COL 864: Planning and Estimation for Autonomous Systems **II Semester 2020-2021**

Assignment 1 Report

Students: Isha Chaudhary (2018EE30614), Sagar Sharma (2018CS10378)

Contents

1 *Q1: Robot Localization on Discrete State Space* **2**
1.1 *a: Simulation of Robot Motion* . . . 2
1.2 *b: Filtering to estimate the beliefs of each of the grid positions at each time step* . . . 3
1.3 *c: Smoothing to estimate the beliefs of each of the grid positions at each time step* . . . 6
1.4 *d: Comparing Filtering and Smoothing with the Ground Truth Path using the Manhattan Distance Error Metric* . . . 9
1.5 e: Predictive Likelihood . . . 11
1.6 *f: Most Likely Path* . . . 15

2 *Q2: Kalman Filter and Continuous Space State Estimation* **18**
2.1 *a: Motion Model and Sensor model* . . . 18
2.2 *b: Kalman Filter for estimation* . . . 19
2.3 *c: Plotting Estimate and Error Ellipses* . . . 20
2.4 *d: Sinusoidal Control* . . . 21
2.5 *e: Variation in Uncertainty of Observation Model* . . . 22
2.6 *f: Variation in Initial Belief* . . . 25
2.7 *g: Dropping Observations at t=10 and t=30 for 10 time steps* . . . 26
2.8 *h: Velocity tracking* . . . 27
2.9 *i: Data Association* . . . 28

1 *Q1: Robot Localization on Discrete State Space*

1.1 *a: Simulation of Robot Motion*

- Simulation Time: 25 steps
- Output: Sensor observations.
- Description of output: The vector of observations at each time step, which is $[z_1, z_2, z_3, z_4]$, comprises of the observation produced by the sensors at (8, 15), (22, 15), (15, 15), (15, 22) respectively at each time step. Each sensor stochastically reports the presence/absence of the target.
- In this simulation, the initial position was chosen to be (10, 10).

Sensor Observations used in the next parts

[0, 1, 0, 0], [1, 1, 0, 0], [1, 1, 0, 0], [0, 1, 0, 0], [0, 1, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0], [1, 0, 1, 0], [0, 0, 0, 0]

Ground Truth Positions of the Target, used to compute error in the following predictive steps

(10, 10), (10, 11), (11, 11), (12, 11), (11, 11), (10, 11), (10, 10), (9, 10), (8, 10), (8, 9), (9, 9), (9, 10), (9, 9), (9, 8), (10, 8), (9, 8), (9, 9), (10, 9), (11, 9), (12, 9), (13, 9), (13, 10), (13, 11), (14, 11), (15, 11), (16, 11)

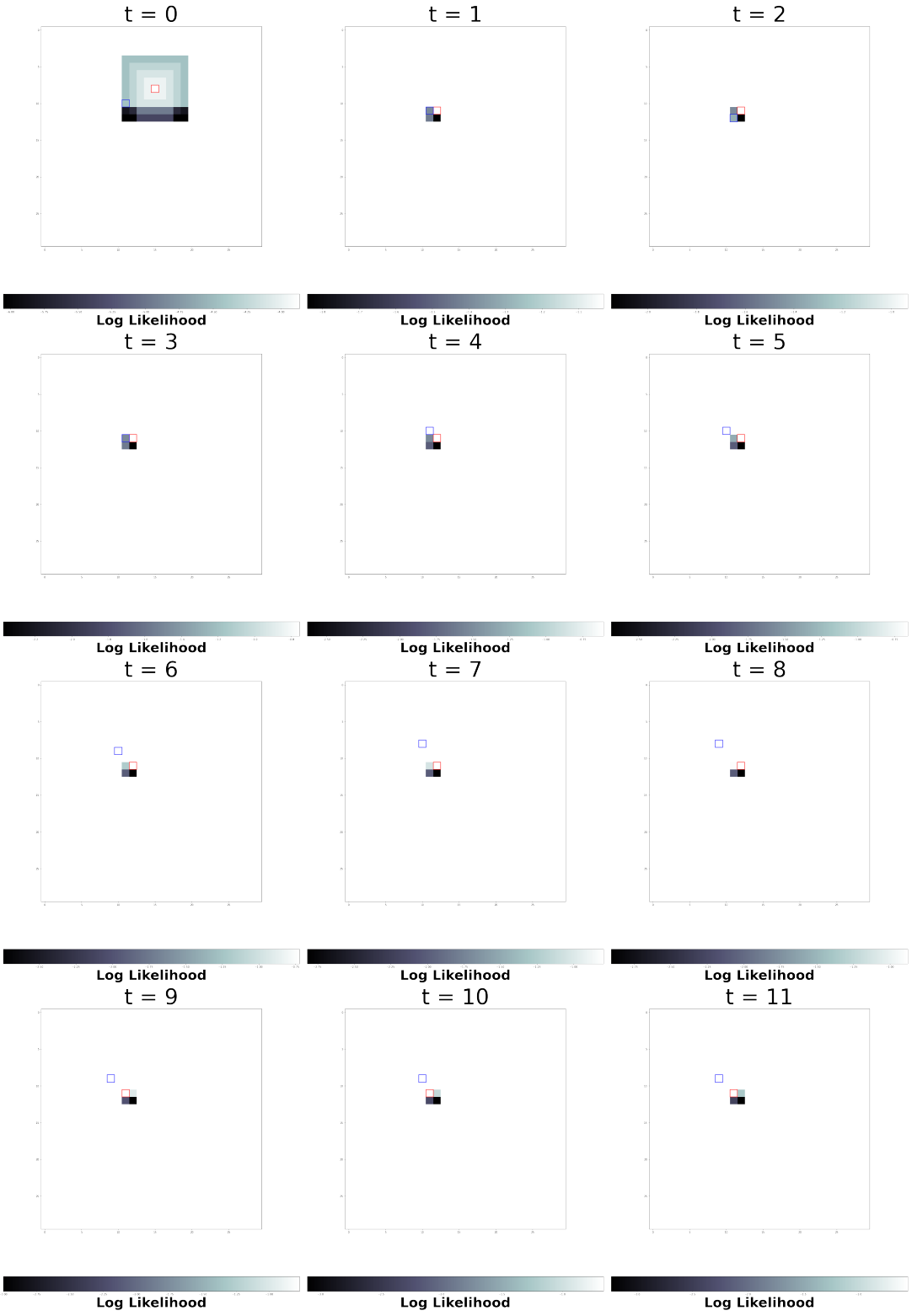
1.2 *b: Filtering to estimate the beliefs of each of the grid positions at each time step*

- Filtering Steps: 25
- Output: Log Likelihood plots (took the natural log of the belief values at each state), indicative of the Belief of the presence of the target over the various grid positions on the state space.
- The log likelihoods have been computed given the sensor observations till that time step and by using the transition model of the problem.
- The filtering operation was performed based on the following recursive operation:

$$P(\mathbf{X}_{t+1}|\mathbf{z}_{1:t+1}) = \eta P(\mathbf{Z}_{t+1}|\mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{X}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{z}_{1:t})$$

  where η is a normalizing constant.

- The transition model is as follows:
  1. $\mathrm{P}(\mathbf{X}_{t+1} = (a-1, b)|\mathbf{X}_t = (a, b)) = 0.4$ which corresponds to going *up* on the given grid.
  2. $\mathrm{P}(\mathbf{X}_{t+1} = (a+1, b)|\mathbf{X}_t = (a, b)) = 0.1$ which corresponds to going *down* on the given grid.
  3. $\mathrm{P}(\mathbf{X}_{t+1} = (a, b+1)|\mathbf{X}_t = (a, b)) = 0.3$ which corresponds to going *right* on the given grid.
  4. $\mathrm{P}(\mathbf{X}_{t+1} = (a, b-1)|\mathbf{X}_t = (a, b)) = 0.2$ which corresponds to going *left* on the given grid.
- The initial prior corresponded to a uniform distribution.
- The Log Likelihood observations are given in the following Fig 1.
- **Estimated and Ground Locations**: The Estimated locations are the ones which have the maximum likelihood over all the states in the state space. The estimated positions at each time step are marked with *red* boxes in the figure. The Ground truth positions at each time step, obtained from the simulation are marked with *blue* boxes at each time step.
- The time steps starting from zero, start from the instant of getting the first observation.


t = 0
t = 1
t = 2
Log Likelihood
Log Likelihood
Log Likelihood
t = 3
t = 4
t = 5
Log Likelihood
Log Likelihood
Log Likelihood
t = 6
t = 7
t = 8
Log Likelihood
Log Likelihood
Log Likelihood
t = 9
t = 10
t = 11
Log Likelihood
Log Likelihood
Log Likelihood

