Assignment - 4

Anurag Sahu (2018121004)
Apoorva Srivastava (20191702014)

Theory:

Robot : The device that moves through the environment and models it.

Localization : Localization is the estimate of the position of our robot in the world knowing other points in the environment. For Example:

- Estimate the robot's position given landmarks

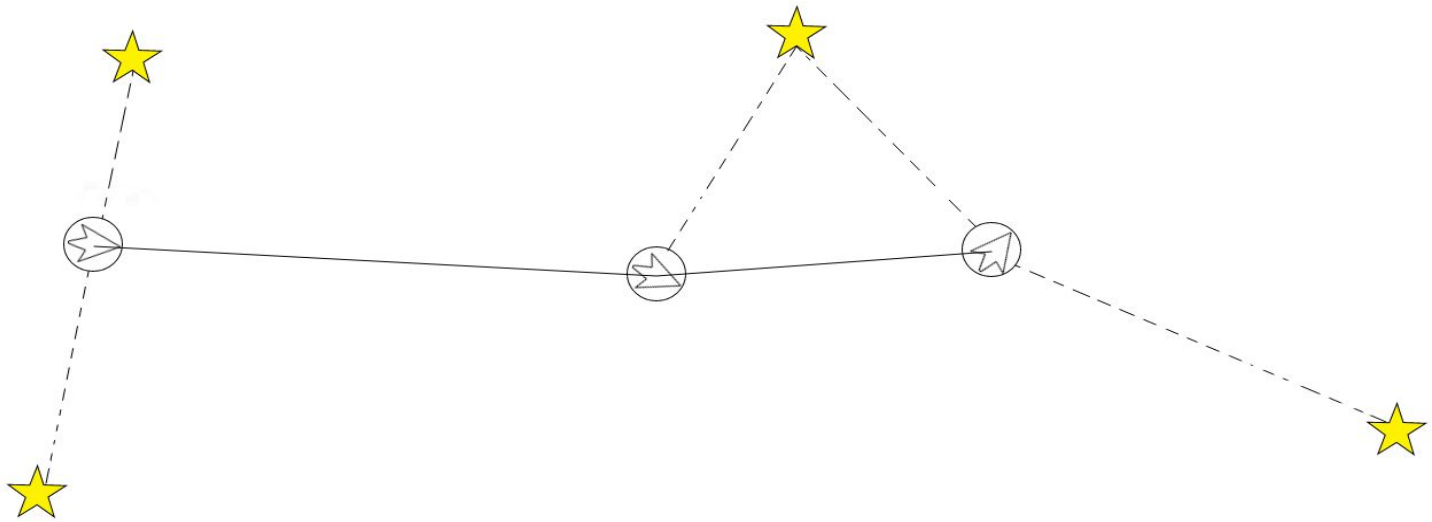


Figure : 1

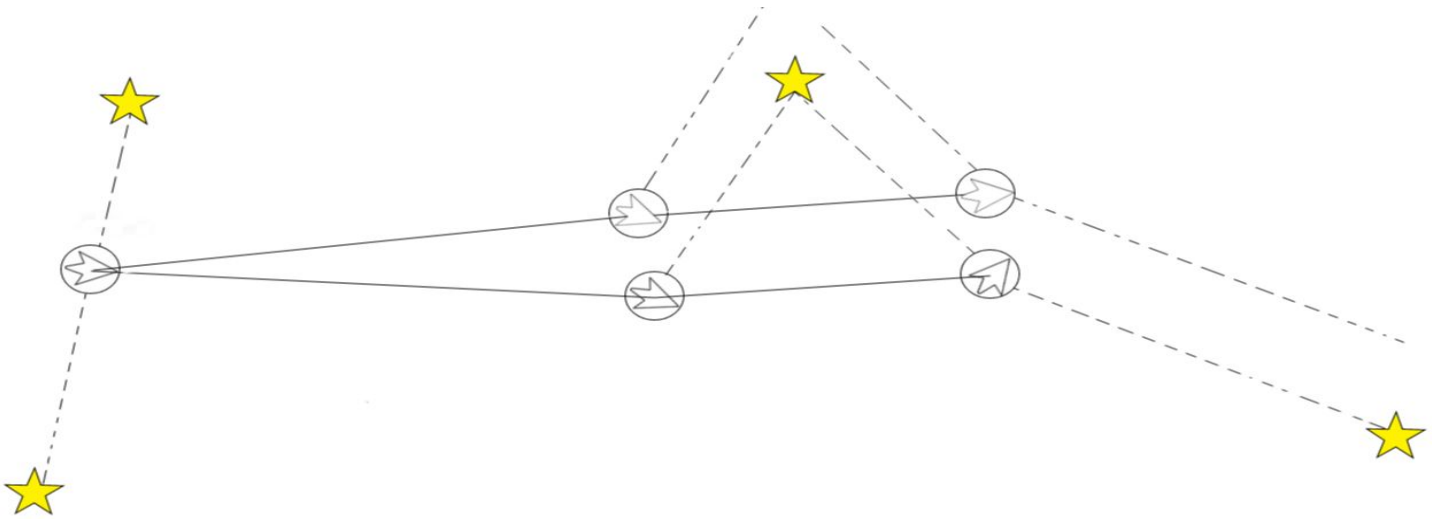


Figure : 2

*In Figure 1 robot has been provided with known landmarks and in Figure 2 it is correcting its trajectory with the help of those landmarks.

Mapping: Mapping is estimating the model of the environment having the knowledge of robots location in the world. Mapping Example: Estimate the landmarks given the robot's pose

