CSCI 1106
Lecture 20

Odometry

Announcements

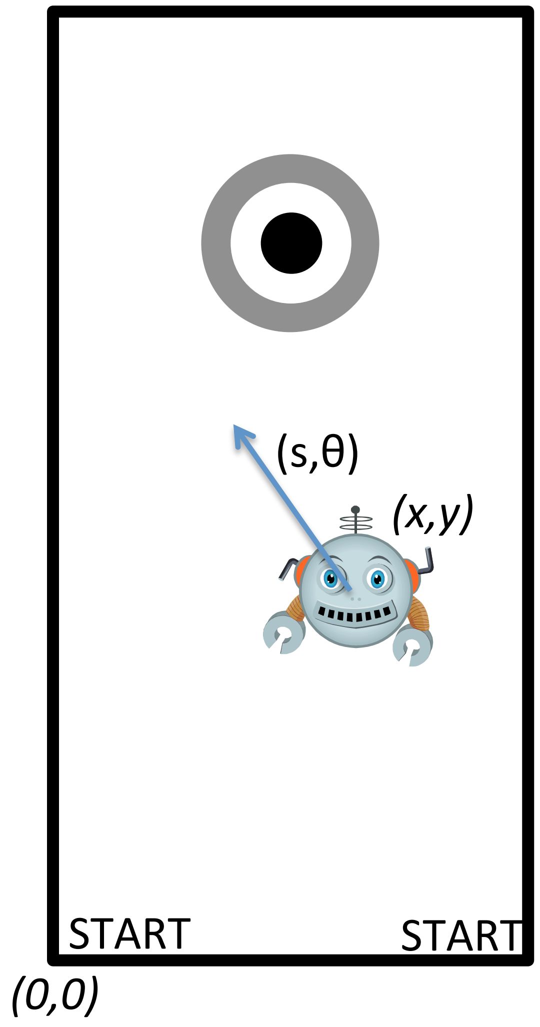
- Today's Topics
  - Motivation
  - Coordinates and Velocity
  - Linear Motion Odometry
  - Angular Motion Odometry
  - Errors in Odometry
  - Visual Odometry
  - Introduction to Search

Motivation: Where Am I?

- For many tasks a robot needs to know its
  - Position: physical location (x,y) in the environment
  - Orientation: direction it is facing
- Initially, robot starts out in a known position and orientation
  - e.g., at the start or a maze or left corner of arena
- As the robot moves it needs to update its known position and orientation
- *Odometry* is the use of movement sensors to estimate the robot*'s current position and orientation*

