

**CSE398/498-011**  
**Advanced Topics in Mobile Robotics**

**A Brief Overview of Particle Filters**  
**13 Feb 07**



**LEHIGH**  
UNIVERSITY

Department of  
Computer Science & Engineering

P.C. Rossin  
College of Engineering  
and Applied Science

CSE398 Advanced Topics in Mobile Robotics

## Administration

---

- Lab internships available this summer for working on the [ATRS](#)
  - *1 graduate & 1 undergraduate positions available*
  - *Requires strong C++ programming background (we're writing code for a commercial robot product)*
  - *If you're interested (or know someone that might be), please let me know*

© JR Spletzer

## Group Updates

---

- General:
  - *Status of PW protected wiki*
- LIDAR Group (Jason, Doug, Hisham):
  - *Other 291s are backordered*
  - *RS422 adapters are in*
  - *Would like the connector soldered TODAY (I will do this if necessary)*
  - *Would like this tested w/500 kbps ASAP*
- Vision Group (Chris, Charles)
  - *Need a main, make file, etc.*
    - *Load image from AVI*
    - *Display image*
    - *Space for image processing*
    - *Display processed image*
  - *Need AVI posted w/lane images (preferably from on Mountain Drive South going north)*

© R. Siegwart, I. Nourbakhsh

## Group Updates

---

- Vision Group (Chao, Thomas)
  - *Focus on data analysis at the Goodman site*
    - *Detect Road*
    - *Detect vehicle parked in a lane*
  - *Focus on more generic lane tracking (general video)*
- Vehicle Group (Jason, Anthony, Matt)
  - *Hardware should be in either today or tomorrow*
  - *Want the rack constructed THAT DAY!*
  - *If there are issues, I want to know ASAP!*
- Test Group (Jason, Anthony, Matt)
  - *Need test procedures list for all to use posted on the Wiki*
  - *Want additional data collected on Thursday if practicable*
    - *Go to Goodman and collect single camera data for a loop with a car parked on each of the 4 sides (4 loops total)*
  - *Want LIDAR data as well for the same trials*
    - *Looking for curbs and automobiles...*

© R. Siegwart, I. Nourbakhsh

## Group Updates

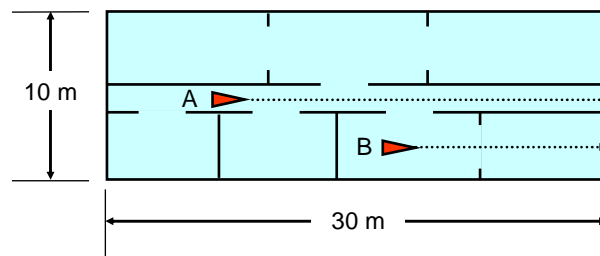
---

- GPS (Charles)
  - *Need a GPS driver ASAP.*
  - *There is a Windows driver that can be used for data format.*
- Goal: By this Friday, I want time-stamped data from
  - *Bumblebee2*
  - *SICK (range and reflectivity)*
  - *GPS (?)*

© R. Siegwart, I. Nourbakhsh

## Motivation

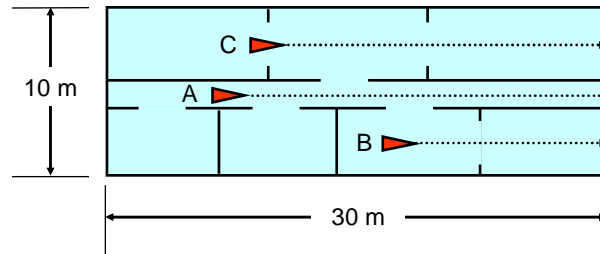
---



- The robot knows its position to be either A or B. It measures the distance to the wall in front of it to be 22 meters. Which position is the robot located?

© JR Spletzer

## Motivation



- The robot knows its position to be either A or B. It measures the distance to the wall in front of it to be 22 meters. Which position is the robot located?
- What if there is a third possible position C?

