Modeling

CSC752 Autonomous Robotic Systems

Ubbo Visser

Department of Computer Science
University of Miami

October 5, 2022

UNIVERSITY
OF MIAMI

Outline

1. Modeling and state estimation
2. Examples
3. State estimation
4. Probabilities
5. Bayes filter
6. Particle filter

Modeling

- The model represents the current state of the environment.

Modeling

- The model represents the current state of the environment.
- All sensors of a physical robot are noisy.
- The model can never be exact.

Modeling

- The model represents the current state of the environment.
- All sensors of a physical robot are noisy.
- The model can never be exact.
- Robots can only estimate states using probabilistic methods for example.

State estimation

- Determines a state X_t that changes over time using a sequence of measurements z_t and u_t.
  - z_t: measurement
  - u_t: state transition measurement

State estimation

- Determines a state X_t that changes over time using a sequence of measurements z_t and u_t.
  - z_t: measurement
  - u_t: state transition measurement
- Useful if a state can not be accurately and directly measured (which means every state for a physical robot).

State estimation

- Determines a state X_t that changes over time using a sequence of measurements z_t and u_t.
  - z_t: measurement
  - u_t: state transition measurement
- Useful if a state can not be accurately and directly measured (which means every state for a physical robot).
  - filter noise

State estimation

- Determines a state X_t that changes over time using a sequence of measurements z_t and u_t.
  - z_t: measurement
  - u_t: state transition measurement
- Useful if a state can not be accurately and directly measured (which means every state for a physical robot).
  - filter noise
  - infer a state from measurements

State estimation

- Determines a state X_t that changes over time using a sequence of measurements z_t and u_t.
  - z_t: measurement
  - u_t: state transition measurement
- Useful if a state can not be accurately and directly measured (which means every state for a physical robot).
  - filter noise
  - infer a state from measurements
- Modeling in our soccer agent
  - Ball tracking, opponent localization (and teammates), self-localization, orientation estimation (upright vector).

Examples

Examples

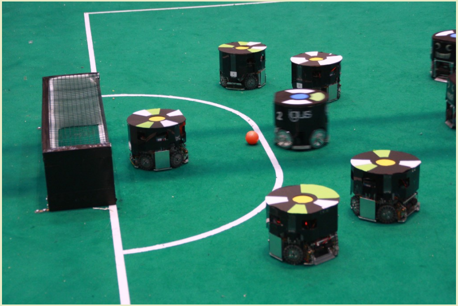
- How noisy can measurements be?

Examples

- How noisy can measurements be?
- How can a state estimation be robust despite all the errors?

Example 1

RoboCup Small-Size League:



Example 1

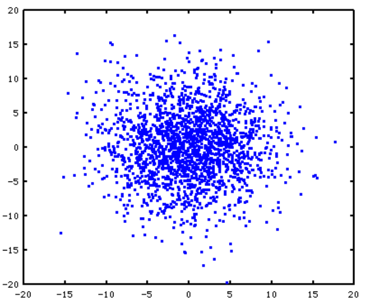
RoboCup Small-Size League:

- x,y positions as measurement z_t.

Example 1

RoboCup Small-Size League:

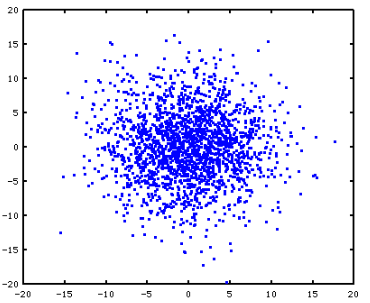
- x,y positions as measurement z_t.
- Only noisy measurements, we need the actual state (the model).

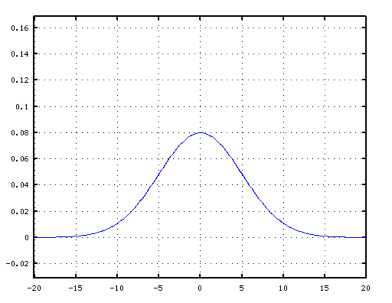


Example 1

RoboCup Small-Size League:

- x,y positions as measurement z_t.
- Only noisy measurements, we need the actual state (the model).

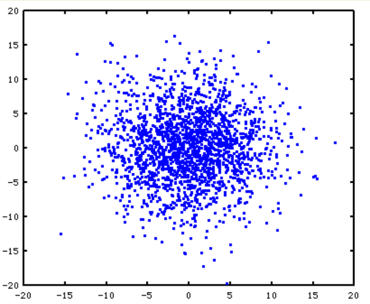


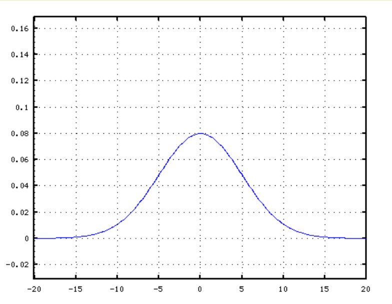


Example 1

RoboCup Small-Size League:

- x,y positions as measurement z_t.
- Only noisy measurements, we need the actual state (the model).





- Problem with two robots: wrong perceptions on other robot.

Example 2

Obstacle avoidance using a laser range finder:

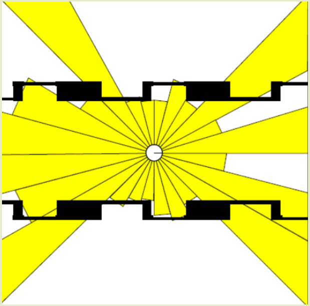
- There can be several different errors in the measurements.

source: slides from http://robots.stanford.edu/probabilistic-robotics/

Example 2

Obstacle avoidance using a laser range finder:

- There can be several different errors in the measurements.

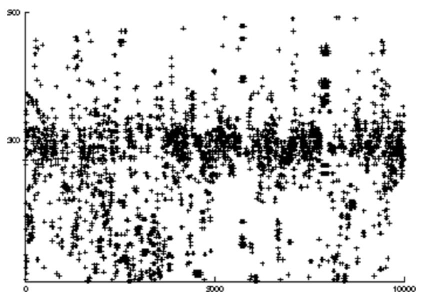


source: slides from http://robots.stanford.edu/probabilistic-robotics/

Example 2

Obstacle avoidance using a laser range finder:

- There can be several different errors in the measurements.

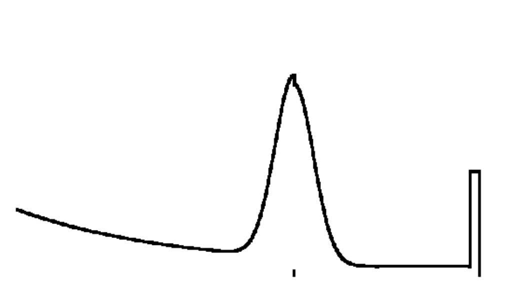


source: slides from http://robots.stanford.edu/probabilistic-robotics/

Example 2

Obstacle avoidance using a laser range finder:

- There can be several different errors in the measurements.
- The general model for a beam based sensor is a mixture of several distributions.

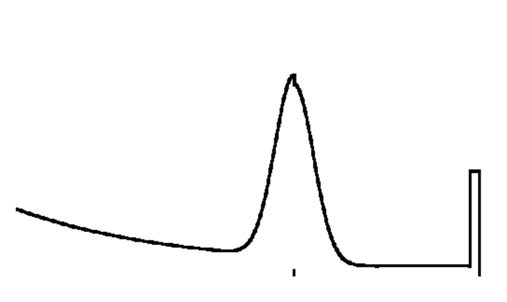


source: slides from http://robots.stanford.edu/probabilistic-robotics/

Example 2

Obstacle avoidance using a laser range finder:

- There can be several different errors in the measurements.
- The general model for a beam based sensor is a mixture of several distributions.



