Assignment: Implementing EKF-SLAM

Artificial Intelligence for Robotics

Due Date: [End-of-Day March 24, 2025]

1 Overview

In this assignment, you will complete the implementation of an **Extended Kalman Filter (EKF)-based SLAM system** for a robot operating in an environment with both **static and dynamic landmarks**. You will apply the mathematical equations provided in the codebase to correctly implement the **prediction and update steps** of the EKF.

You will also investigate the **effect of dynamic landmarks** on the SLAM performance and implement a method to **classify and filter** dynamic landmarks.

2 Objectives

By completing this assignment, you will:

- Understand the **EKF-SLAM algorithm** and its role in robot localization.
- Implement the **predict and update steps** for **robot state estimation**.
- Generate and **track static and dynamic landmarks**.
- Visualize and analyze **robot trajectory and SLAM accuracy**.
- Classify dynamic landmarks to **improve localization accuracy**.

3 Files Provided

You are given the following Python files:

1. `ekf.py` – Implements the EKF **predict and update steps** (incomplete).
2. `slam.py` – Simulates robot motion, landmark detection, and SLAM execution (incomplete).
3. `robot.py` – Defines the **robot movement and sensing** functions.
4. `plotmap.py` – Contains visualization functions for **plotting trajectories**.

4 Instructions

4.1 Part 1: Completing the Predict Step (15 Points)

Modify `ekf.py` to complete the `predict()` function:

1. Implement the **motion model update** using the equations:

$$\begin{aligned} x_{t+1} &= x_t + d\cos(\theta_t + \delta\theta_1) \\ y_{t+1} &= y_t + d\sin(\theta_t + \delta\theta_1) \\ \theta_{t+1} &= \theta_t + \delta\theta_1 + \delta\theta_2 \end{aligned}$$

2. Compute the **motion model Jacobian** G:

$$G = \begin{bmatrix} 1 & 0 & -d\sin(\theta_t + \delta\theta_1) \\ 0 & 1 & d\cos(\theta_t + \delta\theta_1) \\ 0 & 0 & 1 \end{bmatrix}$$

3. Compute the **predicted covariance matrix**:

$$\Sigma' = G\Sigma G^T + R$$

4.2 Part 2: Completing the Update Step (20 Points)

Modify `ekf.py` to complete the `update()` function:

1. Implement the **landmark initialization** if it has never been observed:

$$\begin{aligned} x_j &= x_r + r\cos(\theta + \theta_r) \\ y_j &= y_r + r\sin(\theta + \theta_r) \end{aligned}$$

2. Compute the **expected measurement**:

$$\begin{aligned} \hat{r} &= \sqrt{(x_j - x_r)^2 + (y_j - y_r)^2} \\ \hat{\theta} &= \tan^{-1}\left(\frac{y_j - y_r}{x_j - x_r}\right) - \theta_r \end{aligned}$$

5 Code Grading Criteria

Section	Description	Points
Part 1	Complete `predict()` function	15
Part 2	Complete `update()` function	20
Part 3	Implement SLAM logic in `slam.py`	10
Part 4	Implement dynamic landmark classification	5
Total		**50**

6 Report Grading Criteria

Section	Description	Points
Background	Background of EKF SLAM	10
Method	Describe how did you implement the method	20
Results and Analysis	Discuss the results	10
Format	Using IEEE Conference Format	10
Total		**50**

7 Submission Instructions

Please submit everything to Blackboard:

- Report in Pdf format.
- Functional Codes .