Homework 2 - SLAM using Extended Kalman Filter (EKF-SLAM)

Jinyun Xu jinyunx

March 18, 2023

1 Theory

1.1

$$\begin{cases} x_{t+1} = x_t + d_t cos(\theta_t) \\ y_{t+1} = y_t + d_t sin(\theta_t) \\ \theta_{t+1} = \theta_t + \alpha_t \end{cases}$$

$$\mathbf{p_{t+1}} = \mathbf{p_t} + \begin{bmatrix} d_t cos(\mathbf{p_t}[\mathbf{3}]) \\ d_t sin(\mathbf{p_t}[\mathbf{3}]) \\ \alpha_t \end{bmatrix}$$

1.2

For covariance in the EKF:

$$\Sigma_{t+1} = G_t \Sigma_t G_t^T + R_t$$

where G_t is the pose Jacobian and R_t is the uncertainty in global frame.

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

The transformation matrix from the robot frame to world frame is B_t:

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

and the uncertainty matrix

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix}$$

$$R_t = B_t R B_t^T$$

The distribution of the robot state uncertainty at time $t+1$ will be:

$$\mathcal{N}(0,\, G_t \Sigma_t G_t^T + B_t R B_t^T)$$

1.3

Convert landmark position into global frame need to consider measurement noise. ω_r and ω_β are measurements error for range and bearing angle respectively.

$$\begin{bmatrix} l_x \\ l_y \end{bmatrix} = \begin{bmatrix} x_t + (r + \omega_r) cos(\theta_t + \beta + \omega_\beta) \\ y_t + (r + \omega_r) sin(\theta_t + \beta + \omega_\beta) \end{bmatrix}$$

1.4

From the division of previous equations we can get:

$$tan(\theta_t + \beta + \omega_\beta) = \frac{l_y - y_t}{l_x - x_t}$$

$$\Rightarrow \beta = wrap2pi(np.arctan(l_y - y_t, l_x - x_t) - \theta_t - \omega_\beta)$$

From the square sum of previous equations we can get:

$$(l_y - y_t)^2 + (l_x - x_t)^2 = (r + \omega_r)^2$$

$$\Rightarrow r = \sqrt{(l_y - y_t)^2 + (l_x - x_t)^2} - \omega_r$$

Therefore:

$$\begin{bmatrix} \beta \\ r \end{bmatrix} = \begin{bmatrix} wrap2pi(np.arctan(l_y - \mathbf{p_t}[\mathbf{2}], l_x - \mathbf{p_t}[\mathbf{1}]) - \mathbf{p_t}[\mathbf{3}] - \omega_\beta) \\ \sqrt{(l_y - \mathbf{p_t}[\mathbf{2}])^2 + (l_x - \mathbf{p_t}[\mathbf{1}])^2} - \omega_r \end{bmatrix}$$

1.5

The Jacobin related to robot pose will be:

$$H_p = \begin{bmatrix} \frac{\partial \beta}{\partial x_t} & \frac{\partial \beta}{\partial y_t} & \frac{\partial \beta}{\partial \alpha_t} \\ \frac{\partial r}{\partial x_t} & \frac{\partial r}{\partial y_t} & \frac{\partial r}{\partial \alpha_t} \end{bmatrix}$$

Let $\eta = (l_y - y_t)^2 + (l_x - x_t)^2$

$$H_p = \begin{bmatrix} \frac{l_y - y_t}{\eta} & -\frac{l_x - x_t}{\eta} & -1 \\ -\frac{l_x - x_t}{\sqrt{\eta}} & -\frac{l_y - y_t}{\sqrt{\eta}} & 0 \end{bmatrix}$$

1.6

The Jacobin related to landmark will be:

$$H_l = \begin{bmatrix} \frac{\partial \beta}{\partial l_x} & \frac{\partial \beta}{\partial l_y} \\ \frac{\partial r}{\partial l_x} & \frac{\partial r}{\partial l_y} \end{bmatrix}$$

Let $\eta = (l_y - y_t)^2 + (l_x - x_t)^2$

$$H_l = \begin{bmatrix} -\frac{l_y - y_t}{\eta} & \frac{l_x - x_t}{\eta} \\ \frac{l_x - x_t}{\sqrt{\eta}} & \frac{l_y - y_t}{\sqrt{\eta}} \end{bmatrix}$$

2 Implementation and Evaluation

2.1

6 landmarks are observed over the entire sequence

2.2

The following figure visualization show the trajectory and the landmarks with their covariance.

