Dead Reckoning

- Orientation uncertainty is the largest contributor to Odometry errors.
- Dead Reckoning is the use of an independent heading sensor to find orientation of the robot. Odometry is still used to find distance traveled.
- Problem: accuracy and noise of heading sensor
- Many autonomous vehicles use expensive ($3k – $65k) laser ring gyros with very little drift.

Gyroscope

- MEMS Gyros will typically drift due to temperature dependent bias.
- Bias must be regularly calibrated out
- `rc_calibrate_gyro()` will measure bias and subtract it…
- Bumps and impulses can create false gyro readings
- Gyro data can be fused with magnetometer, but useless indoors

“Gyrodometry”

- Idea: Fuse Odometry with Gyro based heading estimate
- If wheels aren’t moving, we can ignore the gyro
- If gyro reads significantly different than odometry probably hit a bump and should trust the gyro

Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Apr. 22-28, 1996, pp. 423-428.

Gyrodometry: A New Method for Combining Data from Gyros and Odometry in Mobile Robots

by

J. Borenstein and L. Feng

The University of Michigan

“Gyrodometry”

- Experimentally determine a threshold for the algorithm

- Have a better idea? Try it and compare.

$$\Delta_{G\text{-}O} = \Delta\theta_{gyro} - \Delta\theta_{odo}$$

$$\text{if } (|\Delta_{G\text{-}O,i}| > \Delta\theta_{thres})$$

$$\text{then} \quad \theta_i = \theta_{i-1} + \Delta\theta_{gyro,i}\, T$$

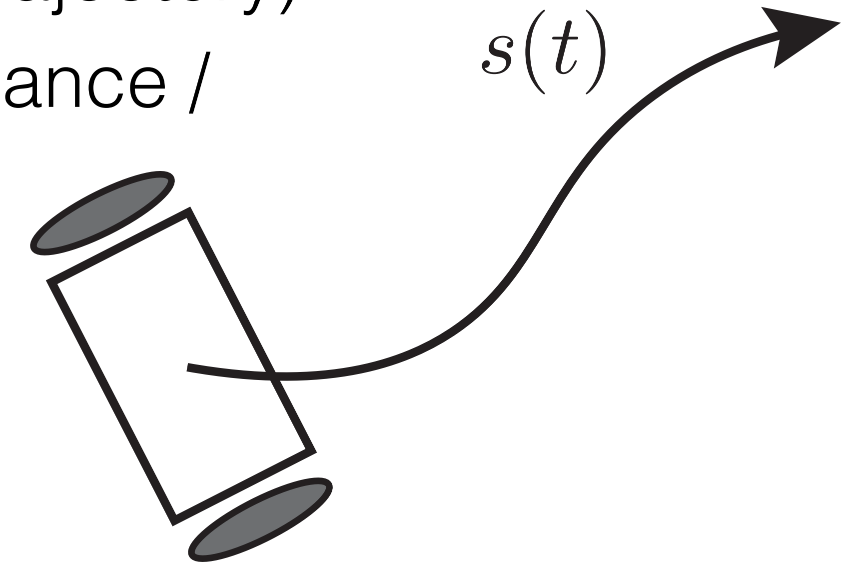
$$\text{else} \quad \theta_i = \theta_{i-1} + \Delta\theta_{odo,i}\, T$$

Trajectory Following

Lecture 16

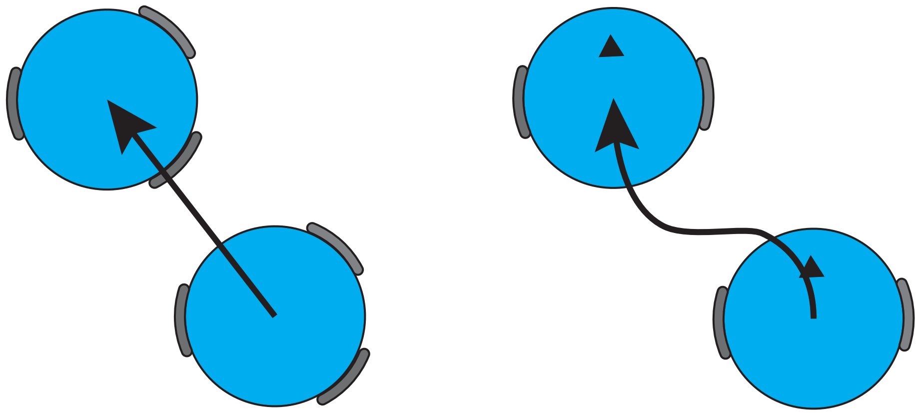
Trajectory Following

- Want to follow a path $s(t)$
- Problem $s(t) \ \rightarrow \ v(t), \omega(t)$
- Path could be predefined (i.e. square trajectory)
- Path could be dynamic (obstacle avoidance / dynamic obstacles)
- Potential paths defined by system
  - Maximum Acceleration
  - Maximum Velocity
  - Turning Velocity/Aceeleration



Holonomic vs. Non Holonomic

- Holonomic
  - controllable degrees of freedom = total degrees of freedom
- Non-Holonomic
  - controllable degrees of freedom < total degrees of freedom



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \qquad \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}$$

Robot Frame Velocity

$$v = \frac{v_R + v_L}{2}$$

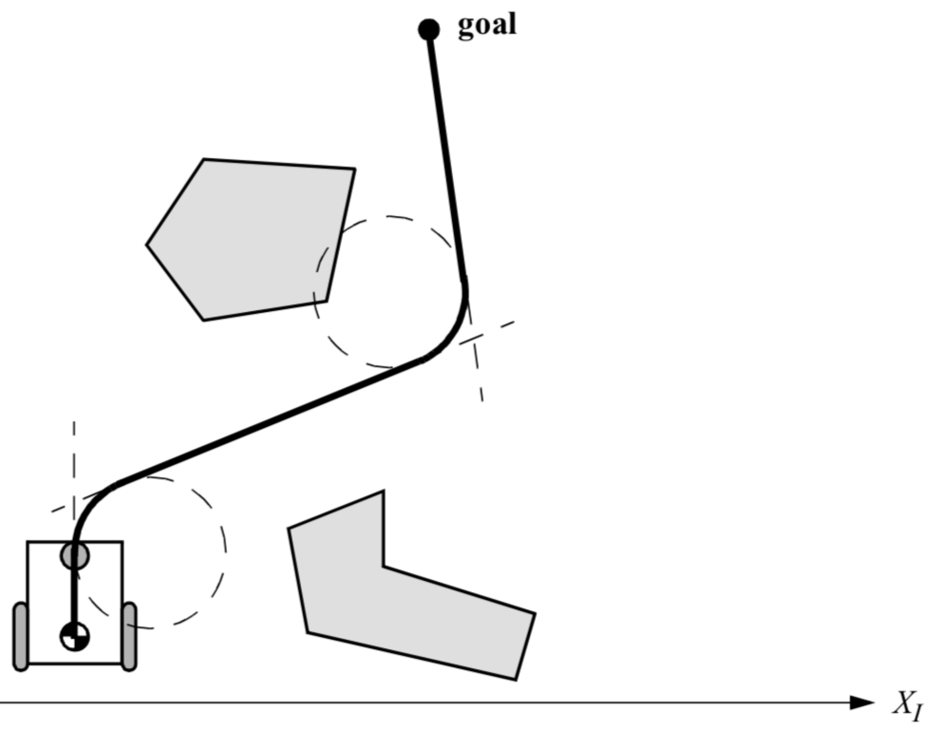
$$\omega = \frac{v_R - v_L}{b}$$

$$v_R = v + \frac{b}{2}\omega$$

$$v_L = v - \frac{b}{2}\omega$$

Trajectory Following (Open Loop)

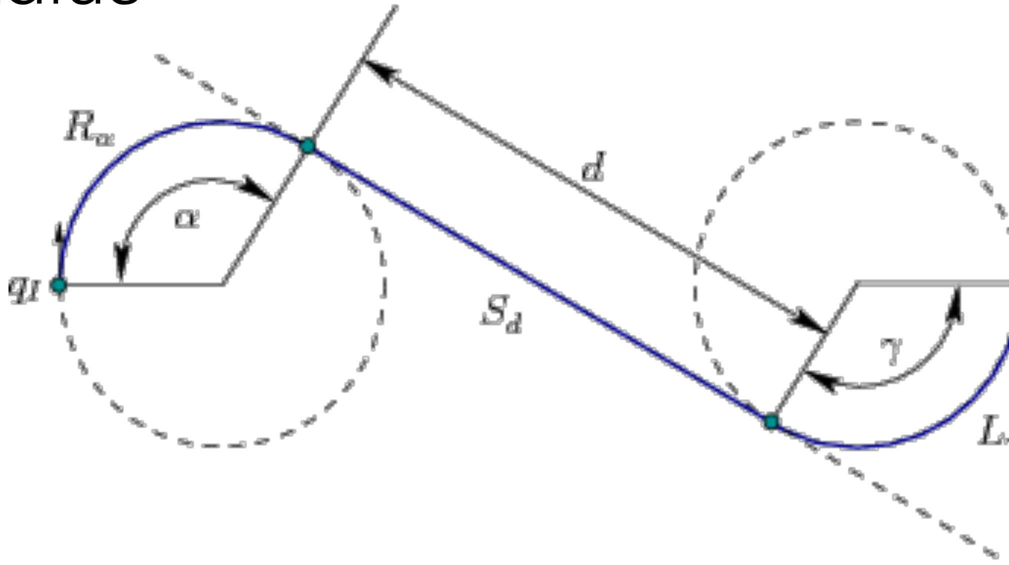
- Precompute trajectory based on motion segments of clearly defined shape
- lines & circle segments
- Send commands that would produce the motion
- Resulting trajectories are usually not particularly accurate



