COL864 Special Topics in AI
Assignment-1

Somanshu Singla 2018EE10314
Lakshya Kumar Tangri 2018EE10222

September 8, 2022

State Estimation Models
Kalman Filter, EKF and HMM

Contents

1 Question 1 : Single Agent Position Estimation

In this problem our agent is an aeroplane flying in continuous 2-D space and has a GPS Sensor on it which can make measurements regarding its position.It also has velocity increment based controller(action) in our case. In this problem we have defined our state X_t to be defined using position and velocity of the aeroplane.

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x_t} \\ \dot{y_t} \end{bmatrix} \tag{1}$$

1.1 (a) Implementing motion and observation model

As we know in this problem motion and sensor model is linear i.e

$$X_{t+1} = A_t X_t + B_t u_t + \epsilon_t \tag{2}$$

$$z_t = C_t X_t + \delta_t \tag{3}$$

where ϵ_t and δ_t are gaussian distributed zero mean noises present during motion and measurement respectively .

Motion Model

$$X_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \dot{x_{t+1}} \\ \dot{y_{t+1}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \dot{x_t} \\ \dot{y_t} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\delta x_t} \\ \dot{\delta y_t} \end{bmatrix} + \epsilon_t \tag{4}$$

here $\epsilon_t \sim \mathcal{N}(0, R)$ where

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix} \tag{5}$$

Sensor Model

$$z_t = \begin{bmatrix} x'_t \\ y'_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \dot{x_t} \\ \dot{y_t} \end{bmatrix} + \delta_t \tag{6}$$

here $\delta_t \sim \mathcal{N}(0, Q)$ where

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{7}$$

Inputs taken to generate the plot:
We took $\Delta t = 0.1$ as a high Δt will lead to us not being able to observe the difference between Actual and Observed Trajectories Properly.
The control inputs were both taken to be 0.

$$X_0 = \begin{bmatrix} 0 \\ 0 \\ 10 \\ 10 \end{bmatrix} \tag{8}$$

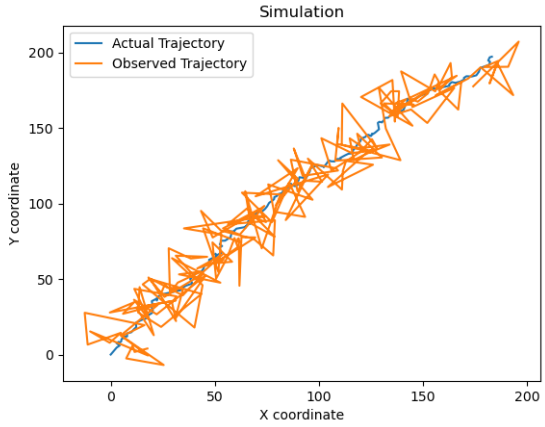


Figure 1: Plot for Actual and Observed Trajectory

1.2 (b) Implementing Kalman filter

Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

$$\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$$
$$\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$$
$$K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$$
return μ_t, Σ_t

To implement this algorithm apart from action u_t and observation z_t , We can see the motion and sensor model present at (4) and (6) which give us values of $A_t, B_t, C_t, R_t = R, Q_t = Q$ and we have been told to assume

$X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ where

$$\mu_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \Sigma_0 = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix} \tag{9}$$

1.3 (c) Estimated and True Trajectory Plot

The control input for this part and all further parts as mentioned in the problem statement was taken to be , Δt was taken to be 0.1:

$$u_t = \begin{bmatrix} cos(n \cdot \Delta t) \\ sin(n \cdot \Delta t) \end{bmatrix} \tag{10}$$

The plots are as follows :

