MOBILE ROBOTS LOCALIZATION LAB

UNIVERSITY OF GENOVA

EMARO+ 2015-2017

Razeen Hussain & Dipendra Subedi

AIMS:

The aim of this lab is to develop a localization system based on the extended kalman filter (EKF). The lab will first focus on tuning the kalman filter and then will evaluate its performance with the case of unknown system parameters. MATLAB will be used to implement and evaluate the filter.

INTRODUCTION:

Localization is the process of determining the *'pose'* of a robot. Typically, the pose of a robot refers to the position as well as orientation of the robot, totaling to 6 degrees of freedom. In the lab, we will consider only a 2-D space model. In this case, we only need to determine the posture (x, y and θ coordinates) of the mobile robot.

Although many different techniques exist for localization, for the purpose of the lab, we will focus on the hybrid localization technique using an extended kalman filter (EKF). This approach has been successfully used in various applications and is computationally efficient and easy to program. The basic logic/equations used in this technique are described below:

EXTENDED KALMAN FILTER (EKF):

In estimation theory, 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 can be modeled in the following way.

Considering the input as a random signal, e.g. when the input is measured with a certain amount of noise. We call U^* the noisy measurement of the input U. Classically, we consider that U^* is U disturbed by an additive noise β characterized by its (m x m) covariance matrix Q_β.

In this case, the equations of the extended Kalman filter can be modified as follows:

$$\begin{cases} X_{k+1} = f(X_k, U_k) + \alpha_k \\ U_k^* = U_k + \beta_k \\ Y_k = g(X_k) + \gamma_k \end{cases}$$

The prediction phase becomes:

$$\begin{cases} \hat{X}_{k+1/k} = f(\hat{X}_{k/k}, U_k^*) \\ P_{k+1/k} = A_k.P_{k/k}.A_k^T + B_k.Q_\beta.B_k^T + Q_\alpha \end{cases}$$

With A_k and B_k defined as:

$$A_k = \frac{\partial f}{\partial X}(\hat{X}_{k/k}, U_k^*)$$

$$B_k = \frac{\partial f}{\partial U}(\hat{X}_{k/k}, U_k^*)$$

In the calculation of the predicted state and of matrices A and B, we have no other choice but to use the measured input, as the real input is unknown.

The estimation phase is:

$$\begin{cases} \hat{X}_{k+1/k+1} = \hat{X}_{k+1/k} + K_k\left[Y_k - g(\hat{X}_{k+1/k})\right] \\ P_{k+1/k+1} = (I - K_k.C_k)P_{k+1/k} \end{cases}$$

With: h

$$\begin{cases} C_k = \frac{\partial g}{\partial X}(\hat{X}_{k+1/k}) \\ K_k = P_{k+1/k}.C_k^T\left(C_k.P_{k+1/k}.C_k^T + Q_\gamma\right)^{-1} \end{cases}$$

<u>THE SYSTEM:</u>

The system composes of a NXT robot equipped with eight custom made Reed sensors. The reed sensors are placed in a horizontal line on the front of the robot. Each sensor is placed 10mm apart. The states of the reed sensors are transmitted to the robot via an I2C bus. The floor is equipped with a square array of magnets which are 55mm apart. They act as beacons of the system as their locations are known. Additionally, the robot has two wheel encoders which have a resolution of 360 dots per revolution.

When the robot passes over a magnet, it returns the state of the sensors which can be used to determine the y-coordinate of the magnet in the robot frame. Also, since the position of the sensors in the robot frame is known, the x-coordinate of the magnets in the robot frame can be simultaneously calculated.

Localization of the system is done by implementing an extended kalman filter (EKF). The measurement equation for the filter will be the x and y-coordinates of the magnets in the robot frame as described previously.

$$ {}^{0}P_{mag} = \begin{bmatrix} {}^{0}x_{mag} \\ {}^{0}y_{mag} \\ 1 \end{bmatrix} $$

The posture of the robot (denoted by *X*) and the transformation matrix are as follows:

$$ X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} $$

$$ {}^{0}T_{m} = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} $$

Using this, the position of the magnet in the robot frame can be easily calculated.

$$ {}^{m}P_{mag} = {}^{0}T_{m}^{-1}.\,{}^{0}P_{mag} = \begin{bmatrix} \cos\theta\left({}^{0}x_{mag} - x\right) + \sin\theta\left({}^{0}y_{mag} - y\right) \\ \cos\theta\left({}^{0}y_{mag} - y\right) - \sin\theta\left({}^{0}x_{mag} - x\right) \\ 1 \end{bmatrix} = \begin{bmatrix} Y \\ 1 \end{bmatrix} = \begin{bmatrix} g_{mag}(X) \\ 1 \end{bmatrix} $$

This is the observation equation which depends on the particular magnet under consideration. This equation expresses the outputs as a function of the state.

The system equations can be formed as follows:

$$ X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \Delta D_k \cos\theta_k \\ y_k + \Delta D_k \sin\theta_k \\ \theta_k + \Delta\theta_k \end{bmatrix} = f(X_k, U_k) $$

$$ U_k = \begin{bmatrix} \Delta D_k \\ \Delta\theta_k \end{bmatrix} = \begin{bmatrix} \dfrac{r_r \Delta q_{r,k} + r_l \Delta q_{l,k}}{2} \\ \dfrac{r_r \Delta q_{r,k} - r_l \Delta q_{l,k}}{e} \end{bmatrix} $$

It is equally valid to take the vector of elementary rotations of the wheels $[\Delta q_r, \Delta q_l]^T$ as input vector.

Note: any parameter of '*f*' which is neither in '*X*' nor in '*U*' must be a known constant. This is true in our case: only 'r_r', 'r_l' and '*e*', which are assumed known constants, have been left out. In some cases, unknown robot parameters can be identified by adding them to the state vector, as will be the case in the later exercise.

Using these, we can now compute the A, B and C matrices of the kalman filter.

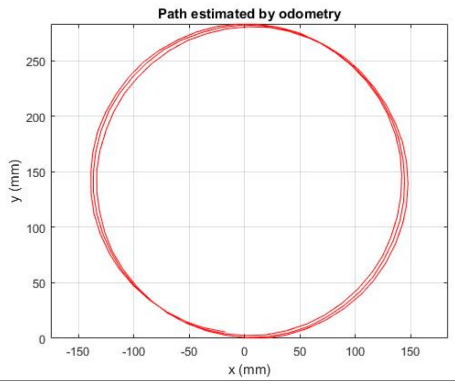
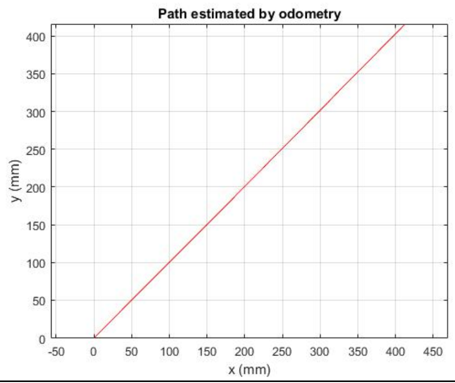
$$A = \frac{\partial f}{\partial X} = \begin{bmatrix} 1 & 0 & -\Delta D \sin\theta \\ 0 & 1 & \Delta D \cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

