HW2 Report

Jinxi Xiao
2021533005
ShanghaiTech University

Abstract

This report contains my solutions to the given problems as well as some of my own understandings. In addition, the sections listed in this report does not one-to-one correspond to the problems, but they do overall cover the whole problem set.[1]

1. Proof for Approximation

Notice that $w_t^{[1]}, ..., w_t^{[M]}$ are i.i.d. random variables with a common mean μ , and we sample $w_t^{[m]}$ by

$$w_t^{[m]} \sim P(z_t|x_t^{[m]}) \tag{1}$$

where $E(x_t^{[m]}) = x_t$, and x_t is the random variable indicates the position of a certain particle at time t, conditioned on past actions and observations $z_{1:t-1}, u_{1:t}$. As a result, we can conclude that $\mathbb{P}(x_t) = P(x_t|z_{1:t-1}, u_{1:t})$.

According to the strong law of large numbers, we have

$$\lim_{M\to+\infty} \frac{1}{M} \sum_{m=1}^{M} w_t^{[m]} = \mu = E[P(z_t|x_t)]. \tag{2}$$

we can further find that

$$\begin{aligned} E[P(z_t|x_t)] &= \int_{x_t} P(z_t|x_t)P(x_t|z_{1:t-1}, u_{1:t})dx_t \\ &= \int_{x_t} P(z_t|x_t, z_{1:t-1}, u_{1:t})P(x_t|z_{1:t-1}, u_{1:t})dx_t \\ &= \int_{x_t} P(z_t, x_t|z_{1:t-1}, u_{1:t})dx_t \\ &= P(z_t|z_{1:t-1}, u_{1:t}) \end{aligned} \tag{3}$$

2. Monte Carlo Localization

2.1. Terms and Configurations

- In this 2D environment, we use a state $s = [x, y, \theta]^T$ to represent the robot or a particle's pose.
- The image has a resolution of 0.05(m/pixel) and an offset of 40 pixels on the y-axis.
- The 2D LiDAR has a minimum distance $z_{min} = 0.15$(m) and a maximum distance $z_{max} = 12.0$(m). The FOV is 360 degrees and the resolution is one degree.
- Laser will return a measurement of range as $\inf$ for failure. To maintain numerical stable, I replace all infinity values with z_{max}. As a result, ranges with values less than z_{min} or greater equal to z_{max} will be considered as failure.
- **Number of Particles:** $M = 1000$
- **Number of Rays:** $K = 360$
- **Occupancy Threshold:** 170
- **Minimal Odometer Distance:** 0.2(m)
- **Minimal Odometer Angle:** 20°
- **Rotation to Rotation Noise:** $\alpha_1 = 0.01$
- **Translation to Rotation Noise:** $\alpha_4 = 0.01$
- **Translation to Translation Noise:** $\alpha_3 = 0.01$
- **Rotation to Translation Noise:** $\alpha_2 = 0.01$
- **Probability Coefficients of Ray Casting Sensor Model:**
  - $\mathbf{z_{hit}}$: 6.0
  - $\mathbf{z_{short}}$: 0.5
  - $\mathbf{z_{max}}$: 0.5
  - $\mathbf{z_{rand}}$: 3.0
  - $\sigma_{\mathbf{hit}}$: 1.0
  - $\lambda_{\mathbf{short}}$: 0.5

[1] Codes are stored at here

2.2. Change of Basis

There are in fact four frames: Image frame I, map frame $\mathcal{M}$, robot frame R and Laser frame L. In fact, We know only two transformations between them.

1. We can obtain the **ranges**, which are the distance measurement from the Laser scan in the bag file, and we represent it by
$$[d_1,\ ...,\ d_{360}]$$
notice that the resolution of the Laser sensor is one degree. Thus, d_1 corresponds to **angle_min** and d_{360} corresponds to **angle_max**. As a result, given a measured distance l and the corresponding angle ϕ, we can find its coordinate in L as $[l\cos(\phi),\ l\sin(\phi)]$.

2. Since we already have the transformation between R and L, we can transform a Laser point to R as $[-l\cos(\phi)+0.2,\ l\sin(\phi)]$.

3. The most important thing is that all poses in **/odom** topic are measured based on $\mathcal{M}$. Thus, give a pose of the 2D robot as $T \in SE(2)$ along with the Laser point $S = [x,\ y,\ 1]$, we can find the Laser point in $\mathcal{M}$ as TS(matrix multiplication).

4. We do know the transformation from $\mathcal{M}$ to I, given a coordinate in R as $[x, y]$, we can find the pixel coordinate as $[\text{int}(20x),\ \text{int}(20y)+40]$

2.3. The Main Process

The general pipeline can be found at the reference book [1] page 252. And I have made some small modifications, which is shown in Algorithm 1.

2.3.1 Select key frames

It would wise to select some key frames when applying MCL, this is due to fact that MCL is both time-consuming and memory-consuming. Thresholds that I use can be found in Section 2.1. And I would like to talk a little bit more about the alignment of odometry and Laser scans.

The fact is that the sampling frequencies of odometry and scans are different. But a good thing is that the timestamp for every laser scan is adjacent to two successive timestamps of odometry. As a result, for the triggered laser scan, I will use the average of the two adjacent odometry as the corresponding calculated odometry.

2.3.2 Motion Model

The motion model that I use can be found in book [1] page 136, and I use Gaussian distribution to sample noise. And notice that since we are using the relative poses, $\theta = \bar{\theta} = 0$.

