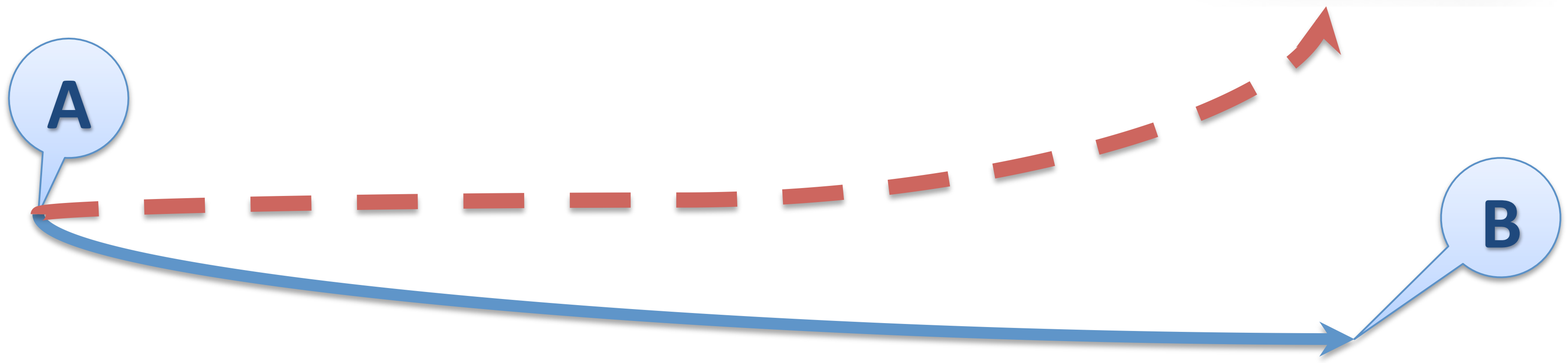
CSCI 1108

Actuators
Kinematics
Odometry
Localization



leds.prox.h(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}

leds.buttons(led0, led1, led2, led3) {0...32}

leds.circle(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}

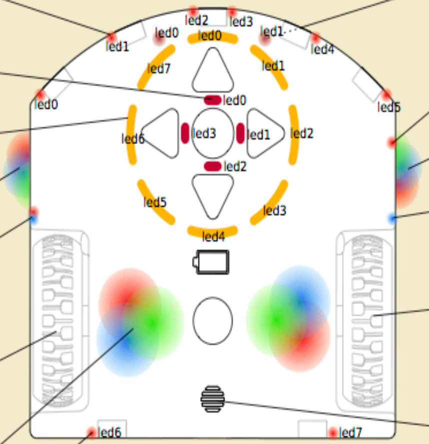
leds.bottom.left(red, green, blue) {0...32}

leds.temperature(red, blue) {0...32}

motor.left.target desired speed {-500...500}, 500 = ~20 cm/s
motor.left.speed actual speed
motor.left.pwm motor command
motor 100 Hz

leds.top(red, green, blue) {0...32}

leds.prox.h(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}



leds.prox.v(led0, led1) {0...32}

leds.rc(led) {0...32}

leds.bottom.right(red, green, blue) {0...32}

leds.sound(led) {0...32}

motor.right.target desired speed {-500...500}, 500 = ~20 cm/s
motor.right.speed actual speed
motor.right.pwm motor command
motor 100 Hz

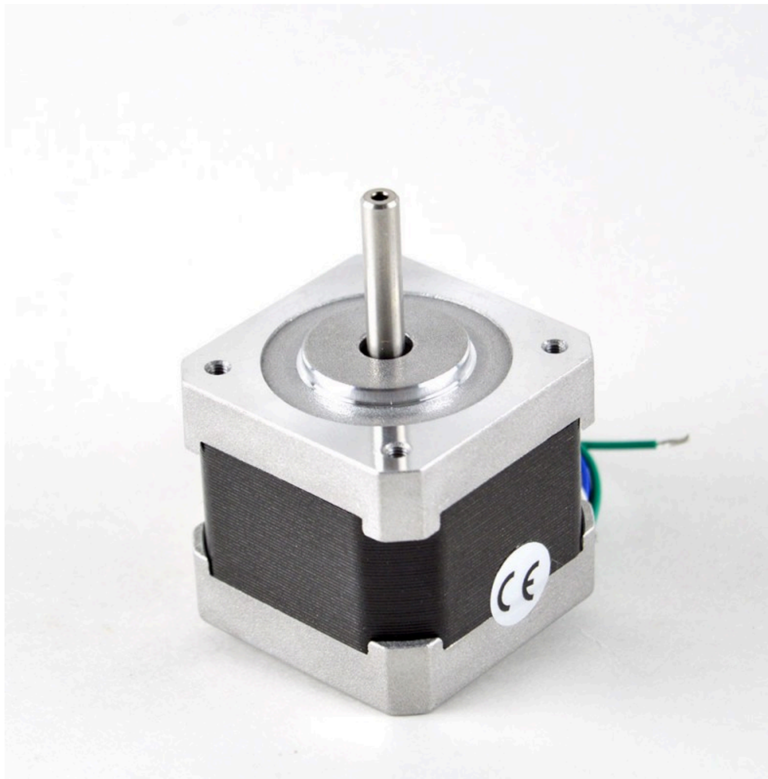
sound.finished a sound finished playing
sound.system(N) N: {0...7}, play system sound N. N=-1, stop playing
sound.freq(Hz,ds) [Hz],[1/60 s]
sound.wave(wave[142]) change primary wave, wave[i] : {-128...127}
sound.play(N) N: {0...32767}, play 'pN.wav'. N=-1, stop playing
sound.replay(N) N: {0...32767}, replay 'rN.wav'. N=-1, stop playing

