

# [16-833] Homework 2 : Written Report

Bharath Somayajula

March 20, 2023

## Contents

<b>1</b>	<b>Theory</b>	<b>1</b>
1.1	Q1	1
1.2	Q2	2
1.3	Q3	3
1.4	Q4	3
1.5	Q5	4
1.6	Q6	4
<b>2</b>	<b>Implementation and Evaluation</b>	<b>4</b>
2.1	Q1	4
2.2	Q2	5
2.3	Q3	5
2.4	Q4	5
2.4.1	Plot	5
2.4.2	Metrics	6
<b>3</b>	<b>Discussion</b>	<b>7</b>
3.1	Q1	7
3.2	Q2	7
3.2.1	$\sigma_x$	7
3.2.2	$\sigma_y$	8
3.2.3	$\sigma_\alpha$	8
3.2.4	$\sigma_\beta$	9
3.2.5	$\sigma_r$	9
3.2.6	Observations	9
3.3	Q3	10

## 1 Theory

### 1.1 Q1

The robot, starting at  $[x_t, y_t, \theta_t]$  moves  $d_t$  along the axis of the robot and then rotates by  $\alpha_t$  radians in the counter-clockwise direction. This means that the displacements along  $x$  and  $y$  axes and change in orientation are

$$\Delta x_t = d_t \cos(\theta_t)$$

$$\Delta y_t = d_t \sin(\theta_t)$$

$$\Delta \theta_t = \alpha_t$$

This gives us

$$\Delta \mathbf{p}_t = [\Delta x_t, \Delta y_t, \Delta \theta_t]^T$$

Therefore the new pose is,

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \Delta \mathbf{p}_t \\ \mathbf{p}_{t+1} &= \mathbf{p}_t + [d_t \cos(\theta_t), d_t \sin(\theta_t), \alpha_t]^T \end{aligned}$$

## 1.2 Q2

From the results in Q1, we know that

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \Delta \mathbf{p}_t$$

Including the uncertainty in the moving mechanism, we get

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \Delta \mathbf{p}_t + \mathbf{e}_t$$

This can be expanded as

$$\mathbf{p}_{t+1} = \begin{bmatrix} x_t + d_t \cos(\theta_t) + e_x \cos(\theta_t) - e_y \sin(\theta_t) \\ y_t + d_t \sin(\theta_t) + e_x \sin(\theta_t) + e_y \cos(\theta_t) \\ \theta_t + \alpha_t + e_\alpha \end{bmatrix}$$

The equation above is non-linear. Moreover, the noise  $\mathbf{e}_t$  is non-additive. This means the prediction step can be modeled as  $\mathbf{p}_{t+1} = g(\mathbf{p}_t, \mathbf{u}_t, \mathbf{e}_t)$ . Here,  $\mathbf{p}_t = [x_t, y_t, \theta_t]^T$  and  $\mathbf{e}_t = [e_x, e_y, e_\alpha]^T$  are groups of variables whose covariances are known

Therefore the covariance of  $\mathbf{p}_{t+1}$  can be written as

$$\Sigma_{t+1} = A \Sigma_t A^T + B \Sigma_{\mathbf{e}_t} B^T \quad (1)$$

where  $A$  and  $B$  are Jacobian matrices and  $\Sigma_{\mathbf{e}_t} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix}$ .

$A$  and  $B$  can be computed as

$$A = \begin{bmatrix} 1 & 0 & -d_t \sin(\theta_t) - e_x \sin(\theta_t) - e_y \cos(\theta_t) \\ 0 & 1 & d_t \cos(\theta_t) + e_x \cos(\theta_t) - e_y \sin(\theta_t) \\ 0 & 0 & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) & 0 \\ \sin(\theta_t) & \cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Evaluating the Jacobians at  $\mathbf{p}_t = [x_t, y_t, \theta_t]^T$  and  $\mathbf{e}_t = [0, 0, 0]^T$ , we get

$$A = \begin{bmatrix} 1 & 0 & -d_t \sin(\theta_t) \\ 0 & 1 & d_t \cos(\theta_t) \\ 0 & 0 & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) & 0 \\ \sin(\theta_t) & \cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

We can estimate the overall covariance by substituting equations 2 in equation 1.