- Knowledge about the behavior of a sensor (the *sensor model*) is very important for a robust state estimation.

source: slides from http://robots.stanford.edu/probabilistic-robotics/

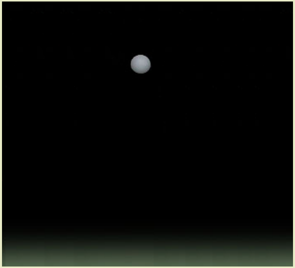
Example 3

3D ball-tracking with a camera:

Example 3

3D ball-tracking with a camera:

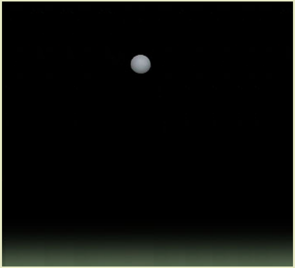
- Uncertainty, especially the distance of the ball to the camera.



Example 3

3D ball-tracking with a camera:

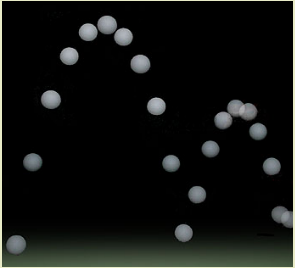
- Uncertainty, especially the distance of the ball to the camera.
- State in world coordinates and should include the velocity.
- A single observation does not contain much information.



Example 3

3D ball-tracking with a camera:

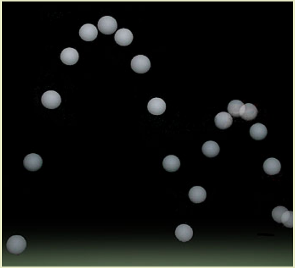
- Uncertainty, especially the distance of the ball to the camera.
- State in world coordinates and should include the velocity.
- A single observation does not contain much information.
- Consider only possible trajectories to reduce uncertainty.



Example 3

3D ball-tracking with a camera:

- Uncertainty, especially the distance of the ball to the camera.
- State in world coordinates and should include the velocity.
- A single observation does not contain much information.
- Consider only possible trajectories to reduce uncertainty.



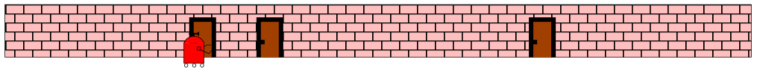
- Knowledge about the behavior of the ball and physics is useful (*state transition model*).

Example 4

Self-localization in 1D with limited sensors:

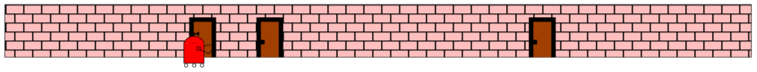
Example 4

Self-localization in 1D with limited sensors:



Example 4

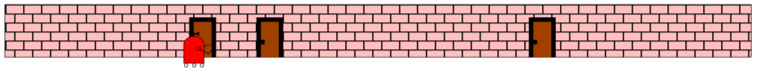
Self-localization in 1D with limited sensors:



- Door sensor → ambiguous.
- Even a sequence of measurements z_t is not enough to localize.

Example 4

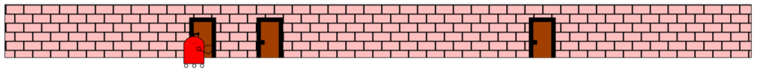
Self-localization in 1D with limited sensors:



- Door sensor → ambiguous.
- Even a sequence of measurements z_t is not enough to localize.
- Another sensor needed: sensor to measure wheel rotations.

Example 4

Self-localization in 1D with limited sensors:



- Door sensor → ambiguous.
- Even a sequence of measurements z_t is not enough to localize.
- Another sensor needed: sensor to measure wheel rotations.
- Measurements u_t needed (*odometry motion model*).

General state estimation

General state estimation

- For one given observation there is a high uncertainty and ambiguity.

General state estimation

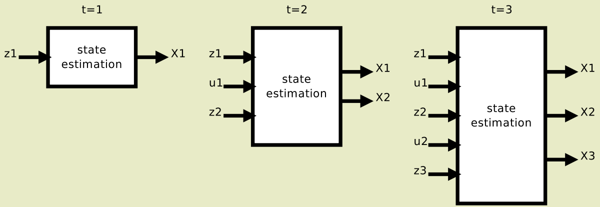
- For one given observation there is a high uncertainty and ambiguity.
- The state estimation gets a sequence of measurements, so the estimation of X_t is based on all measurements $z_0, \ldots, z_t$ and $u_0, \ldots, u_t$.

General state estimation

General state estimation:

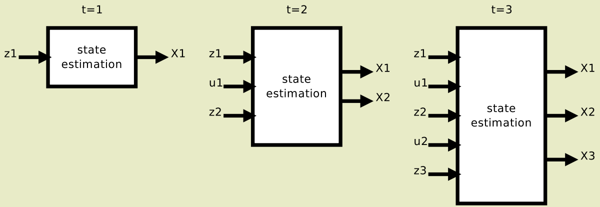
General state estimation

General state estimation:



General state estimation

General state estimation:



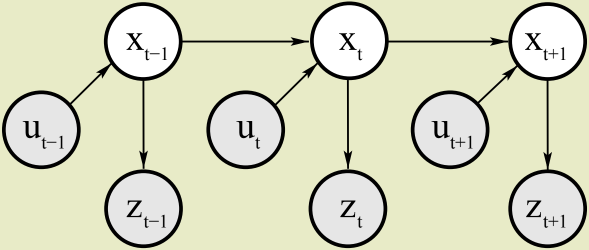
- Problems: more measurements with every time time step $\rightarrow$ increasing amount of computation.

Markov assumptions

- Markov assumption 1:
  The measurement z_t depends only on the state X_t and a random error.
- Markov assumption 2:
  The state transition measurement u_t only depends on the states X_t and X_{t+1} and a random error.

Markov process

Bayesian network with the measurements u_t and z_t:



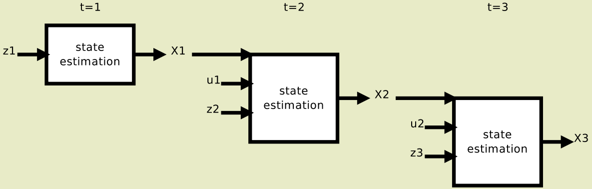
- The states x_t are hidden.

Recursive state estimation / filter

Recursive state estimation:

Recursive state estimation / filter

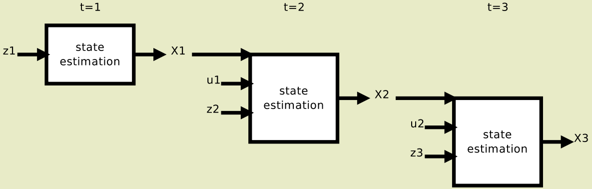
Recursive state estimation:



- X_t includes all the knowledge from the measurements before.

Recursive state estimation / filter

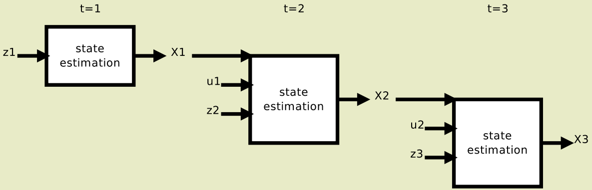
Recursive state estimation:



- X_t includes all the knowledge from the measurements before.
- Needed for X_t is only X_{t-1}, z_t and u_t.