© JR Spletzer

## Basic Probability Review: Some Theorems

- $p(A | B)$ : The probability of A *given that B has occurred* (the conditional probability of A on B)
- Theorem of Compound Probability:

$$p(A \wedge B) = p(A \cap B) = p(B | A) p(A)$$

$$p(A \wedge B) = p(A \cap B) = p(A | B) p(B)$$

- Bayes Law

$$p(B) p(A | B) = p(A) p(B | A)$$

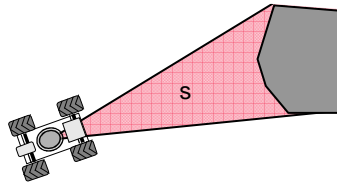
$$\Rightarrow p(B | A) = \frac{p(A | B) p(B)}{p(A)}$$

© JR Spletzer

## A Simple Example (1)

---

- Suppose our robot is trying to detect obstacles from a measurement  $s$
- What is  $p(\text{obstacle}/s)$ ?



© JR Spletzer

## A Simple Example (2)

---

- $p(\text{obstacle}/s)$  in practice is difficult to measure explicitly
- Since we have a sensor model, it is often easier to get  $p(s/\text{obstacle})$
- By applying Bayes law we obtain

$$\Rightarrow p(\text{obst} | s) = \frac{p(s | \text{obst}) p(\text{obst})}{p(s)}$$

© JR Spletzer

## Normalization Process

$$p(\text{obst} | s) = \frac{p(s | \text{obst}) p(\text{obst})}{p(s)}$$

$$p(!\text{obst} | s) = \frac{p(s | !\text{obst}) p(!\text{obst})}{p(s)}$$

$$p(\text{obst} | s) + p(!\text{obst} | s) = 1$$

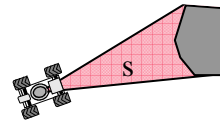
$$\Rightarrow p(s) = p(s | \text{obst}) p(\text{obst}) + p(s | !\text{obst}) p(!\text{obst})$$

$$p(\text{obst} | s) = \frac{p(s | \text{obst}) p(\text{obst})}{p(s | \text{obst}) p(\text{obst}) + p(s | !\text{obst}) p(!\text{obst})}$$

© JR Spletzer

## A Simple Example (3)

- Suppose our robot is trying to detect obstacles from a measurement  $s$
- What is  $p(\text{obstacle} | \text{detection})$  IF
  - $p(\text{obstacle}) = 0.1$
  - $p(\text{detection} | \text{obstacle}) = 0.9$
  - $p(\text{detection} | \text{no obstacle}) = 0.05$



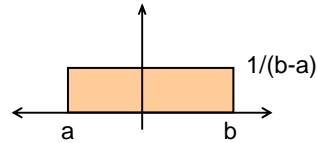
$$p(o | s) = \frac{0.9 * 0.1}{0.9 * 0.1 + 0.05 * 0.9} = \frac{2}{3}$$

© JR Spletzer

## What is a Particle Filter? (1)

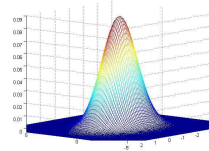
- The *uniform distribution* is defined as

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

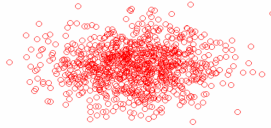


- Our friend, the Gaussian distribution

$$p(\vec{x}) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left[\frac{(x_1-\mu_1)^2}{2\sigma_1^2} + \frac{(x_2-\mu_2)^2}{2\sigma_2^2}\right]}$$



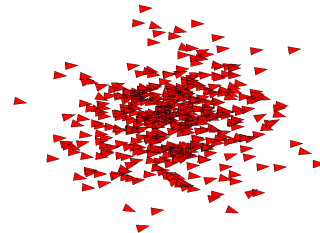
- Particle distributions



© JR Spletzer

## What is a Particle Filter? (2)

- Particle Filters (PF) are known by a variety of names (Importance Sampling, the Metropolis Algorithm, Monte Carlo Methods, CONDENSATION algorithm, etc.)
- We will use the PF moniker from statistical literature
- PFs represent a probability density function as a set of discrete samples or particles, each representing a hypothetical estimate of the state
- For example, to estimate the pose of a robot on the plane each particle would correspond to a hypothetical position and orientation
- We can then take statistics over this set as necessary to estimate pose