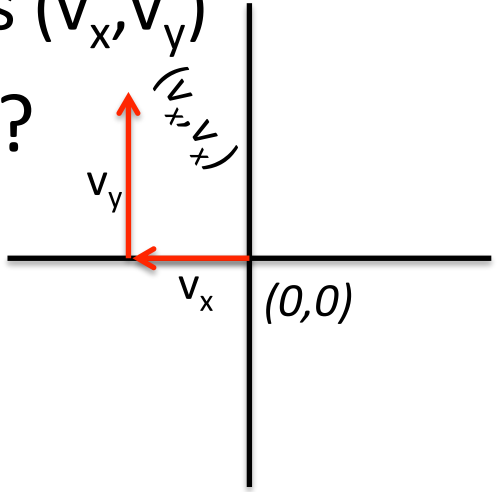
Location, Location, Location

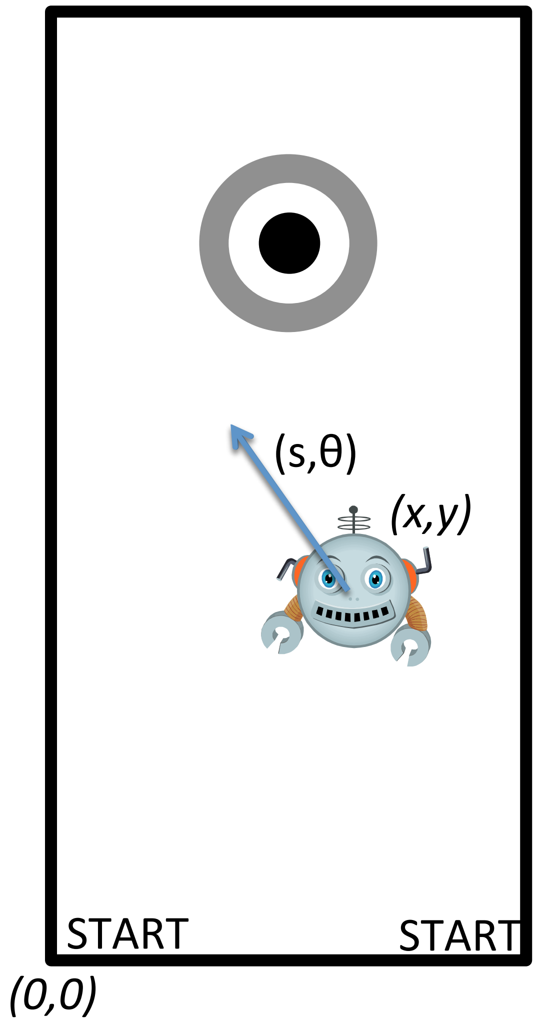
- Observation: You need to know where you are to know where you are going
- At any instant has robot has a
  - Location and orientation
    - Specified by coordinates (x,y) and direction ϕ
  - Velocity
    - Specified by speed *s* and direction θ
- Coordinates are relative to an origin (0,0)
  - Fixed location in the world or
  - Where the robot starts
- Typically assume that the robot
  - Knows where it starts or
  - Can determine its starting location
- Where have we seen this before?



Velocity

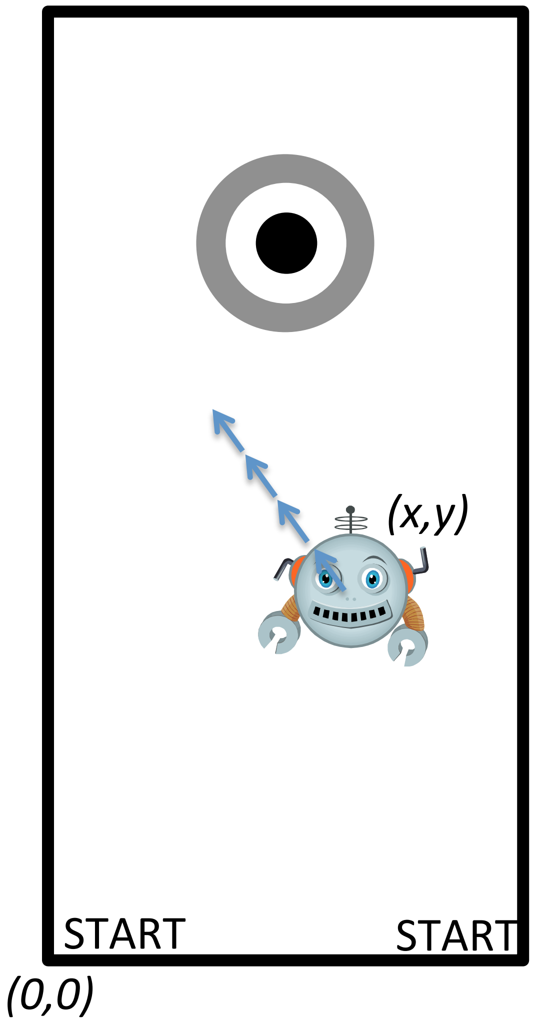
- Velocity can be represented in terms of
  - speed and direction (s,θ) or
  - horizontal and vertical speed components (v_x,v_y)
- What is (0,0)?





