Hochschule Esslingen
University of Applied Sciences

Team Project

Quadrocopter

in the degree course ASM-SB
of the Faculty Graduate School
ASM2

Oliver Breuning
Martin Brodbeck
Jürgen Schmidt
Phillip Woditsch

Periode: Summersemester 2015
Professor: Prof. Dr. Jörg Friedrich

Contents

1 Sensor fusion controlling 1

2 Angle calculation 6
2.1 Calculation of the orientation angles 6
2.2 Magnetic sensor test 8
2.3 Improvement magnetic sensor 9

3 Complementary-Filter 11

4 Kalman-Filter 14

Bibliography 19

List of Figures

1.1 Matlab model 1

2.1 First result Kalman filter 7
2.2 Weak magnetic field strength 8
2.3 Strong magnetic field strength 9
2.4 New positioning of IMU 1 9
2.5 New positioning of IMU 2 10

3.1 First result complementary filter 12
3.2 Final result complementary filter 12
3.3 3D representation of the integrated gyroscope raw values on the left and the fusion filtered on the right 13

4.1 First result Kalman filter 15
4.2 Analyzing the acceleration sensor 16
4.3 Analyzing the gyroscope sensor 16
4.4 Analyzing the magnetic sensor 17
4.5 Final result Kalman filter 18
4.6 3D representation of the integrated gyroscope raw values on the left and the fusion filtered on the right 18

1

1 Sensor fusion controlling

To achieve the statistical data of all sensors of the inertial measurement unit a Matlab model is used. This model also helps to check the results of the two different fusion filters, the Kalman filter and the complementary filter. The data is send via UDP-packets from the Raspberry Pi to the Matlab model on the Host computer. Figure 1.1 shows the used Matlab model. Additional a 3D representation of the rotations is visualized and can be directly compared with the integrated offset corrected gyroscope data.