Recursive state estimation / filter

Recursive state estimation:

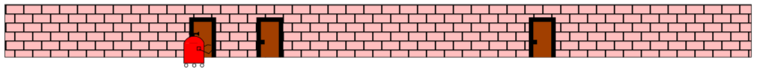


- X_t includes all the knowledge from the measurements before.
- Needed for X_t is only X_{t-1}, z_t and u_t.
- Belief X_t is updated using only the new measurements $\rightarrow$ constant time for each step.

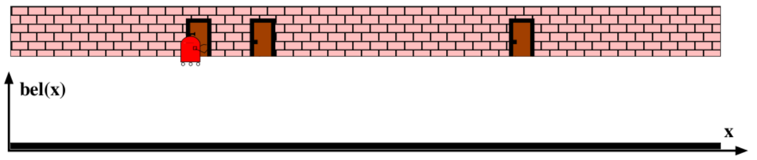
State estimation

- Sensor model and state transition model needed.
- Update belief X_t using
  - z_t and sensor model.
  - u_t and motion model and knowledge about dynamics in the environment.

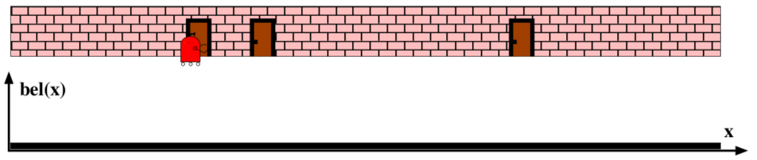
Example state estimation

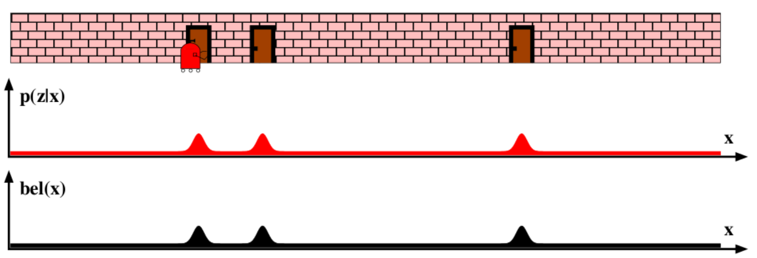


Example state estimation

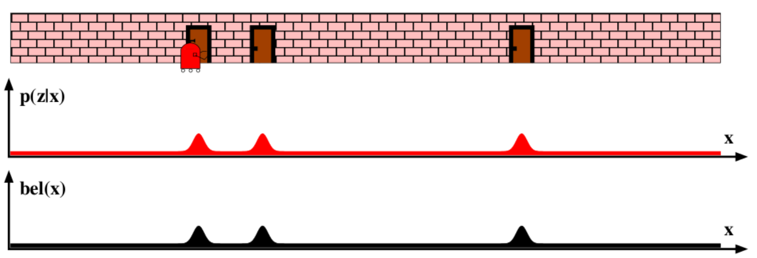


Example state estimation

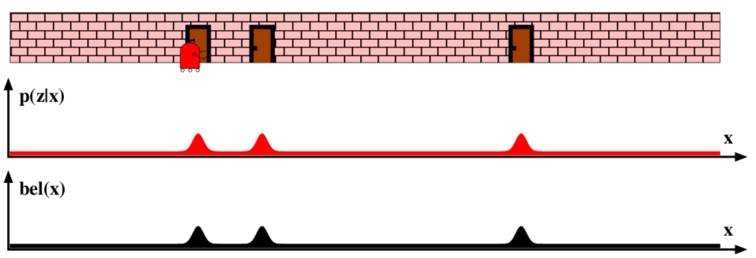


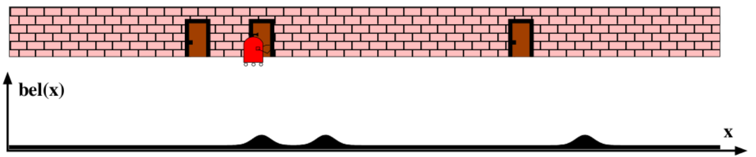


Example state estimation

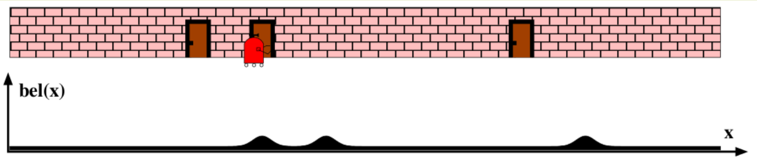


Example state estimation

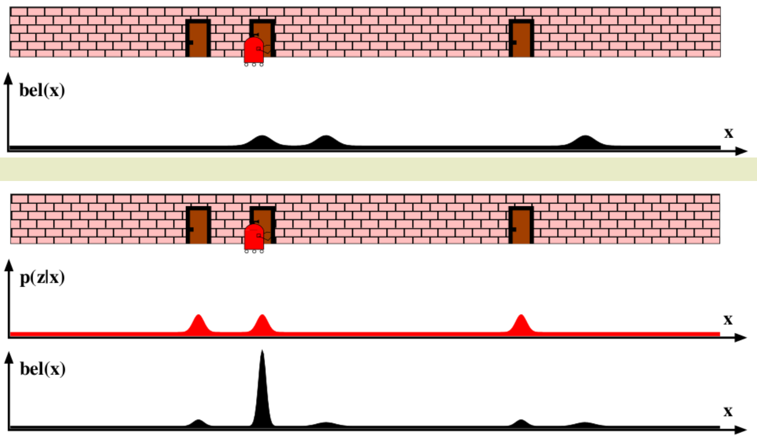




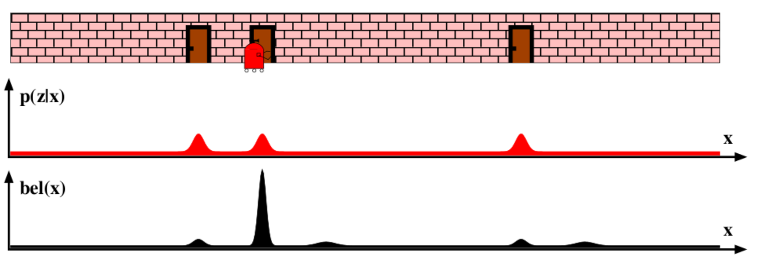
Example state estimation



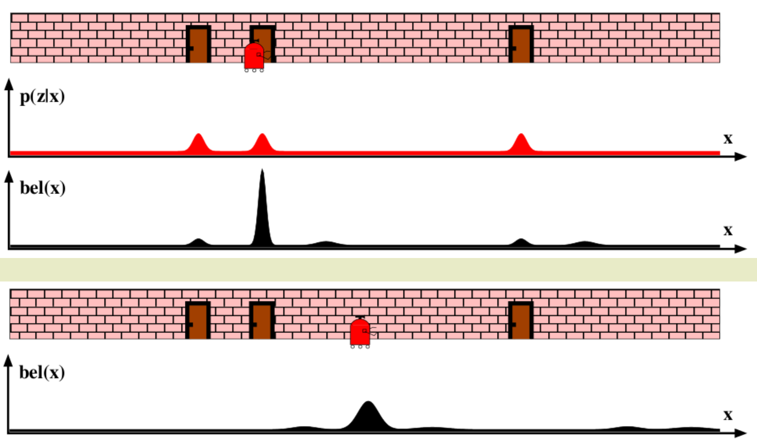
Example state estimation



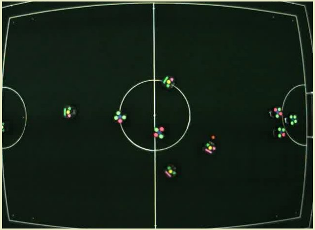
Example state estimation



Example state estimation

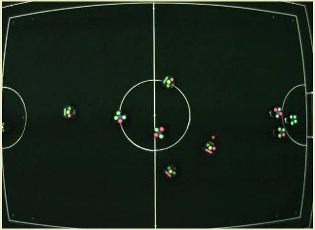


Example 1: Small-Size League



State, z_t, u_t, the sensor model and prediction?

