Project 2 - Parte 2 Report

Bernardo de Almeida Soeiro
Rodrigo de Vasconcelos e Miguel

1 Introduction

The goal of this project is to implement and test a simultaneous localization and mapping (SLAM) system for a robot equipped with a LIDAR sensor. The sensor provides range measurements across 61 angles, spanning from $-30°$ to $30°$, and is aligned with the robot's forward axis.

The project is organized into three main tasks:

1. Develop an algorithm to process LIDAR data and detect corners relative to the robot's position.
2. Design a real-time system to estimate the robot's position and map the detected corners, with the ability to distinguish between previously known and newly detected corners.
3. Reconstruct the robot's entire trajectory and generate a comprehensive map of the environment by integrating all available measurements.

2 Corner Detection

2.1 Douglas-Peucker algorithm

In order to develop an algorithm capable of detecting corners, the Douglas-Peucker algorithm was used. This algorithm simplifies a curve by reducing the number of points while maintaining its original shape and key features [1].

First, we collect the LIDAR readings at a given position of the robot and connect the first and last points of the vector, forming a line. Then, we calculate the distance of the remaining points to their projection on the line. The point with the greatest distance that exceeds a defined threshold is considered a corner (Figure 1).

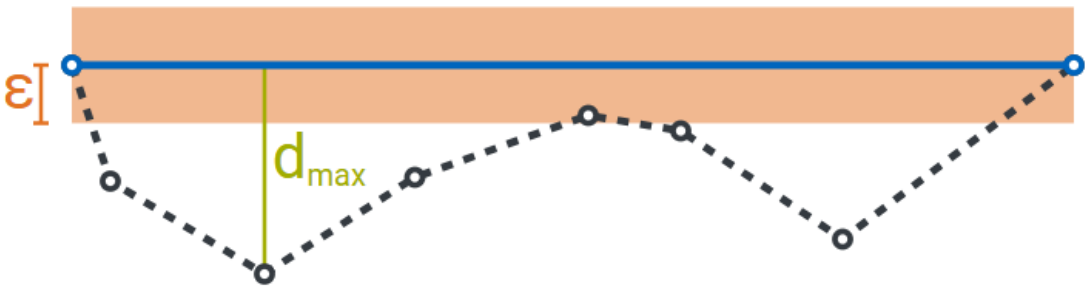

Figure 1: First corner detection[1]

If a corner is detected in the first step, we proceed to divide the line into two parts. The first part connects the first point to the corner, and the second part connects the corner to the last point. Then, we repeat the first step by projecting the points onto the closest line and comparing the distances to the threshold to detect more corners. The division of the lines is done continuously until no points are found with a distance to their projection greater than the threshold (Figure 2).

