Homework-3

CSE 276A Fall'24 — Introduction to Robotics

Pranav Malpure **PID.: A69030926**

Akshar Tumu **PID.: A69032632**

Introduction

The objective of this assignment is to implement the Simultaneous Localization and Mapping (SLAM) technique using a Linear Kalman filter to estimate the locations of landmarks (April tags) in the environment while simultaneously using the estimations to make the robot move in a specified path. We tested the model on 2 paths - a square and an octagon, and compared the performance of the model on both the paths.

Environment

We consider an environment of a square size of 8ft x 8ft. Each edge of this square has 2 landmarks (total 8) arranged in a regular fashion. The top-down view of the ground truth map can be found below:

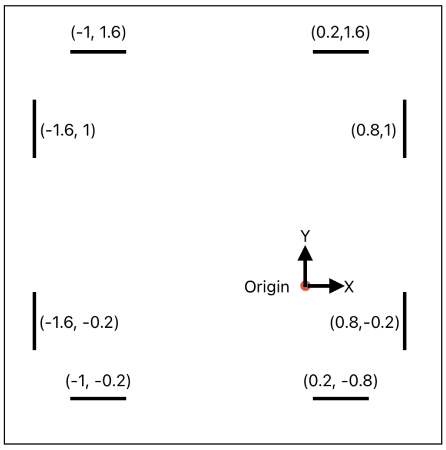

Figure 1: Sketch of the ground truth map

SLAM Algorithm

In our map, each of the 8 april tags has a different ID associated with it. However, we don't provide the SLAM model or the associated parameters with any details about the no. of april tags in the environment or their IDs. Hence, we assume that the total possile no. of april tag IDs in the class of april tags that we are using is 25 and initialize the parameter vectors and matrices accordingly.

We first start off by initializing the state and the variance vectors:

State vector $x_{0|0}$: The state vector is initialized to be a vector of size 53 with the following structure - The first 3 elements of the state vector correspond to the position of the robot in world frame (x, y, θ). After this, in the next elements we store the positions of the detected landmarks as follows:

$$\text{state}\,[2\cdot(\text{Tag\_ID}-1)+3] = x_{\text{tag ID}} \tag{1}$$

$$\text{state}\,[2\cdot(\text{Tag\_ID}-1)+1+3] = y_{\text{tag ID}} \tag{2}$$

Where $x_{\text{tag ID}}$ and $y_{\text{tag ID}}$ correspond to the x and y coordinates of the april tag with the given ID.
The entire state vector is initialized to 0 since we assume that the robot starts at origin (0, 0, 0) and the positions of the april tags are unknown

Covariance matrix $\Sigma_{0|0}$: We initialize the covariance matrix to be a matrix of size (53, 53). The first three diagonal elements correspond to the variance in the x, y, and θ values estimate for the robot position. Following this, the next diagonal elements depict the following:

$$\Sigma\left[2 \cdot (\text{Tag\_ID} - 1) + 3\right]\left[2 \cdot (\text{Tag\_ID} - 1) + 3\right] = variancex_{\text{tag ID}} \tag{3}$$

$$\Sigma\left[2 \cdot (\text{Tag\_ID} - 1) + 3 + 1\right]\left[2 \cdot (\text{Tag\_ID} - 1) + 3 + 1\right] = variancey_{\text{tag ID}} \tag{4}$$

Where $variancex_{\text{tag ID}}$ and $variancey_{\text{tag ID}}$ correspond to the variances in estimation of x and y coordinates of the april tag with the given ID.
Initially, the first 3 diagonal elements are initialized to 0 since the initial state of the robot which we assume as (0, 0, 0) is known without a doubt. The rest of the diagonal elements are all initialized to 100 since we are unaware of the positions of any landmark. The non-diagonal elements are always 0 since we use a Linear Kalman Filter and it doesn't consider the covariance among the landmarks and between the robot and landmarks.
New Landmark Assumption: When we detect a landmark for the first time, we store the measurement of the landmark given by detecting the april tag as its location in the state vector. We also reduce the corresponding x and y variations of this landmark to 1e-4 since the initial uncertainty concerned with this landmark is the uncertainty of the april tag detection. These values get updated in the next steps based on the predictions of the open loop control and future april tag detections.

