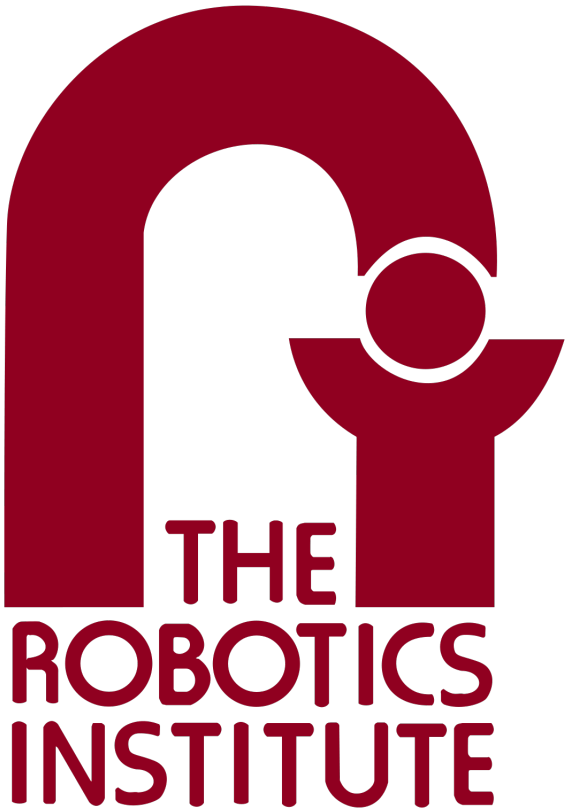
16-833A Robot Localisation and Mapping, Spring 2023
Homework 2: SLAM using Extended Kalman Filter (EKF-SLAM)

Sushanth Jayanth

Andrew ID: sushantj

May 4, 2023



Contents

Introduction to EKF

In this assignment, we use EKF to localize a robot and landmarks in 2D space. The state space of the robot and landmarks are therefore fused into one large state vector.

$$p_t = (\underbrace{x, y, \theta}_{\text{robot's pose}}, \underbrace{l_{1,x}, l_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{l_{n,x}, l_{n,y}}_{\text{landmark n}})^T$$

The above state vector captures **robot pose and landmark position in global coordinates**. Robot pose has three variables (x,y,theta) and landmark position has two variables (x,y) for each landmark. The robot can move only along the x-axis of the robot frame defined as d_t and rotate by α. This is captured in the diagram below:

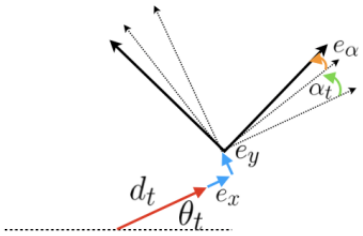


Figure 1: Robot Motion Model

As seen above, the movement has some noise. This is the noise introduced due to **control (process noise)** and this alone effects the robot's pose. Another type of noise exists in the laser rangefinder which gives the landmark locations. This is the **measurement noise**. Since both control and measurement noise affects the robot state, it is easy to maintain a combined covariance matrix P. The below equation shows how the state vector relates to the mean and covariance matrices.

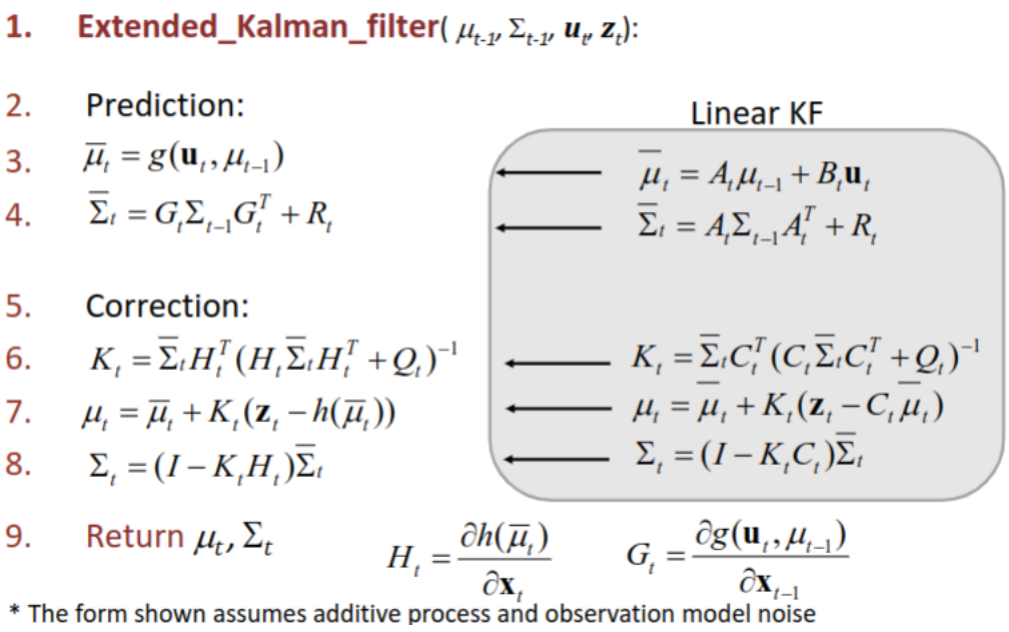
$$\underbrace{\begin{pmatrix} x_t \\ l_1 \\ \vdots \\ l_n \end{pmatrix}}_{\mu} \underbrace{\begin{pmatrix} \Sigma_{x_t x_t} & \Sigma_{x_t l_1} & \cdots & \Sigma_{x_t l_n} \\ \Sigma_{l_1 x_t} & \Sigma_{l_1 l_1} & \cdots & \Sigma_{l_1 l_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{l_n x_t} & \Sigma_{l_n l_1} & \cdots & \Sigma_{l_n l_n} \end{pmatrix}}_{\Sigma}$$