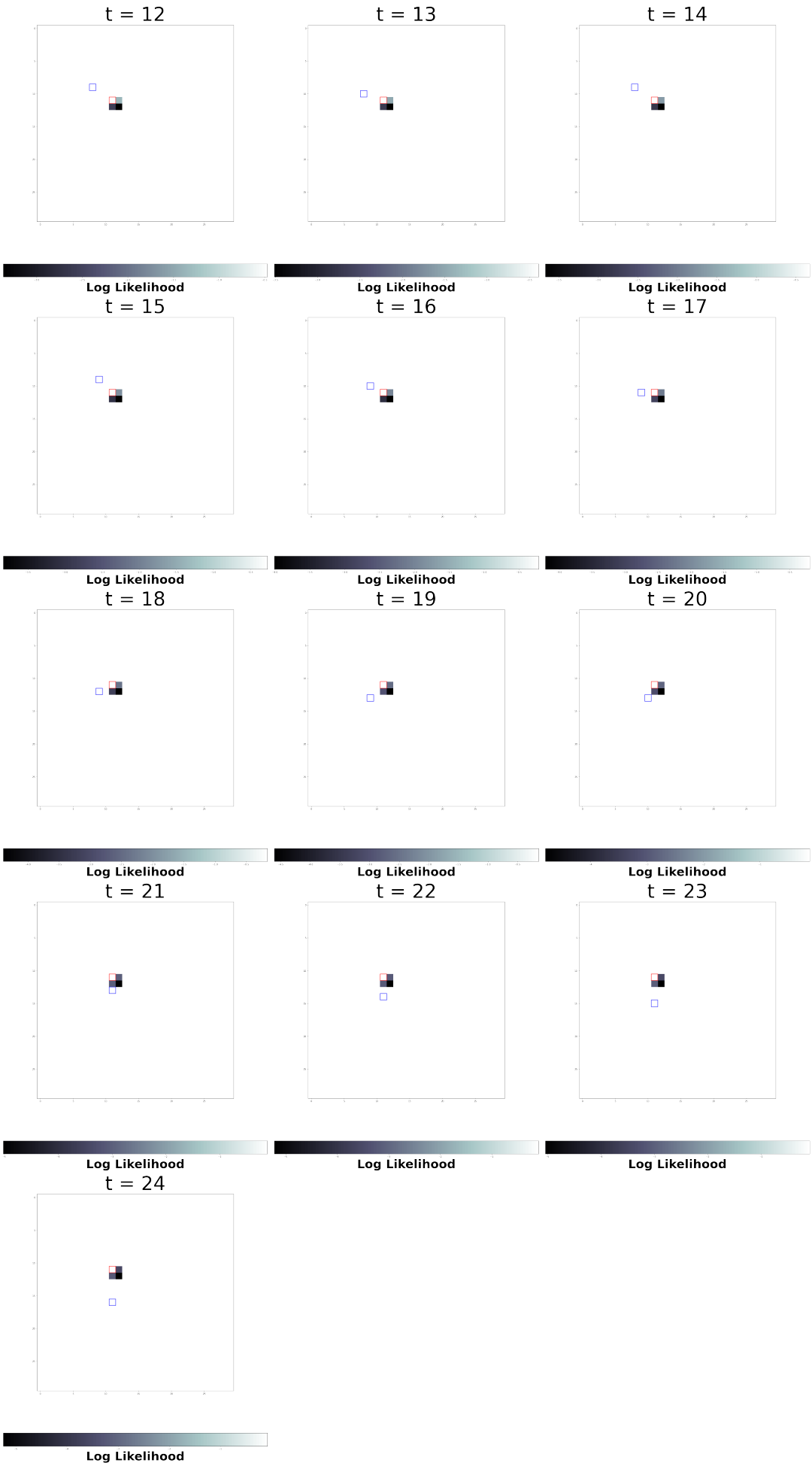


Figure 1: Variation of Log likelihood of Belief maps over time in Filtering Operation (Red: Estimated positions and Blue: Ground Truth positions)

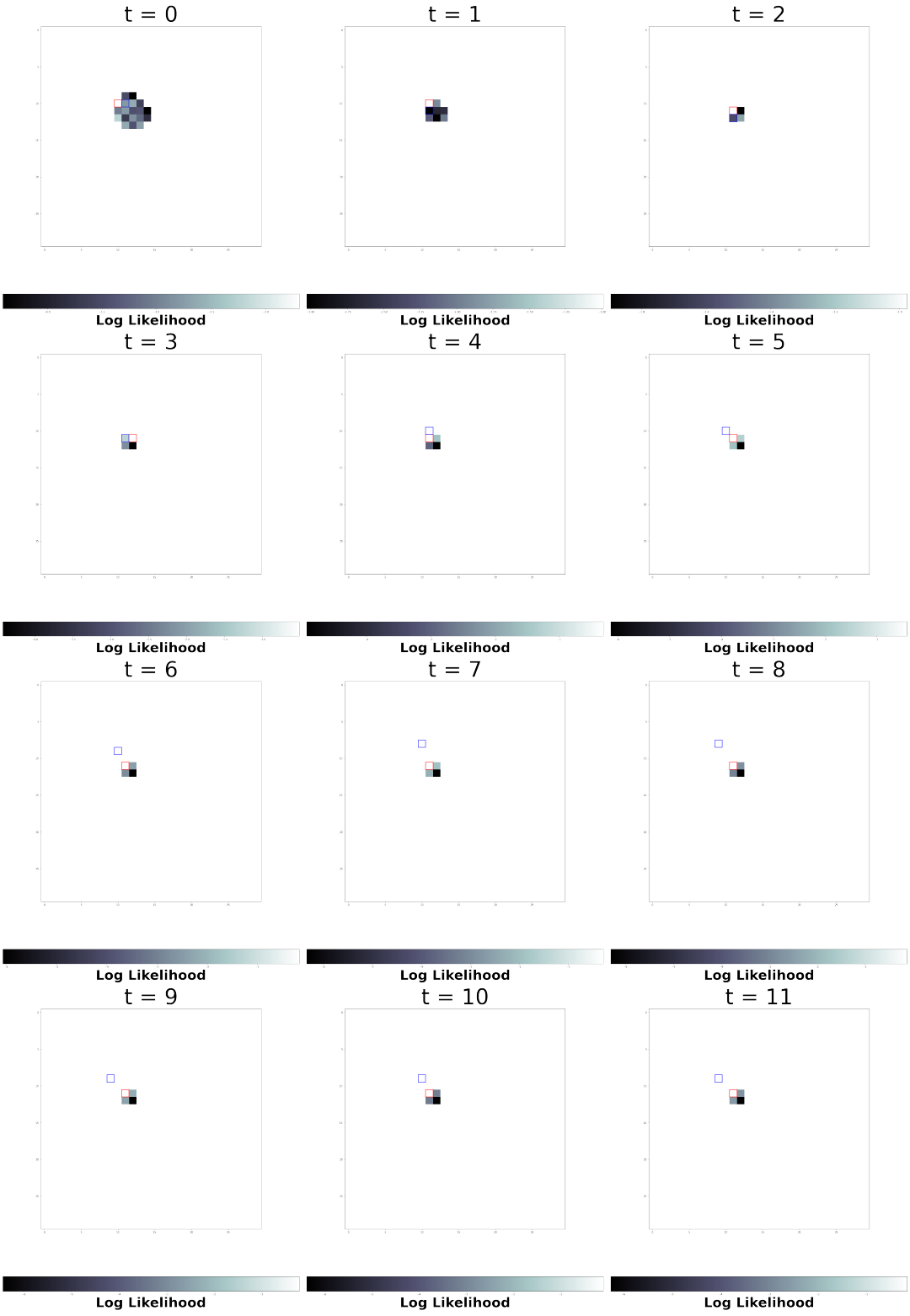
1.3 *c: Smoothing to estimate the beliefs of each of the grid positions at each time step*

- Smoothing Steps: 25
- Output: The log likelihood of all states over 25 time steps was recorded. It represents the beliefs of the presence of the target robot each of states in the grid state space. The belief values have been derived considering all of the sensor observations available to the model (till 25 time steps) and the transition model. Thus, the likelihood maps obtained for each time step are after *smoothing* it by the future observations and making the predictions better.
- The smoothing operation performed here involves a forward pass and a backward procedure, to incorporate the future sensor observations.
- The basic equation governing the algorithm for the Smoothing operation is:

$$\begin{aligned} P(\mathbf{X}_k|\mathbf{z}_{1:t}) &= \alpha P(\mathbf{X}_k|\mathbf{z}_{1:k})P(\mathbf{z}_{k+1:t}|\mathbf{X}_k) \\ &= \alpha f_{1:k} b_{k+1:t} \end{aligned}$$

$$P(\mathbf{z}_{k+1:t}|\mathbf{X}_k) = \sum_{\mathbf{x}_{k+1}} P(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})P(\mathbf{z}_{k+2:t}|\mathbf{x}_{k+1})P(\mathbf{x}_{k+1}|\mathbf{X}_k)$$

- The computation for smoothing is seeded by uniform initial belief values for the forward filtering term in the smoothing probability and $b_{t+1:t} = \mathbf{1}$, which is a matrix of 1.
- The Log Likelihood observations are given in the following Fig 2.
- **Estimated and Ground Locations**: The Estimated locations are the ones which have the maximum likelihood over all the states in the state space. The estimated positions at each time step are marked with *red* boxes in the figure. The Ground truth positions at each time step, obtained from the simulation are marked with *blue* boxes at each time step.
- The time steps starting from zero, start from the instant of getting the first observation.


t = 0
t = 1
t = 2
Log Likelihood
Log Likelihood
Log Likelihood
t = 3
t = 4
t = 5
Log Likelihood
Log Likelihood
Log Likelihood
t = 6
t = 7
t = 8
Log Likelihood
Log Likelihood
Log Likelihood
t = 9
t = 10
t = 11
Log Likelihood
Log Likelihood
Log Likelihood

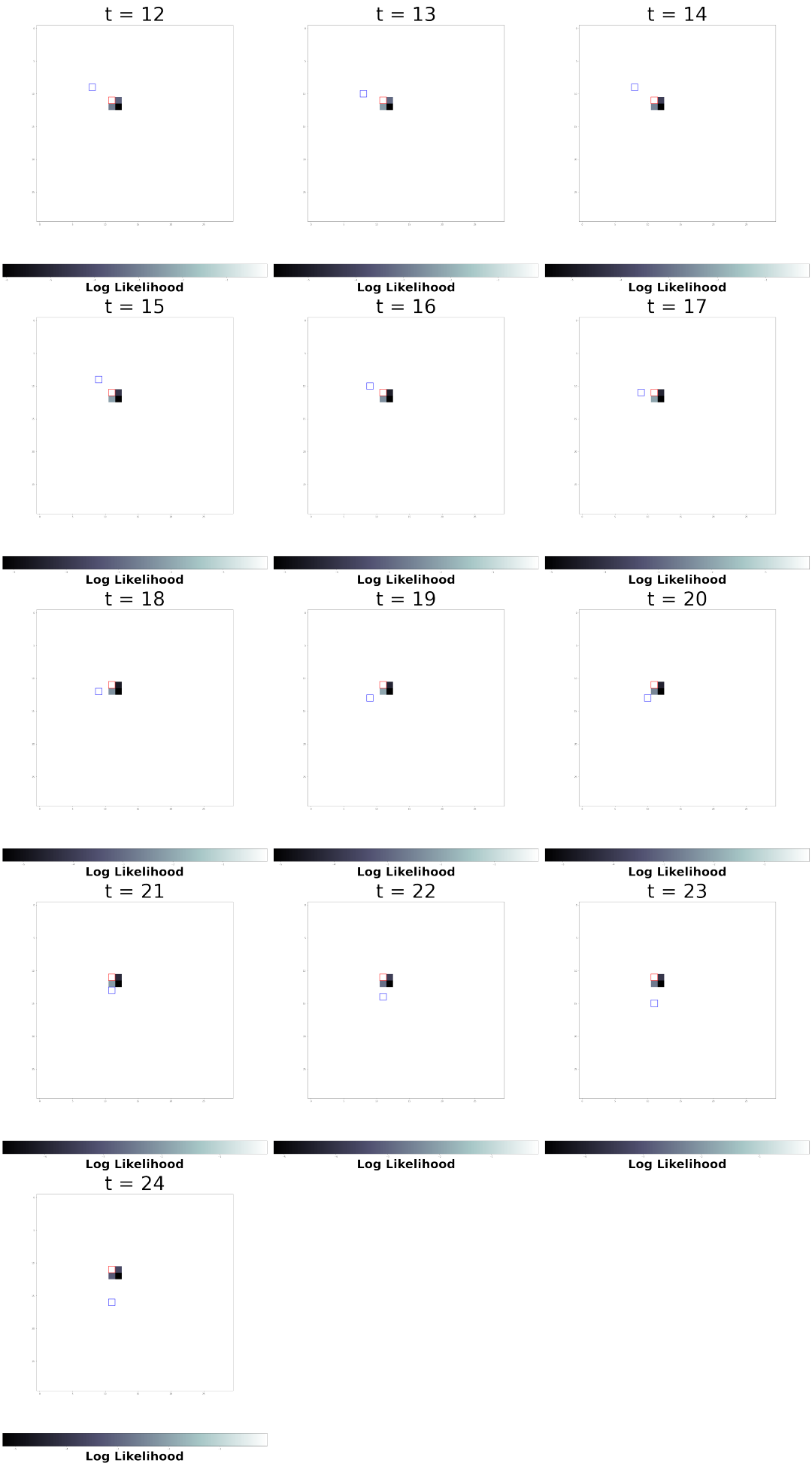


Figure 2: Variation of Log likelihood of Belief maps over time in Smoothing operation (Red: Estimated positions and Blue: Ground Truth positions)

