Introduction to Mobile Robotics

Summary

Wolfram Burgard, Diego Tipaldi
Probabilistic Robotics
Probabilistic Robotics

Key idea: Explicit representation of uncertainty

(using the calculus of probability theory)

- Perception  = state estimation
- Action      = utility optimization
Bayes Formula

\[ P(x, y) = P(x \mid y)P(y) = P(y \mid x)P(x) \]

\[ \Rightarrow \]

\[ P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}} \]
Simple Example of State Estimation

- Suppose a robot obtains measurement $z$
- What is $P(open|z)$?
Causal vs. Diagnostic Reasoning

- $P(\text{open} | z)$ is diagnostic.
- $P(z | \text{open})$ is causal.
- Often causal knowledge is easier to obtain.
- Bayes rule allows us to use causal knowledge:

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

(count frequencies!)
Bayes Filters

$$Bel(x_t) = P(x_t | u_1, z_1, \ldots, u_t, z_t)$$

Bayes

$$= \eta P(z_t | x_t, u_1, z_1, \ldots, u_t) P(x_t | u_1, z_1, \ldots, u_t)$$

Markov

$$= \eta P(z_t | x_t) P(x_t | u_1, z_1, \ldots, u_t)$$

Total prob.

$$= \eta P(z_t | x_t) \int P(x_t | u_1, z_1, \ldots, u_t, x_{t-1})$$

$$\quad \quad \quad \quad \quad P(x_{t-1} | u_1, z_1, \ldots, u_t) dx_{t-1}$$

Markov

$$= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) P(x_{t-1} | u_1, z_1, \ldots, u_t) dx_{t-1}$$

Markov

$$= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) P(x_{t-1} | u_1, z_1, \ldots, z_{t-1}) dx_{t-1}$$

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

$z$ = observation

$u$ = action

$x$ = state
Bayes Filters are Familiar!

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

- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- ...

\[ Bel(x_t) = \eta \ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \ Bel(x_{t-1}) \ dx_{t-1} \]
Sensor and Motion Models

\[ P(z \mid x, m) \quad P(x \mid x', u) \]
Motion Models

- Robot motion is inherently uncertain.
- How can we model this uncertainty?
Probabilistic Motion Models

- To implement the Bayes Filter, we need the transition model $p(x | x', u)$.
- The term $p(x | x', u)$ specifies a posterior probability, that action $u$ carries the robot from $x'$ to $x$. 
Typical Motion Models

- In practice, one often finds two types of motion models:
  - Odometry-based
  - Velocity-based (dead reckoning)

- Odometry-based models are used when systems are equipped with wheel encoders.
- Velocity-based models have to be applied when no wheel encoders are given.
- They calculate the new pose based on the velocities and the time elapsed.
**Odometry Model**

- Robot moves from \( \langle x, y, \bar{\theta} \rangle \) to \( \langle x', y', \bar{\theta}' \rangle \).
- Odometry information \( u = \langle \delta_{rot_1}, \delta_{rot_2}, \delta_{trans} \rangle \).

\[
\begin{align*}
\delta_{trans} &= \sqrt{(x' - x)^2 + (y' - y)^2} \\
\delta_{rot_1} &= \text{atan2} \left( y' - y, x' - x \right) - \bar{\theta} \\
\delta_{rot_2} &= \bar{\theta}' - \bar{\theta} - \delta_{rot_1}
\end{align*}
\]
Sensors for Mobile Robots

- **Contact sensors:** Bumpers
- **Internal sensors**
  - Accelerometers (spring-mounted masses)
  - Gyroscopes (spinning mass, laser light)
  - Compasses, inclinometers (earth magnetic field, gravity)
- **Proximity sensors**
  - Sonar (time of flight)
  - Radar (phase and frequency)
  - Laser range-finders (triangulation, tof, phase)
  - Infrared (intensity)
- **Visual sensors:** Cameras
- **Satellite-based sensors:** GPS
Beam-based Sensor Model

- Scan $z$ consists of $K$ measurements.

$$z = \{ z_1, z_2, \ldots, z_K \}$$

- Individual measurements are independent given the robot position.

$$P(z \mid x, m) = \prod_{k=1}^{K} P(z_k \mid x, m)$$
Beam-based Proximity Model

Measurement noise

\[
P_{\text{hit}}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2} \frac{(z - z_{\text{exp}})^2}{b}}
\]

