Module Five

Henrik Frank, hefra13@student.sdu.dk

Christian Arentsen, chare13@student.sdu.dk

Vasileios Karvouniaris, vakar15@student.sdu.dk

Asbjørn Schou Müller, asmul10@student.sdu.dk

I. Attitude sensing using accelerometers

The purpose of this exercise is to learn how to calculate the aircraft orientation based on these linear accelerations. The exercise is based on data sampled from a SparkFun Razor Inertial Measurement Unit (IMU) (figure 5).

I. Calculate pitch angle

The file `imu_razor_data_pitch_45deg.txt` contains a dataset that was sampled while the IMU was tilted forwards and back

Use the Python script `imu_exercise.py` to perform a calculation of the pitch angle based on the accelerometer values. Use the equation 28 in the document *Tilt Sensing Using a Three-Axis Accelerometer.pdf*

Observe that the plot shows the expected output based on the description of the dataset.

Results Looking at the plot, figure 1, we see the pitch angle as a function of time, and a pitch angle of roughly up to 55 degrees pitch.

II. Calculate roll angle

The file `imu_razor_data_roll_45deg.txt` contains a dataset that was sampled while the IMU was tilted to the side.

Use the Python script `imu_exercise.py` to perform a calculation of the roll angle based on the accelerometer values. Use the equation 29 in the document *Tilt Sensing Using a Three-Axis Accelerometer.pdf*

Observe that the plot shows the expected output based on the description of the dataset.

Results Looking at the plot, figure 2, we see the roll angle as a function of time, which is consistent with shifting sides, rolling to left and right afterwards.

III. Accelerometer noise

The file `imu_razor_data_static.txt` contains a dataset that was sampled for approx 60 seconds while the IMU was held static on a reasonably level surface.

Using the same calculations as in 1. and 2. plot the pitch and roll angles based on this dataset.