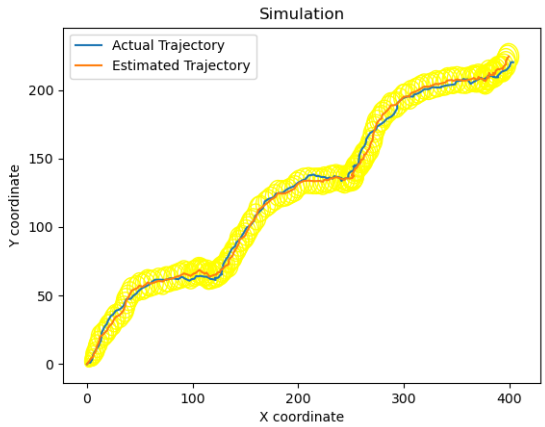


Figure 2: Plot for Actual and Predicted Trajectory

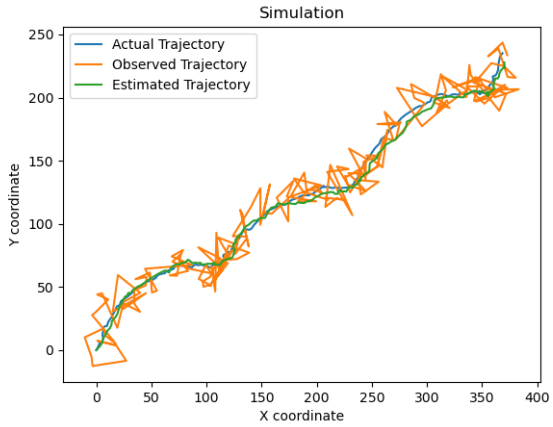


Figure 3: Plot for Actual, Predicted and Observed Trajectory

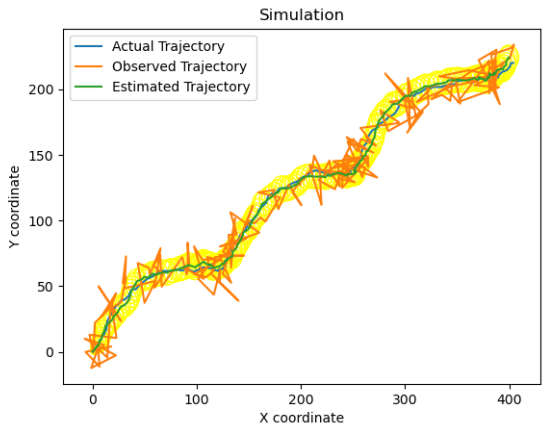


Figure 4: Plot for Actual, Predicted and Observed Trajectory

[0]The Yellow Colour Region shows Uncertainty Ellipses

1.4 (d) Uncertainty in state estimate in absence of measurements

We expect that after measurement updates start again the estimated state is less erratic and is much more closer to the true state.

Experimental results:

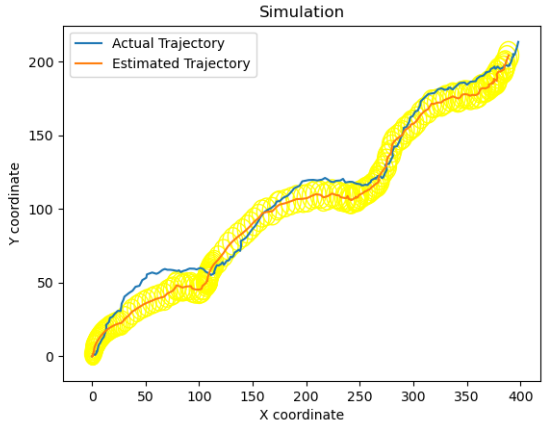


Figure 5: Plot for Actual and Predicted Trajectory

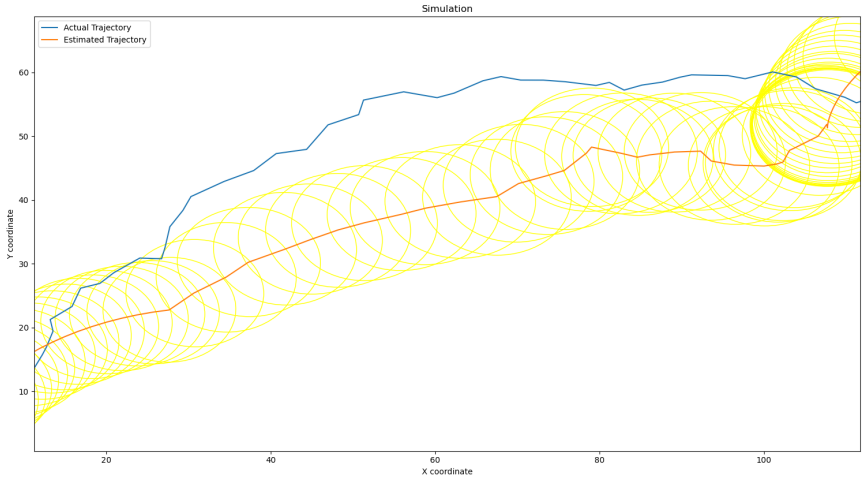


Figure 6: Plot for Trajectories zoomed at the time where observations get lost

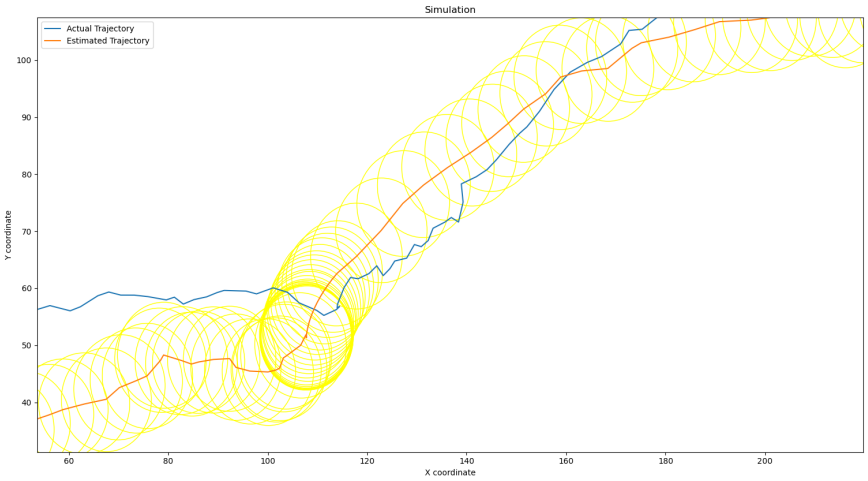


Figure 7: Plot for Trajectories zoomed at the time where observations are present

Our hypothesis is verified by these plots as when the observations get lost the ellipses become sparser and bigger, whereas when observations are present the ellipses are much denser and smaller as visible in above plot when for the brief period observations return the ellipses became much denser and also smaller.

