Department of Electronic & Telecommunication Engineering,
University of Moratuwa,
Sri Lanka.

EN4594
Autonomous Systems

Mini Project

Dissanayaka D.M.S.P - 210141U

Date: 2025.12.24

Contents

1 Introduction **2**

2 Methodology **3**
2.1 System Overview 3
2.2 Mathematical Model 3
2.2.1 State and Motion Model 3
2.3 Implementation 4
2.3.1 Overview 4
2.3.2 ROS2 Node Implementation 4
2.3.3 Visualization and Path Comparison 6
2.3.4 Performance Evaluation 8

3 Results **10**
3.1 Visual Comparison of Robot Trajectories 10
3.2 Quantitative Performance Evaluation 12

4 Discussion **14**

5 Conclusion **15**

Chapter 1

Introduction

This project focuses on a mobile robot developed as part of the final year project. The robot is equipped with a differential drive mechanism and is capable of autonomous navigation within a structured indoor environment. To simulate the robot's motion and evaluate localization algorithms, ROS2 Jazzy was used along with Gazebo for physics-based simulation and RViz for visualization of robot states, paths, and sensor data.

The primary goal of this work is to implement an **Extended Kalman Filter (EKF)** to estimate the robot's trajectory from noisy odometry data. By comparing the EKF-estimated path against the ground truth path obtained from simulation, the effectiveness of the filter in reducing localization errors can be quantitatively evaluated. Key objectives of this project include:

1. Generating a simulated noisy odometry path to mimic realistic sensor noise conditions.
2. Applying an Extended Kalman Filter to estimate the robot's trajectory from noisy measurements.
3. Visualizing and comparing the ground truth, noisy, and EKF-estimated paths in RViz.
4. Evaluating the performance of the EKF using quantitative metrics such as Root Mean Square Error (RMSE).

The report documents the methodology, implementation, and results of the EKF, providing both visual and quantitative analysis. Fig. 1.1 shows the robot in gazebo simulation environment.

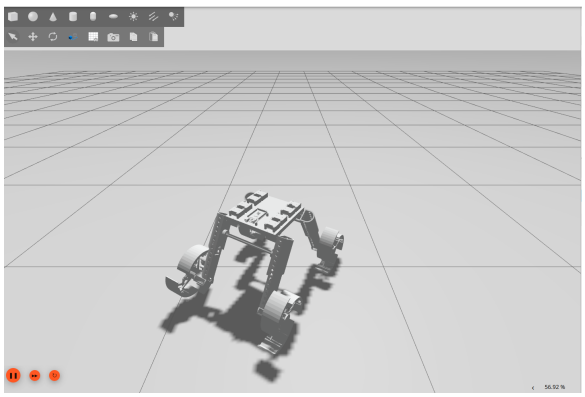

Figure 1.1: Gazebo Simulation Environment with the Robot.

Chapter 2

Methodology

2.1 System Overview

The system uses a simulated robot in Gazebo, controlled via ROS2. Key ROS2 topics used:

- `/odom_path` : Ground truth path
- `/noisy_odom` : Odometry with added Gaussian noise
- `/ekf_path` : EKF estimated path

2.2 Mathematical Model

2.2.1 State and Motion Model

The robot's state vector:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Prediction step (discrete-time kinematics):

$$x_{k+1} = x_k + v\cos(\theta_k)\Delta t$$

$$y_{k+1} = y_k + v\sin(\theta_k)\Delta t$$

$$\theta_{k+1} = \theta_k + \omega\Delta t$$

Covariance update:

$$\mathbf{P}_{k+1} = F\mathbf{P}_k F^T + Q$$

Measurement update:

$$\mathbf{K} = \mathbf{P}_{k+1}H^T(H\mathbf{P}_{k+1}H^T + R)^{-1}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1} + K(\mathbf{z} - H\mathbf{x}_{k+1})$$

$$\mathbf{P}_{k+1} = (I - KH)\mathbf{P}_{k+1}$$

2.3 Implementation

2.3.1 Overview

The implementation was carried out on a final year project robot using ROS 2 (Jazzy) along with Gazebo for simulation and RViz for visualization. The primary objective was to simulate the robot's motion, generate ground truth and noisy odometry data, implement an Extended Kalman Filter (EKF) for pose estimation, and compare the EKF output with ground truth.