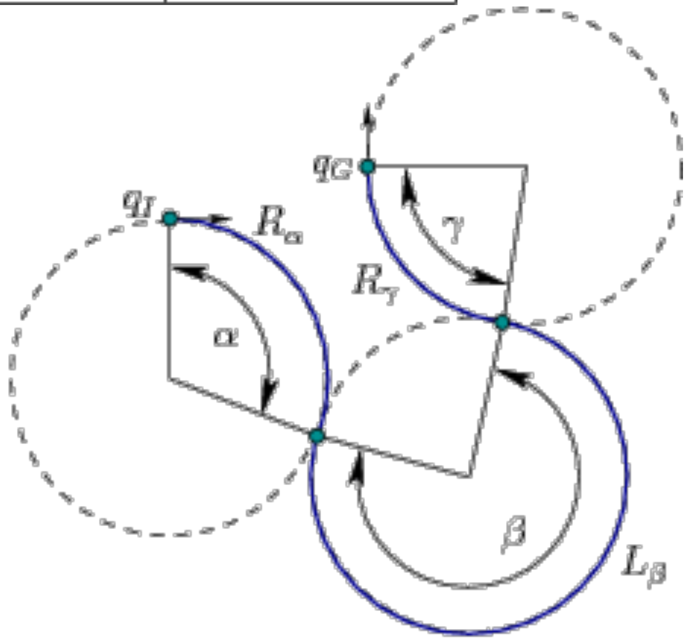
Dubins Path

- Constant velocity paths
- Can always find optimal path for velocity & turning radius

Symbol	Steering: u
S	0
L	1
R	-1



$R_\alpha S_d L_\gamma$

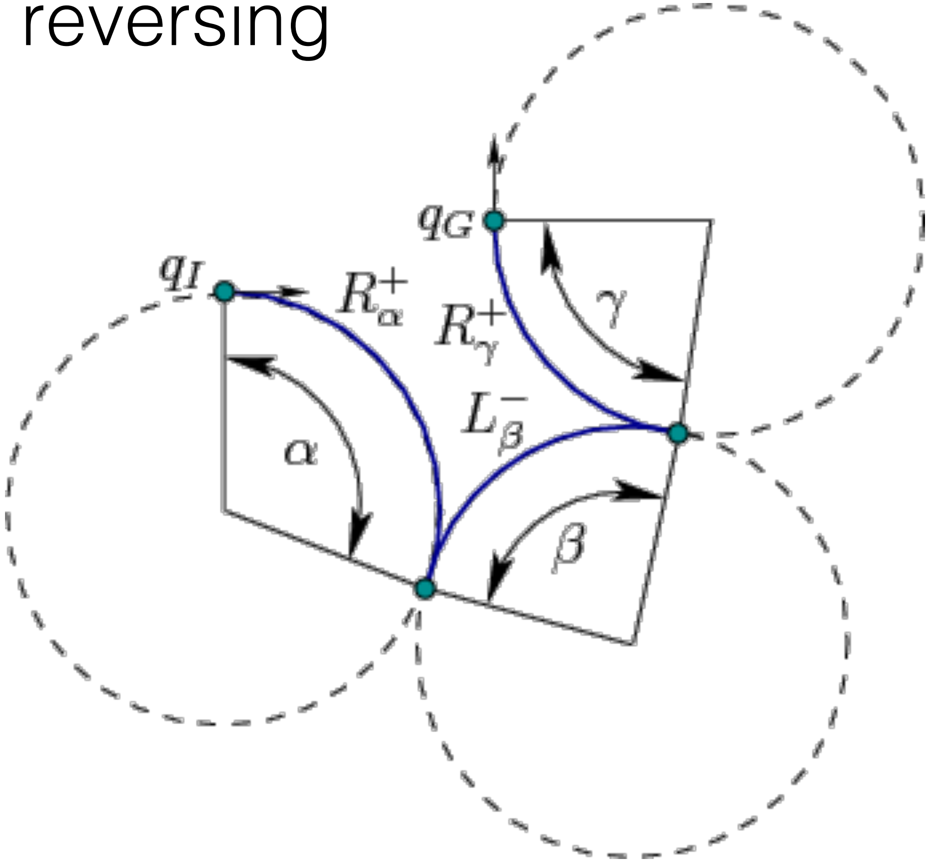


$R_\alpha L_\beta R_\gamma$

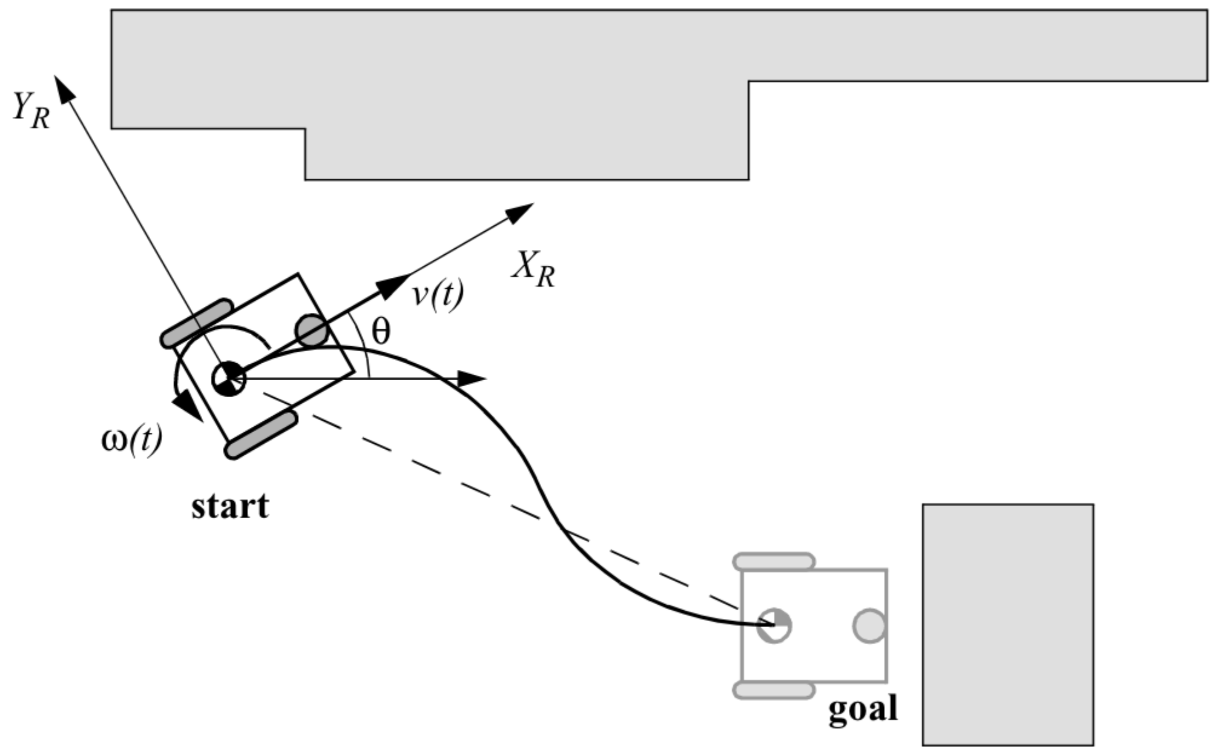
Reeds-Shepp Curves

- Extends Dubins paths to include reversing

Symbol	Gear: u_1	Steering: u_2
S^+	1	0
S^-	-1	0
L^+	1	1
L^-	-1	1
R^+	1	-1
R^-	-1	-1