### 1.3 Q3

Starting from pose  $\mathbf{p}_t = [x_t, y_t, \theta_t]$ , if  $r_t$  and  $\beta_t$  are range and orientations of the landmark, then the noise-free estimates of the location of the landmark are

$$\mathbf{l} = [x_t + r_t \cos(\theta_t + \beta_t), y_t + r_t \sin(\theta_t + \beta_t)]^T \quad (3)$$

where  $\mathbf{p}_t = [x_t, y_t, \theta_t]^T$  and  $\mathbf{z}_t = [\beta_t, r_t]^T$  are groups of variables with uncertainty. Therefore, the overall uncertainty in the measurement of  $\mathbf{l}$  is

$$\Sigma_{\mathbf{l}} = C \Sigma_t C^T + D \Sigma_{\mathbf{z}} D^T \quad (4)$$

where  $\Sigma_{\mathbf{z}} = \begin{bmatrix} \sigma_{\beta}^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$   $C$  and  $D$  are Jacobian matrices that can be computed as

$$C = \left. \frac{\partial \mathbf{l}}{\partial \mathbf{p}_t} \right|_{\mathbf{p}_t = \mathbf{p}_t, \mathbf{z}_t = \mathbf{z}_t} \quad \text{and} \quad D = \left. \frac{\partial \mathbf{l}}{\partial \mathbf{z}_t} \right|_{\mathbf{p}_t = \mathbf{p}_t, \mathbf{z}_t = \mathbf{z}_t}$$

Therefore,

$$C = \begin{bmatrix} 1 & 0 & -r_t \sin(\theta_t + \beta_t) \\ 0 & 1 & r_t \cos(\theta_t + \beta_t) \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} -r_t \sin(\theta_t + \beta_t) & \cos(\theta_t + \beta_t) \\ r_t \cos(\theta_t + \beta_t) & \sin(\theta_t + \beta_t) \end{bmatrix} \quad (5)$$

We can estimate the overall covariance by substituting equations 5 in equation 4.

### 1.4 Q4

From equation 3, we geometry

$$r_t \cos(\theta_t + \beta_t) = l_x - x_t \quad \text{and} \quad r_t \sin(\theta_t + \beta_t) = l_y - y_t$$

Squaring and adding the equations, we get

$$r_t^2 = (l_x - x_t)^2 + (l_y - y_t)^2$$

which translates into

$$r_t = \sqrt{(l_x - x_t)^2 + (l_y - y_t)^2} \quad (6)$$

and by dividing the equations and applying  $\text{np.arctan2}$ , we get

$$\beta_t = \text{np.arctan2} \left( \frac{l_y - y_t}{l_x - x_t} \right) - \theta_t$$

warping the angle, we get

$$\beta_t = \text{warp2pi} \left( \text{np.arctan2} \left( \frac{l_y - y_t}{l_x - x_t} \right) - \theta_t \right) \quad (7)$$

Equations 6 and 7 give us a way to estimate measurements based on the robot state.

Therefore,

$$\mathbf{z}_t = \begin{bmatrix} \text{warp2pi} \left( \text{np.arctan2} \left( \frac{l_y - y_t}{l_x - x_t} \right) - \theta_t \right) \\ \sqrt{(l_x - x_t)^2 + (l_y - y_t)^2} \end{bmatrix} \quad (8)$$

## 1.5 Q5

The equation 8 is non-linear in  $\mathbf{p}_t$  and  $\mathbf{l}$  which means  $\mathbf{z}_t = h(\mathbf{p}_t, \mathbf{l})$ . Therefore, we need to linearize the equation by computing the Jacobians  $\mathbf{H}_p$  and  $\mathbf{H}_l$  where

$$\mathbf{H}_p = \frac{\partial \mathbf{h}(\mathbf{p}_t, \mathbf{l})}{\partial \mathbf{p}_t}$$

Computing the derivatives, we get

$$\mathbf{H}_p = \begin{bmatrix} \frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & \frac{-(l_x - x_t)}{(l_x - x_t)^2 + (l_y - y_t)^2} & -1 \\ \frac{-(l_x - x_t)}{(l_x - x_t)^2 + (l_y - y_t)^2} & \frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & 0 \end{bmatrix}$$

