Course notes, module 1 week 36

UAV attitude estimation

Kjeld Jensen kjen@mmmi.sdu.dk

1 Agenda

1. Welcome
2. Corona precautions
3. Practical information
4. Student expectations https://uaswork.org/sdu/idt/expect.
5. Introduction to the course
6. Defining student groups https://uaswork.org/sdu/idt/groups.
7. Introduction to the module theory and exercises
8. Exercises

2 Preparation for next module

No preparation is needed for the next module.

3 UAV attitude exercises

The purpose of the exercises on UAV attitude is for you to understand how the output from accelerometers and gyros (and magnetometers) are used to determine the attitude.

The three accelerometers contained in an Inertial Measurement Unit (IMU) are used for estimating the absolute pitch and roll (figure 1) orientation of the aircraft with respect to the Gravity vector. The three gyro's in the IMU are used for measuring angular velocities about the pitch, roll and yaw axis of the aircraft. The three magnetometers in the IMU are used for determining the compass course of the aircraft.

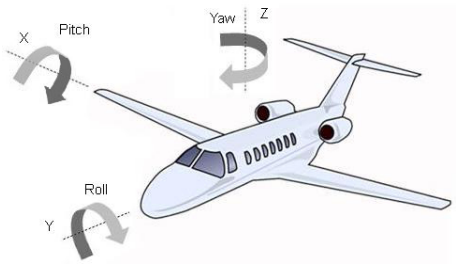

Figure 1: *Pitch, roll og yaw axis of an aircraft.*

3.1 UAV attitude sensors

Please describe briefly how a Microelectromechanical systems (MEMS) accelerometer works.

Please describe briefly how a Microelectromechanical systems (MEMS) gyro works.

Please describe briefly how to determine the compass course using magnetometers - and maybe other sensors?

3.2 UAV attitude sensing using accelerometers

The purpose of this exercise is to learn how to calculate the aircraft orientation based on the linear accelerations. You may wish to consult the reference material provided for this module. The exercise is based on data sampled from a SparkFun Razor Inertial Measurement Unit (IMU) (figure 2).

3.2.1 Calculate pitch angle

The file `imu_razor_data_pitch_55deg.txt` contains a dataset that was sampled while the IMU was tilted approximately 45 degrees forwards and back.

Use the Python[1] 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.

3.2.2 Calculate roll angle

The file `imu_razor_data_roll_65deg.txt` contains a dataset that was sampled while the IMU was tilted first approximately 65 degrees to one side, then approximately 65 degrees to the other 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.

3.2.3 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.

Do the calculated angles show any significant noise or bias?

How could this be mitigated?

3.2.4 Low-pass filtering

Use the file `imu_razor_data_pitch_55deg.txt` or the file `imu_razor_data_roll_65deg.txt` for this exercise, as low-pass filtering does not apply well to the static data.

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 an update rate of 100 Hz of the UAV attitude controller?

[1] Python scripts in this exercise have been tested to work with Ubuntu 20.04 and Python3

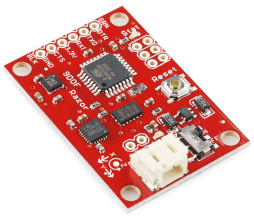

Figure 2: *SparkFun Razor IMU.*

3.2.5 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? How can this problem be mitigated so that all possible states of orientation can be described?

3.3 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 2).

3.3.1 Calculating relative angle

The file `imu_razor_data_yaw_90deg.txt` contains a dataset that was sampled while the IMU was turned $90\,^\circ$ clockwise, then the IMU was turned $90\,^\circ$ 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.

3.3.2 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.

3.3.3 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.

3.3.4 Bias sources

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

3.4 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.

3.4.1 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. On Ubuntu you may need to install the python3-opengl package for the 3D simulation to work.