Android Sensor Fusion Tutorial

Last but not least we can implement the complementary filter. To have more control over its output, we execute the filtering in a separate timed thread. The quality of the sensor signal strongly depends on the sampling frequency, that is, how often the filter method is called per second. That's why we put all the calculations in a *TimerTask* and define later the time interval between each call.

```
class calculateFusedOrientationTask extends TimerTask {
    public void run() {
        float oneMinusCoeff = 1.0f - FILTER_COEFFICIENT;
        fusedOrientation[0] =
            FILTER_COEFFICIENT * gyroOrientation[0]
            + oneMinusCoeff * accMagOrientation[0];

        fusedOrientation[1] =
            FILTER_COEFFICIENT * gyroOrientation[1]
            + oneMinusCoeff * accMagOrientation[1];

        fusedOrientation[2] =
            FILTER_COEFFICIENT * gyroOrientation[2]
            + oneMinusCoeff * accMagOrientation[2];

        // overwrite gyro matrix and orientation with fused orientation
        // to comensate gyro drift
        gyroMatrix = getRotationMatrixFromOrientation(fusedOrientation);
        System.arraycopy(fusedOrientation, 0, gyroOrientation, 0, 3);
    }
}
```

If you've read the first part of the tutorial, this should look somehow familiar to you. However, there is one important modification. We overwrite the gyro based orientation and rotation matrix in each filter pass. This replaces the gyro orientation with the "improved" sensor data and eliminates the gyro drift.

Of course, the second important factor for the signal quality is the *FILTER_COEFFICIENT*. I've determined this value heuristically by rotating a 3D-model of my smartphone using the sensor from my actual device. A value of 0.98 with a sampling rate of 33Hz (this yields a time period of 30ms) worked quite well for me. You can increase the sampling rate to get a better time resolution, but then you have to adjust the *FILTER_COEFFICIENT* to improve the signal quality.

So these are the final additions to our sensor fusion code:

```
public static final int TIME_CONSTANT = 30;
```

```
2  public static final float FILTER_COEFFICIENT = 0.98f;
3  private Timer fuseTimer = new Timer();
4  public void onCreate(Bundle savedInstanceState) {
5
6      // ...
7
8      // wait for one second until gyroscope and magnetometer/accelerometer
9      // data is initialised then scedule the complementary filter task
10     fuseTimer.scheduleAtFixedRate(new calculateFusedOrientationTask(),
11                                   1000, TIME_CONSTANT);
12 }
```

I hope this tutorial is a sufficient explanation on custom Android based sensor fusion. If you find any mistakes I've made or have any questions, don't hesitate, please let me know.

127 RESPONSES TO *ANDROID SENSOR FUSION TUTORIAL*

Paul *says:*
2013-06-16 AT 5.40 PM

Hallo tuvbunn2,
was genau hast du denn schon versucht? Im Prinzip müsstest du die Rotationsmatrix in ein anderes Koordinatensystem umrechnen. Da ich mich schon länger nicht mehr mit dem Code auseinandergesetzt habe, müsste ich mich selber erst mal hinsetzen und darüber nachdenken (was ich leider zur Zeit nicht kann 😟).

tuvbunn2 *says:*
2013-06-19 AT 4.31 PM

Hi,

Ich habe eine Rotationsmatrix um die X achse gebaut und rotationMatrix damit in calculateAccMagOrientation() gedreht.

Leider funktioniert dann mein roll nicht mehr, bzw der Horizont korrigiert dann auf seltsame weise.

Das hier sind meine Anpassungen:

```
public void calculateAccMagOrientation()
{
if (SensorManager.getRotationMatrix(rotationMatrix, null, accel, magnet))
{
//rotation x axis 90 degrees
float[] xRotationMatrix = new float[9];
xRotationMatrix[0]=1f;
xRotationMatrix[1]=0f;
xRotationMatrix[2]=0f;
xRotationMatrix[2]=0f;
xRotationMatrix[0]=0f;
xRotationMatrix[5]=1f;
xRotationMatrix[6]=0f;
xRotationMatrix[7]=-1f;
xRotationMatrix[8]=0f;

rotationMatrix=matrixMultiplication(rotationMatrix, xRotationMatrix);

SensorManager.getOrientation(rotationMatrix, accMagOrientation);
}
}
```

Danke und Grüße,
tuvbunn2

Paul *says:*
2013-06-20 AT 10.34 PM

Hi tuvbunn2,
warum weist du xRotationMatrix[0] einmal 1f zu und überschreibst es weiter unten gleich wieder mit 0f? Ich glaube, dass du da einen copy-paste-Fehler in der Initialisierung deiner Rotationsmatrix hast, denn index 3 und 4 werden garnicht initialisiert, stattdessen 0 und 2 zweimal.