$$x_t = \text{robot pose} = (x, y, \theta)$$

In the EKF algorithm we will update parts the above covariance matrix in different steps:

- Given Control Reading : we update the $\Sigma_{x_t x_t}$ primarily plus the first row and first column elements
- Given Sensor Reading : we update the whole covariance matrix

However, prior to every update step, we will also make predictions on where the robot pose or landmark pose is supposed to be and then compare that to the measurement during our update/correction steps (lines 6,7,8 in EKF algorithm below) and (1)



1 Theory

1.1 Pose Prediction in Noise Free Case

$$p_{t+1} = p_t + \begin{bmatrix} d_t \cdot \cos(\theta_t) \\ d_t \cdot \sin(\theta_t) \\ \alpha_t \end{bmatrix} \tag{1}$$

$$p_{t+1} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} d_t \cdot \cos(\theta_t) \\ d_t \cdot \sin(\theta_t) \\ \alpha_t \end{bmatrix} \tag{2}$$

1.2 Pose Prediction with Uncertainty

$$p_{t+1} = \begin{bmatrix} x_t + d_t \cdot \cos(\theta_t) \\ y_t + d_t \cdot \sin(\theta_t) \\ \theta_t + \alpha_t \end{bmatrix} + \begin{bmatrix} e_y \sin(\theta_t) \\ e_x \cos(\theta_t) \\ e_\alpha \end{bmatrix} \tag{3}$$

The above equation can be represented as a non-linear function $g(x_t, u_t)$ with some added noise ϵ_t

$$p_{t+1} = g(x_t, u_t) + \epsilon_t \tag{4}$$

The covariance of the predicted pose p_{t+1} is defined as:

$$\Sigma_{t+1} = G_t \cdot \Sigma_t \cdot G_t^T + V_t \cdot Q_t \cdot V_t^T \tag{5}$$

G_t = Jacobian of $g(x_t, u_t)$ with respect to robot pose

$$G_t = \left.\frac{\partial g(x_t, u_t)}{\partial p}\right|_{p_t} \tag{6}$$

$$= \begin{bmatrix} \frac{\partial g_1}{\partial x_t} & \frac{\partial g_1}{\partial y_t} & \frac{\partial g_1}{\partial \theta_t} \\ \frac{\partial g_2}{\partial x_t} & \frac{\partial g_2}{\partial y_t} & \frac{\partial g_2}{\partial \theta_t} \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} 1 & 0 & -d_t \cdot \sin(\theta_t) \\ 0 & 1 & d_t \cdot \cos(\theta_t) \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

V_t = Transform mapping error in robot frame to local frame (also can be computed as the Jacobian of $g(x_t, u_t)$ with respect to e_x, e_y, e_α

Here we define V_t as a rotation matrix (about the z-axis)

$$V_t = \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) & 0 \\ \sin(\theta_t) & \cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Q_t is the Control noise.

$$Q_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{10}$$

1.3 Estimate Landmark Position

Given the range r and bearing β readings from a laser rangefinder, we can estimate the location of landmarks in the global frame given a known robot state. We will use this as our measurement prediction model $h(p_t, \beta, r)$

$$\begin{bmatrix} lx \\ ly \end{bmatrix} = \begin{bmatrix} x_t + r \cdot \cos(\beta + \theta_t) \\ y_t + r \cdot \sin(\beta + \theta_t) \end{bmatrix} \tag{11}$$

The noise in measurement of the rangefinder is given below.

$$\eta_\beta \sim \mathcal{N}(0,\, \sigma_\beta^2)\,. \tag{12}$$

$$\eta_r \sim \mathcal{N}(0,\, \sigma_r^2)\,. \tag{13}$$

The predicted covariance in the landmark measurement is shown below.

$$\Sigma_{l_{t+1}} = G_{l_t} \cdot \Sigma_{l_t} \cdot G_{l_t} + L_{l_t} \cdot R_t \cdot L_{l_t}^T \tag{14}$$

