Lab

The goal of this lab is to evaluate the performance of a KF, EKF, and UKF alongside eachother. Mark the position and orientation of the BurgerBot on the ground. Start your state estimation ROS 2 node. Drive the robot in an oval path that returns to the same initial position and orientation (Recommended: Record the data using `ros2bag` so you do not need to be in the lab the entire time). Visualize the paths from each algorithm in RVIZ.

1. Connect to the BurgerBots

2. 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

`wheel_ticks`

`imu`

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.

3. 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.

Show the process and measurement equations for each algorithm

4. 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 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.

5. 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:

5a. 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.

5b. 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), these 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.

5c. Covariance Stability

An interesting thing about these algorithms 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. By looking at the covariance of the states over time, for each algorithm, determine whether the algorithm is stable or not. How could you make the algorithm more stable?

5d. One Ground Truth Point

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

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

6. Decision

Engineering is all about making decisions to solve problems.

Select your "winning" algorithm and briefly describe why.

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