Figure 1.1: Matlab model

With the help of the scopes within this model, the logged data is stored in the Matlab workspace and then it is analyzed with a Matlab file. With the help of this file, a histogram from every sensor is generated. Also the meanvalue and the variance from those sensor is calculated. Finally a plot from the complementary filter and Kalman filter is generated. Also the needed minmum and maximum values of the magnetic sensor is stored. This needs to be done to reduce the influences of hard magnetic parts to the magnetic sensor. Those stored minmium and maximum values are then written in a header file (OrientationDefines.h) which needs to be copied to the folder of the Orientation.c and Orientation.h files. To achieve the best performance, the description of chapter 2 needs to be followed.

```
headerFileName = 'OrientationDefines.h';

%Acceleration sensor
figure(1)
%plot histogram of ax and calculate meanvalue and variance
subplot(3,1,1)
hist(ax.signals.values(:),50);
    M=mean(ax.signals.values(:));
    V=var(ax.signals.values(:));
title('Acceleration sensor')
disp(['Meanvalue: ' num2str(M)])
xlabel({'Raw data of sensor ax' ['Mean: ' num2str(M)] ['Varianz: ' num2str(V)]}) % x-axis label

%plot histogram of ay and calculate meanvalue and variance
subplot(3,1,2)
hist(ay.signals.values(:),50);
    M=mean(ay.signals.values(:));
    V=var(ay.signals.values(:));
xlabel({'Raw data of sensor ay' ['Mean: ' num2str(M)] ['Varianz: ' num2str(V)]}) % x-axis label
ylabel('Count of appearance') % y-axis label

%plot histogram of az and calculate meanvalue and variance
subplot(3,1,3)
hist(az.signals.values(:),50);
    M=mean(az.signals.values(:));
    V=var(az.signals.values(:));
xlabel({'Raw data of sensor az' ['Mean: ' num2str(M)] ['Varianz: ' num2str(V)]}) % x-axis label

%Magnetic sensor
figure(2)
%plot histogram of mx and calculate meanvalue and variance
subplot(3,1,1)
hist(mx.signals.values(:),50);
    M=mean(mx.signals.values(:));
```

```
    V=var(mx.signals.values(:));
title('Magnetic sensor')
disp(['Meanvalue: ' num2str(M)])
xlabel({'Raw data of sensor mx' ['Mean: ' num2str(M)] ['Varianz: ' num2str(V)]}) % x-axis label

%plot histogram of my and calculate meanvalue and variance
subplot(3,1,2)
hist(my.signals.values(:),50);
    M=mean(my.signals.values(:));
    V=var(my.signals.values(:));
xlabel({'Raw data of sensor my' ['Meanvalue: ' num2str(M)] ['Varianz: ' num2str(V)]}) % x-axis label
ylabel('Count of appearance') % y-axis label

%plot histogram of mz and calculate meanvalue and variance
subplot(3,1,3)
hist(mz.signals.values(:),50);
    M=mean(mz.signals.values(:));
    V=var(mz.signals.values(:));
xlabel({'Raw data of sensor mz' ['Meanvalue: ' num2str(M)] ['Variance: ' num2str(V)]}) % x-axis label

%Gyroscope sensor
figure(3)
%plot histogram of pitch and calculate meanvalue and variance
subplot(3,1,1)
hist(pitch.signals.values(:),50);
    M=mean(pitch.signals.values(:));
    V=var(pitch.signals.values(:));
title('Gyroscope sensor')
disp(['Meanvalue: ' num2str(M)])
xlabel({'Raw data of sensor pitch rate' ['Mean: ' num2str(M)] ['Varianz: ' num2str(V)]}) % x-axis label

%plot histogram of roll and calculate meanvalue and variance
subplot(3,1,2)
hist(roll.signals.values(:),50);
    M=mean(roll.signals.values(:));
    V=var(roll.signals.values(:));
xlabel({'Raw data of sensor roll rate' ['Meanvalue: ' num2str(M)] ['Varianz: ' num2str(V)]}) % x-axis label
ylabel('Count of appearance') % y-axis label

%plot histogram of yaw and calculate meanvalue and variance
subplot(3,1,3)
```

```
hist(yaw.signals.values(:),50);
    M=mean(yaw.signals.values(:));
    V=var(yaw.signals.values(:));
xlabel({'Raw data of sensor yaw rate' ['Meanvalue: ' num2str(M)]
   ['Variance: ' num2str(V)]}) % x-axis label


%Complementary filter
figure(4)
subplot(3,1,1)
plot(Comp_pitch.time,Comp_pitch.signals.values(:));
title('Complementary filter')
xlabel('time [s]') % x-axis label
ylabel('pitch [degree]') % y-axis label

subplot(3,1,2)
plot(Comp_roll.time,Comp_roll.signals.values(:));
xlabel('time [s]') % x-axis label
ylabel('roll [degree]') % y-axis label

subplot(3,1,3)
plot(Comp_yaw.time,Comp_yaw.signals.values(:));
xlabel('time [s]') % x-axis label
ylabel('yaw [degree]') % y-axis label


%Kalman filter
figure(5)
subplot(3,1,1)
plot(Kalman_pitch.time,Kalman_pitch.signals.values(:));
title('Kalman filter')
xlabel('time [s]') % x-axis label
ylabel('pitch [degree]') % y-axis label

subplot(3,1,2)
plot(Kalman_roll.time,Kalman_roll.signals.values(:));
xlabel('time [s]') % x-axis label
ylabel('roll [degree]') % y-axis label

subplot(3,1,3)
plot(Kalman_yaw.time,Kalman_yaw.signals.values(:));
xlabel('time [s]') % x-axis label
ylabel('yaw [degree]') % y-axis label

%min max of magnetic sensor
maxz=max(mz.signals.values)*1000000;
```

```
minz=min(mz.signals.values)*1000000;
maxy=max(my.signals.values)*1000000;
miny=min(my.signals.values)*1000000;
maxx=max(mx.signals.values)*1000000;
minx=min(mx.signals.values)*1000000;

fprintf('\n');
disp('Calibration Data:');
disp('=================');
disp(['Min_X: ' num2str(minx)])
disp(['Max_X: ' num2str(maxx)])
disp(['Min_Y: ' num2str(miny)])
disp(['Max_Y: ' num2str(maxy)])
disp(['Min_Z: ' num2str(minz)])
disp(['Max_Z: ' num2str(maxz)])
fprintf('\n');

disp('==================================');
disp('Generating Calibration-Header-File');
disp('==================================');

disp(['Writing header ', headerFileName, ' ... ' ]);
fileID = fopen(headerFileName,'w');
fprintf(fileID,'/*!\n');
fprintf(fileID,' * \\file OrientationDefines.h\n');
fprintf(fileID,' */\n');
fprintf(fileID,'\n');
fprintf(fileID,'#ifndef SIG_ORIENTATION_ORIENTATIONDEFINES_H_\n');
fprintf(fileID,'#define SIG_ORIENTATION_ORIENTATIONDEFINES_H_\n');
fprintf(fileID,'\n');
fprintf(fileID,'//Defines for Acceleration Magnetic angle calculation (automatic calibration via MATLAB)\n');
fprintf(fileID,'#define M_SIGORI_MAG_MINX_F64 %.9f\n',minx);
fprintf(fileID,'#define M_SIGORI_MAG_MAXX_F64 %.9f\n',maxx);
fprintf(fileID,'#define M_SIGORI_MAG_MINY_F64 %.9f\n',miny);
fprintf(fileID,'#define M_SIGORI_MAG_MAXY_F64 %.9f\n',maxy);
fprintf(fileID,'#define M_SIGORI_MAG_MINZ_F64 %.9f\n',minz);
fprintf(fileID,'#define M_SIGORI_MAG_MAXZ_F64 %.9f\n',maxz);
fprintf(fileID,'\n');
fprintf(fileID,'#endif /* SIG_ORIENTATION_ORIENTATIONDEFINES_H_ */\n');
fclose(fileID);
disp('Done');
fprintf('\n');
disp('Calibration successfully processed.');
```

