Institute for Aerospace Studies
UNIVERSITY OF TORONTO

LABORATORY IV

Kalman Filtering

ROB301 Introduction to Robotics
Fall 2020

1 Introduction

In the previous laboratory exercise, you devised and implemented various controllers to follow a well indicated path. However, you were never tasked with the problem of knowing *where* along the path your robot was at any given time. The task was simply to continue on the path. In many circumstances, you would not even be given a path to follow, only a map and you have to find where you are on the map. (In some cases, you don't even have a map!) The problem of *where you are* is the problem of *localization*. This is central to just about every real-world robotics application and it is the subject of this lab.

Imagine yourself as a budding robotics engineer working at Amazon Robotics and you're tasked with designing and implementing a delivery robot that will allow Amazon to deliver packages autonomously and more efficiently to waiting customers. Your boss, who recently moved to Toronto from Seattle (having recently left his position at another major corporation of the new economy), is pondering how the robot will navigate the confusing streets of Toronto. As any proud Torontonian might, you suggest using the CN Tower to localize the delivery robot.

Suppose your robot's designated route is to be along Bloor Street, making its deliveries to various homes and businesses along the street. In this incarnation of the robot, it would use line following to keep it on Bloor Street but you still have to determine where along the street it is to enable the robot to establish the address of a building. An odometry estimate can be accurate for short trajectories but it becomes unreliable as sensor drift and wheel slippage lead to substantial growth in position error over time; therefore you propose localizing with the help of a well established landmark, the CN Tower.

You recognize that owing to the rapid development of condominium buildings (whose architecture generally leaves much to be desired), there are stretches of road where the Tower is occluded. Having recently discovered the wonders of Kalman filtering, you propose using an extended Kalman filter (EKF) to accomplish the task.

2 Objective

The objective of this exercise is to acquire hands-on experience with Kalman filtering for robot localization using the Turtlebot 3 Waffle Pi. You will be required

- *To implement an extended Kalman filter for one-dimensional localization*
- *To test the filter on an idealized robotic delivery task*

You will be provided with a test course for your robot.

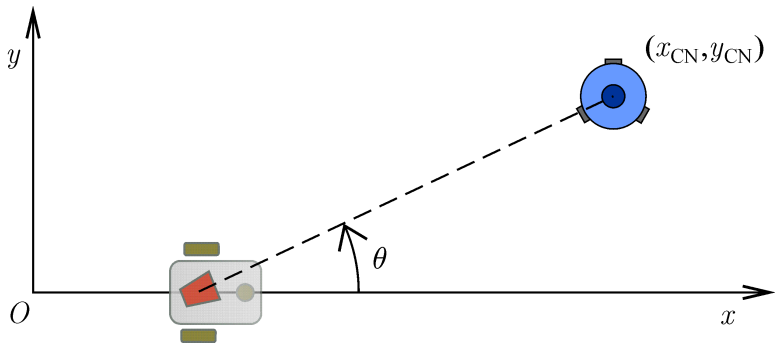

Figure 1. Scenario for delivery robot

3 Preparation

The scenario you will be addressing is shown in Figure 1. The controller you developed in Lab III for line following can be used here and accordingly you can assume that the robot's "workspace" is a one-dimensional straight line, where x gives the position of the robot relative to some designated reference point in a global frame. The control input is the speed command u. (Note that the line-following controller is applied only to keep the robot on the line; it need not figure into the state equation for the robot.)

The sensor reading is the bearing θ of the CN Tower, measured by the Turtlebot's laser sensor (lidar), with respect to the x axis subtended at the robot. The global position $(x_{\text{CN}}, y_{\text{CN}})$ of the Tower is to be considered given.

You are asked to do some preparatory work for this lab. You will no doubt recognize some similarity of the present task to that which you addressed in Problem Set II. Prior to writing any code, therefore, you are to complete the following:

1. Establish the state model in continuous time.
2. Establish the measurement model in continuous time.
3. Linearize your models to facilitate an extended Kalman filter.
4. Express the linearized state and measurement models in discrete time, which is what you will code.

Lab Deliverable 1. Include these four expressions in your report.

Table 1. Delivery route

Address	Coordinates [m]
4 Bloor St W	$(2.1, 0)$
8 Bloor St W	$(3.2, 0)$
16 Bloor St W	$(5.5, 0)$

4 Assignment

Having completed the preparation, you are required to implement and test your extended Kalman filter on your robot using the test course provided in the lab. You will moreover have to test it under different conditions.