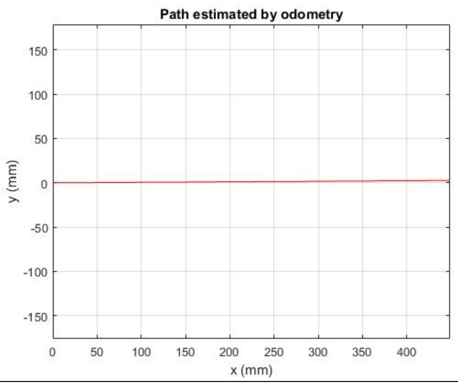
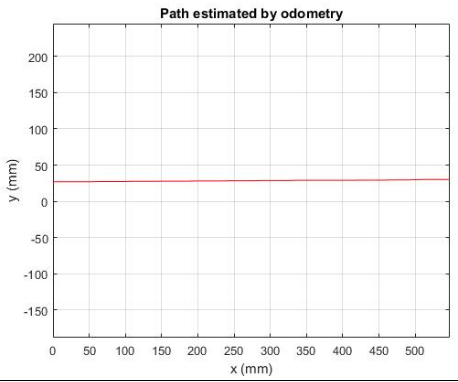
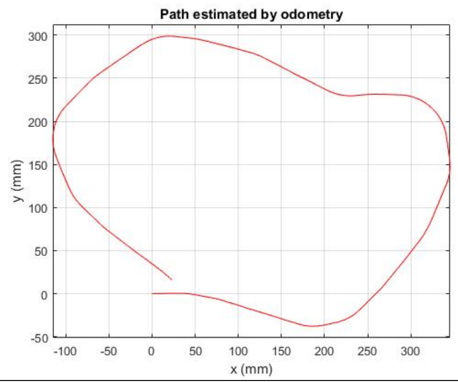
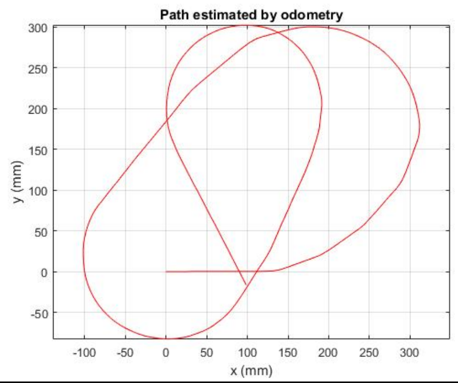
$$B = \frac{\partial f}{\partial U} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}$$

$$C = \frac{\partial g_{mag}(X)}{\partial X} = \begin{bmatrix} -\cos\theta & -\sin\theta & \sin\theta\left(x - {}^{0}x_{mag}\right) - \cos\theta\left(y - {}^{0}y_{mag}\right) \\ \sin\theta & -\cos\theta & \sin\theta\left(y - {}^{0}y_{mag}\right) + \cos\theta\left(x - {}^{0}x_{mag}\right) \end{bmatrix}$$

ODOMETRY / INITIALIZATION:

For evaluation purposes, data of different trajectories will be used. These data were taken while doing actual experiments. The initial posture and the different trajectories under consideration can be seen below:

TRAJECTORY	INITIAL POSTURE	ODOMETRY
Circle	(0,0,0)	
Diagonal at 45^{o}	(0,0,pi/4)	

Straight Line with 1 Magnets	(0,0,0)	
Straight Line with 2 Magnets	(0,27,0)	
One Loop	(0,0,0)	
Two Loops	(0,0,0)	

SIGMA MEASUREMENTS:

There are some uncertainties present in the system. The first one is that the robot is being placed on the floor by hand. Since humans are not 100% accurate, our initial posture is an approximation of the real initial posture. We can compensate for this source into account by introducing the following uncertainties in the initialization values:

$$sigmaX = 3mm$$
$$sigmaY = 3mm$$
$$sigma\theta = 3°$$

Using these, we can obtain the initial *'P'* matrix:

$$P_{init} = \begin{bmatrix} sigmaX^2 & 0 & 0 \\ 0 & sigmaY^2 & 0 \\ 0 & 0 & sigma\theta^2 \end{bmatrix}$$

Another uncertainty exists in the form of measurement noise. This is denoted by the 'Q_γ' matrix:

$$Q_\gamma = \begin{bmatrix} sigmaX_{measurement}^2 & 0 \\ 0 & sigmaY_{measurement}^2 \end{bmatrix}$$

The uncertainty in the y-coordinate measurement is due to the fact that the sensors are placed 10mm apart so when the magnet is detected, we do not know the exact value of the y-coordinate as it can be anywhere in between the sensors. To estimate this uncertainty, we can assume a random distribution with maximum error as 20mm which is 10mm on one side of the sensor and 10mm on the other. Thus the value of the *'$sigmaY_{measurement}$'* becomes:

$$sigmaY_{measurement} = \frac{20}{\sqrt{12}}$$

The uncertainty in the x-coordinate measurement can be approximated using the raw sensor data. The simplest data is the one where the trajectory is a straight line.

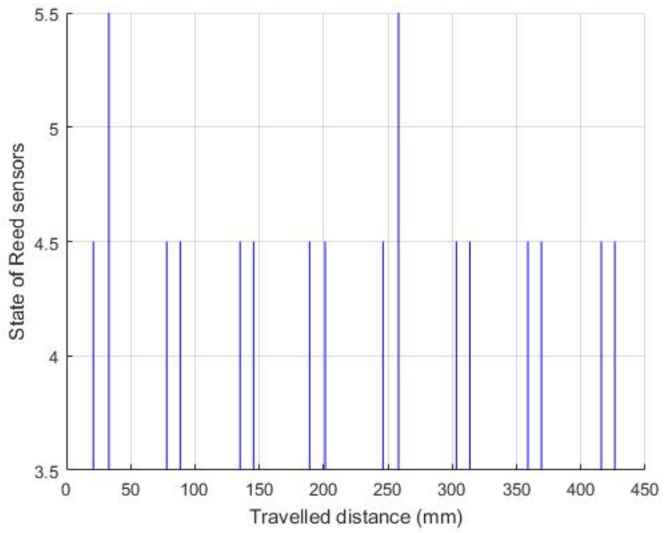


Figure 1: Raw sensor data for straight line path

It can be seen in the figure that the maximum distance between the two adjacent lines is 12mm. Since the magnets have a range in which their fields act, we can add 6mm uncertainties on each side making the maximum error as 24mm. Like in the case of y-measurement noise, we can also model the x-measurement noise as a random distribution. This results in the following value of '$sigmaY_{measurement}$':

$$sigmaY_{measurement} = \frac{24}{\sqrt{12}}$$

Once the above uncertainties have been accounted for, the 'Q_α' and 'Q_β' remain. For simplicity and a better behavior we assume '$Q_\alpha = 0$' and we try to tune 'Q_β'. In that case, '*sigmaWheels*' is our tuning parameter. This parameter can be tuned using the mahalanobis distances. Each time, the sensor detects a magnet; its mahalanobis distance is calculated. In the following image, the green dots represent the closest real magnet while the red dots represent the nearest neighbors.