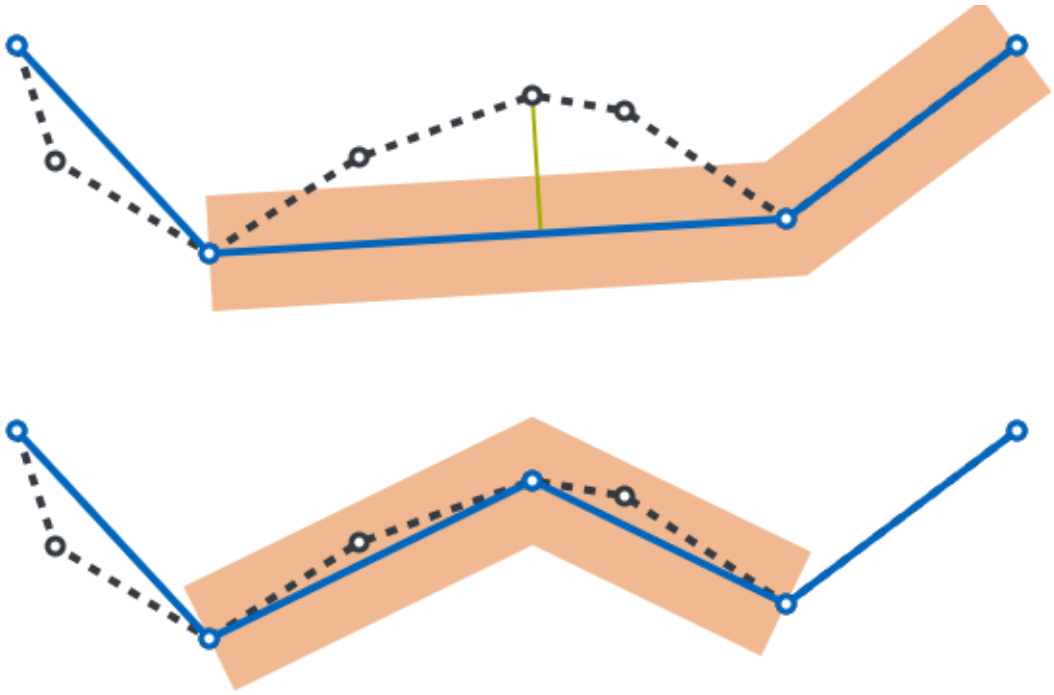

Figure 2: Dividing line [1]

2.2 Extra features

To enhance the detected corners with additional features, we developed functions to classify each corner as either convex or concave and to assign a directional vector representing its orientation. The classification is determined by analyzing the angle between vectors formed by the corner and its adjacent points, while the directional vector captures the corner's geometric orientation in space. These features are particularly useful for distinguishing nearby corners and improving data association in systems like the Kalman filter, enabling more robust mapping and localization in complex environments

2.3 Results

After applying the algorithm, the corners in the LIDAR data were successfully detected.Below are the results, showing the detected corners and their respective positions relative to the robot's position.