The system consists of the following ROS2 nodes:

- **Path Controller:** Publishes velocity commands to drive the robot along a predefined path.
- **Odometry Path Node:** Converts raw odometry data from the robot into a path message for visualization.
- **Noisy Odometry Node:** Introduces Gaussian noise into the ground truth path to simulate sensor inaccuracies.
- **EKF Node:** Implements an Extended Kalman Filter that estimates the robot's pose based on noisy odometry.
- **EKF Path Node:** Converts the EKF pose estimates into a path message for visualization.

The robot's simulation is performed in Gazebo, with the robot's wheels controlled by a differential drive controller. RViz is used to visualize the ground truth path, noisy path, and EKF-estimated path simultaneously.

2.3.2 ROS2 Node Implementation

Odometry Path Node

The `odom_path_node` subscribes to the robot's odometry topic and publishes a `nav_msgs/Path` message for visualization in RViz. This node provides the ground truth trajectory of the robot.

Listing 2.1: Odometry Path Node

```
import rclpy
from rclpy.node import Node
from nav_msgs.msg import Odometry, Path
from geometry_msgs.msg import PoseStamped

class OdomPathNode(Node):
    def __init__(self):
        super().__init__('odom_path_node')
        self.sub = self.create_subscription(Odometry, '/
            diff_drive_base_controller/odom', self.callback, 10)
        self.pub = self.create_publisher(Path, '/odom_path', 10)
        self.path = Path()
        self.path.header.frame_id = "odom"

    def callback(self, msg):
```

```
        pose = PoseStamped()
        pose.header = msg.header
        pose.pose = msg.pose.pose
        self.path.header.stamp = self.get_clock().now().to_msg()
        self.path.poses.append(pose)
        self.pub.publish(self.path)
```

Noisy Odometry Node

The `noisy_odom_node` subscribes to the ground truth path and adds Gaussian noise to both position and yaw. This simulates real-world sensor noise, allowing us to test the EKF's performance.

Listing 2.2: Noisy Odometry Node

```
import rclpy
from rclpy.node import Node
import numpy as np
from nav_msgs.msg import Path
from geometry_msgs.msg import PoseStamped

class NoisyOdom(Node):
    def __init__(self):
        super().__init__('noisy_odom_node')
        self.sub = self.create_subscription(Path, '/odom_path', self.
            odom_callback, 10)
        self.pub = self.create_publisher(Path, '/noisy_path', 10)
        self.noise_std_pos = 0.05  # meters
        self.noise_std_yaw = 0.02  # radians

    def odom_callback(self, msg: Path):
        noisy_msg = Path()
        noisy_msg.header = msg.header
        noisy_msg.poses = []

        for pose_stamped in msg.poses:
            noisy_pose = PoseStamped()
            noisy_pose.header = pose_stamped.header
            noisy_pose.pose.position.x = pose_stamped.pose.position.x +
                np.random.normal(0, self.noise_std_pos)
            noisy_pose.pose.position.y = pose_stamped.pose.position.y +
                np.random.normal(0, self.noise_std_pos)
            # Convert quaternion to yaw, add noise, convert back
            q = pose_stamped.pose.orientation
            yaw = np.arctan2(2*(q.w*q.z), 1 - 2*q.z*q.z)
            yaw_noisy = yaw + np.random.normal(0, self.noise_std_yaw)
            noisy_pose.pose.orientation.z = np.sin(yaw_noisy/2)
            noisy_pose.pose.orientation.w = np.cos(yaw_noisy/2)
            noisy_msg.poses.append(noisy_pose)

        self.pub.publish(noisy_msg)
```

Extended Kalman Filter Node

The EKF node subscribes to noisy odometry (`/noisy_odom`) and velocity commands (`/cmd_vel`) and estimates the robot's pose using the standard EKF equations.