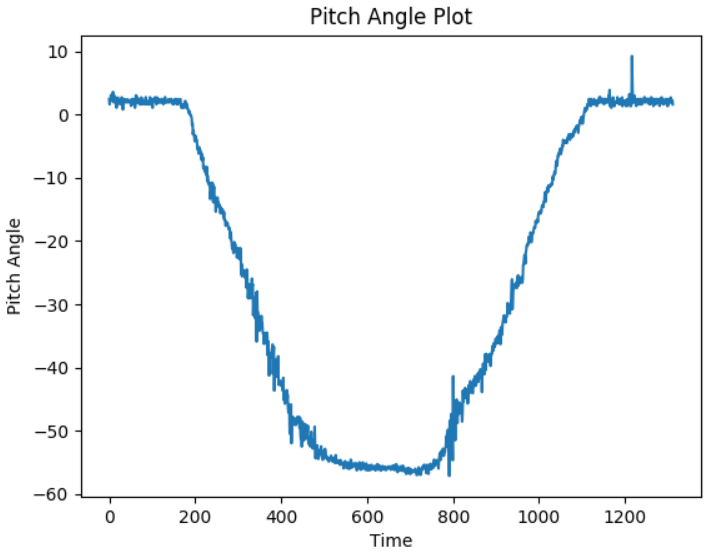

Figure 1: Figure of a pitch up to roroughly 55 degrees

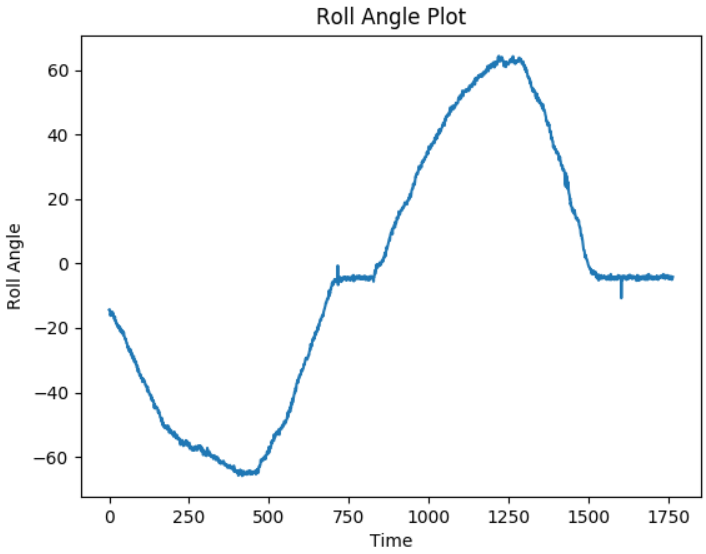

Figure 2: Plot of rolling from side to side

Do the calculated angles show any significant noise or bias? Yes, in figure 1 shows a jump around 800th millisecond, and same goes for figure 2 at around 700 and again at 1600. The static plots in fig 3 shows a general bias of $+/-$ 2 degrees.

How could this be mitigated? Using filtering, such as a Kalman filter.

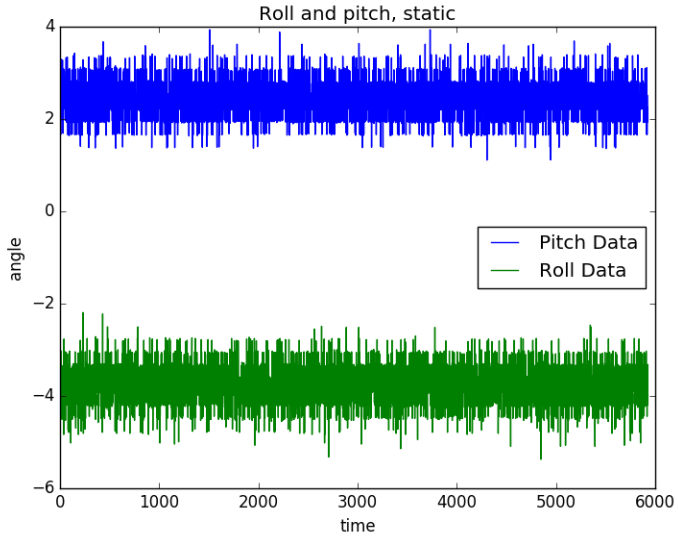

Figure 3: Plot both roll and pitch in static position

IV. Low-pass filtering

Try adding a low-pass filter and see if you can reduce the noise. How much delay do you then add to the estimation of the pitch and roll angle? Is this acceptable given the update velocity of the stability controller?

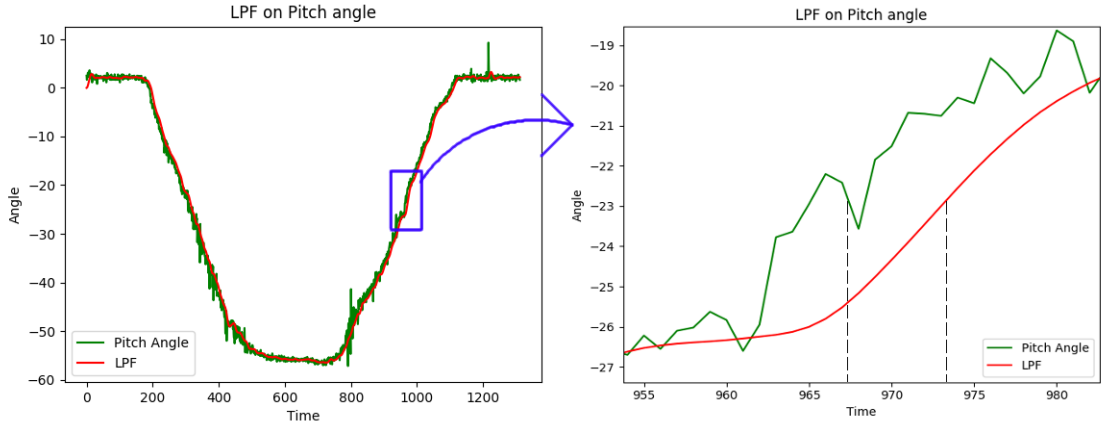

Figure 4: LPF on Pitch angle data

Results As seen in figure 4, implementing a low pass filter can smoothen a graph, yet, if one looks closer to the filtered graph, the right hand side of figure 4, on can clearly see that that using the low pass filter, the information returned by the filter is delayed but only a little, however, this has huge impact on flight controlling.

