Project Report

Jun-Jee Chao

1 Introduction

This project aims to implement SLAM to localize the robot's position and unknown landmarks' positions while navigating through the environment to reach the goal position. As shown in Figure 1, the robot starts at the right corridor, and the goal position is indicated as the red sphere in the left corridor. Several spheres with unknown positions and ids are added to the environment to serve as landmarks for the robot to recognize via camera. LiDAR sensor on the robot is used for obstacle avoidance and navigation. The goal of this project is to navigate the robot to reach the red sphere while keeping track of the robot's state and unknown landmarks' positions.

2 SLAM

2.1 Setup

The robot state is represented as $[x\ y\ \theta]$, and takes input $[v\ \omega]$. Since there's a maximum speed limit for the E-puck robot's wheel in Webots, the v and ω are limited to be within $0.06m/s$ and $0.58rad/s$. The control noise is introduced as zero-mean Gaussian noise with standard deviation of 0.01 times the maximum speed/omega. Which gives the control noise:

$$\Sigma_n = \begin{bmatrix} 3.6e-7 & 0 \\ 0 & 3.37446446e-05 \end{bmatrix}$$

In this project, the measurement is represented as the relative position from the landmarks to the robot. A zero mean Gaussian with 0.05 std is added to both x and y measurement, which gives the measurement noise:

$$\Sigma_m = \begin{bmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{bmatrix}$$

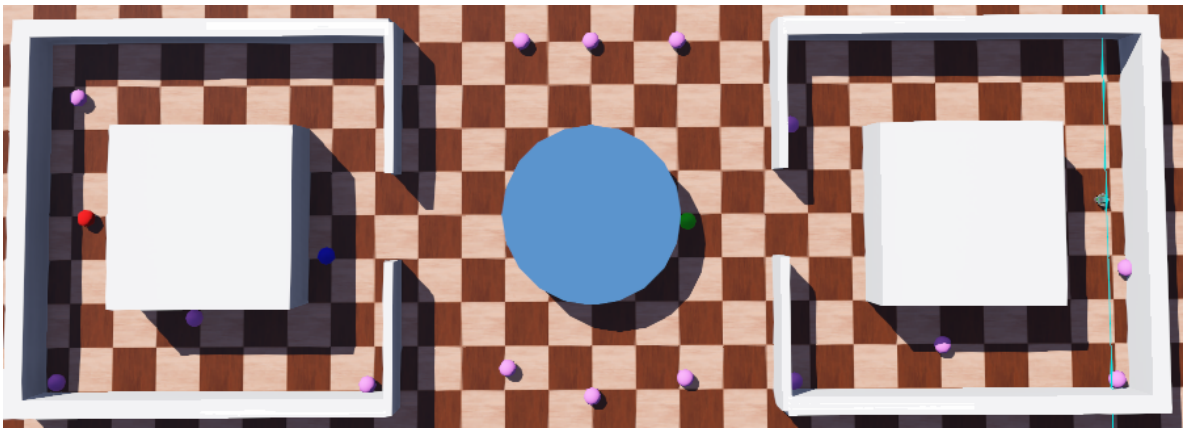

Figure 1: The simulation environment.

2.2 SLAM propagation

In each iteration, after the control signal $u = [v\ \omega]^T$ is sent to the robot with the control noise, we perform the SLAM propagation step. Specifically, given the current state:

$$X = [x_R\ x_{L1}\ ...\ x_{L_n}]^T$$

and state covariance:

$$\Sigma_X = \begin{bmatrix} \Sigma_{RR} & \Sigma_{RL_1} & \dots & \Sigma_{RL_n} \\ \Sigma_{L_1R} & \Sigma_{L_1L_1} & \dots & \Sigma_{L_1L_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{L_nR} & \Sigma_{L_nL_1} & \dots & \Sigma_{L_nL_n} \end{bmatrix}$$

The next state propagated with control input u is:

$$X_{t+1} = [f(x_{R_t}, u, 0)\ x_{L1}\ ...\ x_{L_n}]^T$$

where

$$f(x_{R_t}, u, 0) = \begin{bmatrix} x_t + v \cdot dt \cdot cos(\theta_t) \\ y_t + v \cdot dt \cdot sin(\theta_t) \\ \theta_t + \omega \cdot dt \end{bmatrix}$$