- **State vector:** $x = [x, y, \theta]^T$
- **Process model:**

$$x_{k+1} = \begin{bmatrix} x_k + v\cos\theta_k \Delta t \\ y_k + v\sin\theta_k \Delta t \\ \theta_k + \omega \Delta t \end{bmatrix}$$

- **Covariance update:** $P_{k+1} = F P_k F^T + Q$
- **Measurement update:**

$$K = PH^T(HPH^T + R)^{-1}, \quad x \leftarrow x + K(z - Hx), \quad P \leftarrow (I - KH)P$$

Listing 2.3: EKF Node Prediction and Update

```
def predict(self):
    v = self.cmd_vel[0, 0]
    w = self.cmd_vel[1, 0]
    theta = self.x[2, 0]
    self.x[0, 0] += v * np.cos(theta) * dt
    self.x[1, 0] += v * np.sin(theta) * dt
    self.x[2, 0] += w * dt
    F = np.array([[1,0,-v*np.sin(theta)*dt],
                  [0,1, v*np.cos(theta)*dt],
                  [0,0,1]])
    self.P = F @ self.P @ F.T + self.Q

def odom_callback(self, msg):
    z = np.array([[msg.pose.pose.position.x],
                  [msg.pose.pose.position.y],
                  [euler_from_quaternion([...])[2]]])
    H = np.eye(3)
    y = z - H @ self.x
    S = H @ self.P @ H.T + self.R
    K = self.P @ H.T @ np.linalg.inv(S)
    self.x += K @ y
    self.P = (np.eye(3) - K @ H) @ self.P
```

2.3.3 Visualization and Path Comparison

To compare the robot trajectories produced by different state estimates, multiple ROS2 nodes were implemented to generate and visualize paths in RViz. These paths enable qualitative evaluation of the Extended Kalman Filter by comparing ground truth, noisy odometry, and EKF-estimated trajectories.

The following paths were visualized:

- **Ground Truth Path** generated from raw odometry
- **Noisy Odometry Path** generated from corrupted odometry data
- **EKF Estimated Path** generated from EKF pose output

Path Generation from EKF Output

The EKF path is generated by subscribing to the EKF pose estimate topic and accumulating poses into a `nav_msgs/Path` message. This allows continuous visualization of the estimated robot trajectory.

Listing 2.4: EKF Path Generation Node (ekf_path_node.py)

```
class EKFPathNode(Node):
    def __init__(self):
        super().__init__('ekf_path_node')
        self.sub = self.create_subscription(
            Odometry,
            '/ekf_pose',
            self.callback,
            10
        )
        self.pub = self.create_publisher(Path, '/ekf_path', 10)
        self.path = Path()
        self.path.header.frame_id = "odom"

    def callback(self, msg):
        pose = PoseStamped()
        pose.header = msg.header
        pose.pose = msg.pose.pose
        self.path.poses.append(pose)
        self.pub.publish(self.path)
```

Ground Truth Path Generation

The ground truth path is created using the odometry output from the differential drive controller. This path is considered the reference trajectory for performance evaluation.

Listing 2.5: Ground Truth Path Node (odom_path_node.py)

```
self.sub = self.create_subscription(
    Odometry,
    '/diff_drive_base_controller/odom',
    self.callback,
    10
)
```

Noisy Odometry Path Generation

To simulate realistic sensor noise, Gaussian noise was added to the ground truth odometry. The resulting noisy odometry was then converted into a path for visualization.

Listing 2.6: Noisy Odometry Path Node (noisy_odom_path_node.py)

```
self.sub = self.create_subscription(
    Odometry,
    '/noisy_odom',
    self.callback,
    10
)

pose.pose = msg.pose.pose
```

```
self.path.poses.append(pose)
self.pub.publish(self.path)
```

Path Following Controller

The robot motion was controlled using a path controller node that generates velocity commands based on a predefined trajectory. These commands drive the robot in simulation and produce consistent motion for evaluation.

Listing 2.7: Path Controller Velocity Command (path_controller.py)

```
cmd_vel.linear.x = linear_velocity
cmd_vel.angular.z = angular_velocity
self.cmd_pub.publish(cmd_vel)
```

All paths were visualized simultaneously in RViz, allowing clear qualitative comparison between ground truth, noisy, and EKF-estimated trajectories.