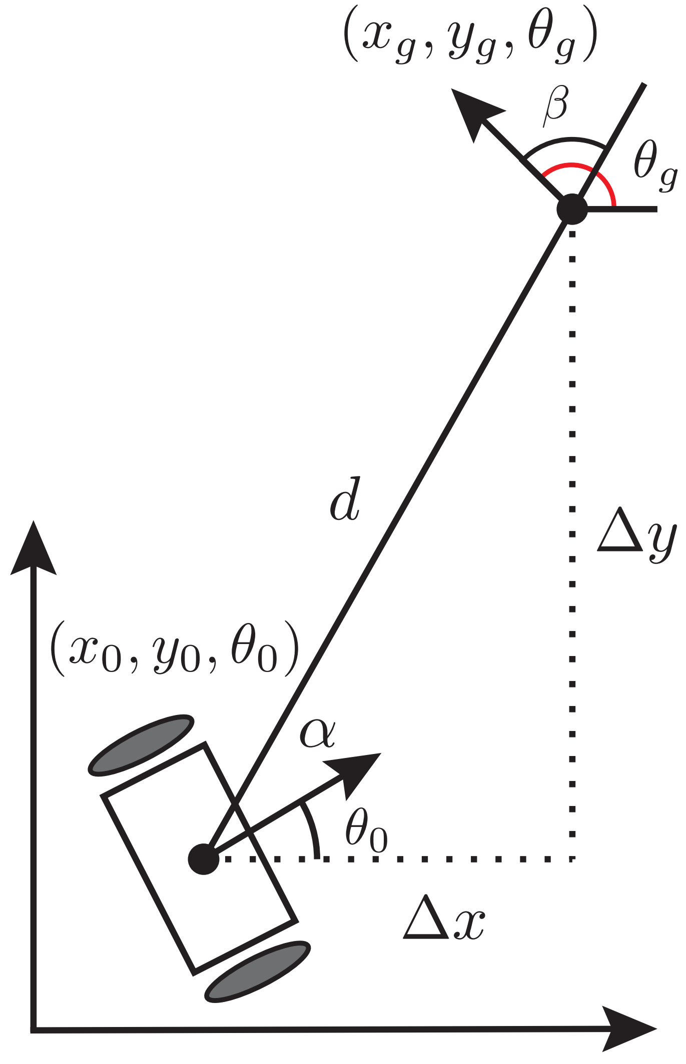


Feedback Control



- Drive the robot from current position to goal position by minimizing error to goal
- Optionally ignore end orientation
- Simple way: Rotate, Translate, Rotate (RTR)
- More advanced method: Use state feedback.

Conversion to Polar Coordinates



- Offsets from goal position

$$\Delta x = x_g - x$$

$$\Delta y = y_g - y$$

- Distance to goal position

$$d = \sqrt{\Delta x^2 + \Delta y^2}$$

- Angle between current heading and heading towards goal

$$\alpha = \mathrm{atan2}(\Delta y, \Delta x) - \theta$$

- Angle between current heading and goal heading

$$\beta = \theta_g - \theta$$

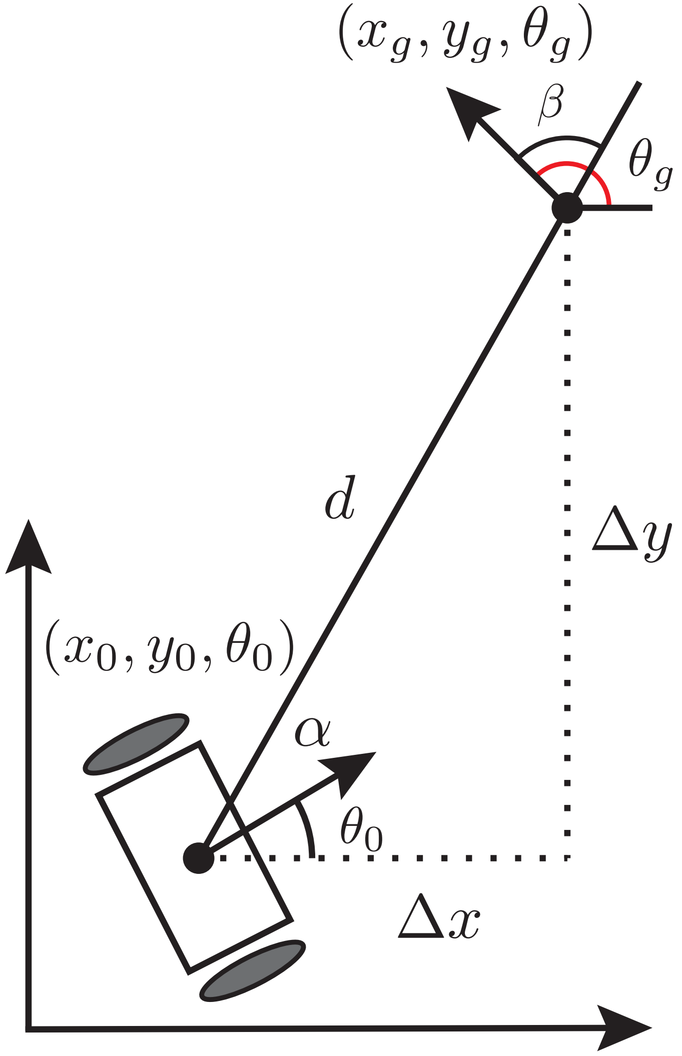
Control Law

- We want a control law to drive

$$d \to 0$$

$$\alpha \to 0$$

$$\beta \to 0$$



Feedback Control to a Point

- ignore final orientation
- In world frame we want to move:

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \omega$$

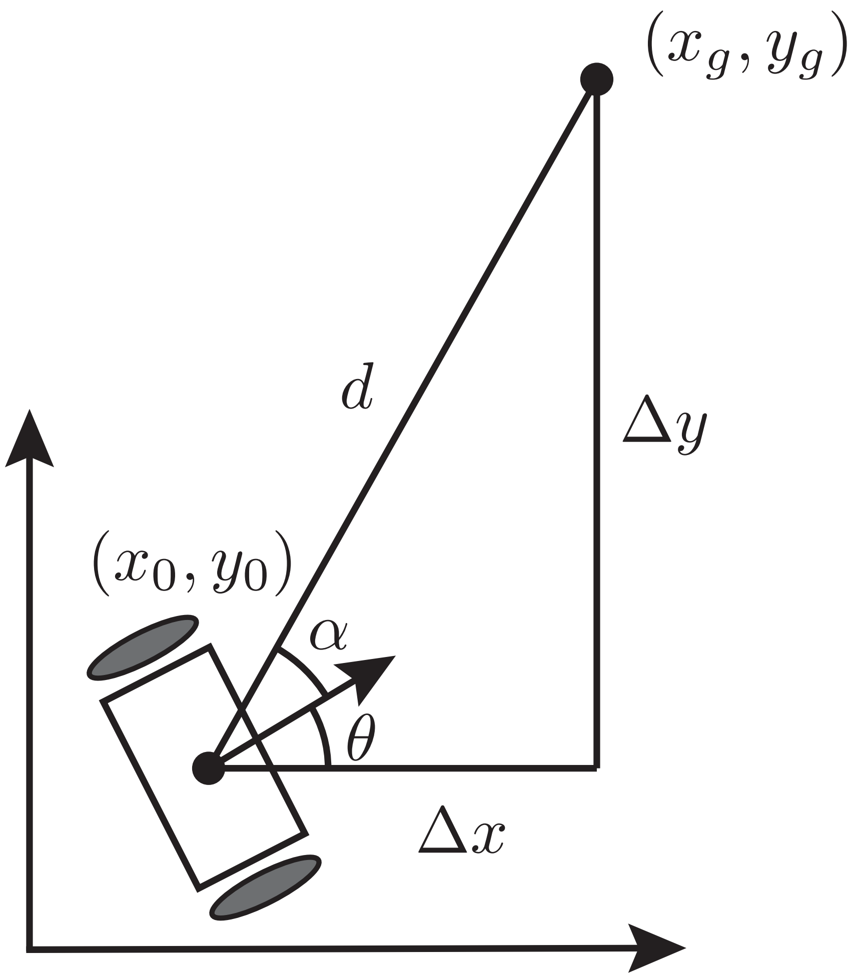
- Convert to polar coordinates in robot frame:

$$d = \sqrt{\Delta x^2 + \Delta y^2}$$
$$\alpha = \mathrm{atan2}(\Delta y, \Delta x) - \theta$$

- Control Law:

$$v_{sp} = K_v d$$
$$\omega_{sp} = K_\omega \alpha$$

- What should happen if goal is behind us?