Actuators

DC motor

Servo Motor

Stepper motor



Our experimental approach (Tutorial 4)

Model the effect of running the motors with different power

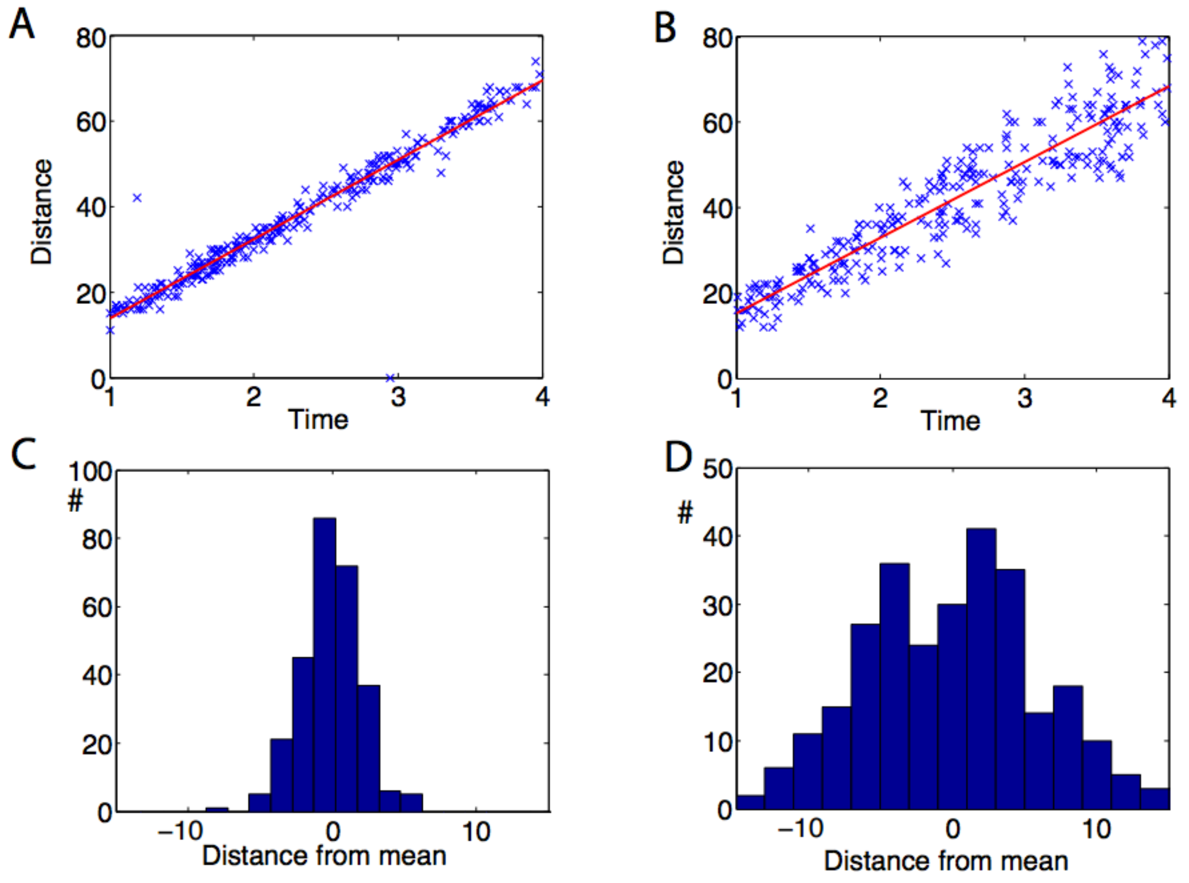
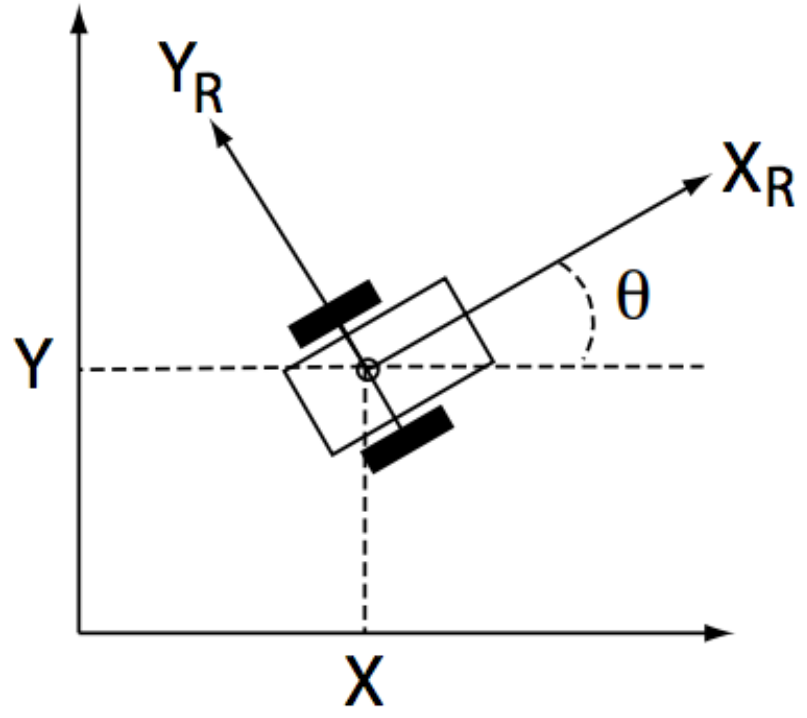