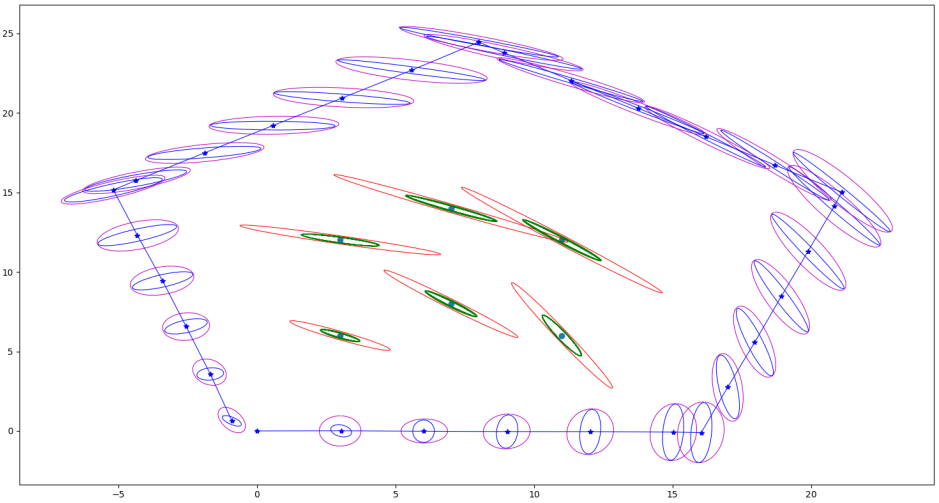


Figure 1: EKF result

2.3

Compare the range of magenta and blue ellipses and red and green ellipses, we can find EKF decrease the uncertainty. The blue and green covariance ellipses are smaller than originally prediction result. Although the prediction step will increase uncertainty, we use Kalman gain during the update step to reduce uncertainty, which regulates weights of motion model measurement.

2.4

The following figure shows landmark ground truth as well. The blue points are ground truth for 6 landmarks. All of them are inside the smallest corresponding ellipse, which means the estimated position of landmarks are very accurate.

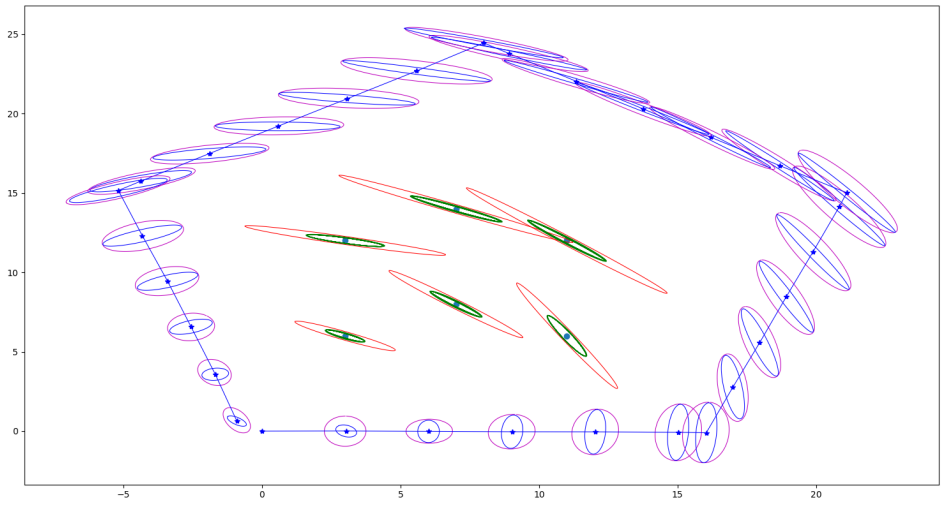


Figure 2: EKF result with landmark ground truth

The Euclidean and Mahalanobis distances of each landmark estimation with respect to the ground truth are shown in the table below. Euclidean distance reflects distance between estimated point and ground truth point. Mahalanobis distance reflects the distance between a point and the distribution. Both distances are small which means the estimated result is very close to the ground truth.

Landmark	Euclidean Distance	Mahalanobis Distance
1	0.0024196996063141417	0.05452482478163033
2	0.005824844894182079	0.06477739336726142
3	0.0014075712224309073	0.03507501051259749
4	0.003256765775521814	0.0641040546440559
5	0.0017959422214237544	0.02195203612781326
6	0.005583788669758309	0.09435980674589028

3 Discussion

3.1

In the update function, we update the covariance matrix P by the Jacobian matrix H and the Kalman gain.

During the covariance calculation, we assumed all measurements of landmark position are independent with robot pose, which is not correct for actual condition.