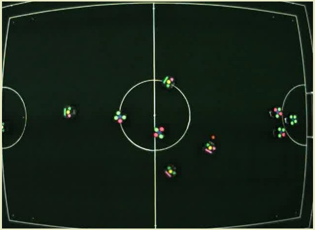
Example 1: Small-Size League



State, z_t, u_t, the sensor model and prediction?

- State: position x, y, θ and speed x', y', θ'

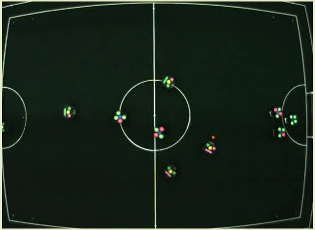
Example 1: Small-Size League



State, z_t, u_t, the sensor model and prediction?

- State: position x, y, θ and speed x', y', θ'
- z_t: x,y,θ

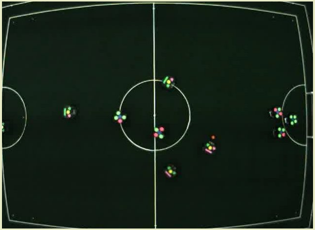
Example 1: Small-Size League



State, z_t, u_t, the sensor model and prediction?

- State: position x, y, θ and speed x', y', θ'
- z_t: x,y,θ
- u_t: Driving command sent to the robot.

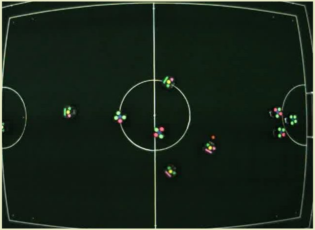
Example 1: Small-Size League



State, z_t, u_t, the sensor model and prediction?

- State: position x, y, θ and speed x', y', θ'
- z_t: x,y,θ
- u_t: Driving command sent to the robot.
- Sensor model:
  - Gaussian distribution around robot
  - Maybe also small probabilities at other robots

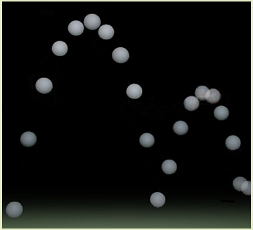
Example 1: Small-Size League



State, z_t, u_t, the sensor model and prediction?

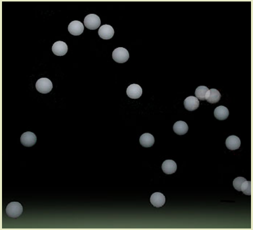
- State: position x, y, θ and speed x', y', θ'
- z_t: x,y,θ
- u_t: Driving command sent to the robot.
- Sensor model:
  - Gaussian distribution around robot
  - Maybe also small probabilities at other robots
- Prediction using X_{t-1}, u_t, odometry motion model

Example 3: Ball tracking



State, z_t, u_t, the sensor model and prediction?

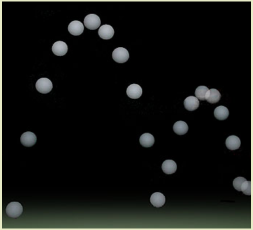
Example 3: Ball tracking



State, z_t, u_t, the sensor model and prediction?

- state: position x, y, z and velocity x', y', z'

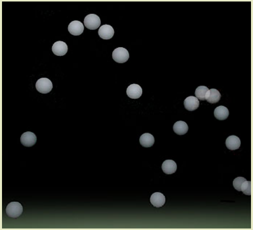
Example 3: Ball tracking



State, z_t, u_t, the sensor model and prediction?

- state: position x, y, z and velocity x', y', z'
- z_t: image x, y

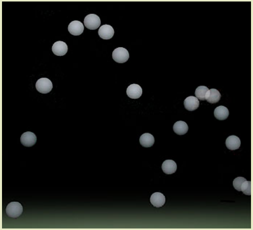
Example 3: Ball tracking



State, z_t, u_t, the sensor model and prediction?

- state: position x, y, z and velocity x', y', z'
- z_t: image x, y
- u_t: none

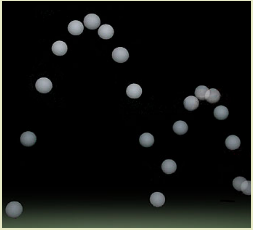
Example 3: Ball tracking



State, z_t, u_t, the sensor model and prediction?

- state: position x, y, z and velocity x', y', z'
- z_t: image x, y
- u_t: none
- Sensor model: transformation from state to image, Gaussian distribution in image

Example 3: Ball tracking



State, z_t, u_t, the sensor model and prediction?

- state: position x, y, z and velocity x', y', z'
- z_t: image x, y
- u_t: none
- Sensor model: transformation from state to image, Gaussian distribution in image
- Prediction: state transition model using physics

Bayes filter

- Previous slides have shown the principle of a *Bayes filter*.
- Why does this work exactly?
  - Probabilities
  - Bayes rule
  - Recursive Bayesian estimation

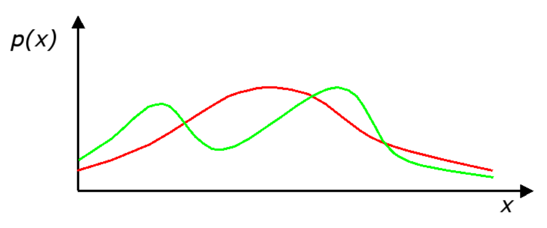
Source for the following slides: Thrun et al., Probabilistic Robotics; http://robots.stanford.edu/probabilistic-robotics/

Discrete random variables

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

Continuous random variables

- X takes on values in the continuum.
- $p(X = x)$ (or short $p(x)$) is a probability density function.
- Example: $Pr(x \in [a, b]) = \int_a^b p(x)dx$



Joint and Conditional Probabilities

- $P(X = x \text{ and } Y = y) = P(x, y)$.
- If X and Y are independent then $P(x, y) = P(x)P(y)$.
- $P(x|y)$ is the probability of x given y.
- If X and Y are independent then $P(x|y) = P(x)$.