1.4 *d: Comparing Filtering and Smoothing with the Ground Truth Path using the Manhattan Distance Error Metric*

- Number of Time Steps: 25
- Procedure: The Manhattan Error has been computed between the estimated and ground truth positions at all the time steps and displayed in Fig 3. The Manhattan Error has also been computed between the estimated and ground truth paths, which are generated by the Filtering and Smoothing Procedures till each time step. They are displayed in Fig 4.
- The estimated positions are the states which have the maximum likelihood over the state space.
- The plots indicate the following:

  1. The Path error plot clearly indicates that the *Filtering* procedure gives higher path error than Smoothing procedure at each time step.
  2. This is because, Filtering takes into account only the observations which were generated till the time step of estimating the state beliefs. Smoothing, on the other hand, considers all the sensor observations available. Thus, it is expected that Smoothing will produce better results and outperform Filtering.
  3. When high accuracy of the prediction is the main objective, then Smoothing should be used. In other cases, where the time taken by the algorithm is of concern, Filtering is preferred.
  4. The Manhattan error at each time step, between the estimated and ground truth positions indicates that due to Smoothing operation, the error in the initial positions of the trajectory falls drastically. This is because, Smoothing is using the entire set of Sensor observations to generate the beliefs for the states in the initial time steps.
  5. The Manhattan error is same for Smoothing and Filtering operations both at the last time step, as both the models are using the entire set of sensor observations to compute the likelihoods.
  6. The error due to Smoothing is seen to be lesser than or equal to that produced by filtering at each time step.

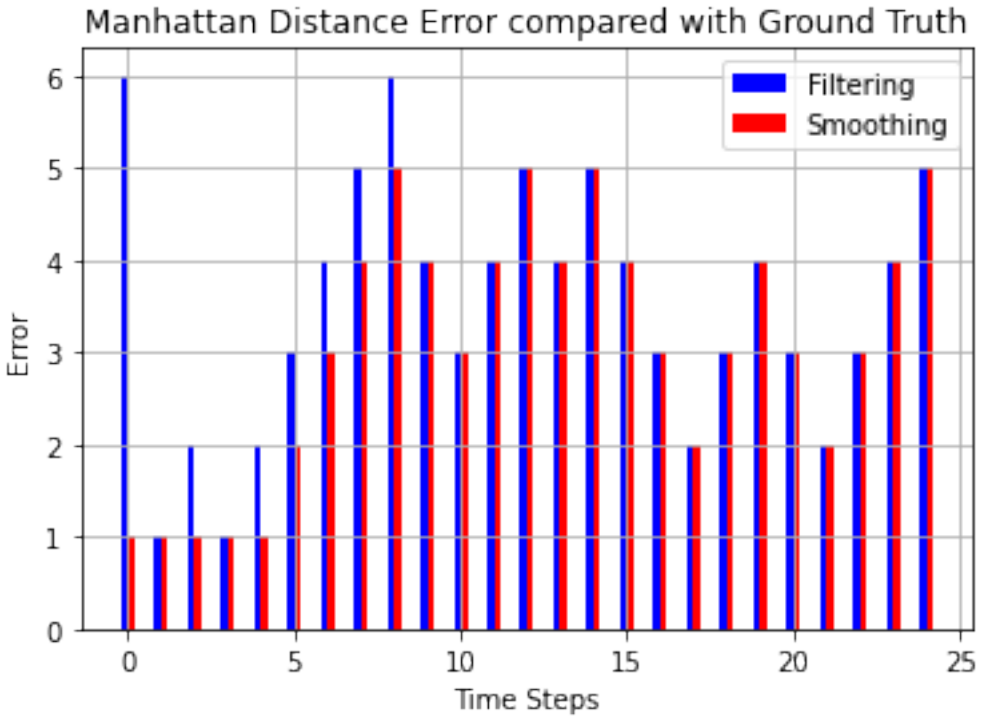


Figure 3: Manhattan error obtained in Filtering and Smoothing Steps

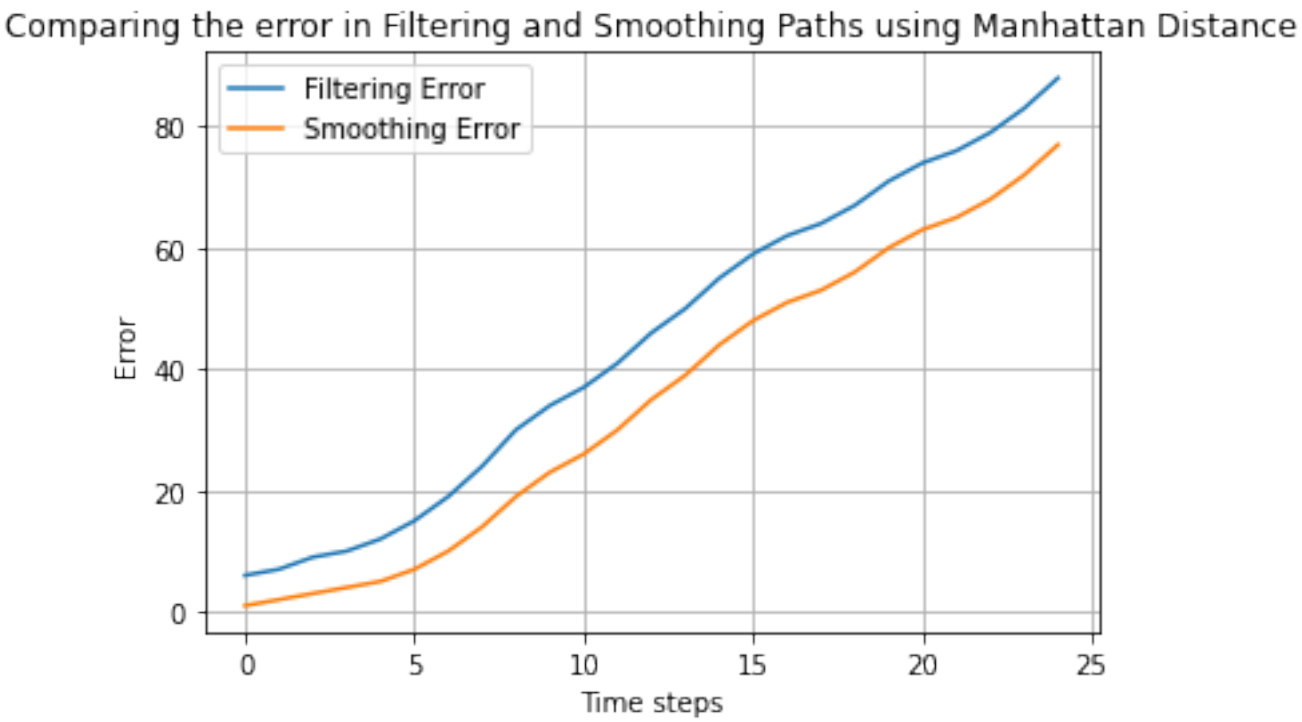


Figure 4: Manhattan error of Filtering and Smoothing Paths

1.5 e: Predictive Likelihood

- Output: Log Likelihood plots (took the natural log of the belief values at each state), indicative of the Belief of the presence of the target over the various grid positions on the state space.
- The log likelihoods have been computed using the transition model of the problem only. This is because no observations exist for future states.
- The Prediction operation was performed based on the following recursive operation:

$$P(X_{t+k+1}|z_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1}|x_{t+k})P(x_{t+k}|z_{1:t})$$

- The transition model is as follows:
  1. $\mathrm{P}(X_{t+1} = (a-1, b)|X_t = (a, b)) = 0.4$ which corresponds to going *up* on the given grid.
  2. $\mathrm{P}(X_{t+1} = (a+1, b)|X_t = (a, b)) = 0.1$ which corresponds to going *down* on the given grid.
  3. $\mathrm{P}(X_{t+1} = (a, b+1)|X_t = (a, b)) = 0.3$ which corresponds to going *right* on the given grid.
  4. $\mathrm{P}(X_{t+1} = (a, b-1)|X_t = (a, b)) = 0.2$ which corresponds to going *left* on the given grid.
- The initial prior for the prediction operation is the belief matrix generated at the end of the filtering procedure on the same problem.
- The Log Likelihood observations are given in the following figure.
  1. **With 10 steps of Future** Prediction Steps: 10: Fig 5
  2. **With 25 steps of Future** Prediction Steps: 25: Fig 6
- **Estimated Locations**: The Estimated locations are the ones which have the maximum likelihood over all the states in the state space. The estimated positions at each time step are marked with *red* boxes in the figure.
- The time steps starting from zero, start from the first instant of entering into the future, that is the first time instant from which we have no sensor observations, in both the figures.

Discussion

- The plots help in verifying the fact that the applying the transition model adds to the uncertainty in the likelihood of the presence of the target.
- Sensor Observations decrease the uncertainty in the position of the target as is clear from the Filtering and Smoothing Observations.
- As it can be seen, the likelihood of each state being occupied with the target, rises by increasing the number of steps into the future. The plots with 10 steps of progress indicate a lower likelihood of the occupancy of a state with the target than the plots with 25 steps of progress.
- The randomness in the position of the target thus rises.

