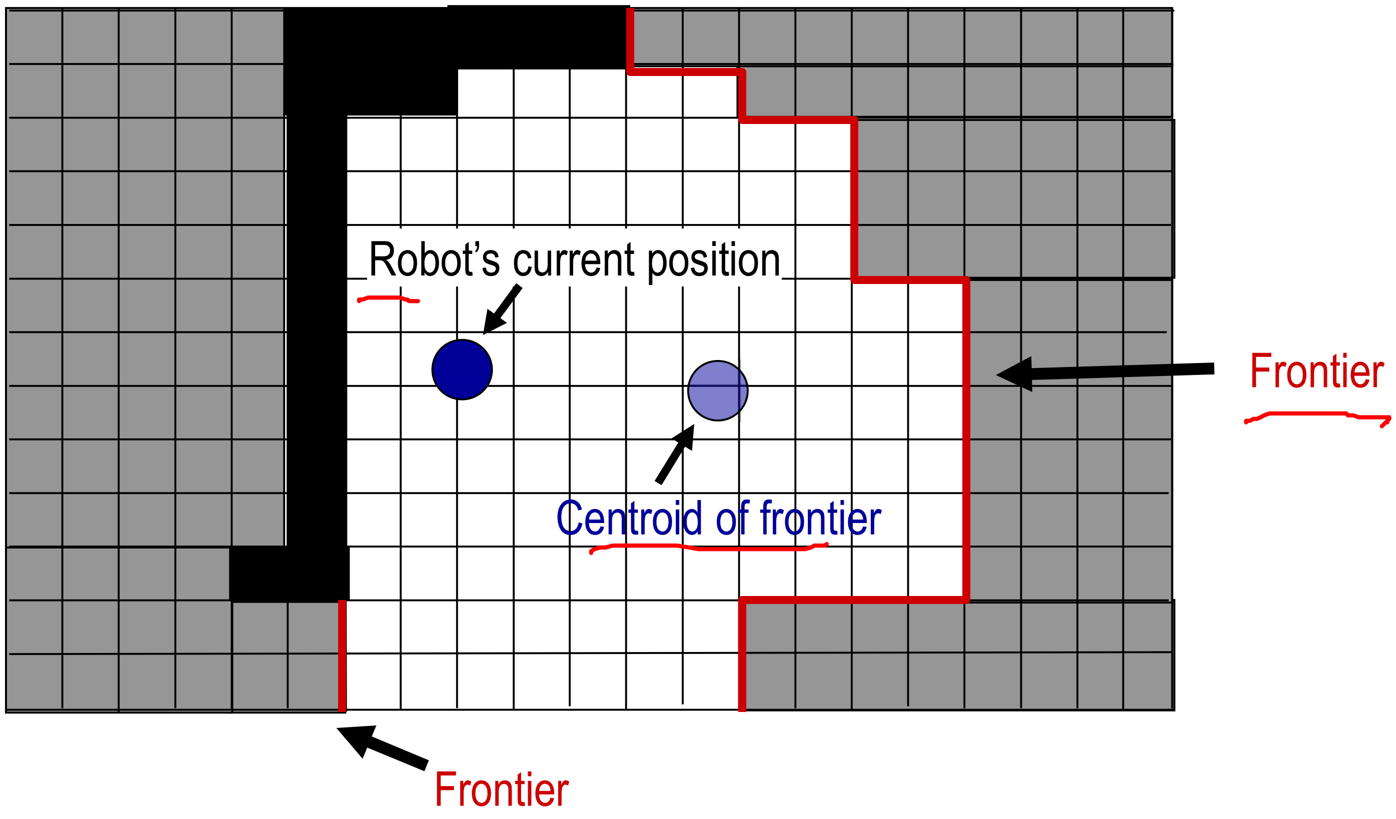


Exploration

- Key question: Where hasn't robot been?
- Central concern: how to cover an unknown area efficiently
- Possible approaches:
  - Random walk
  - Use proprioception to avoid areas that have been recently visited
  - Exploit evidential information in the occupancy grid
- Two basic styles of exploration:
  - Frontier-based
  - Generalized Voronoi graph
- Both methods work OK indoors, not so clear on utility outdoors
  - GVG: Susceptible to noise, hard to recover nodes
  - Frontier: Have to rate the frontiers so don't trash

Frontier-Based Exploration

- Assumes robots uses Bayesian occupancy grid
- When robot enters new area, find boundary ("Frontier") between sensed (and open) and unsensed areas
- Head towards centroid of Frontier



Calculating Centroid

- Centroid is average (x,y) location:

```
x_c = y_c = count = 0

for every cell on the frontier line with a location of (x,y)
 {   x_c = x_c + x
     y_c = y_c + y
     count++
 }

x_c = x_c/count
y_c = y_c/count
```

