Modeling
CSC752 Autonomous Robotic Systems

Ubbo Visser

Department of Computer Science
University of Miami

October 5, 2022
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 cannot 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

- **Modeling and state estimation**
- **Examples**
- **State estimation**
- **Probabilities**
- **Bayes filter**
- **Particle filter**

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

Example 2

Obstacle avoidance using a laser range finder:

- There can be several different errors in the measurements.

Example 2

Obstacle avoidance using a laser range finder:

- There can be several different errors in the measurements.

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.

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.

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).
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 \((\text{state transition model})\).
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).
Example 4

Self-localization in 1D with limited sensors:

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).
Example 4

Self-localization in 1D with limited sensors:

- Door sensor $\rightarrow$ ambiguous.
- Even a sequence of measurements $z_t$ is not enough to localize.
Example 4

Self-localization in 1D with limited sensors:

- Door sensor $\rightarrow$ 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*).
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

- 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:

Problems: more measurements with every time step → increasing amount of computation.
General state estimation:

- Problems: more measurements with every time step → 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.
The states $x_t$ are hidden.
Recursive state estimation / filter

Recursive state estimation:
Recursive state estimation:

- $X_t$ includes all the knowledge from the measurements before.
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

Belief $X_t$ is updated using only the new measurements → constant time for each step.
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 → 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

\[ \text{bel}(x) \]

\[ \text{p}(z|x) \]

\[ \text{bel}(x) \]
Example state estimation

$p(z|x)$

$bel(x)$
Example state estimation

\[ p(z|x) \]

\[ \text{bel}(x) \]

\[ \text{bel}(x) \]
Example state estimation
Example state estimation
Example state estimation

$p(z|x)$

$\text{bel}(x)$
Example state estimation

\[
p(z|x)
\]

\[
\text{bel}(x)
\]
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 $x, y, \theta$ and speed $x', y', \theta'$
Example 1: Small-Size League

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

- **State**: position $x, y, \theta$ and speed $x', y', \theta'$
- $z_t$: $x, y, \theta$
Example 1: Small-Size League

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

- State: position $x, y, \theta$ and speed $x', y', \theta'$
- $z_t$: $x, y, \theta$
- $u_t$: Driving command sent to the robot.
Example 1: Small-Size League

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

- **State:** position $x, y, \theta$ and speed $x', y', \theta'$
- $z_t$: $x, y, \theta$
- $u_t$: 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 $x, y, \theta$ and speed $x', y', \theta'$
- $z_t$: $x, y, \theta$
- $u_t$: 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

- \( p(x|y)p(y) = p(x, y) = p(y|x)p(x) \)
- \( p(x|y) = \frac{p(y|x)p(x)}{p(y)} \)
Bayes rule

\[ p(x|y)p(y) = p(x, y) = p(y|x)p(x) \]

\[ p(x|y) = \frac{p(y|x)p(x)}{p(y)} \propto p(y|x)p(x) \]
Bayes rule

1. \( p(x|y)p(y) = p(x, y) = p(y|x)p(x) \)

2. \( p(x|y) = \frac{p(y|x)p(x)}{p(y)} \propto p(y|x)p(x) \)

3. 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 $P(\text{open}|z)$?
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

- $P(\text{open} | z)$ is diagnostic.
- $P(z | \text{open})$ 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 $P(\text{open} | z)$:

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

- $P(z|\text{open}) = 0.6 \quad P(z|\neg\text{open}) = 0.3$
- $P(\text{open}) = P(\neg\text{open}) = 0.5$
Example

- \( P(z | \text{open}) = 0.6 \quad P(z | \neg \text{open}) = 0.3 \)
- \( P(\text{open}) = P(\neg \text{open}) = 0.5 \)
- \( P(\text{open} | z) = \frac{P(z | \text{open})P(\text{open})}{P(z)} \)
Example

- $P(z|\text{open}) = 0.6$ \quad $P(z|\neg \text{open}) = 0.3$
- $P(\text{open}) = P(\neg \text{open}) = 0.5$
- $P(\text{open}|z) = \frac{P(z|\text{open})P(\text{open})}{P(z)}$
- $P(\text{open}|z) = \frac{P(z|\text{open})P(\text{open})}{P(z|\text{open})P(\text{open}) + P(z|\neg \text{open})P(\neg \text{open})}$
Example