1.5 (e) Estimated and True Velocity Plot

Yes the estimator can track values when our initial guess is close and not when initial belief is off, this is expected as we don't observe velocities directly and so if initial belief is far the model fails to track.

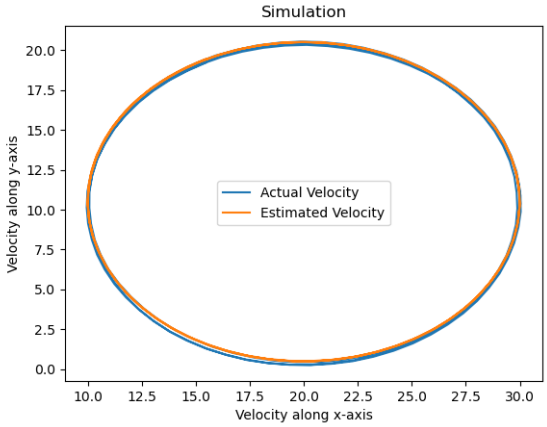


Figure 8: Velocity Estimation with good initial belief

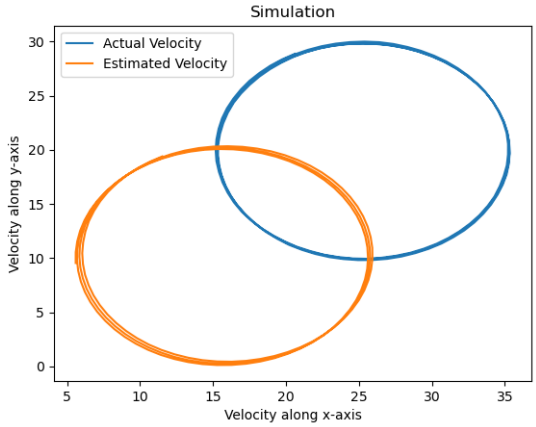


Figure 9: Velocity Estimation with poor initial belief

This happens because velocity is part of the unobserved state hence our model is not the best estimator of velocities.

1.6 (f) Data Association

The strategy that we have adopted for this problem is nearest neighbour based.
Formally:
Given a set of beliefs $\{\mu_1, \mu_2 \cdots \mu_n\}$ and a set of observations $\{z_1, z_2 \cdots z_n\}$ where n is the number of agents.
So, for each μ_i we assign a z_i which is the closest to it based on some distance metric.

Some choices we explored for the distance metric are:

1. Euclidean Distance
2. Manhattan Distance

We finally decided to go with *Euclidean Distance* as that was working well and that seemed like a better way to estimate the closeness of belief and observation. The plots below are based on that strategy.

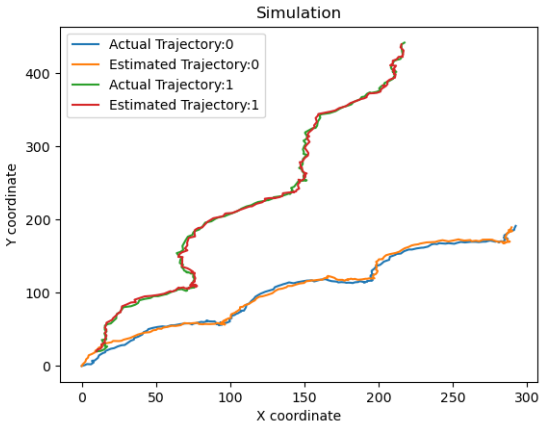


Figure 10: Data Association on 2 agents

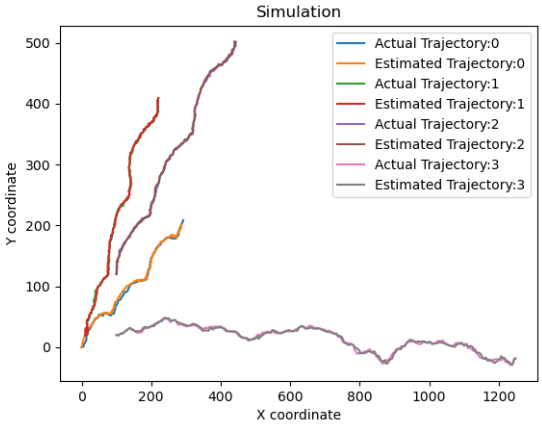


Figure 11: Data Association on 4 agents

2 Question 2 : Position Estimation with help of Landmarks

2.1 (a)Incorporating distance from landmark into estimator

We modified our agent to change its observation vector to include landmark if its within range So,Now we have the same action model but a nonlinear observation model