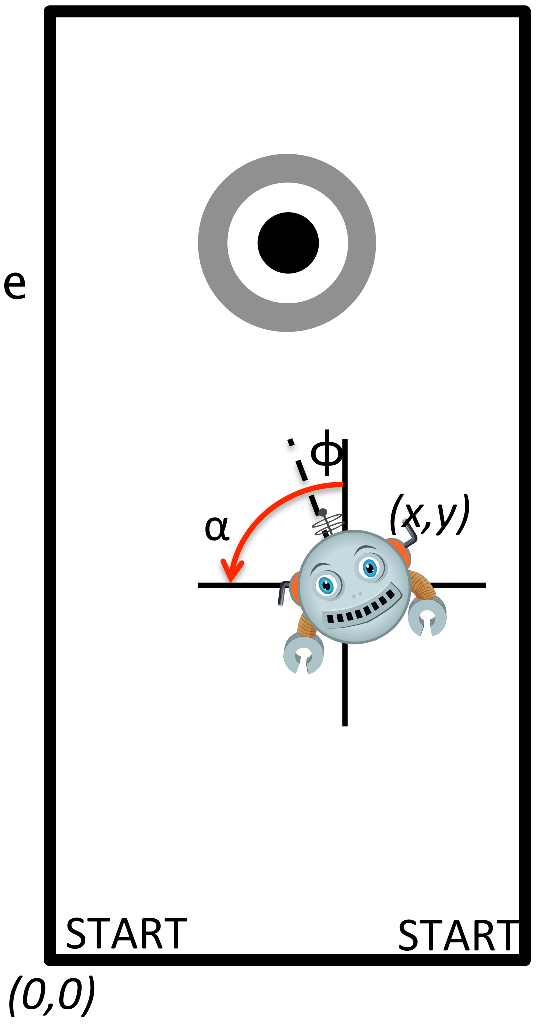
Linear Motion Odometry

- Obs: The velocity vector represents distance per unit time, e.g., (cm/s)
- Idea: Update position by adding velocity to position proportionally to elapsed times Δt
  - new position = old position + velocity
- Suppose velocity is represented by (s,θ)
  - $x' = x + s \times \sin(\theta) \times \Delta t$
  - $y' = y + s \times \cos(\theta) \times \Delta t$
- Suppose velocity is represented by (v_x,v_y)
  - $x' = x + v_x \times \Delta t$
  - $y' = y + v_y \times \Delta t$



Angular Motion Odometry

- Obs: Robots sometimes need to turn
- Assumption: Robot will turn on the spot
  - Orientation ϕ will change
  - Position (x,y) does not change
  - Angular velocity α (deg/s) is does not change
- Idea: Update orientation every second
  - new orient. = old orient. + angular velocity × time
  - $\phi' = \phi + (\alpha \times \Delta t)$
- How do we determine (v_x,v_y)?
- Observations: We know the velocity (s,θ)
  - Speed s is based on motor power
  - Direction θ is equal to the orientation ϕ
- Hence
  - $v_x = s \times \sin(\theta)$
  - $v_y = s \times \cos(\theta)$



Errors in Odometry

- We know
  - The initial position and orientation
  - The speed of the motors and the robot
- We always know where we are, right?
- Problem: Errors are introduced into the odometry computations
  - Speed is not constant
  - Motion is not straight

