CS-475 Assignment 1
Kalman Filter

Michael Maravgakis
maravgakis@csd.uoc.gr

Release date: 05/03/25
Deadline: 11/03/25

1 Introduction

In this assignment, we are going to help our turtlebot to localize itself by using the Kalman Filter (**KF**). The Kalman filter allows us to combine a variety of potentially erroneous/inaccurate sources and obtain an estimate about the state of our robot that is more accurate than any individual source. Our robot has 2 sources of positional data:

1. It is equipped with a GPS sensor which returns the position of the robot (x,y).
2. It has an IMU sensor that returns how much the robot has turned (yaw).

In a perfect world, we would just need the IMU sensor in order to perform successful localization. In practice the wheels are subject to slip which may cause the robot to believe it's turned a different amount than it actually has; this error accumulates over time (aka "drift"). While GPS simply is not accurate enough (its resolution isn't high enough) on its own for the tasks we want our robot to do. For the purpose of this assignment we'll assume that all sensing uncertainties can be modeled in terms of Gaussian white noise and therefore the KF can be utilized to filter data streams and obtain more reliable estimates about the robot's position.

2 Setup

Copy the directory `assign1` with its contents inside your workspace:

```
/home/<user_name>/475_ws/src
```

To run the simulation and visualize the data with rviz, simply compile your workspace and run:

```
roslaunch assign1 burger.launch
```

Then you should see the robot moving and also the visualization of the IMU-based odometry (dead-reckoning) and the GPS measurements. Everything is ready for you regarding the package "setup", you don't need to change anything in the CMakeLists.txt or package.xml. Inside this package you will find the launch/ folder, which contains the launch file (burger.launch). This file provides a convenient way to start up multiple nodes and a master(`roscore`), as well as other initialization requirements such as setting parameters.

3 Implementation

For this assignment, you will implement the Kalman Filter by fusing the noisy measurements of the orientation and the GPS to get a more accurate state estimate. Practically, you will need to code the mathematics from slide 17 of Bayes Filter Implementations presentation (2.kalman.ppt). Write your KF inside the src/kalman.py script. Once you get the state estimate, you should publish it using the **Odometry** type ros message. The measurement error of your sensors is gaussian and for the GPS is **0.3m** while for the IMU orientation is **0.01rad**.

Simulation

This time, instead of Gazebo (running on the background) you are going to use the Rviz. Rviz is a powerful tool, built-in for ROS and helps with the visualization of data. You can start the simulation by opening a terminal window and type:

```
roslaunch assign1 burger.launch
```

Then, an Rviz window (displayed at Figure 1) should appear and the robot will start moving forever with **constant linear speed=0.3m/s**. When you have completed your implementation, Rviz will subscribe to the **"/kalman_est"** topic and visualize the state of your robot. Start up the kalman node with the command:

```
rosrun assign1 kalman.py
```

In the above image, the purple dots illustrate the noisy GPS measurements and the red arrows the noisy odometry. Your kalman filter estimates will be illustrated by green arrows when published to the appropriate topic.

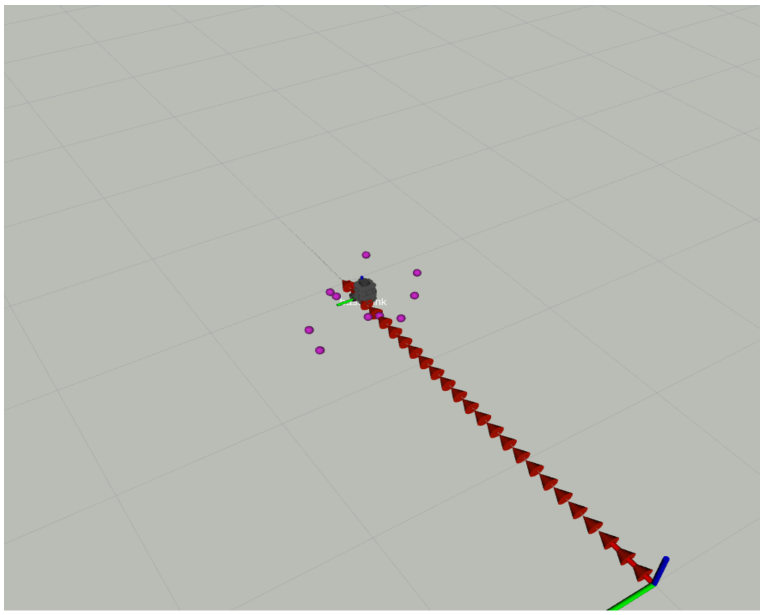

Figure 1: Screenshot taken from Rviz

4 Submission

Send your node (`kalman.py`) attached via email at: **maravgakis@csd.uoc.gr** Don't forget to add at the top of the file your name and your registration number as a comment. The subject of the email should be the following: **[CS475]: Assignment 1 submission**. The deadline is due to **Tuesday 11/03/2025 23:59**