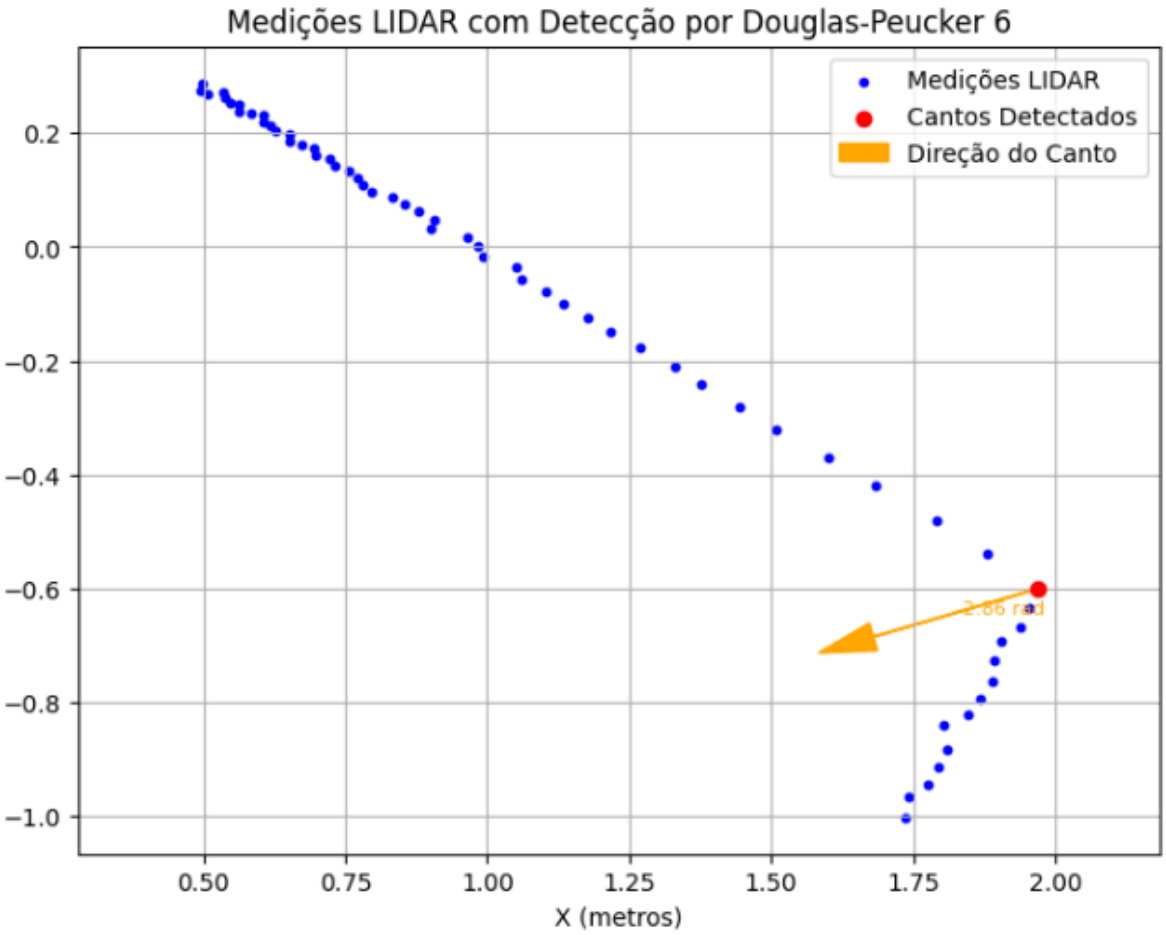

Figure 3: Detected Corners

3 Pose Estimation

3.1 Extended Kalman Filter

To accurately estimate the robot's position and update the map in real time, an Extended Kalman filter was implemented. This approach allows the system to fuse data from the robot's motion model (odometry) with the detected corners from the LIDAR sensor, providing robust state estimation despite the presence of noise and uncertainties in the measurements.

3.1.1 Prediction Step

To **optimize** the system, the SLAM algorithm employs an Extended Kalman Filter (EKF). In this model, the state vector encompasses both the robot's pose and an array of landmarks (corners). Consequently, the state vector has a dimensionality of $3+2N$, where 3 corresponds to the robot's pose (x, y, θ) and $2N$ accounts for the x- and y-coordinates of the N detected landmarks. This combined state allows the system to simultaneously estimate the robot's position and build a map of its surroundings [2].

The covariance matrix, $\mathbf{P}$, captures the uncertainties associated with both the robot's pose and each landmark, as well as the correlations between them. Consequently, $\mathbf{P}$ is a square matrix with dimensions $(3 + 2N) \times (3 + 2N)$ [2].

$$X = \begin{bmatrix} x \\ y \\ \theta \\ x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_n \\ y_n \end{bmatrix}, \quad P = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xx_1} & \sigma_{xy_1} & \cdots & \sigma_{xy_n} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{yx_1} & \sigma_{yy_1} & \cdots & \sigma_{yy_n} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta x_1} & \sigma_{\theta y_1} & \cdots & \sigma_{\theta y_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{y_n x} & \sigma_{y_n y} & \sigma_{y_n \theta} & \sigma_{y_n x_1} & \sigma_{y_n y_1} & \cdots & \sigma_{y_n y_n} \end{bmatrix}.$$

Figure 4: State vector X and covariance matrix P [2]

The prediction step propagates the state and covariance estimates forward in time based on the motion model. This phase is characterized by the following steps

1. **Predict** the new state using the motion model:

$$X_t^- = f(X_{t-1}, u_t)$$

2. **Linearize** the motion model by computing the Jacobian matrix F_t:

$$F_t = \frac{\partial f(X, u)}{\partial X}\Big|_{X_{t-1}, u_t}$$

3. **Update** the covariance matrix, where Q_t is the process noise covariance:

$$P_t^- = F_t P_{t-1} F_t^\top + Q_t$$

By integrating the motion model with the state and covariance updates, the prediction step initializes the state estimate and prepares the system for incorporating sensor measurements in the update phase. This process ensures a single, coherent estimate of the robot's pose and the environment's landmarks, along with their associated uncertainties, over time [2].

3.1.2 Update Step

In the measurement step, the detected corners serve as landmarks, characterized by features such as convexity, concavity, and directional vectors. During the update step, these features are used to match detected corners to known landmarks in the state. Once a match is identified, the Kalman Filter adjusts both the robot's pose and the positions of the landmarks in the map. By minimizing the error between predicted and observed measurements, the filter refines the robot's state and improves the accuracy of the map over time.

The update phase reduces the uncertainty by incorporating the difference between the predicted and observed measurements. This phase is characterized by the following steps:

1. Compute the **expected sensor measurement** using the observation model:

$$\hat{z}_t = h(X_t^-)$$

2. **Linearize** the observation model by computing the Jacobian matrix H_t:

$$H_t = \frac{\partial h(X)}{\partial X}\Big|_{X_t^-}$$

3. Compute the **innovation**:

$$y_t = z_t - \hat{z}_t$$

4. Compute the **Kalman gain:**

$$K_t = P_t^- H_t^\top \left(H_t P_t^- H_t^\top + R_t\right)^{-1}$$

5. **Update the state estimate:**

$$X_t = X_t^- + K_t y_t$$

6. **Update the covariance matrix:**

$$P_t = (I - K_t H_t) P_t^-$$

3.2 Estimated Trajectory and Corners

After applying the Kalman Filter to the SLAM system, the robot's estimated trajectory and the detected corners are computed.The trajectory, which is reconstructed by fusing odometry and LIDAR measurements, represents the robot's path through the environment. Meanwhile, the mapped corners serve as a representation of the surrounding features.