Algorithm sample_motion_model_odometry(u_t, x_{t-1})**:**

$\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
$\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$
$\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

$\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \textbf{sample}(\alpha_1\delta^2_{\text{rot1}} + \alpha_2\delta^2_{\text{trans}})$
$\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \textbf{sample}(\alpha_3\delta^2_{\text{trans}} + \alpha_4\ \delta^2_{\text{rot1}} + \alpha_4\ \delta^2_{\text{rot2}})$
$\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \textbf{sample}(\alpha_1\delta^2_{\text{rot2}} + \alpha_2\delta^2_{\text{trans}})$

$x' = x + \hat{\delta}_{\text{trans}}\ \cos(\theta + \hat{\delta}_{\text{rot1}})$
$y' = y + \hat{\delta}_{\text{trans}}\ \sin(\theta + \hat{\delta}_{\text{rot1}})$
$\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$

return $x_t = (x', y', \theta')^T$

Figure 1. Motion model

2.3.3 Sensor Model

Algorithm that I use can be found in book [1] page 158:

Algorithm beam_range_finder_model(z_t, x_t, m)**:**

$q = 1$
for $k = 1$ *to* K *do*
  compute z_t^{k*} *for the measurement* z_t^k *using ray casting*
  $p = z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k \mid x_t, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t, m)$
  $\quad + z_{\text{max}} \cdot p_{\text{max}}(z_t^k \mid x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t, m)$
  $q = q \cdot p$
return q

Figure 2. Sensor model

$$P_{hit}\left(z_t^k|x_t,\mathcal{M}\right) = \begin{cases}\eta_1\mathcal{N}\left(z_t^k; z_t^{k*}, \sigma^2_{hit}\right), & z_{min} \le z_t^k \le z_{max}\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

where

$$\mathcal{N}\left(z_t^k; z_t^{k*}, \sigma^2_{hit}\right) = \frac{1}{\sqrt{2\pi\sigma^2_{hit}}} e^{-\frac{1}{2}\frac{\left(z_t^k - z_t^{k*}\right)^2}{\sigma^2_{hit}}}$$
$$\eta_1 = \left(\int_{z_{min}}^{z_{max}} \mathcal{N}\left(z_t^k; z_t^{k*}, \sigma^2_{hit}\right) dz_t^k\right)^{-1}$$

$$P_{short}\left(z_t^t|x_t,\mathcal{M}\right) = \begin{cases}\eta_2\lambda_{short}e^{-\lambda_{short}z_t^k}, & z_{min} \le z_t^k \le z_t^{k*}\\ 0 & \text{otherwise}\end{cases} \tag{5}$$

where

$$\eta_2 = \left(-e^{-\lambda_{short}z_t^{k*}} + e^{-\lambda_{short}z_{min}}\right)^{-1}$$

$$P_{max}\left(z_t^t|x_t,\mathcal{M}\right) = \begin{cases}1, & z_t^k \ge z_{max} \text{ or } z_t^k < z_{min}\\ 0 & \text{otherwise}\end{cases} \tag{6}$$

Algorithm 1 Monte Carlo Localization (MCL) Algorithm

```
Initialize configurations and load grid map
Initialize a list of Robot objects particles
Initialize last_triggered_state and prev_state
Open ROS bag file
for each message in bag file do
    if connection topic is "/scan" then
        Update scan_ranges with message data
        continue
    end if
    Update current_state with message data
    if excess threshold check passes then
        Calculate motion between last_triggered_state and prev_state
        Update each particle's state
        Calculate weights for each particle
        Resample particles based on weights
        Update last_triggered_state
        Visualize the result
    end if
end for
```

$$P_{rand}\left(z_t^t|x_t,\mathcal{M}\right)=\begin{cases}\frac{1}{z_{max}-z_{min}}, & z_{min}\leq z_t^k<z_{max}\\ 0 & \text{otherwise}\end{cases} \tag{7}$$

$$P\left(z_t^t|x_t,\mathcal{M}\right)=\begin{pmatrix}z_{hit}\\ z_{short}\\ z_{max}\\ z_{rand}\end{pmatrix}^T\begin{pmatrix}P_{hit}\left(z_t^t|x_t,\mathcal{M}\right)\\ P_{short}\left(z_t^t|x_t,\mathcal{M}\right)\\ P_{max}\left(z_t^t|x_t,\mathcal{M}\right)\\ P_{rand}\left(z_t^t|x_t,\mathcal{M}\right)\end{pmatrix} \tag{8}$$

where

$$z_{hit}+z_{short}+z_{max}+z_{rand}=1$$

Given all the equations and the map $\mathcal{M}$, we can find the weight for each particle. But there would like to be big problem, that is the numerical unstableness and giant range of weights.

Suppose that the probability of one beam is 0.2, then the weight would be $0.2^K\approx 10^{-63}$; and the weight of another particle is $0.8^K\approx 10^{-9}$. As a result, I have got to scale some factor to z_{hit}... to make the result of product stable. And in practice I set the scale to be 10. However, despite that the values are stable, the relative range does not change, some particles with small values will never be sampled, if we resample particles proportionally to their weights.

2.3.4 Resample

We make full use of the *Low_variance_sampling* algorithm in book [1] page 110. The main idea is to sample new particles proportional to their weights. And in practice, I adopt a more efficient implementation in GitHub [2].

2.4. Results

I have tried to parallelize my codes with *Numpy* and multiprocessing. The time consumption of takes about 100 seconds, including the main process as well as visualizing. During the process, I have found out that it would be extremely important to set the hyperparameters (and of course mine are not the optimal ones), as a result, I feel that the method based on Ray Casting has a significant demerit. For more visual results please visit the github repo.

3. KLD-Sampling

The general algorithm can be found in book [1] page 264. In practice, the minimum number of particles are 100 and the maximum number of particles is 1000. By using the KLD sampling method, we can observe a much smaller time consumption, which is around 50s. For more visual results please visit the github repo.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[2] "Github:robond-mcl-lab." `https://github.com/udacity/RoboND-MCL-Lab`.