Things Go Wrong

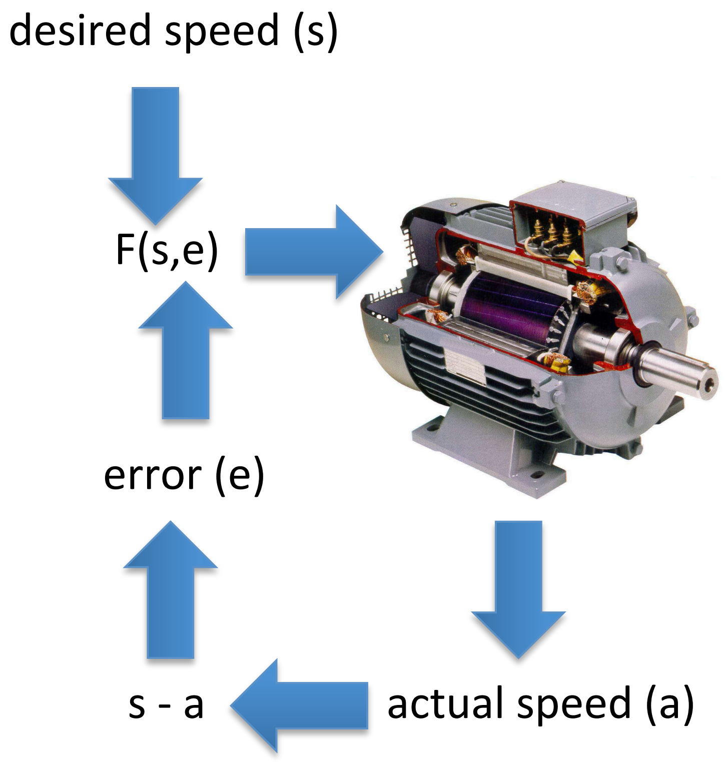
- What could go wrong?
  - Tires don't fully grip
  - Tires are not identical
  - Motors are slightly different
  - Battery is not fully charged
  - Speed sensors have variability
  - Motors engage at different times
  - Robot may bump into objects
- Can we compensate?
- Use additional sensors to correct for errors

Sources of Data for Odometry

- Motor sensors
  - rotation sensors (how fast the motor is turning)
- Motion sensors
- Accelerometers and Gyroscopes
- Compass
  - Very useful for orientation
- Cameras
- Rangefinders (infrared, ultrasonic, or laser)

Rotation Sensors and the Control Loop

- Idea: Many motors have built in rotation (speed) sensors
  - Motor's *actual speed* can deviate from *desired speed*
  - *Actual speed* can be adjusted to match *desired speed*
  - A rotation sensor measures the motor's *actual speed* to adjust motor's speed as needed
- Idea: We use rotation sensors implicitly
  - Robot's motors have a built in control loop
  - We set the desired speed of the motors
  - Assume that the motors run at the desired speed
- What about using other sensors?