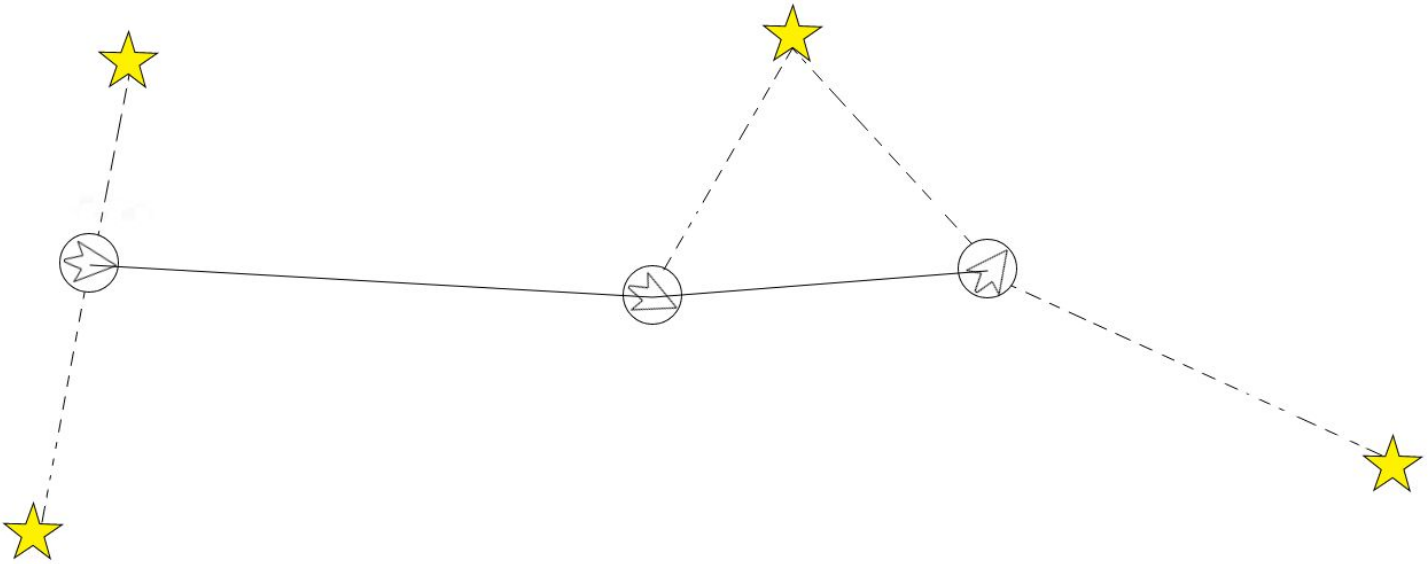


Figure : 3

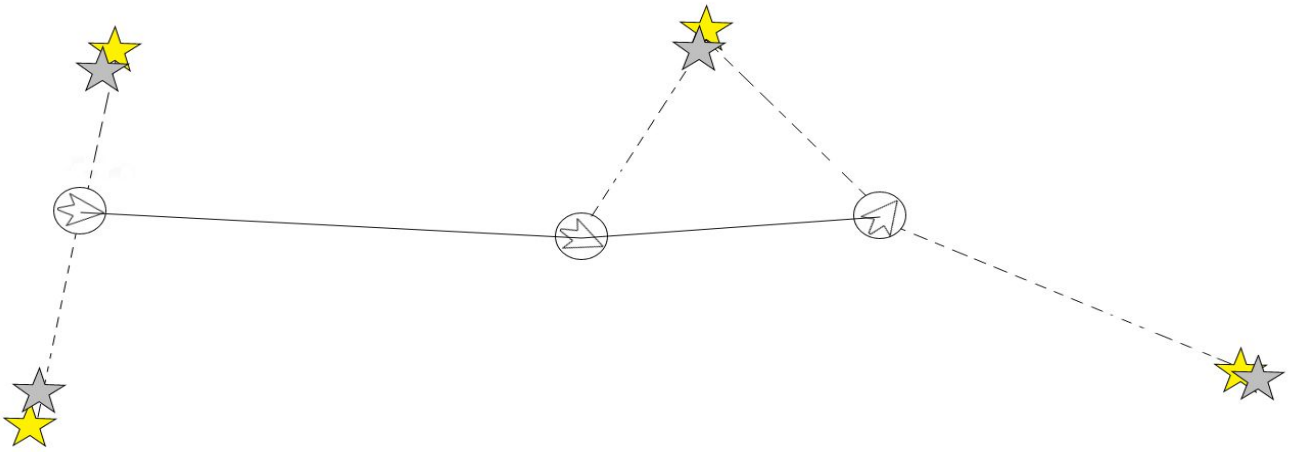


Figure : 4

* In Figure 3, Robot has been provided with its location in the environment, using which it estimates the location of landmarks in Figure 4.

SLAM(Simultaneous Localization and Mapping) : SLAM is the problem of estimating