Reading a IMU Without Kalman: The Complementary Filter

Submitted by Pieter-Jan on Fri, 26/04/2013 - 08:38

These days, IMU's (Intertial Measurement Units) are used everywhere. They are e.g. the sensors that are responsible for keeping track of the oriëntation of your mobile phone. This can be very useful for automatic screen tilting etc. The reason I am interested in this sensor is because I want to use it to stabilize my quadrocopter. If you are here for another reason, this is not a problem as this tutorial will apply for everyone.

When looking for the best way to make use of a IMU-sensor, thus **combine the accelerometer and gyroscope data**, a lot of people get fooled into using the very powerful but complex Kalman filter. However the Kalman filter is great, there are 2 big problems with it that make it hard to use:

- Very complex to understand.
- Very hard, if not impossible, to implement on certain hardware (8-bit microcontroller etc.)

In this tutorial I will present a solution for both of these problems with another type of filter: the complementary filter. It's extremely easy to understand, and even easier to implement.

Why do I need a filter?

Most IMU's have 6 DOF (Degrees Of Freedom). This means that there are 3 accelerometers, and 3 gyrosocopes inside the unit. If you remember anything from a robotics class you might have taken, you might be fooled into thinking that the IMU will be able to measure the precise position and orientation of the object it is attached to. This because they have told you that an object in free space has 6DOF. So if we can measure them all, we know everything, right? Well ... not really. The sensor data is not good enough to be used in this way.

We will use both the accelerometer and gyroscope data for the same purpose: obtaining the angular position of the object. The gyroscope can do this by integrating the angular velocity over time, as was explained in a previous article. To obtain the angular position with the accelerometer, we are going to determine the position of the gravity vector (g-force) which is always visible on the accelerometer. This can easily be done by using an atan2 function. In both these cases, there is a big problem, which makes the data very hard to use without filter.

The problem with accelerometers

As an accelerometer measures all forces that are working on the object, it will also see a lot more than just the gravity vector. Every small force working on the object will disturb our measurement completely. If we are working on an actuated system (like the quadrocopter), then the forces that drive the system will be visible on the sensor as well. The accelerometer data is reliable only on the long term, so a "low pass" filter has to be used.

The problem with gyroscopes

In one of the previous articles I explained how to obtain the angular position by use of a gyroscope. We saw that it was very easy to obtain an accurate measurement that was not susceptible to external forces. The less good news was that, because of the integration over time, the measurement has the tendency to drift, not returning to zero when the system went back to its original position. The gyroscope data is reliable only on the short term, as it starts to drift on the long term.

The complementary filter

The complementary filter gives us a "best of both worlds" kind of deal. On the short term, we use the data from the gyroscope, because it is very precise and not susceptible to external forces. On the long term, we use the data from the accelerometer, as it does not drift. In it's most simple form, the filter looks as follows:

$$angle = 0.98 * (angle + gyrData * dt) + 0.02 * (accData)$$

The gyroscope data is integrated every timestep with the current angle value. After this it is combined with the low-pass data from the accelerometer (already processed with atan2). The constants (0.98 and 0.02) have to add up to 1 but can of course be changed to tune the filter properly.