Where G_{l_t} = Jacobian of the measurement model w.r.t the robot pose

$$G_{l_t} = \left.\frac{\partial l}{\partial p}\right|_{p_t} \tag{15}$$

$$= \begin{bmatrix} \frac{\partial l_1}{\partial x_t} & \frac{\partial l_1}{\partial y_t} & \frac{\partial l_1}{\partial \theta_t} \\ \frac{\partial l_2}{\partial x_t} & \frac{\partial l_2}{\partial y_t} & \frac{\partial l_2}{\partial \theta_t} \end{bmatrix} \tag{16}$$

$$= \begin{bmatrix} 1 & 0 & -r \cdot \sin(theta_t + \beta) \\ 0 & 1 & r \cdot \cos(theta_t + \beta) \end{bmatrix} \tag{17}$$

L_{l_t} = Jacobian of the measurement model w.r.t the range and bearing

$$L_{l_t} = \left.\frac{\partial l}{\partial z}\right|_{p_t} \tag{18}$$

$$= \begin{bmatrix} \frac{\partial l_1}{\partial \beta} & \frac{\partial l_1}{\partial r} \\ \frac{\partial l_2}{\partial \beta} & \frac{\partial l_2}{\partial r} \end{bmatrix} \tag{19}$$

$$L_{l_t} = \begin{bmatrix} -r \cdot \sin(theta_t + \beta) & \cos(theta_t + \beta) \\ r \cdot \cos(theta_t + \beta) & \sin(theta_t + \beta) \end{bmatrix} \tag{20}$$

R_t is the measurement noise.

$$R_t = \begin{bmatrix} \sigma_\beta^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \tag{21}$$

1.4 Bearing and Range from Measurement

Using predicted state vector p_t (contains robot pose and landmark) for the **j'th** landmark, the bearing β and range r estimate for the **j'th** landmark is captured as $h(p_t, j)$

Where $h(p_t, j)$ = measurement model

$$h(p_t, j) = \begin{bmatrix} \beta \\ r \end{bmatrix} = \begin{bmatrix} \text{warp2pi}\left(\arctan 2\left(l_y^j - y_t, l_x^j - x_t\right) - \theta_t\right) \\ \sqrt{\left(l_x^j - x_t\right)^2 + \left(l_y^j - y_t\right)^2} \end{bmatrix} \tag{22}$$

1.5 Jacobian of Measurement w.r.t Robot Pose (H_p)

Jacobian of $h(p_t, j)$ w.r.t to robot pose is defined below.

$$H_p = \left.\frac{\partial h}{\partial p}\right|_{p_t} \tag{23}$$

$$= \begin{bmatrix} \frac{\partial h_1}{\partial x_t} & \frac{\partial h_1}{\partial y_t} & \frac{\partial h_1}{\partial \theta_t} \\ \frac{\partial h_2}{\partial x_t} & \frac{\partial h_2}{\partial y_t} & \frac{\partial h_2}{\partial \theta_t} \end{bmatrix} \tag{24}$$

$$\therefore H_p = \begin{bmatrix} \frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & \frac{-(l_x - x_t)}{(l_x - x_t)^2 + (l_y - y_t)^2} & -1 \\ \frac{-(l_x - x_t)}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} & \frac{-(l_y - y_t)}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} & 0 \end{bmatrix} \tag{25}$$

1.6 Jacobian of Measurement w.r.t Landmark Position ($\mathbf{H}_l$)

Jacobian of $h(p_t, j)$ w.r.t to **j'th** landmark (l_x and l_y) is defined below.

$$H_l = \left.\frac{\partial h}{\partial l}\right|_{p_t} \tag{26}$$

$$= \begin{bmatrix} \frac{\partial h_1}{\partial l_x^j} & \frac{\partial h_1}{\partial l_y^j} \\ \frac{\partial h_2}{\partial l_x^j} & \frac{\partial h_2}{\partial l_y^j} \end{bmatrix} \tag{27}$$

$$\therefore H_l = \begin{bmatrix} \frac{-\left(l_y^j - y_t\right)}{\left(l_x^j - x_t\right)^2 + \left(l_y^j - y_t\right)^2} & \frac{l_x^j - x_t}{\left(l_x^j - x_t\right)^2 + \left(l_y^j - y_t\right)^2} \\ \frac{l_x^j - x_t}{\sqrt{\left(l_x^j - x_t\right)^2 + \left(l_y^j - y_t\right)^2}} & \frac{l_t}{\sqrt{\left(l_x^j - x_t\right)^2 + \left(l_y^j - y_t\right)^2}} \end{bmatrix} \tag{28}$$

These Jacobians are part of the 1'st order Taylor Expansion and are just the derivative of the measurement model $h(p_t, j)$ (mentioned as $h(y_t, j)$ in the textbook), where p_t is the robot state and j is the landmark visible at that time.

