Monte Carlo Localization using Particle Filter

Lab Exercise 03

1. The goal

The goal of this practical exercise is to implement a Monte-Carlo Localization (MCL) algorithm to localize an one-dimensional Mobile robot being moved, with constant velocity, in a hallway. You will program all the code in Matlab.

2. The MCL Problem

The MCL localization is an implementation of the Markovian localization problem where the involved pdfs are represented through samples (particles) and the Bayes filter is implemented through the Particle Filter. Markov localization addresses the problem of state estimation from sensor data. Markov localization is a probabilistic algorithm: Instead of maintaining a single hypothesis as to where in the world a robot might be, Markov localization maintains a probability distribution over the space of all such hypotheses. The probabilistic representation allows it to weigh these different hypotheses in a mathematically sound way.

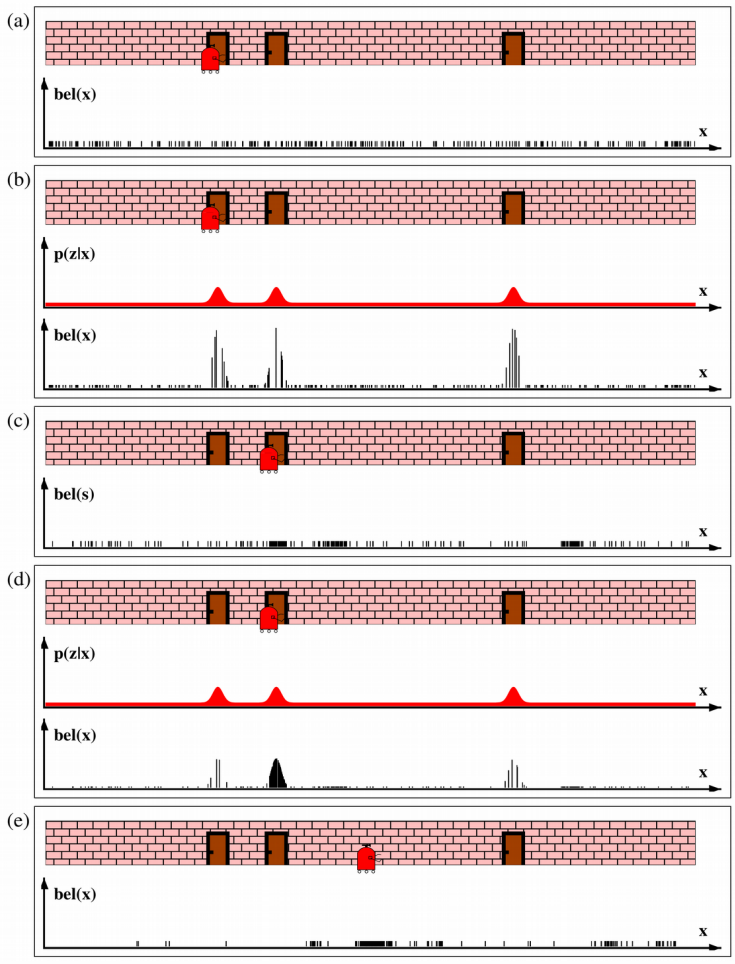

Figure 1: MCL of a Mobile Robot Moving in a Hallway.

The initial global uncertainty is achieved through a set of pose particles drawn at random and uniformly over the entire pose space, as show in Figure 1a. As the robot senses the door, line 5 of in the algorithm MCL assigns importance factors to each particle. The resulting particle set is shown in Figure 1b. The height of each particle in this figure shows its importance weight. It is important to notice that this set of particles is identical to the one in Figure 1a—the only thing modified by the measurement update are the importance weights.

Figure 1c shows the particle set after re-sampling (line 8-11 in the algorithm MCL) and after incorporating the robot motion (line 4). This leads to a new particle set with uniform importance weights, but with an increased number of particles near the three likely places. The new measurement assigns non-uniform importance weights to the particle set, as shown in Figure 1d. At this point, most of the cumulative probability mass is centered on the second door, which is also the most likely location. Further motion leads to another re-sampling step, and a step in which a new particle set is generated according to the motion model (Figure 1e). As should be obvious from this example, the particle sets approximate the correct posterior, as would be calculated by an exact Bayes filter.

3. The Algorithm

The Figure 2 shows a general Particle Filter and Figure 3 the pseudocode of the MCL Algorithm as an instance of the Particle Filter.

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Figure 2: Particle filter algorithms.

1: **Algorithm MCL**($\mathcal{X}_{t-1}, u_t, z_t, m$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Figure 3: MCL Algorithm.

4. Work to do

During these lab session you must complete the m-file containing the MCL algorithm. These are the steps you must follow:

1. **Read the m-file** and try to understand how it works. Even though it is incomplete, it is possible to run it. For the animation to work, it is necessary to have the `animation.mat` file provided.
2. Read the help of the **random** and **pdf** MATLAB functions. They will be useful for this lab.
3. Program the **sample_motion_model** and the MCL functions (focusing on the prediction section of the last one) and complete the main iteration.
4. Program the **measurement_model** and the MCL function (focusing on the update section of the last one).
5. Program the **re-sampling** step of the MCL function.
   All the code sections to be programmed are tagged with the tag
   `% COMPLETE THIS FUNCTION`

   Note: you cannot use the built-in Matlab function `randsample()`, it is part of the assignment to implement a resampling method.
6. After finishing the lab exercise: u**pload the final m-file.**

   Note: Report is NOT required! If the code is running it should be enough. Only write a report if you believe your solution is not working properly, explaining the details. In case you decide to write a report, it shouldn't be more than **4 pages.**

5. Documentation (complement material)

The lab can be solved with few lines of code, however understanding is key. Since no report is required, most of the time will be expending on (self)-learning. In order to complement the lecture (which doesn't include MCL), it is highly recommended to see some of the great videos from Sebastian Thrun in the Udacity curse: "Artificial Intelligence for Robotics".

5.1 Videos

Link of the Udacity course can be found next. In particular for the Lesson 1, it is also recommended to complete the Quiz questions (which are in Python), it will give you some insight in how to complete the lab, the code is quite similar.

- Lesson 1 [Localization]
- Lesson 3 [Particle Filters] (some of them)

5.2 Book

This lab exercise is based on the **Probabilistic Robotics** book [ETH library]. Available digital version [pdf].

If you want to complement the learning even further, the recommended chapter are: Chapter 1., Chapter 2. (Bayes filter included), Chapter 8. (until 8.3.3)