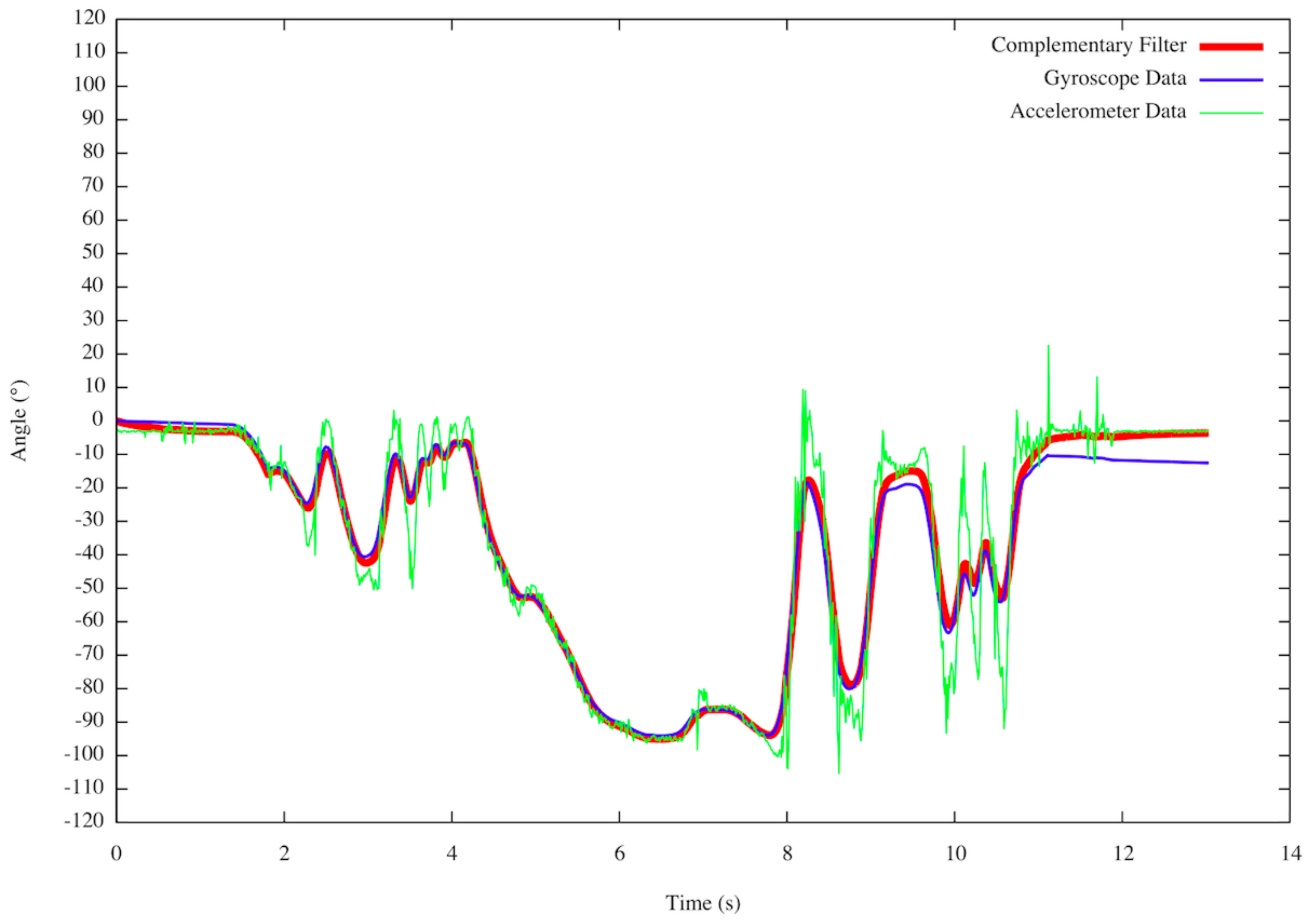
I implemented this filter on a Raspberry Pi using a MPU6050 IMU. I will not discuss how to read data from the MPU6050 in this article (contact me if you want the source code). The implementation of the filter is shown in the code snippet below. As you can see it is very easy in comparison to Kalman.

The function "ComplementaryFilter" has to be used in a infinite loop. Every iteration the pitch and roll angle values are updated with the new gyroscope values by means of integration over time. The filter then checks if the magnitude of the force seen by the accelerometer has a reasonable value that could be the real g-force vector. If the value is too small or too big, we know for sure that it is a disturbance we don't need to take into account. Afterwards, it will update the pitch and roll angles with the accelerometer data by taking 98% of the current value, and adding 2% of the angle calculated by the accelerometer. This will ensure that the measurement won't drift, but that it will be very accurate on the short term.