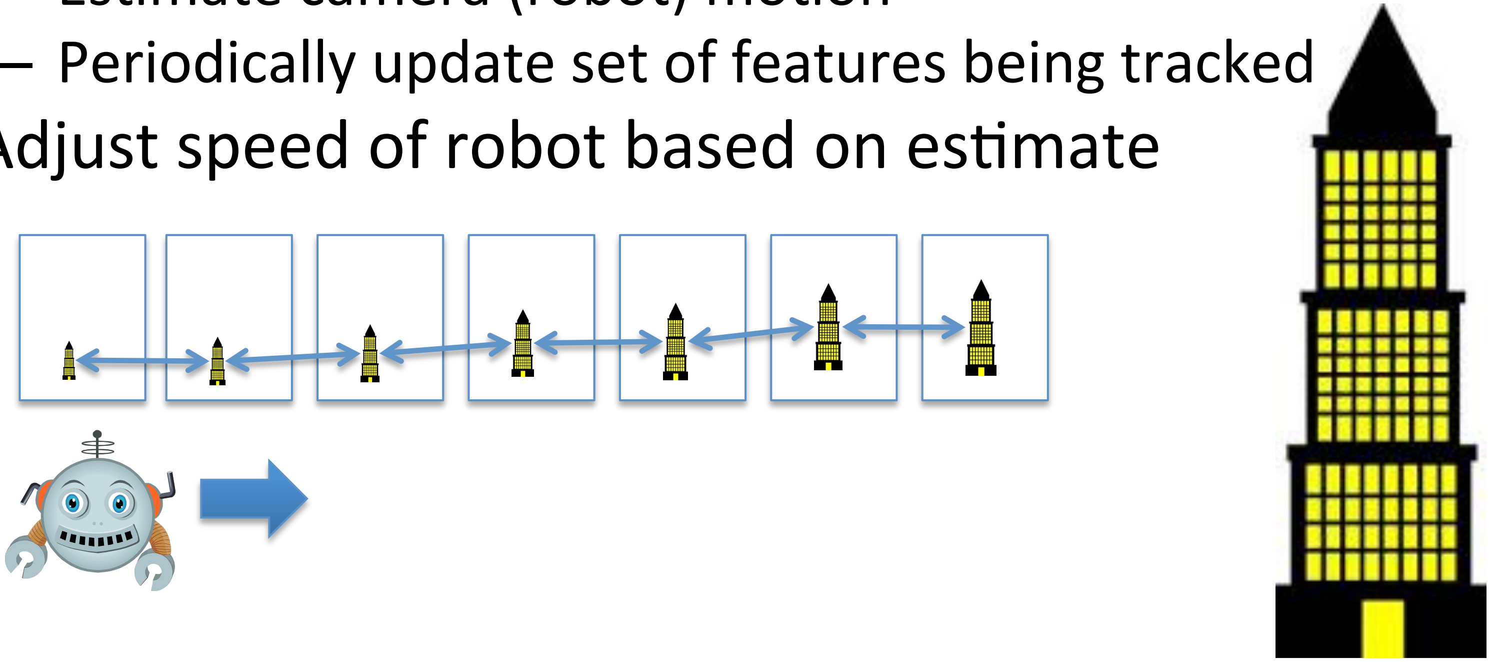
Control Loop

Visual Odometry

- Idea: Use landmarks to gauge position and speed
- Approach 1: Optical Flow based
  - Compute velocity using consecutive camera images
- Approach 2: Landmark (map) based
  - Compute location by matching known landmarks in camera images

