PacSLAM

Arun Kumar Byravan, Tanner Schmidt, Ernie Wang
SLAM
Simultaneous Localization and Mapping

Acting in an unknown/known environment, create/update a map of the environment, while localizing on the same map

Estimate: \( p(x_{1:t}, m | z_{1:t}, u_{1:t-1}) \)

Chicken and egg problem

Three techniques to accurately solve SLAM:
- FastSlam
- DPSlam
- GridSlam
Simultaneous Localization and Mapping (SLAM) is often solved using particle filter approaches:

- Given particles $S = \{s_1, \ldots, s_m\}$, resample new states $S' = \{s_1', \ldots, s_m'\}$

- Update the position of each particle $s_m$ using the motion model, $P(s'' | s')$

- Assign a weight to each particle based on the probability of the observation, $P(o | s'')$

- Normalize the weights over all particles
FastSLAM

The SLAM problem:

\[
p(s_t, \Theta_n | s_{t-1}, z_t, u_t) = p(s_t | s_{t-1}, z_t, u_t)p(\Theta_n | s_t, z_t, u_t)
\]

- \(s_t\): pose of the robot
- \(u_t\): motion control model
- \(\Theta_n\): positions of the landmarks
- \(z_n\): measured positions of the landmarks

Bayes Net

SLAM posterior factors:
FastSLAM (contd.)

- Particle Filter
  - A particle: \((w; s; \Theta_1…\Theta_n)\)
  - Estimate robot pose \(s\) with probabilistic motion model, \(p(s_t|s_{t-1}, z_t, u_t)\)
  - Estimate parameters \(\Theta_i\) of landmarks using Extended Kalman Filter (EKF), \(p(\Theta_n|s_t, z_t, u_t)\)
  - Weight the particles
  - Resample
- Drawback: data association problems when updating \(\Theta\)
Stores only a single physical map in memory

The map stores a balanced tree at each grid location, keyed by unique unique particle ID numbers

Particles update their own maps by associating data with their particle ID

Particles are stored in a particle ancestry tree

Particles only update grid squares if it causes the value to differ from the parent
DP-SLAM
Distributed Particle SLAM
GridSLAM

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

Split problem into two sub-problems:

1) Estimate robot's trajectory given observations & controls
   \( p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \)

2) Estimate map given robot's trajectory and observations
   \( p(m \mid x_{1:t}, z_{1:t}) \)

Use particle filter to estimate trajectory - each particle has a potential trajectory

Estimate map for each particle - "N" maps if "N" particles
GridSLAM

Two key techniques
1) Accurate sampling of particles:

2) Adaptive resampling:
   Sample only when dispersion of weights is high
GridSLAM - Implementation

1) Initial guess $x_t = x_{t-1} \oplus u_{t-1}$ for robot’s position from odometry
   Use encoders, IMU & Gyro

2) Scan matching: Find best pose $x_t$ matching observation $z_t$ to particle's map "m_t-1"
   Use Laser scan data & map of robot

3) Sample points around best pose from scan matching

4) Compute target distribution and sample new pose

5) Update weights, map

6) Resample if needed
Challenges & Future work

Challenges:

1) Getting SLAM to work in the pacman framework
2) Discrete nature of the pacman framework

Future work:

1) Get gridSLAM working accurately on real data & pacman world
2) Implement DP-SLAM 2.0 with probabilistic occupancy
References