It should be noted that this code snippet is only an example, and should not be copy pasted as it will probably not work like that without using the exact same settings as me.

```c
#define ACCELEROMETER_SENSITIVITY 8192.0
#define GYROSCOPE_SENSITIVITY 65.536

#define M_PI 3.14159265359

#define dt 0.01                                                 // 10 ms sample rate!

void ComplementaryFilter(short accData[3], short gyrData[3], float *pitch, float *roll)
{
    float pitchAcc, rollAcc;

    // Integrate the gyroscope data -> int(angularSpeed) = angle
    *pitch += ((float)gyrData[0] / GYROSCOPE_SENSITIVITY) * dt; // Angle around the X-axis
    *roll -= ((float)gyrData[1] / GYROSCOPE_SENSITIVITY) * dt;    // Angle around the Y-axis

    // Compensate for drift with accelerometer data if !bullshit
    // Sensitivity = -2 to 2 G at 16Bit -> 2G = 32768 && 0.5G = 8192
    int forceMagnitudeApprox = abs(accData[0]) + abs(accData[1]) + abs(accData[2]);
    if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)
    {
        // Turning around the X axis results in a vector on the Y-axis
        pitchAcc = atan2f((float)accData[1], (float)accData[2]) * 180 / M_PI;
        *pitch = *pitch * 0.98 + pitchAcc * 0.02;

        // Turning around the Y axis results in a vector on the X-axis
        rollAcc = atan2f((float)accData[0], (float)accData[2]) * 180 / M_PI;
        *roll = *roll * 0.98 + rollAcc * 0.02;
    }
}
```

If we use the sweetness of having a **real** computer (Raspberry Pi) collecting our data, we can easily create a graph using GNUPlot. It's easily seen that the filter (red) follows the gyroscope (blue) for fast changes, but keeps following the mean value of the accelerometer (green) for slower changes, thus not feeling the noisy accelerometer data and not drifting away eiter.



The filter is very easy and light to implement making it perfect for embedded systems. It should be noted that for stabilization systems (like the quadrocopter), the angle will never be very big. The atan2 function can then be approximated by using a small angle approximation. In this way, this filter could easily fit on an 8 bit system.

Hope this article was of any help.

Tags:
Complementary Filter Gyroscope Accelerometer IMU Quadrocopter

Parker
Sat, 11/05/2013 - 04:09
permalink

I also tried to use the same

I also tried to use the same complimentary filter on ARdrone for attitude estimation with weight being 0.99, 0.01. I find that for small movement, the estimation is fine, but the filter can not track quick movement of the drone in flight. Do you have any suggestions?

reply

Pieter-Jan
Sat, 11/05/2013 - 04:45
permalink

The quick movement should not

The quick movement should not be dependant on the type of filter, as this would be almost purely the data from the gyroscope integration. What is your sampling rate? Do you have the same problem if you don't use a filter and just look at the gyroscope data?
reply

AbdElmoneam
Thu, 16/05/2013 - 16:19

I'm using the same IMU (MPU

permalink

I'm using the same IMU (MPU-6050) with Arduino to calculate the x-axis angle, I'm doing it with simplified kalman filter and its giving a stable data when its rotating without any linear motion, when I start to move it in the x direction the value of the angle gets missed up till I stop moving it, I'm working on project (inverted pendulum on two wheels) so I need the angle stable regardless the motion in the x direction, what can I do to fix it?

reply

Pieter-Jan
Fri, 17/05/2013 - 06:57
permalink

Your algorithm is probably

Your algorithm is probably too dependant on the data from the accelerometer. Why don't you use the Complementary Filter I proposed in this article? Notice that I only use 2% of the data from the accelerometer? This ensures that linear motion does not interfere with our angle measurement. Also the "if" condition checks if the accelerometer data is not too big, which would be the case if there is a large force working on the object. You don't want to use the data then. I hope it works out for you.

reply

CJ Luu
Sun, 19/05/2013 - 22:28
permalink

Why do you subtract roll but

Why do you subtract roll but add pitch?

reply

Pieter-Jan
Mon, 20/05/2013 - 13:08
permalink

CJ Luu, if you combine the

CJ Luu, if you combine the accelerometer angle and the gyroscope angle, you have to make sure that they have the same sign for the same excitation direction. This is why ...

reply

Ankit_Raj
Thu, 23/05/2013 - 03:42
permalink

I am using arduIMU which has

I am using arduIMU which has IMU6000 (3-axis gyro and accelerometer) and on-board Atmega328p to make a self-stabilizing platform . Is complementary (or kalman) filter preferred for this. What should be the suitable tuning values for gyro and acce. ??