2.3.4 Performance Evaluation

While visual comparison provides qualitative insight, quantitative evaluation is essential to objectively assess filter performance. Therefore, the Root Mean Square Error (RMSE) was used to measure positional accuracy.

RMSE Metric

The RMSE between an estimated path and the ground truth path is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[(x_i^{est} - x_i^{gt})^2 + (y_i^{est} - y_i^{gt})^2\right]}$$

where N is the number of pose samples.

RMSE Computation Node

A dedicated ROS2 node was implemented to compute RMSE values in real time by comparing:

- Noisy odometry path vs ground truth path
- EKF estimated path vs ground truth path

The node continuously accumulates pose data and computes RMSE as the simulation progresses.

Listing 2.8: RMSE Calculation Logic (path_rmse_node.py)

```
def compute_rmse(path_est, path_gt):
    n = min(len(path_est), len(path_gt))
    error_sum = 0.0

    for i in range(n):
        dx = path_est[i].pose.position.x - path_gt[i].pose.position.x
        dy = path_est[i].pose.position.y - path_gt[i].pose.position.y
```

```
        error_sum += dx**2 + dy**2

    rmse = np.sqrt(error_sum / n)
    return rmse
```

The computed RMSE values were logged during runtime, enabling continuous comparison between noisy odometry and EKF-based estimation.

Chapter 3

Results

3.1 Visual Comparison of Robot Trajectories

Figure 3.1 illustrates the ground truth trajectory of the robot obtained from the simulation odometry. This path represents the ideal robot motion without any artificial noise and is used as the reference for performance evaluation.

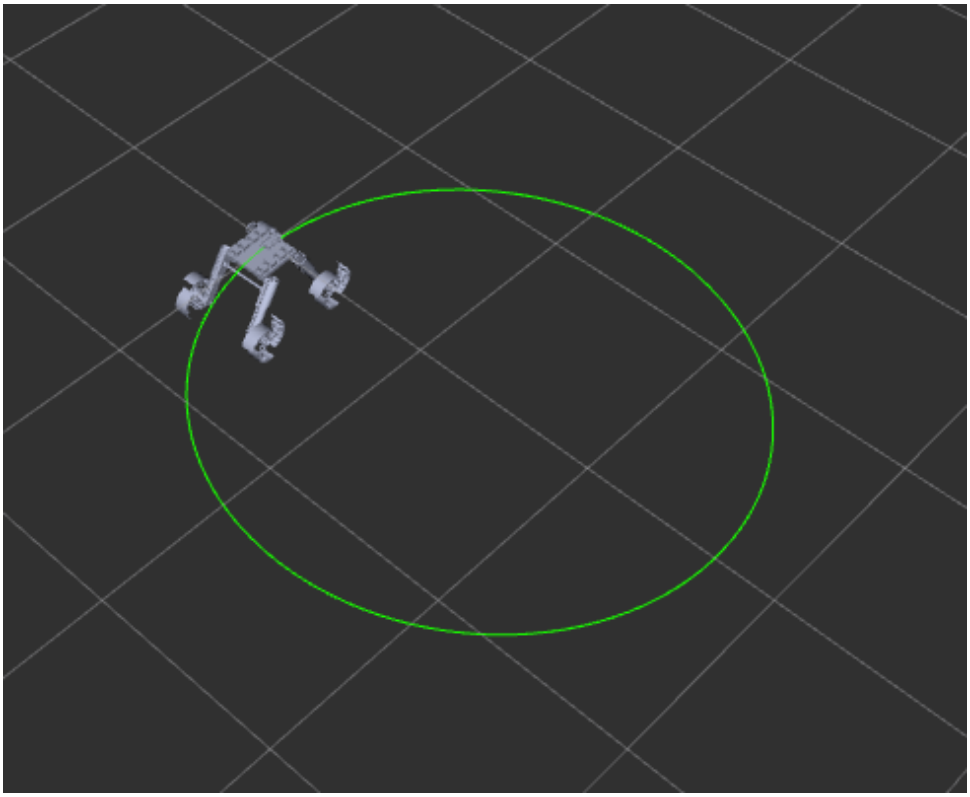

Figure 3.1: Ground truth robot trajectory obtained from simulation odometry

Figure 3.2 shows the trajectory generated using noisy odometry data, where Gaussian noise was intentionally added to the ground truth measurements. As observed, the noisy path deviates from the reference trajectory, exhibiting increased positional uncertainty and small oscillations along the path.