## 1.6 Q6

The Jacobian  $\mathbf{H}_l$  can be written as

$$\mathbf{H}_l = \frac{\partial \mathbf{h}(\mathbf{p}_t, \mathbf{l})}{\partial \mathbf{l}}$$

Computing the derivatives we get

$$\mathbf{H}_l = \begin{bmatrix} \frac{-(l_y - y_t)}{(l_x - x_t)^2 + (l_y - y_t)^2} & \frac{l_x - x_t}{(l_x - x_t)^2 + (l_y - y_t)^2} \\ \frac{l_x - x_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & \frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} \end{bmatrix}$$

We need not compute Jacobians w.r.t other landmarks i.e  $\frac{\partial \mathbf{h}(\mathbf{p}_t, \mathbf{l}_i)}{\partial \mathbf{l}_j}$  since the measurements of landmarks are assumed to be independent of each other. Therefore such a Jacobian would be made of all zeros.

# 2 Implementation and Evaluation

## 2.1 Q1

The size of the landmark measurement vector is 12. Therefore, there are 6 landmarks.

## 2.2 Q2

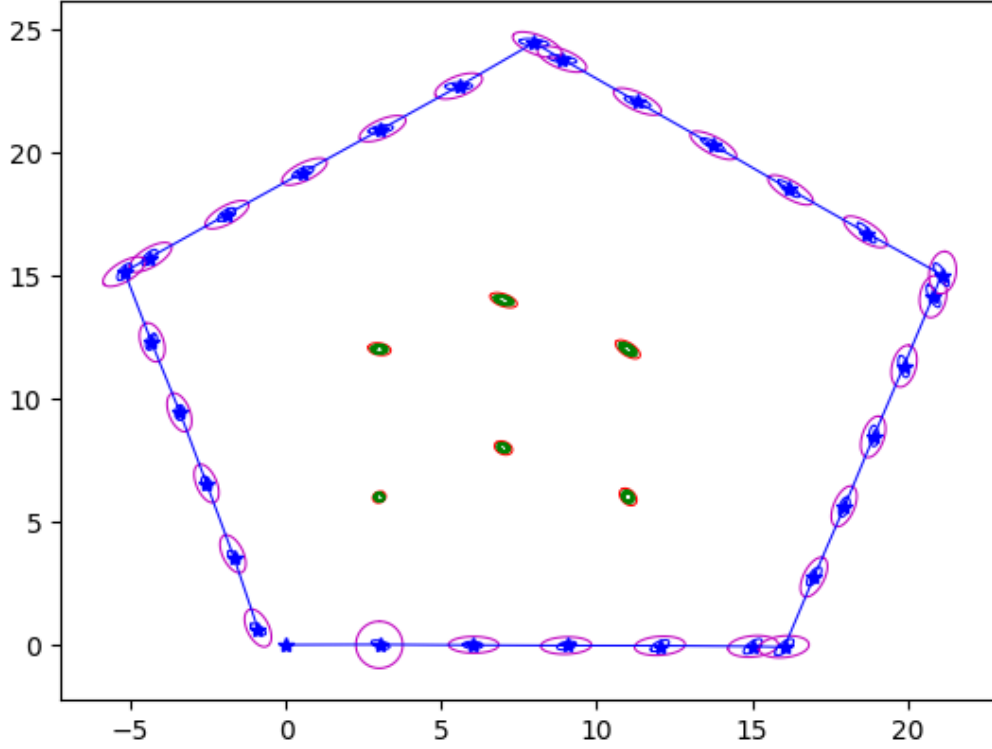


Figure 1: Visualization with default settings

## 2.3 Q3

The EKF algorithm improves the estimation of trajectory in two steps. In the prediction step, the magenta ellipse is moved towards the true location but the uncertainty is still high given the relatively larger size of magenta ellipses. In the update step, this uncertainty is reduced based on measurements given the relatively small size of blue ellipses.

Similarly, the map is improved with time. This can be observed by noticing that the size of green ellipses reduces with time implying that evidence is accumulated and leading to a drop in uncertainty estimated by the EKF algorithm.