- \( P(z | open) = 0.6 \) \( P(z | ¬open) = 0.3 \)
- \( P(open) = P(¬open) = 0.5 \)

\[
P(open | z) = \frac{P(z | open)P(open)}{P(z)}
\]

\[
P(open | z) = \frac{P(z | open)P(open)}{P(z | open)P(open) + P(z | ¬open)P(¬open)}
\]

\[
P(open | z) = \frac{0.6 \times 0.5}{0.6 \times 0.5 + 0.3 \times 0.5} = \frac{2}{3} \approx 0.67
\]
Example

- \( P(z|\text{open}) = 0.6 \) \( P(z|\neg\text{open}) = 0.3 \)
- \( P(\text{open}) = P(\neg\text{open}) = 0.5 \)

\[
P(\text{open}|z) = \frac{P(z|\text{open})P(\text{open})}{P(z)}
\]

\[
P(\text{open}|z) = \frac{P(z|\text{open})P(\text{open})}{P(z|\text{open})P(\text{open}) + P(z|\neg\text{open})P(\neg\text{open})}
\]

\[
P(\text{open}|z) = \frac{0.6 \times 0.5}{0.6 \times 0.5 + 0.3 \times 0.5} = \frac{2}{3} \approx 0.67
\]

- 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 $x_t$:

$$p(z_t|x_t, \ldots) = p(z_t|x_t)$$
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 $z_1, ..., z_t$ and action data/transition measurements $u_1, ..., u_t$.
  - Sensor model: $p(z|x)$.
  - Action model: $p(x|u, x')$.
  - Prior probability of the state $p(x)$.

- **Wanted:**
  - Belief of the state: $Bel(x_t) = p(x_t|z_t, u_{t-1}, ..., u_1, z_1)$
Recursive Bayesian estimation

\[ Bel(x_t) = p(x_t | z_t, u_{t-1}, z_{t-1}, ...) \]
Recursive Bayesian estimation

\[ \text{Bel}(x_t) = p(x_t | z_t, u_{t-1}, z_{t-1}, ...) \]

Bayes \[ = \frac{p(z_t | x_t, u_{t-1}, z_{t-1}, ...) p(x_t | u_{t-1}, z_{t-1}, ...)}{p(z_t | u_{t-1}, z_{t-1}, ...)} \]
Recursive Bayesian estimation

\[ \text{Bel}(x_t) = p(x_t | z_t, u_{t-1}, z_{t-1}, ...) \]

Bayes

\[ = \frac{p(z_t | x_t, u_{t-1}, z_{t-1}, ...) p(x_t | u_{t-1}, z_{t-1}, ...)}{p(z_t | u_{t-1}, z_{t-1}, ...)} \]

\[ z_t \text{ const.} = \eta p(z_t | x_t, u_{t-1}, z_{t-1}, ...) p(x_t | u_{t-1}, z_{t-1}, ...) \]
Recursive Bayesian estimation

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

Bayes

\[
\frac{p(z_t | x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t | u_{t-1}, z_{t-1}, \ldots)}{p(z_t | u_{t-1}, z_{t-1}, \ldots)}
\]

\[
z_t \text{ const.} = \eta p(z_t | x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t | u_{t-1}, z_{t-1}, \ldots)
\]

Markov

\[
\eta p(z_t | x_t) p(x_t | u_{t-1}, z_{t-1}, \ldots)
\]
Recursive Bayesian estimation

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

Bayes

\[ = \frac{p(z_t | x_t, u_{t-1}, z_{t-1}, ...) p(x_t | u_{t-1}, z_{t-1}, ...)}{p(z_t | u_{t-1}, z_{t-1}, ...)} \]

\[ = \eta p(z_t | x_t, u_{t-1}, z_{t-1}, ...) p(x_t | u_{t-1}, z_{t-1}, ...) \]

\[ = \eta p(z_t | x_t) p(x_t | u_{t-1}, z_{t-1}, ...) \]

Total prob.

\[ = \eta p(z_t | x_t) \int p(x_t | x_{t-1}, u_{t-1}, z_{t-1}, ...) p(x_{t-1} | u_{t-1}, z_{t-1}, ...) dx_{t-1} \]
Recursive Bayesian estimation

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

Bayes
\[
= \frac{p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t|u_{t-1}, z_{t-1}, \ldots)}{p(z_t|u_{t-1}, z_{t-1}, \ldots)}
\]

\[ z_t \text{ const.} = \eta p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t|u_{t-1}, z_{t-1}, \ldots) \]

