LAB #5 ONE-STATE EXTENDED KALMAN FILTER (2-SESSIONS)

OBJECTIVE

- Learn how to implement an Extended Kalman Filter (EKF) for simple robot applications;
- Learn how to perform sensor fusion for a 1D navigation problem.

INTRODUCTION

I. Kalman Filter

Kalman filter is an optimal recursive stochastic state estimator that works with a linear dynamic system with the process and measurement noises assumed to be independent from each other, white, and Gaussian. The extended Kalman Filter (EKF) is a modification of the Kalman filter to handle nonlinear problems. It linearizes the nonlinear system equations at each time step so that the linear Kalman filter equations can be used.

Kalman filter and Extended Kalman Filter are widely used in robotics. Please check the class handout for additional information about Kalman filter and its implementation.

II. 1D Navigation

Within this lab, we will try to solve a 1D (rotational) aided inertial navigation problem, as shown in Figure 1.

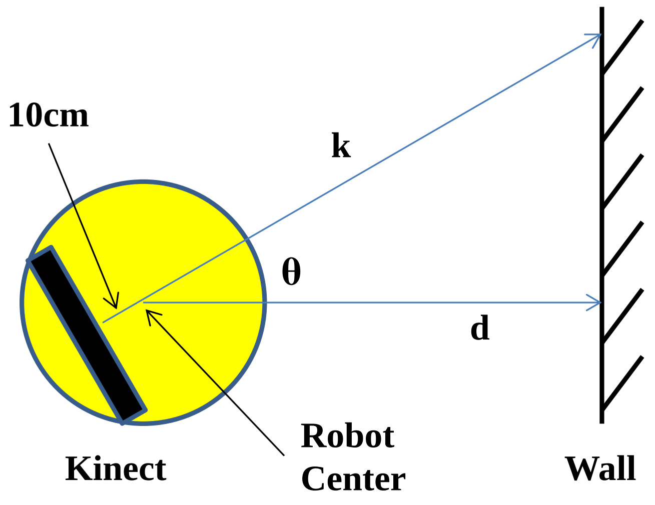

Fig.1. Kinect and Its Layout

The goal is to estimate the heading angle θ using a rate gyroscope and a Microsoft Kinect sensor. The robot is assumed to only rotate around its center without any translational movement. There are two information sources for estimating θ: the integration of yaw rate gyro (r) (dead reckoning); and the use of a Trigonometric relationship among the distance between the robot center and the wall, d; the Kinect measured distance at its center point, k; and the angle θ. Keep in mind that the Kinect is offset about 10cm from the center of the robot.

To solve this problem, you need to go through the following steps: formulate the problem in a state-space model; discretize the model; calculate the Jacobian matrices; and then come up with measurement and process noise assumptions (Q and R matrices). You can then solve the problem using the standard EKF equations.

III. Warnings

1. Do not keep the Create powered on unattended. The robot may drive itself off the table. Only turn on the robot power when you are ready for a test and turn it off right away when you are done with the test.
2. Flip the laptop screen up when you are driving the robot. Do not use the laptop as a bumper!
3. If your laptop needs an update when restarting, let it finish. Do not turn off the laptop in the middle of an update.
4. Make sure to properly shutdown the laptop after each session. Do not put the laptop in sleep! It will just drain the battery after sitting idle for a week!
5. Kinect use LASER for mapping. This is a low power laser that is projected at many directions. Microsoft claim it's harmless to our eyes but we all know how much we can trust Microsoft. My recommendation is not to stare at the laser window (when it's red) for an extended period of time. At least we are not those kids that play video games all day long!

PROCEDURE

I. Evaluate the Dead Reckoning Performance

1. Open up MATLAB code "SMART_RUN" and uncomment the portions (3 places) that are related to Kinect;
2. Save the file with a different name;
3. Modify the code so that the robot will repeatedly (for at least 10 times) turn left 30° then return to 0°. Use only the following function for the robot wheel control: SetDriveWheelsSMART(S_Create,rightWheelVel,leftWheelVel,SD.CliffLeft(i),SD.CliffRight(i),SD.CliffFrontLeft(i),SD.CliffFrontRight(i));
4. Point the robot straight at a wall 80cm away and run the program;
5. Check the iRobot Create reported angle and see if it matches your observation (i.e. if the robot is still pointing at the same angle towards the wall);
6. Modify the code to integrate the yaw rate gyro (SD.R) data to estimate the yaw angle. Note: use SD.Time_Diff(i) at each time step as the sampling time T_s;
7. Run the code again and check if the estimated yaw angle agrees with the observation;
8. Subtract the raw rate gyro bias from the measurement at each time step and run the program again and see if you can observe any improvement. Hint: the gyro bias can be estimated by taken the average value when the gyro is stationary;
9. Show the instructor your results.

II. Sensor Fusion with Extended Kalman Filter

1. Modify the code so that the Kinect will work as a laser range finder that detects the distance between itself and the object straight in front of it;
2. Determine the variance for the rate gyro and Kinect distance measurement errors;
3. Implement the EKF for θ estimation; Use the dead reckoning for predication and use the Kinect measurement for update;
4. Run the code and check if the estimated yaw angle agrees with the observation;
5. Show the instructor your results.

DELIVERABLE

An abbreviated lab report of your experiments and answer the following questions:

1. Which one of the following four methods had more estimation error: the Create encoder integration, rate gyro integration without bias removal, rate gyro integration with bias removal, and EKF based sensor fusion algorithm? Did the error grow with time for each method?
2. Show all the steps including all equations that you used during the lab for the EKF implementation.
3. What if the robot also had translational movements along its *x* axis? What would you do differently to accommodate for this change?