Unexpected obstacles

\[
P_{\text{unexp}}(z \mid x, m) = \begin{cases} 
\eta \lambda e^{-\lambda z} & z < z_{\text{exp}} \\
0 & \text{otherwise}
\end{cases}
\]
Beam-based Proximity Model

Random measurement

\[ P_{\text{rand}} (z \mid x, m) = \eta \frac{1}{z_{\text{max}}} \]

Max range

\[ P_{\text{max}} (z \mid x, m) = \eta \frac{1}{z_{\text{small}}} \]
Resulting Mixture Density

\[
P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \begin{pmatrix} P_{\text{hit}} (z \mid x, m) \\ P_{\text{unexp}} (z \mid x, m) \\ P_{\text{max}} (z \mid x, m) \\ P_{\text{rand}} (z \mid x, m) \end{pmatrix}
\]

How can we determine the model parameters?
Bayes Filter in Robotics
Bayes Filters in Action

- Discrete filters
- Kalman filters
- Particle filters
Discrete Filter

- The belief is typically stored in a histogram / grid representation
- To update the belief upon sensory input and to carry out the normalization one has to iterate over all cells of the grid
Piecewise Constant
Kalman Filter

- Optimal for linear Gaussian systems!
- Most robotics systems are **nonlinear**!
- Polynomial in measurement dimensionality $k$ and state dimensionality $n$:

$$O(k^{2.376} + n^2)$$
Kalman Filter Algorithm

1. Algorithm **Kalman_filter**\(( \mu_{t-1}, \Sigma_{t-1}, u_t, z_t )\):

2. Prediction:
3. \( \overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \)
4. \( \Sigma_t = A_t \Sigma_{t-1} A_t^T + Q_t \)

5. Correction:
6. \( K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + R_t)^{-1} \)
7. \( \mu_t = \mu_{\overline{\mu}_t} + K_t (z_t - C_t \mu_{\overline{\mu}_t}) \)
8. \( \Sigma_t = (I - K_t C_t) \Sigma_t \)
9. Return \( \mu_t, \Sigma_t \)
Extended Kalman Filter

- Approach to handle non-linear models
- Performs a linearization in each step
- Not optimal
- Can diverge if nonlinearities are large!
- Works surprisingly well even when all assumptions are violated!
- Same complexity than the KF
Particle Filter

- Basic principle
  - Set of state hypotheses ("particles")
  - Survival-of-the-fittest

- Particle filters are a way to efficiently represent non-Gaussian distributions
Mathematical Description

- Set of weighted samples

\[ S = \left\{ \left( s[i], w[i] \right) \mid i = 1, \ldots, N \right\} \]

- The samples represent the posterior

\[ p(x) = \sum_{i=1}^{N} w_i \cdot \delta_{s[i]}(x) \]
Particle Filter Algorithm in Brief

- Sample the next generation for particles using the proposal distribution

- Compute the importance weights:
  \[ \text{weight} = \frac{\text{target distribution}}{\text{proposal distribution}} \]

- Resampling: “Replace unlikely samples by more likely ones”
Importance Sampling Principle

- We can even use a different distribution \( g \) to generate samples from \( f \)
- By introducing an importance weight \( w \), we can account for the “differences between \( g \) and \( f \)”
- \( w = \frac{f}{g} \)
- \( f \) is often called target
- \( g \) is often called proposal
- Pre-condition: \( f(x) > 0 \rightarrow g(x) > 0 \)
Particle Filter Algorithm

1. Algorithm `particle_filter( S_{t-1}, u_{t-1} z_t)`: 
2. \( S_t = \emptyset, \quad \eta = 0 \)
3. For \( i = 1 \ldots n \)
   Generate new samples
4. Sample index \( j(i) \) from the discrete distribution given by \( w_{t-1} \)
5. Sample \( x_t^i \) from \( p(x_t | x_{t-1}, u_{t-1}) \) using \( x_{t-1}^{j(i)} \) and \( u_{t-1} \)
6. \( w_t^i = p(z_t | x_t^i) \)
   Compute importance weight
7. \( \eta = \eta + w_t^i \)
   Update normalization factor
8. \( S_t = S_t \cup \{ < x_t^i, w_t^i > \} \)
   Insert
9. For \( i = 1 \ldots n \)
10. \( w_t^i = w_t^i / \eta \)
    Normalize weights
Particle Filter Algorithm

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

- Draw \( x^i_{t-1} \) from \( Bel(x_{t-1}) \)
- Draw \( x^i_t \) from \( p(x_t \mid x^i_{t-1}, u_{t-1}) \)
- Importance factor for \( x^i_t \):

\[ w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta \ p(z_t \mid x_t) \ p(x_t \mid x_{t-1}, u_{t-1}) \ Bel(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1}) \ Bel(x_{t-1})} \propto p(z_t \mid x_t) \]
Resampling

- Roulette wheel
- Binary search, $n \log n$
- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance
MCL Example
Mapping
Why Mapping?

