Project4

Overview

The goal of this lab is to evaluate the performance of a KF, EKF, and UKF alongside each other.

As a group, determine what path(s) you want to record with TurtleBot and share these paths among the group. Students may also share a common ROS 2 node (e.g. create parameters in the ROS 2 node that select for the type of each algorithm while using the same subscribers, publishers, and plotting tools). Each student should work through steps 4-8 individually, compare results, and create a unified response to each step (steps are described in the `Project4.pdf` at the provided github link).

All plots and answers to questions should be placed in a markdown file in your repository. Provide instructions in your repository such that I could generate any of the plots described in step 5. If a question requires a written response, the written response is expected to be 1-2 paragraphs of your own writing.

1. Connect to the TurtleBot

1. Go through Ian's Lab 4 (`lab_4.pdf` in this repository).

2. Record TurtleBot Path

1. Use the teleop feature described in Ian's Lab 4 to drive the robot in a path. **The path must start and end at the same point and have more than one turn**
2. Use `ros2 bag record -a`. **Make sure this command is executed on the remote laptop, not the turtlebot!**
3. All the data can be replayed without the turtlebot using `ros2 bag play <bag file name>`

Note: It may be helpful to record paths of different lengths. **Only one path is required for analysis**.

3. Create ROS 2 Node

Transforms

We want to follow the REP 105 standard for coordinate frames. Therefore, the pose of the BurgerBot should be expressed in the **odom** frame. However, inertial and wheel tick measurements are with respect to the robot's internal frame, so these values are expressed in the **base_link** frame (also known as *proprioception*). **Do not forget to account for these different frames in your state estimators!**

Input Topics

- `/joint_states`

- Type: `/sensor_msgs/msg/JointState`
- Rate: 20 Hz
- Use only the `position` field

- `/imu`
  - Type:`/sensor_msgs/msg/Imu`
  - Rate: 20 Hz
  - Use only the `linear_acceleration` and `angular_velocity` measurements
- `/cmd_vel`
  - Type: `/geometry_msgs/msg/TwistStamped`
  - Rate: 10 Hz
  - Hint: Incorporate this into your process equations in your KF system models.

Output Topics

Paths

- `/localization_node/kf/path`
  - Type: `nav_msgs/Path`
  - Rate: 20 Hz
- `/localization_node/ekf/path`
  - Type: `nav_msgs/Path`
  - Rate: 20 Hz
- `/localization_node/ukf/path`
  - Type: `nav_msgs/Path`
  - Rate: 20 Hz

Odometry (Pose and Covariance in the odom frame)

- `/localization_node/kf/odometry`
  - Type: `nav_msgs/Odometry`
  - Rate: 20 Hz
- `/localization_node/ekf/odometry`
  - Type: `nav_msgs/Odometry`
  - Rate: 20 Hz
- `/localization_node/ukf/odometry`
  - Type: `nav_msgs/Odometry`
  - Rate: 20 Hz

Other Outputs

Residual and state covariance plots (may publish more topics and use ROS 2 `rqt_plot` or any python plotting library). See "Analysis" section for more details.

4. Implement KF, EKF, and UKF

Implement a KF, EKF, and UKF in a single ROS 2 node such that all algorithms can run simultaneously. The Kalman Filter wikipedia page is a good resource (Scroll down to the Nonlinear filters section to see info on the EKF and UKF).

Show the process and measurement equations for each algorithm. Show the value of each element in each vector (x and z) and matrix (F and H) of the state space model. (15 pts)

5. Tune the Algorithms

Run through various parts of the "Analysis" section to see how well your algorithms are working. As we discussed in lecture, these algorithms weight the predicted measurement with the current measurement to create an optimal estimate. These weights are primarily determined by the **process noise** Q and **measurement noise** R. In this lab, we do not directly give you the values for these matrices (which is often the case in real life). Some things to try:

- Calculate the covariance of the measurements when the robot is motionless
- Find the IMU's and wheel encoders' specifications
- If the path is too noisy (e.g. jagged or not smooth), try increasing Q or decreasing R. **Remember, Q and R must by symmetric and positive semidefinite!**
- Try leaving the off-diagonal terms as zero to start (no correlations).
- Try converting the diagonal terms (e.g. variances) to standard deviations to get more interpretable units.

6. Analysis

Since no ground truth is available for a localization system (which is often the case), analyzing the performance of optimal state estimators can be difficult. However, there are a few ways to check these algorithms:

6a. Smoke Test

Visualize the paths and odometry topics in RVIZ. Is there anything out of the ordinary?

Make a short screen recording of RVIZ showing paths and odometry topics populated by your filters (10 pts)

6b. Reducing Epistemic Uncertainty: Statistics of the Residual

The residual compares the true measurement with the predicted measurement. The residual should be zero-mean, white, and conform to the calculated covariance. If the residual is correlated (e.g. not white), this means there is information not being modeled by the KF.

Plot each residual (including +/- 3σ bounds) at each time step for each algorithm. For each algorithm, briefly describe how well you believe the algorithm is working by how well the residual matches the expected statistics (5 pts)

To show improvements from tuning, select one algorithm, and plot each residual for the poorly tuned algorithm and the well-tuned algorithm. Briefly explain how you tuned the algorithm (5 pts)

6c. Covariance Stability

An interesting thing about the Kalman filter is the Kalman gain **does not depend on receiving real measurements**. This implies we can look at the predicted covariance as an indicator of how well the algorithm converges to the true value.

*Plot each of the diagonal elements of the state covariance matrix at each time step for each algorithm **without using measurements**. By looking at the covariance of the states over time, for each algorithm, determine whether the algorithm is stable or not. Briefly explain how you could make the algorithm more stable. (5pts)*

6d. One Ground Truth Point

There is actually one easy way to obtain one ground truth evaluation. After driving your robot in a loop, the estimated robot pose should be near the exact pose where it started.

Calculate the absolute translational and rotational error from the beginning pose (Hint: Should be (0,0,0)). Which algorithm did the best? (2 pts)

7. Decision

Engineering is all about making decisions to solve problems.

Select your "winning" algorithm and briefly describe why. (3 pts)

8. Improvement

Engineers always try to figure out what could be changed in the next version.

Briefly describe how you would go about creating a more accurate localization system. (5 pts)

Extra Credit: Covariance Ellipses

Instead of visualizing the covariance ellipses using built-in tools in RVIZ, calculate the covariance ellipse for each state and visualize using the `/visualization_msgs/Marker` ROS 2 message type with the `LINE_STRIP` type field selected.

Create a short screen recording showing your covariance ellipses plotted along the paths for each algorithm (5 pts)