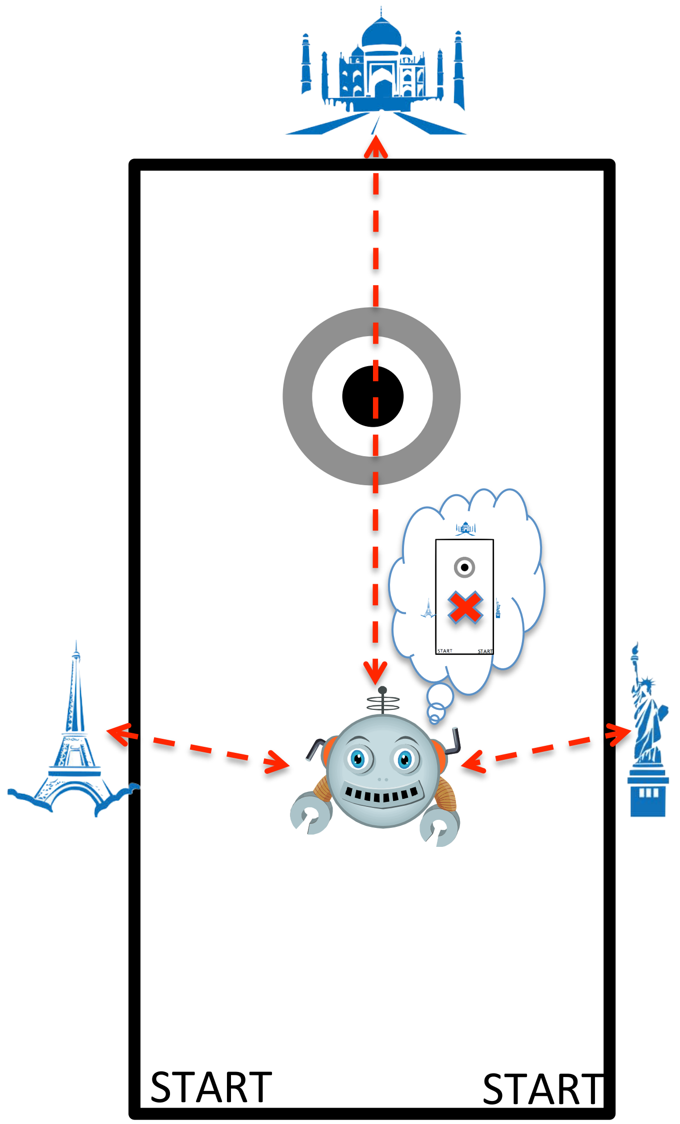
Optical Flow based Odometry

- Idea: Gauge the robot's velocity by comparing objects (features) in consecutive camera images
  - Extract features from image
  - Match from image to image (construct optial flow)
  - Estimate camera (robot) motion
  - Periodically update set of features being tracked
- Adjust speed of robot based on estimate



Landmark based Odometry

- Idea: Triangulate robot's location using known landmarks
  - Create a map of known landmarks
  - Periodically
    - Take images of surround environment
    - Extract known landmarks from images
    - Estimate distance to landmarks
    - Triangulate position
- Use location estimate to refine future velocity estimates



Problems with Vision based Odometry

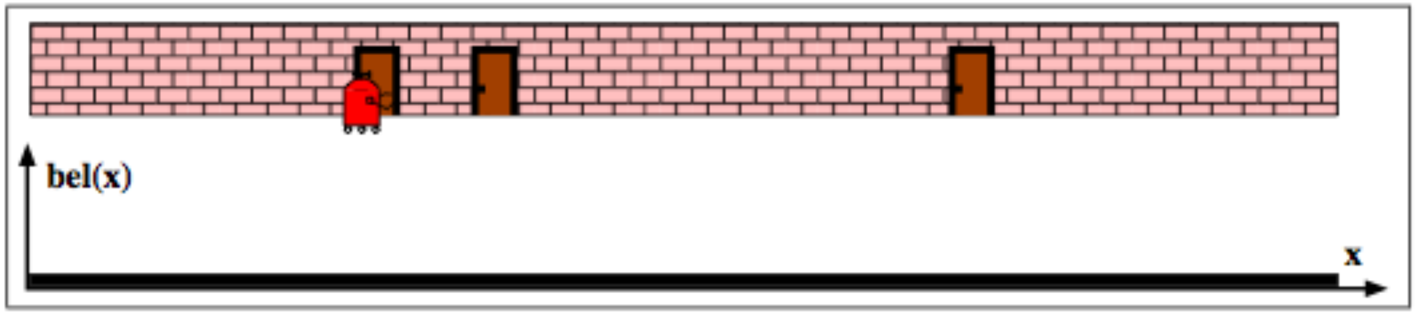
- Images are affected by environment conditions
  - light, fog, rain, dust, etc
- Objects can become occluded
- Feature extraction is expensive and imperfect
- Distance estimation is error-prone
- Landmarks can change

- Entire process is highly variable
- Other technologies are use specific but more accurate
  - range finders, GPS, etc

- Why do we care?