- Learning maps is one of the fundamental problems in mobile robotics
- Maps allow robots to efficiently carry out their tasks, allow localization ...
- Successful robot systems rely on maps for localization, path planning, activity planning etc
Occupancy Grid Maps

- Discretize the world into equally spaced cells
- Each cell stores the probability that the corresponding area is occupied by an obstacle
- The cells are assumed to be conditionally independent
- If the pose of the robot is known, mapping is easy
Updating Occupancy Grid Maps

- Update the map cells using the inverse sensor model

\[
Bel \left( m_t^{[xy]} \right) = 1 - \left( 1 + \frac{P \left( m_t^{[xy]} \mid z_t, u_{t-1} \right)}{1 - P \left( m_t^{[xy]} \mid z_t, u_{t-1} \right)} \cdot \frac{1 - P \left( m_t^{[xy]} \right)}{P \left( m_t^{[xy]} \right)} \cdot \frac{Bel \left( m_{t-1}^{[xy]} \right)}{1 - Bel \left( m_{t-1}^{[xy]} \right)} \right)^{-1}
\]

- Or use the log-odds representation

\[
\overline{B} \left( m_t^{[xy]} \right) = \log \text{odds} \left( m_t^{[xy]} \mid z_t, u_{t-1} \right) - \log \text{odds} \left( m_t^{[xy]} \right) + \overline{B} \left( m_{t-1}^{[xy]} \right)
\]

\[
\overline{B} \left( m_t^{[xy]} \right) = \log \text{odds} \left( m_t^{[xy]} \right)
\]

\[
\text{odds} \left( x \right) = \left( \frac{P \left( x \right)}{1 - P \left( x \right)} \right)
\]
Reflection Probability Maps

- **Value of interest:** $P(\text{reflects}(x,y))$

- **For every cell count**
  - $\text{hits}(x,y)$: number of cases where a beam ended at $<x,y>$
  - $\text{misses}(x,y)$: number of cases where a beam passed through $<x,y>$

\[
\text{Bel} \left( m^{[xy]} \right) = \frac{\text{hits}(x,y)}{\text{hits}(x,y) + \text{misses}(x,y)}
\]
SLAM
The SLAM Problem

A robot is exploring an unknown, static environment.

**Given:**
- The robot’s controls
- Observations of nearby features

**Estimate:**
- Map of features
- Path of the robot
Chicken-and-Egg-Problem

- SLAM is a chicken-and-egg problem
  - A map is needed for localizing a robot
  - A good pose estimate is needed to build a map
- Thus, SLAM is regarded as a hard problem in robotics
- A variety of different approaches to address the SLAM problem have been presented
- Probabilistic methods outperform most other techniques
SLAM: Simultaneous Localization and Mapping

- **Full SLAM:** \( p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \)

  Estimates entire path and map!

- **Online SLAM:**

  \[
  p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \int \cdots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) dx_1 dx_2 \cdots dx_{t-1}
  \]

  Integrations typically done one at a time

  Estimates most recent pose and map!
Why is SLAM a hard problem?

- In the real world, the mapping between observations and landmarks is unknown.
- Picking wrong data associations can have catastrophic consequences.
- Pose error correlates data associations.
(E)KF-SLAM

- Map with N landmarks: \((3+2N)\)-dimensional Gaussian

\[
\text{Bel} \left( x_t, m_t \right) = \begin{pmatrix}
\sigma^2_x & \sigma_{xy} & \sigma_{x\theta} \\
\sigma_{xy} & \sigma^2_y & \sigma_{y\theta} \\
\sigma_{x\theta} & \sigma_{y\theta} & \sigma^2_{\theta}
\end{pmatrix}
\begin{pmatrix}
\sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} \\
\sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} \\
\vdots & \vdots & \vdots \\
\sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N}
\end{pmatrix}
\]

- Can handle hundreds of dimensions
EKF-SLAM

Map

Correlation matrix
EKF-SLAM

Map

Correlation matrix
EKF-SLAM

Map

Correlation matrix
FastSLAM

- Use a particle filter for map learning
- Problem: the map is high-dimensional
- Solution: separate the estimation of the robot’s trajectory from the one of the map of the environment
- This is done by means of a factorization in the SLAM posterior often called Rao-Blackwellization
Rao-Blackwellization

poses | map | observations & movements

\[
p(x_{1:t}, m \mid z_{1:t}, u_{0:t-1}) = \]

\[
p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(m \mid x_{1:t}, z_{1:t})
\]

SLAM posterior

Robot path posterior

Mapping with known poses

Factorization first introduced by Murphy in 1999
Rao-Blackwellized Mapping

- Each particle represents a possible trajectory of the robot.