RTR Controller

- Rotate toward desired goal, until $\alpha \approx \mathbf{0}$

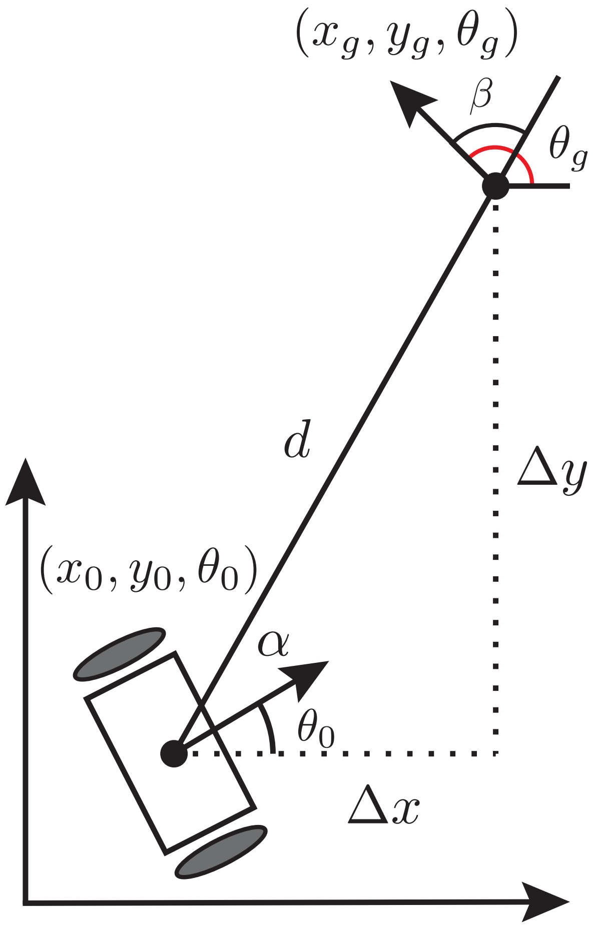
$$\omega_{sp} = K_\omega \alpha$$
$$v_{sp} = 0$$

- Maintain heading and drive to goal until $d \approx \mathbf{0}$

$$\omega_{sp} = K_\omega \alpha$$
$$v_{sp} = K_v d$$

- Rotate to final orientation

$$\omega_{sp} = K_\omega \beta$$
$$v_{sp} = 0$$



$$\Delta x = x_g - x$$
$$\Delta y = y_g - y$$
$$d = \sqrt{\Delta x^2 + \Delta y^2}$$
$$\alpha = \mathrm{atan2}(\Delta y, \Delta x) - \theta$$
$$\beta = \theta_g - \theta$$

Feedback Control to a Pose

- It can be shown that with:

$$v = K_d d$$

$$\omega = K_\alpha \alpha + K_\beta \beta$$

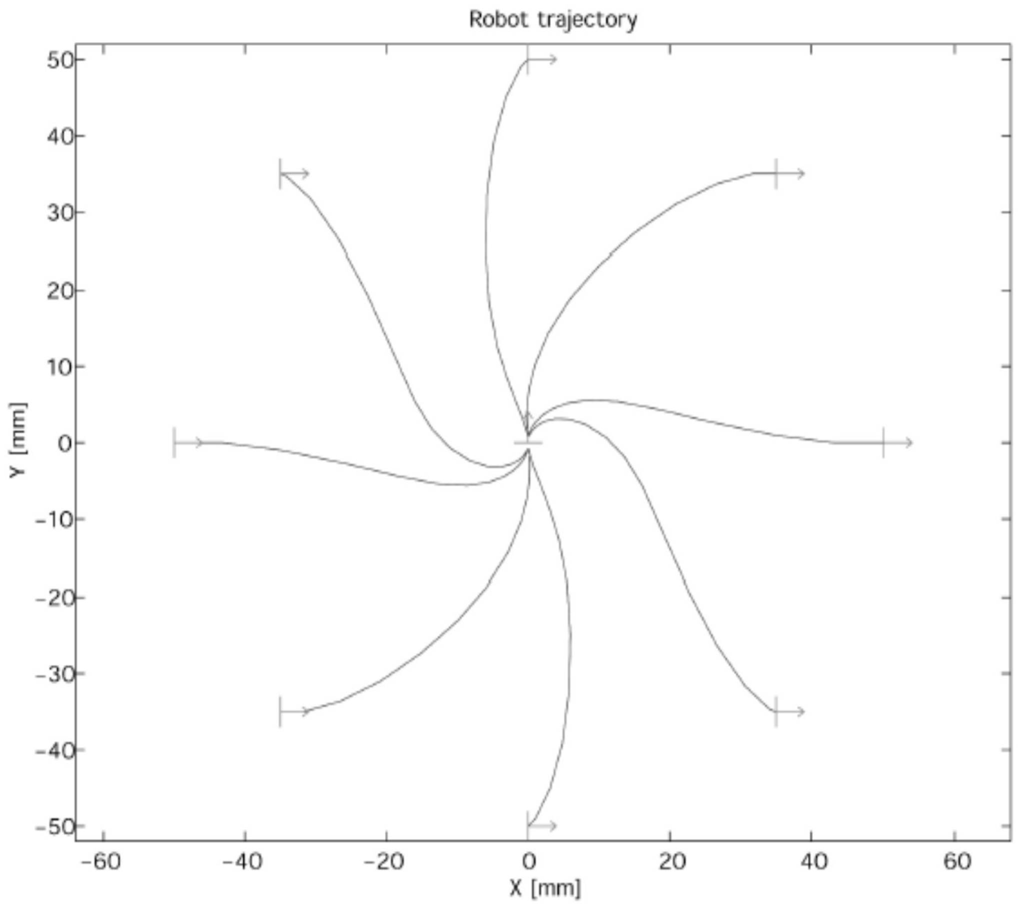
- The feedback controlled system shown will drive the robot to:

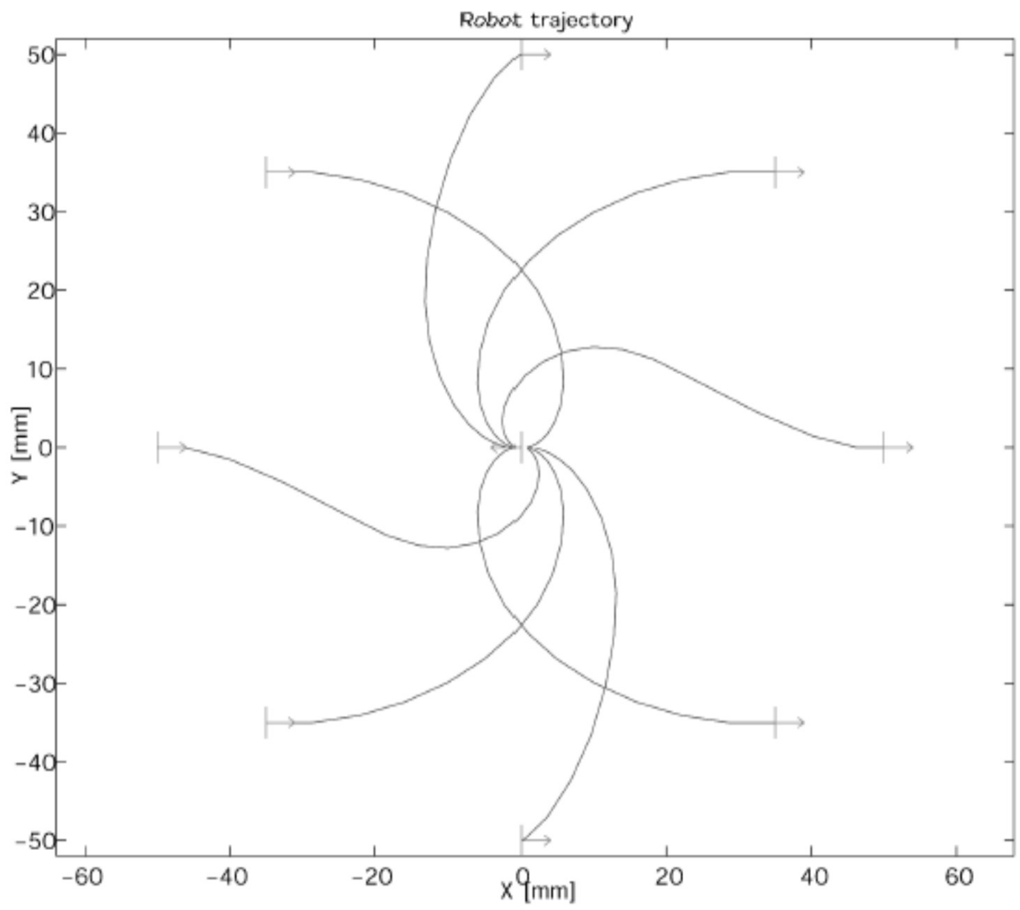
$$(\rho, \alpha, \beta) \to (0, 0, 0)$$

$$\begin{bmatrix} \dot{d} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -K_d d \cos\alpha \\ K_d \sin\alpha - K_\alpha \alpha - K_\beta \beta \\ -K_d \sin\alpha \end{bmatrix}$$