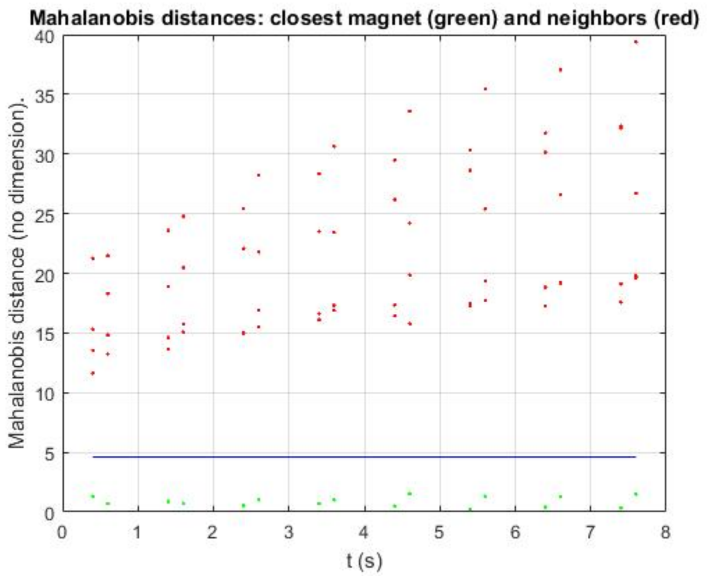


Figure 2: Mahalanobis distances for straight line path

The blue line is the threshold for the mahalanobis distances which is calculated using an inbuilt MATLAB function:

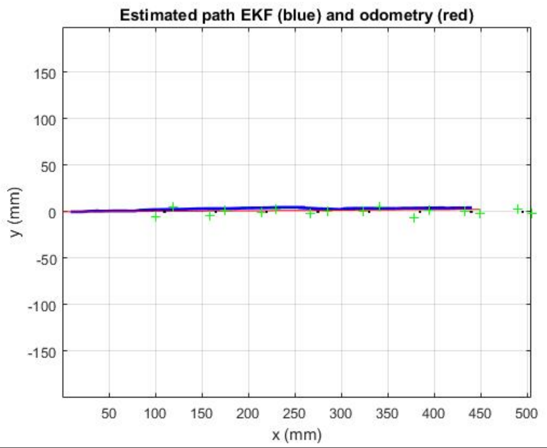
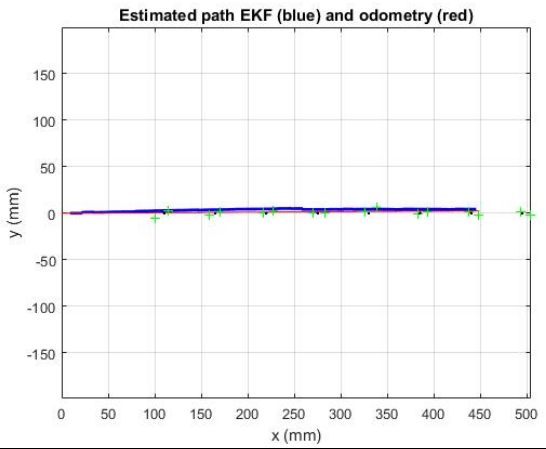
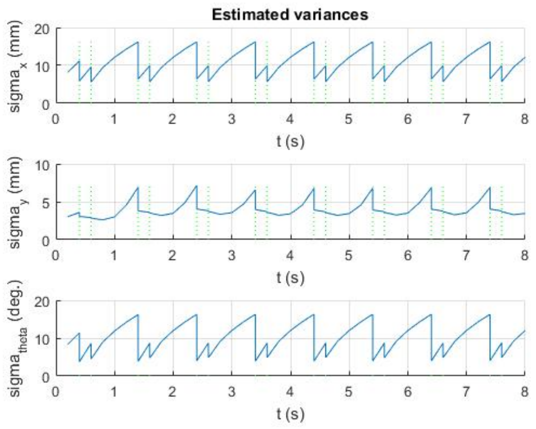
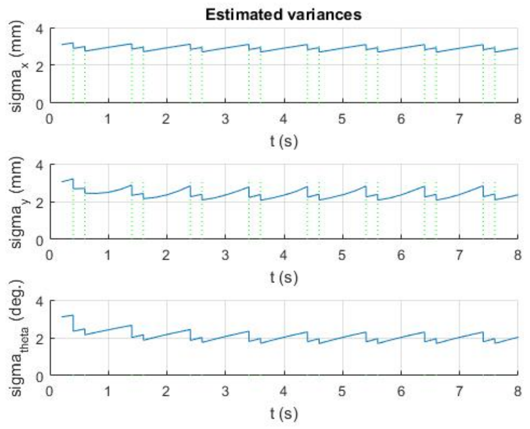
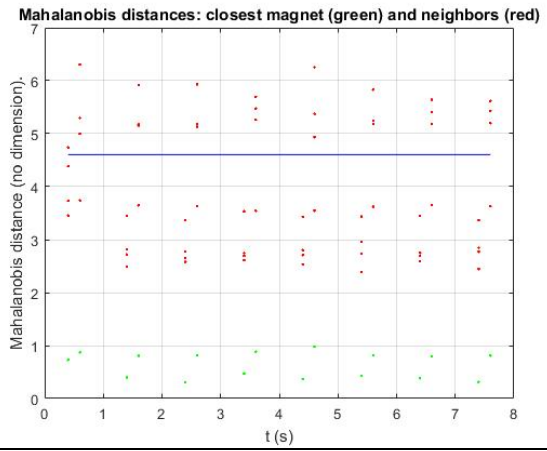
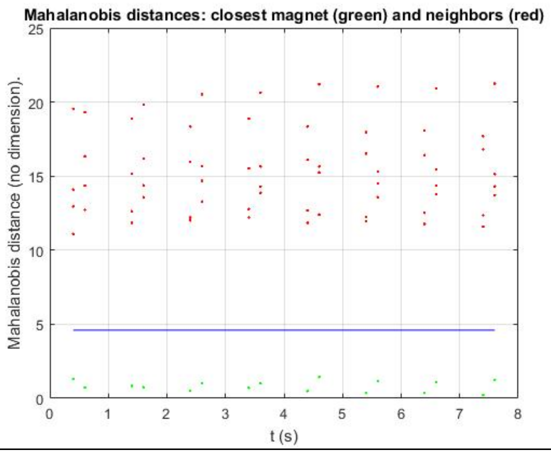
$$mahaThreshold = chi2inv(P, V)$$

The first argument is the probability and the second one is the degree of freedom. In our case, the values of '*P=0.90*' is sufficient and since we our system has two degree of freedom in the observation equation, we can use '*V=2*'.

Ideally, if the 'sigmaWheels' parameter is tuned properly, all the green dots should be below the blue line and all the red dots should be significantly above the blue line. Also, the estimated error variances' curves should have decreasing slopes. Simulations for the trajectories described previously are shown below:

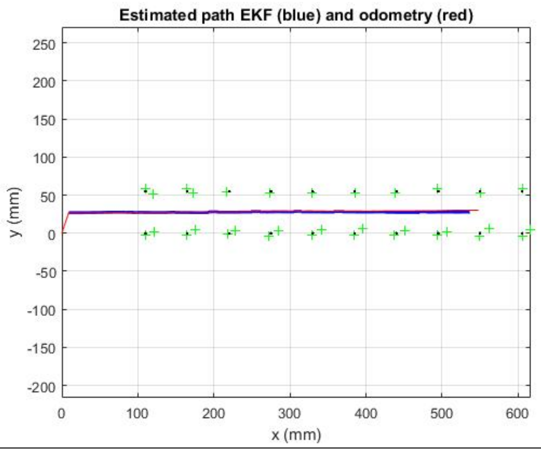
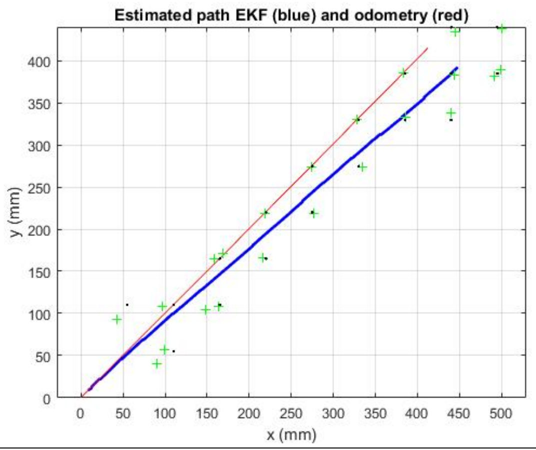
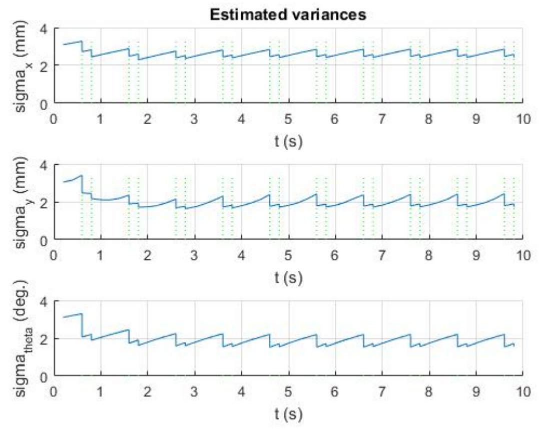
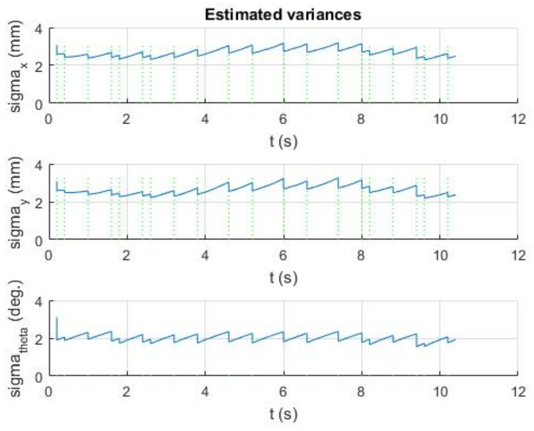
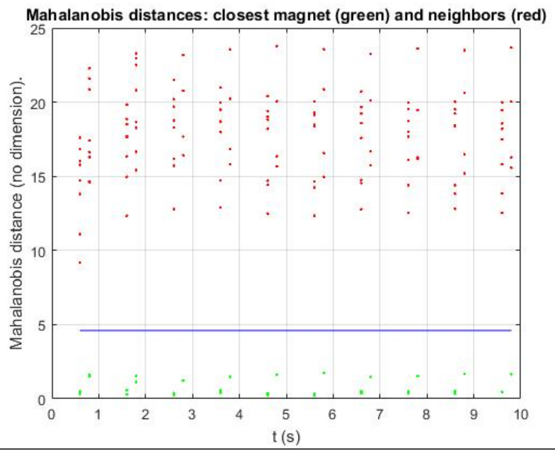
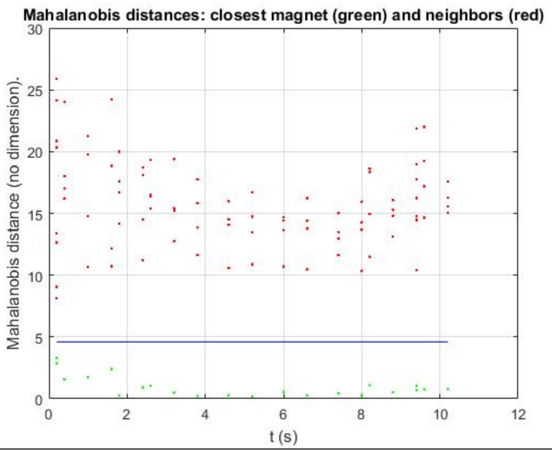
SIMULATIONS (Lab 1):

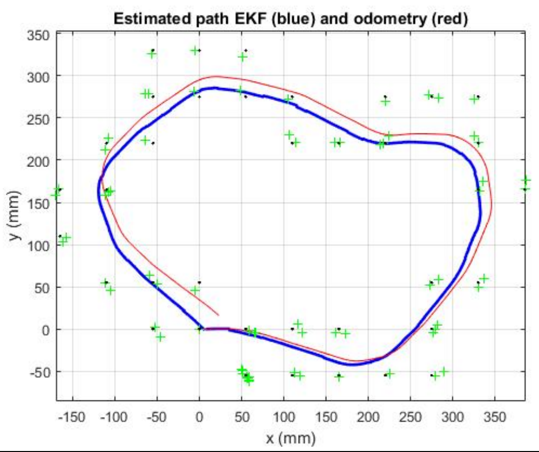
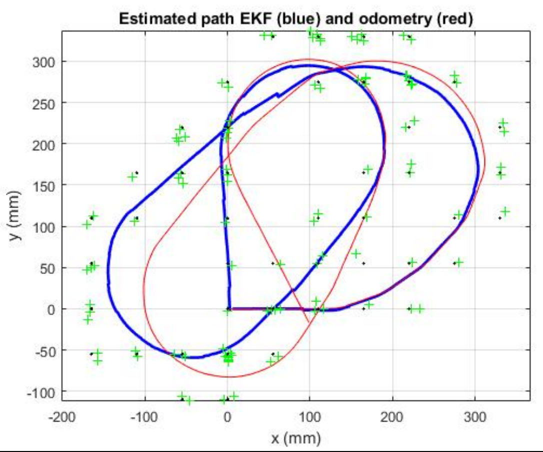
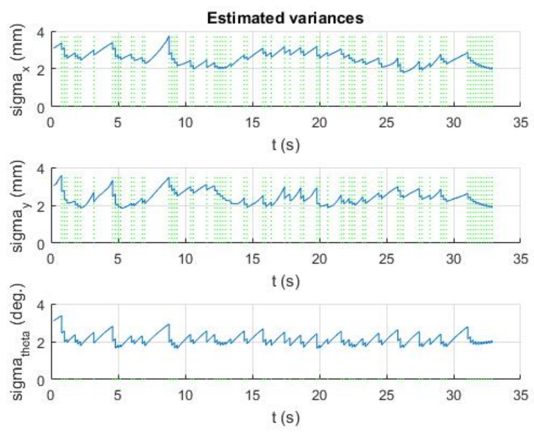
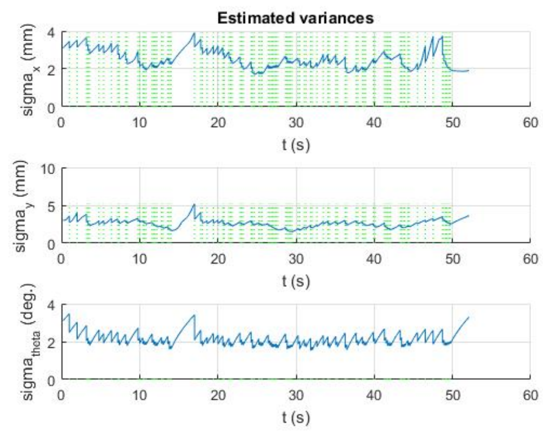
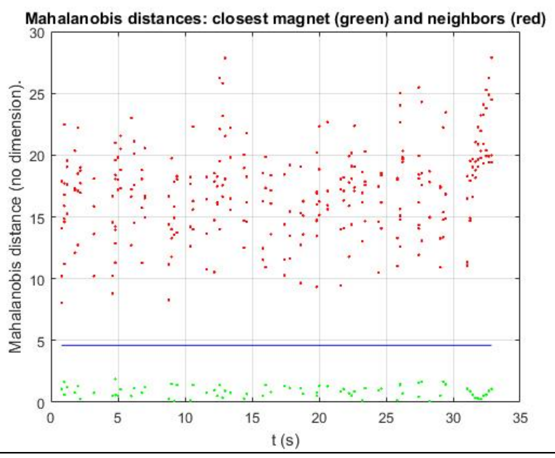
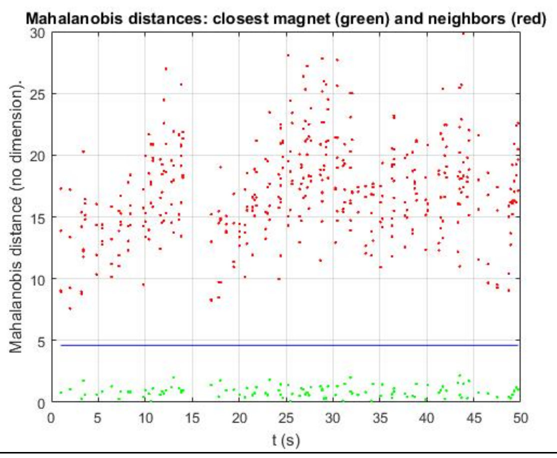
The simulations results for different paths are as follows:

Line1 Magnets (Filter not Tuned properly) sigmaWheels = 0.5	**Line1 Magnets (Filter Tuned Properly)** sigmaWheels = 0.05
	
	
	

In the above simulations, we can see that the estimated variances are very high when *‘sigmaWheels = 0.5’*. So in order to reduce the estimated variances, we decrease the value of *‘sigmaWheels’* to 0.05 to obtain the acceptable values of estimated variances.

With properly tuned filter we obtained the following results for different paths.

Line2 Magnets (sigmaWheels=0.05)	**Diagonal 45 Degrees (sigmaWheels=0.05)**
	
	
	

Loop1 (sigmaWheels=0.05)	Loop2 (sigmaWheels=0.05)
	
	
	

In the two cases of loops, we can now see that the EKF estimated path returns the robot to its original position. This was not the case in the odometry in which robot returns to a different location. This was so because the noises were not taken into account.

<u>WHEEL RADIUS ESTIMATION:</u>

Previously, we saw the implementation of the extended kalman filter for the purpose of localization for a wheeled mobile robot. Now, we explore the EKF usage in the case where some parameters slowly evolve over time. An example of such a parameter is the wheel radius. The radius of a wheel changes with time. This is due to various reasons such as wear of the wheels, effect of temperature, loss of pressure etc. The wheel radius is an important parameter in determining the odometry of a mobile robot.

For this purpose, we add the wheel radii to our state vector.

$$X = \begin{bmatrix} x \\ y \\ \theta \\ r_r \\ r_l \end{bmatrix}$$

Due to this addition, the system equations are modified as follows:

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ r_{r,k+1} \\ r_{l,k+1} \end{bmatrix} = \begin{bmatrix} x_k + period\,\frac{\dot{q}_{r,k} r_{r,k} + \dot{q}_{r,k} r_{r,k}}{2} \cos\theta_k \\ y_k + period\,\frac{\dot{q}_{r,k} r_{r,k} + \dot{q}_{r,k} r_{r,k}}{2} \sin\theta_k \\ \theta_k + \frac{period}{trackgauge} \frac{\dot{q}_{r,k} r_{r,k} + \dot{q}_{r,k} r_{r,k}}{2} \\ r_{r,k} \\ r_{l,k} \end{bmatrix}$$

$$U_k = \begin{bmatrix} \dot{q}_{r,k} \\ \dot{q}_{l,k} \end{bmatrix}$$

Also, the kalman filter matrices are also changed.

$$A = \begin{bmatrix} 1 & 0 & -period\,\frac{r_r \dot{q}_r + r_l \dot{q}_l}{2} \sin\theta & \dot{q}_r \frac{period}{2} \cos\theta & \dot{q}_l \frac{period}{2} \cos\theta \\ 0 & 1 & period\,\frac{r_r \dot{q}_r + r_l \dot{q}_l}{2} \cos\theta & \dot{q}_r \frac{period}{2} \sin\theta & \dot{q}_l \frac{period}{2} \sin\theta \\ 0 & 0 & 1 & \dot{q}_r \frac{period}{trackgauge} & -\dot{q}_l \frac{period}{trackgauge} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} r_r \frac{period}{2}\cos\theta & r_l \frac{period}{2}\cos\theta \\ r_r \frac{period}{2}\sin\theta & r_l \frac{period}{2}\sin\theta \\ \frac{period}{trackgauge} r_r & -\frac{period}{trackgauge} r_l \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} -\cos\theta & -\sin\theta & \sin\theta\left(x - {}^0x_{mag}\right) - \cos\theta\left(y - {}^0y_{mag}\right) & 0 & 0 \\ \sin\theta & \cos\theta & \sin\theta\left(y - {}^0y_{mag}\right) + \cos\theta\left(x - {}^0x_{mag}\right) & 0 & 0 \end{bmatrix}$$

PARAMETER TUNING:

We now need to introduce some further noise considerations. For the initial noises, we add two further noises in addition to the previous ones.

$$sigmaRR = 2mm$$
$$sigmaRL = 2mm$$

Also, the 'P_{init}' matrix modifies to:

$$P_{init} = \begin{bmatrix} sigmaX^2 & 0 & 0 & 0 & 0 \\ 0 & sigmaY^2 & 0 & 0 & 0 \\ 0 & 0 & sigma\theta^2 & 0 & 0 \\ 0 & 0 & 0 & sigmaRR^2 & 0 \\ 0 & 0 & 0 & 0 & sigmaRL^2 \end{bmatrix}$$

Now, setting the initial noise of wheel radii and state noise to zero and setting '*sigmaWheels*' to an arbitrary value, we observe that the output is similar to the one in the previous case. Also, the '*P*' matrix decreases with time. Now we fine tune the value of '*sigmaWheels*' such that we get the blue line on the mahalanobis distance separating the green and red dots. Once, this parameter has been fine-tuned, we introduce the initial noises of '*+1*' on the left wheel radius and '*-1*' on the right wheel radius and then tune the 'Q_α' parameter which now looks as follows:

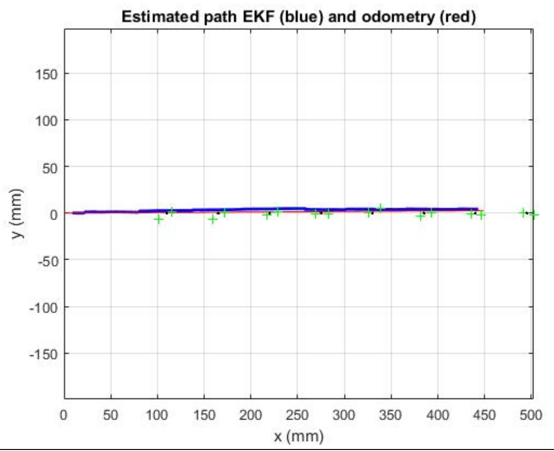
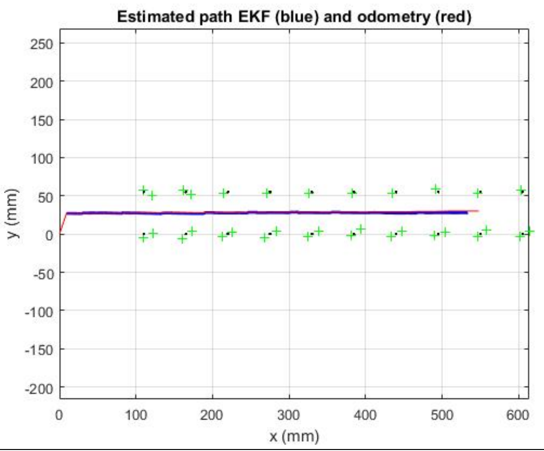
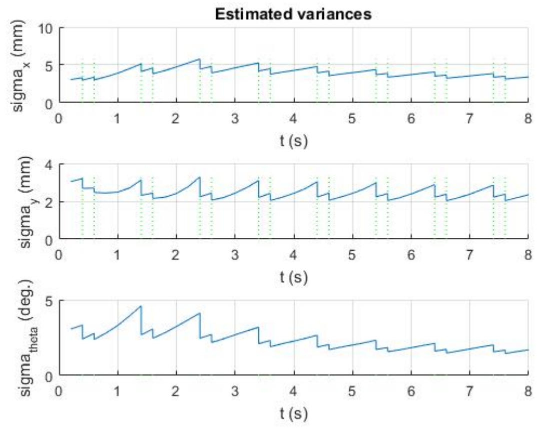
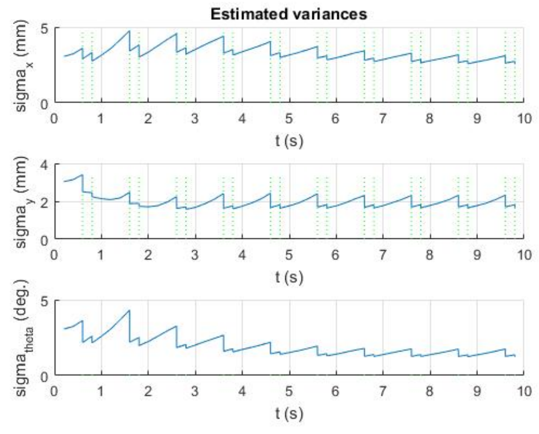
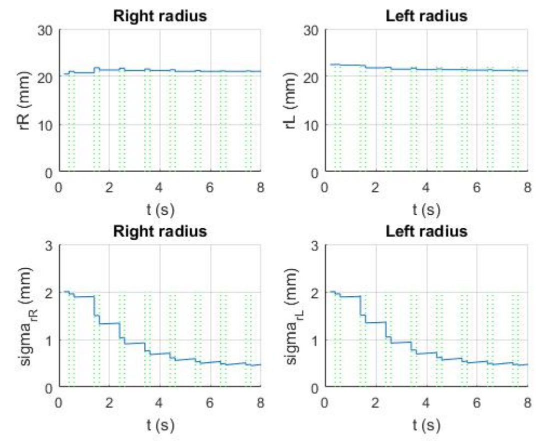
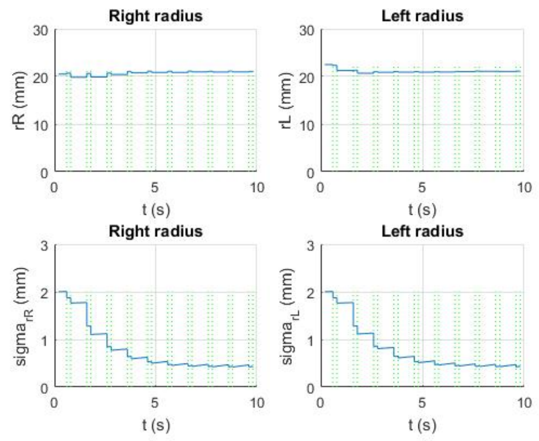
$$Q_\alpha = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & sigmaR^2 & 0 \\ 0 & 0 & 0 & 0 & sigmaL^2 \end{bmatrix}$$

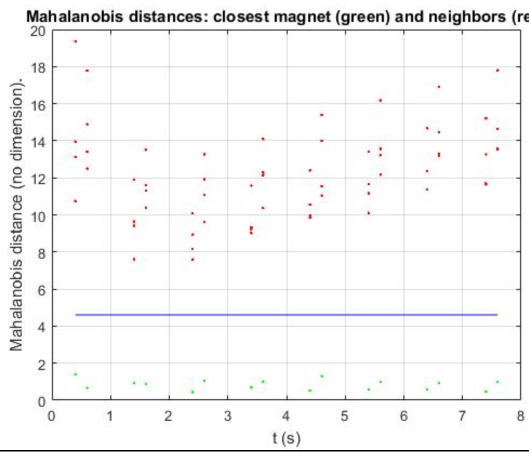
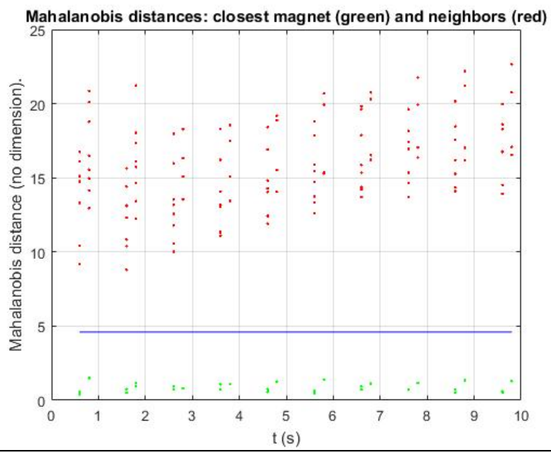
When the error variances start giving a convergence, the 'Q_α' has been tuned properly.