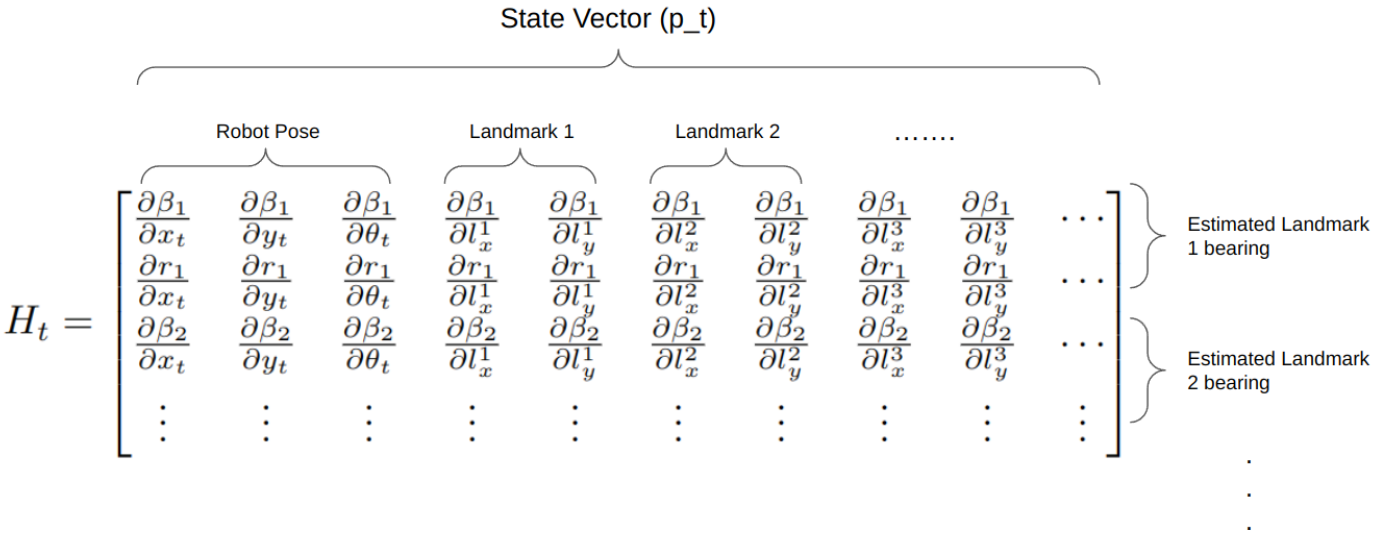
$$h\left(p_t, j\right) \approx h\left(\bar{\mu}_t, i\right) + H_t^i \tag{29}$$

Where $h\left(\bar{\mu}_t, j\right)$ is because at this stage, we would have an approximate robot pose already captured in our prediction step (where control input is used to predict robot pose)

We will use the first order derivative naming it just as H_t while evauating the elements of this jacobian only at p_t and assuming all landmarks are visible at all measurements (assumptionm mentioned in the write-up).

Therefore, simply put:

$$H_t = h'(y_t) \text{ evaluated at } \mu_t \ (\mu_t \text{ is our prediction of } p_t)$$



However, since there exists elements such as $\frac{\partial \beta_2}{\partial l_x^1} = 0$, the matrix becomes sparse as seen below.

$$H_t = \begin{bmatrix} \frac{\partial \beta_1}{\partial x_t} & \frac{\partial \beta_1}{\partial y_t} & \frac{\partial \beta_1}{\partial \theta_t} & \frac{\partial \beta_1}{\partial l_x^1} & \frac{\partial \beta_1}{\partial l_y^1} & 0 & 0 & 0 & 0 & \cdots \\ \frac{\partial r_1}{\partial x_t} & \frac{\partial r_1}{\partial y_t} & \frac{\partial r_1}{\partial \theta_t} & \frac{\partial r_1}{\partial l_x^1} & \frac{\partial r_1}{\partial l_y^1} & 0 & 0 & \frac{\partial r_1}{\partial l_x^3} & 0 & \cdots \\ \frac{\partial \beta_2}{\partial x_t} & \frac{\partial \beta_2}{\partial y_t} & \frac{\partial \beta_2}{\partial \theta_t} & 0 & 0 & \frac{\partial \beta_2}{\partial l_x^2} & \frac{\partial \beta_2}{\partial l_y^2} & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{30}$$

2 Implementation and Evaluation

Based on the theory described above, we will implement EKF in the following steps:

1. At every step below, we will use the state vector which consists of both robot pose and landmark positions:

$$\text{state vector} = X = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ l_x^1 \\ l_y^1 \\ l_x^2 \\ l_y^2 \\ \vdots \end{bmatrix}$$

2. **init_measure:** Given the first reading of measurements ($\beta and \theta$) we **estimate the landmark position(mean) and the covariance** involved in that estimate as shown in section 1.3:

$$mean = \begin{bmatrix} lx \\ ly \end{bmatrix} = \begin{bmatrix} x_t + r \cdot \cos(\beta + \theta_t) \\ y_t + r \cdot \sin(\beta + \theta_t) \end{bmatrix}$$

$$covariance = \Sigma_{l_{t+1}} = G_{l_t} \cdot \Sigma_{l_t} \cdot G_{l_t} + L_{l_t} \cdot R_t \cdot L_{l_t}^T$$