- Control signal v has constant sign
- Can prove stability if: $K_d > 0; K_\beta < 0; K_\alpha - K_d > 0$
- See Siegwart & Nourbakhsh

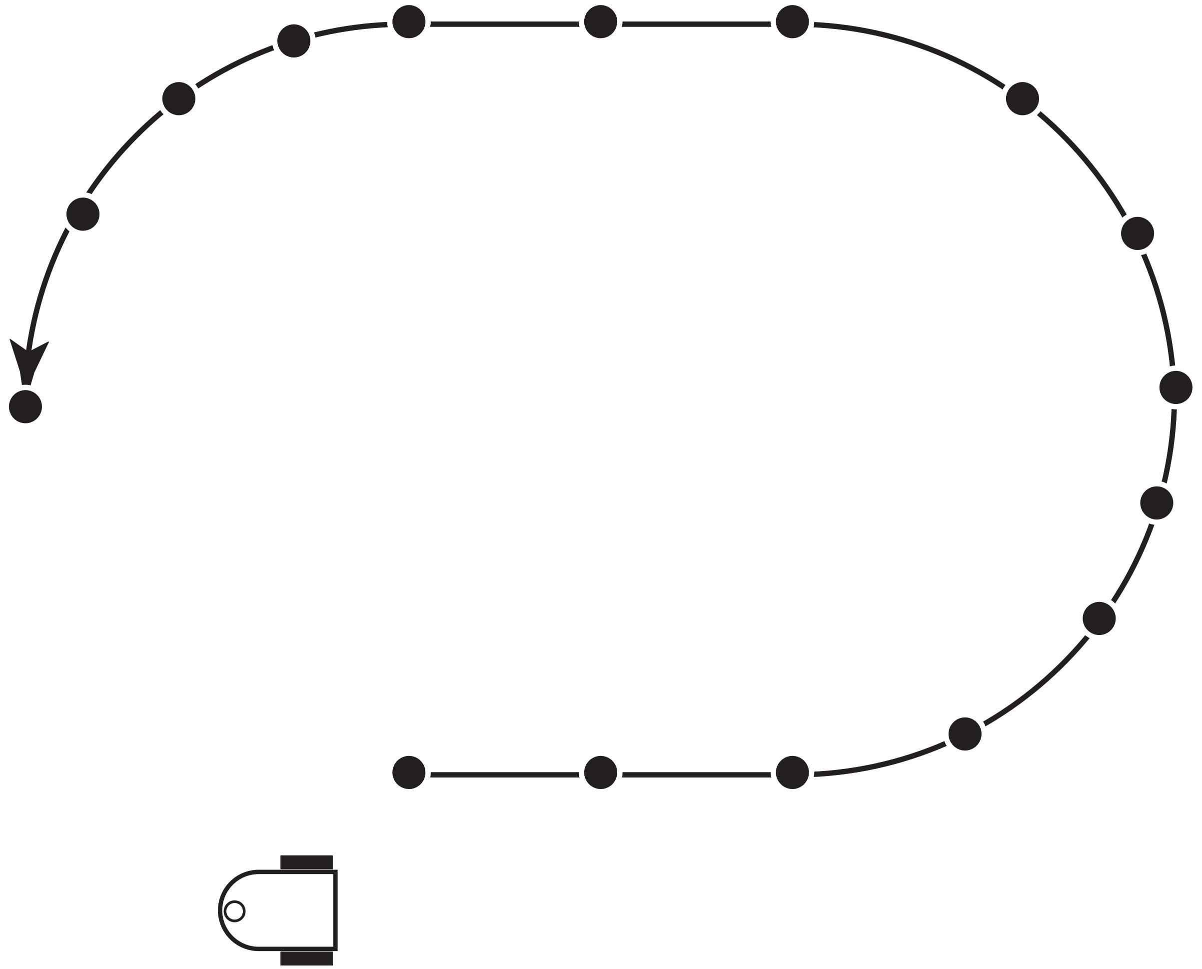
Resulting Path





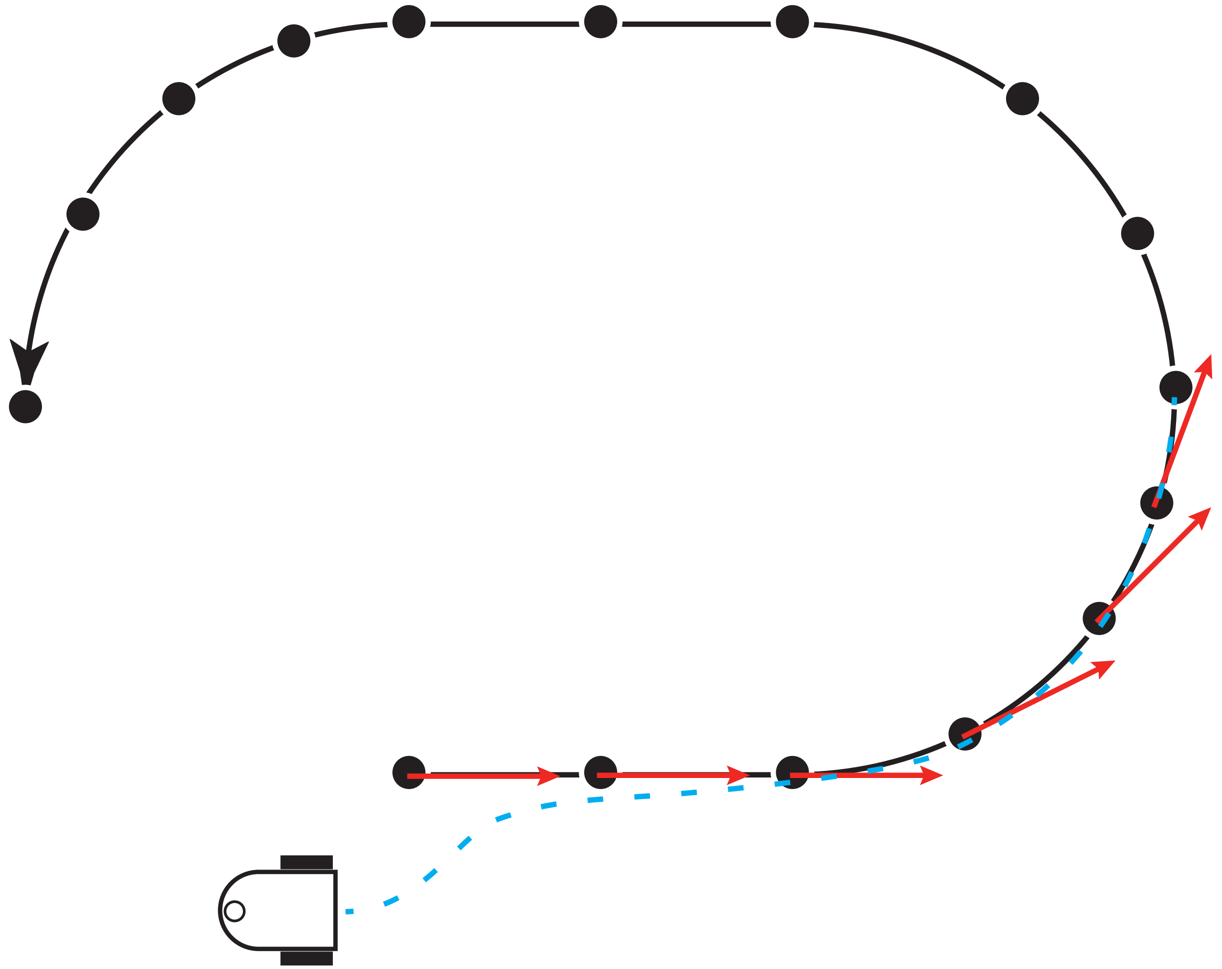
Carrot Following

- Consider a desired path $s(t)$
- Discretize path



Carrot Following

- Consider a desired path $s(t)$
- Discretize path
- Calculate Poses based on tangents
- Use pose control to next point
- Switch to next point when within threshold



Intro to Probabilistic Robotics

Probabilistic Robotics

Main idea: explicit representation of uncertainty using the calculus of probability theory