the robots location and Exploring the environment's landmarks simultaneously.SLAM Example: Estimate the robot's poses and the landmarks at the same time.

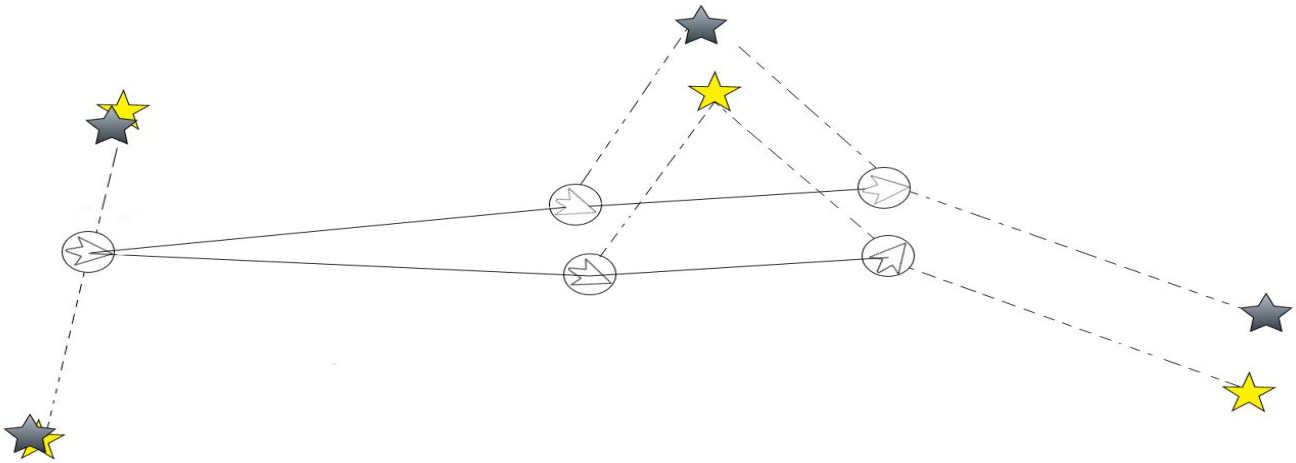


Figure : 5

*In figure 5, Robot is solving a SLAM Problem by estimating its own location and the landmarks locations simultaneously.