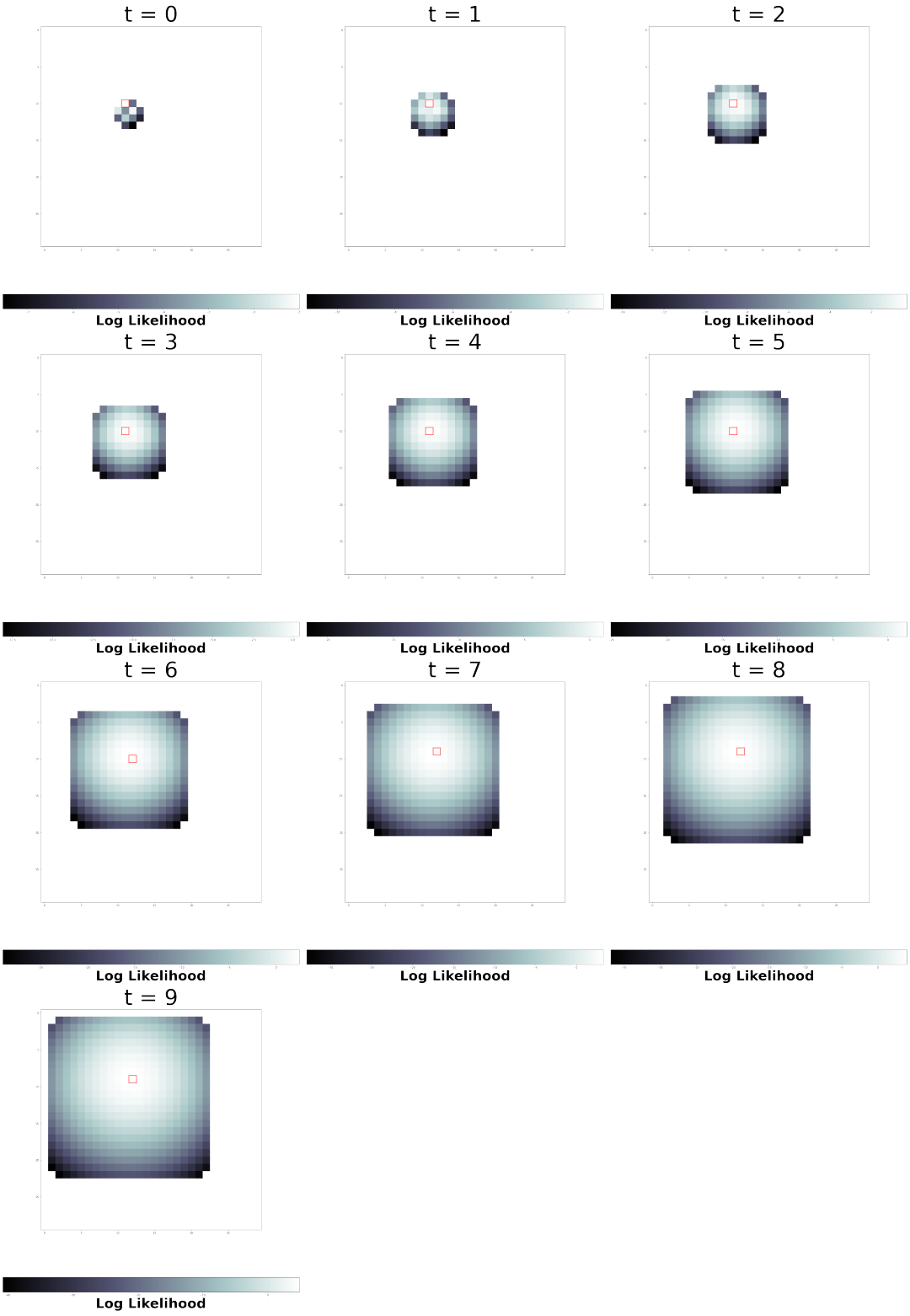
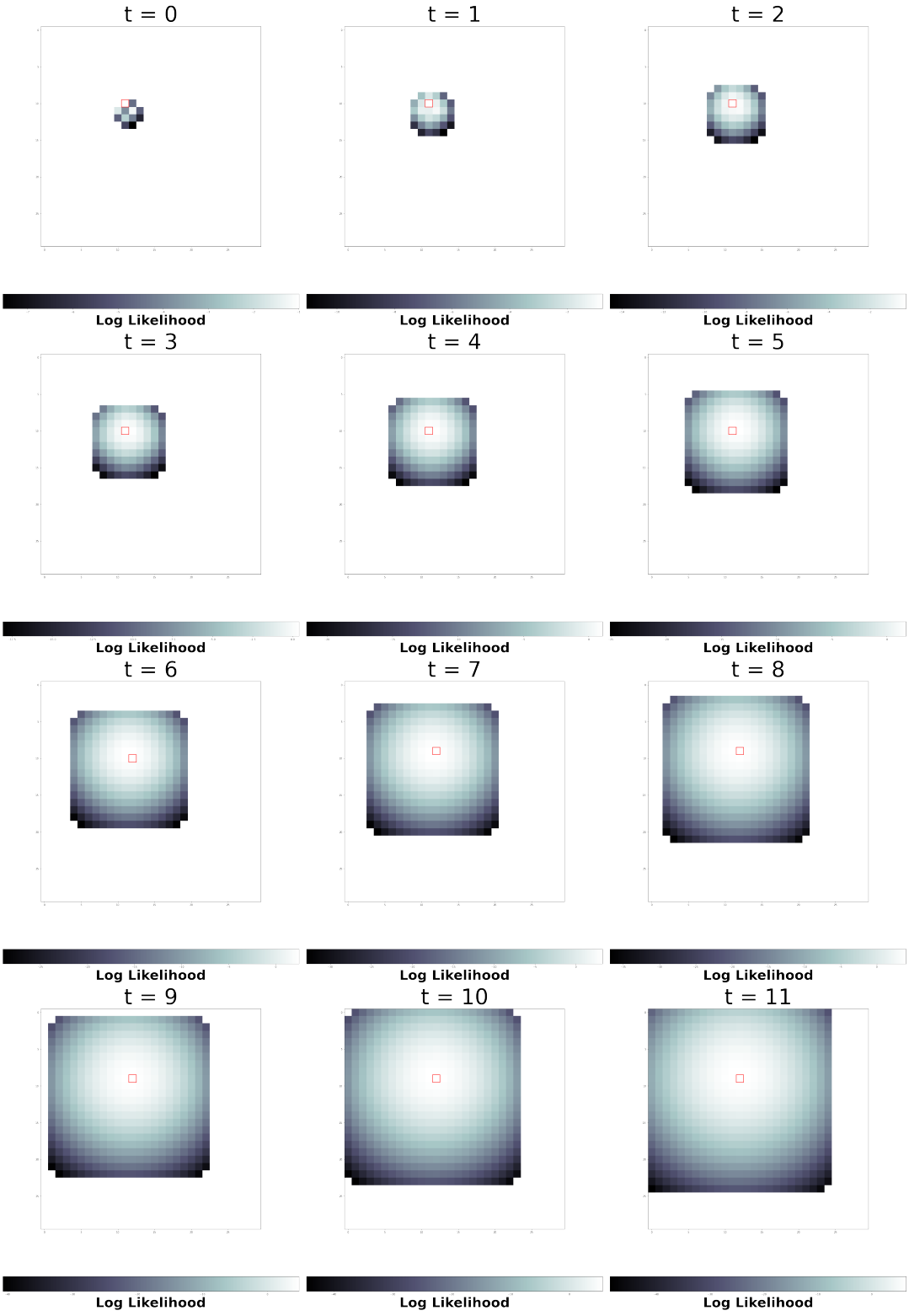


Figure 5: Predictive Likelihood for 10 steps (Red: Estimated positions


t = 0
t = 1
t = 2
Log Likelihood
Log Likelihood
Log Likelihood
t = 3
t = 4
t = 5
Log Likelihood
Log Likelihood
Log Likelihood
t = 6
t = 7
t = 8
Log Likelihood
Log Likelihood
Log Likelihood
t = 9
t = 10
t = 11
Log Likelihood
Log Likelihood
Log Likelihood