3. **predict:** Given control inputs to the robot, we find robot pose (mean) and covariance as shown in section 1.2. We ignore the noise for mean calculation (noise is accounted for in covariance calculation)

$$mean = p_{t+1} = \begin{bmatrix} x_t + d_t \cdot \cos(\theta_t) \\ y_t + d_t \cdot \sin(\theta_t) \\ \theta_t + \alpha_t \end{bmatrix}$$

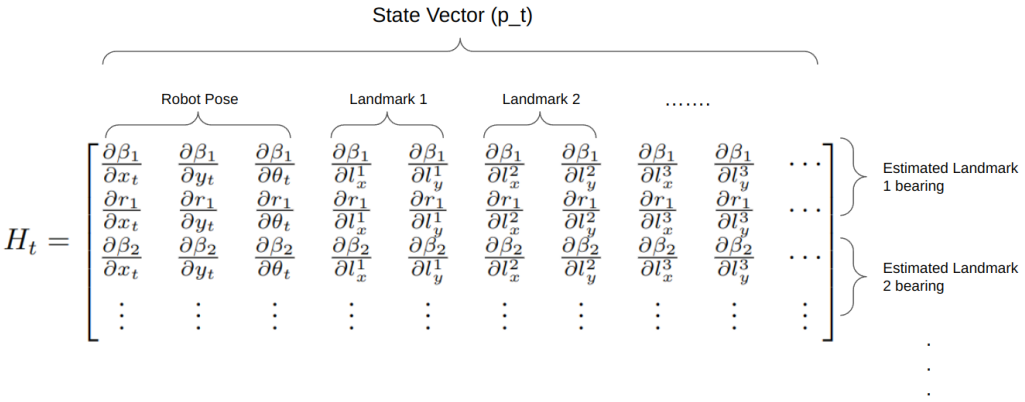
$$covariance = \Sigma_{t+1} = G_t \cdot \Sigma_t \cdot G_t^T + V_t \cdot Q_t \cdot V_t^T$$

4. **update:** Already having estimate landmark locations and estimate robot pose, we check the difference between actual sensor reading and predicted sensor reading.

   The sensor reading is predicted as in 1.4:

$$h(p_t, j) = \begin{bmatrix} \beta \\ r \end{bmatrix} = \begin{bmatrix} \text{warp2pi}\left(\arctan 2\left(l_y^j - y_t, l_x^j - x_t\right) - \theta_t\right) \\ \sqrt{\left(l_x^j - x_t\right)^2 + \left(l_y^j - y_t\right)^2} \end{bmatrix} \tag{31}$$

We linearize the above predicted sensor reading model about the last predicted robot pose $\bar{\mu_{p_t}}$. This linearization is just a 1'st order Taylor approximation as seen in:



$$H_t = \begin{bmatrix} \frac{\partial \beta_1}{\partial x_t} & \frac{\partial \beta_1}{\partial y_t} & \frac{\partial \beta_1}{\partial \theta_t} & \frac{\partial \beta_1}{\partial l_x^1} & \frac{\partial \beta_1}{\partial l_y^1} & \frac{\partial \beta_1}{\partial l_x^2} & \frac{\partial \beta_1}{\partial l_y^2} & \frac{\partial \beta_1}{\partial l_x^3} & \frac{\partial \beta_1}{\partial l_y^3} & \cdots \\ \frac{\partial r_1}{\partial x_t} & \frac{\partial r_1}{\partial y_t} & \frac{\partial r_1}{\partial \theta_t} & \frac{\partial r_1}{\partial l_x^1} & \frac{\partial r_1}{\partial l_y^1} & \frac{\partial r_1}{\partial l_x^2} & \frac{\partial r_1}{\partial l_y^2} & \frac{\partial r_1}{\partial l_x^3} & \frac{\partial r_1}{\partial l_y^3} & \cdots \\ \frac{\partial \beta_2}{\partial x_t} & \frac{\partial \beta_2}{\partial y_t} & \frac{\partial \beta_2}{\partial \theta_t} & \frac{\partial \beta_2}{\partial l_x^1} & \frac{\partial \beta_2}{\partial l_y^1} & \frac{\partial \beta_2}{\partial l_x^2} & \frac{\partial \beta_2}{\partial l_y^2} & \frac{\partial \beta_2}{\partial l_x^3} & \frac{\partial \beta_2}{\partial l_y^3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

We use the above huge matrix H_t in the EKF update equations (updating entire state vector (robot pose and **all** landmarks at once, textbook was written before vectorization was popular and does it weirdly instead).

