CSE-490R Robotics

SLAM + Fast-SLAM

(Partial slides borrowed from Dieter Fox, Wolfram Burgard & Cyril Stachniss)
What is SLAM?

- **Localization**: Estimate current pose given map, controls and observations
  \[ p(x_t \mid u_{1:t}, z_{1:t}, m) \]

- **Mapping**: Build map given poses and observations
  \[ p(m \mid x_{1:t}, z_{1:t}) \]

- **Simultaneous Localization And Mapping (SLAM)**: Find poses and map given controls and observations
  \[ p(x_{1:t}, m \mid u_{1:t}, z_{1:t}) \]
SLAM

- A robot moving through an unknown, static environment

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

- **Estimate:**
  - Map of environment
  - Path of the robot
Map representations

- Typical representations are:
  - Feature-based
  - Grid maps (occupancy maps)
  - 3D representations (voxels, surfels, octrees etc)
Why is SLAM hard?

- SLAM is the task of building a map while estimating the pose of the robot relative to this map.

- Chicken-or-egg problem:
  - A map is needed to localize the robot.
  - A pose estimate is needed to build a map.
Why is SLAM hard?

**SLAM**: robot path and map are both *unknown*!

Robot path error correlates errors in the map.

Robot pose uncertainty.
Particle Filters

- Represent belief by random samples
- Sampling Importance Resampling (SIR) principle
  - Draw the new generation of particles
  - Assign an importance weight to each particle
  - Resampling
- Applications are localization, tracking, …
Particle Filter Algorithm

1. Sample the particles from the proposal distribution
   \[ x_t^{[j]} \sim \pi(x_t | \ldots) \]

2. Compute the importance weights
   \[ w_t^{[j]} = \frac{\text{target}(x_t^{[j]})}{\text{proposal}(x_t^{[j]})} \]

1. Resampling: Draw sample \( \hat{i} \) with probability \( w_t^{[\hat{i}]} \) and repeat \( J \) times

Courtesy: C. Stachniss
Particle Filters for SLAM

- Localization: state space is \( <x, y, \theta> \)

- SLAM: state space is \( <x, y, \theta, \text{map}> \)
  - For grid maps: \( <c_{11}, c_{12}, ..., c_{1n}, c_{21}, ..., c_{nm}> \)
  - For feature maps: \( <l_1, l_2, ..., l_m> \)

- Problem: The number of particles needed to represent a posterior grows exponentially with the dimension of the state space!
Dependencies

- Is there a dependency between the dimensions of the state space?
- If so, can we use the dependency to solve the problem more efficiently?

- In the SLAM context
  - The map depends on the poses of the robot.
  - We know how to build a map given the position of the sensor is known.
Can We Exploit Dependencies Between the Different Dimensions of the State Space?

\[ x_{1:t}, m \]
If We Know the Poses of the Robot, Mapping is Easy!

$x_{1:t}, m$

\[ x_{1:t}, m \]
Key Idea

If we use the particle set only to model the robot’s path, each sample is a path hypothesis. For each particle, we can compute an individual map using its path.
Rao-Blackwellization

- Factorization to exploit dependencies between variables:

\[ p(a, b) = p(b \mid a) \ p(a) \]

- If \( p(b \mid a) \) can be computed efficiently, represent only \( p(a) \) with samples and compute \( p(b \mid a) \) for every sample.

Courtesy: C. Stachniss
Rao-Blackwellization for SLAM

- Factorization of the SLAM posterior

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

First introduced for SLAM by Murphy in 1999

Rao-Blackwellization for SLAM

- Factorization of the SLAM posterior

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

First introduced for SLAM by Murphy in 1999

Rao-Blackwellization for SLAM

- Factorization of the SLAM posterior

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

Grid cells are conditionally independent given the poses

First exploited in FastSLAM by Montemerlo et al., 2002
Rao-Blackwellization for SLAM

- Factorization of the SLAM posterior

\[
p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = \\
p(x_{1:t} \mid z_{1:t}, u_{1:t}) \ p(m \mid x_{1:t}, z_{1:t}) \\
p(x_{1:t} \mid z_{1:t}, u_{1:t}) \prod_{i=1}^{N} p(m^i \mid x_{1:t}, z_{1:t})
\]

