University of California San Diego

CSE276A: Introduction to Robotics

HW3: SLAM based on EKF

November 15th, 2023

Author	Student ID
Mazeyu Ji	A59023027
Zejia Wu	A59026752

Link to the Video:

https://drive.google.com/file/d/1psMzvefEMNy5pa5hADf5doajydOedV3U/view?usp=sharing

1 Environmental Setup

We set up a square area measuring 1.5 meters by 1.5 meters. The origin of the world coordinates is located at the exact center of the square field. Two AprilTags are placed on each side of the square. The coordinates of the eight AprilTags are $(0.75, 0.375)$, $(0.75, -0.375)$, $(0.375, -0.75)$, $(0.375, 0.75)$, $(-0.375, 0.75)$, $(-0.75, 0.375)$, $(-0.75, -0.375)$, and $(-0.375, -0.75)$, which is shown as in Figure 1. We set $(-0.375, -0.375, 0)$ as the initial pose of the robot. However, there are errors in the actual placement of the robot. After completing the mapping, we can obtain the initial pose error of the robot through an optimization algorithm.

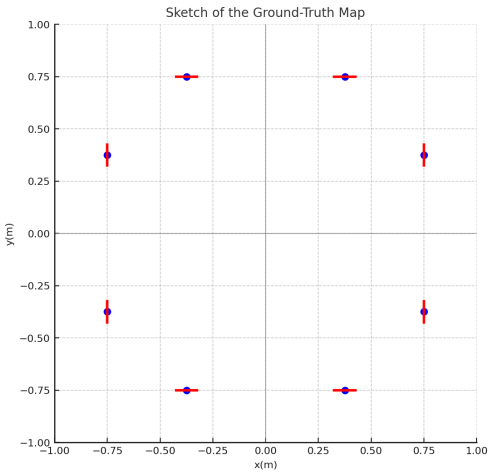


Figure 1: The sketch of our ground-truth map in the top-down view layout.

2 The Algorithm

2.1 Basic Setups

We assume that the state vector of the robot is

$$\mu_t = [x, y, \theta, \mu_{1,x}, \mu_{1,y}, ..., \mu_{n,x}, \mu_{n,y}]^T, \tag{1}$$

where the first three variables represent the pose of the robot, and every two variables thereafter represent the position of a landmark (center point). Initially, the state of the robot is $\mu_0 = [x_0, y_0, \theta_0]^T$.

As the robot moves and observes its environment, its state vector continually expands. For example, when the robot first observes a new landmark, its state vector expands to $x_t = [x, y, \theta, \mu_{1,x}, \mu_{1,y}]^T$.

The covariance matrix of the robot Σ represents the uncertainty in the state estimation. Initially, the covariance matrix can be represented as

$$\Sigma_0 = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta}, \end{bmatrix} \tag{2}$$

where the diagonal elements represent the variance of the state variables, while the off-diagonal elements represent the covariance between different state variables. At the beginning, if we know the pose of the robot at startup, we can set these values to 0. As the robot observes its environment, the covariance matrix dynamically expands. For example, after observing a new landmark, the covariance matrix becomes

$$\Sigma_t = \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0} & \infty \end{bmatrix}. \tag{3}$$

The uncertainty in the position of a newly observed landmark is initialized as infinity (denoted by ∞) here because initially nothing is known about it. As more observations are made, these values will be updated to reflect a better understanding of the environment.