$$K_t = \bar{\Sigma}_t H_t^T \left(H_t \bar{\Sigma}_t H_t^T + Q_t \right)^{-1} \tag{32}$$

$$\begin{aligned} \mu_t &= \bar{\mu}_t + K_t \left(z_t - \hat{z}_t \right) \\ \Sigma_t &= \left(I - K_t H_t \right) \bar{\Sigma}_t \end{aligned} \tag{33}$$

2.1 Questions

1. What is the fixed number of landmarks being observed over the entire sequence? **Ans.** There are 6 landmarks being observed at all times

2. Attach a clear figure of your visualization in the write up once all the steps are finished. **Ans.**

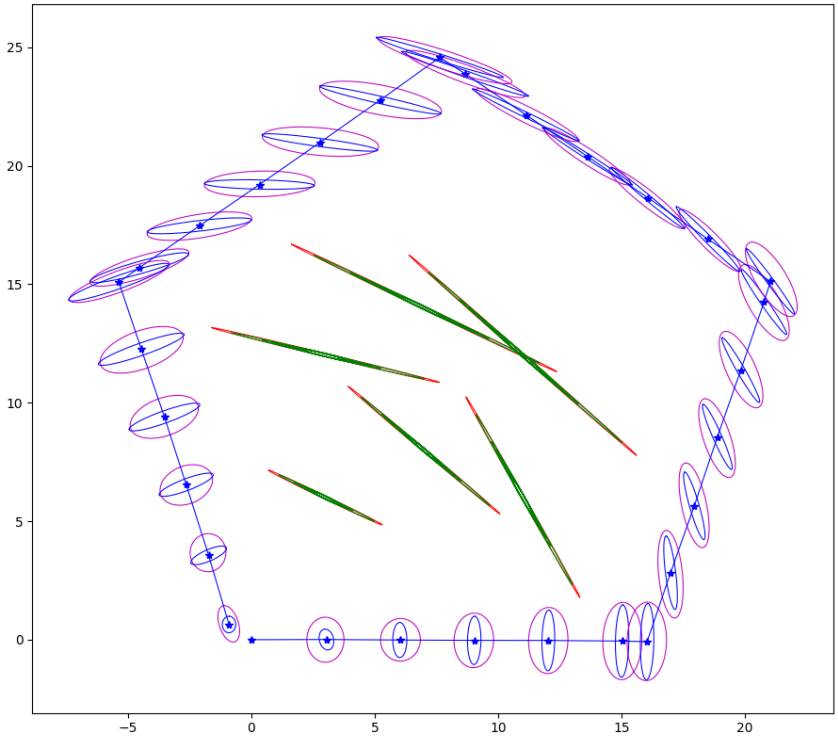


Figure 2: EKF output

3. Describe how EKF-SLAM improves the estimation of both the trajectory and the map by comparing the uncertainty ellipses. **Ans.** The uncertainty ellipses for robot pose expands after every prediction step and shrinks after every update step. This is expected since the measurement allows us to correct the prediction and decrease the variance (or covariance) in prediction.

$$\Sigma_t = (I - K_t H_t)\,\bar{\Sigma}_t$$

The above equation shows that predicted covariance $\bar{\Sigma}_t$ is always multiplied with $(I - K_t H_t)$, where:

$$0 < (I - K_t H_t) < I \tag{34}$$

Hence, the covariance always shrinks for robot pose after the initial expansion in prediction step.

For the landmark position covariance, we only predict the landmark once (during initialization) and only keep updated it as per measurements. Hence it does not expand and shrink at every update step, instead it keeps shrinking with increased measurements.

4. Plot the ground truth positions of the landmarks in the output figure. Is each of them inside the smallest corresponding ellipse? What does that mean?
Compute and report the Euclidean and Mahalanobis distances of each landmark estimation with respect to the ground truth. What do the numbers tell you?

Ans.