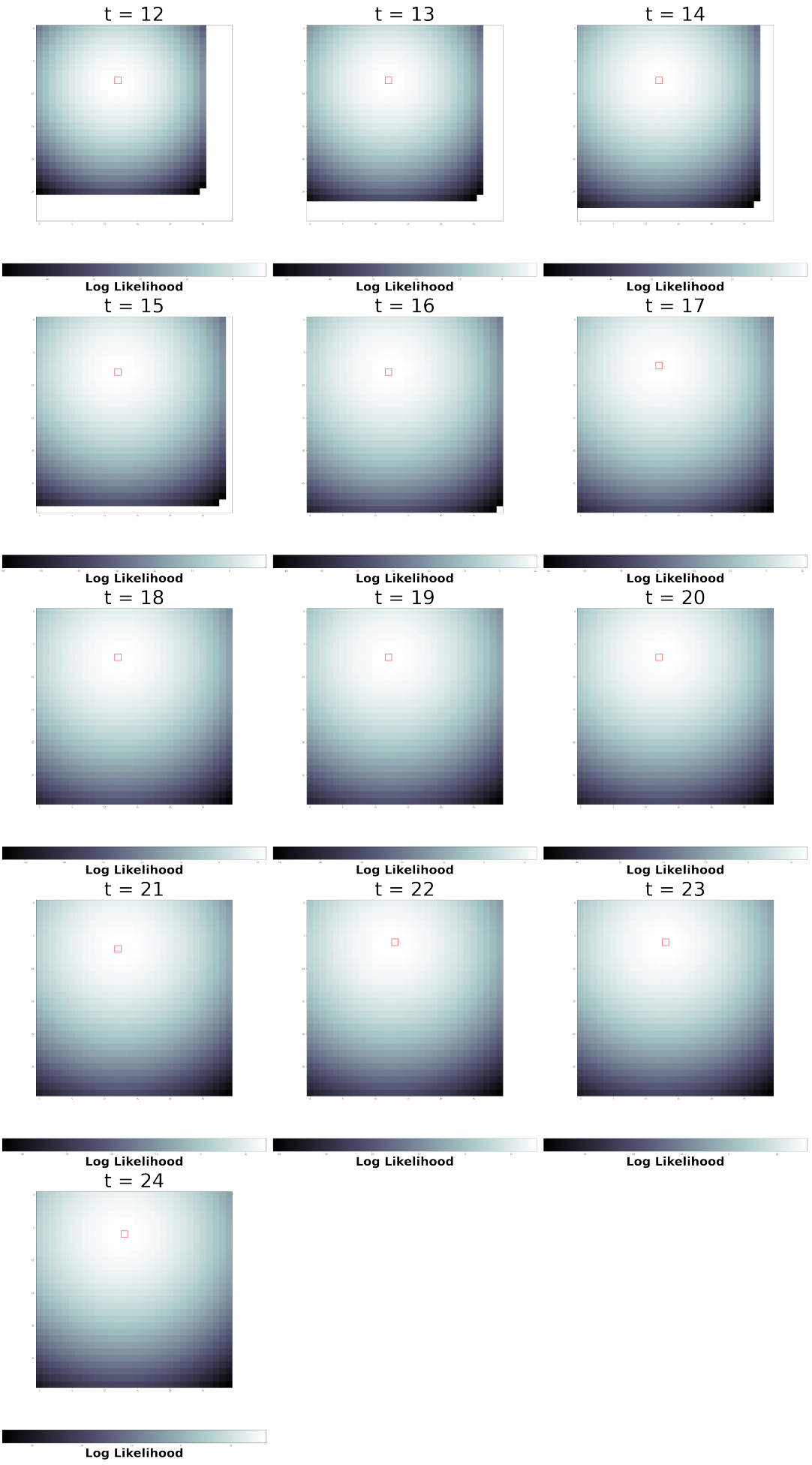


Figure 6: Predictive Likelihood over 25 steps (Red: Estimated positions)

1.6 *f: Most Likely Path*

- The most likely path for the given sensor observations and transition model is determined using the Viterbi Algorithm.
- The recursive step in the algorithm uses the following relation:

$$\max_{\mathbf{x}_1,...,\mathbf{x}_t} P(\mathbf{x}_1,...,\mathbf{x}_t,\mathbf{X}_{t+1}|\mathbf{z}_{1:t+1}) = \alpha P(\mathbf{z}_{t+1}|\mathbf{X}_{t+1}) \max_{\mathbf{x}_t}(P(\mathbf{X}_{t+1}|\mathbf{x}_t) \max_{\mathbf{x}_1,...,\mathbf{x}_{t-1}} P(\mathbf{x}_1,...,\mathbf{x}_t|\mathbf{z}_{1:t}))$$

- The initial prior involved in the algorithm corresponded to a uniform distribution.
- The Estimated and Ground truth locations over all the time steps till T = 25 are shown in the following Fig 7
- **Estimated and Ground Locations**: The Estimated locations are the ones which have the maximum likelihood path over all the paths possible, till their time step. The estimated positions at each time step are marked with *red* boxes in the figure. The Ground truth positions at each time step, obtained from the simulation are marked with *blue* boxes at each time step.
- The time steps starting from zero, start from the instant of getting the first observation.


t = 0
t = 1
t = 2
t = 3
t = 4
t = 5
t = 6
t = 7
t = 8
t = 9
t = 10
t = 11

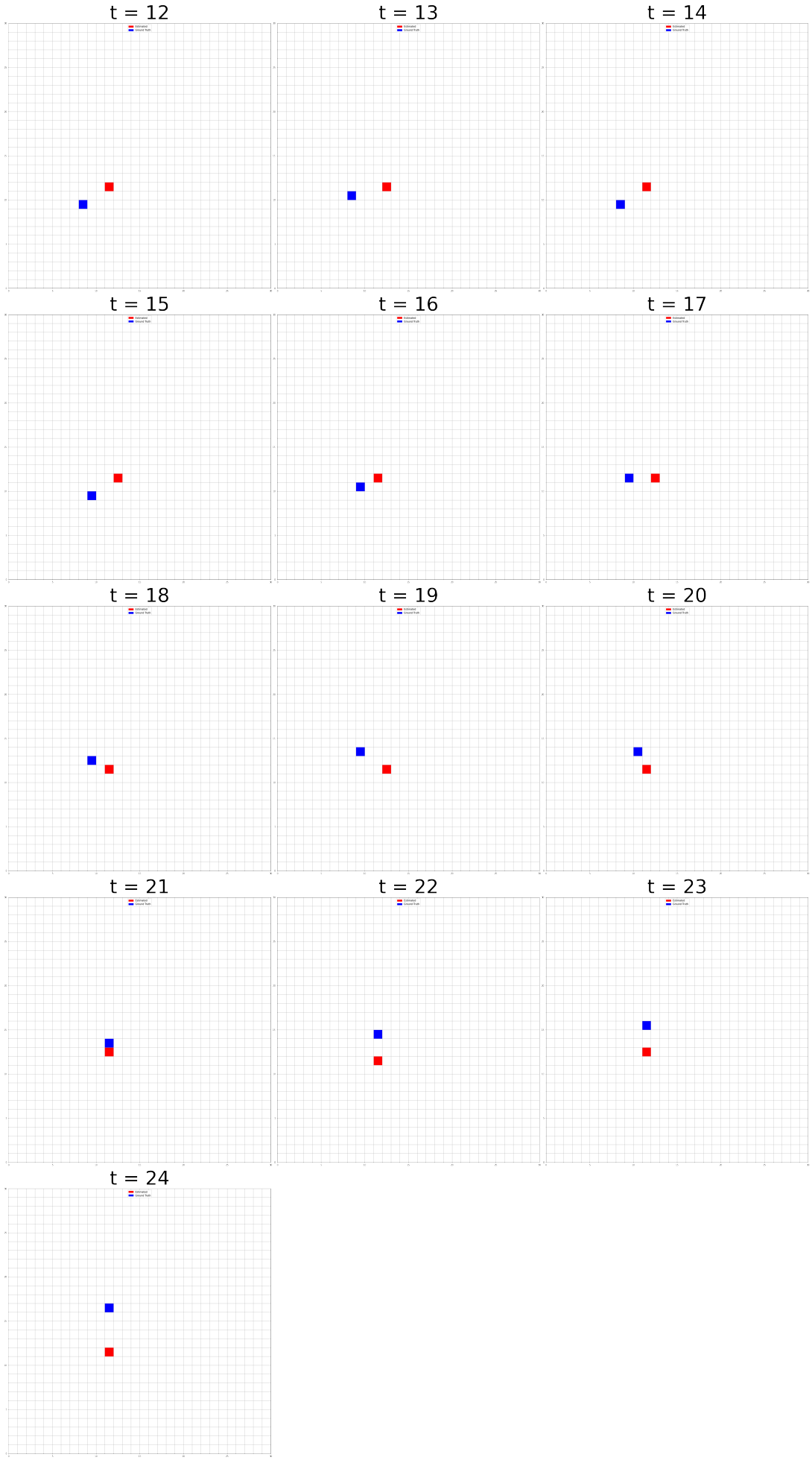


Figure 7: Most Likely Path (Red: Estimated positions and Blue: Ground Truth positions)

2 *Q2: Kalman Filter and Continuous Space State Estimation*

2.1 *a: Motion Model and Sensor model*

Motion Model

$$X_{t+1} = AX_t + Bu_t + \epsilon \sim \mathcal{N}(0,R)$$

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10^{-4} & 0 \\ 0 & 0 & 0 & 10^{-4} \end{bmatrix}$$

Thus if $X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix}$ then $X_{t+1} = \begin{bmatrix} x_t + \dot{x}_t \\ y_t + \dot{y}_t \\ \dot{x}_t + \delta_x \\ \dot{y}_t + \delta_y \end{bmatrix}$

SENSOR MODEL

$$Z_t = CX_t + \delta \sim \mathcal{N}(0,Q)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$$

Thus if $X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix}$ then $Z_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \delta$

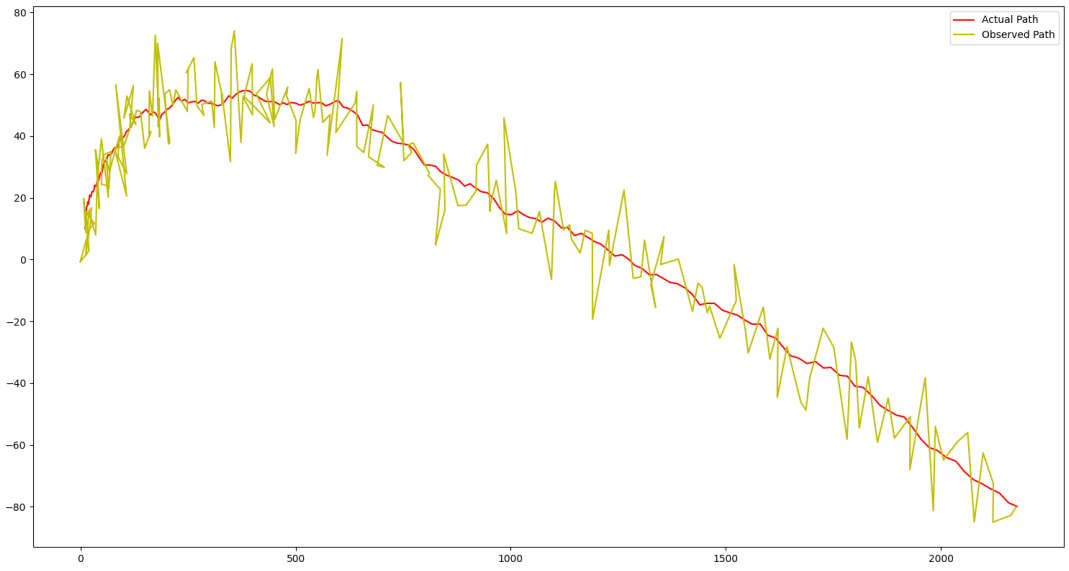


Figure 8: Actual Trajectory and Observed Path

NOTE: Here the initial state is [[10],[10],[1],[1]] and constant control u = [[0.1],[-0.015]]

We generate these simulations by assuming some X_0 and then calculate X_1 and so on. Similarly for each X_t we generate observation according to above equation.