2.2 Motion Prediction

The control input consists of the rotational speeds $(\omega_1, \omega_2, \omega_3, \omega_4)$ of four wheels. However, since the independent noise in these speeds is difficult to measure, we use v_x, v_y, and ω_z as proxy variables, which represent the velocity of the robot in the x-direction, velocity in the y-direction, and rotational angular velocity, all within the robot coordinate system. These variables have a linear relationship with $(\omega_1, \omega_2, \omega_3, \omega_4)$ and are also affected by Gaussian noise. Therefore, we represent the discrete-time control vector u_k as

$$u_k = \begin{bmatrix} \Delta u_x \\ \Delta u_y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} v_x \Delta t \\ v_y \Delta t \\ \omega_z \Delta t \end{bmatrix}, \tag{4}$$

where Δu_x and Δu_y represent the displacement increments along the X and Y axes over the time Δt, while $\Delta\theta$ is the angular increment of rotation about the Z-axis. Then the state update equation is as follows:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t|t-1} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t-1} + \begin{bmatrix} \cos\theta_{t-1} & -\sin\theta_{t-1} & 0 \\ \sin\theta_{t-1} & \cos\theta_{t-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta u_x \\ \Delta u_y \\ \Delta\theta \end{bmatrix}. \tag{5}$$

Simplifying the above equation, we get:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t|t-1} = \begin{bmatrix} x_{t-1} + \Delta u_x \cos\theta_{t-1} - \Delta u_y \sin\theta_{t-1} \\ y_{t-1} + \Delta u_x \sin\theta_{t-1} + \Delta u_y \cos\theta_{t-1} \\ \theta_{t-1} + \Delta\theta \end{bmatrix} \tag{6}$$

Considering that Δu_x and Δu_y are changes in control input and do not directly depend on the state x_{t-1}, the Jacobian matrix G_t can be expressed as

$$G_t = F_x \frac{\partial}{\partial x} \begin{bmatrix} x + (\Delta u_x \cos\theta - \Delta u_y \sin\theta) \\ y + (\Delta u_x \sin\theta + \Delta u_y \cos\theta) \\ \theta + \Delta\theta \end{bmatrix} F_x^T = F_x \begin{bmatrix} 1 & 0 & -\Delta u_x \sin\theta_{t-1} - \Delta u_y \cos\theta_{t-1} \\ 0 & 1 & \Delta u_x \cos\theta_{t-1} - \Delta u_y \sin\theta_{t-1} \\ 0 & 0 & 1 \end{bmatrix} F_x^T, \tag{7}$$

where F_x is a spatial mapping matrix, with the top-left 3×3 matrix corresponding to the pose of the robot being an identity matrix and the rest being zeros:

$$F_x = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix}_{3 \times (2N+3)}. \tag{8}$$

This Jacobian matrix G_t reflects the dependency of the state prediction on the current state, which will be used in the update of the state covariance matrix Σ_t:

$$\Sigma_{t|t-1} = G_t \Sigma_{t-1|t-1} G_t^T + F_x R_t F_x^T, \tag{9}$$

where R_t is the covariance matrix of the system noise, representing the uncertainty in the prediction model. It is worth mentioning that, although the motion process only changes the pose of the robot, the correlation between the robot pose and the position of each landmark will also be updated.

2.3 Correction with Observations

We define the observation vector z_i to represent the observation of the i-th landmark, which includes the distance and angle to the landmark:

$$z_t^i = \begin{bmatrix} r_t^i \\ \phi_t^i \end{bmatrix}, \tag{10}$$

where r_t^i is the distance from the robot to the landmark, and ϕ_t^i is the directional angle from the robot to the landmark. The observation error matrix Q_t^i represents the uncertainty of the measurements (typically assumed to be Gaussian distributed):

$$Q_t^i = \begin{bmatrix} \sigma^2_{r_t^i} & 0 \\ 0 & \sigma^2_{\phi_t^i} \end{bmatrix}, \tag{11}$$

where $\sigma^2_{r_t^i}$ is the variance of the distance measurement, and $\sigma^2_{\phi_t^i}$ is the variance of the angle measurement. This matrix can be adjusted based on the actual precision of the measurements.