Law of total probability

- Discrete case:

$$\sum_x P(x) = 1$$

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

$$P(x) = \sum_y P(x|y)P(y)$$

Law of total probability

- Discrete case:

$$\sum_x P(x) = 1$$

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

$$P(x) = \sum_y P(x|y)P(y)$$

- Continuous case:

$$\int p(x)dx = 1$$

$$p(x) = \int p(x, y)dy$$

$$p(x) = \int p(x|y)p(y)dy$$

Bayes rule

- $p(x|y)p(y) = p(x, y) = p(y|x)p(x)$

Bayes rule

- $p(x|y)p(y) = p(x, y) = p(y|x)p(x)$
- $p(x|y) = \dfrac{p(y|x)p(x)}{p(y)}$

Bayes rule

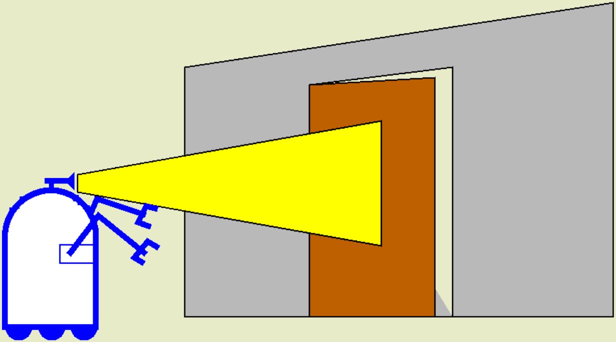
- $p(x|y)p(y) = p(x,y) = p(y|x)p(x)$
- $p(x|y) = \dfrac{p(y|x)p(x)}{p(y)} \overset{const\ y}{\propto} p(y|x)p(x)$

Bayes rule

- $p(x|y)p(y) = p(x,y) = p(y|x)p(x)$
- $p(x|y) = \dfrac{p(y|x)p(x)}{p(y)} \overset{const\ y}{\propto} p(y|x)p(x)$
- Bayes rule with background knowledge:

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

Example for a simple measurement



- The robot obtains the measurement z.
- What is $P(open|z)$?

Diagnostic vs. causal reasoning

- $P(open|z)$ is diagnostic.
- $P(z|open)$ is causal.
- Often the causal knowledge is much easier to obtain (the *sensor models*).

Diagnostic vs. causal reasoning

- $P(open|z)$ is diagnostic.
- $P(z|open)$ is causal.
- Often the causal knowledge is much easier to obtain (the *sensor models*).
- The bayes rule allows us to use causal knowledge to get $P(open|z)$:

$$P(open|z) = \frac{P(z|open)P(open)}{P(z)}$$

Example

- $P(z|open) = 0.6$ $\qquad$ $P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$

Example

- $P(z|open) = 0.6 \qquad P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$
- $P(open|z) = \dfrac{P(z|open)P(open)}{P(z)}$

Example

- $P(z|open) = 0.6 \qquad P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$
- $P(open|z) = \dfrac{P(z|open)P(open)}{P(z)}$
- $P(open|z) = \dfrac{P(z|open)P(open)}{P(z|open)P(open) + P(z|\neg open)P(\neg open)}$

Example

- $P(z|open) = 0.6 \qquad P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$
- $P(open|z) = \dfrac{P(z|open)P(open)}{P(z)}$
- $P(open|z) = \dfrac{P(z|open)P(open)}{P(z|open)P(open) + P(z|\neg open)P(\neg open)}$
- $P(open|z) = \dfrac{0.6 * 0.5}{0.6 * 0.5 + 0.3 * 0.5} = \dfrac{2}{3} \approx 0.67$

Example

- $P(z|open) = 0.6 \qquad P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$
- $P(open|z) = \dfrac{P(z|open)P(open)}{P(z)}$
- $P(open|z) = \dfrac{P(z|open)P(open)}{P(z|open)P(open) + P(z|\neg open)P(\neg open)}$
- $P(open|z) = \dfrac{0.6 * 0.5}{0.6 * 0.5 + 0.3 * 0.5} = \dfrac{2}{3} \approx 0.67$
- The measurement z raises the probability that the door is open.

Actions

- Actions increase uncertainty.

Actions

- Actions increase uncertainty.
- Update belief with action model (e.g. *odometry, motion model*):

$$P(x|u, x')$$

Actions

- Actions increase uncertainty.
- Update belief with action model (e.g. *odometry, motion model*):

$$P(x|u,x')$$

- Outcome of actions:
  - Discrete: $P(x|u) = \sum_{x'} P(x|u,x')P(x')$

Actions

- Actions increase uncertainty.
- Update belief with action model (e.g. *odometry, motion model*):

$$P(x|u, x')$$

- Outcome of actions:
  - Discrete: $P(x|u) = \sum_{x'} P(x|u, x')P(x')$
  - Continuous: $p(x|u) = \int p(x|u, x')p(x')dx'$

Markov assumptions

- Measurement z_t only depends on x_t:

$$p(z_t|x_t, ...) = p(z_t|x_t)$$

Markov assumptions

- Measurement z_t only depends on x_t:

$$p(z_t|x_t, ...) = p(z_t|x_t)$$

- State x_t only depends on x_{t-1} and u_{t-1}:

$$p(x_t|u_{t-1}, x_{t-1}, ...) = p(x_t|u_{t-1}, x_{t-1})$$

Bayes filter

- Given:
  - Measurements $z_1, \ldots, z_t$ and action data/transition measurements $u_1, \ldots, u_t$.
  - Sensor model: $p(z|x)$.
  - Action model: $p(x|u, x')$.
  - Prior probability of the state $p(x)$.
- Wanted:
  - Belief of the state: $Bel(x_t) = p(x_t|z_t, u_{t-1}, \ldots, u_1, z_1)$

Recursive Bayesian estimation

$$Bel(x_t) = p(x_t | z_t, u_{t-1}, z_{t-1}, ...)$$

Recursive Bayesian estimation

$$Bel(x_t) = p(x_t|z_t, u_{t-1}, z_{t-1}, \ldots)$$

$$\text{Bayes} \quad = \frac{p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots)p(x_t|u_{t-1}, z_{t-1}, \ldots)}{p(z_t|u_{t-1}, z_{t-1}, \ldots)}$$

Recursive Bayesian estimation

$$
\begin{aligned}
Bel(x_t) &= p(x_t | z_t, u_{t-1}, z_{t-1}, \ldots) \\
\text{Bayes} \qquad &= \frac{p(z_t | x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t | u_{t-1}, z_{t-1}, \ldots)}{p(z_t | u_{t-1}, z_{t-1}, \ldots)} \\
z_t \text{ const.} \qquad &= \eta p(z_t | x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t | u_{t-1}, z_{t-1}, \ldots)
\end{aligned}
$$

Recursive Bayesian estimation

$$
\begin{aligned}
Bel(x_t) &= p(x_t|z_t, u_{t-1}, z_{t-1}, \dots) \\
\text{Bayes} \quad &= \frac{p(z_t|x_t, u_{t-1}, z_{t-1}, \dots)p(x_t|u_{t-1}, z_{t-1}, \dots)}{p(z_t|u_{t-1}, z_{t-1}, \dots)} \\
z_t \text{ const.} \quad &= \eta p(z_t|x_t, u_{t-1}, z_{t-1}, \dots)p(x_t|u_{t-1}, z_{t-1}, \dots) \\
\text{Markov} \quad &= \eta p(z_t|x_t)p(x_t|u_{t-1}, z_{t-1}, \dots)
\end{aligned}
$$

Recursive Bayesian estimation

$$
\begin{aligned}
Bel(x_t) &= p(x_t|z_t, u_{t-1}, z_{t-1}, \ldots) \\
\text{Bayes} \quad &= \frac{p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots)p(x_t|u_{t-1}, z_{t-1}, \ldots)}{p(z_t|u_{t-1}, z_{t-1}, \ldots)} \\
z_t \text{ const.} \quad &= \eta p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots)p(x_t|u_{t-1}, z_{t-1}, \ldots) \\
\text{Markov} \quad &= \eta p(z_t|x_t)p(x_t|u_{t-1}, z_{t-1}, \ldots) \\
\text{Total prob.} \quad &= \eta p(z_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}, z_{t-1}, \ldots)p(x_{t-1}|u_{t-1}, z_{t-1}, \ldots)dx_{t-1}
\end{aligned}
$$