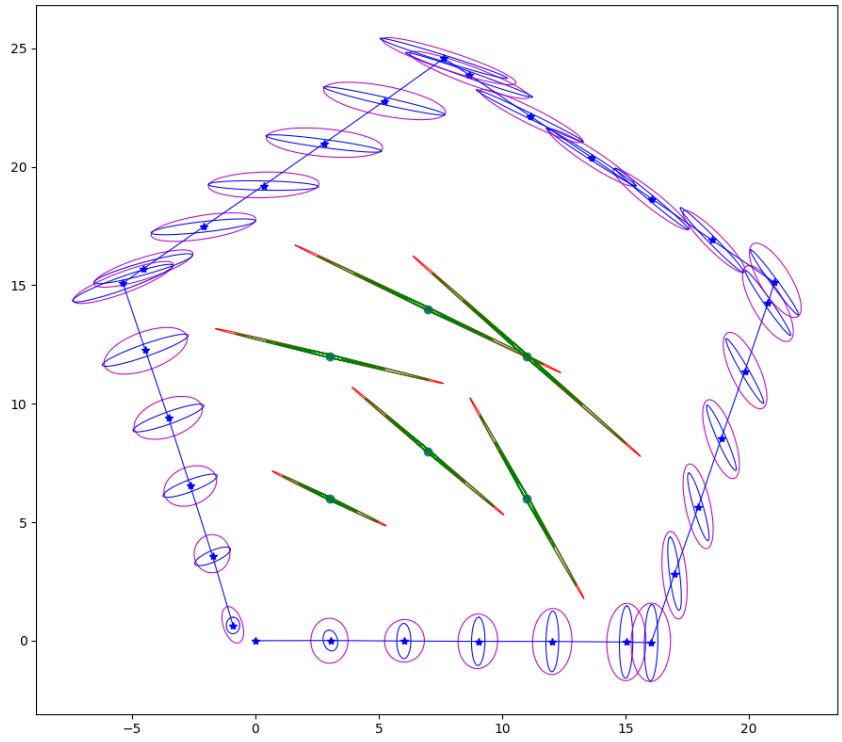


Figure 3: EKF output with ground truth of landmarks shown

Each of the ground truth points are inside the smallest ellipse. This means that the estimate of the landmark positions was accurate and the mean of the landmark estimate gaussian was very close to the ground truth (from visual inspection only)

The Mahalanobis and Euclidean distance is described below:

Landmarks	L1	L2	L3	L4	L5	L6
Ground Truth	(3,6)	(3,12)	(7,8)	(7,14)	(11,6)	(11,12)
Euclidean Distance	0.0579	0.1051	0.0869	0.1277	0.1032	0.1350
Mahalanobis Distance	0.0874	0.3156	0.0809	0.1109	0.3694	0.2720

Figure 4: Distance Measures from Landmarks

The Mahalanobis distance is useful to mesure distances along axes (like x, y, θ in our case) which are not decoupled, i.e. they have non-zero elements in the off-diagonal parts of the covariance matrix. In such scenarios, euclidean distance would not be a good measure.

3 Discussion

1. Explain why the zero terms in the initial landmark covariance matrix become non-zero in the final state covariance matrix (print out the final P matrix to check it). Additionally, when setting the initial value for the full covariance matrix P (line 201 in ekf slam.py) an assumption is made regarding certain cross-covariances that is not necessarily correct. Can you point out what that is?

Ans. The initial and final covariance matrices P_init and P_final are shown below:

```
Init covariance
[[ 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.    0.01  0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.59 -0.29  0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.   -0.29  0.15  0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    2.37 -0.59  0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.   -0.59  0.15  0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    1.05 -0.92  0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.   -0.92  0.8   0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    3.21 -1.61  0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.   -1.61  0.8   0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.59 -1.08  0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   -1.08  1.99  0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    2.36 -2.17]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   -2.17  1.99]]
Final covariance
[[ 0.01  0.   -0.    0.03 -0.02  0.06 -0.01  0.04 -0.03  0.07 -0.03  0.03 -0.05  0.06 -0.05]
 [ 0.    0.01 -0.01  0.06 -0.03  0.12 -0.03  0.08 -0.07  0.14 -0.07  0.06 -0.11  0.12 -0.11]
 [-0.   -0.01  0.01 -0.07  0.04 -0.14  0.04 -0.1   0.08 -0.17  0.08 -0.07  0.13 -0.14  0.13]
 [ 0.03  0.06 -0.07  0.42 -0.21  0.84 -0.21  0.56 -0.49  0.99 -0.49  0.42 -0.78  0.84 -0.77]
 [-0.02 -0.03  0.04 -0.21  0.11 -0.43  0.11 -0.28  0.25 -0.5   0.25 -0.21  0.39 -0.43  0.39]
 [ 0.06  0.12 -0.14  0.84 -0.43  1.7  -0.42  1.13 -0.99  1.99 -0.99  0.85 -1.56  1.7  -1.56]
 [-0.01 -0.03  0.04 -0.21  0.11 -0.42  0.1  -0.28  0.25 -0.49  0.24 -0.21  0.39 -0.42  0.39]
 [ 0.04  0.08 -0.1   0.56 -0.28  1.13 -0.28  0.75 -0.66  1.32 -0.66  0.56 -1.04  1.13 -1.04]
 [-0.03 -0.07  0.08 -0.49  0.25 -0.99  0.25 -0.66  0.58 -1.16  0.58 -0.5   0.91 -0.99  0.91]
 [ 0.07  0.14 -0.17  0.99 -0.5   1.99 -0.49  1.32 -1.16  2.32 -1.15  0.99 -1.83  1.99 -1.82]
 [-0.03 -0.07  0.08 -0.49  0.25 -0.99  0.24 -0.66  0.58 -1.15  0.57 -0.49  0.91 -0.99  0.91]
 [ 0.03  0.06 -0.07  0.42 -0.21  0.85 -0.21  0.56 -0.5   0.99 -0.49  0.43 -0.78  0.85 -0.78]
 [-0.05 -0.11  0.13 -0.78  0.39 -1.56  0.39 -1.04  0.91 -1.83  0.91 -0.78  1.44 -1.57  1.43]
 [ 0.06  0.12 -0.14  0.84 -0.43  1.7  -0.42  1.13 -0.99  1.99 -0.99  0.85 -1.57  1.71 -1.56]
 [-0.05 -0.11  0.13 -0.77  0.39 -1.56  0.39 -1.04  0.91 -1.82  0.91 -0.78  1.43 -1.56  1.43]]
```