Fig. 4.1 (A) Measurements of distance travelled by the tribot when running the motor for different number of milliseconds with constant motor power. (B) Same as (A) with random motor power. (C,D) Corresponding histogram of differences between data and hypothesis.

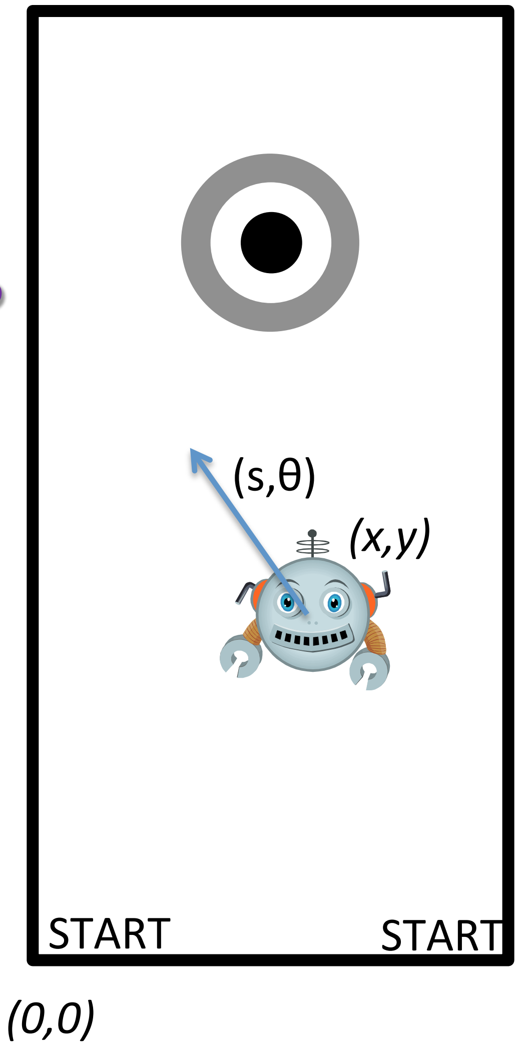
Kinematics is the branch of classical mechanics which describes the motion of points … without consideration of the masses of those objects nor the forces that may have caused the motion. (Wikipedia)