Recursive Bayesian estimation

$$
\begin{aligned}
Bel(x_t) &= p(x_t|z_t, u_{t-1}, z_{t-1}, \ldots) \\
\text{Bayes} \quad &= \frac{p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots)p(x_t|u_{t-1}, z_{t-1}, \ldots)}{p(z_t|u_{t-1}, z_{t-1}, \ldots)} \\
z_t \text{ const.} \quad &= \eta p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots)p(x_t|u_{t-1}, z_{t-1}, \ldots) \\
\text{Markov} \quad &= \eta p(z_t|x_t)p(x_t|u_{t-1}, z_{t-1}, \ldots) \\
\text{Total prob.} \quad &= \eta p(z_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}, z_{t-1}, \ldots)p(x_{t-1}|u_{t-1}, z_{t-1}, \ldots)dx_{t-1} \\
\text{Markov} \quad &= \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})p(x_{t-1}|z_{t-1}, u_{t-2} \ldots)dx_{t-1}
\end{aligned}
$$

Recursive Bayesian estimation

$$
\begin{aligned}
Bel(x_t) &= p(x_t|z_t, u_{t-1}, z_{t-1}, \ldots) \\
\text{Bayes} \quad &= \frac{p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots)p(x_t|u_{t-1}, z_{t-1}, \ldots)}{p(z_t|u_{t-1}, z_{t-1}, \ldots)} \\
z_t \text{ const.} \quad &= \eta p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots)p(x_t|u_{t-1}, z_{t-1}, \ldots) \\
\text{Markov} \quad &= \eta p(z_t|x_t)p(x_t|u_{t-1}, z_{t-1}, \ldots) \\
\text{Total prob.} \quad &= \eta p(z_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}, z_{t-1}, \ldots)p(x_{t-1}|u_{t-1}, z_{t-1}, \ldots)dx_{t-1} \\
\text{Markov} \quad &= \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})p(x_{t-1}|z_{t-1}, u_{t-2}\ldots)dx_{t-1} \\
&= \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})Bel(x_{t-1})
\end{aligned}
$$

Bayes filter implementations

$$Bel(x_t) = \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1})$$

Bayes filter implementations

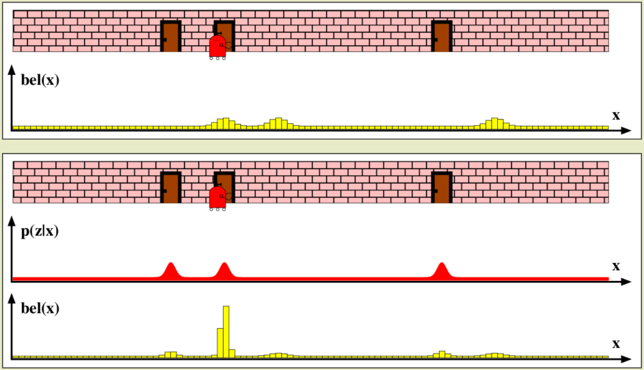
$$Bel(x_t) = \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1})$$

Some methods based on this equation:

- Grid-based estimator
- Kalman filter
- Particle filter

Grid-based estimator

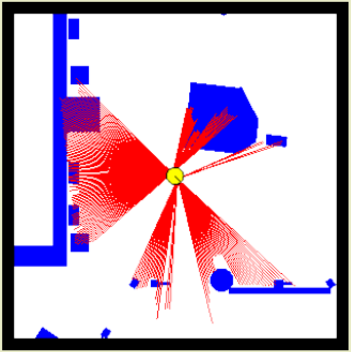
- Probability density function (belief) is represented using a discretized state space.
- Can be a simple grid with a constant step size.

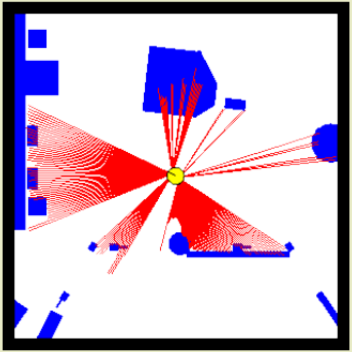


- Tree-based methods using e.g. octrees for more efficiency.

Grid-based estimator

- Can be useful e.g. for localizations using a grid-based environment map.





Kalman filter

- The belief is represented by multivariate normal distributions.
- Very efficient.
- Optimal for linear Gaussian systems.

Kalman filter

- The belief is represented by multivariate normal distributions.
- Very efficient.
- Optimal for linear Gaussian systems.

- Most robotics systems are nonlinear.
- Limited to Gaussian distributions.

Kalman filter

- The belief is represented by multivariate normal distributions.
- Very efficient.
- Optimal for linear Gaussian systems.

- Most robotics systems are nonlinear.
- Limited to Gaussian distributions.

- Extensions of the Kalman Filter for nonlinearity:
  - Extended Kalman Filter
  - Unscented Kalman Filter

Particle filter

- Belief represented by samples (particles).
- State estimation for non-Gaussian, nonlinear systems.

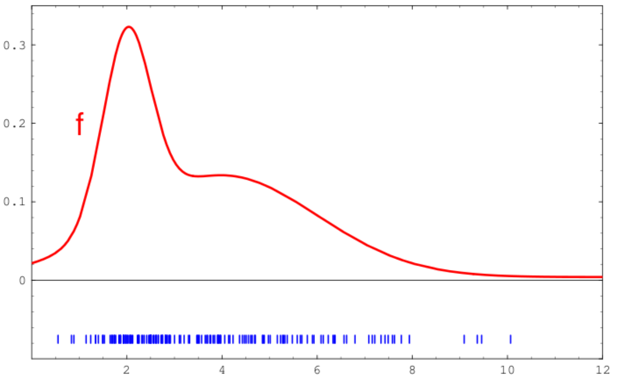
Particle filter

- Belief represented by samples (particles).
- State estimation for non-Gaussian, nonlinear systems.

- Particles have weights.
- A high probability in a given region can be represented by
  - many particles.
  - few particles with higher weights.

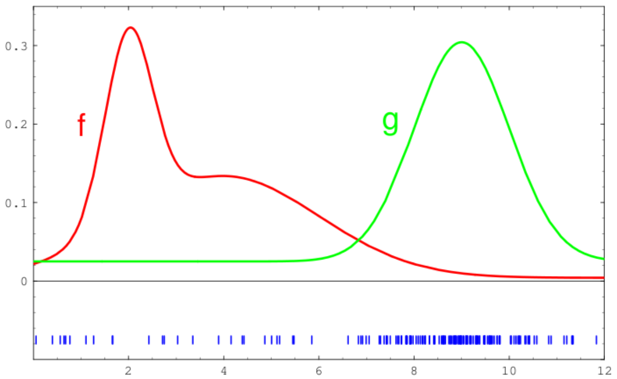
Importance sampling

- Suppose we want to approximate a target density f.