The intended route for your robot consists of three addresses (and corresponding coordinates) as given in Table 1.

4.1 Line-Following Control

To begin, apply again your line-following control (from Lab III): Ensure that the robot can smoothly follow the designated line provided in the test course (in `lab_04_empty.launch`). If there's any additional tuning of gains that you deem necessary this is the time to do it. There is no need to demonstrate this capability again to the TA.

4.2 Unfiltered Localization

The first demonstration will be to localize the robot without the benefit of any filtering of sensor data. This will provide a baseline capability against which you can assess the benefits of Kalman filtering.

To accomplish this task, you have been provided with starter code in `lab04_kf.py`, which subscribes to the `/cmd_vel_noisy` topic. Using this velocity input, and `rospy`'s timing functions, implement a simple localization node that integrates the velocity over each discrete-time period to produce an estimate of the robot's position. Note that `/cmd_vel_noisy` is the `/cmd_vel` topic with Gaussian noise added. (Owing to our limited amount of space, odometry error does not significantly grow; we add the noise to simulate traveling over the longer distances that a delivery robot would be required to traverse.) This node is initialized when running the Lab IV launch files—for this task, use `lab_04_empty.launch`. The robot will be centered at the origin, and the "road" is aligned with the robot along the x-axis.

Test the robot's ability to deliver packages (we'll use our imagination) to the

three indicated addresses; we suggest a translational speed of $u = 0.05$ to $0.1\ \mathrm{m\,s^{-1}}$. At each location, stop the robot for 4 s before continuing. Observe the robot's true position (through the topic `/tf`) and compare it to your estimated position.

Lab Deliverable 2. When you have successfully completed this task, you are required to demonstrate functionality by recording the successful run. Note that the robot shouldn't be stopping at the true poses due to errors in the observed command velocities, so do not spend time trying to account for the error buildup in your code somehow. Record the following topics:

```
$ rosbag record /scan_angle /tf /odom
```

prior to running your code. Remember to include in your report a summary of what the robot does within your rosbag file, and describe if you completed the task, or why you were not able to.

4.3 Extended Kalman Filter

Now implement the EKF that fuses odometry measurements with the angular measurements of the Tower to localize your robot. You will be required to test your localization procedure under different conditions; that is, you will be required to show the following to the TA:

Lab Deliverable 3. Demonstrate your localization procedure on the test course with *no* obstructions (use `lab_04_empty.launch`) by recording the same topics as specified in Deliverable 2. Plot the estimated position and the covariance over time, and include these plots in your report.

Lab Deliverable 4. Demonstrate your localization procedure on the test course with obstructions (use `lab_04_occlusions.launch`), by recording the same topics as specified in Deliverable 2. Plot the estimated position and the covariance over time again. How do these compare to the results with no obstructions? Can the estimation algorithm tolerate interruptions in sensor inputs? Comment on this in your report.

Extra Extra. An EKF should be sufficiently robust to deal with erroneous measurements and even missing measurements. Another way to test its robustness is to start from various erroneous initial positions. Try it. Plot the estimated position and the covariance over time again. Does the filter still converge (in terms of covariance)? How much error can be tolerated? This is for your own interest, and there will not be bonus marks allocated if you include this in your report.

5 Concluding Remarks

By now you should appreciate the problem of localization in robotics. Not much can be accomplished by a robot if it cannot establish with some degree of accuracy where it is in the world. Kalman filtering provides a powerful tool in estimating a robot's pose in the face of erroneous initial conditions, erroneous measurements and even missing measurements but it is important always to beware of the limitations of one's tools.

6 Additional Resources

1. Introduction to ROS, ROB301 Handout, 2020.
2. Robotis e-Manual, `http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/`.
3. "SSH: Remote control your Raspberry Pi," The MagPi Magazine, `https://www.raspberrypi.org/magpi/ssh-remote-control-raspberry-pi/`.
4. Official ROS Website, `https://www.ros.org/`.
5. ROS Wiki, `http://wiki.ros.org/ROS/Introduction`.
6. Useful tutorials to run through from ROS Wiki, `http://wiki.ros.org/ROS/Tutorials`.
7. ROS Robot Programming Textbook, by the TurtleBot3 developers, `http://www.pishrobot.com/wp-content/uploads/2018/02/ROS-robot-programming-book-by-turtlebo3-developers-EN.pdf`.