## 2.4 Q4

### 2.4.1 Plot

The ground truth locations are shown in cyan color. The marker has a non-zero radius and hence doesn't neatly fall inside the ellipses but the centers of the ground truth markers fall inside most of the ellipses estimated by the EKF algorithm which proves that EKF algorithm estimates are very close to the ground truth positions of landmarks.

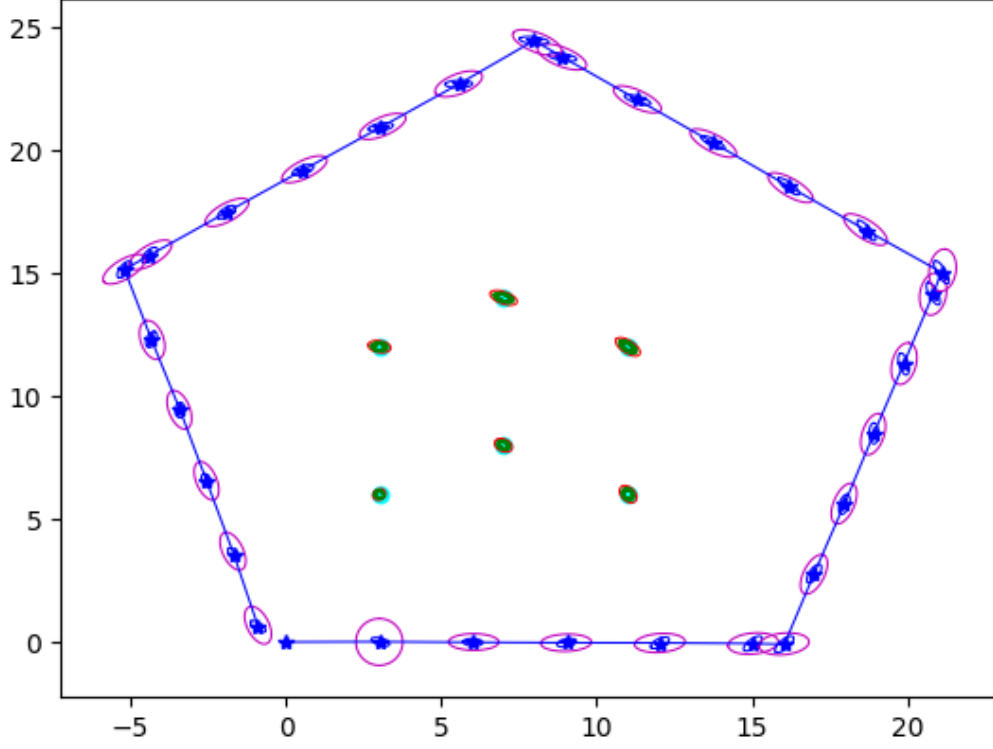


Figure 2: Visualization of ground truth(in cyan)

#### 2.4.2 Metrics

The table below summarizes the Euclidean and Mahalanobis distances for the 6 landmarks.

Index	Euclidean Distance	Mahalanobis Distance
1	0.00248222	0.00010843
2	0.00222136	0.00010859
3	0.00454603	0.00019824
4	0.00886395	0.00050701
5	0.0085716	0.00041845
6	0.00904752	0.00049835

Table 1: Metrics for the landmarks

The absolute values of these numbers indicate that the EKF algorithm converged approximately to the true land landmark locations. The Mahalanobis distance differs from the Euclidean distance since it normalizes the error using the standard deviation. For example, in 1 dimension, the Mahalanobis distance measures  $\frac{|x-\mu|}{\sigma}$  which is different from the Euclidean distance  $|x - \mu|$ .