Pose of the robot

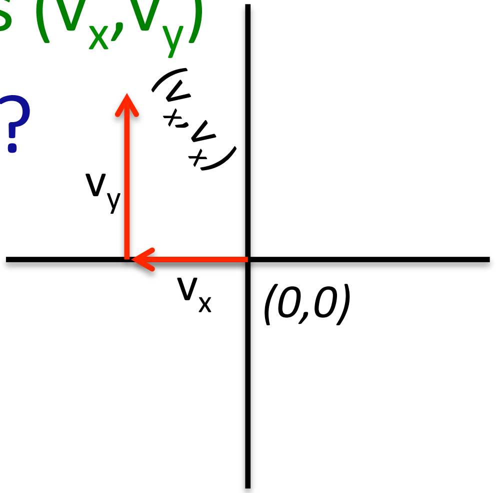
Localization

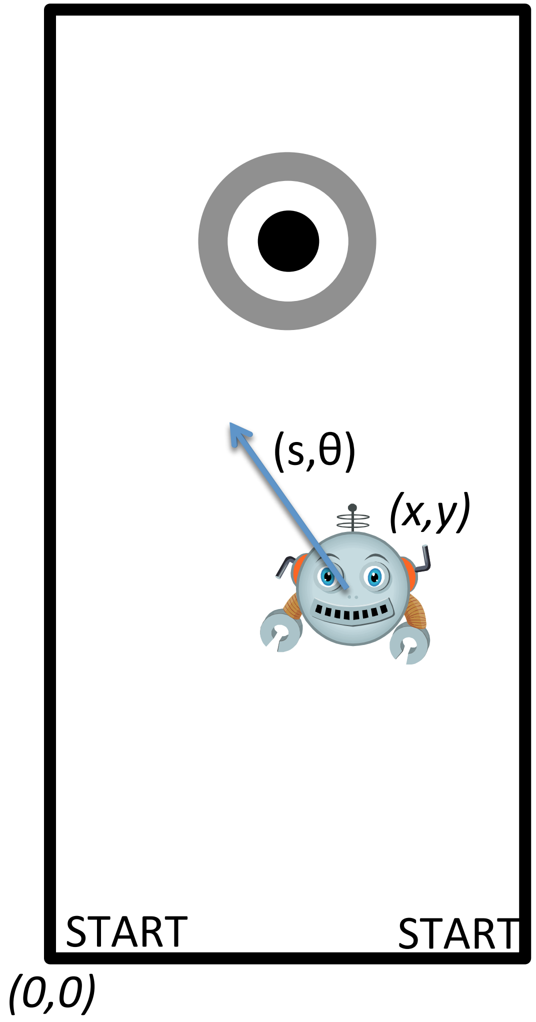
- **Observation:** You need to know where you are to know where you are going
- At any instant robot has a
  - Location and orientation
    - Specified by coordinates **(x,y)** and direction **ϕ**
  - Velocity
    - Specified by speed ***s*** and direction **θ**
- Coordinates are relative to an origin (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

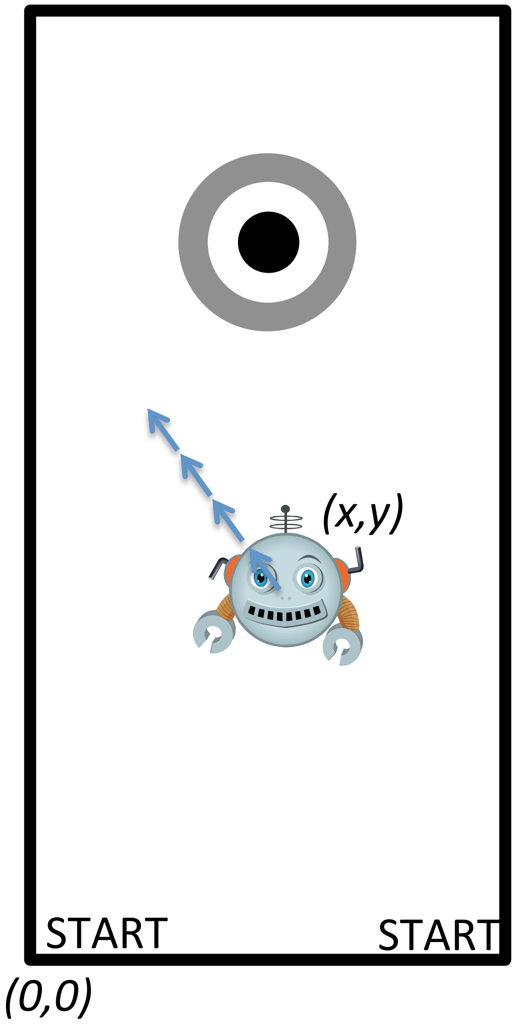
- Velocity can be represented in terms of
  - speed and direction (s,θ) or
  - horizontal and vertical speed components (v_x,v_y)
- What is (0,0)?