$$z_t = \begin{bmatrix} x'_t \\ y'_t \\ d'_t \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \sqrt{(x_t - x_l)^2 + (y_t - y_l)^2} \end{bmatrix} + \delta_t \tag{11}$$

here $\delta_t \sim \mathcal{N}(0, Q)$ where

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & \sigma_l^2 \end{bmatrix} \tag{12}$$

and σ_l is standard deviation(1) given in problem. Now since observation $z_t = h_t(x_t) + \delta_t$ is no longer linear we need to implement Extended Kalman Filter which requires Jacobian H_t of h_t during measurement update.Therefore we developed a Hybrid filter which does measurement updates based on the observation provided.If no landmark is involved then Kalman Filter's measurement update is performed by estimator else EKF's update is performed

2.2 (b) Implementation for simulation

We extended the simulation to account for landmarks given in question with respective range of 30 m and used the above part's idea to implement our filter

2.3 (c) Simulation with Uncertainty Ellipses

Now we simulated plotting actual and estimated trajectory for 800 time steps. Unfortunately the agent doesn't come in landmark's sphere of influence so change in uncertainty ellipse is minimal but when we added landmark in path in part 2.5.We see that inside the landmark since we receive additional information,the uncertainty ellipse shrinks and localization performance improves.

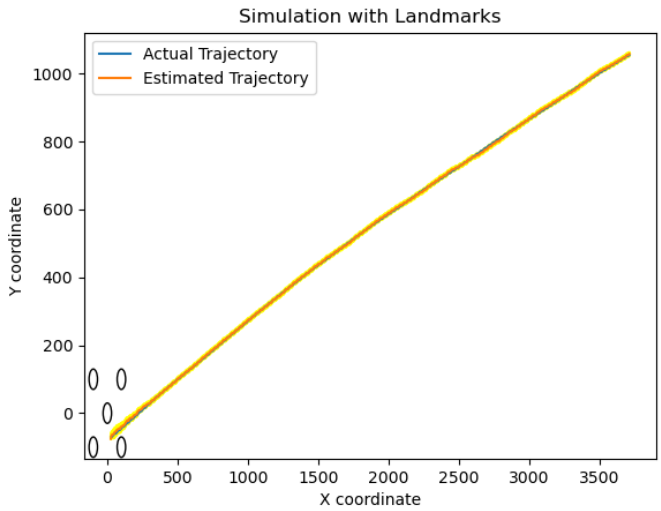


Figure 12: 2c no obs

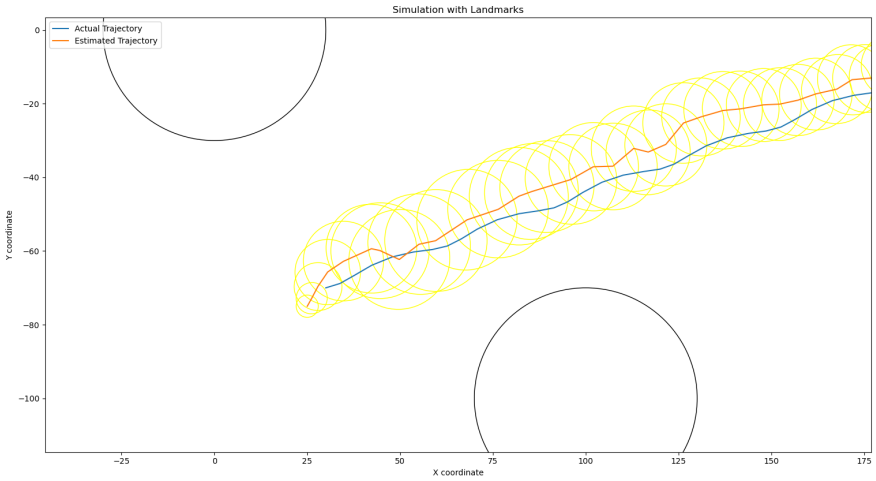


Figure 13: Seeing effect on uncertainity ellipse

2.4 (d) Standard Deviation Variation

As we make distance measurement more noisy ,we can expect that our localization/state estimation problem would become much more difficult.Since in the above parameters the agent doesn't get a chance to recieve landmark based distance,We don't see a specific drop in localization performance

