First laboratory exercise

Inertial measurement unit (IMU)

1. Introduction

In this laboratory exercise you will get a sense of working with the Internal Measurement Unit (IMU). IMUs tipically consist of an accelerometer, gyroscope and magnetometer. You will go through each component individually and explore their use cases and their strengths and weaknesses.

IMUs are essential components in the guidance and control of autonomous systems, but IMUs can serve as orientation sensors in many consumer products, such as mobile phones. For this laboratory exercise, first we will connect your mobile phone to the Matlab instance and enable IMU data streaming using Matlab Mobile [1].

First make sure that you are logged in with your personal account on both on the Matlab instance on your PC and your phone's Matlab Mobile application. After successfull login, on your PC write the command:

$m = mobiledev$;

Variable m now holds the handle to the mobile instance which will be used throughout this exercise.

2. Accelerometer

With the stream of acceleration measurements it is possible to estimate the velocity and position by numerical integration. The process of determining the position in this way is called dead reckoning. While dead reckoning can give the best available information on the present position with very little math, it is subject to significant errors of approximation. Even small noise in the acceleration measurement may largely impact the position estimate because of the repeated numerical integration.

Assignment 1

The first assignment with the accelerometer will be the sensor calibration using simple sensor model. Due to accelerometer inability to distinguish between acceleration and gravity, we know that when sensor is standing still it should measure the gravitational acceleration, $g = 9.80665\ m/s^2$.

Assume that the model of the sensor is

$$\boldsymbol{acc}_c = \boldsymbol{A}(\boldsymbol{acc} - \boldsymbol{b}) \tag{1}$$

Vector $\boldsymbol{acc}$ is a sensor measurement and $\boldsymbol{acc}_c$ represents the corrected measurement. $\boldsymbol{A}$ is a symmetrical 3×3 matrix and $\boldsymbol{b}$ is 3×1 vector. They constitute the unknown parameters of the

[1] https://www.mathworks.com/products/matlab-mobile.html

sensor model which can be identified using raw measurements:

$$\boldsymbol{A} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_4 & x_5 \\ x_3 & x_5 & x_6 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} x_7 & x_8 & x_9 \end{bmatrix}^T \tag{2}$$

The parameter values will be calculated using matlab *fmincon* function which nonlinearly finds the optimal (minimal) value of the given function. Error function we will optimize will be the mean squared error:

$$L = \frac{1}{N} \sum_{i=0}^{N} (||\boldsymbol{acc}_c|| - g)^2 \tag{3}$$

where N is the total number of measurement samples and the $||\cdot||$ is the Euclidean norm.

Your task is to implement the *accError.m* matlab function that calculates the error function whose minimum will yield the solution for the sensor model parameters. Inputs to the function are a vector $x = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]^T$ and the list of acceleration measurements of dimension $N \times 3$. **Attach the implemented *accError.m* function here.**

Assignment 2

For accelerometer calibration, first clear the recorded logs by using *m.discardlogs* command in matlab. Place your phone on the table and record around 30 seconds of measurements on 100 Hz sample rate. Use *loadMobileLogs.m* script to load mobile data to matlab by setting the sample time and setting the name of the data to *acc_calib_logs*.

Use *calibrateAcc.m* script to calibrate the accelerometer. The script first splits the data so that two thirds is used for calibration, and the rest is used to verify the results. The calibration data is stored as *data* variable and test data as *test* variable.

Now, implement the *getVelocityAndPosition.m* function for calculating current velocity and position from the acceleration. Function receives current acceleration measurement *acc*, current rotation matrix estimate R, velocity in previous step *prev_vel*, position in previous step *prev_pos*, and sample time T_s. Your job is to subtract the gravity vector based on the current orientation and to estimate current position and velocity. **Attach the implemented *getVelocityAndPosition.m* function here.**

Now use your script and get the velocities and positions for the *test* data. Make three plots for acceleration, velocity and position over time, and on every plot two graphs, one with pure acceleration measurements, and another with corrected measurements using calibration parameters. **Attach 3 plots here.**

Compare graphs, errors and comment on the importance of calibration.