Perception = state estimation

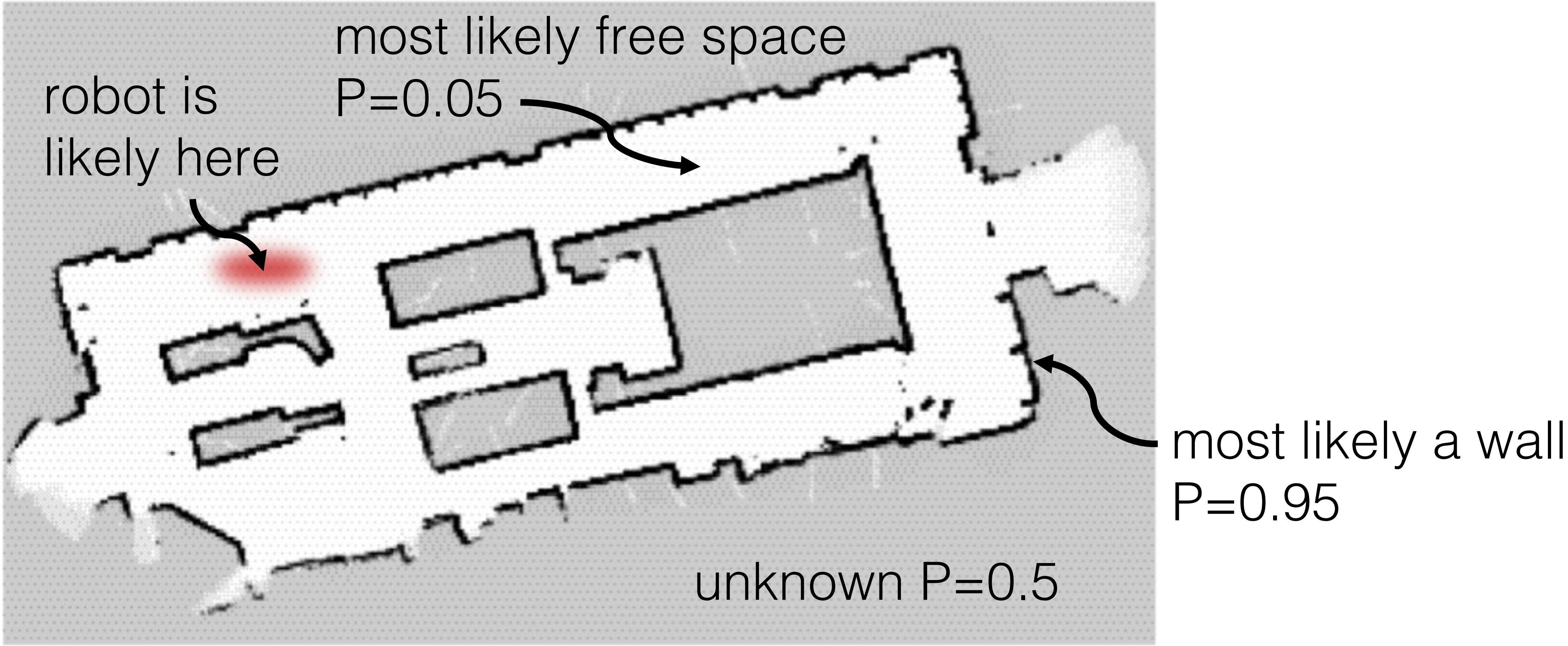
Action = utility optimization

More in-depth, from Wolfram Burgard: https://www.youtube.com/watch?v=kZNk4kSt7OM

Localization & Mapping with Uncertainty

- Uncertainty in location is product of uncertainty in actions of the robot (odometry) and map of the environment coupled with uncertainty in our sensors.
- Want to find *most probable* location given *most probable* map of the environment and knowledge about the errors introduced by our sensors

Localization & Mapping with Uncertainty



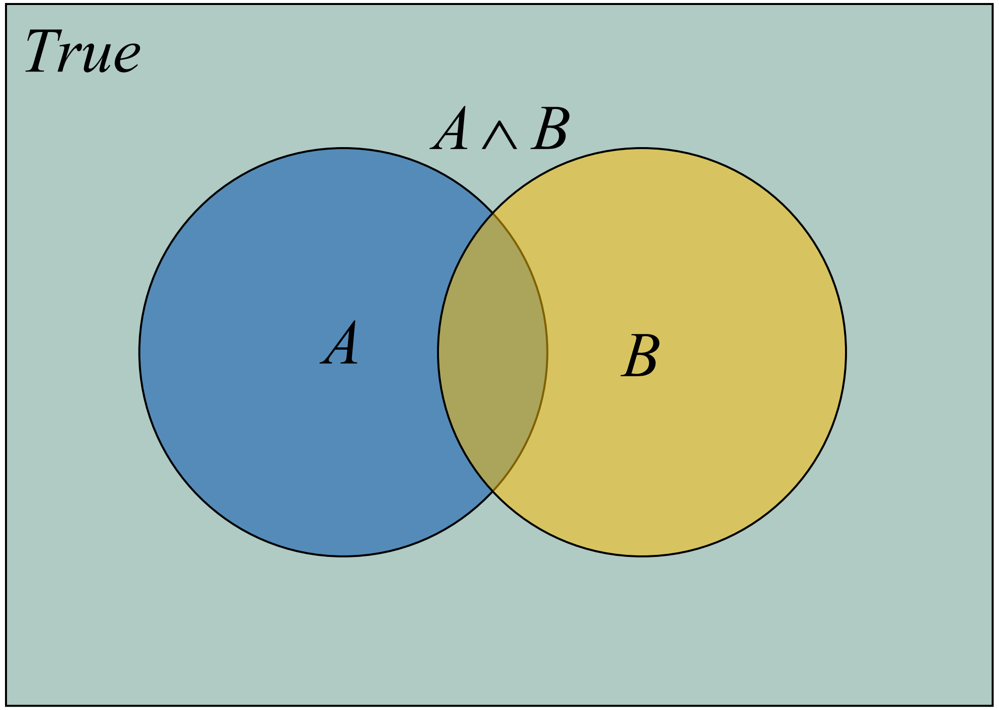
Basics of Probability

$\Pr(A)$ denotes probability that proposition A is true.

- $0 \le \Pr(A) \le 1$
- $\Pr(True) = 1 \quad \Pr(False) = 0$
- $\Pr(A \vee B) = \Pr(A) + \Pr(B) - \Pr(A \wedge B)$

A Closer Look at Axiom 3

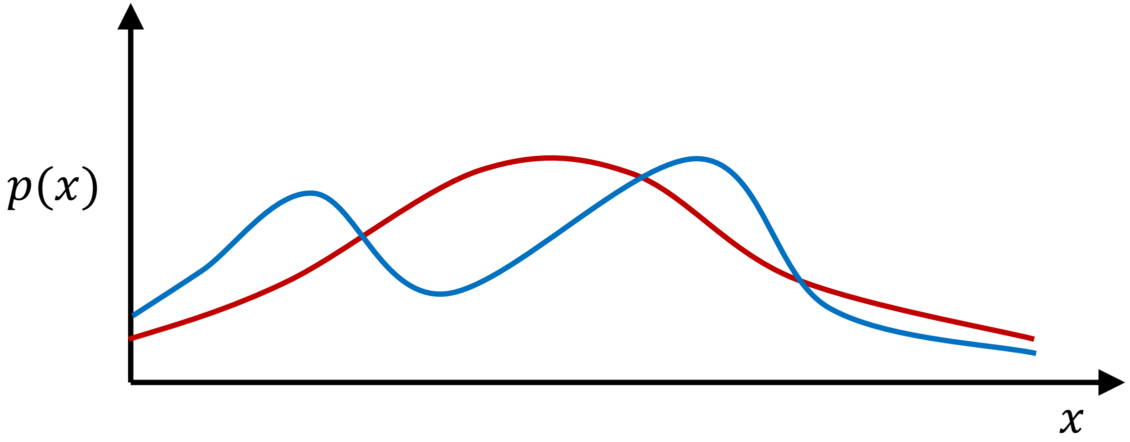
$$\Pr(A \vee B) = \Pr(A) + \Pr(B) - \Pr(A \wedge B)$$



Continuous Random Variables

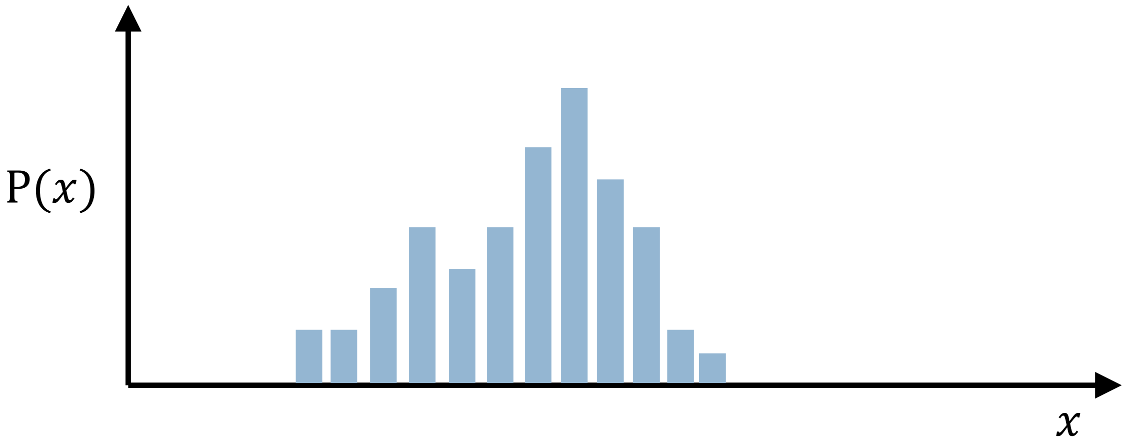
- X takes on any value in the continuum
- $p(X = x)$, or $p(x)$, is a probability density function

$$\Pr(x \in (a,b)) = \int_a^b p(x)dx$$



Discrete Random Variables

- X denotes a random variable
- X can take on a countable number of values in $\{x_1, x_2, \ldots, x_n\}$
- $P(X = x_i)$ or $P(x_i)$, is the probability that the random variable X takes on value x_i.
- $P(*)$ is called probability mass function