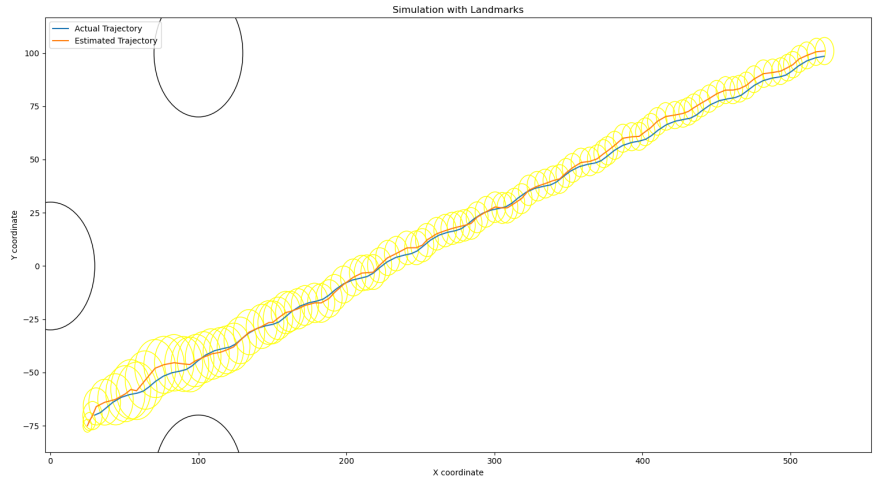


Figure 14: Std Deviation=5

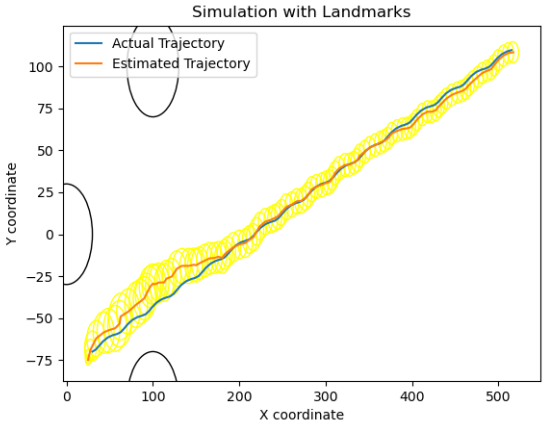


Figure 15: Std Deviation=10

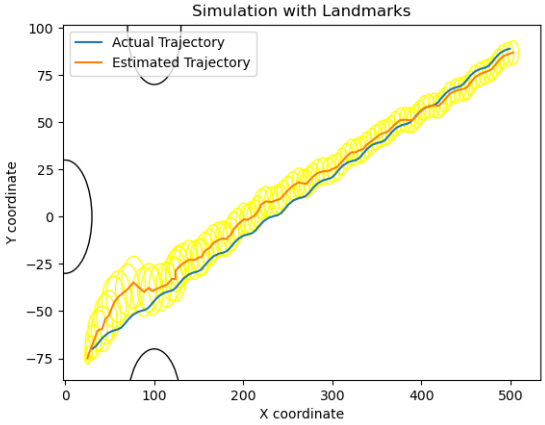
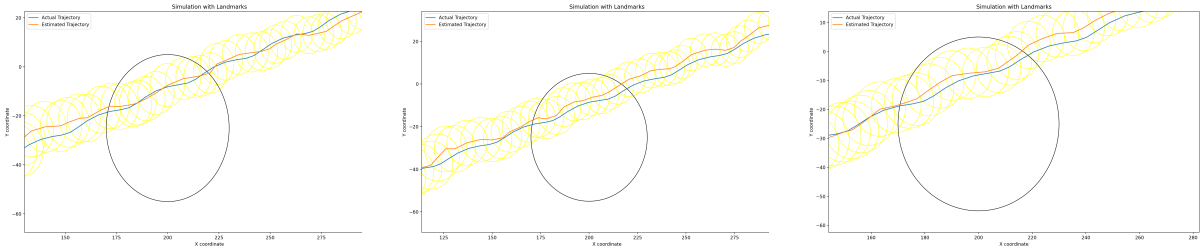


Figure 16: Std Deviation=20

Now to see how actually changes in standard deviation affect localization when landmark is present as standard deviation increases estimated path becomes far and ellipse expands.



(a) Std Deviation=1 (b) Std Deviation=10 (c) Std Deviation=20

Figure 17: Beliefs at different time steps

2.5 (e)Effect of Additional Landmark

We see that inside the landmark since we receive additional information,the uncertainty ellipse shrinks and localization performance improves

