CS 287, Fall 2011 Problem Set #1 Version 4 Particle Filtering, Importance Sampling, Localization, Mapping, SLAM

Deliverable: (1) 6-10 page write-up in pdf format (2) zip file with your source code and clear "main" file that can be run to generate your results for each programming related question. Due date/time: Tuesday October 25, 23:59pm. Email to pabbeel@berkeley.edu.

Please refer to the class webpage for the homework policy.

Various starter files are provided on the course website.

When making your write-up, make sure to answer all questions, and include and discuss plots (and if helpful, snippets of code) which are helpful in demonstrating that your system works and in answering the questions.

---

1. Particle Filtering is a Biased Sampling Procedure for Finite Sample Sizes

This is closely (but not exactly) following exercise 8.4 (a) and (b) from Probabilistic Robotics.

Particle filtering is biased for any finite sample size—i.e., the distribution from which the particles are drawn is different from the true distribution we 'd want to sample from. In this question you are asked to quantify this bias.

Consider a world with four possible robot locations $X = \{x_1, x_2, x_3, x_4\}$ Initially we draw $N \geq 1$ samples uniformly from among those locations. As usual, it is perfectly acceptable if more than one sample is generated for any of the locations. Let Z be a Boolean sensor variable characterized by the following conditional probabilities:

$$\begin{aligned} p(z|x_1) &= 0.8 = 1 - p(\neg z|x_1) \\ p(z|x_2) &= 0.4 = 1 - p(\neg z|x_2) \\ p(z|x_3) &= 0.1 = 1 - p(\neg z|x_3) \\ p(z|x_4) &= 0.1 = 1 - p(\neg z|x_4) \end{aligned}$$

Particle filtering uses these probabilities to generate particle weights, which are subsequently normalized and used in the resampling process. For simplicity, let us assume we only generate one new sample in the resampling process, regardless of the initial number of samples N. This sample might correspond to any of the four locations X. Thus, the sampling process defines a probability distribution over X.

1. What is the resulting probability distribution over X for this new sample? Answer this question separately for $N = 1, 2, \ldots, 8$, and for $N = \infty$.

2. The difference between two probability distributions p and q can be measured by the KL divergence, which is defined as

$$KL(p, q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)}.$$

What are the KL divergences between the distribution in (a) and the true posterior? Plot the KL divergence as a function of N.

Hint: The above questions might be most easily answered through writing a small program that computes the requested distributions (exactly!). No sampling should be run in such program used to answer the above questions exactly.

2. Importance Sampling

Consider the probability distribution with density

$$p(x, y, z) \propto f(x, y, z) = e^{-\frac{1}{2}\left(x^2+y^2+z^2+x^2y^2+y^2z^2+x^2z^2+\sin(x+0.1y+0.2z)+1\right)}.$$

1. Describe an importance sampling procedure to obtain samples from this distribution.
2. Describe how through importance sampling you can estimate the integral $\int_{x,y,z} f(x, y, z)dxdydz$. Implement your procedure. Run it 10 times with up to 10,000 samples and plot as a function of the number of samples drawn your estimate of this integral for each of the runs. What is your estimate for the value of this integral? (Hint: in Matlab "randn" samples from a standard normal distribution.)

3. Localization

In this question you are asked to implement Monte Carlo localization and empirically investigate some of its properties.

Debugging hint: make sure your code runs relatively quickly so you are not wasting time sitting around waiting for it to finally show its results. When using Matlab, for-loops tend to be bottlenecks, you should consider vectorizing your code. Also, some of these algorithms just take a long time to run (e.g., when it involves ray-casting). Be clever about trying out small test-cases first rather than always debugging on the entire data sequence.

1. **Motion Model.** Implement the "sample_motion_model_velocity(u_t, x_{t-1})" from Table 5.3 in Probabilistic Robotics, also shown here:

$$\begin{aligned}
\hat{v} &= v + \text{sample}(0, \alpha_1 v^2 + \alpha_2 \omega^2) \\
\hat{\omega} &= \omega + \text{sample}(0, \alpha_3 v^2 + \alpha_4 \omega^2) \\
\hat{\gamma} &= \text{sample}(0, \alpha_5 v^2 + \alpha_6 \omega^2) \\
x' &= x + \frac{\hat{v}}{\hat{\omega}}(\sin(\theta + \hat{\omega}\Delta t) - \sin(\theta)) \\
y' &= y + \frac{\hat{v}}{\hat{\omega}}(\cos(\theta) - \cos(\theta + \hat{\omega}\Delta t)) \\
\theta' &= \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t
\end{aligned}$$

   For "sample(μ, σ^2)" use a Gaussian distribution with mean μ and variance σ^2.

   Make the robot (up to noise) go forward 2 meters, then make a left turn of approximately 90 degrees with turning radius of 0.5 meters, and then go forward another 4 meters. Use discrete time-steps of 0.1s and a velocity of 1m/s. Set all $\alpha_i = 0$ and plot its trajectory. Then set all $\alpha_i = 0.1$ and run the motion model 100 times, plot the trajectories. Verify that increasing and decreasing each α_i has the expected effect. For α_1 and α_3, report your findings, include plots illustrating the variation in sampling distribution as a function of α_1 and α_3.