2.2 *b: Kalman Filter for estimation*

The model equations are same as above.
We take initial μ_0 = [[-100],[-75],[0],[0]] and $\Sigma_0 = 10^{-4}$ * identity matrix
The function for estimation is

```
def kalmanFilter(mu,sigma,u,z,A,B,R,C,Q):
- Passage of time step
- mu1 = np.matmul(A,mu)+np.matmul(B,u)
- sigma1 = np.matmul(np.matmul(A,sigma),np.transpose(A))+R

- Evidence Incorporation Step
- k = np.matmul(np.matmul(sigma1,np.transpose(C)),
      - np.linalg.inv(np.matmul(np.matmul(C,sigma1),np.transpose(C))+Q))

- mu2 = mu1 + np.matmul(k,(z-np.matmul(C,mu1)))
- temp = np.matmul(k,C)
- l1,l2 = np.shape(temp)
- sigma2 = np.matmul((np.identity(l1)-temp), sigma1)
- return mu2,sigma2
```

We apply kalman filter to the observations according to algorithm above.
We use same A,B,R,C,Q from last part. The results plotted in next part.

2.3 *c: Plotting Estimate and Error Ellipses*

For the above equations we get

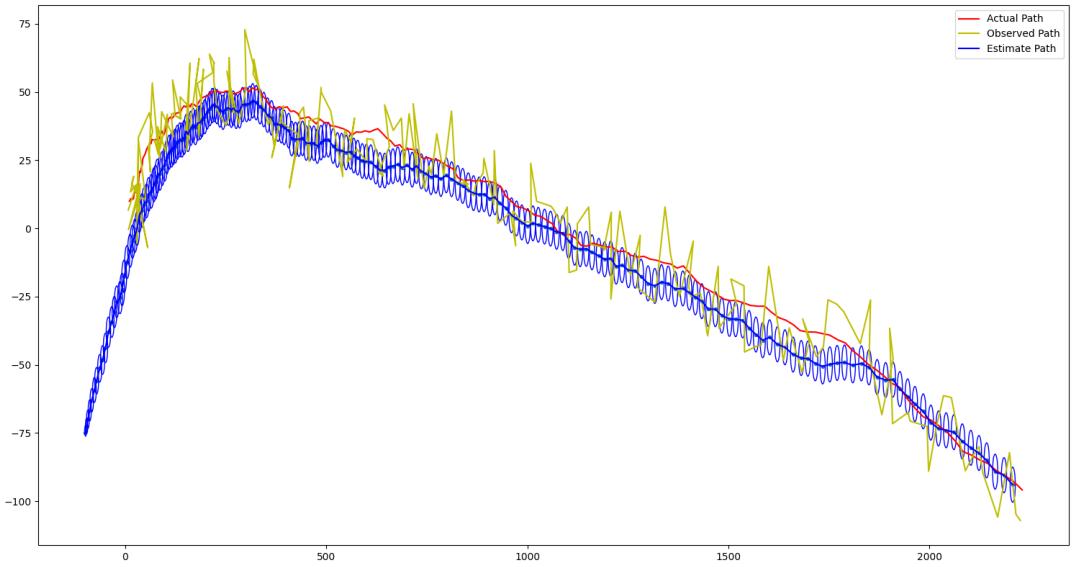


Figure 9: Actual Path(red) and Observed Path(yellow) and Estimate Path(blue) with Error Ellipse

OBSERVATIONS

- From the above plot we can observe that even though we start from different belief we start converging towards the actual trajectory.
- The size of ellipse is smaller(due to certain initial belief) in the beginning but later on it enlarges and saturates to a certain size. This saturation is determined by observation noise parameters.
- The actual trajectory converges toward the estimate and thus toward center of ellipses, but as there is noise it can converge completely.

2.4 *d: Sinusoidal Control*

The control is of the form u = [[sin(i.10)], [cos(i/10)]] $\forall i \in (0, 200)$
μ_0 = [[0],[0],[0],[0]] and rest everything is same as before

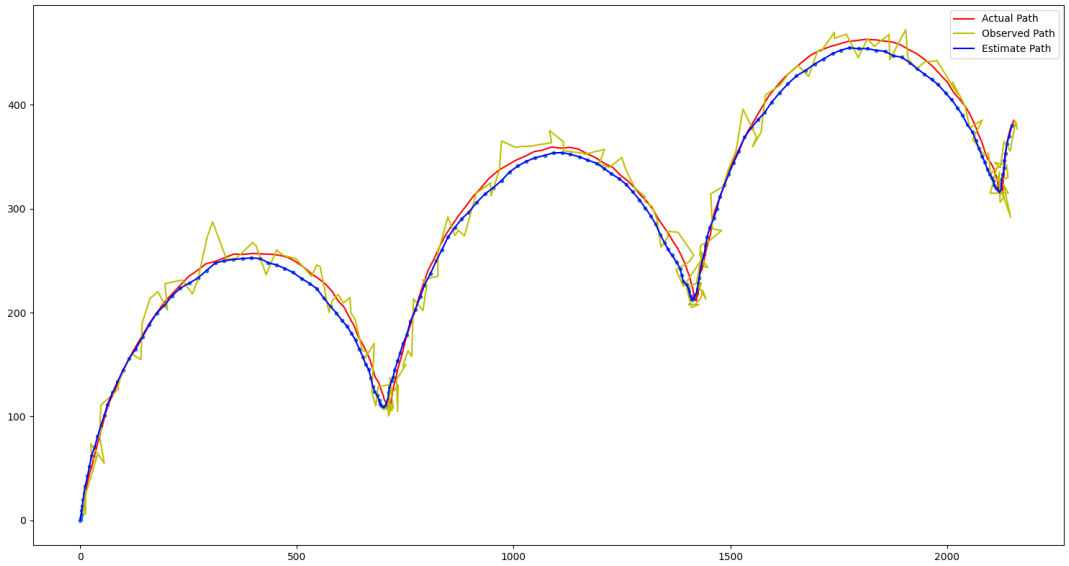


Figure 10: Actual Path(red) and Observed Path(yellow) and Estimate Path(blue) with Error Ellipse

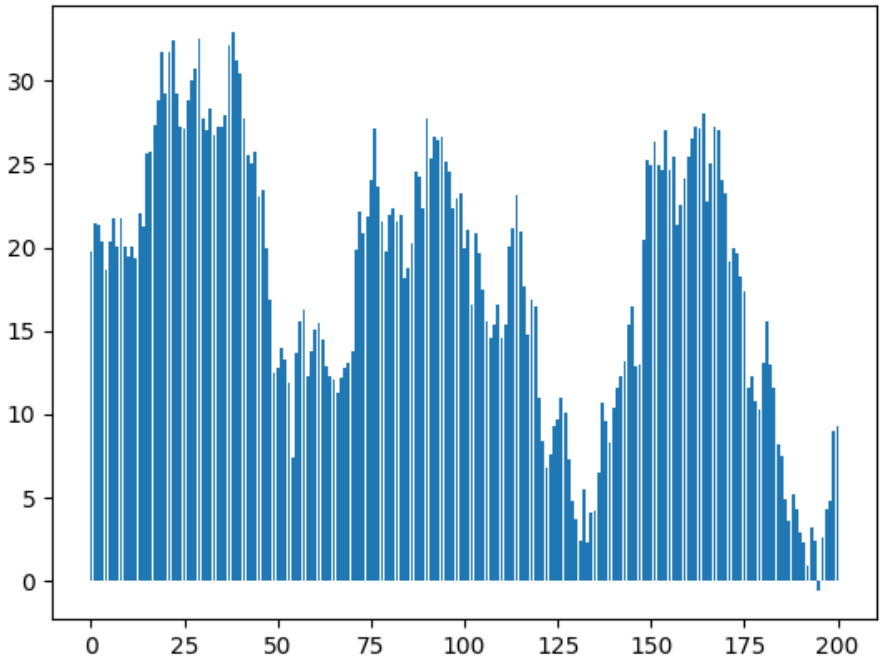


Figure 11: Euclidean Distance Error between true trajectory and estimated trajectory

Note: x denotes time steps, y denotes error

OBSERVATIONS

- The Estimate is very close to true trajectory, most parameters are same as before. We can observe that the considerable deviation always around the peak, same is observed in the euclidean distance error graph, the deviation peak is periodic and reaches maximum around peak. This happens because speed reach max around here and the displacements are large, leaving less time for correction, and generating more noise.
- Around the trough and ascent, the error is least and the convergence is max, because the displacement is least and thus more chances for convergence.

2.5 *e: Variation in Uncertainty of Observation Model*

We will see the variation for parabolic trajectory
same equations as part 2, and we will be varying variance in Q