The next state covariance matrix is calculated as:

$$\begin{aligned} \Sigma_{X_{t+1}} &= A\Sigma_{X_t}A^T + B\Sigma_n B^T \\ &= \begin{bmatrix} A_R & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} \begin{bmatrix} \Sigma_{RR} & \Sigma_{RL_1} & \dots & \Sigma_{RL_n} \\ \Sigma_{L_1R} & \Sigma_{L_1L_1} & \dots & \Sigma_{L_1L_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{L_nR} & \Sigma_{L_nL_1} & \dots & \Sigma_{L_nL_n} \end{bmatrix} \begin{bmatrix} A_R & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix}^T + \\ &\quad \begin{bmatrix} N_R \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Sigma_n \begin{bmatrix} N_R \\ 0 \\ \vdots \\ 0 \end{bmatrix}^T \\ &= \begin{bmatrix} A_R\Sigma_{RR}A_R^T + N_R\Sigma_n N_R^T & A_R\Sigma_{RL_1} & \dots & A_R\Sigma_{RL_n} \\ \Sigma_{L_1R}A_R^T & \Sigma_{L_1L_1} & \dots & \Sigma_{L_1L_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{L_nR}A_R^T & \Sigma_{L_nL_1} & \dots & \Sigma_{L_nL_n} \end{bmatrix} \end{aligned}$$

where

$$A_R = \begin{bmatrix} 1 & 0 & -v \cdot dt \cdot sin(\theta_t) \\ 0 & 1 & v \cdot dt \cdot cos(\theta_t) \\ 0 & 0 & 1 \end{bmatrix}$$

$$N_R = \begin{bmatrix} dt \cdot cos(\theta_t) & 0 \\ dt \cdot sin(\theta_t) & 0 \\ 0 & dt \end{bmatrix}$$

2.3 SLAM update

The estimates from the previous step are updated with the observed measurements from the unknown landmarks. Since the landmarks' ids and global positions are unknown in this project, Mahalanobis distance is first applied to check if each measurement matches the landmark position in the current state. Then we update those landmarks' states that are receiving corresponding measurements. Finally, we add new landmarks to the state with those unmatched measurements.

2.3.1 Mahalanobis distance check

Mahalanobis distance is defined as:

$$d^2 = (z-h)^T S^{-1}(z-h)$$

where h maps the current robot state and each landmark state to the measurement:

$$h(x_R, x_{L_i}) = \begin{bmatrix} cos(\theta)\cdot(x_{L_i}-x_R) + sin(\theta)\cdot(y_{L_i}-y_R) \\ -sin(\theta)\cdot(x_{L_i}-x_R) + cos(\theta)\cdot((y_{L_i}-y_R) \end{bmatrix}$$

and

$$S = H\Sigma_{RR}H^T + \Sigma_m$$

$$H = \begin{bmatrix} -cos(\theta) & -sin(\theta) & -sin(\theta)(x_{L_i}-x_R) + cos(\theta)(y_{L_i}-y_R) \\ sin(\theta) & -cos(\theta) & -cos(\theta)(x_{L_i}-x_R) - sin(\theta)(y_{L_i}-y_R) \end{bmatrix}$$

For every received measurement z_i, the Mahalanobis distance is calculated with every existing landmark position x_{L_j} in the current state. If $d_{ij} < \epsilon_1$, measurement z_i is assigned to landmark L_j. If $d_i > \epsilon_2$ for every existing landmark, z_i is considered a measurement from a new landmark. If d lies between ϵ_1 and ϵ_2, measurement z_i is considered noisy and thus ignored in this step. In practice, ϵ_1 is set to 10 and ϵ_2 is set to 100.

2.3.2 State update with the existing landmarks

Now that the measurements belong to the existing landmarks are found, the state is updated with these measurements. Specifically, for each existing landmark that has a corresponding measurement, the state is updated as:

$$X_{t+1} = X_{t+1} + K(z-h)$$

$$\Sigma_{x_{t+1}} = \Sigma_{x_{t+1}} - \Sigma_{x_{t+1}} H^T S^{-1} H \Sigma_{x_{t+1}}$$

where

$$H = [H_R\ 0 \ldots\ 0\ H_{L_i}\ 0\ \ldots]$$

$$S = H\Sigma_{x_{t+1}}H^T + \Sigma_m$$

$$K = \Sigma_{x_{t+1}} H^T S^{-1}$$

$$H_R = \begin{bmatrix} -cos(\theta) & -sin(\theta) & -sin(\theta)(x_{L_i}-x_R) + cos(\theta)(y_{L_i}-y_R) \\ sin(\theta) & -cos(\theta) & -cos(\theta)(x_{L_i}-x_R) - sin(\theta)(y_{L_i}-y_R) \end{bmatrix}$$

$$H_{L_i} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}$$

2.3.3 Add new landmarks to the state

Finally, new landmarks are added to the state with those unmatched measurements.

$$X = [x_R\ x_{L_1}\ \ldots\ x_{L_n}\ H_L^T z + x_R]^T$$

$$\Sigma_X = \begin{bmatrix} \Sigma_{RR} & \Sigma_{RL_1} & \ldots & \Sigma_{RL_n} & -\Sigma_{RR}H_R^T H_L^{-T} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Sigma_{L_nR} & \Sigma_{L_nL_1} & \ldots & \Sigma_{L_nL_n} & -\Sigma_{L_nR}H_R^T H_L^{-T} \\ -H_L^{-1}H_R\Sigma_{RR} & -H_L^{-1}H_R\Sigma_{RL_1} & \ldots & -H_L^{-1}H_R\Sigma_{RL_n} & H_L^{-1}(H_R\Sigma_{RR}H_R^T + \Sigma_m)H_L^{-T} \end{bmatrix}$$

3 Navigation

While the SLAM algorithm keeps track of the robot's state with camera information, the LiDAR sensor is used to perform a simple wall follower in both corridors and avoid obstacles in the open space.