Außerdem reicht es nicht, wenn du allein den AccMag-Anteil rotierst. Den Gyro-Anteil (siehe gyroMatrix) musst du auch transformieren, sonst hast du später in der Sensor-Fusion inkonsistente Daten.

Grüße,
Paul

tuvbunn2 *says:*
2013-06-25 AT 11.49 AM

Oh vielen Dank,
Da ist wohl beim kopieren etwas durcheinander gekommen.
Durch die Rotationen der rotationMatrix um 90° sowie der Rotation der gyroMatix um 90° passt jetzt alles.

Vielen Dank.

Grüße,
tuvbunn2

PINGBACK: *GYROSCOPE ISSUES WITH DEVICE ORIENTATION | BLOGOSFERA*

Phillip *says:*
2013-07-26 AT 2.50 PM

Hi Paul,

This is so useful article!

But i have the same problem as tuvbunn2, i rotate the rotationMatrix by 90 before pasing to accMagOrientation which makes pitch work fine but the roll value deviate (no deviation for 45 degrees pitch though).
Where exactly in the alogrithm should i rotate gyro matrix?
I would be so grateful for helping me out with this

Paul *says:*
2013-07-26 AT 10.51 PM

Hi Phillip,
you hav to rotate both, the GyroMatrix and the AccMagMatrix (which is called rotationMatrix within the code) by 90° around the x-axis. you have to create a rotation matrix for that, just like tuvbunn2 did:

```
 1  0  0

0  0  1

0 -1  0
```

You can apply the rotation using the MatrixMultiplication-Method. The above rotation matrix has to be the second parameter (remember that matrix multiplications are not commutative). Just apply the rotation right after GyroMatrix/AccMagMatrix have been calculated respectively. For the AccMagMatrix that would be somewhere at the end of the method calculateAccMagOrientation and for the gyroMatrix somewhere at the end of the method gyroFunction.
Hope that helps.

Phillip *says:*
2013-07-27 AT 12.56 PM

Unfortunetely it doesnt work, after mulitplying gyroMatrix the values go crazy…
I am using your project and testing on few mobile devices:

```
float[] my90DegRotationMatrix = {1, 0, 0,
0, 0, 1,
0, -1, 0,
};
```

then in calculateAccMagOrientation:

```
if(SensorManager.getRotationMatrix(rotationMatrix, null, accel, magnet)) {
rotationMatrix = matrixMultiplication(rotationMatrix, my90DegRotationMatrix);
SensorManager.getOrientation(rotationMatrix, accMagOrientation);
}
```

and in gyroFunction at the very end:

```
// apply the new rotation interval on the gyroscope based rotation matrix
gyroMatrix = matrixMultiplication(gyroMatrix, deltaMatrix);

// rotate gyro matrix NEW
gyroMatrix = matrixMultiplication(gyroMatrix, my90DegRotationMatrix);
```

// get the gyroscope based orientation from the rotation matrix
SensorManager.getOrientation(gyroMatrix, gyroOrientation);

I tried rotating gyromatrx in few other places like at the end of calculateFusedOrientationTask etc. It seem like not matter where i do that it breaks everything.

Can you please help me, am i doing the gyro matrix multiplication wrong?

Paul *says:*
2013-07-30 AT 7.51 PM

Hi Phillip,
I'm sorry, but currently I don't have the project setup on my local machine so I cannot reproduce the behavior of your code. I was looking through the code and couldn't find anything, yet. I will report back, if I find something.

Jeff *says:*
2013-08-01 AT 4.40 PM

Hi Paul,
This article is amazing. Well done.

I am experiencing an issue I was hoping you can help with. In my own application and in your sample application, the Pitch value goes from 0 (device laying flat) to -90 (device completely vertical), then back down toward 0 (device flat but upside down). This means the pitch value is -45 both when the phone is titled back (\) and when its tilted forward (/). I am using a Samsung Galaxy Nexus. Are these the values you would expect to see? How can I tell the difference between the two positions?

Thanks!