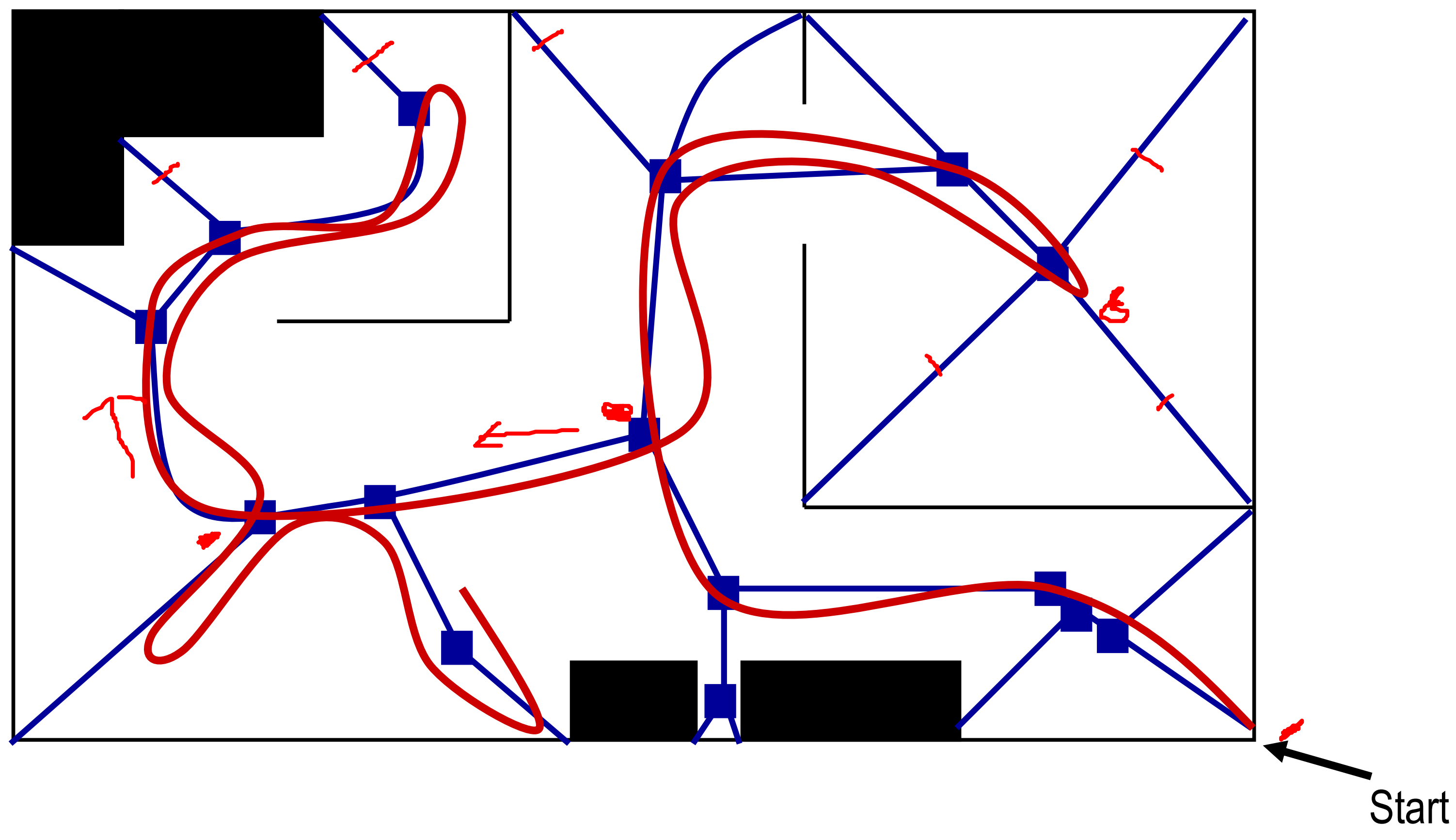
Motion Control Based on Frontier Exploration

- Robot calculates centroid
- Robot navigates using:
  - Move-to-goal and avoid-obstacle behaviors
  - Or, plan path and reactively execute the path
  - Or, continuously replan and execute the path
- Once robot reaches frontier, map is updated and new frontiers (perhaps) discovered
- Continue until no new frontiers remaining

Generalized Voronoi Graph Methods of Exploration

- Robot builds reduced generalized Voronoi graph (GVG) as it moves through world
- As robot moves, it attempts to maintain a path that is equidistant from all objects it senses (called “GVG edge”)
- When robot comes to gateway, randomly choose branch to follow, remembering the other edge(s)
- When one path completely explored, backtrack to previously detected branch point
- Exploration and backtracking continue until entire area covered.

Example of Voronoi Exploration



Summary

- SLAM works well for indoors, especially if range scans and multiple loops through an area
- How can you simultaneously map the world and be sure where you are?
  - You can use probabilistic methods to be reasonably certain where you are
- Frontier-based and GVG are two types of exploration, but not necessarily as intuitively efficient as might be expected