   Make sure to use "axis equal" to ensure both axes are scaled equally.

2. **Motion Model and Map.** Load the map "simple_world" and run the motion model a few times, this time starting from x_0 as assigned in the starter file, and with the control inputs provided in the starter file. Check for collisions, and illustrate effectiveness by plotting the map and 5 generated trajectories in which states in collision are plotted in green, states in open space are plotted in red.

3. **Laser Beam Sensor Model.** Implement sampling from the laser beam sensor model. Plot the laser beams generated from the model for locations and parameter settings specified in the starter file.

4. **Probability of Sensory Readings under Laser Beam Sensor Model.** Implement a function that evaluates the probability of a laser range finder scan given a pose and a map. Evaluate the probability for the laser range finder scan, pose and map provided in the starter file. For the provided location and map generate a noise-free laser range finder scan. Then for a range of variances for the laser range finder evaluate and plot the log probability of the measurements when varying x, y, and θ respectively. Briefly discuss these plots. For the provided scan, laser model and map, find the most likely pose(s).

5. **Particle Filter, Known Initial State.** Implement the "basic" particle filter, which for each time step: (i) samples from the dynamics model, (ii) re-weights samples based on the likelihood of the measuremetns, (iii) randomly resamples from the weighted particles according to their weights. Do so for all four worlds provided. Discuss performance as a function of the number of particles. (Just a few particles should suffice for this particular data.) Make sure to report on your results for the 4th world (for which you are not provided with ground-truth data) .

6. **Multiple Hypotheses, Low Variance Sampling.** For this question investigate how often your particle filter is able to extract the "right" result when initialized with a set of particles around four potential initial locations, which have pairwise symmetry. Implement low-variance sampling and report on its performance.

7. **Global Localization.** Extend your particle filter with regularization (i.e., add a bit of additional random noise after resampling) and when resampling draw some samples uniformly at random from the entire state-space. Also try out downweighting the likelihood by raising it to a power between 0 and 1. For the three different worlds discuss performance of your particle filter as a function of regularization and uniform resampling. If successful, try to perform localization for the three settings for which no ground-truth is available. I suggest staying in the regime in which your algorithm runs for at most 5 minutes for a given experiment. You might need to achieve this, for example, by initializing by sampling uniformly at random in only 1/10 of the state space, by running over a shorter horizon, by only going up to a certain number of particles. Run the MATLAB profiler ("help profile" to find out more) and report on your computational bottlenecks. I expect the raycasting and evaluation of the laser scan measurement probability to be your bottlenecks, and we will resolve that in the next sub-questions through using the likelihood field. If not the case, and you can't find a way to speed up the other parts, talk to some other students or come talk with me.

8. **Likelihood Field.** We now replace the beam sensor model with a faster (albeit less accurate) sensor model: the likelihood field. Implement a function that computes the likelihood field from a gridmap and plot the likelihood fields for each of the three worlds in consideration.

9. **Global Localization with Likelihood Field.** Run global localization with the likelihood field. Play around with amount of regularization, down-weighting the likelihood of the measurements (through exponentiating the likelihood), fraction of samples being drawn uniformly at random. Try at least one additional little improvement not listed in preceding sentence, this improvement could relate to how much you resample uniformly at random, letting the number of particles vary over time, or yet something else you come up with. Discuss performance for the three cases for which the path of the robot is known as a function of the various choices and parameter settings in the particle filter. Submit your estimated (probability distributions over) paths for the robot for the three cases when it is unknown.

4. Mapping

For this question you will implement grid-mapping using reflection maps. The starter code includes the results of my implementation for three worlds. Make sure to compare your results with mine. Also make sure to discuss why (if anywhere) the obtained map does not look like the real map. The starter code also contains a trajectory with measurements for which the map is unknown. Make sure to include your obtained map into your report.

I updated the slides of Lecture 5 to include a table you need from the book. Get the "v-3" slides on mapping here: http://www.cs.berkeley.edu/%7Epabbeel/cs287-fa11/slides/Lecture5/.

5. SLAM

1. **Map from Raw Motion Model.** Run the motion model in open loop and perform mapping assuming the path generated by running the motion model in open-loop is the true path. Compare to the real maps.

2. **Optional/Extra Credit: Particle Filter: Motion Model Proposal, No Resampling.** Provide the pseudo-code of a particle filter for SLAM that uses the motion model as its proposal distribution and does not perform resampling. Discuss and illustrate its performance by including a few maps found and the relative weights for these maps.

3. **Optional/Extra Credit: Particle Filter: Motion Model Proposal, Resampling.** Now also perform resampling. Discuss experimental performance.

4. **Optional/Extra Credit: Particle Filter: Improved Proposal, Resampling.** Provide the pseudo-code of a particle filter for SLAM that uses an improved proposal distribution. Make sure to clearly describe both sampling from the improved proposal and re-weighting. Discuss experimental performance.

6. Feedback

Anonymous electronic feedback form forthcoming. Towards improving this assignment for next year any constructive suggestions are greatly appreciated. E.g., in question X it would have been nice had you provided the code for Y b/c it took a really long time compared to the amount of learning involved; in question Z it would have been a great learning experience if we had to do W ourselves instead of being given the code; it would be great if the lecture slides had a bit more detail on V.