Bayesian Filter : **Bayes filter**, is a general probabilistic approach for estimating an unknown probability density function recursively over time using incoming measurements and a mathematical process model.
The Bayesian filter can be expressed as the following two steps:

a. Prediction Step

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx_{t-1}$$

b. Correction Step

$$bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t))$$

Extended Kalman Filter : The **Extended Kalman filter (EKF)** is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. It assumes gaussian distribution and Nonlinear for error.

Given :

t: a 12609×1 array containing the data timestamps [s].

x_true: a 12609×1 array containing the true x-position, x_k, of the robot [m].

y_true: a 12609×1 array containing the true y-position, y_k, of the robot [m].

th_true: a 12609×1 array containing the true heading, θ_k, of the robot [m].

l: a 17×2 array containing the true xy-positions, (x^l, y^l), of all the 17 landmarks [m].

r: a 12609×17 array containing the range, r_k^l, between the robot's laser rangefinder and each landmark as measured by the laser rangefinder sensor [m] (a range of 0 indicates the landmark was not visible at that timestep).

r_var: the variance of the range readings (based on groundtruth) [m^2].

b: a 12609×17 array containing the bearing, ϕ_k^l, to each landmark in a frame attached to the laser rangefinder, as measured by the laser rangefinder sensor [rad] (if the range is 0 it indicates the landmark was not visible at that timestep and thus the bearing is meaningless). The measurements are spread over a 240° horizontal-FoV, and the bearing is 0° when the landmark is straight ahead of the rangefinder.

b_var: the variance of the bearing readings (based on groundtruth) [rad^2].

v: a 12609×1 array containing the translational speed, v_k, of the robot as measured by the robot's odometers [m/s].

v_var: the variance of the translational speed readings (based on groundtruth) [m^2/s^2].

om: a 12609×1 array containing the rotational speed, ω_k, of the robot as measured by the robot's odometers [rad/s].

om_var: the variance of the rotational speed readings (based on groundtruth) [rad^2/s^2].

d: the distance, d, between the center of the robot and the laser rangefinder [m].

Motion Model :

It tells how the state of robot is changing as the control inputs are applied with time.Here v and om are the control inputs and x,y,and theta are the c=variables determining the next and current states of the robot.

$$\begin{bmatrix} x_k \\ y_k \\ \Theta_k \end{bmatrix} = f\left(x_{k-1},\ y_{k-1},\ \Theta_{k-1}\right) = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \Theta_{k-1} \end{bmatrix} + \begin{bmatrix} cos\Theta_{k-1} & 0 \\ sin\Theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} v_r \\ \omega_r \end{bmatrix} + W_r \right)$$

Sensor Model:

Sensor Model provides the required correction in the states, estimated using motion model,by using the observations of the known landmark through range and bearing sensors.

$$\begin{bmatrix} \mu_k^l \\ \theta_k^l \end{bmatrix} = \begin{bmatrix} \sqrt{(x_l - x_k - dcos(\theta_k))^2 + (y_l - y_k - dsin(\theta_k))^2} \\ tan^{-1}(\frac{y_l - y_k - dsin(\theta_k)}{x_l - x_k - dcos(\theta_k)}) - \theta_R \end{bmatrix} + n_k$$

<u>Algorithm</u>:

Below is the algorithm for one time step carried out with respect to 17 landmarks.We have repeated this for 12609 time steps to get the trajectory of the robot.

$$\begin{aligned} \text{predictor:} \quad \check{\mathbf{P}}_k &= \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}'_k \\ \check{\mathbf{x}}_k &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}) \\ \text{Kalman gain:} \quad \mathbf{K}_k &= \check{\mathbf{P}}_k\mathbf{G}_k^T\left(\mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^T + \mathbf{R}'_k\right)^{-1} \\ \text{corrector:} \quad \hat{\mathbf{P}}_k &= \left(\mathbf{1} - \mathbf{K}_k\mathbf{G}_k\right)\check{\mathbf{P}}_k \\ \hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k\underbrace{\left(\mathbf{y}_k - \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0})\right)}_{\text{innovation}} \end{aligned}$$

Steps:

We estimate the current state of the robot using the previous state and the control inputs using the motion model.