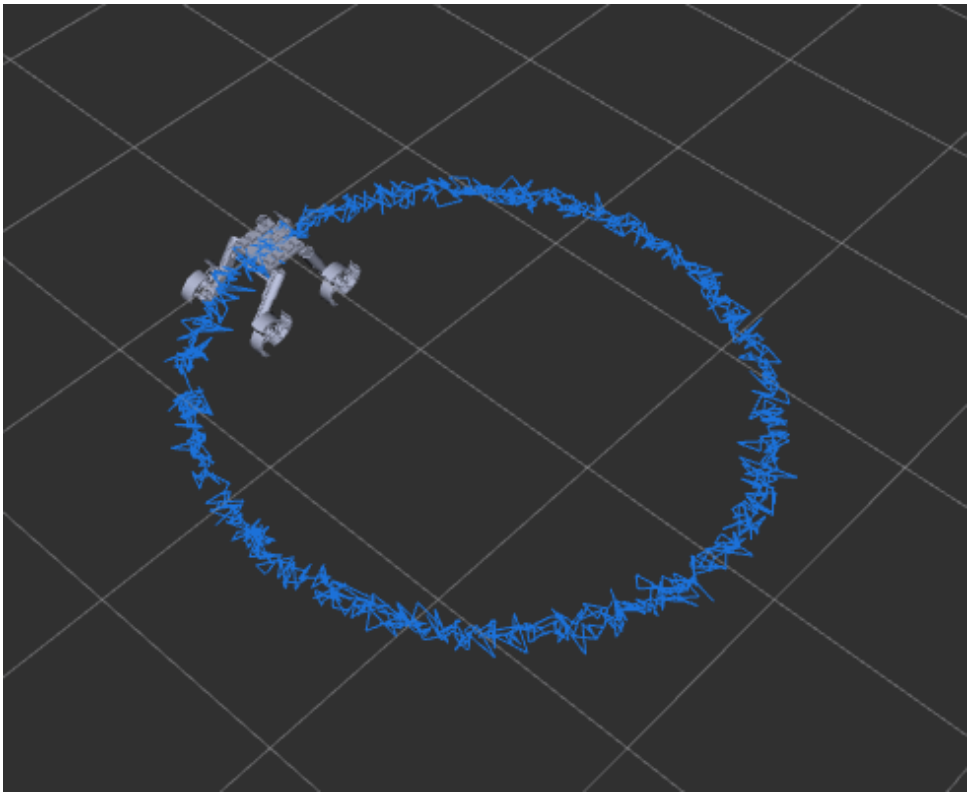

Figure 3.2: Robot trajectory generated from noisy odometry data

Figure 3.3 presents the trajectory estimated using the Extended Kalman Filter (EKF). Compared to the noisy path, the EKF-estimated path follows the ground truth trajectory more closely, demonstrating the filter's ability to reduce noise and provide a smoother and more accurate state estimate.