3.1 Wall follower

To implement a simple wall following algorithm, only several beams are extracted from the LiDAR data to detect if there's a wall in the front, right, and left of the robot. A simple left wall follower logic is shown as the following:

Left wall detected	Front wall detected	Action
Yes	No	Go straight
No	No	Turn left
Yes	Yes	Turn right
No	Yes	Turn right

In the right corridor, the robot follows the wall until it reaches the open space. Similarly, in the left corridor, the robot follows the wall until it sees the goal sphere. When the goal sphere becomes visible, the robot is simply navigated with the relative position to approach the goal sphere.

3.2 Open space navigation

Once the robot reaches the open space, several LiDAR beams are extracted to detect if there's an obstacle in the front. Similar to the wall following logic, the robot turns left until there's no obstacle in the front. When there's no obstacle in the front, the robot is navigated with the goal location and the SLAM robot state to drive itself towards the direction of the goal location until it enters the left corridor.

4 Result

The attached video shows that the algorithm is able to navigate the robot to the goal position. As shown in Figure 2, the SLAM algorithm is able to keep track of the robot trajectory and estimates the landmarks' positions.

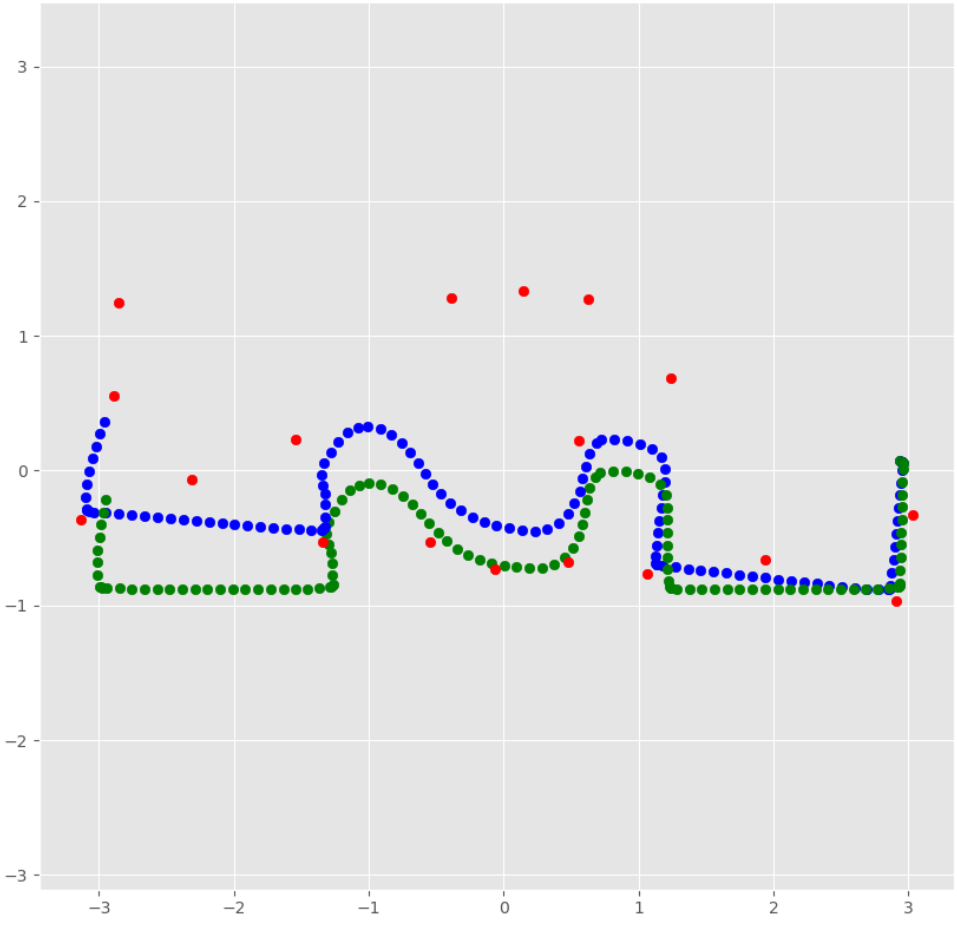

Figure 2: SLAM result. Ground truth robot trajectory is shown in green. Blue dots are the robot trajectory estimated by SLAM and the red dots are the final estimated positions of the landmarks.