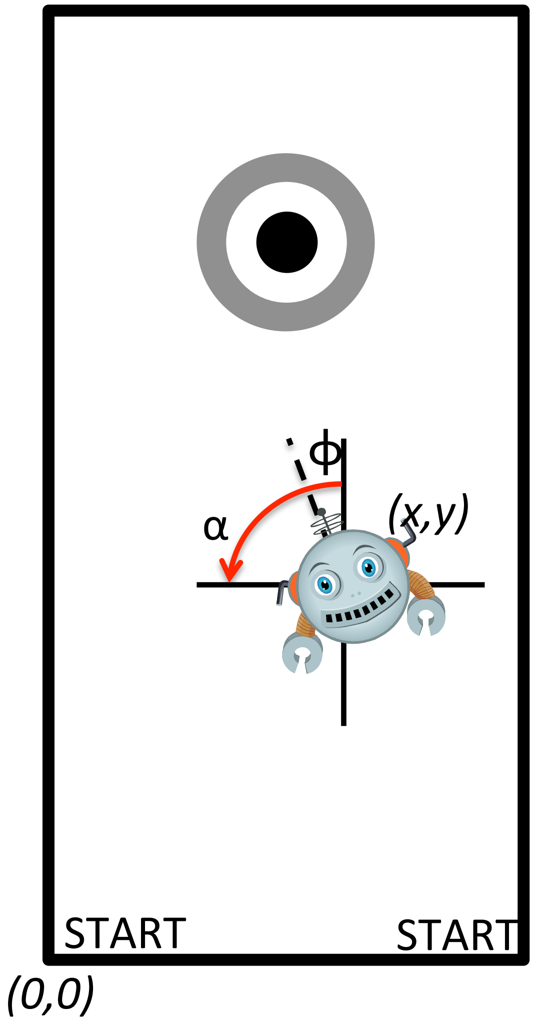
Differential (Linear) Motion

- **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 × time
- 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

- **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) 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 model-based localization

- 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 location 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

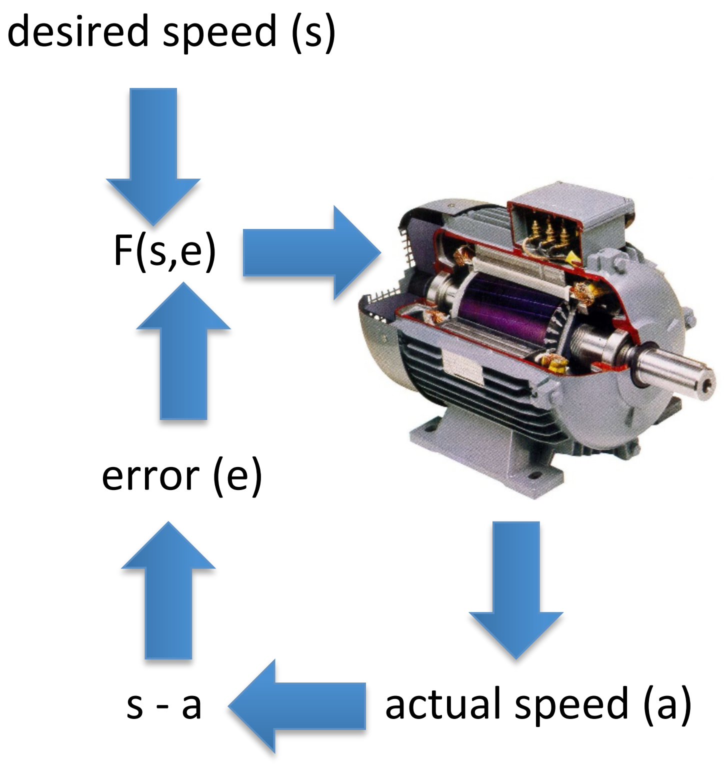
Odometry is the use of data from motion sensors to estimate change in position over time (Wikipedia)

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?