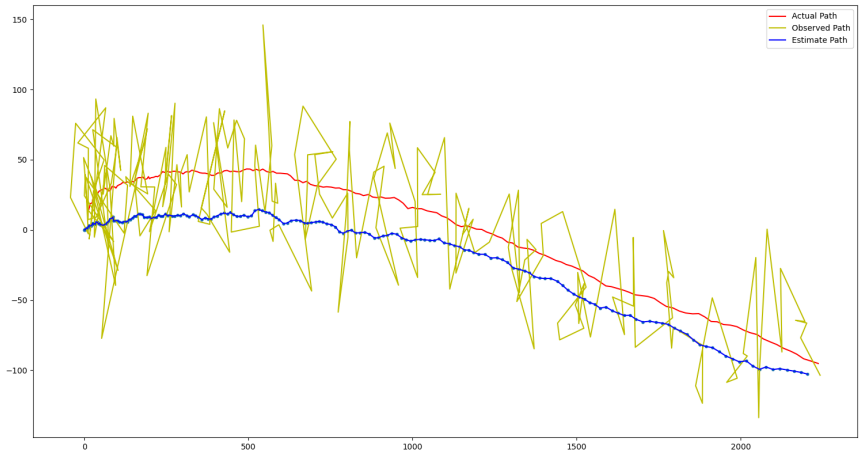


Figure 12: By increasing standard deviation in sensor model from 100 to 1000

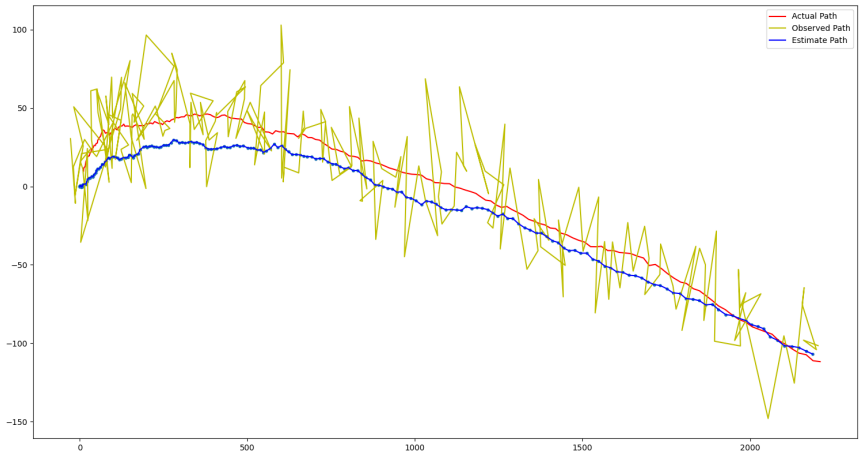


Figure 13: By increasing standard deviation in sensor model from 100 to 500

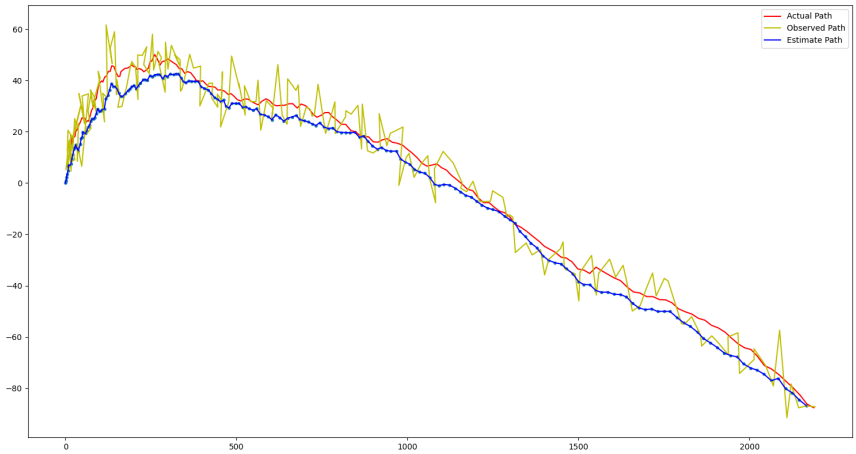


Figure 14: By decreasing standard deviation in sensor model from 100 to 50

OBSERVATIONS

- The observed trajectory shows large variations with increasing Q.
- The more the noise observation the more time it takes for convergence.
- In the beginning as velocities are small, displacement is small, therefore much of the convergence happens in initial 70-80 timesteps.

For Sinusoidal curves we get

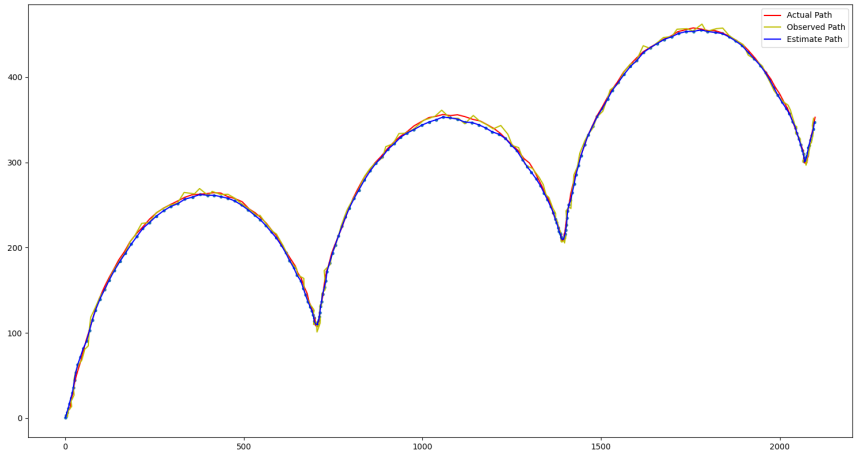


Figure 15: scaling Q by 0.1

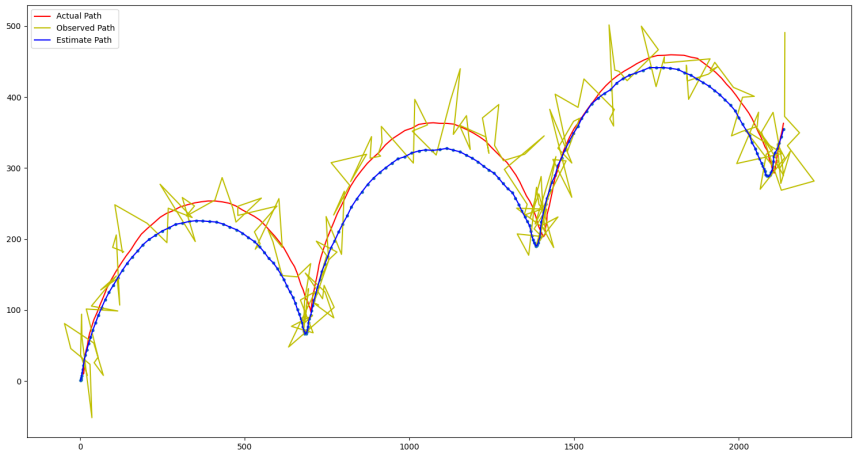


Figure 16: scaling Q by 10

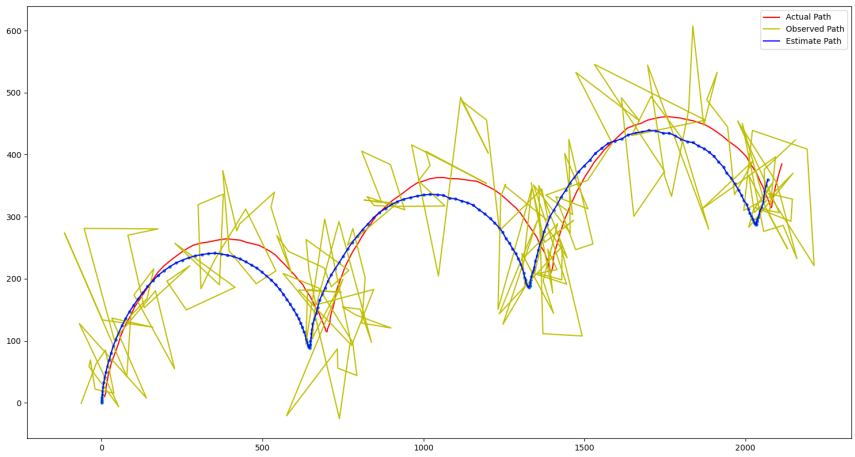


Figure 17: scaling Q by 50

OBSERVATIONS

- Again in Sinusoidal curve more variation in noise leads to more deviation from actual trajectory.
- Most deviation takes place around peaks, at sharp cuts the deviation reduces due to less displacement there.

2.6 *f: Variation in Initial Belief*

Assuming different initial pos and covariance matrix we get

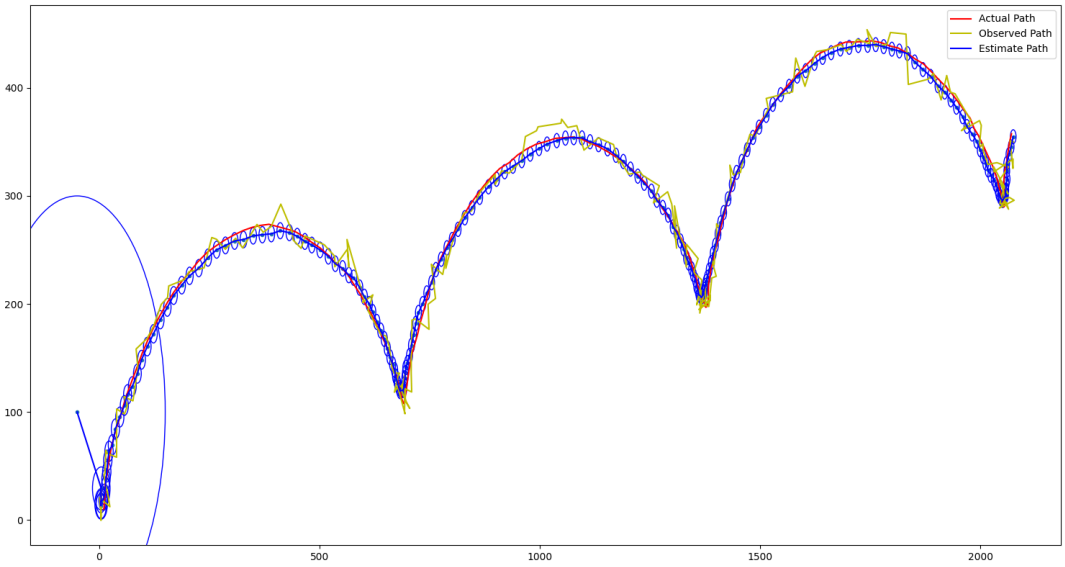


Figure 18: $\mu_0 = [[-50], [100], [0], [0]]$ and $\Sigma_0 = 10^4 * I$

OBSERVATION

- We can see that change in initial covariance matrix and μ by the initial position and size of Error Ellipse. The ellipse size is large but after 2 updates it becomes small and mostly remains of same size.
- This happens because incorporating observation deflates sigma, thus its eigen values and thus the ellipse formed by them.

2.7 *g: Dropping Observations at t=10 and t=30 for 10 time steps*

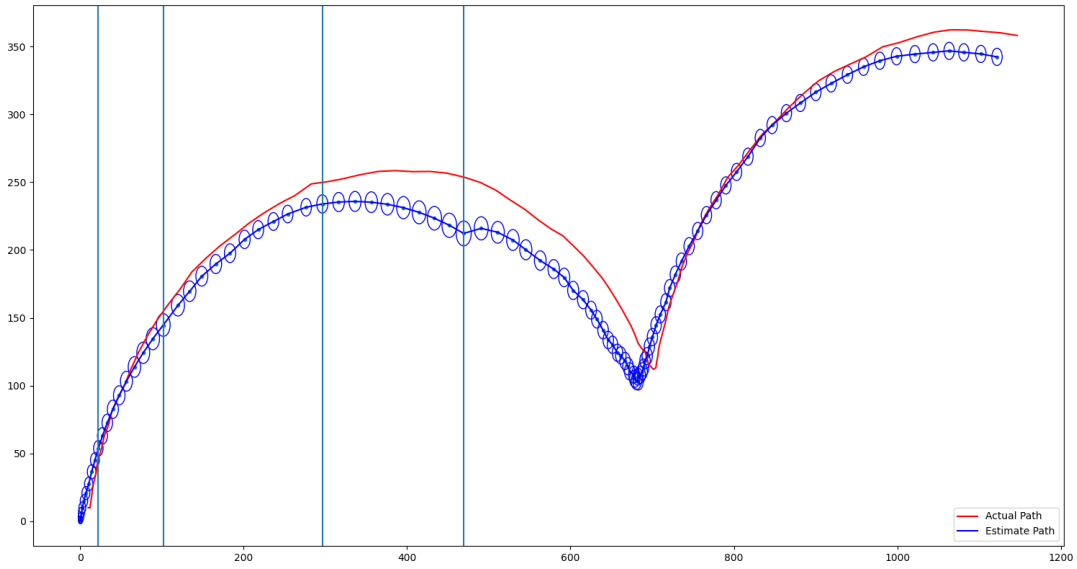


Figure 19: vertical lines denote time of dropping observations; 100 sim steps

Observations

- From vertical line 1 to vertical line 2, the estimator follows only motion model, thus deviating slightly from actual trajectory.
- In this region the size of ellipse also increasing slightly, as uncertainty is increasing.
- From vertical line 3 to verticle line 4, again only motion model is followed, and we can observe the deviation from actual trajectory, the estimator has started moving towards the trough already.
- Increase in size of ellipse is more evident here.
- As soon as we pass line 4 the observations are incorporated, sigma is deflated decreasing the size of ellipse and convergence starts.