Joint and Conditional Probability

- Joint Probability: $P(X = x \,\&\, Y = y) \;=\; P(x, y)$

- If X and Y are independent then

$$P(x, y) \;=\; P(x)\,P(y)$$

- Conditional Probability: $P(x \mid y)$ “the probability of x given y”

$$P(x \mid y) \;=\; P(x, y) \,/\, P(y)$$

$$P(x, y) \;=\; P(x \mid y)\,P(y)$$

- If X and Y are independent:

$$P(x \mid y) \;=\; P(x)$$

Laws of Total & Marginal Probability

Discrete case	Continuous case
$\sum_x P(x) = 1$	$\int p(x)\,dx = 1$
$P(x) = \sum_y P(x, y)$	$p(x) = \int p(x, y)\,dy$
$P(x) = \sum_y P(x \mid y) P(y)$	$p(x) = \int p(x \mid y)\, p(y)\,dy$

Bayes Formula

$$P(x, y) = P(x \mid y)P(y) = P(y \mid x)P(x)$$

$$\Rightarrow$$

$$P(x \mid y) = \frac{P(y \mid x)\ P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

- x is value of interest
- y is new information

Bayes' Rule

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

- If A is a quantity we would like to infer from data B:
- $p(A)$ is the *prior* probability of A
- $p(A|B)$ is the *posterior* probability after taking data B into account
- $p(B|A)$ is the *likelihood* of the data given the hypothesis
- $p(B)$ is the *prior* probability of the data and effectively normalizes the posterior.

Why is Bayes' Rule so useful?

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

- Diagnostic evidence $p(disease \mid symptom)$ is often hard to get but what you want to know.
- Causal evidence $p(symptom \mid disease)$ is often easier to get.
- $p(disease)$ is easy to get
- $p(symptom)$ is just a normalizer

Bayes Filters: Framework

- Given:
  - Stream of observations z and action data u: $d_t = \{u_1, z_2 \ldots, u_{t-1}, z_t\}$
  - Sensor model $P(z|x)$.
  - Action model $P(x|u,x')$.
  - Prior probability of the system state $P(x)$.
- Wanted:
  - Estimate of the state X of a dynamical system.
  - The posterior of the state is also called Belief:

$$Bel(x_t) = P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

Goal: SLAM Problem

Given:

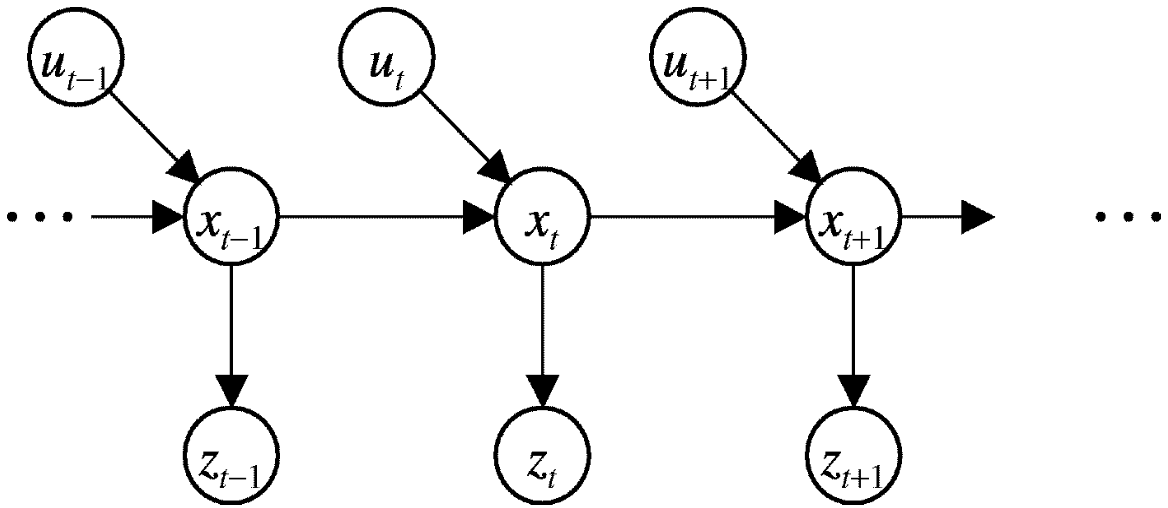
- Robot Commands: $\mathbf{U}_{0:k} = \{u_1, u_2, ..., u_k\}$
- Sensor Measurements: $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k\}$

Desired:

- Map of features: $\mathbf{m} = \{m_1, m_2, ..m_n\}$
- Path of robot: $\mathbf{X}_{0:k} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\}$

$$p(m, \mathbf{X}_{0:k} | \mathbf{U}_{0:k}, \mathbf{Z}_{0:k})$$

Markov Assumption



$$p(z_t \mid x_{0:t}, z_{1:t}, u_{1:t}) = p(z_t \mid x_t)$$

Underlying Assumptions

- Static world
- Given the present, the future is independent of the past
- Given the state x_t, the observation z_t is independent of the past

Examples of Bayes Filters

$$Bel(x_t) = \eta \ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \ Bel(x_{t-1}) \ dx_{t-1}$$

- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)

Bayes Filter Algorithm

$$Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}$$

Algorithm Bayes_filter(Bel(x),d):