First exploited in FastSLAM by Montemerlo et al., 2002

Grid cells are conditionally independent given the poses

Courtesy: C. Stachniss
Rao-Blackwellization for SLAM

- Factorization of the SLAM posterior

\[ p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = \]
\[ p(x_{1:t} \mid z_{1:t}, u_{1:t}) \ p(m \mid x_{1:t}, z_{1:t}) \]
\[ p(x_{1:t} \mid z_{1:t}, u_{1:t}) \prod_{i=1}^{N} p(m^i \mid x_{1:t}, z_{1:t}) \]

particle filter for localization  
occupancy grid mapping

First exploited in FastSLAM by Montemerlo et al., 2002

Courtesy: C. Stachniss
Modeling the Robot’s Path

- Sample-based representation for
  \[ p(x_{1:t} \mid z_{1:t}, u_{1:t}) \]

- Each sample is a path hypothesis
  \[ x_1 \quad x_2 \quad x_3 \quad \ldots \]
  - Starting location, typically (0,0,0)
  - Pose hypothesis at time \( t=2 \)

- Past poses of a sample are not revised
- No need to maintain past poses in the sample set

Courtesy: C. Stachniss
FastSLAM

- Proposed by Montemerlo et al. in 2002 (for landmark based SLAM)
- Each particle has a pose and a map

<table>
<thead>
<tr>
<th>Particle</th>
<th>$x, y, \theta$</th>
<th>Occupancy grid map</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FastSLAM – Particle representation

Figure 13.17 Application of the grid-based variant of the FastSLAM algorithm.

Each particle carries its own map and the importance weights of the particles are computed based on the likelihood of the measurements given the particle's own map.

Figure 13.18 Occupancy grid map generated from laser range data and based on pure odometry. All images courtesy of Dirk Hähnel, University of Freiburg.
FastSLAM Algorithm

Algorithm FastSLAM_occupancy_grids($X_{t-1}, u_t, z_t$):

1. \( \tilde{X}_t = X_t = \emptyset \)
2. \( \text{for } k = 1 \text{ to } M \text{ do} \)
3. \( x_t^{[k]} = \text{sample_motion_model}(u_t, x_{t-1}^{[k]}) \)
4. \( w_t^{[k]} = \text{measurement_model_map}(z_t, x_t^{[k]}, m_{t-1}^{[k]}) \)
5. \( m_t^{[k]} = \text{updated_occupancy_grid}(z_t, x_t^{[k]}, m_{t-1}^{[k]}) \)
6. \( \tilde{X}_t = \tilde{X}_t + \langle x_t^{[k]}, m_t^{[k]}, w_t^{[k]} \rangle \)

7. \( \text{endfor} \)
8. \( \text{for } k = 1 \text{ to } M \text{ do} \)
9. \( \text{draw } i \text{ with probability } \propto w_t^{[i]} \)
10. \( \text{add } \langle x_t^{[i]}, m_t^{[i]} \rangle \text{ to } X_t \)
11. \( \text{endfor} \)
12. \( \text{return } X_t \)

Table 13.4

The FastSLAM algorithm for learning occupancy grid maps. To learn this map, as few as 500 particles were used. During the overall process the robot encountered two lags. A map calculated from pure odometry data is shown in Figure 13.18, illustrating the amount of error in the robot's odometry. The importance of using multiple particles becomes evident in Figure 13.20, which visualizes the trajectories of the samples shortly before and after closing a loop. As the left image illustrates, the robot is uncertain about its position relative to the starting position, hence the wide spread of particles at the time of loop closure. However, a few resampling steps after the robot re-enters known terrain suffice to reduce the uncertainty drastically (right image).
Pure odometry
FastSLAM – Best particle

Figure 13.19
Occupancy grid map corresponding to the particle with the highest accumulated importance weight obtained by the algorithm listed in Table 13.4 from the data depicted in Figure 13.18. The number of particles to create this experiment was 500. Also depicted in this image is the path represented by the particle with the maximum accumulated importance weight.

