CSC477 – Introduction To Mobile Robotics

assignment 3, 15 points
due: Nov 18, 2020, at 6pm

Course Page: http://www.cs.toronto.edu/~florian/courses/csc477_fall20

Overview: This assignment deals with estimation and mapping. You will implement occupancy grid mapping based on 2D LiDAR observations and known poses. You will also implement a simplified version of GraphSLAM for range observations to uniquely identifiable landmarks. Finally, you will write down the main components of an Extended Kalman Filter for a localization problem.

1 Occupancy grid mapping (5pts)

In this exercise you are going to implement parts of the occupancy grid mapping system discussed in class. In particular, you are going to map an environment based on known odometry estimates and known 2D laser scans. Do `git pull` under your CSC477 repository to get the starter code. The functionality that you need to implement is marked using to-do comments in the file `estimation_assignment/python/occupancy_grid_mapper.py`. To run your code, `cd path/to/csc477_fall20/estimation_assignment` and execute the following commands on three different terminals:

```
rosbag play laser_and_odometry.bag
roslaunch  estimation_assignment occupancy_grid_mapping.launch
           odometry_orientation_noise_std_dev:=2.5
           odometry_position_noise_std_dev:=0.1
rosrun rviz rviz
```

When rviz initializes, go to `File` > `OpenConfig` and then load the configuration file in `estimation_assignment/resources/comp417.rviz` which is going to start the visualization of laser scan messages, frames of reference, and debugging visualizations. Save this configuration file as the default in your `/home/username/.rviz/default.rviz`, so you won't have to do this every time you restart rviz. In the `roslaunch` command given above, the standard deviation of the odometry orientation noise is given in degrees. In this example the true yaw of the robot is corrupted by random noise from $\mathcal{N}(0, 2.5^2)$. Similarly, the true position is corrupted by random noise from $\mathcal{N}(0, 0.1^2)$, which is expressed in meters. `laser_and_odometry.bag` is a recording of laser scans and odometry data in the environment you used in Assignment 1. The expected result for perfect odometry looks similar to Figure 1.

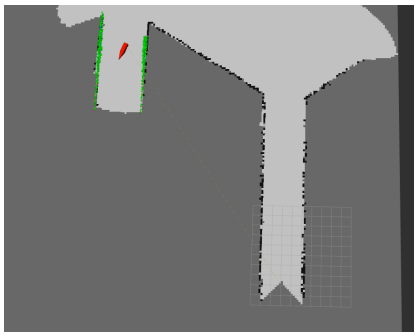

Figure 1: Part of the robot's mapping task during the (0,0) noise trajectory.

What you need to submit: 5 images of maps produced by your mapper, with noise parameters (0, 0), (2.5,

0), (5, 0), (2.5, 0.1), (5, 0.1) respectively. Your images should be named `map_[0|1|2|3|4]_firstname_lastname.png`. Also, submit a video recording of the rviz visualization for the odometry noise combination (0,0), demonstrating your map being built from beginning to end. Your video should be named `map_0_firstname_lastname.mp4/avi/ogg`.

2 Least squares localization (5pts)

In this exercise you are going to solve the localization problem in a map of known landmarks. In particular, assume your robot has 2D state $\mathbf{x}_t = [p_x(t)\ p_y(t)]^T$ and an omnidirectional motion model $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}_t\delta t + \mathbf{w}_t$, where $\mathbf{w}_t \sim \mathcal{N}(0, \sigma_w^2\mathbf{I})$ is zero-mean Gaussian noise. The robot is moving through an environment that has L static landmarks $\mathbf{l}_1, ..., \mathbf{l}_L$ whose positions are known. Occasionally, the robot makes the following set of measurements: $z_t^{(i)} = ||\mathbf{x}_t - \mathbf{l}_i|| + n_t$ where $n_t \sim \mathcal{N}(0, \sigma_n^2)$, for some of the available landmarks. Your goal is to estimate the sequence of states $\mathbf{x}_{1:T}$ by minimizing a least squares cost function.