2 Angle calculation

2.1 Calculation of the orientation angles

The Kalman filter and the complementary filter need both angles as input. Those angles are calculated with the acceleration sensor and with the gyroscope. The gyroscope sensor is used as main input and is corrected by the acceleration/magnetic sensor. The reason why both are used is already described in the Sensor fusion chapter.
To achieve an angle from the gyroscope, the values from the sensor needs to be integrated. The following formulas show the integration part of the function of the gyroscope which runs on the Raspberry Pi. It is located in the folder 'sig/Orientientation/' in the file Orientation.c in the function 'm_sigOri_calcGyroAnglePerStep_st()'. This function calculates the angle which since the last call is made.

$$roll_{angle} = roll_{rate} * deltaT \tag{2.1}$$

$$pitch_{angle} = pitch_{rate} * deltaT \tag{2.2}$$

$$yaw_{angle} = yaw_{rate} * deltaT \tag{2.3}$$

To make this code independent from any timestep, it calculates locally the time difference 'deltaT' between the last call. To make this possible the 'gettimeofday' function is used. Because of that any jitter will not make a problem. Also the gyroscope sensor gets automatically offset corrected when the code is started.

To get an angle from the acceleration/magnetic sensor the following calculation is needed. It is located in the function 'm_sigOri_calcAccMagAngle_st()' in the same file like mentioned before.
First the roll angle is calculated:

$$roll_{angle_rad} = atan2\left(\frac{acc_{y_axis}}{acc_{z_axis}}\right) \tag{2.4}$$

$$roll_{angle_deg} = -roll_{angle_rad} * 180/Pi \tag{2.5}$$

Then the pitch is calculated:

$$pitch_{angle_rad} = atan\left(-\frac{acc_{x_axis}}{acc_{y_axis} * sin(roll_{angle_rad}) + acc_{z_axis} * cos(roll_{angle_rad})}\right) \tag{2.6}$$

$$pitch_{angle_deg} = -pitch_{angle_rad} * 180/Pi \tag{2.7}$$

Last step is the calculation of the yaw angle. The yaw angle can not be calculated by the acceleration sensor, only the magnetic sensor can be used for this. But the magnetic sensor can not directly be used. Additional a magnetic tilt compensation needs to be done. The following formulas show the calculation steps.

$$divider = mag_{x_axis} * cos(pitch_{angle_rad}) + \tag{2.8}$$

$$mag_{y_axis} * sin(pitch_{angle_rad}) * sin(roll_{angle_rad}) + \tag{2.9}$$

$$mag_{z_axis} * sin(pitch_{angle_rad}) * cos(roll_{angle_rad}) \tag{2.10}$$

$$yaw_{angle_rad} = atan2\left(\frac{mag_{z_axis} * sin(roll_{angle_rad}) - mag_{y_axis} * cos(roll_{angle_rad})}{divider}\right) \tag{2.11}$$

$$yaw_{angle_deg} = yaw_{angle_rad} * 180/Pi \tag{2.12}$$

With this, the implementation of the fusion filter were done and proved. First the results seem to be correct like can be seen in figure 2.1.