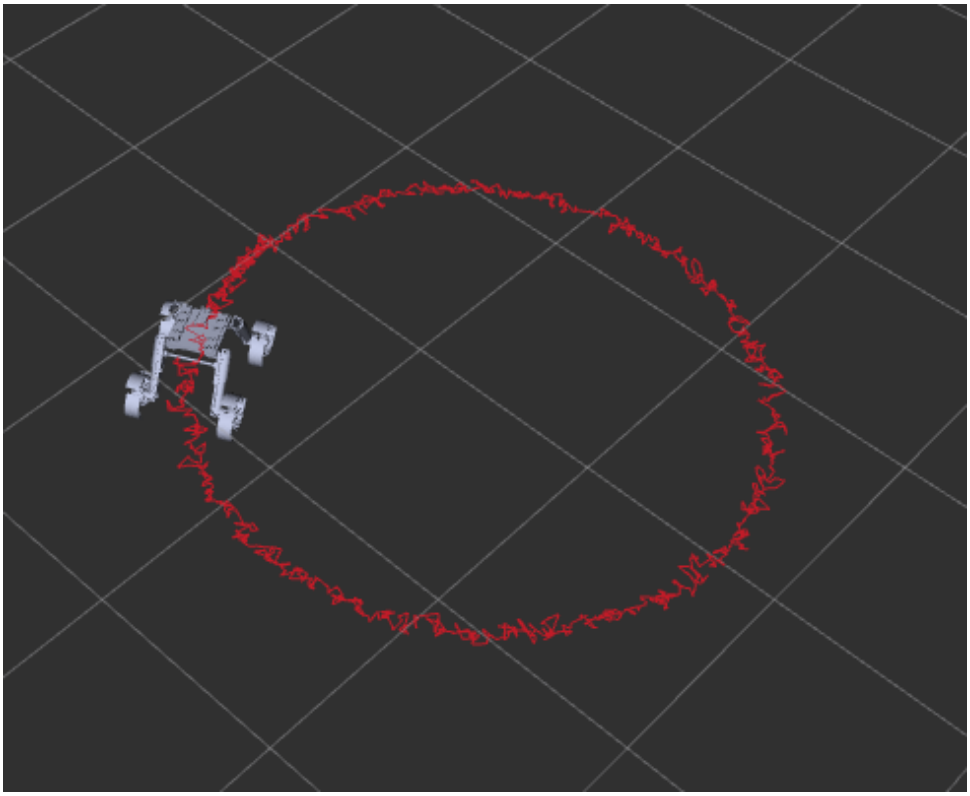

Figure 3.3: Robot trajectory estimated using the Extended Kalman Filter

Overall, the visual comparison highlights the effectiveness of the EKF in improving localization accuracy by mitigating the effects of sensor noise, which motivates the quantitative evaluation presented in the following section.

3.2 Quantitative Performance Evaluation

To quantitatively evaluate the localization performance, the Root Mean Square Error (RMSE) was computed for both the noisy odometry and the EKF-estimated trajectory with respect to the ground truth path.

Table 3.1 summarizes the RMSE values obtained at different time instances (number of poses). It can be observed that the EKF consistently achieves lower RMSE values compared to the noisy odometry measurements, indicating improved estimation accuracy.

Table 3.1: RMSE of noisy and EKF paths with respect to ground truth.

Number of Poses (N)	RMSE Noisy vs GT (m)	RMSE EKF vs GT (m)
161	0.0689	0.0399
254	0.0705	0.0416
343	0.0692	0.0409
438	0.0695	0.0409
533	0.0694	0.0410
624	0.0694	0.0416

Figure 3.4 illustrates the variation of RMSE with respect to the number of poses. The noisy odometry error remains relatively higher and more fluctuating, whereas the

EKF error converges to a lower and more stable range. This behavior demonstrates the effectiveness of the EKF in reducing accumulated localization error over time.