## 3 Discussion

### 3.1 Q1

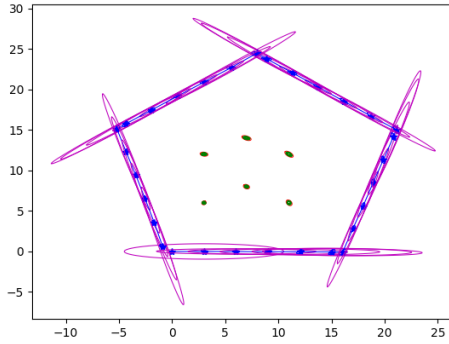
The zero-terms in the covariance matrix are non-zero by the end of the EKF algorithm due to the operations we perform in the update step where we update the covariance as  $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$ .  $K_t$  matrix is a dense matrix that informs ways in which various variables interact with each other. This causes output cross-covariances to be non-zero between various landmarks.

In the initial covariance matrix, there are two kinds of cross-covariances- the ones between pose parameters and landmark locations and the ones between various landmarks. We assumed that there are no cross covariances between pose and landmark locations. This is a reasonable assumption since the uncertainty in the initial robot location, which is potentially measured using a system such as GPS, is independent of the system used to measure the locations of key points. However, the assumption that the cross-covariances between various landmarks are zero is not reasonable since the locations of all landmarks are measured using the same method and therefore error in measurement of one landmark can lead to error in measurement of another landmark.

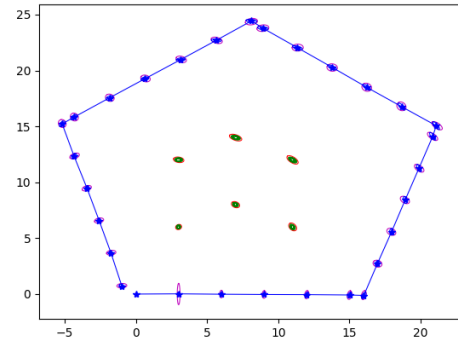
### 3.2 Q2

The plots below show the impact of increasing and decreasing each of the five parameters by a factor of 10.