Markov Localization

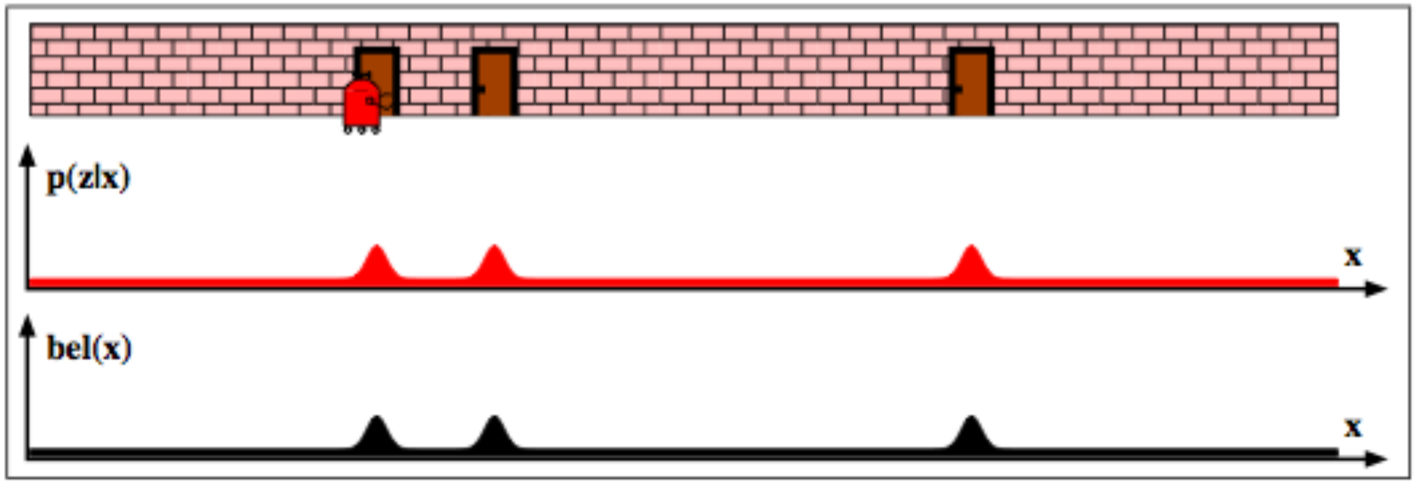
(a)



The robot doesn’t know where it is. Thus, a reasonable initial believe of it’s position is a uniform distribution.

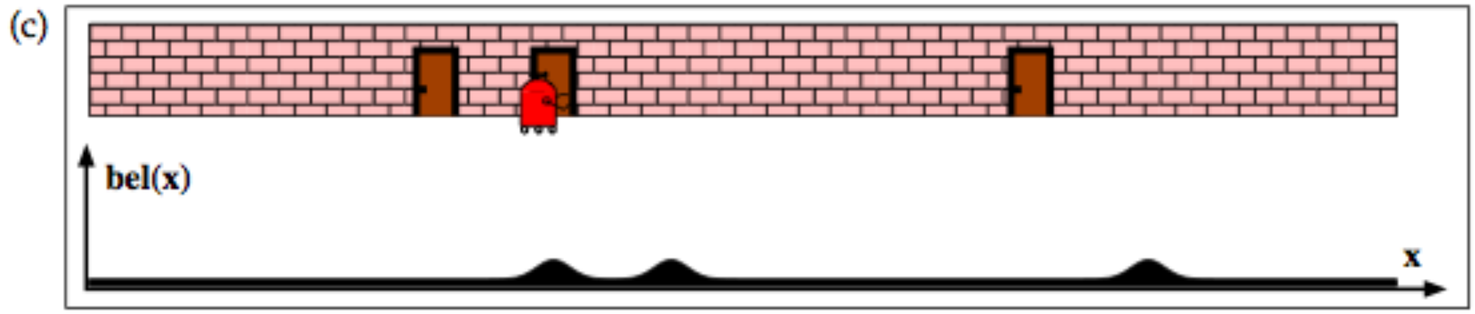
Markov Localization

(b)