reply

Pieter-Jan
Fri, 24/05/2013 - 07:10
permalink

You can use either the Kalman

You can use either the Kalman or complementary filter. Just choose what you think is best. I would suggest that you start reading your datasheet very carful. This is the start of any electronics project. The "tuning values" (I guess that you mean sensitivity values) will be in there.

reply

pablopaolus
Sat, 25/05/2013 - 13:56
permalink

Hi Pieter-Jan,

Hi Pieter-Jan,

Thank you for sharing your work. I'm using PIC18F46J50 as MCU along with Sparkfun IMU - 6DOF ITG3200/ADXL345, and I'm trying to combine accelerometer and giroscope data using the complementary filter. I've ported your code (to CCS v4.140):

```
float RwAcc[3]={0, 0, 0}; //Projection of normalized gravitation force vector on x/y/z axis, as measured by accelerometer
float Gyro_ds[3]={0, 0, 0}; //Gyro readings

float interval = 0;
unsigned int timerCount250us=0; //Timer count. Each unit counts 250 us
unsigned int timerCount63750us=0;

float pitch=0, roll=0;

void ComplementaryFilter()
{
float pitchAcc, rollAcc;

interval = 0.250*(float)timerCount250us + 63.750*(float)timerCount63750us; //Units: ms
interval /= 1000.0; //Conversion to sec.
//Restart the counters
timerCount250us = 0;
timerCount63750us = 0;

// Integrate the gyroscope data -> int(angularSpeed) = angle
// Gyro_ds is already divided by sensitivity (14.375 LSB per °/s )
pitch += Gyro_ds[0] * interval; // Angle around the X-axis
roll -= Gyro_ds[1] * interval; // Angle around the Y-axis

// Compensate for drift with accelerometer data if !bullshit
// Sensitivity= -2g to 2g at 10bit -> 2g = 512 && 0.5g = 128... ok???
// RwAcc is already divided by sensitivity (256 LSB per g) -> 128/256=0.5 && 512/256=2
float forceMagnitudeApprox = abs(RwAcc[0]) + abs(RwAcc[1]) + abs(RwAcc[2]);
if (forceMagnitudeApprox > 0.5 && forceMagnitudeApprox < 2.0)
{
// Turning around the X axis results in a vector on the Y-axis
pitchAcc = atan2(RwAcc[1], RwAcc[2]) * 180 / PI;
pitch = pitch * 0.98 + pitchAcc * 0.02;

// Turning around the Y axis results in a vector on the X-axis
rollAcc = atan2(RwAcc[0], RwAcc[2]) * 180 / PI;
roll = roll * 0.98 + rollAcc * 0.02;
}
}
```

And I use an internal interruption so as to calculate the interval. Timer overflows every 250us:

```
#int_TIMER0
void TIMER0_isr(void)
{
timerCount250us+=1;
if(timerCount250us==255) {
timerCount63750us+=1;
}
set_timer0(64036);
}
```

However, it doesn't seem to be working properly. When I tilt the sensor it works, but then it returns to the original position... I've recorded a video to clarify my problem:

http://www.youtube.com/watch?v=0hNkNRYQg2I

I would greatly appreciate if you could help me.

Thank you.

reply

Pieter-Jan
Sun, 26/05/2013 - 06:10
permalink

Mmh I don't really have time

Mmh I don't really have time to evaluate your code into detail but from what I see everything looks like it is programmed correctly.

From the video it seems like your gyroscope data is correct, as the fast changes are recorded correctly. Are you sure that your accelerometers and gyroscopes are oriented in the same way as in my case (MPU6050)? What if you only read the angle from the accelerometer data? Is it the angle you expect?

