EKF-SLAM: Simultaneous Localization and Mapping Simulation

This project provides a Python-based simulation of the Simultaneous Localization and Mapping (SLAM) problem using an Extended Kalman Filter (EKF). The simulation demonstrates how a robot can navigate an unknown environment, build a map of landmarks, and simultaneously track its own position within that map, all while accounting for real-world uncertainties like motion and sensor noise.

The visualization, powered by matplotlib, dynamically shows the robot's true path, its estimated path, the true landmark positions, and the EKF's estimated landmark positions along with their uncertainty ellipses.

How to Run

Prerequisites

- Python 3.x
- NumPy
- Matplotlib
- SciPy

Installation

1. **Install the required Python libraries:**
   pip install numpy
2. pip install matplotlib
3. pip install scipy

Execution

To run the simulation, execute the main scriptl:

```
python main.py
```

A plot will appear and update in real-time, showing the SLAM process.

Project Structure

The project is organized into four main Python files:

- main.py: The main script that initializes the simulation, runs the primary loop, and handles all visualization.

- robot.py: Defines the Robot class, which simulates the robot's movement and sensor measurements in the environment, including the addition of noise to mimic real-world conditions.
- ekf_slam.py: Contains the core EKF_SLAM class. This class implements the Extended Kalman Filter logic, including the predict and update steps, state vector management, and data association.
- utils.py: A collection of utility functions used across the project, such as motion models, Jacobian calculations, angle normalization, and the Mahalanobis distance calculation.

Core Concepts Implemented

1. Simulation Environment

- **Robot** (**Robot class)**: A simulated agent that moves based on control inputs (v, ω). Its motion and sensor readings are intentionally corrupted with noise (Q_SIM_MOTION, R_SENSOR_TRUE) to simulate the imperfections of a real robot.
- **Map (MAP_LANDMARKS)**: A predefined set of true landmark coordinates that the robot's sensors can detect.
- **Measurements**: The robot perceives landmarks as range and bearing measurements, which are also subject to noise.

2. Extended Kalman Filter (EKF)

The EKF is used to estimate a state vector that includes the robot's pose (x,y,theta) and the 2D coordinates (x_i,y_i) of all observed landmarks. The algorithm operates in a two-step cycle:

1. **Prediction**: The EKF uses the robot's motion commands (odometry) to predict its new pose. This step increases the uncertainty in the system, which is reflected by the growth of the covariance matrix P_est.
2. **Update (Correction)**: When the robot observes a landmark, the EKF uses the measurement to correct its state estimate. This feedback loop reduces the uncertainty of both the robot's pose and the observed landmark's position.

3. Data Association

A critical challenge in SLAM is determining whether a new measurement corresponds to a previously seen landmark or a new, undiscovered one. This implementation uses the **Mahalanobis distance** to solve this:

- It calculates the Mahalanobis distance between a new measurement and all known landmarks in the state vector.
- A **chi-squared test** (ASSOCIATION_THRESHOLD_PERCENTILE) is used to

determine if the best match is statistically valid.
- If a valid match is found, the EKF updates the existing landmark.
- If no valid match is found, a new landmark is initialized and added to the state vector.

Configuration

Key parameters which can be manipulated in main.py:

- DT: The time step for the simulation.
- SIM_TIME: The total duration of the simulation.
- Q_ODOM_EKF: The motion noise covariance matrix used by the EKF. It represents the filter's *belief* about the robot's motion uncertainty.
- R_EKF_SENSOR: The measurement noise covariance matrix used by the EKF. It represents the filter's *belief* about the sensor's uncertainty.
- ASSOCIATION_THRESHOLD_PERCENTILE: The confidence level for the chi-squared test used in data association. A value of 0.99 means we are 99% confident in our association.