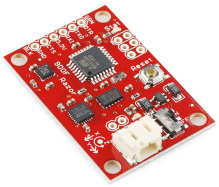

Figure 5: *SparkFun Razor IMU.*

Figure 6: *VectorNav VN-100 IMU.*

V. Limitations of Euler angles

The equations 28 and 29 used above are not able to describe all states of orientation, this is well known as *Gimbal lock*. Which particular orientations may cause problems?

VI. Extra: Quaternions

Implement a pitch and roll estimation ot suffering from the gimbal lock limitations using Quaternions.

II. Gyro measurements

The purpose of this exercise is to learn how to integrate the angular velocities measured by a gyro to obtain a relative measure of the angle about that axis.

The exercise is based on data sampled from a SparkFun Razor Inertial Measurement Unit (IMU) (figure 5) and a VectorNav VN-100 IMU (figure 6).

I. Calculating relative angle

The file `imu_razor_data_yaw_90deg.txt` contains a dataset that was sampled while the IMU was turned 90° clockwise, then the IMU was turned 90° counter-clockwise.

Use the Python script `imu_exercise.py` to perform a numerical integration of the angular velocity about the z-axis ω_z to obtain a measure of the relative angle. Use the actual time since the previous received angular velocity for the integration.

Observe that the plot shows the expected output based on the description of the dataset.

Results The graph in figure 7 shows how the IMU moves to the right for approximately 90 degrees, and then turns back to original position.

II. Static data

The file `imu_razor_data_static.txt` contains a dataset that was sampled for approx 60 seconds while the IMU was held static on a reasonably level surface.

Perform the same numerical integration of the angular velocity and observe the result.

Results The plot in figure 8 shows how the gyro, though in a static position, has a little positive drift value, resulting in a gain of around 2.5 degrees in just 6 minutes. Should we rely on gyro alone, this is a rather high bias.

Figure 7: Plot showing IMU turning right 90 degrees, and back to original position

Figure 8: Plot showing IMU in static position

III. Observing bias

The output from the integration shows that the relative angle is drifting. The drift is the visible effect of a bias on the angular velocity.

Try to estimate the bias and subtract it before integrating.

Results The drifting bias is around 0.5 degree a minute. More accurately 0.0008 per step.

IV. Bias sources

Consider the potential sources of noise and bias for a gyro?

Results Bias error tends to vary, both with temperature and over time. The bias error of a gyro is due to a number of components:

- calibration errors
- switch-on to switch-on
- bias drift
- efects of shock (g level)[1].

For reducing drift bias, one uses a warm-up period.

V. Extra: Integration using average time

Perform a similar numerical integration but this time use a calculated average time between updates in the file.

Consider why there is a difference and what gives the most accurate result?

VI. Extra: Record more datasets

Use the Python script `nmea_data_logger.py` and an IMU to record other datasets.

III. Kalman filter

The purpose of this exercise is to learn how a Kalman filter will improve the attitude estimation using the accelerometers and gyros as input.

I. Implementing a scalar Kalman filter

Implement a scalar Kalman filter estimating the Pitch angle in the Python script `imu_exercise_kalman.py`. To do this you must add the calculation of pitch and roll from the accelerometers and then the gyro relative angles from the previous exercises.

Please notice that within the script two sections are clearly marked *Insert initialize code below* and *Insert loop code below*. You do not need to change anything in the file outside these sections.

Results We have had some issues with implementing the kalman filter, see progress at GitHub.

References

[1] Gyroscope. http://sensorwiki.org/doku.php/sensors/gyroscope, 2016. [Online; accessed 17-March-2017].