Predict Step: This step estimates the state update ($x_{t|t-1}$) and the covariance update ($\Sigma_{t|t-1}$) based on the control signal provided to the robot and the noise associated with the robot movement in open loop control. The equations in this step are as follows:

$$x_{t|t-1} = F x_{t-1|t-1} + G u_t \tag{5}$$

$$\Sigma_{t|t-1} = F \Sigma_{t-1|t-1} F^T + Q \tag{6}$$

The **system matrix F** gives the transformation of $x_{t-1|t-1}$ when there is no control signal passed to the system (the state remains the same). We define F to be an identity matrix of size (53, 53) to make this happen.
The **control vector 'u'** is of the form $[v_x, v_y, w_z]$ which gives the velocities given to the robot in x, y directions and its angular velocity. This is obtained by multiplying the control signals given to each wheel with the kinematic matrix of the robot.
The **control matrix G** maps the movement of the robot caused by the control vector. It is a matrix of shape (53, 3) with the first three rows being an identity matrix multiplied by 'Δt', which we set to 0.1 seconds.
The system noise matrix Q considers the noise in the estimation of the state vector and the april tag positions in this step. In the predict state of the Kalman Filter, we predict the state of the system on giving an input u, and since there will be no change in the landmark position on giving an input u to the robot, the corresponding Q matrix elements to the april tags will be zero. The rest of the diagonal terms corresponding to the robot state vector have been taken as 0.01 as the first diagonal element, 0.02 as the second diagonal element, and rest diagonal elements as zero. These values were obtained through extensive trial and errors.
Update Step: This step updates the state vectors and covariances by the measurements obtained from the april tag detections by the camera and the predictions obtained in the predict step. The equations in this step are as follows:

$$S_t = H \Sigma_{t|t-1} H^T + R \tag{7}$$

$$K_t = \Sigma_{t|t-1} H^T S_t^{-1} \tag{8}$$

$$s_{t|t} = s_{t|t-1} + K_t \left(z_t - H s_{t|t-1}\right) \tag{9}$$

$$\Sigma_{t|t} = \left(I - K_t H\right) \Sigma_{t|t-1} \tag{10}$$

The 'z' vector consists of the locations of the april tags as detected by the camera. It is a vector (50, 1) that each pair of points corresponds to the (x, y) location of the april tag. These values are w.r.t robot in robot frame. The 'H' matrix helps convert the april tag positions in the state vector (in world frame w.r.t assumed origin) to robot frame w.r.t the robot. For this, we design the H matrix to be of size (50, 53). Each pair of rows correspond to one april tag.
We have defined a 'get measurement' function which checks for april tags 25 tims and records the frame IDs and the detections of each obtained april tag. We initialize H to be an array of all 0's. But then, for each detected april tag, we set the H matrix as follows:

$$H[(\text{tag\_ID} - 1) \cdot 2][0] = -\cos(\theta) \tag{11}$$

$$H[(\text{tag\_ID} - 1) \cdot 2][1] = -\sin(\theta) \tag{12}$$

$$H[(\text{tag\_ID} - 1) \cdot 2 + 1][0] = \sin(\theta) \tag{13}$$

$$H[(\text{tag\_ID} - 1) \cdot 2 + 1][1] = -\cos(\theta) \tag{14}$$

$$H[(\text{tag\_ID} - 1) \cdot 2][(\text{tag\_ID} - 1) \cdot 2 + 3] = \cos(\theta) \tag{15}$$

$$H[(\text{tag\_ID} - 1) \cdot 2][(\text{tag\_ID} - 1) \cdot 2 + 1 + 3] = \sin(\theta) \tag{16}$$

$$H[(\text{tag\_ID} - 1) \cdot 2 + 1][(\text{tag\_ID} - 1) \cdot 2 + 3] = -\sin(\theta) \tag{17}$$

$$H[(\text{tag\_ID} - 1) \cdot 2 + 1][(\text{tag\_ID} - 1) \cdot 2 + 1 + 3] = \cos(\theta) \tag{18}$$

If a landmark disappears, its estimate won't be updated as the corresponding H rows are set to 0. If an old landmark reappears, its H rows are set as the above equations, updating the landmark's position in this iteration.
K_t is the kalman gain factor which decides how much of the difference in estimated april tags and measured april tags should be considered for updating the state. R gives the noise in the april tag measurement. We set this matrix to a diagonal matrix with each diagonal element as 0.01. We have set this 0.01 to account for any small error which can occur due to the inaccurate reading of the april tag. We did set this as 0.1 initially, but the noise cumulated up a lot giving us worse results.

Angle Measurement

For angle measurement and rotation, we rely on the pitch measurements (angle between the z-axes of the robot and the April tag) from the detected april tags. We consider the april tag which is closest to the robot (measured from the stored positions of the april tag and the robot in the state vector) and visible both before and after rotating. We calculate the change in the pitch of this april tag to obtain how much the robot rotated.

Robot Movement

For each edge of the square / octagon, we first start by making the robot move straight to the next waypoint. Following this, the root works on correcting its lateral position and also the angle based on the errors occurred during moving straight. Once the robot reaches a waypoint (vertex), it rotates towards the next waypoint and also corrects its x and y values if required.