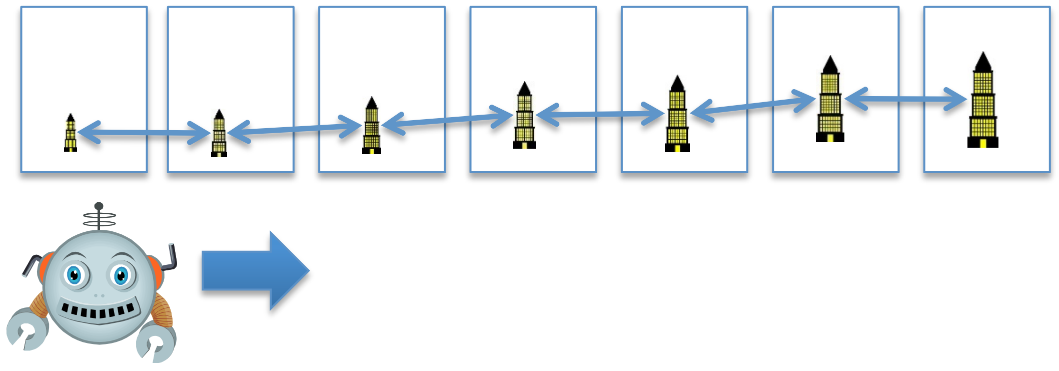
Control Loop

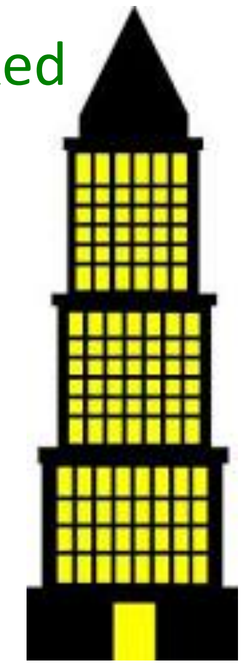
Visual Odometry

- **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 optical flow)
  - Estimate camera (robot) motion
  - Periodically update set of features being tracked
- Adjust speed of robot based on estimate

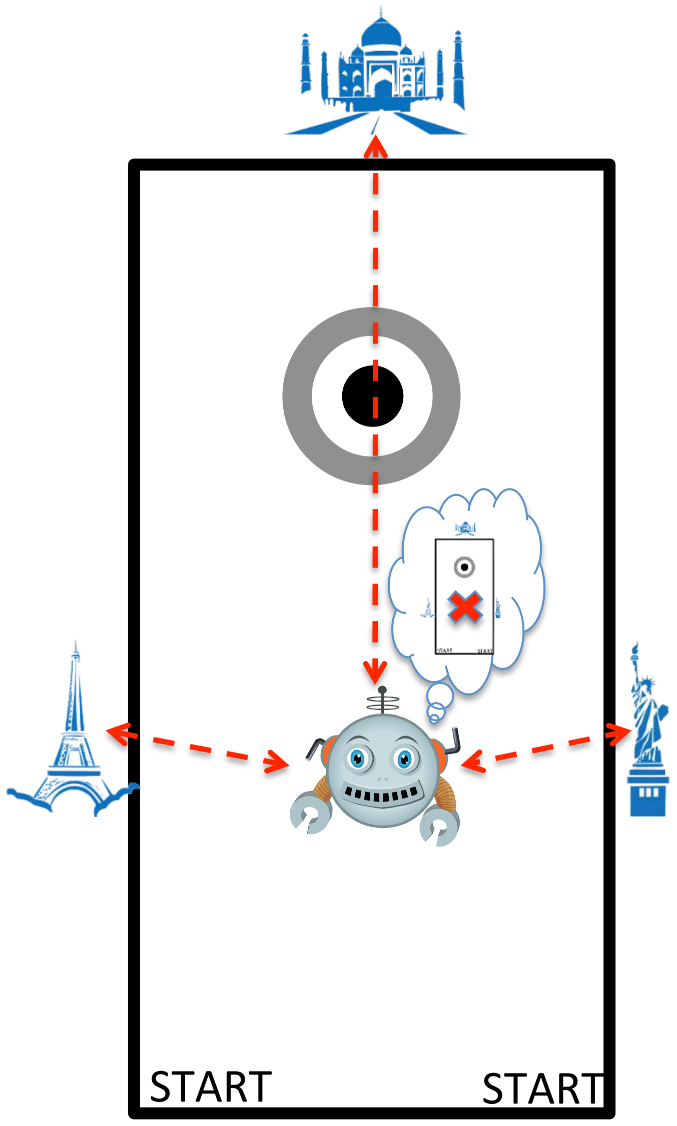




Video by James Bowman and Kurt Konolige's work.

