Name: Siddharth Chamarthy
UNI : sc4253

Assignment 5
Extended Kalman Filter Vs Particle Filter

Merits of my implementation of Particle Filter:

- It works best with multimodal systems
- It can be used to implement state estimation regardless of the number of variables or dynamic equations. (Both linear and Non-Linear systems)
- Since the random generation of particles depends on the previous state of those particles, its approximation of the state is quite close to the true state.
- It scales well over time. It becomes more and more accurate as we increase the number of particles or increase the amount of computation time.
- It uses a random generation of particles which doesn't require a specific Probability Density Function. In my case to generate the random particle states, I used "**numpy.random.uniform"** to generate noises to each of system's dynamic equations which gives a prediction to the estimate.
- It assigns equally likely weights initially to all the particles as they are assumed to be equally likely to be the estimate of the current state of the system. In my case, I have given weights as **1/(number of particles)**, so that the weights will add up to 1.
- Using these weights, I have performed resampling of the particle using weights using an algorithm that is as follows:
    - find sum of weights of all the particles
    - generate random number between 0 and the total sum of weights
    - start a loop:
        - subtract the weight of each particle from the sum of weights and assign it to a variable
        - check if that variable is less than the random number generated
            - if True, then add the variable to a new list of objects
            - if False, continue subtracting
    - This above algorithm is repeated till the new list is filled with the required number of particles.
- Once the new list is complete, I have calculated mean state using the weighted average of each particle's state.
- Publish this state.

Demerits of my implementation of Particle Filter:

- When we have no sensor data, the state of the robot depends completely on random noise which is an extremely bad demerit as we cannot predict its state between two intervals of sensor readings.
- It is computation heavy and hence has a delay in publishing states.
- It can be inaccurate if the sensor readings are not informative enough. For example, there can be two clusters of particles in two different regions of the map representing a similar state due to multiple solutions to a specific system dynamics. In such a case the particle filter can show the wrong state estimate.
- The filter is non-deterministic, so it cannot determine is exact state of the robot as they are all random particles whose state we cannot predict or debug.

Extended Kalman Filter:

The behavior observed in the extended Kalman filter was better compared to the particle filter because:

- It gives proper estimates when there are atleast 2 landmarks within the sensor range.
- It doesn’t give random estimated positions when no landmark is within range.
- It gives an approximation to the estimate when there is one landmark.
- It works best with unimodal systems

Performance Requirements:

Extended Kalman Filter:

- It gives the most optimal solution for both linear and non-linear model of system dynamics.
- Works best with unimodal systems.
- If the initial estimate is wrong, the filter will quickly diverge without converging to a solution.
- Works only on the basis of a normal distribution for its probability density function.
- Performs best in a unimodal linear system

Particle Filter:

- It gives the optimal solution for any set of system dynamic equations.
- It can be used to estimate multimodal systems.
- It requires a large number of particles to be sampled so as to get a very accurate estimate.
- It requires all the sensors to be very informative so as to choose the correct cluster of randomly generated weighted particles.