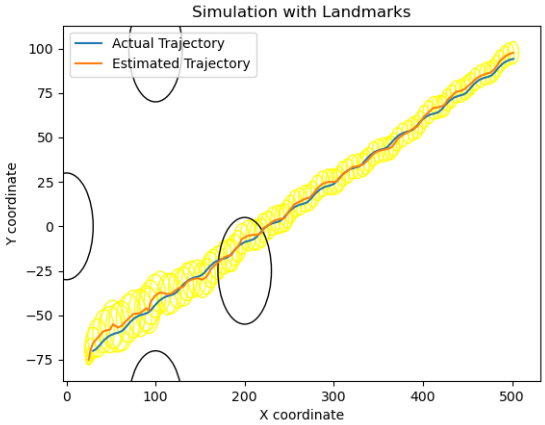


Figure 18: Extra landmark

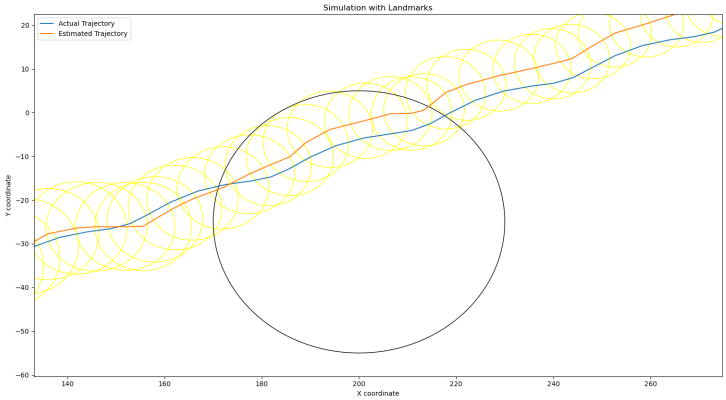


Figure 19: Zoomed to see effect of landmark

3 Question 3 : Robot Localization

3.1 (a)Simulation of Robot Motion

In this simulation we have robot agent which can give observations and updates its state randomly as given in the question.The grid used has 20 randomly placed obstacles and the robot starts from state (3,6). The grid can be seen below

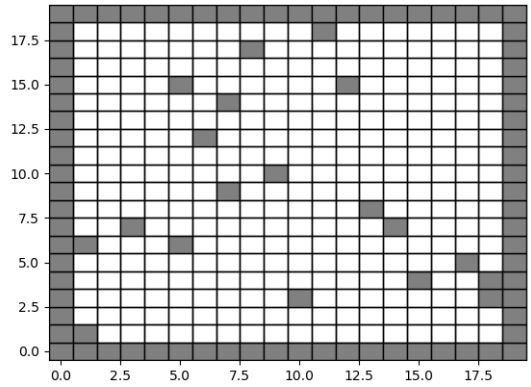


Figure 20: Maze

After simulation we observe the path taken by the agent is

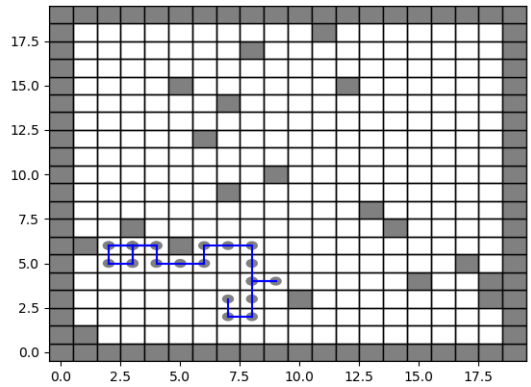


Figure 21: Actual Path

Actual state at the end was (7,3) A hidden markov model has been implemented and used to get state estimates at each time step. The estimated state was (9, 5) Along with that Viterbi algorithm is implemented to get the most likely path from observations recieved by agent

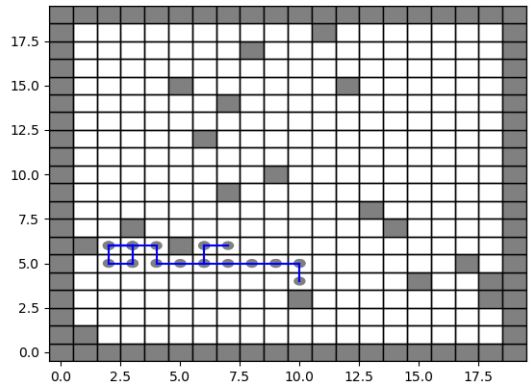
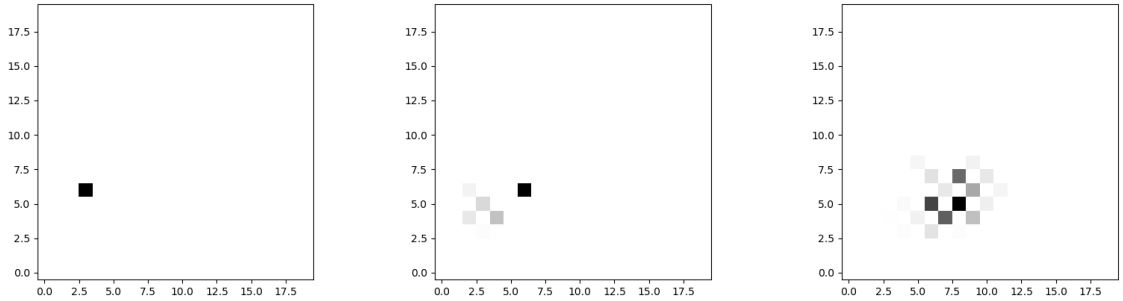


Figure 22: Estimated Path

3.2 (b)Belief Updates

The video showing beliefs is present here These are some frames at different timesteps showing log likelihood with probability of being at the state directly proportional to darkness.