Trajectories

We chose the side lenght of the octagon and the square in a way to balance between the maximum number of april tags visibility while also moving the robot considerably to get enough points for the kalman filter to work. Apart from this, as we calculate the angle the robot has turned with respect to a fixed april tag, it was important for us that atleast one april tag is always in visibility when turning 90 degrees, and hence the side of the square smaller.
The octagon to be traversed has been less accurate than the square, and since the octagon has the robot going towards the edge of the boundary while traversing some sides, atleast one april tag should be visible, which further motivated us to keep the size and expanse of the octagon small.

Results

We calculated the average error between the estimated map and the ground truth map by summing the distances between the estimated and ground truth landmarks and dividing by the total number of landmarks detected during the path traversal. This provided a measure of the overall mapping accuracy.
Plots of estimated trajectories and landmarks:
Square: Figure 2 (Average Error = 0.366m)

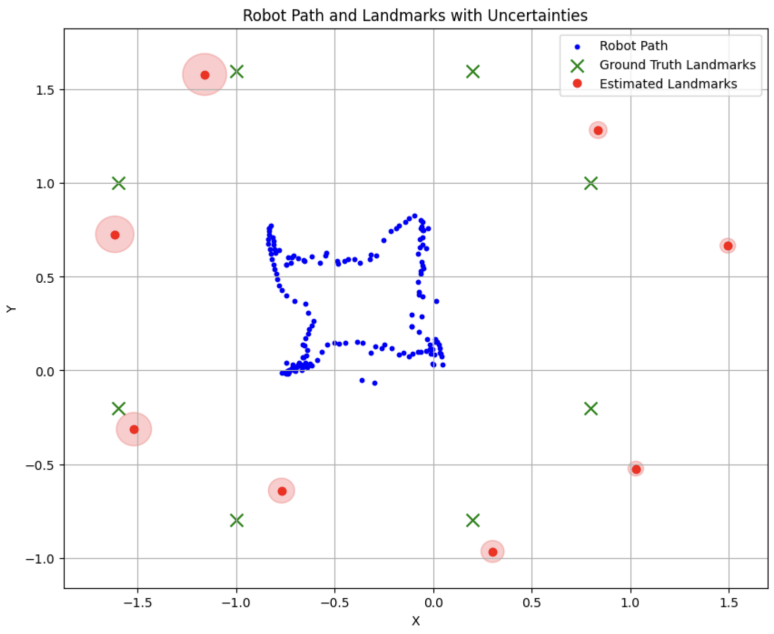

Figure 2: Square Path

Octagon: Figure 3 (Average Error = 0.542m)

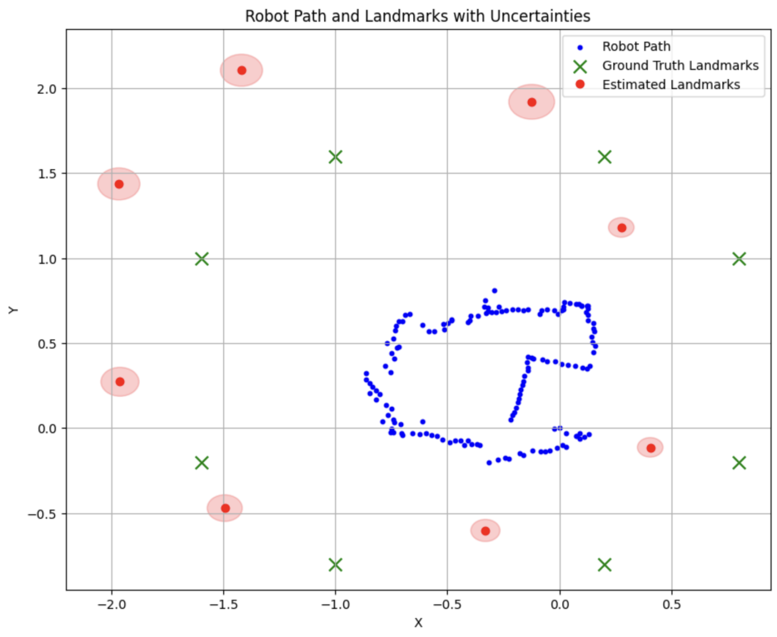

Figure 3: Octagon Path

The average error for Square trajectory is less than the average error for the Octagon trajectory. This is because the robot adhered to the square trajectory waypoints with more accuracy than the octagonal waypoints. in the square, the april tags in the left half in Figure 2 have been better localized (with the least error being 15 cm) compared to the april tags on the right side. However, for octagon, the errors are more evenly distributed.

Multiple Loops

We also ran the robot for 3 loops of square trajectory. We observed that the error kept reducing as we did more loops of the trajectory. Table 1 shows the results:

	1 loop	2 loops	3 loops
Square motion Average Error	0.366	0.324	0.296

Table 1: Average Error across loops

Video Links

YouTube Video Link for the Square path is ***here***: https://youtu.be/sMARKSFCCP0
YouTube Video link for Octagon path is ***here***: https://youtu.be/ioBoPGjurnE