3.2

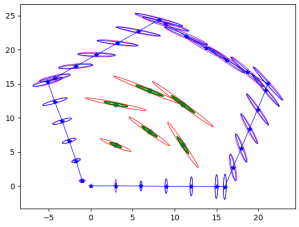
The following figure shows EKF results with different uncertainty. From sub-figure (a), (b), and (c), when σ_x, σ_y, σ_α reduce, the covariance of robot position also decrease and accuracy increase, verse vice from sub-figure (d), (e), and (f). Changing in σ_x, σ_y, σ_α don't have large effect of landmark estimation.

For σ_β and σ_r, they have influences in both robot position estimation and landmark estimation. When they are larger, both estimation are more uncertain and less accurate. When they are smaller, the estimation will converge closly to the ground truth.

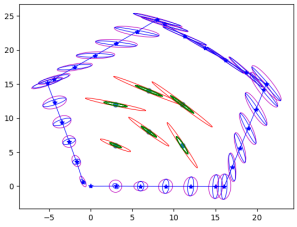
3.3

The first method is we can only hold constant number of landmarks over time. Set the constant landmark number be n. If we find more landmarks than n, just remove landmarks with highest uncertainty until the total amount back to n again.

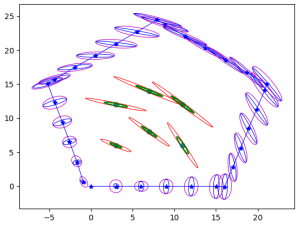
The second method is using total amount k of all landmarks in the environment to create a vector to hold all landmarks estimated position. Using constant size vectorized method will save time compared with looping method.



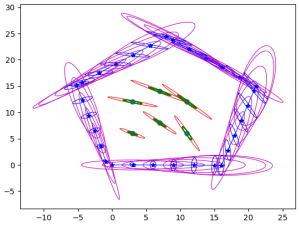
(a) $0.1\sigma_x$



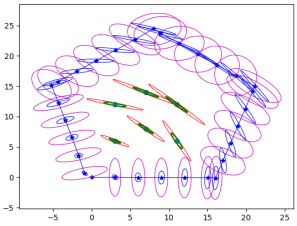
(b) $0.1\sigma_y$



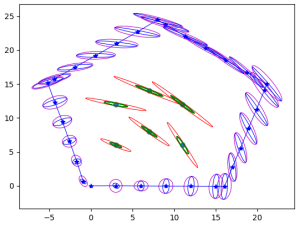
(c) $0.1\sigma_\alpha$



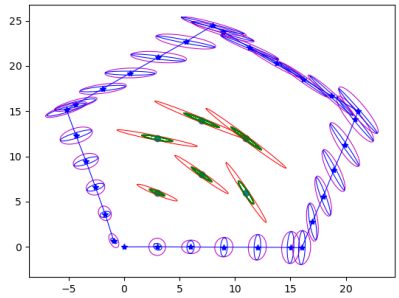
(d) $10\sigma_x$



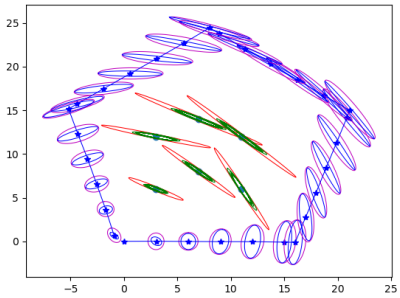
(e) $10\sigma_y$



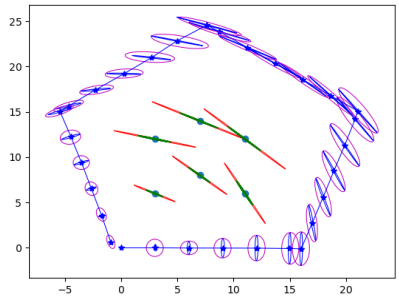
(f) $10\sigma_\alpha$



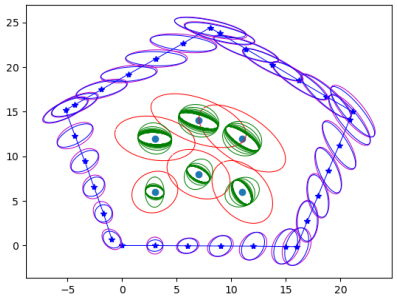
(g) $0.1\sigma_\beta$



(h) $10\sigma_\beta$



(i) $0.1\sigma_r$



(j) $10\sigma_r$

Figure 3: Different uncertainty