If the orientation is correct, it might be that your gyroscope angles drift much harder than mine and "wins" from the compensation of the accelerometer. Do you use the full resolution of your sensors? Try a slower sampling interval? MEMs are not made to be sampled that fast ... Mechanical processes are slow! And most importantly. Tweak the values from your complementary filter so that you use more accelerometer and less gyroscope so that you get rid of that nasty drift!

reply

pablopaolus
Fri, 31/05/2013 - 11:44
permalink

Pieter-Jan: Thank you very

Pieter-Jan: Thank you very much for your quick response and guide! It was a matter of different orientation of my accelerometer and gyroscope. Now it works well enough... except for one thing. If I tilt, for instance, pitch angle to 90 or -90 deg, roll angle goes crazy. The same occurs to pitch if I tilt roll to +-90. I don't know if I've explained myself clearly, so I've made a video:

http://www.youtube.com/watch?v=V1IMvj_iWlo

I don't know why this is happening.
Anyway, here is my code:

```
void ComplementaryFilter()
{
float pitchAcc, rollAcc;

interval = (0.250*(float)timerCount250us + 63.750*(float)timerCount63750us)/1000.0; //Units: sec
//Restart the counters
timerCount250us = 0;
timerCount63750us = 0;

rollAcc = atan2(RwAcc[1], RwAcc[2]) * 180 / PI;
pitchAcc = atan2(RwAcc[0], RwAcc[2]) * 180 / PI;

// Integrate the gyroscope data -> int(angularSpeed) = angle
// Gyro_ds is already divided by sensitivity (14.375 LSB per °/s )
roll += Gyro[0] * interval; // Angle around the X-axis
pitch -= Gyro[1] * interval; // Angle around the Y-axis

// Compensate for drift with accelerometer data if !bullshit
// Sensitivity= -2g to 2g at 10bit -> 2g = 512 && 0.5g = 128...
// RwAcc is already divided by sensitivity (256 LSB per g) -> 128/256=0.5 && 512/256=2
float forceMagnitudeApprox = abs(RwAcc[0]) + abs(RwAcc[1]) + abs(RwAcc[2]);
if (forceMagnitudeApprox > 0.5 && forceMagnitudeApprox < 2.0)
{
// Turning around the X axis results in a vector on the Y-axis
roll = roll * 0.97 + rollAcc * 0.03;
// Turning around the Y axis results in a vector on the X-axis
pitch = pitch * 0.97 + pitchAcc * 0.03;
}
}
```

If you have any idea of what the problem can be, please tell me.
Thank you very much.

reply

Pieter-Jan
Fri, 31/05/2013 - 12:42
permalink

Good that it's working now!

Good that it's working now!

I see the same behaviour as you when one of the angles goes to 90°. This is because at this value, the gravity vector shifts to a different axis and the atan2 function does not behave properly. I did not fix this in my code as I am only interested in variations of +-60°. You could try this document: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf It describes mathematically how to read the orientation from the accelerometers, and gives a different result as the one I used. Maybe it is better suited. The document should be good as it is referenced to a lot on the internet. Sorry I can not help you with more detail. Let me know if you can fix it!

reply

pablopaolus
Fri, 31/05/2013 - 13:39
permalink

Ok, I'll let you know if I am

Ok, I'll let you know if I am able to fix it!
Anyway, I think I won't probably need +/- 90º variations... I just asked in case I was doing something wrong, or it existed a quick solution.
I'd like to reiterate my gratitude to your help and quick responses. Congratulations for your blog, it has helped me a lot when I have gone crazy trying to implement Kalman Filter... unaware of the Complementary Filter goodness.

reply

Bunower
Mon, 15/07/2013 - 08:11
permalink

Hallo Pieter-Jan,

Hallo Pieter-Jan,
thank you for the great work.
I want to use the complementary filter for stabilize my quadcopter, there is only one problem.
when the motors are working, the vibration lets the data drift, how can i filter out the vibrations?

reply

Bunower
Mon, 15/07/2013 - 08:13
permalink

And what if i dont use the

And what if i dont use the filter on an infinite loop?
should i change dt? or is dt always 0.01?

reply

koorosh
Tue, 30/07/2013 - 06:56
permalink

I need an example of a very

I need an example of a very simple program with codevision(Complementary Filter)
I am using the adxl203&mlx90609(gyro) module and i need filter data
please help me...
thanks a lot...
Sorry for my bad English because I'm from Iran & my English very poor

reply

Marc
Wed, 14/08/2013 - 07:10
permalink

Hi Pieter-Jan,

Hi Pieter-Jan,
I am using an MPU6050 on my device that is connected to a Ti AM3517 via i2c. The driver I am using is ported from the RaspberryPi and generates angle and accelerometer data and it all looks nice.
I get angle in +/-degrees and acceleration is around 4000 for z when stationary (which I take to be 1g)

I have an initial problem with drift when I have initialised the 6050 to use the DMP, it takes around 12 seconds to settle for the x axis, do you see any issues with the startup of your device ?