Figure 5: Comparison of initial and final covariance matrices

The off-diagonal elements in the covariance matrix is set to zero and changes to non-zero elements after all the update steps. **This is because the EKF updates to correct the covariance shows that some form of correlation exists between the various measurements of robot pose and all landmarks.**

The initial assumption to set all off-diagonal elements to zero in the covariance matrix is therefore only an approximation, since noise amongst measurements and robot pose is not de-coupled as seen by the EKF updates. Another way of saying this would be that if there was no correlation between measurement noise (bearing and range) and robot pose (x,y, θ), then all off-diagonal elements should have remained zero or very close to zero even after the Kalman updates.

2. Play with the parameters (line 163-167). Modify each parameter $\sigma_x, \sigma_y, \sigma_\alpha, \sigma_\beta, \sigma_r$ at a time and fix the others. In particular, you should set each parameter to be 10 times larger/smaller to the original parameters to discuss how each parameters influence the results. Attach figures for better explanation.

Ans. The results of increase each of the variance by a factor of 10 has been shown below:

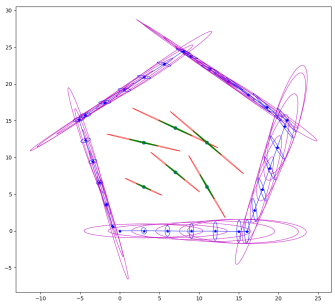


Figure 6: $\sigma_x * 10$

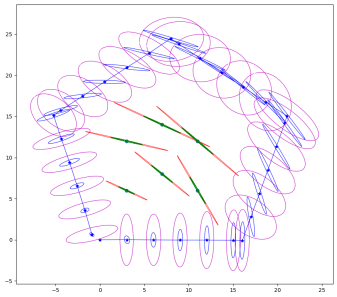


Figure 7: $\sigma_y * 10$

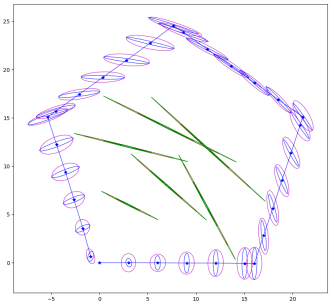


Figure 8: $\sigma_\alpha * 10$

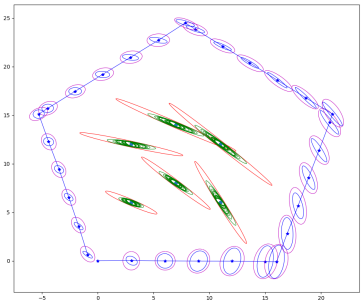


Figure 9: $\sigma_\beta * 10$

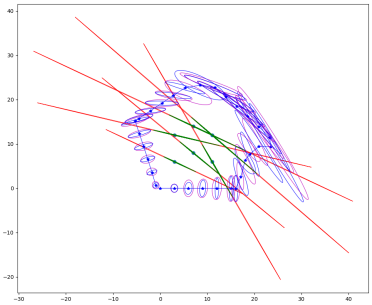


Figure 10: $\sigma_r * 10$

- The Figure 6 shows that increase in σ_x causes the robot motion to be less accurate along the robot's direction of motion. However, this has little to no effect on the landmark positions.
- Increasing σ_y in Figure 7 shows again the ellipse is larger in the y-axis of the robot's local frame. This too gets corrected well by the update step.
- Increasing σ_α seems to have little effect on the overall tracking of the robot except for some small drift in the robot motion.
- Increasing σ_β has a positive effect on the localisation of landmarks and the robot tracking as well. This indicates that our initial value for σ_β was probably not a good approximation

- Increasing σ_r causes the landmark localisation covariance to be too large and infeasible.

1. With the same set of landmarks being observed all the time, the EKF-SLAM system runs in constant time in each iteration. However, for real-world prob- lems, the total number of landmarks can grow unbounded if the robot keeps exploring new environments. This will slow down the EKF-SLAM system a lot and become a crucial problem for real-time applications. What changes can we make to the EKF-SLAM framework to achieve constant computation time in each cycle or at least speed up the process (list as many possible solutions as you can think of)?

Ans. Since only a bunch of landmarks may be seen at every section of the map, once the robot moves out and a bunch of landmarks go out of FOV, we can store these landmark locations separately and remove it from the state vector. If the landmarks appear again, we can then re-attach them to the state vector.

References

[1] Wolfram Burgard Sebastian Thrun and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.