**Robot Localization using Particle Filters**

**Short description about the task:**

We have a robot with is moving inside a building and we have the building map, but we don’t know where in map robot is located at this moment and to which direction robot is looking (i.e., we don’t know the location and orientation of the robot). Robot has an imaginary sensor which return the distance to closest door to the robot. if there are two doors in the corridor that robot is moving inside the robot only knows the distance to the closest one. Distance to the door is our measurement in this task and we also have a robot movement model which returns the next state of the robot given the current state and the action that robot takes at this state. At state *n*, robot has a location and an orientation and it is defined as a set of three values (). Robot can take one action among a set of four possible actions at each state, actions are 1- moving forward, 2- moving backward, 3- rotating to the left and 4- rotating to the right. Robot take one of these actions at each state and the next state is calculated using the robot movement model defined as eq (1):

(1)

Where is equal 10 units (here unit is pixels). Each time robot takes an action, it moves based on the movement model and added noise.

There is an imaginary sensor on robot which gives the distance and orientation of the closet door proportional to the robot, which is considered as our measurement in this task. As mentioned, measurement has two elements, distance, and angle, and it is defined based on the eq (2):

(2)

At the beginning of the game a robot and 1000 particles (i.e., red square is the robot and empty yellow squares are particles) with random position and random orientations are generated in the middle of a maze. The distance and the angle difference between robot and the closet door in the maze is measured and using eq (2) and then based on this measurement each particle is given a weight. Particles weight is calculated by measuring the distance and angle difference to the closest door to that particle using eq (2) and then calculating the weight for that particle based on the eq (3):

(3

After calculating wight for each particle, weights are normalized. Then we calculate a weighted mean of the particles’ location and orientation as an estimation of robot’s location (i.e., robots estimated location and orientation is shown as a blue empty square).

**Prediction**

As soon as the robot takes an action, we do resampling. Resampling is generating new particles based on the particles with higher weights from last state. So, we generate 1000 new particles, based on the last set of particles and their weights, weight of each particle is the probability of that particle to be chosen in new set of particles. Then a gaussian noise is added to the particles. Each particle has three elements *x, y* and and we add noise to each of these elements separately based on and respectively for x, y and .

**Update**

These newly generated samples are transformed with the movement model of the robot which is shown in eq (1), and generate a set of new particles which are potential prediction for the new robot state.

**Measurement**

We measure the robot distance and angle compared to the closet door and we do the exact same thing for each particle and then generate weight for particles based on the similarity of their distance and angle to the closet door, with robot’s distance and angle to the closet door and generate a new prediction of robot’s current position and orientation by weighted mean of these particles.