- Each particle maintains its own map and updates it upon “mapping with known poses”.

- Each particle survives with a probability proportional to the likelihood of the observations relative to its own map.
**FastSLAM**

- Rao-Blackwellized particle filtering based on landmarks
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain $M$ EKFs

<table>
<thead>
<tr>
<th>Particle #1</th>
<th>x, y, θ</th>
<th>Landmark 1</th>
<th>Landmark 2</th>
<th>...</th>
<th>Landmark M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle #2</td>
<td>x, y, θ</td>
<td>Landmark 1</td>
<td>Landmark 2</td>
<td>...</td>
<td>Landmark M</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Particle N</td>
<td>x, y, θ</td>
<td>Landmark 1</td>
<td>Landmark 2</td>
<td>...</td>
<td>Landmark M</td>
</tr>
</tbody>
</table>
Grid-based FastSLAM

- Similar ideas can be used to learn grid maps
- To obtain a practical solution, an efficiently computable, informed proposal distribution is needed
- Idea: in the SLAM posterior, the observation model dominates the motion model (given an accurate sensor)
Proposal Distribution

\[
p(x_t|x_{t-1}^{(i)}, m^{(i)}, z_t, u_t) \sim \frac{p(z_t|x_t, m^{(i)})}{\int_{x_t \in \{x|p(z_t|x, m^{(i)}) > \epsilon\}} p(z_t|x_t, m^{(i)}) \, dx_t}
\]

Approximate this equation by a Gaussian:

maximum reported by a scan matcher

Gaussian approximation

Sampled points around the maximum

Draw next generation of samples
Typical Results
Robot Motion
Robot Motion Planning

Latombe (1991): “... eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world.”

Goals:

- Collision-free trajectories.
- Robot should reach the goal location as fast as possible.
Two Challenges

- Calculate the optimal path taking potential uncertainties in the actions into account

- Quickly generate actions in the case of unforeseen objects
Classic Two-layered Architecture

Planning

Collision Avoidance

sensor data

map

low frequency

sub-goal

high frequency

motion command

robot
Information Gain-based Exploration

- SLAM is typically passive, because it consumes incoming sensor data.
- Exploration actively guides the robot to cover the environment with its sensors.
- Exploration in combination with SLAM: Acting under pose and map uncertainty.
- Uncertainty should/needs to be taken into account when selecting an action.
- Key question: Where to move next?
The mutual information $I$ is given by the reduction of entropy in the belief action to be carried out.

$$I(X, M; Z^a) = \text{uncertainty of the filter} - \text{“uncertainty of the filter after carrying out action } a\text{”}$$
Integrating Over Observations

- Computing the mutual information requires to integrate over potential observations

\[
I(X, M; Z^a) = H(X, M) - H(X, M \mid Z^a)
\]

\[
H(X, M \mid Z^a) = \int_z p(z \mid a)H(X, M \mid Z^a = z) \, dz
\]

potential observation sequences
Integral Approximation

- The particle filter represents a posterior about possible maps

map of particle 1  map of particle 2  map of particle 3
Integral Approximation

- The particle filter represents a posterior about possible maps
- Simulate laser measurements in the maps of the particles

\[ H(X, M \mid Z^a) = \sum_z p(z \mid a) H(X, M \mid Z^a = z) \]

\[ = \sum_i \omega^{[i]} H(X, M \mid Z^a = z^{[i]}_{sim_a}) \]
Summary on Information Gain-based Exploration

- A decision-theoretic approach to exploration in the context of RBPF-SLAM
- The approach utilizes the factorization of the Rao-Blackwellization to efficiently calculate the expected information gain
- Reasons about measurements obtained along the path of the robot
- Considers a reduced action set consisting of exploration, loop-closing, and place-revisiting actions
The Exam is Approaching ...

- This lecture gave a short overview over the most important topics addressed in this course
- For the exam, you need to know at least the basic formulas (e.g., Bayes filter, MCL eqs., Rao-Blackwellization, entropy, ...)

Good luck for the exam!