Here F is the Jacobian of the motion model $f(x_{k-1},\ y_{k-1},\ \Theta_{k-1})$ with respect to x,y and theta

$$F = I + \begin{bmatrix} 0 & 0 & -(v_k + v_{var})sin(\theta_k) \\ 0 & 0 & (v_k + v_{var})cos(\theta_k) \\ 0 & 0 & 0 \end{bmatrix} \qquad P = \begin{bmatrix} \sigma_{x,x} & \sigma_{x,y} & \sigma_{x,\theta} \\ \sigma_{y,x} & \sigma_{y,y} & \sigma_{y,\theta} \\ \sigma_{\theta,x} & \sigma_{\theta,y} & \sigma_{\theta,\theta} \end{bmatrix}$$

Choose P initial = diag[1, 1, 0.1]

Here,

$$\begin{aligned} \mathbf{w}_k &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \\ \mathbf{n}_k &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \end{aligned}$$

$$\mathbf{w}'_k = \left.\frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{w}_k}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}} \mathbf{w}_k \qquad \underbrace{E\left[\mathbf{w}'_k \mathbf{w}'^T_k\right]}_{\mathbf{Q}'_k}$$

$$\mathbf{n}'_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{n}_k}\right|_{\hat{\mathbf{x}}_k, \mathbf{0}} \mathbf{n}_k \qquad \underbrace{E\left[\mathbf{n}'_k \mathbf{n}'^T_k\right]}_{\mathbf{R}'_k}$$

$$Q'_k = \begin{bmatrix} v_{var}cos^2(\theta_k) & v_{var}cos(\theta_k)sin(\theta_k) & 0 \\ v_{var}cos(\theta_k)sin(\theta_k) & v_{var}sin^2(\theta_k) & 0 \\ 0 & 0 & \omega_{var} \end{bmatrix} \qquad R'_k = \begin{bmatrix} \mu_{var} & 0 \\ 0 & b_{var} \end{bmatrix}$$

$y_k = \begin{bmatrix} r^l_k \\ b^l_k \end{bmatrix}$ is the estimated range and angle of a particular landmark measured from range and bearing sensors.

Here,G is the Jacobian of the sensor model h(x,y,theta) with respect to x,y and theta

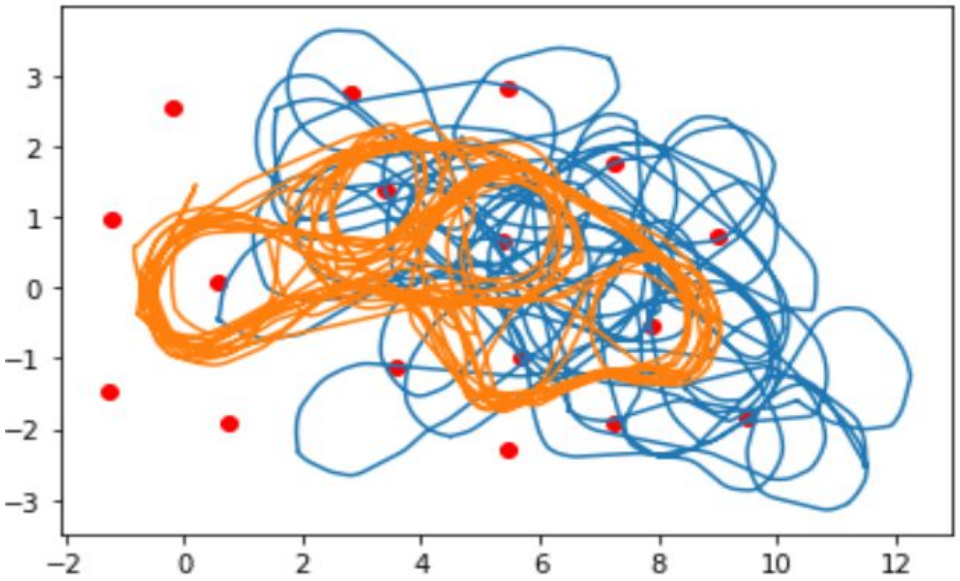
G = $\frac{\partial^3 h}{\partial x, \partial y, \partial z} = \begin{bmatrix} \frac{-(x_l - x - dcos(\theta)}{\sqrt{(x_l - x - dcos(\theta))^2 + (y_l - y - dsin(\theta))^2}} & \frac{-(y_l - y - dsin(\theta)}{\sqrt{(x_l - x - dcos(\theta))^2 + (y_l - y - dsin(\theta))^2}} & \frac{(x_l - x - dcos(\theta))(dsin(\theta)) - (y_l - y - dsin(\theta)(dcos(\theta))}{\sqrt{(x_l - x - dcos(\theta))^2 + (y_l - y - dsin(\theta))^2}} \\ \frac{(y_l - y - dsin(\theta)}{(x_l - x - dcos(\theta))^2 + (y_l - y - dsin(\theta))^2} & \frac{-(x_l - x - dcos(\theta)}{(x_l - x - dcos(\theta))^2 + (y_l - y - dsin(\theta))^2} & \frac{-(x_l - x - dcos(\theta))(dcos(\theta)) - (y_l - y - dsin(\theta)(dsin(\theta))}{(x_l - x - dcos(\theta))^2 + (y_l - y - dsin(\theta))^2} \end{bmatrix}$