If it is a new landmark, initialize the state of the landmark and correspondingly expand the state matrix μ_t and the covariance matrix Σ_t. The state vector μ_t adds the coordinates of the landmark:

$$\begin{aligned} \mu_{j,x} &= x + r_t^i \cos(\phi_t^i + \theta) \\ \mu_{j,y} &= y + r_t^i \sin(\phi_t^i + \theta), \end{aligned} \tag{12}$$

where j is the index of the landmark in the state vector. The corresponding covariance matrix is initialized to infinity as in Equation 3.

For all observed features z_t^i, let j be the index of the landmark in the state vector, we can calculate the predicted observation:

$$\hat{z}_t^i = \begin{bmatrix} \sqrt{q} \\ \arctan(\frac{\delta_y}{\delta_x}) - \theta \end{bmatrix}, \quad \delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \mu_{j,x} - x \\ \mu_{j,y} - y \end{bmatrix}, \quad q = \delta^T \delta. \tag{13}$$

Then the Jacobian matrix H_t^i can be constructed as follow:

$$H_t^i = \frac{1}{q} \begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix} F_{x,j} \tag{14}$$

where $F_{x,j}$ is a spatial mapping matrix, with only the robot pose and the landmark(j) pose corresponding part as identity matrices, the rest being zeros. With the above equations, we can calculate the Kalman gain:

$$K_t^i = \Sigma_{t|t-1} H_t^{i^T} \left(H_t^i \Sigma_{t|t-1} H_t^{i^T} + Q_t^i \right)^{-1}. \tag{15}$$

Finally, we can update the state vector μ_t and the covariance matrix Σ_t, which are composed of the robot state and the landmark states:

$$\begin{aligned} \mu_t &= \mu_t + K_t^i \left(z_t^i - \hat{z}_t^i \right) \\ \Sigma_{t|t} &= \left(I - K_t^i H_t^i \right) \Sigma_{t|t-1}. \end{aligned} \tag{16}$$

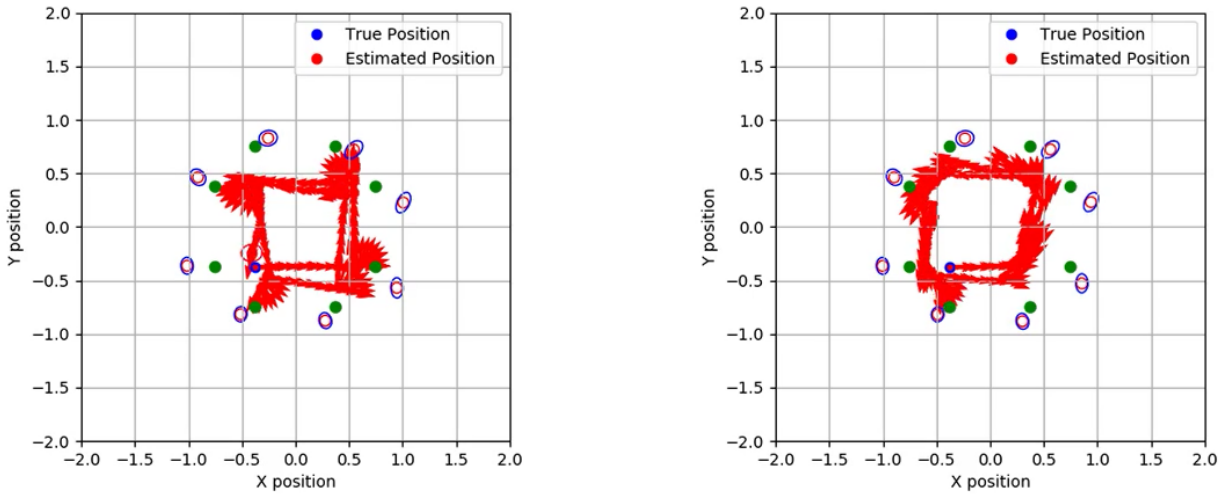
In this algorithm, the system noise covariance matrix R and the observation noise covariance matrix Q need to be determined. We approximately determined the values by measuring the errors in motion and observation multiple times. Then during the operation, we fine-tune them to achieve the best results.