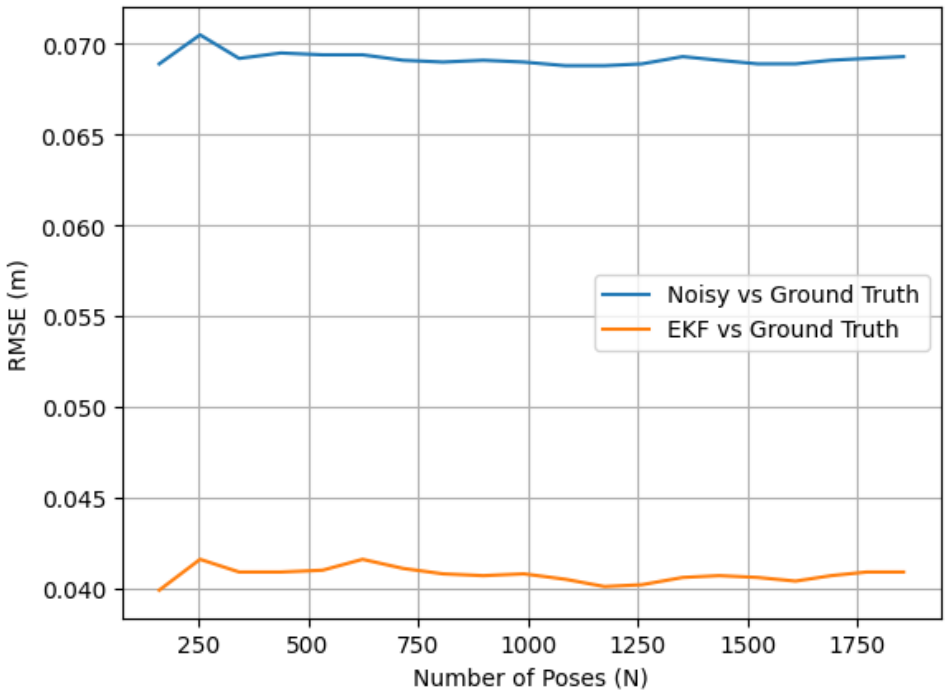

Figure 3.4: RMSE vs Number of Poses: Noisy vs GT (red), EKF vs GT (blue).

Overall, both the numerical results and the plotted trends confirm that the EKF provides a more accurate and reliable pose estimate compared to raw noisy odometry.

Chapter 4

Discussion

The results obtained from both qualitative and quantitative evaluations demonstrate that the Extended Kalman Filter (EKF) significantly improves the robot's localization accuracy when compared to raw noisy odometry measurements.

From the visual path comparisons presented in Section 3.1, it is evident that the noisy odometry path deviates noticeably from the ground truth trajectory due to accumulated sensor noise and integration errors. In contrast, the EKF-estimated path closely follows the ground truth, indicating that the filter effectively mitigates noise and drift. This improvement is particularly important for mobile robot navigation tasks, where small localization errors can accumulate over time and lead to substantial deviations.

The quantitative RMSE analysis further validates this observation. As shown in Table 3.1 and Figure 3.4, the RMSE of the EKF-estimated path consistently remains lower than that of the noisy odometry across different numbers of poses. While the RMSE of the noisy odometry stabilizes around approximately 0.07 m, the EKF maintains an error close to 0.04 m. This reduction highlights the effectiveness of the EKF in fusing motion inputs with noisy measurements to produce a more accurate state estimate.

The stability of the EKF error over time also indicates that the filter parameters, particularly the process noise covariance and measurement noise covariance matrices, were reasonably tuned for the simulated environment. However, it should be noted that these parameters were selected empirically and optimized for the given simulation scenario. Different robot dynamics, sensor characteristics, or operating environments may require further tuning to achieve optimal performance.

One limitation of the current implementation is that the EKF relies on a simplified motion model and odometry-based measurements. In real-world applications, wheel slippage, uneven terrain, and sensor delays can introduce additional uncertainties that are not fully captured in simulation. Moreover, the ground truth data used in this study is derived from the simulator, which provides idealized pose information. As a result, the observed performance may represent a best-case scenario.

Chapter 5

Conclusion

This study presented the implementation and evaluation of an Extended Kalman Filter (EKF) for mobile robot localization using a ROS 2-based simulation framework. The localization system was developed for the final year project robot and evaluated using Gazebo for simulation and RViz for visualization under ROS 2 Jazzy.

To assess the effectiveness of the EKF, ground truth odometry obtained from the simulator was treated as the reference trajectory. Artificial noise was introduced to generate a noisy odometry path, which served as the input to the EKF. The estimated trajectory produced by the EKF was then compared against the ground truth using both visual and quantitative evaluation methods.

Visual comparisons of the trajectories demonstrated that the EKF-estimated path closely follows the ground truth path, while the noisy odometry path shows noticeable deviations due to accumulated noise. Quantitative evaluation using Root Mean Square Error (RMSE) further confirmed this observation. Across different trajectory lengths, the EKF consistently achieved lower RMSE values compared to the noisy odometry, indicating a significant reduction in localization error.

These results validate that the Extended Kalman Filter effectively reduces the impact of odometry noise and provides a more accurate and stable pose estimate. The implemented approach successfully demonstrates the practical benefits of probabilistic state estimation for mobile robot localization within a ROS 2 simulation environment.