Homework 3: Sampling and Particle Filter

Due: 11:59PM, April 30, 2025

Software

Python packages *NumPy* and *Matplotlib* are sufficient for the calculation used this assignment. The computation of this assignment could take a while to finish on your computer, but it should not take more than *a few minutes*. Even though not required, if you are interested in accelerating the computation, we encourage using Python package *JAX* or *PyTorch* with GPU acceleration (Google Colab provides free GPU access). Feel free to use Jupyter Notebook as the programming environment.

1. (40 pts) Given the image here, convert it into a continuous probability density function over a space of 1 meter by 1 meter (you can find the example code for how to do it here). Implement rejection sampling to sample 5000 points from this image-based probability distribution. Select two different proposal distributions of your choice for your implementation.

   Turn in: Two plots showing the resulting samples, one for each of the proposal distributions that you choose. Specify which proposal distribution you choose to use. The resulting samples should look similar to the figure below.

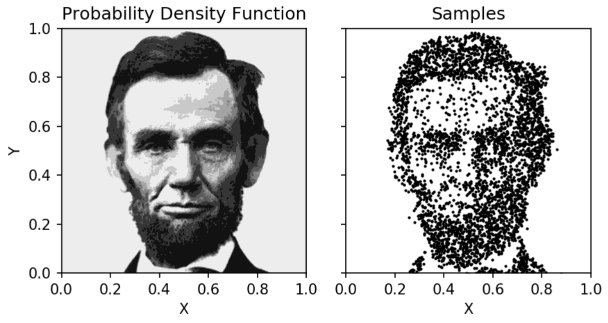

Figure 4: Example of the rejection sampling from the image.

2. (60 pts) Consider the following differential drive vehicle robot. The robot continuously executes a constant control signal $[u_1(t), u_2(t)[= [0.5, -0.63]$ for a length of time $T = 5$ seconds. The dynamics and the initial state of the robot are:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta)u_1 \\ \sin(\theta)u_1 \\ u_2 \end{bmatrix}, \ (x(0), y(0), \theta(0)) = (0, 0, \pi/2).$$

   To simulate the trajectory of the robot, use a time interval $dt = 0.05$ and implement the Runge-Kutta method for numerical integration. The ground truth trajectory of the robot is noisy, with a $2d$ zero-mean Gaussian noise with a covariance of $diag([0.04, 0.02])$ to the control signal at every time step.

   Even though you will need to simulate the noisy ground truth state of the robot at each time step, your Bayesian filter does not have access to this information. Instead, we start with an initial belief of the system as a Gaussian distribution with the mean being the initial state

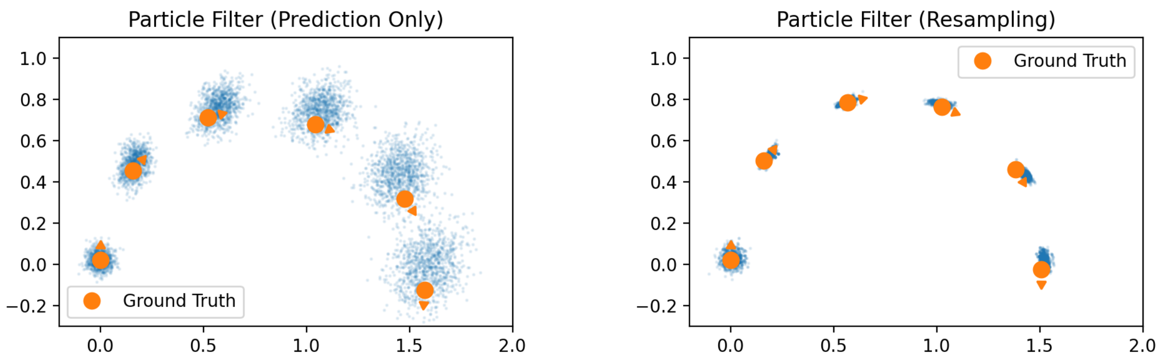

Figure 5: Example visualization of the particle filter with and without resampling.

and the covariance being $diag[10^{-3}, 10^{-3}, 10^{-4}]$ around the initial state. At each time step, you will receive (simulate) a noisy measurement from the hidden ground truth state of the robot. The measurement model is a Gaussian distribution with the mean being the current ground truth state of the robot and the covariance being $diag[0.004, 0.004, 0.002]$.

Turn in: First, implement only the prediction step of the particle filter (without incorporating the measurement for belief update) and simulate the system for 100 time steps using 1000 particles. Turn in the plots of the particle-based belief estimation and the ground truth state at time steps: 0, 20, 40, 60, 80, 100.

Second, implement the resampling step in the particle filter. Turn in the plots of the particle-based belief estimation and the ground truth state at time steps: 0, 20, 40, 60, 80, 100. For resampling, a very helpful utility function would be the "random.choice" function from NumPy.

An example of the visualization is provided.