EECS 498: Introduction to Algorithmic Robotics
Fall 2020
Homework Assignment #5
Due 11/30/2020 at 2:59pm

Rules:

1. **All homework must be done individually, but you are encouraged to post questions on Piazza.**
2. No late homework will be accepted.
3. The goal of this homework is to develop your understanding of the algorithms presented in class. You should use python to implement solutions. You may not use any other language, only python will be accepted.
4. Submit your python files along with a `pdf` of your answers in a zip file to Gradescope. Do not paste your code into your `pdf`.
5. Remember that copying-and-pasting code from other sources is not allowed.

Questions

1. (10 points) Consider the binary random variable A. Starting from $P(a \vee \neg a) = 1$, use the axioms of probability to derive $P(\neg a) = 1 - P(a)$.
2. (10 points) AI Book Chapter 13 #13.8
3. (10 points) AI Book Chapter 13 #13.16
4. (10 points) AI Book Chapter 14 #14.5 part a
5. (15 points) AI Book Chapter 14 #14.6
6. (10 points) AI Book Chapter 15 #15.2
7. (15 points) AI Book Chapter 15 #15.13
8. (10 points) Describe the advantages and disadvantages of a Particle filter vs. an Extended Kalman Filter vs. an Unscented Kalman Filter. Specifically, consider the computational cost, the ability to handle different kinds of dynamics, and the type of state uncertainty distributions each can handle.

Software

1. Download and unzip `HW5_files.zip`. Run `kf.py`. You should see some lines and points plotted. If you do not see this, make sure `matplotlib` is installed.

Implementation

The following implementation problem should be done in python starting from the provided templates. Only edit what is inside the `### YOUR CODE/IMPORTS HERE ###` block. Feel free to look around the internet for example code for reference, but you should implement your own code from scratch unless explicitly stated otherwise. **Include all code you write in your `zip` file.**

1. In this problem we will implement a Kalman filter given the motion model, sensor model, and data. Consider a robot with the following motion model:

$$\begin{aligned} x_{t+1} &= x_t + 1.5u_1 + 0.1u_2 + \zeta_x \\ y_{t+1} &= y_t + 0.2u_1 - 0.5u_2 + \zeta_y \end{aligned} \tag{1}$$

   and sensor model:

$$\begin{aligned} z_1 &= 1.05x + 0.01y + \delta_1 \\ z_2 &= 0.01x + 0.90y + \delta_2 \end{aligned} \tag{2}$$

   a (10 points) To use a Kalman filter, we first need to specify the models in the standard form:

$$\begin{aligned} x_t &= Ax_{t-1} + Bu_t + \zeta_t \\ z_t &= Cx_t + \delta_t \end{aligned} \tag{3}$$

   where ζ is the motion noise and δ is the sensor noise. For this form, specify x (the state), u (the controls), and the matrices A, B, and C from the equations above in your `pdf`. Input your A, B, and C matrices in `kfmodel.py`.

   b (15 points) We also need to specify the matrices R and Q, the motion and sensor noise covariances, respectively. To do this, we will compute the covariances from data. We will assume that both the motion and sensor noise are zero-mean and normally-distributed. Write code in `tuning.py` to compute the covariances. This script loads in the data file `kfdata.dat`, which contains the measurements, ground truth states, and actions. `tuning.py` loads in the A, B, and C matrices from `kfmodel.py`, so make sure you have completed part (a) before you do this part. Include what you compute for R and Q in your `pdf`.

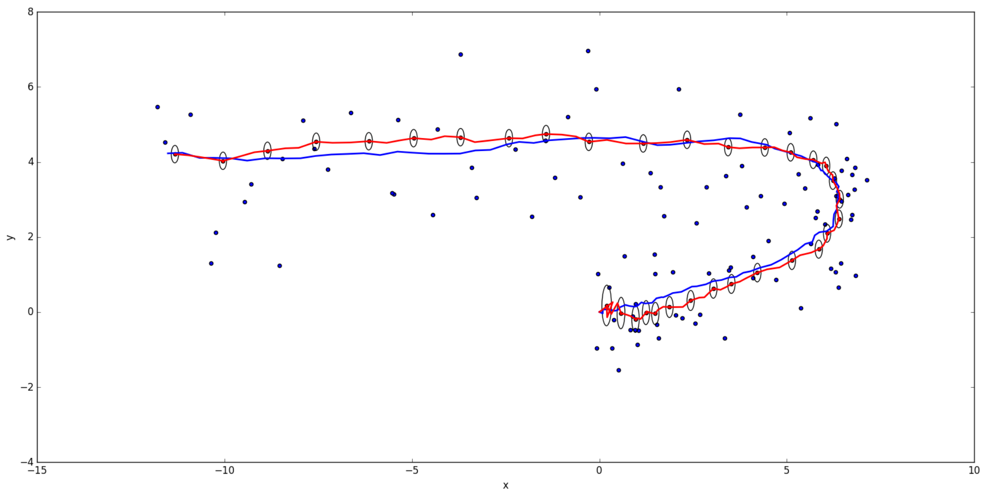
   c (30 points) Now we are ready to write the Kalman Filter. Open `kf.py` and carefully read through the code to understand what it is doing. When you run `python kf.py`, you will see a plot with the following:

   - Blue line: ground truth state trajectory
   - Blue dots: measurements
   - Red line: estimate state trajectory (the output of your Kalman Filter)
   - Black ellipses: covariance plotted as an ellipse at 2σ. The covariance is only drawn at every 3rd state so the plot doesn't get too cluttered

   It will also print out the total error, which is the sum of the magnitudes of the errors between your estimated states and the ground truth states.

   `kf.py` loads in the A, B,C,R, and Q matrices from `kfmodel.py`, so make sure you complete parts (a) and (b) before doing this part.

You will fill in the `KalmanFilter` function in `kf.py` so that it outputs the estimated mean and covariance after receiving a new measurement and action. Your Kalman filter should not use the ground truth states in any way, those are only used to compute the total error. If you have completed all the parts correctly, your output should look similar to the figure below.



The total error should be 22 ± 1. Include a screenshot of your output and your total error in your `pdf`. We will run your code to verify it. If the code crashes, you will not receive credit for this part of the assignment.