I would like to find a way to stop the error as I may need the x,y,z data to b ready as soon as it has been initialised.

What are you using for your setup ? do you use the DMP and also what are the gyro and accel resolutions you are using (I'm using +/-2000 deg)

Hope you can help.

Thanks
Marc

reply

bob
Tue, 10/09/2013 - 23:42
permalink

your code wrong?

your code wrong?
*pitch = *pitch * 0.98 + pitchAcc * 0.02; <=== where is the angle in first term?

From the given Eq.
angle = 0.98*(angle+gryData*dt) + 0.02(accData)

So It should be?
Oldpitch = 0.98 (*Oldpitch + Pitch)+ pitchAcc * 0.02;
when
pitch += ((float)gyrData[0] / GYROSCOPE_SENSITIVITY) * dt; // Angle around the X-axis

reply

Pieter-Jan
Wed, 11/09/2013 - 02:11
permalink

No, the code I posted is

No, the code I posted is correct.
reply

Nguyễn Tiến Sự
Thu, 12/09/2013 - 21:25
permalink

can you give me the source

can you give me the source code to read data from MPU-6050?
my email is: nguyentiensu91@gmail.com
thanks!

reply

Nguyễn Tiến Sự
Sun, 15/09/2013 - 06:03
permalink

Hi!

Hi!
why did you choose the scale is -2G to +2G (why didn't -4G to +4G?), the sensitive of the accelerometer is 8192 (0.5G) and the sensitive of the gyroscope is 65.536?
thanks!

reply

HK
Sat, 21/09/2013 - 05:18
permalink

Hi ,

Hi ,
Can You please send me source code for reading data from MPU600 module.

reply

HK
Sat, 21/09/2013 - 05:23
permalink

my email id is hk1117@gmail

my email id is hk1117@gmail.com

reply

BD
Mon, 30/09/2013 - 04:46
permalink

Hi!

Hi!
First thank you for sharing your work, it is well explained and very helpful to me.
I juste have one question about the results you had : how much accurate do you think this method could be? Is the error approximatively 5°, 1°, 0.1°?
I think it is very interesting to have good results with a simple method!

reply

nima
Sat, 05/10/2013 - 07:29
permalink

how can i use this filter for

how can i use this filter for 9dof (gy-80) ?

reply

Jonathan R
Wed, 09/10/2013 - 13:30
permalink

Hi Pieter,

Hi Pieter,

I will soon be working on a quadrotor and I will be using the MPU-6000 IMU.
While waiting for my mcu, I started looking into how I should treat the gyro and accel data and I had started looking into the Kalman filter. As you said, it is quite difficult to implement and all so I was quite happy when I found your blog in which you used the complimentary filter.
I was wondering if you could send me your code so I can better understand how you acquire the accelerometer and gyroscope data as well as how it's implemented in the complimentary filter.
my email is: jonathan_rompre@hotmail.com

Thank you very much.

reply

Mostafa
Tue, 05/11/2013 - 04:32
permalink

Hi, Peter !!

Hi, Peter !!

thanks for ur work .. please, do u mind if send me the code of reading from MU6050 .?!!

i am working on platform STM32F303 (i know different Gyroscope and ACC model) i am interesting to ss ur code .. my mail mostafa.e.mansour@gmail.com

Best regards

reply

horoscope chinois
Wed, 06/11/2013 - 05:41
permalink

Hi my friend! I want to say

Hi my friend! I want to say that this post is amazing, great written and come with approximately all vital infos. I would like to see more posts like this
.

reply

Andrej
Tue, 19/11/2013

Hi Pieter-Jan,

- 17:01
permalink

Hi Pieter-Jan,

your code calculates roll and pitch. Is it possible to calculate yaw also?

Kind regards

reply