2.8 *h: Velocity tracking*

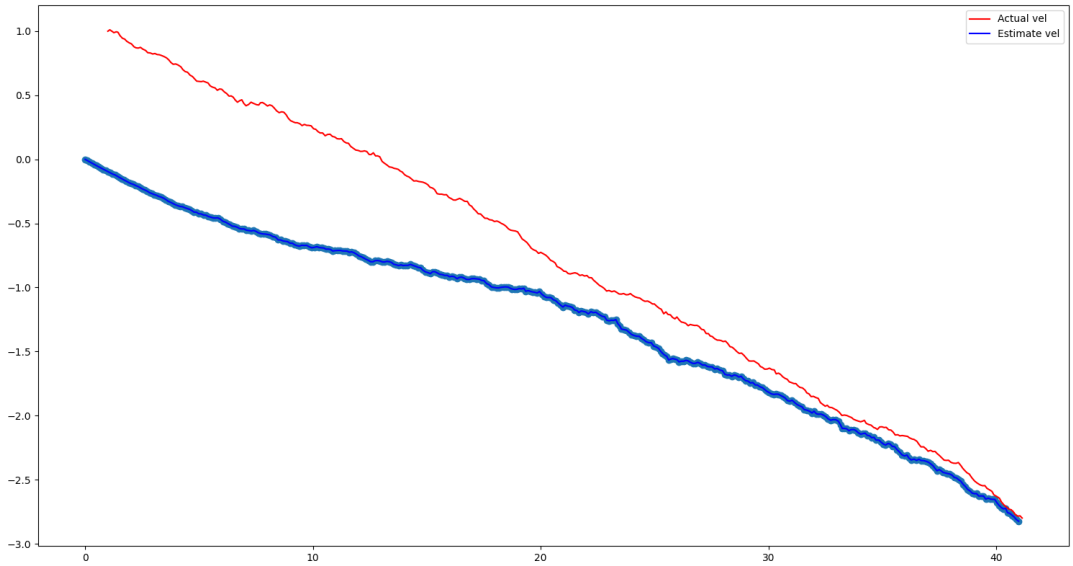


Figure 20: Parabolic trajectory velocity tracking with constant u=[[0.1],[-0.01]]

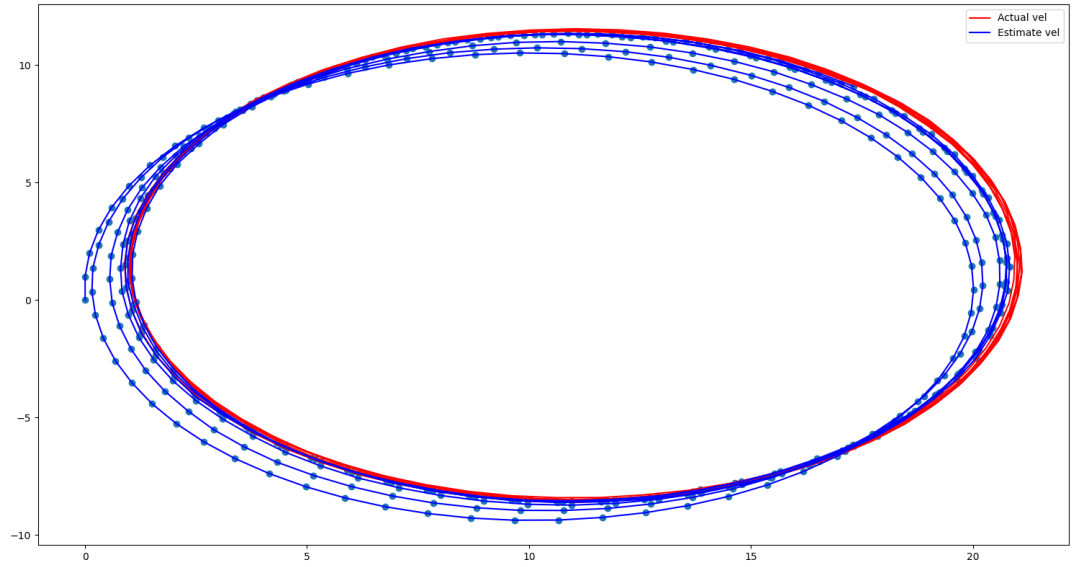


Figure 21: With sinusoidal u

OBSERVATIONS

- We can observe that estimator can track velocity although it is taking more time for velocities to converge.
- Even in sinusoidal velocity tracking, the blue ellipse is moving closer to red ellipse to overlap.
- Estimator is able to track velocities even though it doesn't observe it is because it can track displacement, and time units are fixed, thereby giving indirect method to track velocity. For it to predict better trajectory it will have to correct it velocity.

2.9 *i: Data Association*

We used ford fulkerson algorithm to solve the max matching problem.
We will form a bipartite graph with objects on left and observations on right. Edge weight between any object and observations is 1/mahalonobis distance(obj,observation). mahalonobis distance = $\sqrt[2]{(x-\mu)^T * \Sigma^{-1} * (x-\mu)}$
Some problems with it are: It is sensitive to observation noise, It is sensitive to velocities
It performed well for these examples with considerable noise.

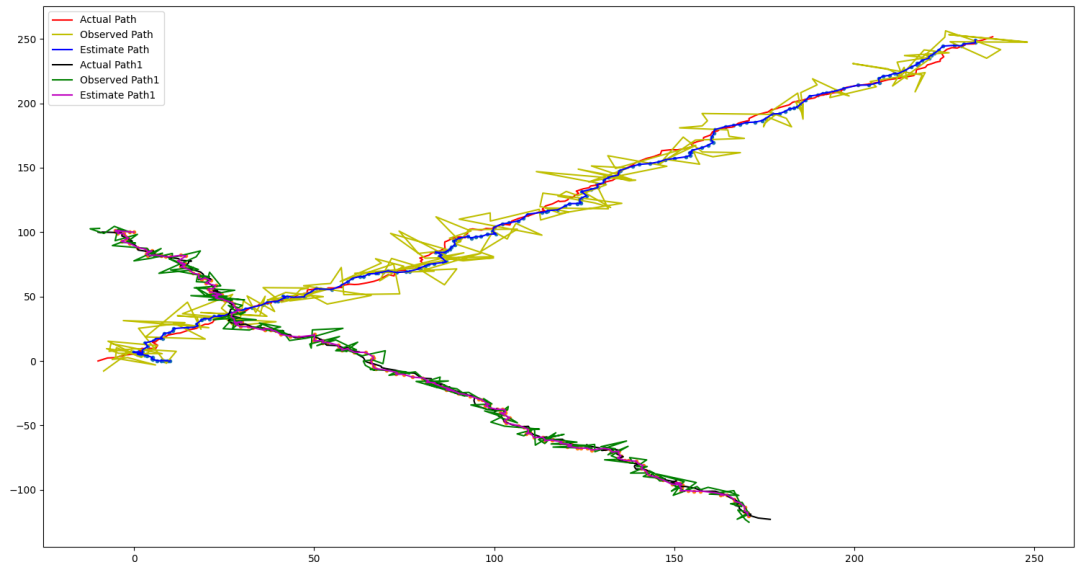


Figure 22: Intersecting trajectories, estimate is still maintained after intersection

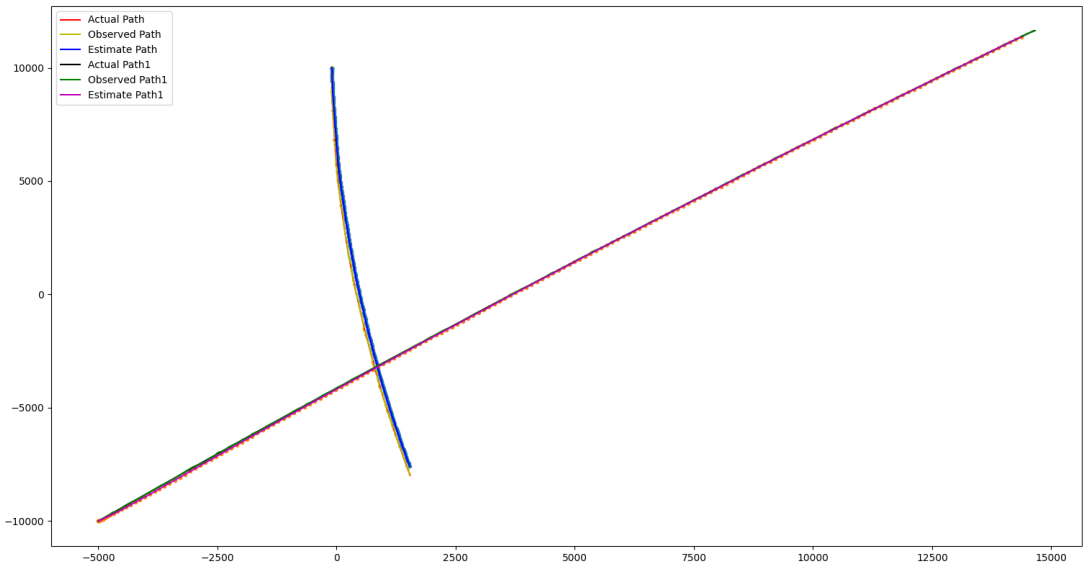


Figure 23: With faster velocities

Yes the solution will scale for 4-5 objects, as ford fulkerson algorithms scales pretty well. The problem will still be same sensitivity to noise and sensitivity to velocities during intersection.
Note: As this was only 2 object problem we didn't write code for algorithm but did it specifically for 2, but the code can be scaled for any number of objects