Corrector Step gives the present state and the Covariance matrix of the present state with the help of the observations from sensor model for each landmark and the calculated Kalman gain as per the EKF algorithm for each time step .

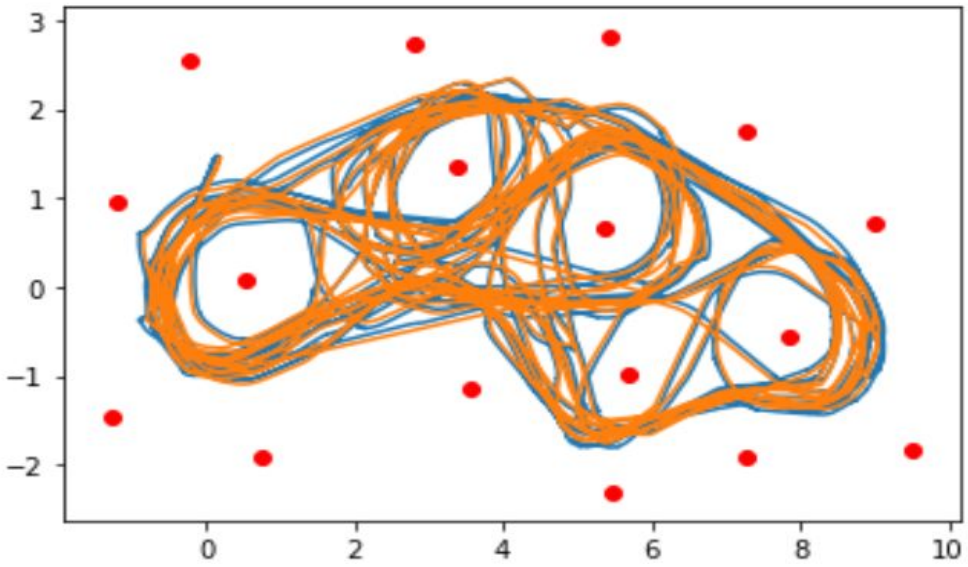
Link to the Code : Github Link, Google Drive Link

Output :

Plot of Robots trajectory without EKF along with the Ground Truth



Plot of Robots trajectory using EKF along with the Ground Truth:



Bonus question:

Algorithm for Bonus Question:

To plot the estimated 3-sigma Covariance Ellipse associated with every position,we need to perform the Eigendecomposition of the Covariance Matrix and further they help in retrieving the shape of the ellipse.So we perform following steps:

1)Find the eigenvalues ($\lambda 1$ and $\lambda 2$) and eigenvectors (**e1** and **e2**) of the covariance matrix, *P* at each position. The eigenvectors form an orthogonal basis for the coordinate system for plotting the ellipse. The first eigenvector (**e1**) points in the direction of the greatest variance and defines the major axis for the prediction ellipse. The second eigenvector (**e2**) points in the direction of the minor axis.

2) Major axis=sqrt(Bigger_EigenValue); Minor axis=sqrt(Smaller_EigenValue);
Angular Orientation of Ellipse=atan2(Bigger_EigenValue,Smaller_EigenValue);

3)We pass the above calculated values to the Ellipse Object of matplotlib to get the estimated 3-sigma Covariance Ellipse associated with every position.

Output:

A video showing EKF estimate as the robot drives around, as a dot on a map,
along with the true robot position and the landmark locations.
The estimated 3-sigma Covariance Ellipse is associated with every position: YouTube Link