Figure 2.1: First result Kalman filter

2.2 Magnetic sensor test

As can be seen there is a problem with the yaw angle when a pitch angle is applied. This is no solution which can be used in the quadrocopter. So additional tests are made. After some time the problem was detected. Because just the yaw angle makes some problem, the main observation layed on the magnetic sensor. Figure 2.2 shows the measured magnetic field.

Figure 2.2: Weak magnetic field strength

The problem can directly be seen. The measured field strength should be nearly the same on all axis. The field strength on the x-axis and y-axis is the same. The z-axis delivers only just one tenth of the normal value. Because of that the noise on the sensor is in the same height like the signal and therefor aquiring not possible. First the position of the mounted IMU is disputed. After changing the following sensor values of the magnetic sensor can be achieved.

Figure 2.3: Strong magnetic field strength

By comparing the measurement of the initial positioning of the IMU with the new one, the strength on the z-axis is significantly higher. Also the strength on the three axis are nearly the same. So for further usage the new position is used.

2.3 Improvement magnetic sensor

The figures 2.4 and 2.5 shows the comparison of the mounting.

Figure 2.4: New positioning of IMU 1

Figure 2.5: New positioning of IMU 2

The left PCB shows the old position and the right the new position. As can be seen just the height of the mounted IMU has to be changed. Due to the tests a distance of approximately 1cm seems to be enough. So the wiring can be kept as it is.
Additionally to reduce the really high influences of near metallic parts, a hard magnetic offset compensation needs to be done. Also a scaling is done. The following calculations which run in the function 'm_sigOri_calcAccMagAngle_st()' are done within every step.

$$mag_{x_axis} = (mag_{x_axis} - logged_{min_x})/(logged_{max_x} - logged_{min_x}) * 2 - 1 \quad (2.13)$$
$$mag_{y_axis} = (mag_{y_axis} - logged_{min_y})/(logged_{min_y} - logged_{min_y}) * 2 - 1 \quad (2.14)$$
$$mag_{z_axis} = (mag_{z_axis} - logged_{min_z})/(logged_{min_z} - logged_{min_z}) * 2 - 1 \quad (2.15)$$

To achieve the full range of the all three sensors the scope of Matlab can be used. The m-file which is mentioned before creates directly the header-file for the code. The improved results due to the changes which are made here can be seen in the second output of the Kalman filter and complementary filter in chapter 3 and 4.

3 Complementary-Filter

As written in the chapter about the sensor fusion for the complementary filter a filter time needs to be chosen. This constant is used for the lowpass filter for the acceleration/magnetic sensor and for the highpass filter of the gyroscope.

The complementary filter uses the following calculation:

$$angle = alpha * (angle + integrated_Gyro) + (1 - alpha) * Acc_Mag_angle \tag{3.1}$$

As an initial guess for the first try the complementary filter uses the filter constant of 0.995. The sampling period of the sensors is 800 Hz. Usage of the filter constant of 0.995 and the sampling period of 800 Hz leads to a cut off frequency of 4 Hz.

$$alpha = \frac{time_constant}{time_constant + sample_period} \tag{3.2}$$

$$alpha = 0.995 \tag{3.3}$$

$$sample_period = \frac{1}{800Hz} = 0.00125sec \tag{3.4}$$

$$\tag{3.5}$$

This leads to:

$$time_constant = 0.2488sec \approx \frac{1}{4Hz} \tag{3.6}$$

With this constant the following result was achieved. To make a test, first the yaw angle is changed in the range of ±90°, after that a roll angle is changed in the range of ±90° and finally the pitch angle is changed in the range of ±90°.

Figure 3.1: First result complementary filter

First what catches somebodys eye is that the filter is to slow. So the steady state value is achieved after consuming to much time. Also the yaw angle is not equally distributed over the whole 360 degree. In one direction the angle changes just around 70 degree. When changing roll the influences in yaw is just because the rotation was not straight in roll direction. Changing the pitch angle to much lead to a change in yaw. The next step is to improve the yaw angle, so that it will reach also the +90 degree. Also the yaw change during pitch will be observed and compared to the initial measurement. Also the time constant will be changed that a faster response can be seen.