You can find starter code in `estimation_assignment/python/localization.py`. It provides the set of measurements made at each timesteps. Your task is to implement the cost function using `numpy` and the nonlinear least squares solver in `scipy.optimize.least_squares`. There are to-do comments in the code that explain in detail what you need to implement. After you are done with your implementation, the expected outcome is the following figure:

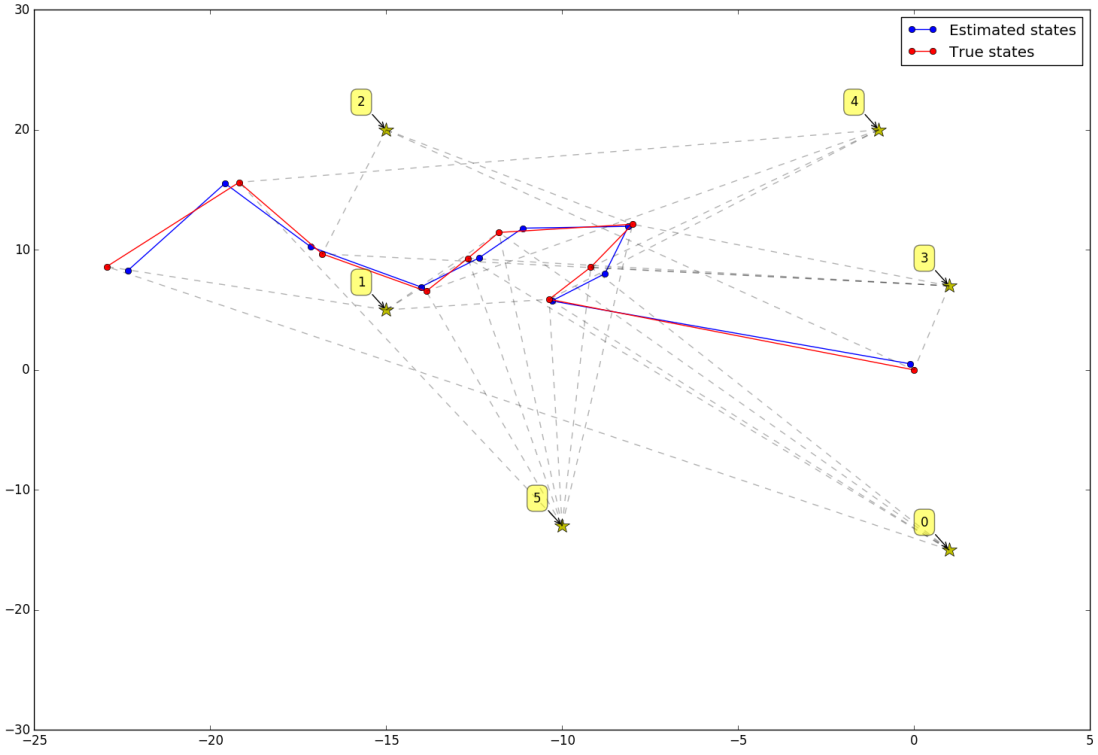

Figure 2: The robot starts at (0,0) and moves throughout the world, making measurements of static, known landmarks.

What you need to submit: your code in the file `estimation_assignment/python/localization.py`.

P.S.: You can think of cell phone localization based on static cell towers as a particular application of this problem.

3 Extended Kalman Filter (5pts)

Assume you have a robot that follows Dubins car dynamics on the plane. Its state is $\mathbf{x}_t = [p_x(t)\ p_y(t)\ \theta(t)]$. Also assume that you have three landmarks $\mathbf{l}_i = [l_x^{(i)}\ l_y^{(i)}]$ in the world. At each point in time the robot observes all three landmarks, with some sensor noise. The measurement model consists of the Euclidean distances from the robot's current state to each of the three landmarks, plus noise. Formulate an EKF estimator for this problem. Specifically, formally define the dynamics model, the observation model, their Jacobians, and the covariances of their noise processes. There is no implementation required for this question, but if you want to do it, you can find starter code at `path/to/csc477_fall20/filtering_examples/python`. Submit your work for this question as a pdf document or as a scanned image of your handwritten solution.

4 How to submit

Submit all your work in a file called `estimation_assignment.zip` that contains your extensions to the provided starter code, as well as the five images, the video, and the pdf file. Submissions will be done on Quercus.