(a) T=0 (b) T=15 (c) T=20

Figure 23: Beliefs at different time steps

3.3 (c)Trajectory Comparison

We stored path using state estimate and most likely path given by viterbi algorithm and compute manhattan distance between those paths and actual path followed by the agent

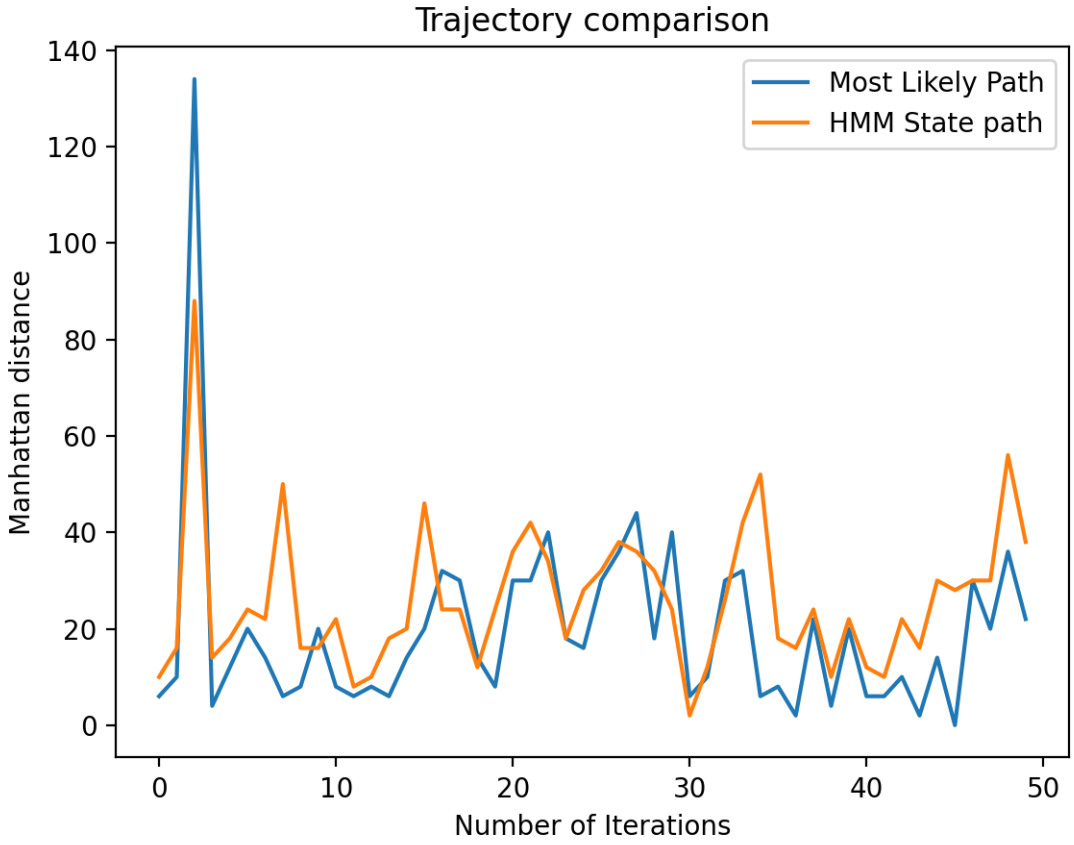
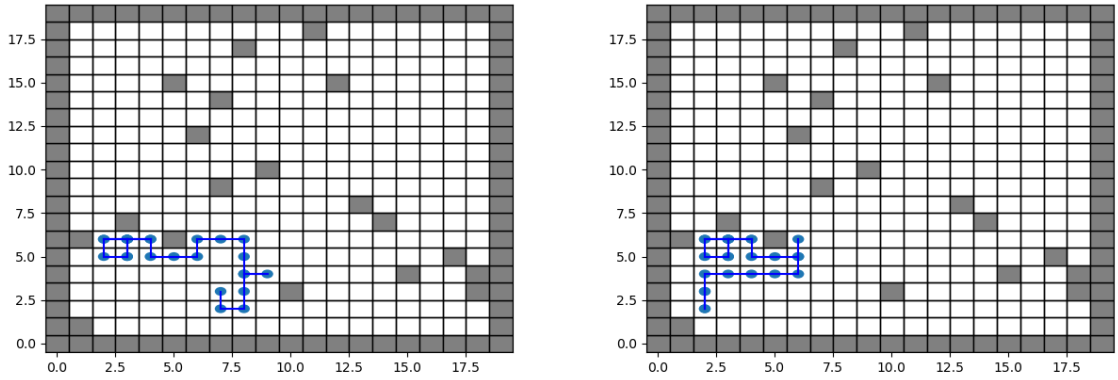


Figure 24: Trajectory Comparison with blue line denoting Most likely path and orange line showing HMM path

We observe that both most likely path and trajectory made by state estimate don't vary a lot from one another with most likey path having low mean manhattan distance of 19.36 and high variance of 402.7994 while path from state estimate has higher mean of 25.96 and lower variance of 222.47. Thus closest path to actual path is the one given by viterbi algorithm