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

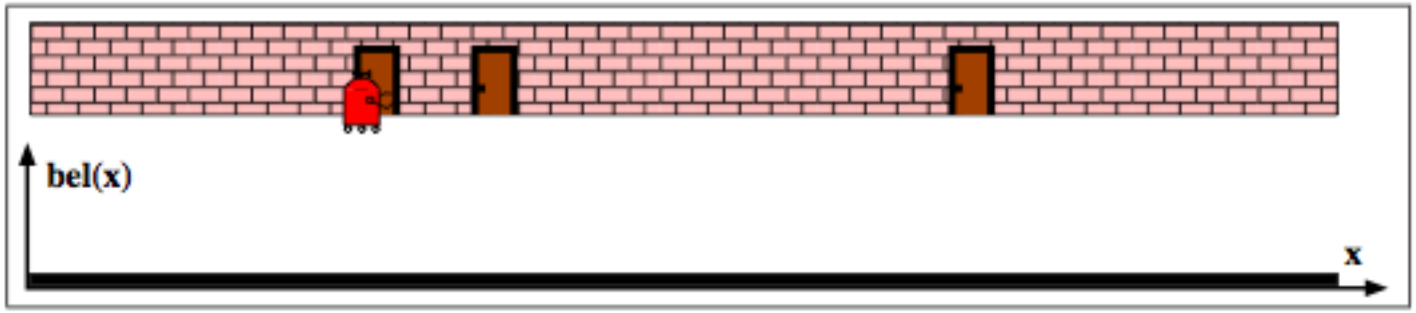
- Images are affected by environment conditions
  - light, fog, rain, dust, etc
- Objects can become occluded
- Feature extraction is expensive and imperfect
- Distance estimation is error-prone
- Landmarks can change
- Entire process is highly variable
- Other technologies are use specific but more accurate
  - range finders, GPS, etc
- Why do we care?
  - One of the most common tasks in robotics is to map (explore) a given environment
    - Robot must know where it is and where it was
    - This includes searching (avoid searching same place twice)

Outlook

Markov Localization
Particle Filters
SLAM

Markov Localization

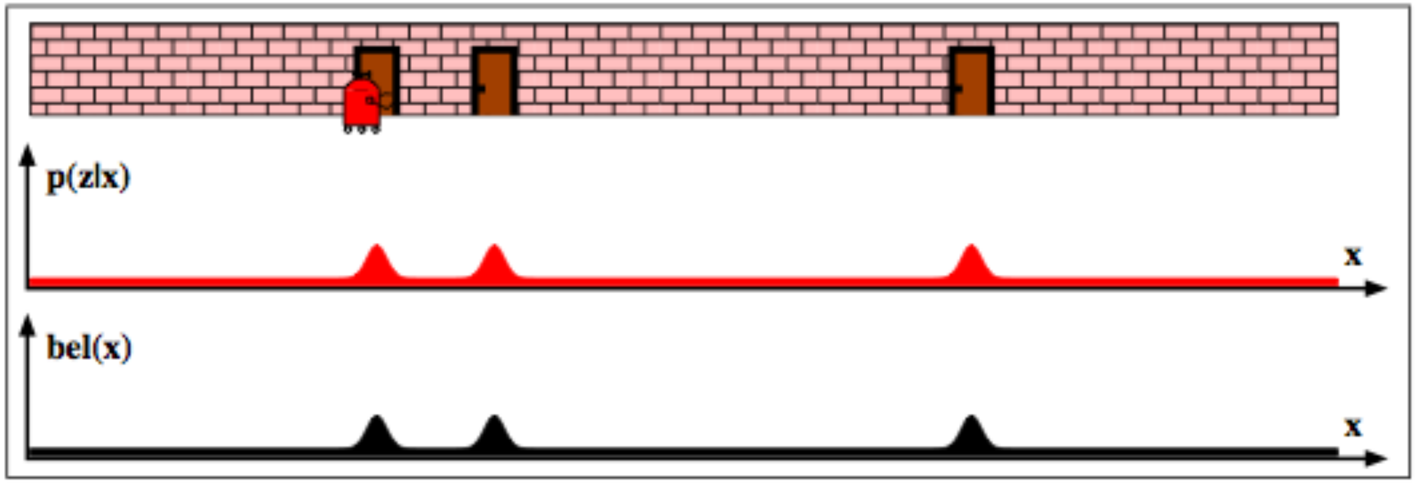
(a)



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

Markov Localization

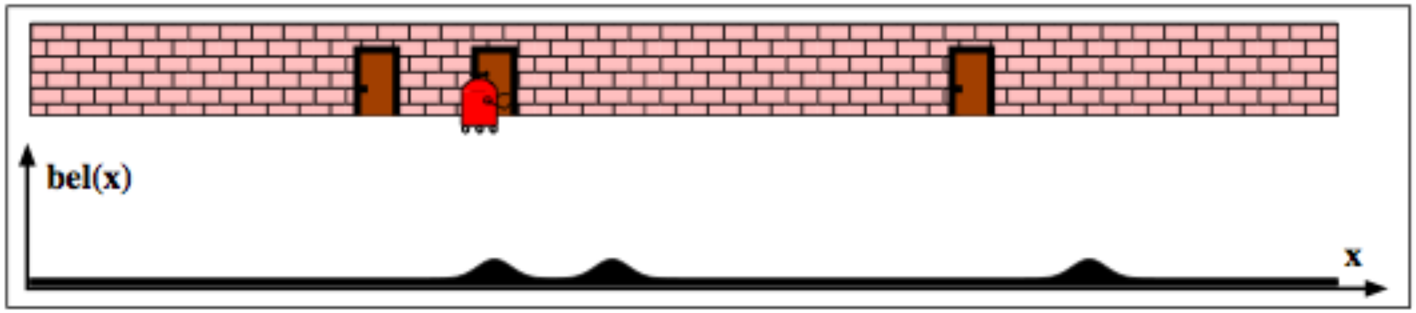
(b)



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).