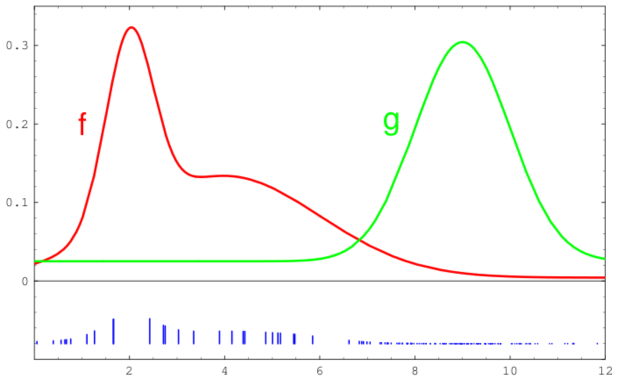
Importance sampling

- Assume we can only draw samples from a density g.



Importance sampling

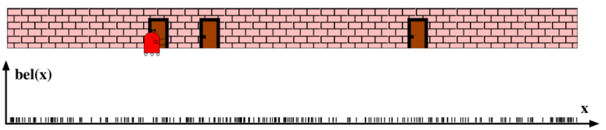
- The target density f can be approximated by attaching the weight $w = f(x)/g(x)$ to each sample x.



Example Monte Carlo localization

Sensor information (importance sampling)

$$Bel(x) \leftarrow \alpha p(z|x) Bel(x)$$

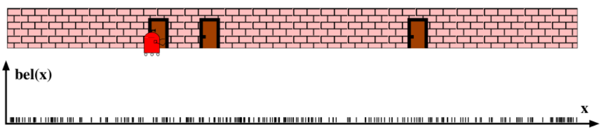


Example Monte Carlo localization

Sensor information (importance sampling)

$$Bel(x) \leftarrow \alpha p(z|x) Bel(x)$$

$$w \leftarrow \frac{\alpha p(z|x) Bel(x)}{Bel(x)} = \alpha p(z|x)$$

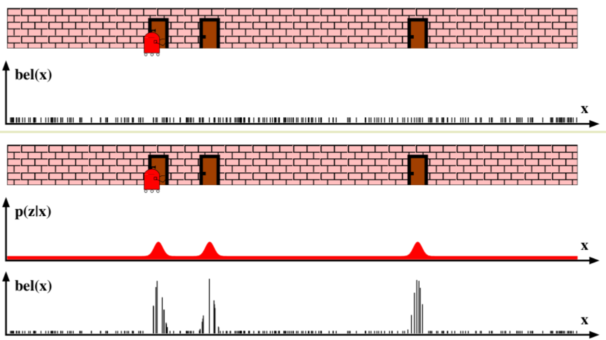


Example Monte Carlo localization

Sensor information (importance sampling)

$$Bel(x) \leftarrow \alpha p(z|x) Bel(x)$$

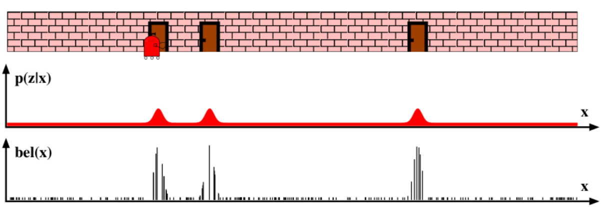
$$w \leftarrow \frac{\alpha p(z|x) Bel(x)}{Bel(x)} = \alpha p(z|x)$$



Example Monte Carlo localization

Robot motion (resampling and prediction)

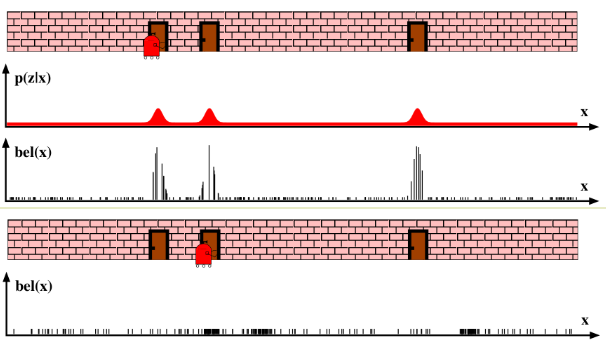
$$Bel(x) \leftarrow \int p(x|u,x')Bel(x')dx'$$



Example Monte Carlo localization

Robot motion (resampling and prediction)

$$Bel(x) \leftarrow \int p(x|u,x')Bel(x')dx'$$

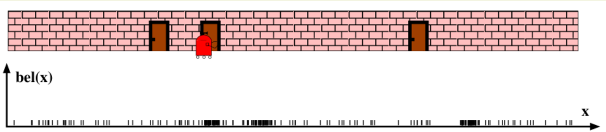


Example Monte Carlo localization

Sensor information (importance sampling):

$$Bel(x) \leftarrow \alpha p(z|x) Bel(x)$$

$$w \leftarrow \frac{\alpha p(z|x) Bel(x)}{Bel(x)} = \alpha p(z|x)$$

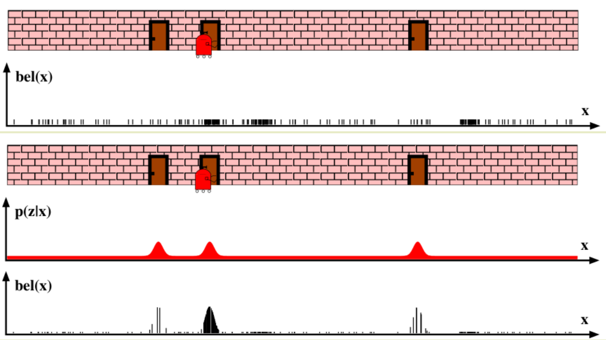


Example Monte Carlo localization

Sensor information (importance sampling):

$$Bel(x) \leftarrow \alpha p(z|x) Bel(x)$$

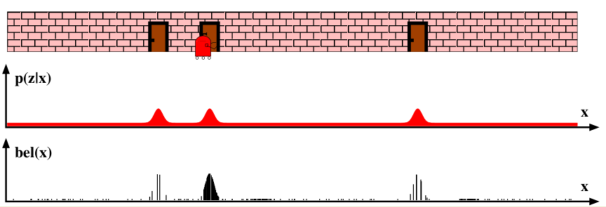
$$w \leftarrow \frac{\alpha p(z|x) Bel(x)}{Bel(x)} = \alpha p(z|x)$$



Example Monte Carlo localization

Robot motion (resampling and prediction):

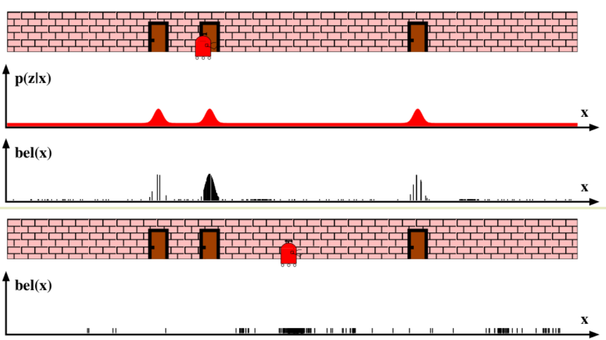
$$Bel(x) \leftarrow \int p(x|u,x')Bel(x')dx'$$



Example Monte Carlo localization

Robot motion (resampling and prediction):

$$Bel(x) \leftarrow \int p(x|u,x')Bel(x')dx'$$



Particle filter steps

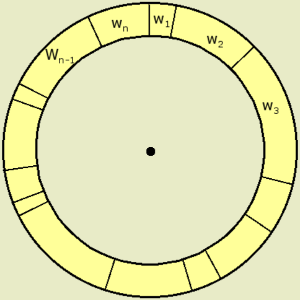
- State transition/prediction: Sample new particles using $p(x|u_{t-1}, x_{t-1})$.
  - In the context of localization: Move particles according to a motion model.
- Sensor update: Set particle weights using the likelihood $p(z|x)$.
- Resampling: Draw new samples from the old particles according to their weights.

