8

Grid and Monte Carlo Localization

8.1) Introduction

⇒ This chapter describes two localization algorithms that are capable of solving global localization problems.

⇒ These algorithms possess a number of differences to Gaussian techniques:

- They can process raw sensor measurements.
  - There is no need to extract features from sensor values.
  - As a direct implication, they can also process negative information.
- They are not bound to a uni-modal distribution as was the case with the EKF localizer.
- They can solve global localization and in some instances - Kidnapped robot problems.

Grid localization

Monte Carlo Localization {Most Popular}

8.2) Grid Localization

8.2.1) Basic Algorithm

⇒ Grid localization approximates the posterior using a histogram filter over a grid decomposition of the pose space.

⇒ It maintains as posterior a collection of discrete probability values

$$bel(x_t) = \{P_{K,t}\}$$

Where each probability $P_{K,t}$ is defined over a grid cell X_K.

$$range(X_t) = X_{1,t} \cup X_{2,t} \cup \cdots X_{K,t}.$$

⇒ In the most basic versions of grid localization, the partitioning of the space of all poses is time-invariant, and each grid cell is of the same size.

Common Granularity Used

{for indoor environment}

15 cm for x and y dimention

5° for rotational dimention.

Input: $\{P_{K,t-1}\}, U_t, Z_t, m$

Output: $\{P_{K,t}\}$

Scanned by CamScanner

Algorithm Grid localization $(\{p_{k,t-1}\}, u_t, z_t, m)$:

for all k do

$$\bar{p}_{k,t} = \sum_i p_{i,t-1} \text{ motion\_model}(\text{mean}(x_k), u_t, \text{mean}(x_i))$$

$$p_{k,t} = \eta \text{ measurement\_model}(z_t, \text{mean}(x_k), m)$$

endfor

return $\{p_{k,t}\}$

8.2.2) Grid Resolutions

- Coarse, Variable resolution grid
- Fine fixed resolution grid

8.2.3) Computational Considerations

⇒ There exist a number of techniques to reduce the computational complexity of grid localization.

1. Pre-caching
2. Sensor subsampling
3. Delayed motion update
4. Selective updating.

8.3> Monte Carlo Localization

8.3.1> The MCL Algorithm

- Represents belief $bel(x_t)$ by particles.
- Applicable for both local and global localization Problem.
- One of the most popular localization algorithm in robotics.

Algorithm MCL $(\mathcal{X}_{t-1}, u_t, z_t, m)$:

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  $x_t^{[m]} =$ sample_motion_model $(u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} =$ measurement_model $(z_t, x_t^{[m]}, m)$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

end for

for $m = 1$ to M do

  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

end for

return $\mathcal{X}_t$.

8.3.2> Properties of MCL

- MCL can approximate almost any distribution of practical importance.

→ The accuracy of the approximation is easily determined by the size of the particle set M.

→ A common strategy for setting M is to keep sampling until the next pair u_t and z_t has arrived.

↳ Such a resource - adaptive is difficult to achieve for grid localization and Gaussian techniques.

8.3.3> Random Particle MCL: Recovery from Failures

→ MCL, in its present form, solves the global localization problem but cannot recover from robot kidnapping or global localization failures.

→ Fortunately, this problem can be solved by a rather simple heuristic.

↳ The idea of this heuristic is to add random particles to the particle set.

→ This random ~~states~~ particles add an additional level of robustness even when robot is not kidnapped.

→ This approach of adding particles raises two questions:

1. How many particles should be added at each iteration of the algorithm.
2. From which distribution should we generate these particles.

→ One might add a fixed number of random particles at each iteration; but a better idea is to add particles based on some estimate of the localization accuracy.

→ One way to implement this idea is to monitor the probability of sensor measurements and relate it to the average measurement probability.

→ In particle filter, an approximation to this quantity is easily obtained by the importance factor.

$$p(z_t \mid z_{t-1}, u_t, m) \approx \frac{1}{M} \sum_{m=1}^{M} w_t^{[m]}$$

⇒ It is usually good idea to smooth this estimation by averaging it over multiple time steps.

⇒ The second problem of determining which sample distribution to use, can be addressed in two ways:

1. Draw particle according to a uniform distribution over the pose space, and then weigh them with the current observation.

2. Generate particles directly in accordance to the measurement distribution.

Algorithm Augmented_MCL $(\mathcal{X}_{t-1}, u_t, z_t, m)$:

static w_{slow}, w_{fast}

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  $x_t^{[m]}$ = sample_motion_model $(u_t, x_{t-1}^{[m]})$

  $w_t^{[m]}$ = measurement_model $(z_t, x_t^{[m]}, m)$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

  $w_{avg} = w_{avg} + \frac{1}{M} w_t^{[m]}$

endfor

$w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$

$w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$

for $m = 1$ to M do

  with probability $\max(0.0, 1.0 - w_{fast}/w_{slow})$ do

    add random pose to $\mathcal{X}_t$

  else

    draw $i \in \{1, \dots N\}$ with probability $\propto w_t^{[i]}$

    add $x_t^{[i]}$ to $\mathcal{X}_t$

  endwith

endfor

return $\mathcal{X}_t$

$\Rightarrow$ The algorithm requires that $0 \leq \alpha_{slow} \ll \alpha_{fast}$.

$\Rightarrow$ The parameters α_{slow} and α_{fast}, are decay rates for the exponential filters that estimate the long-term and short-term, averages respectively.

8.3.4 Modifying the Proposal Distribution

⇒ A related limitation of MCL arises from its Proposal mechanism.

→ Particle filter uses motion model as Proposal distribution.

→ But it seeks to approximate a Product of this distribution and the perceptual likelihood.

→ The larger the difference between the Proposal and the target distribution, the more samples are needed.

⇒ Luckily, a simple trick provides remedy.

→ Simply use measurement model that artificially inflates the amount of noise in the sensor.

⇒ An alternative, more sound solution involves a modification of the sampling process which we already discussed.

→ Idea is that a small fraction of particles, the role of the motion model and the measurement model are reversed.

→ Particles are generated according to the measurement model.

→ This is called <u>MCL with mixture distribution</u> or mixture MCL

8.4) Localization in Dynamic Environment

=> There exists two fundamental techniques for dealing with dynamic environments:

① Including hidden state into the state estimated by the filter.

② Preprocess sensor measurements to eliminate measurements affected by hidden state.

=> A key property of our rejection mechanism is that it tends to filter out measurements that are surprisingly short, but leaves others in place that are surprisingly long.

↳ This asymmetry reflects the facts that people's presence tends to cause shorter than expected measurements.

8.5> Comparison of different implementations of Markov localization

	EKF	MHT	Coarse (Topo-logical) Grid	fine (metric) Grid	MCL
Measurement	landmark	landmark	landmark	raw measurement	raw measurement
Measurement noise	Gaussian	Gaussian	any	any	any
Posterior	Gaussian	mixture of Gaussian	histogram	histogram	Particle
Efficiency (Memory)	++	++	+	—	+
Efficiency (Time)	++	+	+	—	+
Ease of implementation	+	—	+	—	++
Resolution	++	++	—	+	+
Robustness	—	+	+	++	++
Global Localization	no	no	yes	yes	yes.