#### 3.2.1 $\sigma_x$



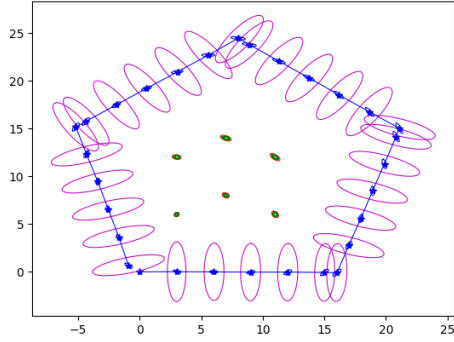
(a) Increased by a factor of 10



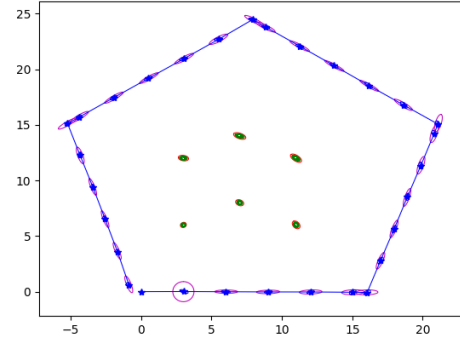
(b) Decreased by a factor of 10

Figure 3: Effect of changing  $\sigma_x$

### 3.2.2 $\sigma_y$



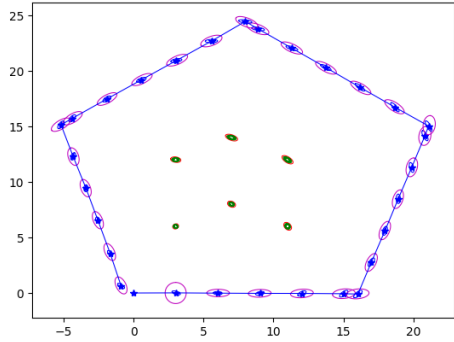
(a) Increased by a factor of 10



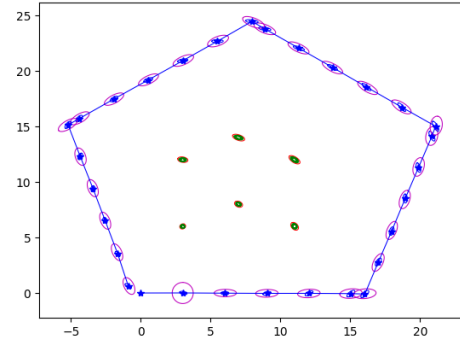
(b) Decreased by a factor of 10

Figure 4: Effect of changing  $\sigma_y$

### 3.2.3 $\sigma_\alpha$



(a) Increased by a factor of 10

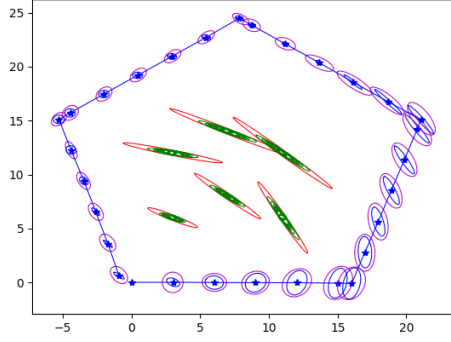


(b) Decreased by a factor of 10

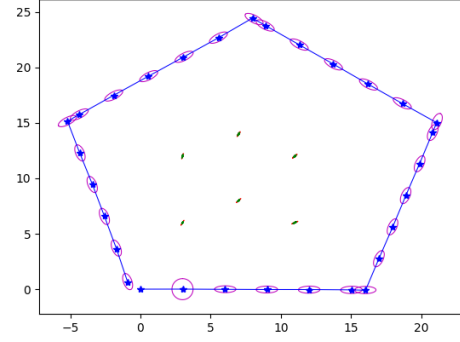
Figure 5: Effect of changing  $\sigma_\alpha$



### 3.2.4 $\sigma_\beta$



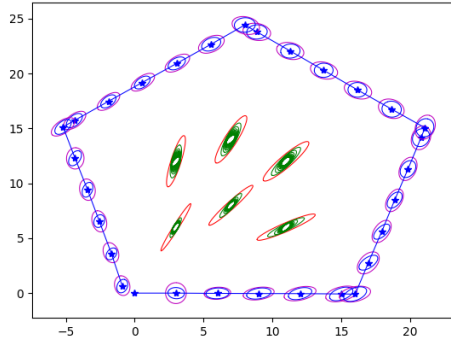
(a) Increased by a factor of 10



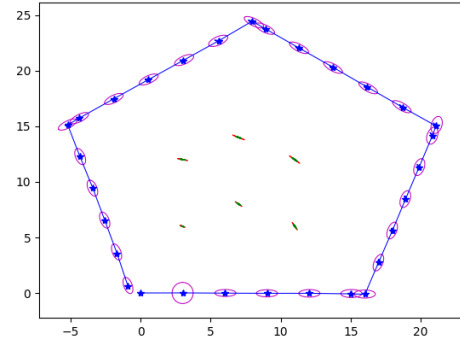
(b) Decreased by a factor of 10

Figure 6: Effect of changing  $\sigma_\beta$

### 3.2.5 $\sigma_r$



(a) Increased by a factor of 10



(b) Decreased by a factor of 10

Figure 7: Effect of changing  $\sigma_r$

### 3.2.6 Observations

From the figures above, we can observe that increases in  $\sigma_x$  and  $\sigma_y$  lead to an increase in uncertainty of robot location along the x and y axes respectively which is clearly observed in the elongated sizes of ellipses drawn around the robot location in blue and green colors. Similarly, increases in  $\sigma_r$  and  $\sigma_\beta$  cause an increase in uncertainty of landmark location as seen by elongated red and green ellipses. Change in  $\sigma_\alpha$  doesn't change the Visualization noticeably since robot orientation is not visualized in the 2D figure.

Decreasing the values by a factor of 10 has the opposite effect and uncertainties decrease as seen in the reduced sizes of ellipses in the figures above.

### 3.3 Q3

1. The number of landmarks might increase unbounded in real-world problems but the number of *visible* landmarks at any given instance is limited. Therefore, we can update the means and covariances of only the landmarks that are visible. This can significantly speed up the process.
2. If the number of landmarks visible at any time step is too large, we can select top-k landmarks and update their means and covariances alone where k is a parameter that can be selected to manage computational resources. One metric that can be used to select top-k landmarks is to sort the landmarks based on covariances and select the landmarks with the greatest degree of uncertainty for updation.