Assignment 3

Next we will try to use accelerometer to measure a distance. First, we will keep the phone still on the one side of the table for about 10-15 seconds. This data will be used for sensor recalibration. After that, you will move the phone on the table and stop the recording on the other side of the table so that we minimize the influence of the noise. Remember to clear the mobile logs.

Now, start recording sensor measurements and do the above described steps. Use the *loadMobileLogs.m* script to save the data under *distance_measurement_logs*.

Use the *plotData.m* script to exactly find the time when the movement started. Set this time to *measureDistance.m* script under *pivotTime* variable. Make sure that the *logs* variable has the same name you used for loading the data (distance_measurement_logs). Same as in the previous assignment, use your *getVelocityAndPosition.m* function and make three plots of acceleration, velocity and position over time the same way as in Assignment 3, with and without calibration. **Attach 3 plots here.**

Comment on graphs and the usefulness of the calibration. Is the calibration model valid? Explain the results. What would you recommend for fixing the problem of determining the velocities and positions more precisely?

3. Gyroscope

Next we will take a look at the phone gyroscope. To enable gyroscope data go to *Sensors* on the mobile app and enable Angular Velocity measurement data stream.

Assignment 4

Implement the *getAnglesFromGyro.m* function to calculate the current angles based on current angular velocity measurement *ang_vel*, previous angle estimate *prev_ang* and sampling time T_s. **Attach your script here.**

Make sure to clear the mobile data logs before another data recording. For this experiment you will rotate mobile phone first around its X axis for 2 whole turns, then around Y for 2 whole turns and in the end around Z for 2 whole turns. Use *loadMobileLogs.m* and load data as *angle_measurement_logs*.

Use the *measureAnglesGyro.m* script and plot the obtained angles data versus the *true_angles* variable. **Attach 1 plot here.**

Comment on the results. Compare the gyro measurement to the accelerometer measurement in terms of the precision and noise. In your opinion, could gyroscope be used for counting the number of revolutions around a certain axis?

4. Magnetometer

Next we will take a look at the phone magnetometer. To enable magnetometer data go to *Sensors* on the mobile app and enable Magnetic Field measurement data stream.

NOTE: phones come with calibration and filtering implemented on chip, so before using phone magnetometer move it around a bit to recalibrate sensor.

Assignment 5

First, clear the recorded data from previous experiments:

m.discardlogs

Now, place the phone horizontally on the table, approximately oriented to the north, and start data stream. Record data for about 10s. Using matlab, calculate the mean μ, standard deviation σ and the magnitude F of the mean of magnetic field measurements.

$\mu =$ $\sigma =$ $F =$

Enable the Position measurement on the mobile phone and try to get at least one measurement. Go to `https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml#igrfwmm` and enter the recorded latitude, longitude and altitude to get exact values for the magnetic field using WWM model.

Enter the estimated values for North, East and Vertical components together with the Total field.

$North =$ $East =$ $Vertical =$ $Total =$

Comment on the obtained results and compare them to the measurements.

Assignment 6

For the next assigment, first clear the mobile sensor data logs. Now, you will place a magnet in the proximity of the mobile phone so that it is fixed with respect to it. Collect the measurement data in many different orientations of the phone but make sure that the magnet is fixed.

Use the *magnetometer3d.m* script to calibrate the phone magnetometer to the outside magnet. Back up your answer with the plot of the calibration result. **Attach 2 plots here.**

What is the name of this kind of magnetometer calibration? What does it mean?

Assignment 7

Final assignment will be the use of an accelerometer and magnetometer for orientation estimation. Implement the *getRollPichFromAcceleration.m* and *getHeadingFromMag.m* functions for calculating the roll, pitch and yaw angles.

Run the *orientationEstimate.m* function and test the behaviour. Rotate the mobile phone to show that orientation estimate is working well. **Attach the plot here.**

Why do we need both accelerometer and magnetometer to get the orientation? As you can see, mobile phones may provide you the raw measurements of the orientation. How do you think it is calculated? Compare to your results.