Preamble

Before starting, read the entire manual at least once from beginning to the end.

In this final project, you will implement a Kalman Filter, perform tests and write a report.

Final project policies:

- The final project is **individual** and requires code implementation and a report (see last page for submission details).
    - Do: talk to each other and have discussions on your approaches, reach out to the instruction team for clarifications/doubts.
    - Don't: copy each other's work, copy from online resources, ask someone else to do your work.
- The final project is graded on a 0-100 scale and is 30% of your final grade (not including bonus)
- **Project files must not be disclosed to any other person.**
- **Due date**: **Dec. 16th, 11:59PM (+15 min for uploading)**

Do not wait until the last days to start! Reach out for help during drop-in sessions, on Piazza, and/or schedule an appointment.

Description

In Lab 2, you implemented the Particle Filter to localize the pose of the robot, where the motion model was not required as you had to localize only one pose. In this final project, you are asked to implement the **Kalman Filter Localization** for a moving robot using IMU and encoders. A partially implemented code is provided on LEARN (both in Python and Matlab).

For this implementation, you can choose either of the following options:

a. Implement in Python as ROS package (+5 bonus points for this option)
b. Implement in Python (no ROS package)
c. Implement in Matlab

While the ROS package is recommended, choosing options b. or c. will not incur any penalty (you can get full mark). Choosing option a. will result in a bonus of maximum 5 points on the final project grade (depending on the quality and working status of the implementation).

For the implementation, it is recommended to use the provided code, change the necessary blocks of code, and make improvements as you see need. But you can also implement it from scratch if you prefer (please indicate it clearly in the report if you choose the latter).

Steps for implementing the Kalman Filter Localization:

1. The first step in the implementation requires the motion model of the robot. For Kalman Filter, this model must be represented as a linear system. In this part you should define matrices A, B and Q (see Fig. 1). A and B being the matrices describing the system, and Q being the covariance matrix characterizing the noisy state.

2. The second step is to characterize the measurements from sensors. In this part you should define matrices C and R (see Fig. 1), R being the covariance matrix characterizing the measurements.
3. The third step is to implement the Kalman Filter algorithm and feed the motion model and measurements to the algorithm to obtain the localization. This should be a straight-forward implementation of the pseudo code presented in the lectures (Fig. 1). This part is already implemented, and you should not need to change it.

Note: the algorithm provided in Fig. 1 is slightly different from the one you have in the lecture slides, though it is mostly a matter of notation (for example, Q and R are swapped in the lecture slides and on the Probabilistic Robotics book). The provided code follows the notation in Fig. 1.

Start with initial guess:

$$\hat{x}_{0|-1} = E[x_o] = mean$$
$$P_{o|-1} = var(x_o)$$

Prediction

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1}$$
$$P_{k|k-1} = AP_{k-1,k-1}A^T + Q$$

Update

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K(y_k - C\hat{x}_{k,k-1})$$
$$K = P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1}$$
$$P_{k,k} = (I - KC)P_{k|k-1}$$

Fig. 1 Kalman Filter algorithm

Notes on implementation:

- **To do**:
  - Steps 1 and 2 in the code illustrate an example that you need to adapt with the model of the robot and the appropriate sensor data. Refer to the lecture of Dec. 6th (video uploaded on LEARN) for further understandings.
  - In the main loop of the Kalman Filter, there is a simulation using added random noise for illustration and testing purposes, you need to modify this part as you have actual data obtained from the execution of a path in the simulator.
- Do NOT implement a hard coded/ad-hoc tuned solution, the implementation should work for any given path.
- You are allowed to use any ROS/Python/Matlab library/toolbox but not a direct implementation of the Kalman Filter.

Testing of the Kalman Filter Localization:

1. You are given a bag file (file.bag) that contains information recorded with the simulator where the TurtleBot3 is performing a known path. All topics were recorded, the ones you may be more interested in are the following:
    a. /tf
    b. /imu
    c. /joint_states (encoders)

    You can check the type of the messages in the .yaml file.

    Useful links to message types:

    - https://index.ros.org/p/sensor_msgs/
    - https://github.com/ros2/common_interfaces/blob/humble/sensor_msgs/msg/Imu.msg
    - https://github.com/ros2/common_interfaces/blob/humble/sensor_msgs/msg/JointState.msg

    Note: the path that you are given is a slightly curved path (almost straight) as you are implementing a linear Kalman Filter which would not be able to cope with large variations in angles. If you want to test your filter with other paths, remember this limitation.
2. Process the data you need from the bag file into your code (similar to what you did for Lab 2, you can modify the bagreader that was provided for Lab 2, or any method you've used before to read data from bag files) and run the Kalman Filter Localization code you implemented to obtain the estimated state of the robot during the entire executed path.
3. Use /tf as ground truth to assess the quality of your localization. Compute the mean square error (MSE) as a measure of performance. Tune your parameters to assess how they change the performance of the Kalman Filter.

Write your report:

- In writing your report, make sure to respect the format (see Deliverables), and write the theoretical part based on your understanding (do not copy-paste from textbooks and/or online resources).
- Summarize your results concisely and refer to them for your discussion.
- Discuss the performance of the Kalman Filter, choices you made for the implementation/performance, possible future improvements, etc.

Deliverables

Due date: **Dec. 16th, 11:59PM (+15 min for uploading)**

Grading scheme:

- Code: 40/100, code must be adequately commented
- Report: 60/100, structure as below

Report:

- Page limit: 4 (not including cover, bonus page, and Appendix) – penalty applies if limit is exceeded
- Format:
    - Nicely formatted, either single or double column
    - Figures, tables labeled, no superfluous figures, no excessive displays of raw data
    - Appropriate references provided, if necessary
    - No table of contents needed
- Content of the report:
    - Introduction
    - Theory and Methodology
    - Results
    - Discussion
    - Conclusion

This is a **two parts deliverable – deliver both for grading**:

1. Crowdmark: submit a pdf version of your report
2. LEARN Dropbox: submit your code