3.4 (d)Changing Sensor Radius

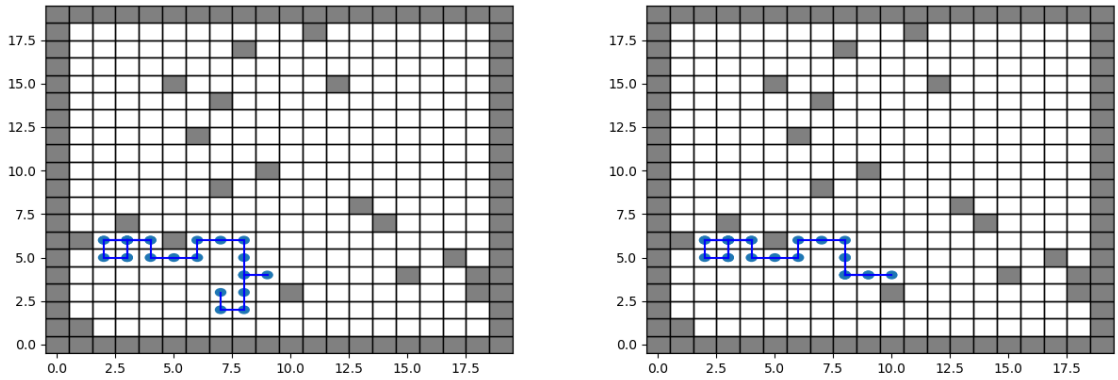
As we decrease our sensor accuracy by limiting radius to 1 what we can expect is decreased performance of HMM and Viterbi algorithms to predict states,paths and increased manhattan distance between the actual and estimated trajectories to 50 units .



(a) Actual Path (b) Estimated Path

Figure 25: $R_{max} = 1$

Now if we improve our sensor accuracy using $R_{max} = 10$ we observe that actual estimated trajectories are similar with reduced manhattan distance between them to 10 units.

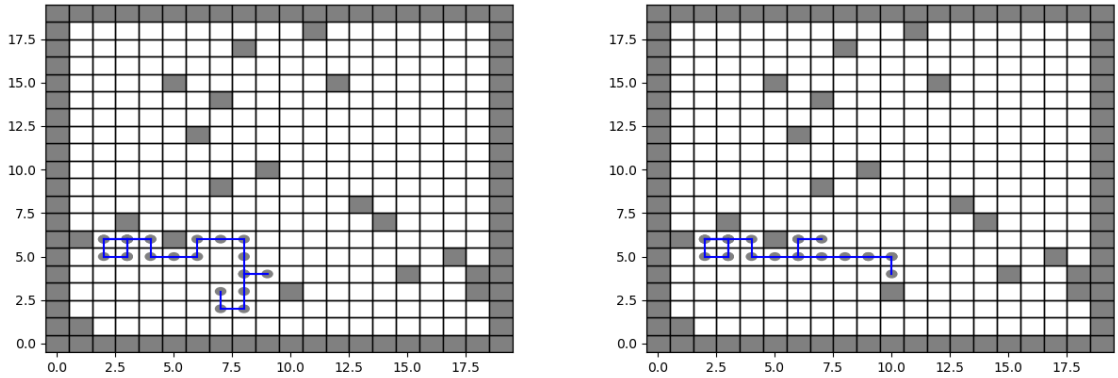


(a) Actual Path (b) Estimated Path

Figure 26: $R_{max} = 10$

3.5 (e)Grid Modification

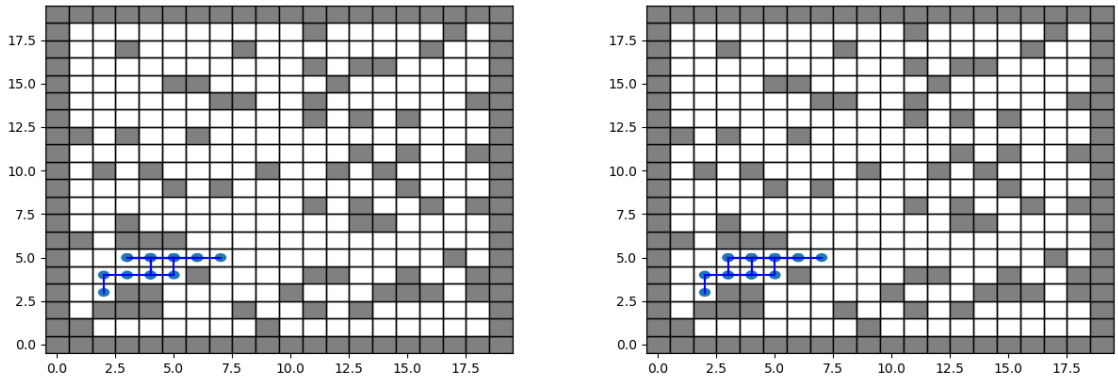
We can observe that it is difficult to do localization in grids with less number of obstacles as observations in different regions are going to be similar.Increasing number of obstacles can actually increase localization performance as observations which are linked to nearby obstacles can tell much more about location. For challenging localization we can refer to maze present in part 3.1



(a) Actual Path (b) Estimated Path

Figure 27: Challenging Localization

Increasing obstacles to around 90 in 400 cells sized grid box resulted in much better localization and path trajectory estimation by HMM and Viterbi Algorithm.



(a) Actual Path (b) Estimated Path

Figure 28: Easier Localization