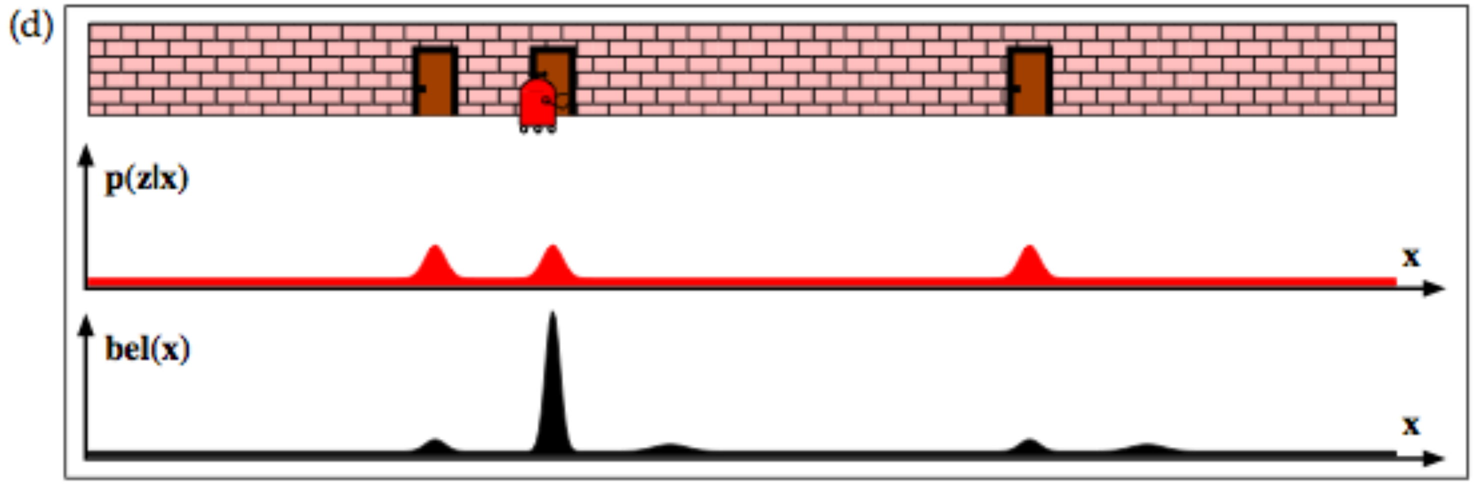
A sensor reading is made (USE SENSOR MODEL) indicating a door at certain locations (USE MAP). This sensor reading should be integrated with prior believe to update our believe (USE BAYES).

Markov Localization



The robot is moving (USE MOTION MODEL) which adds noise.

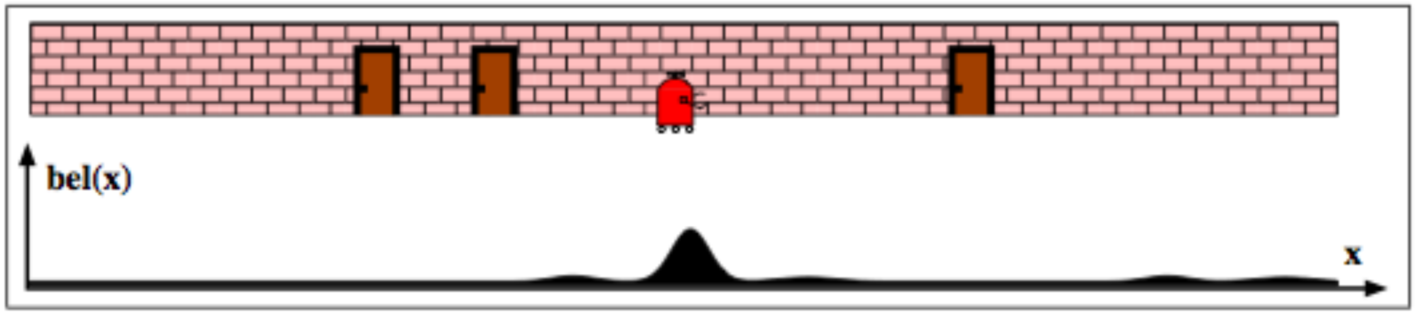
Markov Localization



A new sensor reading (USE SENSOR MODEL) indicates a door at certain locations (USE MAP). This sensor reading should be integrated with prior believe to update our believe (USE BAYES).

Markov Localization

(e)



The robot is moving (USE MOTION MODEL) which adds noise. …

Modern Solutions SLAM

- Particle filters https://www.youtube.com/watch?v=H0G1yslM5rc
- SLAM https://www.youtube.com/watch?v=bq5HZzGF3vQ