3 Results

We ran the algorithm described above in our environment described in Section 1. The resulting motion trajectories and mapping results are shown in the Figure 2. The left Figure 2a shows the results of the robot moving along a square trajectory for two laps, and the right Figure 2b shows the result of the robot moving along an octagonal trajectory for two laps. In the figure, the red hollow circles represent the landmark positions estimated by the algorithm, while the corresponding blue ellipses represent the covariance. The green dots indicate the actual positions of the landmarks.

From the Figure 2, we can see that although there are some errors, our algorithm has successfully enabled the robot to achieve simultaneous localization and mapping within a reasonable error range. The mapping error obtained using the octagonal motion trajectory is relatively smaller.

Due to the uncertainty of the initial position, we convert the estimated landmark positions into the world coordinate system and obtain the error of the initial robot position (x_0, y_0, θ_0) by optimizing the average Euclidean distance between the converted estimated positions and the true landmark positions. We assume that the matrix containing all estimated landmark positions is X, and the matrix containing all true landmark



(a) Square (b) Octagon

Figure 2: SLAM results and robot trajectory, where the red arrows represent the trajectory of the robot, the blue dot indicates the true starting position of the robot, the green dots represent the ground-truth position of the activated landmarks. The red hollow circles represent the estimated position of landmarks (before alignment), and the corresponding blue ellipses represent the corresponding covariance.

positions is Y, both having dimensions $N \times 2$, where N is the number of landmarks. Our goal is the following optimization problem:

$$(x_0, y_0, \theta_0) = \arg\min_{x,y,\theta} \frac{1}{N} \sum_{i=1}^{N} \sqrt{\delta_i \delta_i^T}, \quad \text{where} \quad \delta_i = X_i \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T + \begin{bmatrix} x & y \end{bmatrix} - Y_i, \tag{17}$$

We solved this optimization problem using a numerical optimization algorithm, and the results are shown in the Table 1.

Trajectory	x_0	y_0	θ_0	**Average Error**
Square	-0.020	0.056	0.142	0.153
Octagon	-0.040	0.043	0.137	0.130

Table 1: Results of Alignment and Average Errors.

The results show that the actual initial position of the robot differs from the set initial position by only a small error of less than 5 centimeters, which is quite reasonable, as we placed the robot according to the set position. **The average errors between the estimated positions of the landmarks and the real map are** 0.153 **meters and** 0.130 **meters, respectively. This might be due to errors in placing the landmarks or from cumulative errors during movement. When the robot first observes a landmark, it calculates the estimated position of the landmark using its own estimated position, in which process the error is inherited and accumulates.**

The average estimation error resulting from the octagonal trajectory is smaller than that from the square trajectory, which is consistent with our observations from the Figure 2. This could be because the robot undergoes several minor rotations when moving along the octagonal trajectory, resulting in a wider range of observations and more frequent observations of the same landmark.

Driving multiple times. By observing our vedio, we found that when the robot returns to the starting point, the estimated error significantly decreases. This is because when the robot sees landmarks it has seen before, it can be more certain about its previous trajectory and estimation results. Repeated observations

of the same landmark in some ways act as a form of feedback. Similarly, when we drive along the trajectory multiple times, the estimated error becomes smaller at the beginning. When the estimated state values have converged to a relatively certain position, the estimates will stabilize. At this point, updates to the algorithm will not significantly alter the estimated values.

Trade-off of the trajectory. When considering the size of the trajectories, such as size of the square and the octagon, it's important to weigh the trade-offs. Setting a smaller size is beneficial for the robot to see multiple landmarks simultaneously, which makes the initial localization more accurate and thus the motion more precise. However, if the area is set too small, the relative error in the robot movement becomes larger, and the observation of the landmarks may become unstable. As a result, the robot might not be able to complete the exploration and mapping of the entire area.