Markov
\[ = \eta p(z_t|x_t) p(x_t|u_{t-1}, z_{t-1}, \ldots) \]

Total prob.
\[ = \eta p(z_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}, z_{t-1}, \ldots) p(x_{t-1}|u_{t-1}, z_{t-1}, \ldots) dx_{t-1} \]

Markov
\[ = \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1}) p(x_{t-1}|z_{t-1}, u_{t-2}) dx_{t-1} \]
Recursive Bayesian estimation

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

Bayes

\[
= \frac{p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t|u_{t-1}, z_{t-1}, \ldots)}{p(z_t|u_{t-1}, z_{t-1}, \ldots)}
\]

\[ z_t \text{ const.} = \eta p(z_t|x_t, u_{t-1}, z_{t-1}, \ldots) p(x_t|u_{t-1}, z_{t-1}, \ldots) \]

Markov

\[ = \eta p(z_t|x_t) p(x_t|u_{t-1}, z_{t-1}, \ldots) \]

Total prob.

\[ = \eta p(z_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}, z_{t-1}, \ldots) p(x_{t-1}|u_{t-1}, z_{t-1}, \ldots) dx_{t-1} \]

Markov

\[ = \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1}) p(x_{t-1}|z_{t-1}, u_{t-2}, \ldots) dx_{t-1} \]

\[ = \eta p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1}) \]
Bayes filter implementations

\[ \text{Bel}(x_t) = \eta p(z_t | x_t) \int p(x_t | u_{t-1}, x_{t-1}) \text{Bel}(x_{t-1}) \]
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.
  
  ![](image)
  
  bel(x)

- 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 $f$. 

![Graph showing a target density $f$.]
Importance sampling

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

![Graph with densities $f$ and $g$]
Importance sampling

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

![Graph showing the target density $f$ and proposal distribution $g$.]
Example Monte Carlo localization

Sensor information (importance sampling)

\[ \text{Bel}(x) \leftarrow \alpha p(z|x) \text{Bel}(x) \]
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):

\[ \text{Bel}(x) \leftarrow \alpha p(z|x) \text{Bel}(x) \]

\[ w \leftarrow \frac{\alpha p(z|x) \text{Bel}(x)}{\text{Bel}(x)} = \alpha p(z|x) \]
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

1: **procedure** PARTICLE_FILTER($X_{t-1}, u_t, z_t$)
2:       $\tilde{X}_t = \emptyset, X_t = \emptyset$
3:   **for** $i = 1, \ldots, n$ **do** ▷ Generate new samples
4:          Sample $x^i_t$ from $p(x_t | x^i_{t-1}, u_t)$
5:          $w^i_t = p(z_t | x^i_t)$ ▷ Compute importance weight
6:       $\bar{X}_t = \bar{X}_t + \langle x^i_t, w^i_t \rangle$ ▷ Update and insert normalization factor
7:   **end for**
8:   **for** $i = 1, \ldots, n$ **do** ▷ Resampling
9:          draw $i$ with probability $\propto w^i_t$
10:         add $w^i_t$ to $X_t$
11:   **end for**
12: **end procedure**
Resampling
Resampling

- Binary search, \( n \log n \)
- High variance
- Systematic resampling
- Stochastic universal sampling
- Linear time complexity
- Low variance
Resampling
Resampling

Resampling techniques include:

- Binary search, which has a time complexity of $n \log n$.
- High variance methods.
- Systematic resampling.
- Stochastic universal sampling.
- Low variance methods.
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
Resampling algorithm

1: **procedure** SYSTEMATIC_RESAMPLING($X_t$, $n$)  
2: $X'_t = \emptyset$, $c_1 = w^1$  
3: for $i = 2, \ldots, n$ do
   ▷ Generate cdf
4: $c_i = c_{i-1} + w^i$  
5: $u_1 \sim U[0, n^{-1}]$, $i = 1$  
6: end for  
7: for $j = 1, \ldots, n$ do
   ▷ Draw samples
8: while $u_j > c_i$ do
    ▷ Skip until next threshold reached
9: $i = i + 1$
10: $S' = S' \cup \{\langle x^i, n^{-1} \rangle\}$  
11: $u_{j+1} = u_j + n$  
12: end while
13: end for
14: Return $X'_t$  
15: **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

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