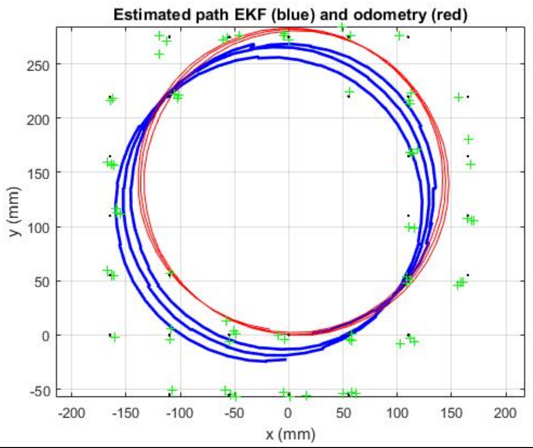
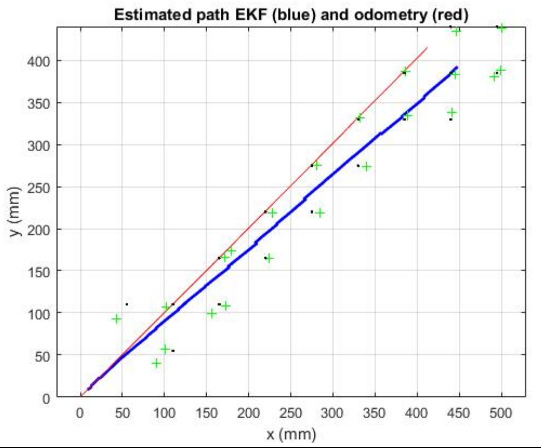
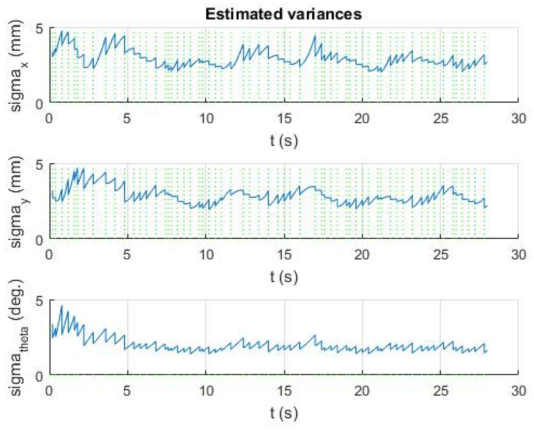
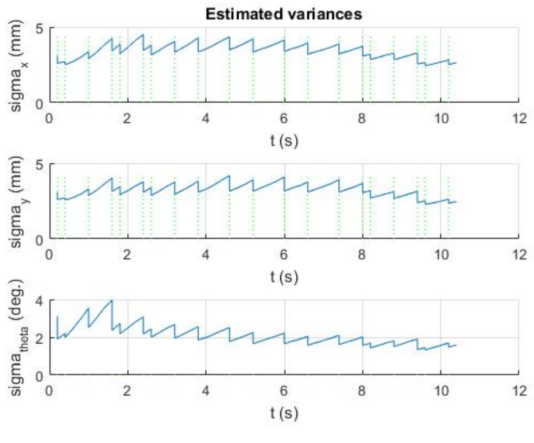
SIMULATIONS (Lab 2):

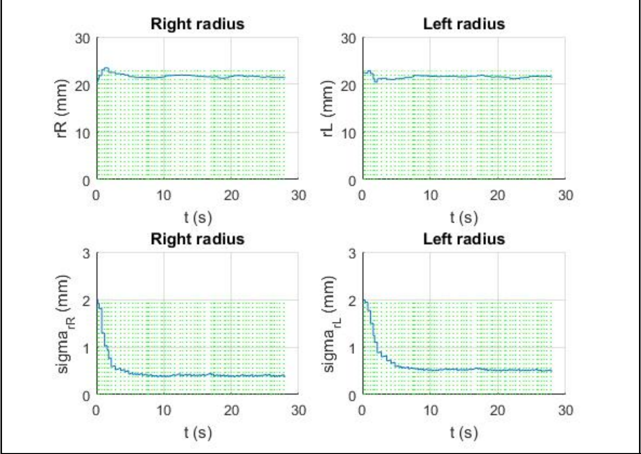
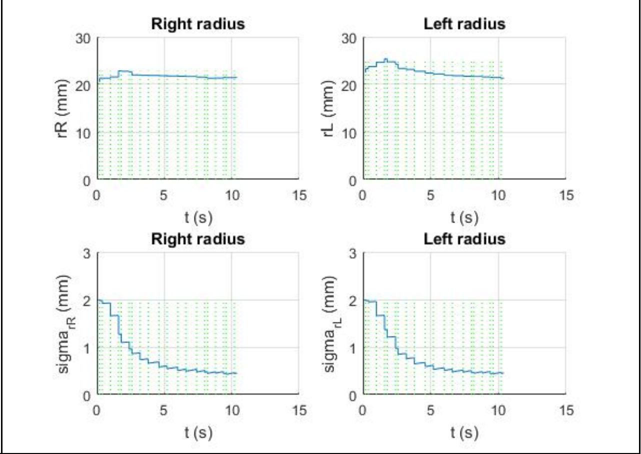
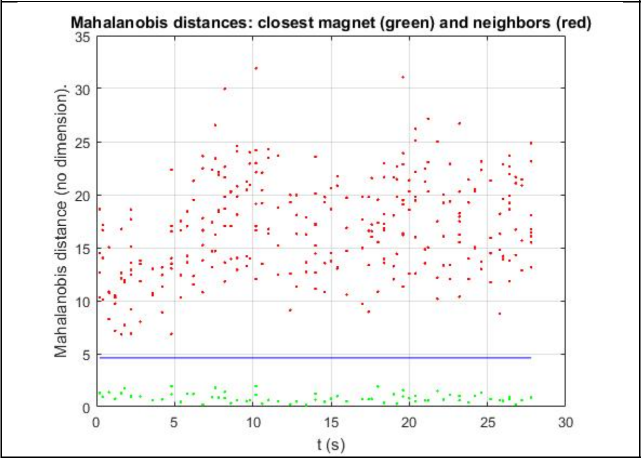
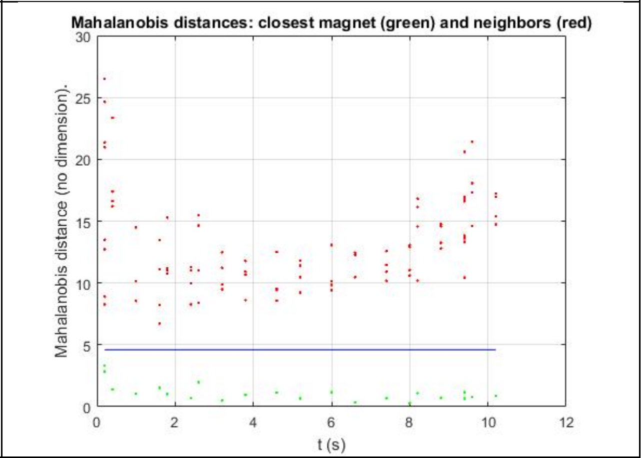
The simulations results for different paths are as follows:

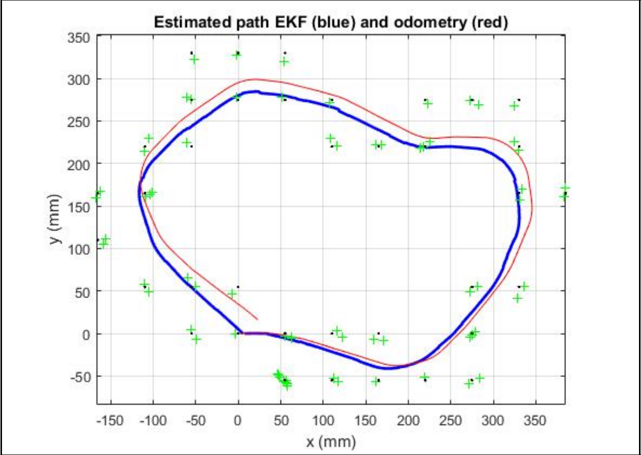
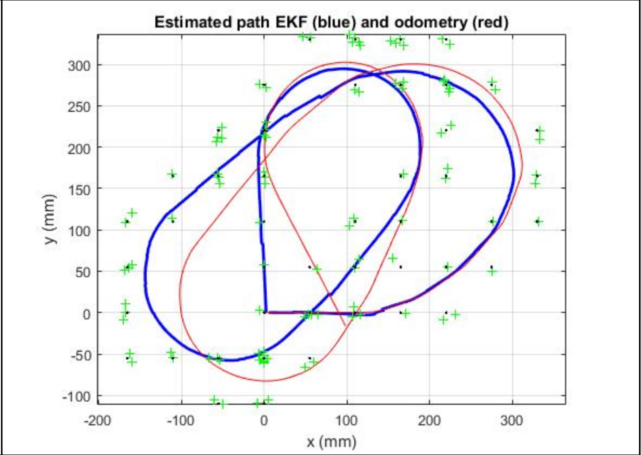
For SingmaWheels = 0.05, SigmaR = 0.1, SigmaL = 0.1

