SLAM Homework 2:
Monte Carlo Localization

Prof. Laurent Kneip

March 31th 2024

Instructions:

1. Deadline: **2024-4-16 23:59:59**
2. No handwritten homework is accepted.
3. Your homework should be submitted in PDF format and packed with your code, and the naming format of the file is *studentID-name-hw2*.
4. Please submit your homework through email to *daizj2022@shanghaitech.edu.cn* with the subject line "studentID-name-hw2"

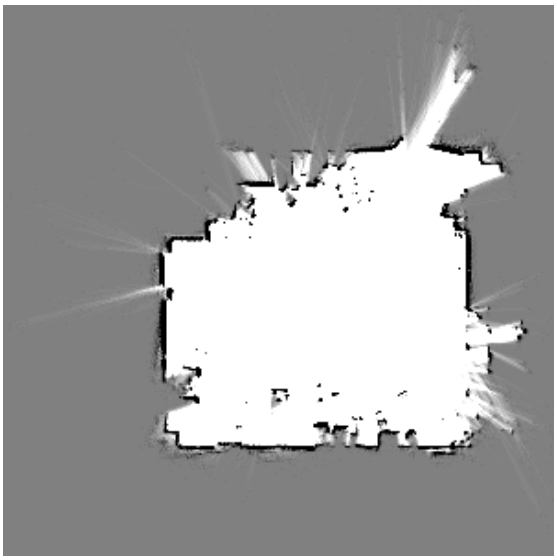

Figure 1: The grid map from hw2 data.

The purpose of this assignment is to gain proficiency in robot localization through the application of particle filters, often referred to as Monte Carlo Localization[1].

Specifically, you are tasked with developing a global localization filter to assist a misplaced indoor mobile robot (where 'global' implies that the robot's initial position is unknown). This lost robot navigates within Wean Hall using only odometry and a laser rangefinder. Luckily, you possess a map of the area shown as figure 1, and a comprehensive knowledge of particle filtering techniques to aid in its localization.

Algorithm 1 shows the basic MCL algorithm, which is obtained by substituting the appropriate probabilistic motion and perceptual models into the algorithm particle_filters. The basic MCL algorithm represents the belief $bel(x_t)$ by a set of M particles $\mathcal{X}_t = \{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}$. Line 5 in MCL algorithm1 samples from the motion model, using particles from present belief as starting points. The measurement model is then applied in line 6 to determine the importance weight of that particle. The initial belief $bel(x_0)$ is obtained by randomly generating M such particles from the prior distribution $p(x_0)$, and assigning the uniform importance factor M^{-1} to each particle.

Algorithm 1 Monte Carlo Localization (MCL)

Require: $\mathcal{X}_{t-1}, u_t, z_t, \mathcal{M}$
Ensure: $\mathcal{X}_t$
  $\overline{\mathcal{X}_t} = \mathcal{X}_t = \emptyset$
  for $m = 1$ to M **do**
    $x_t^{[m]} =$ sample_motion_model$(u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} =$ measurement_model$(z_t, x_t^{[m]}, \mathcal{M})$
    $\overline{\mathcal{X}_t} = \overline{\mathcal{X}_t} + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  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 $\mathcal{X}_t$

Here, we detail what each of the variables and symbols represent in the MCL algorithm above:

- $\mathcal{X}_t$ is a set of M particles representing the belief $bel(x_t)$, where the m-th particle in $\mathcal{X}_t$ is represented as $x_t^{[m]}$
- u_t represents the actions of the robot (i.e., control data) at time t
- z_t represents the measurement data the robot receives (i.e., sensor data) at time t
- $\mathcal{M}$ represents the map the robot has access to of its environment
- $\mathcal{W}_t$ is a set of M particles representing the particle importance weights, where the m-th particle in $\mathcal{W}_t$ is represented as $w_t^{[m]}$
- $\propto$ is equivalent to "proportional to"

The performance of your algorithm is dependent on (i) parameter tuning and (ii) number of particles. While increasing the number of particles gives you better performance, it also leads to increased computational time. An ideal implementation has a reasonable number of particles, while also not being terribly slow. Consider these factors while deciding your language of choice—e.g. choosing between a faster implementation in C++ or vectorized optimization in Python vs. using the raw Python skeleton code.

There are three tasks for you, the last one is optional:

1) Prove the equation below, maybe you will need to use the strong law of large numbers. (20%):

$$\lim_{M \to +\infty} \frac{1}{M} \sum_{m=0}^{M} w_t^{[m]} = p(z_t \mid z_{1:t-1}, u_{1:t}) \tag{1}$$

2) Monte Carlo localization. The target of this task is to predict where the robot is using particle filter. The detailed steps are explained in README.txt. Although there is no real-time-ness requirement, we encourage you to apply some implementation 'hacks' to improve efficiency:

   - Intializing particles in completely unoccupied areas instead of uniformly everywhere on the map.
   - Subsampling the laser scans to say, every 5 degrees, instead of considering whole measurements.
   - Since motion model and sensor model computations are independent for all particles, parallelizing your code would make it much faster.

   Make sure you describe your approach, performance, time and results, as well as the parameters you have to tune. Because the environment is too square and the results may be bad. Just show me the prediction – correction – resampling works well. (80%)

3) KLD-sampling increases the efficiency of particle filters by adapting the size of sample sets over time, implement it on your Monte Carlo Localization. There is a material[2] to read that will help you grasp this strategy. (bonus 20%)

Data Description:

The data has been packed into a rosbag file with two topics: [**/scan**, **/odom**].

- **/scan** contains ranges information of every laser scan.
- **/odom** contains the change in position over time of the robot.

Tips: The **/odom** might work in a higher frequency than the **/scan**. So you could add an function to check whether their time stamp is approximately same before feeding them into MCL.

Notes:

1) Both C++ and Python are acceptable. You can only use third-party libraries for matrix computation (i.e. NumPy and Eigen) and visualization (i.e. evo or Open3D or Matplotlib).

2) Attach your implementation with pdf in the zip. In the package, you also need to include a file named README.txt/md to identify the function of each file. Make sure that your codes can run and are consistent with your homework. We would then arrange a meeting after the deadline in which we would ask each one of you to come in for 10 minutes to demonstrate your solution on your own computer.

3) If submitted after the deadline but still within 24hrs, a 50% penalty is applied. If submitted more than 24hrs after the deadline, a zero score will be given. In special case, please contact Prof Kneip.

References

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1322–1328 vol.2, 1999. 1

[2] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005. 3