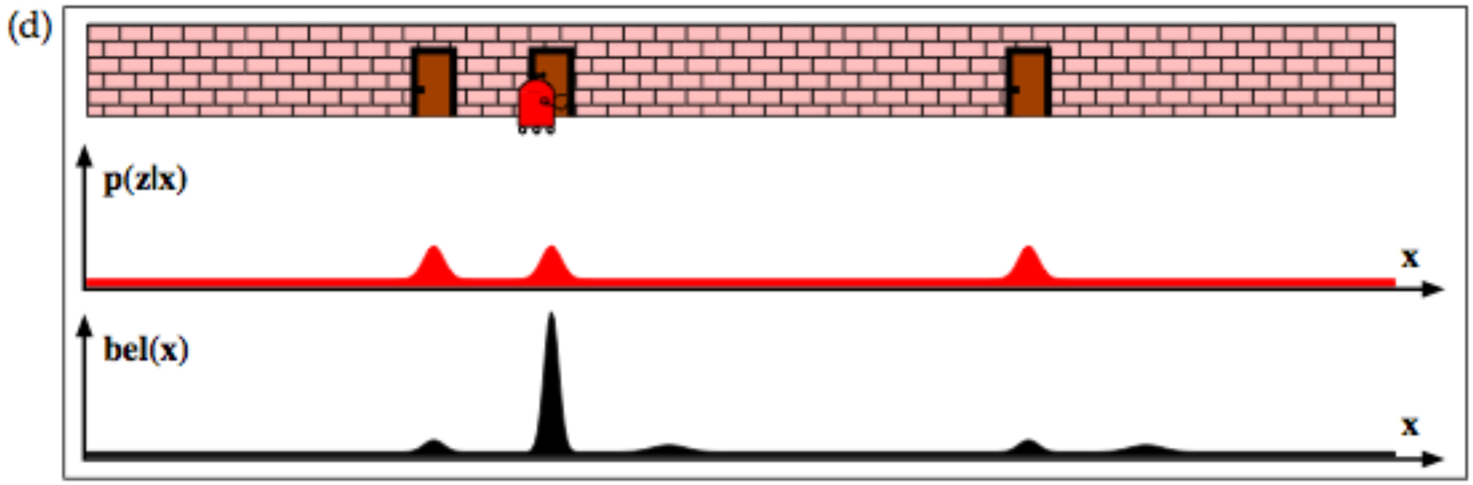
Markov Localization

(c)



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

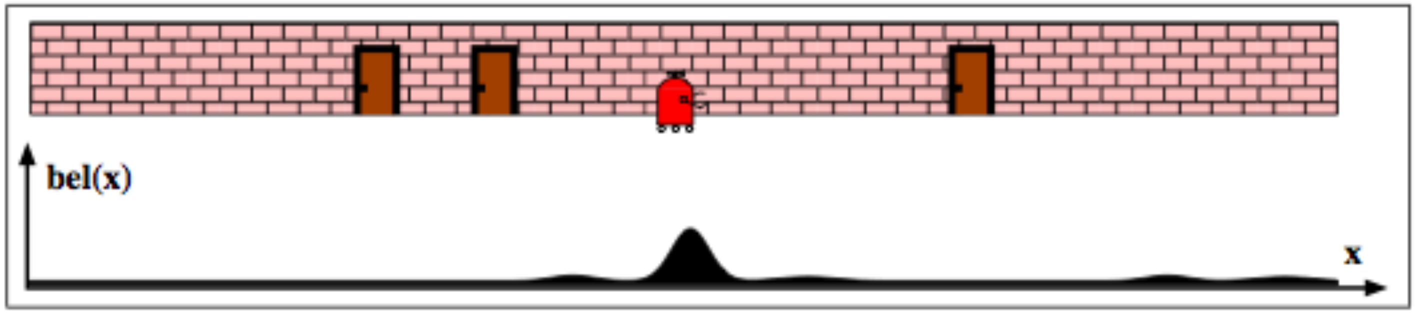
Markov Localization



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 (Simultaneous Localization and Mapping)

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