© JR Spletzer

## Bayesian Filters (1)

---

- PFs and Kalman Filters (KF/EKF) are example of Bayesian Filters
- Bayesian filters do not *explicitly* estimate the state
- Instead, they propagate a *posterior* probability density function for the state from which it can be inferred
- In the KF, a gaussian distribution  $\mathbf{P}$  is propagated at each timestep with mean  $\mu$  and variance  $\sigma^2$ . The former is used as the state estimate
- In the PF, a (weighted) particle set corresponds to the posterior from which an estimate for the state can be inferred

© JR Spletzer

## Bayesian Filters (2)

---

- Let  $\mathbf{x}_t$  denote the state at time  $t$
- Bayesian filters estimate the conditional pdf for  $\mathbf{x}_t$  or the *belief* denoted by

$$Bel(\bar{\mathbf{x}}_t) = p(\bar{\mathbf{x}}_t | d_{0:t})$$

where  $d_{0:t}$  corresponds to all of the available data that the pdf has been conditioned upon

- For robot localization,  $d_{0:t}$  would correspond to sensor measurements and robot motions. Denoting these as  $z$  and  $u$ , respectively we obtain

$$Bel(\bar{\mathbf{x}}_t) = p(\bar{\mathbf{x}}_t | z_t, u_t, z_{t-1}, u_{t-1}, \dots, z_0, u_0)$$

where the belief is conditioned upon *all* available sensor measurements and robot actions

© JR Spletzer

## The Particle Filter

- Unlike the KF, which represents the *pdf* parametrically as a gaussian, the PF approximates it as a sample set

$$Bel(\bar{x}) \approx \{x^i, w^i\} \quad i \in [1..m]$$

- $m$  denotes the number of particles in the sample set
  - $x^i$  corresponds to a hypothetical state estimate
  - $w^i$  corresponds to a weight reflecting a “confidence” in how well the particle  $x^i$  reflects the true state  $x$
- $\sum_m w^i = 1$ , so that the sample set corresponds to a discrete probability density function
  - It has been shown that as the number of samples approaches infinity, the sample set converges to the true posterior [Tanner, *Tools for Statistical Inference*, 1996]. However, no proofs for rates of convergence exist

© JR Spletzer

## The Particle Filter Algorithm

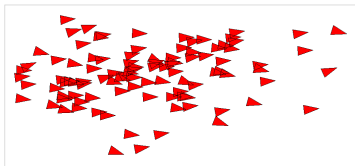
```
function  $X_{k+1} = \text{runFilter}(X_k, z_k, u_k)$ 
 $X_{k+1} = \emptyset$ 
for  $i = 1 : m$ 
  generate random  $x$  from  $X_k$  based on sample weights;
  generate random  $x' \sim p(x' | u_k, x)$ ;
   $w = p(z_k | x')$ ;
  Insert  $(x', w) \in X_{k+1}$ ;
end
Normalize weight factors  $\forall w_i \in X_{k+1}$ ;
return  $X_{k+1}$ ;
```

© JR Spletzer

## Predictor-Corrector Example

1) We have a *prior* of uniform weighted particle

$t = k^-$



At this point, we have  $m$  unique samples

2) Particles are weighted based on the sensor measurement and resampled according to weight to generate our posterior.

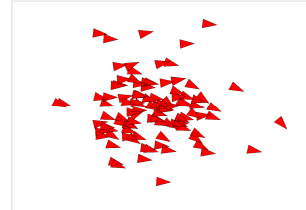
$t = k$



We still have  $m$  samples, but they are all *equally weighted* and not necessarily unique

3) Particles are passed through our motion model to generate a new posterior

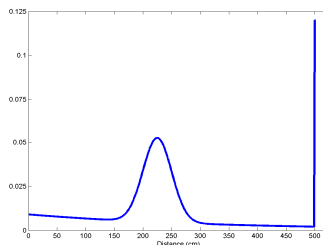
$t = (k + 1)^-$



Particles are again unique and equally weighted

## Generating Particle Weights

- For Monte-Carlo Localization (MCL), we require a mechanism to weight the individual particles
- This is accomplished via a sensor model that is typically generated empirically
- The resulting probability distribution yields a weight associated with each particle



## Monte Carlo Localization

---