Figure 13.20
Trajectories of all samples shortly before (left) and after (right) closing the outer loop of the environment depicted in Figure 13.19. Images courtesy of Dirk Hähnel, University of Freiburg.
Weakness of FastSLAM 1.0

- Proposal Distribution
- Importance weighting
FastSLAM 1.0 to FastSLAM 2.0

- FastSLAM 1.0 uses the motion model as the proposal distribution
  \[ x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t) \]

- FastSLAM 2.0 considers also the measurements during sampling

- Especially useful if an accurate sensor is used (compared to the motion noise)

[Montemerlo et al., 2003]
FastSLAM 2.0 (Informally)

- FastSLAM 2.0 samples from
  \[ x_t^{[k]} \sim p(x_t \mid x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}) \]

- Results in a more peaked proposal distribution
- Less particles are required
- More robust and accurate
- But more complex...

[Montemerlo et al., 2003]  
Courtesy: C. Stachniss
Generating better proposals

- Use scan-matching to compute highly accurate odometry measurements from consecutive range scans.
- Use the improved odometry in the prediction step to get highly accurate proposal distributions.
Motion Model for Scan Matching

Raw Odometry
Scan Matching
Rao-Blackwellized Mapping with Scan-Matching
Loop Closure

- Loop closure involves
  - Recognizing when the robot has returned to a previously mapped region
  - Using this information to reduce the uncertainty in the map estimate

- Without loop closure, the uncertainty can grow without bounds
Loop Closure in FastSLAM

- Each particle has its own map
- Maps which agree to closing the loop are weighed higher than others
- These maps are more likely to be resampled

Key: Need diversity of paths/particles/maps
Loop Closure Example

map of particle 1

3 particles

map of particle 2

map of particle 3
Rao-Blackwellized Mapping with Scan-Matching
Rao-Blackwellized Mapping with Scan-Matching
Example (Intel Lab)

- 15 particles
- Four times faster than real-time P4, 2.8GHz
- 5cm resolution during scan matching
- 1cm resolution in final map

Work by Grisetti et al.
Outdoor Campus Map

- 30 particles
- 250x250m²
- 1.088 miles (odometry)
- 20cm resolution during scan matching
- 30cm resolution in final map

Work by Grisetti et al.
FastSLAM for feature-based maps
FastSLAM in Action
FastSLAM – Video – All Maps
Results – Victoria Park

- 4 km traverse
- < 2.5 m RMS position error
- 100 particles

Blue = GPS
Yellow = FastSLAM

Courtesy: M. Montemerlo
Results – Victoria Park (Video)

Courtesy: M. Montemerlo
Results (Sample Size)

Accuracy of FastSLAM vs. the EKF on Simulated Data

RMS Pose Error (meters)

Number of Particles

Courtesy: M. Montemerlo
Results (Motion Uncertainty)

Comparison of FastSLAM and EKF Given Motion Ambiguity

![Graph showing comparison of FastSLAM and EKF in motion uncertainty]

Courtesy: M. Montemerlo
FastSLAM Problems

- How to determine the sample size?
- Particle deprivation, especially when closing (multiple) loops
DP-SLAM: High-Res Fast-SLAM via History Sharing

Run at real-time speed on 2.4GHz Pentium 4 at 10cm/s
Consistency
Results obtained with DP-SLAM 2.0 (offline)
Close up

End courtesy of Eliazar & Parr
FastSLAM Summary

- Particle filter-based SLAM
- Rao-Blackwellization: model the robot’s path by sampling and compute the landmarks given the poses
- Allow for per-particle data association
- FastSLAM 1.0 and 2.0 differ in the proposal distribution
- Complexity $\mathcal{O}(N \log M)$
Literature

FastSLAM

- Thrun et al.: “Probabilistic Robotics”, Chapter 13.1-13.3 + 13.8 (see errata!)
RGBD SLAM
Resulting Map
Experiments: Overlay 1
Experiments: Overlay 2