Figure 3.2: Final result complementary filter

The comparison of the first result and the final result shows the increased filter speed. The signal needs not so much time to reach the final value. Also the yaw angle reaches the 90 degree when turning 90 degree. The extreme influences on yaw directly after changing of the rotation angle results from a hand made rotation. The influences on yaw while an other angle is applied is extremely reduced.

The last figure shows how off an angle can be when just an gyroscope is used. On the left side the integrated gyroscope can be seen. On the right side, the 3D representation of the fusioned sensors are displayed.

Figure 3.3: 3D representation of the integrated gyroscope raw values on the left and the fusion filtered on the right

4 Kalman-Filter

For the Kalman filter the noise of the measurement and the process has to be set up. This noise values are the variances of the used sensors and of the process. In the calculation process are those matrices named $\underline{Q}$ and $\underline{R}$. The matrix $\underline{Q}$ uses the variances of the process noise and the matrix $\underline{R}$ uses the variances of the noise of the sensors.

<u>Prediction step:</u>

Predict the next state:

$$\vec{x}_k = \underline{A}\vec{x}_{k-1} + \underline{B}\vec{u}_{k-1} \tag{4.1}$$

Predict the covariance for the next step:

$$\underline{P}_k = \underline{A}\underline{P}_{k-1}\underline{A}^T + \underline{Q} \tag{4.2}$$

<u>Correction step:</u>

Computation of the Kalman gain:

$$\underline{K}_k = \underline{P}_k\underline{H}^T(\underline{H}\underline{P}_k\underline{H}^T + \underline{R})^{-1} \tag{4.3}$$

Updating state prediction with new measurement:

$$\vec{x}_k = \vec{x}_k + \underline{K}_k(\vec{z}_k - \underline{H}\vec{x}_k) \tag{4.4}$$

Updating the error covariance:

$$\underline{P}_k = (\underline{I} - \underline{K}_k\underline{H})\underline{P}_k \tag{4.5}$$

As an initial guess the matrices are set to:

$$\underline{Q} = \begin{pmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0 \\ 0 & 0 & 0.0001 \end{pmatrix} \tag{4.6}$$

$$\underline{R} = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.01 \end{pmatrix} \tag{4.7}$$

With those matrices the following result was achieved. To make a test, first the yaw angle is changed in the range of ±90°, after that a roll angle is changed in the range of ±90° and finally the pitch angle is changed in the range of ±90°.

Figure 4.1: First result Kalman filter

The filter responses in a adequate time. Yaw angle change is like with the complementary filter from -90 degree to just 70 degree. The roll angle changes sufficient. But when changing the pitch angle to high a extreme yaw angle change can be seen. So the next step is to improve the yaw angle, so that it will reach also the +90 degree. Also the yaw change during pitch will be observed. After measuring the meanvalue and variances of all sensors they can be used to improve the Kalman filter. The figures 4.2, 4.3 and 4.4 show the logged data.

Figure 4.2: Analyzing the acceleration sensor

Figure 4.3: Analyzing the gyroscope sensor

Figure 4.4: Analyzing the magnetic sensor

First the matrix $\underline{Q}$, the variances of the process noise and the matrix $\underline{R}$, the variances of the measured sensor noises are changed by using the measured values. The new values are:

$$\underline{Q} = \begin{pmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0 \\ 0 & 0 & 0.005 \end{pmatrix} \tag{4.8}$$

$$\underline{R} = \begin{pmatrix} 0.06 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.07 \end{pmatrix} \tag{4.9}$$

Figure 4.5: Final result Kalman filter

The wanted improving of the yaw angle in reaching the 90 degree when turning 90 degree is succesfully reached. The extreme influences on yaw directly after changing of the rotation angle results from a hand made rotation. The influences on yaw while an other angle is applied is extremely reduced.

The last figure shows how off an angle can be when just an gyroscope is used. On the left side the integrated gyroscope can be seen. On the right side, the 3D representation of the fusioned sensors are displayed.

Figure 4.6: 3D representation of the integrated gyroscope raw values on the left and the fusion filtered on the right

Bibliography

[BorEng] Alex, The quadcopter : control the orientation, 2012, http://theboredengineers.com/2012/05/the-quadcopter-basics/

[STM] Jay Esfandyari, Roberto De Nuccio, Gang Xu; STMicroelectronics, http://uk.mouser.com/applications/sensor_solutions_mems/