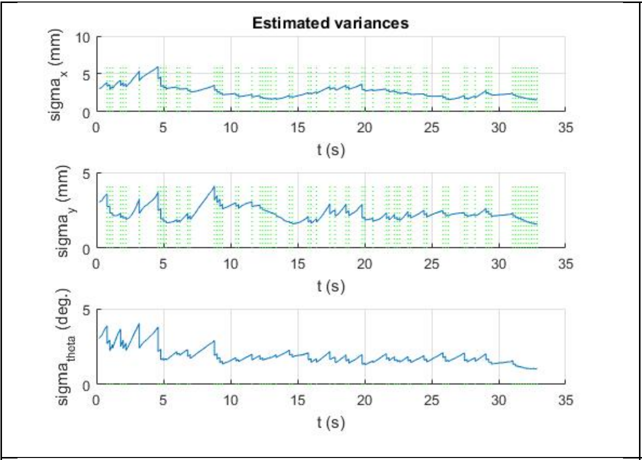
Line1 Magnets	Line2 Magnets
	
	
	

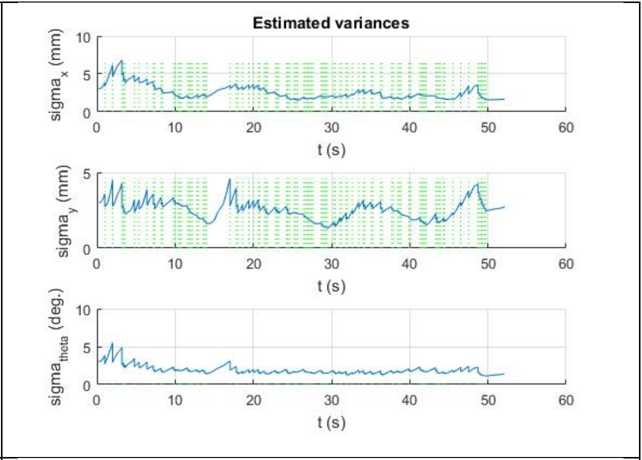
	

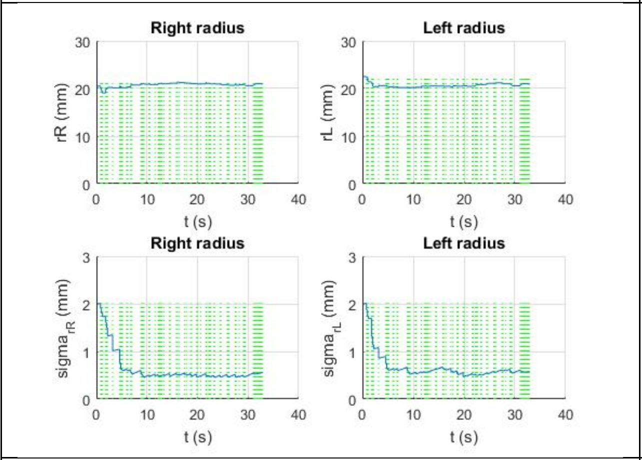
Circles	**Diagonal 45 Degrees**
	
	

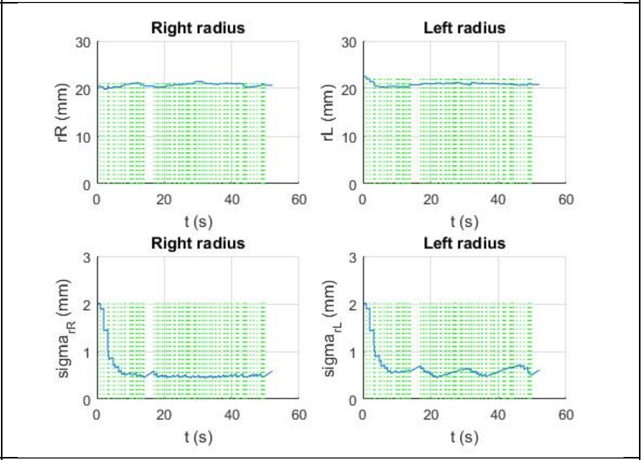
	
	

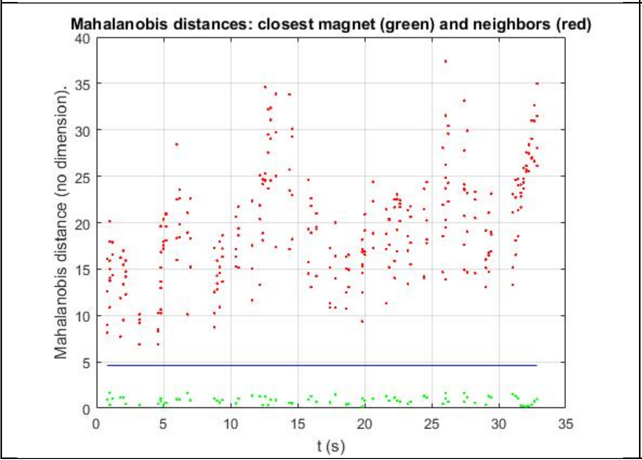
One Loop	Two Loops
	

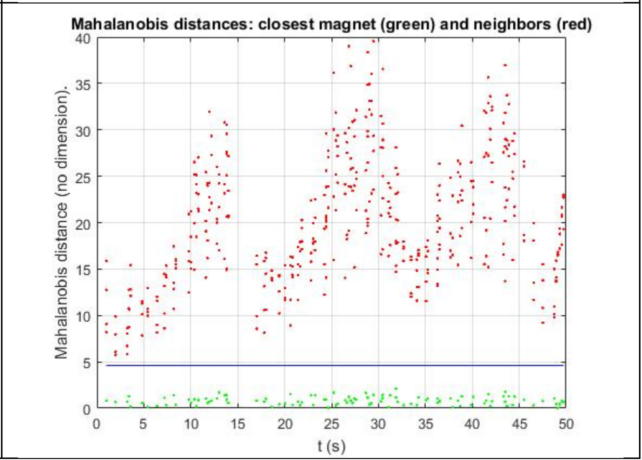












In the simulations above, we can see that the EKF trajectory reaches its original position in the loop odometries and also, the error variances show a converging behavior, demonstrating that all the parameters have been tuned properly.

In the circular odometry, it can be seen that the circles are not concentric. This is due to the effects of drag force.

CONCLUSION:

In the lab, hybrid localization technique using extended kalman filter was implemented for various trajectories. It was seen how to fine tune the EKF parameters so that the error variances can converge and a better odometry estimate can be achieved.

Later, the power of EKF was further explored with an application to an online parameter estimation scenario. The wheel radii are an important factor in the localization process especially when the odometry relies on wheel encoders. Since, the wheel radii may change during the application due to several reasons. The EKF proved to be a powerful and a robust tool in the localization process.