Week 3 Lab – Sensor fusion with Kalman Filters

This lab will lead you through the process of constructing a sensor fusion Kalman Filter. This mirrors what you saw during Week 3 lectures.

In contrast to previous weeks, the remaining lab sessions will require more independent thought. It is very likely that you will need to refer to the lecture notes on SurreyLearn in order to complete the exercises.

Your solution to this week's lab will form your entry into next week's "Sensor Fusion Challenge". The lab itself is designed to be undertaken individually. However, before next week's session you should meet with your group members and come up with a joint solution based on what you each did in the labs.

It is important that you follow the structure of this lab to ensure that you can easily switch between recorded and live data feeds during next week's challenge.

Exercise 1: Playing back sensor data

First we must tell ROS what turtlebot we are using. Run the following command to permanently add this to every new bash terminal. Make sure to start a new terminal afterwards.

```
echo " export TURTLEBOT3_MODEL=burger" >> ~/.bashrc
```

Next, in your catkin workspace, create a new package called lab3. Create a "launch" folder inside the lab3 package, and create ex1.launch. This lab will primarily deal with playing back recorded data; therefore at the top of your launch file you should include the following setting to cause nodes to respect the timestamps of the recordings

```
    <param name="use_sim_time" value="true" />
```

The recorded data will be from a real turtlebot. You should also add the following lines to your launch file, to ensure that the robot's URDF is loaded.

```
<include file="$(find turtlebot3_bringup)/launch/includes/description.launch.xml">
        <arg name="model" value="burger" />
</include>
<node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher"/>
```

Next, download the rosbag file containing the recorded robot sensor data from SurreyLearn. For reference, the trajectory followed by the robot is roughly a figure of eight.

In a terminal, begin playing back one of the rosbag files. Make sure to pass the --clock argument to rosbag so that it simulates the timings from the bag file correctly:

```
    % rosbag play --clock bagfilename
```

In the terminal you can display a list of all the topics currently being published from the bag file, and the -v flag will tell you the message type of each topic:

```
    % rostopic list -v
```

The IMU sensor (topic /imu_corrected) cannot be easily displayed in RVIZ by default, but you can look at the messages in the terminal while the bag is playing, using

```
    % rostopic echo topicname
```

Which parts of the message is the IMU actually returning?
Can you see any interesting biases in the IMU data?
Where do you think these come from?

Next add RVIZ to your launch file and run it in a terminal. You should now be able to run your launchfile while the bag is playing, and look at the wheel odomety (topic /odom_corrected).

Finally, for the GPS sensor (topic /gps/fix) you cannot visualize the data directly in RVIZ. Instead you should add a navsat_transform_node to your launch file. Run the launch file and use rqt_graph to check if the navsat_transform_node is receiving the data it needs. As outlined in the lecture notes, it should subscribe to 3 things, the wheel odometry, the imu and the GPS. However, because of differences in topic names it is currently only receiving the GPS.

Remember that you can remap topic names by adding the following within the node in your launch file. Add two of these for the IMU and odometry

```
    <remap from="default_subscribe_topic" to="actual_subscribe_topic"/>
```

Once your navsat_transform_node appears to be subscribed to the 3 correct topics, try to display the resulting output topic (/odometry/gps) in rviz.

Do you notice any difference in behaviour of the wheel odometry and GPS? Try displaying them at the same time.

Exercise 2: Kalman Filter

We will now create a number of Kalman Filters and explore their behaviour and the effect of their settings. During this exercise we will focus on single-sensor Kalman Filters (i.e. simply combining motion-model prediction with sensor readings).

In the folder src/lab3/launch/ make three copies of your ex1.launch file named ex2_odom.launch, ex2_gps.launch and ex2_imu.launch.

Every launch file should include an ekf_localization_node with a single input topic and input configuration specified. Refer to the lecture notes for configuration of the EKF node for various input message types. **Note that the GPS launch file is the only one which will also need the navsat_transform_node in addition to the EKF.**

Run each launch file in turn, use RVIZ to examine the output of the Kalman filter (topic /odometry/filtered). For all except the IMU you can compare this against the original (unfiltered) sensor data. For the IMU, remember to remove the bias as mentioned in the lecture, and you may even disable some of the unreliable parts of the message. Do not worry if some sensors cannot produce a reliable KF output on their own.

Try toggling the covariance display on and off. If you have any difficulty making your EKF node work, you can add output="screen" to your node tag in the launch file to see what is happening.

Can you identify any advantages or disadvantages of the filtered output compared to the original sensor data?

Exercise 3 – Sensor Fusion

Make a copy of the ex2_gps.launch file called ex3.launch. Now modify the ekf_localization_node in this file so it includes both the odometry (odom1) and IMU (imu0) sensors, and the output of the navsat_transform_node (odom0). Remember this output is published as an additional odometry message on the topic /odometry/gps. You should observe a significant change in behaviour of the output (remember to try toggling the covariance display on and off).

If you notice a mismatch between the orientation of the robot and the direction of travel, try setting the imu0_relative parameter to true.

What differences do you observe in behaviour or accuracy compared to the unfused sensors?

This launchfile comprises your entry into next week's sensor fusion challenge.

Consult with your group members and bring one such launchfile per group on a USB stick to next week's session

Make sure that your chosen launchfile doesn't include a rosbag player node, and don't set use_sim_time

Exercise 4 – Sensor Fusion

If you still have time, you may wish to spend the rest of the session exploring how the Kalman filter settings affect the output, in order to improve your submission for next week's challenge. An exhaustive list of parameters can be found here (with GPS specific parameters here). However, the most important to experiment with are:

Week 3 Lab – Sensor fusion with Kalman Filters

1. `[sensor]_differential` – For a relative odometry sensor such as wheel odometry, you may be able to set this to true and then set the sensor config to constrain on the pose rather than velocity, which is often more reliable
2. `dynamic_process_noise_covariance` – Scales **Q** based on robot speed, meaning uncertainty grows slower when robot is stationary or moving slowly
3. `two_d_mode` – Whether we should assume the ground is flat and thus ignore the 7 degrees of freedom involving the z axis
4. `process_noise_covariance` – This corresponds to **Q** in the slides and quantifies how uncertain we get over time. This should be a 15 by 15 matrix, with numbers along the diagonal and zeros elsewhere.
5. `[sensor]_threshold` – This can be used to filter out noisy sensor measurements