Particle filter algorithm

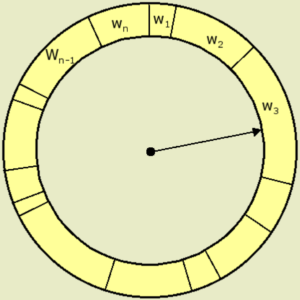
procedure PARTICLE_FILTER(X_{t-1}, u_t, z_t)
    $\bar{X}_t = \emptyset, X_t = \emptyset$
    for $i = 1, \ldots, n$ **do** ▷ Generate new samples
        Sample x_t^i from $p(x_t | x_{t-1}^i, u_t)$
        $w_t^i = p(z_t | x_t^i)$ ▷ Compute importance weight
        $\bar{X}_t = \bar{X}_t + \langle x_t^i, w_t^i \rangle$ ▷ Update and insert normalization factor
    end for
    for $i = 1, \ldots, n$ **do** ▷ Resampling
        draw i with probability $\propto w_t^i$
        add w_t^i to X_t
    end for
end procedure

Resampling

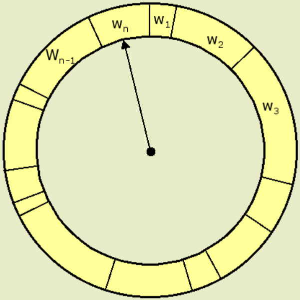
Resampling



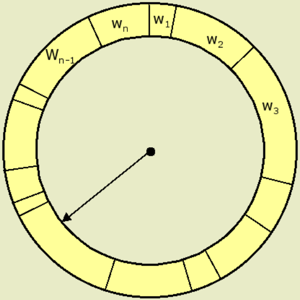
Resampling



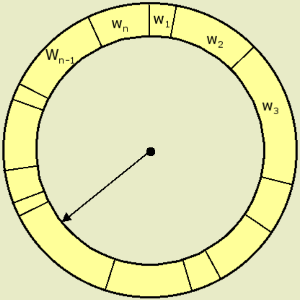
Resampling



Resampling

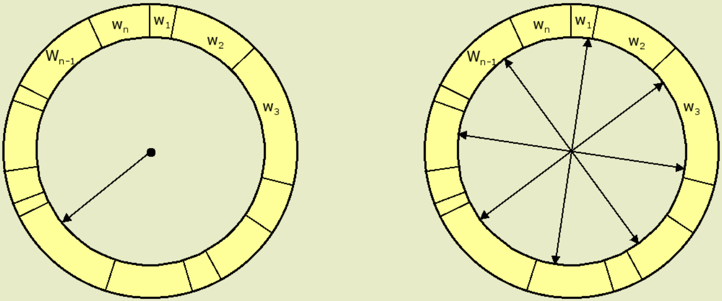


Resampling



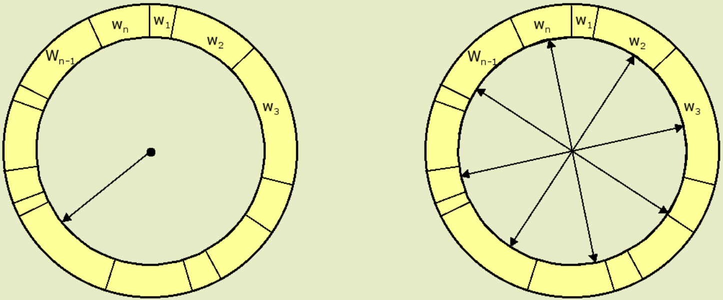
- Binary search, *n log n*
- High variance

Resampling



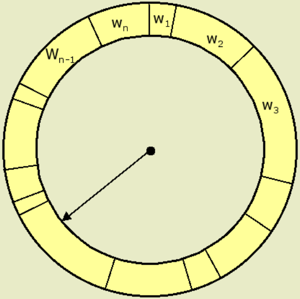
- Binary search, *n log n*
- High variance

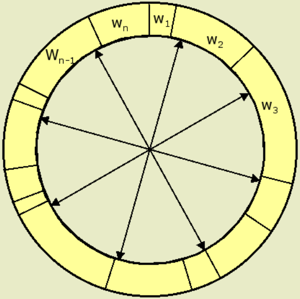
Resampling



- Binary search, *n log n*
- High variance

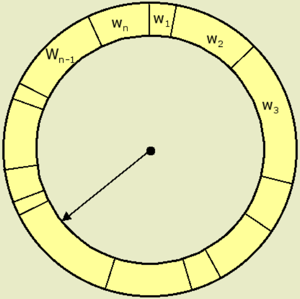
Resampling

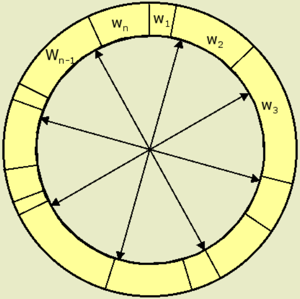




- Binary search, *n log n*
- High variance

Resampling

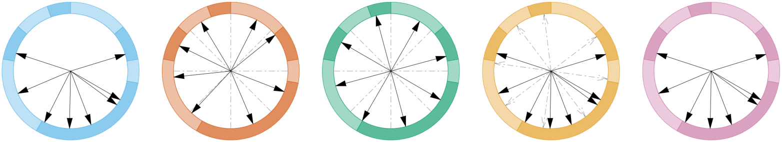




- Binary search, *n log n*
- High variance

Systematic resampling

- Stochastic universal sampling
- Linear time complexity
- Low variance



(a) Multinomial (b) Stratified (c) Systematic (d) Metropolis (e) Rejection

Source: Murray, Lawrence M., Anthony Lee, and Pierre E. Jacob. "Parallel resampling in the particle filter." arXiv preprint arXiv:1301.4019 (2013).

Resampling algorithm

procedure SYSTEMATIC_RESAMPLING(X_t, n)
    $X_t^{'} = \emptyset,\ c_1 = w^1$
    for $i = 2, \ldots, n$ **do** ▷ Generate cdf
        $c_i = c_{i-1} + w^i$
        $u_1 \sim U]0, n^{-1}], i = 1$ ▷ Initialize threshold
    end for
    for $j = 1, \ldots, n$ **do** ▷ Draw samples
        while $u_j > c_i$ **do** ▷ Skip until next threshold reached
            $i = i + 1$
            $S^{'} = S^{'} \bigcup \{\langle x^i, n^{-1} \rangle\}$ ▷ Insert
            $u_{j+1} = u_j + n$ ▷ Increment threshold
        end while
    end for
    Return $X_t^{'}$
end procedure ▷ Also called: **stochastic universal resampling**

Summary

- Particle filters are an implementation of a recursive Bayesian filter.
- Belief is represented by a set of weighted samples.

- Samples can approximate arbitrary probability distributions.
- Works for non-Gaussian, nonlinear systems.
- Relatively easy to implement.

- Depending on the state space a large number of particles might be needed.
- Re-sampling step: new particles are drawn with a probability proportional to the likelihood of the observation.

Problems

- Global localization problem (initial position).
- Robot kidnapping problem.

Problems

- Global localization problem (initial position).
- Robot kidnapping problem.
- Augmented Monte Carlo Localization:
  - Inject new particles when the average weight decreases.
  - New random particles or particles based on current perception.

Acknowledgement

Acknowledgement

The slides for this lecture have been prepared by Andreas Seekircher.