The following figure illustrates the estimated trajectory of the robot along with the identified corners, showing the alignment between the predicted states and the sensor observations over time.

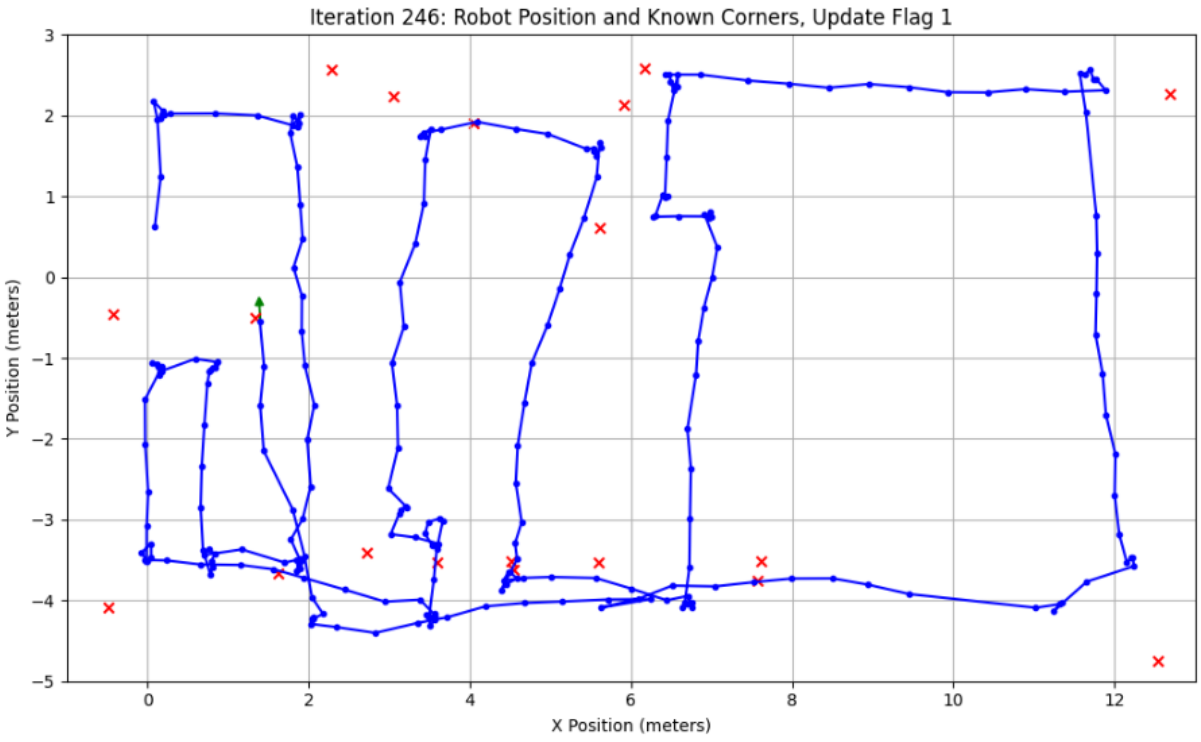

Figure 5: Estimated Trajectory

The evolution of the state throughout the process can be visualized in a video. This video illustrates how the trajectory and the corners location are progressively refined as the SLAM system processes the sensor data: **Click here to watch the video**.

4 Mapping

The results presented in this section illustrate how the mapping process captures the environment's features based on the estimated trajectory. The reconstructed map demonstrates the alignment between the robot's path and the spatial distribution of landmarks, offering a clear visualization of the explored area.