η=0

If d is a perceptual data item z then

For all x do

$$Bel'(x) = P(z \mid x) Bel(x)$$

$$\eta = \eta + Bel'(x)$$

For all x do

$$Bel'(x) = \eta^{-1} Bel'(x)$$

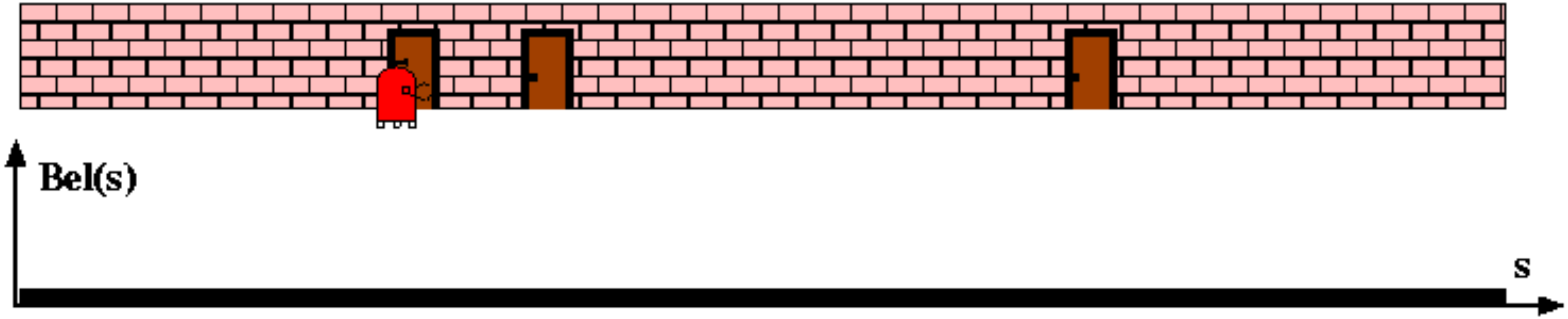
Else if d is an action data item u then

For all x do

$$Bel'(x) = \int P(x \mid u, x') \; Bel(x') \; dx'$$

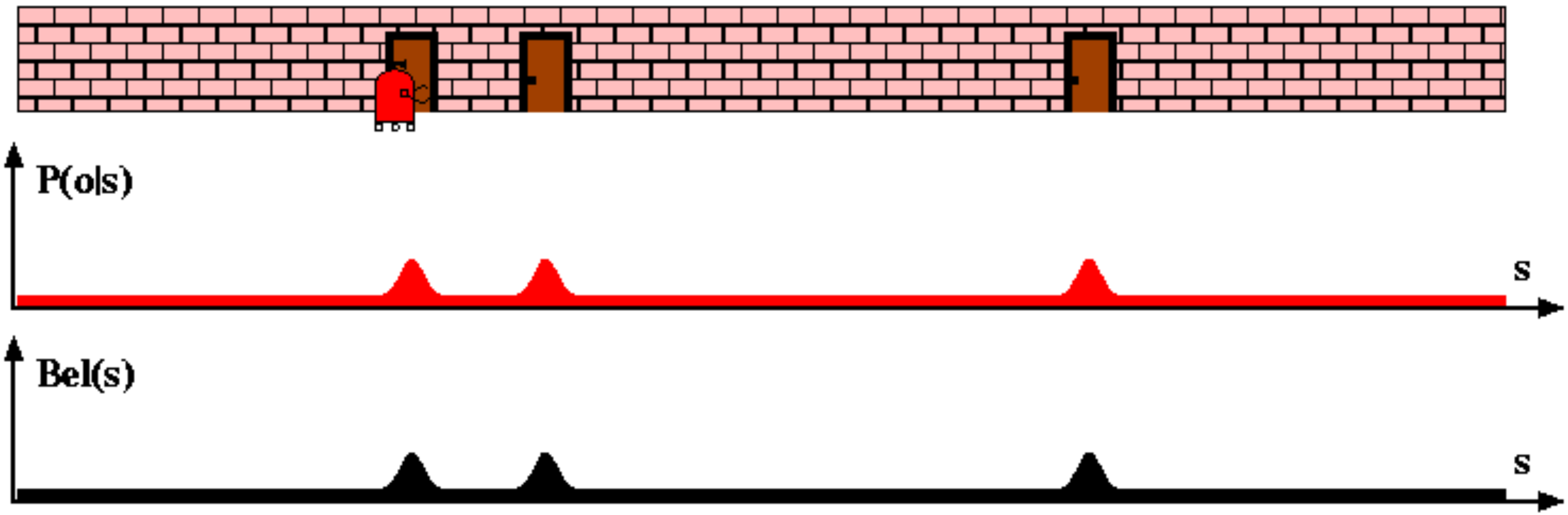
Return Bel'(x)

Example Localization Filter



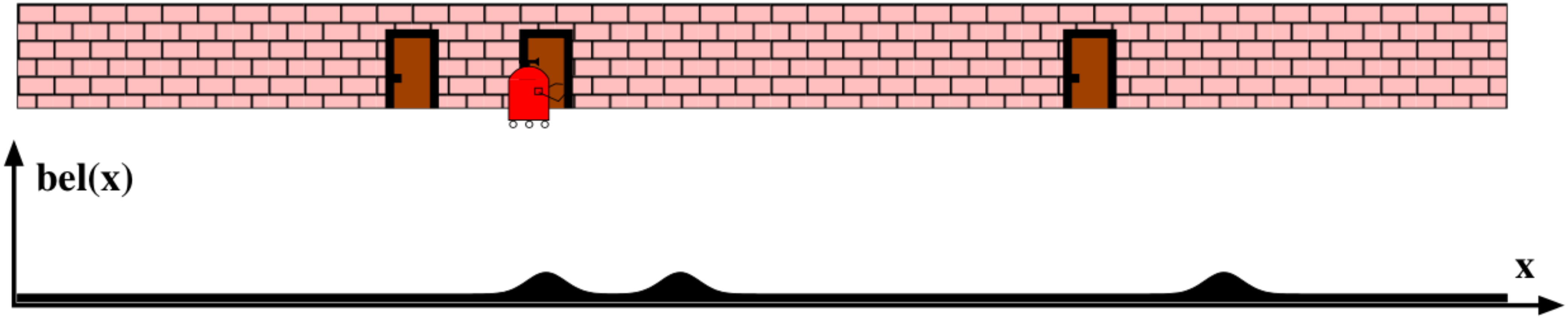
- Begin with unknown location in known map
- Uniform prior probability $Bel^{-}(x_0)$

Example Localization Filter



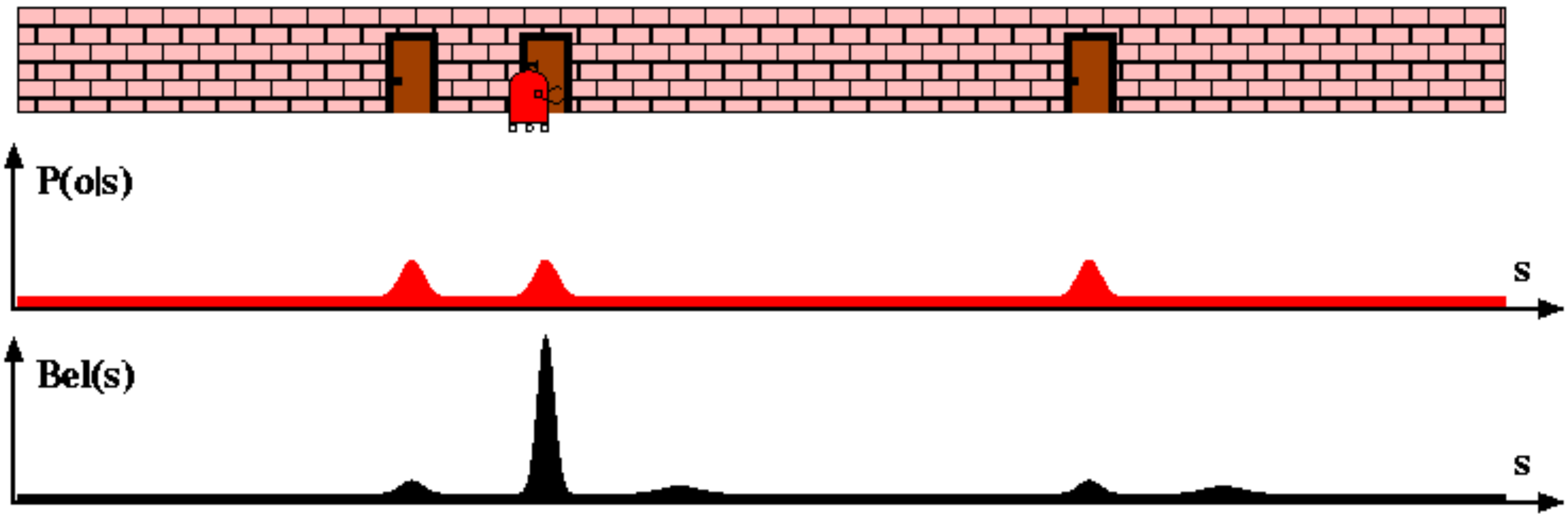
- Sensor observation $P(z_0|x_0)$
- $Bel(x_0) = \eta P(z_0|x_0) Bel^{-}(x_0)$

Example Localization Filter



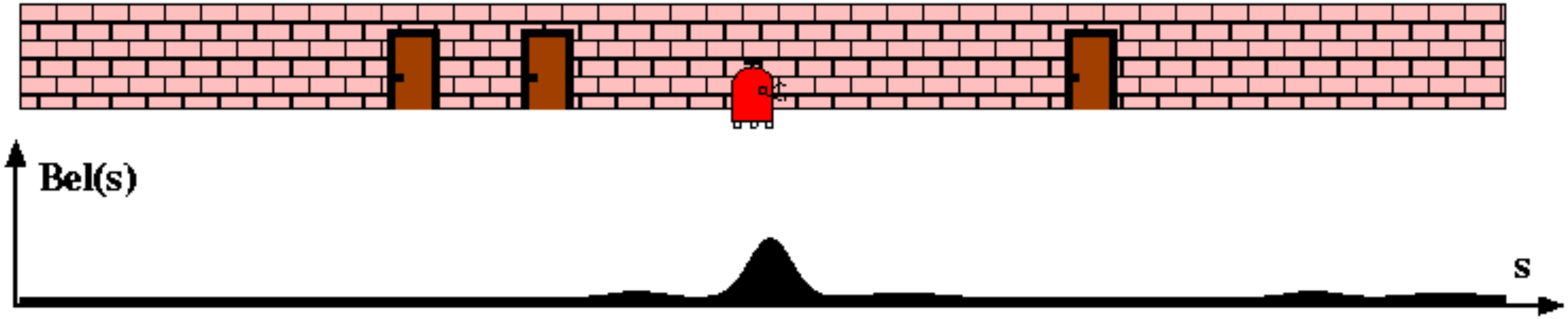
- Apply action model $P(x_1|u_0, x_0)$
- Proposal: $Bel^{-}(x_1) = \int P(x_1|\, u_0, x_0)\, Bel\,(x_0)dx_0$

Example Localization Filter



- Combine with sensor observation $P(z_0|x_0)$
- $Bel(x_1) = \eta P(z_1|x_1) Bel^{-}(x_1)$

Example Localization Filter



- Apply action model again: $P(x_1|u_0, x_0)$
- Proposal: $Bel^{-}(x_2) = \int P(x_2|\, u_1, x_1)\; Bel\;(x_1) dx_1$