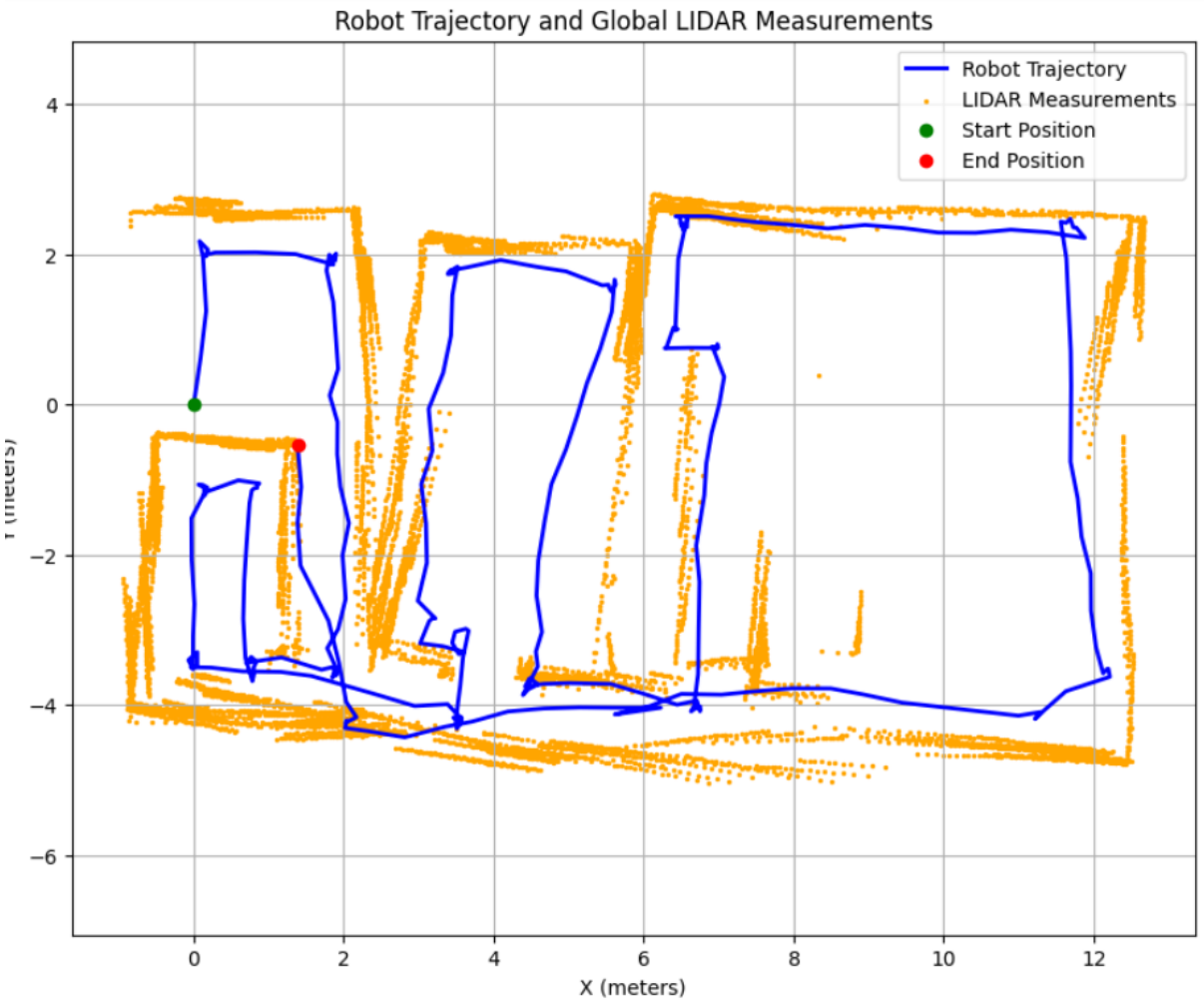

Figure 6: Final Mapping

5 Conclusion

This project successfully developed a SLAM system that combines corner detection with the Douglas-Peucker algorithm and trajectory estimation using a Kalman Filter. The system accurately reconstructed the robot's trajectory and mapped the environment by fusing LIDAR data and odometry.

The results demonstrate the system's ability to detect and map landmarks while maintaining a consistent and accurate trajectory. Future work could focus on optimizing performance and extending the approach to more complex environments.

6 References

References

[1] C. Playground, *Douglas-peucker algorithm playground*, Accessed: 2024-12-22, n.d. [Online]. Available: https://cartography-playground.gitlab.io/playgrounds/douglas-peucker-algorithm/.

[2] A. Sakai, *Ekf slam python implementation*, Accessed: 2024-12-22, n.d. [Online]. Available: https://atsushisakai.github.io/PythonRobotics/modules/slam/ekf_slam/ekf_slam.html.