Paul *says:*
2013-08-02 AT 5.31 PM

Hi Jeff,
pitch, roll and azimuth always keep their relations to each other. That said, you can read the azimuth value in order to interpret which orientation, (/) or (\) is meant by a pitch value of -45°. If the reading direction (–>) is front, an azimuth of 0° and a pitch of -45° would be (\) (with the earpiece pointing down) while with an azimuth of 180° it would be (/). Of course if you flip the roll angle by 180° the whole situation is different. So I'm affraid you have to take into account all three angles.

oh *says:*
2013-08-12 AT 9.39 AM

It would be grand to apply this to android Dead Reckoning (continue to navigate (supplement) with weak gps or weak data (tower masts))

fariba *says:*
2013-08-17 AT 12.17 PM

Hi
The program is very good and very useful, thank you.
I've used your program in my project.Could you explain more about the TIME_CONSTANT and the FILTER_COEFFICIENT And how did you get them?

Paul *says:*
2013-08-18 AT 5.52 PM

Hi fariba,
thank you for your nice feedback. I think I already explained those two constants pretty much in detail further down in the comments. Try to look on page 2 of the comments. I believe there is a more lengthy explanation on this which I posted some time ago.

Pok *says:*
2013-08-19 AT 1.29 AM

Thank you for the tutorial Paul. Very useful. Is it possible to install it on Android 2.2 devices? Im getting a parsing error so just wondering it might be because of incompatibility (?)

Pok *says:*
2013-08-19 AT 1.45 AM

I think the reason is getrotationmatrixfromvector which has been added in API9.

Paul *says:*
2013-08-19 AT 5.29 PM

That could be. I didn't test the example with older API levels.

Rajesh *says:*
2013-09-02 AT 11.31 PM

Hi
Nice article ! I have a question !
How can I convert linear acceleration given by accelerometer sensor with the device frame of reference to the earth frame of reference ? I think it will need sensors (Acc, gyro, Mag) fusion to determine that.

From (Xa, Ya, Za) to (Xe, Ye, Ze) ?

please help

fariba *says:*
2013-09-09 AT 6.08 PM

Hey Paul
If I want to hold the phone in one of the following three conditions Should I change something in the code?
1 – when the x-axis parallel and opposite to the vector of gravity .
2 – when the y-axis parallel and opposite to the vector of gravity .
3 – when the z-axis parallel and opposite to the vector of gravity .
please explain what should I do For each case . Just have to change the rotation matrix? How?
please help me

ASHWINI *says:*
2013-10-06 AT 6.46 PM

Hello sir,

Thanks a lot for the tutorial,
actually am working on indoor tracking in gps denied areas using sensor fusion in android devices,
the source code you provided helps in gathering sensor values?
or fusing them as per sensor fusion so that they used for different purposes???

Android Example *says:*
2013-10-07 AT 12.35 PM

Very nice dude…

I have also found one good link here….

Accelerometer Basic Example – Detect Phone Shake Motion

Paul *says:*
2013-11-02 AT 6.58 PM

@ASHWINI: It's basically sensor fusion, nothing more.

Aaron *says:*
2013-11-07 AT 2.41 PM

Hi Paul,
Good job.
I'm interested in Android PDR and can you send me the source code? I can't find the newest version of you code…

Paul *says:*
2013-11-11 AT 5.42 PM

Hi Aaron,

there is a link in the post after the fourth paragraph. The text starts wit “Update (March 22, 2012)”. There you can download the code to this tutorial. Otherwise try this link.

Bartek *says:*
2013-11-19 AT 12.53 PM

Hello Paul,
Thanks for such a great tutorial!

I don’t understand why are you using the magnetometer if it’s so inaccurate and generate noise? Why not use accelerometer instead?

Even the accelerometer output look the same as magnetometer output.

Hao *says:*
2013-11-27 AT 1.35 AM

Dear Paul,

Can your also calculate the horizontal degree to the North?
Can it also be benefit from the filter your define?
Thanks.

Hao

Paul *says:*
2013-11-28 AT 12.29 AM

@Bartek:
The magnetometer is the only non-gyro reference for the azimuth angle we have available. You cannot determine the azimuth angle with the accelerometer, only pitch and yaw.
@Hao:
Could you be more specific? Do you mean the angle between the devices ‘forward’ vector and the noth direction? In 3D or projected on a horizontal plane?
The main problem would always be to determine north accurately enough. This could be problematic with the noisy magnetometer. You could point the device

towards north using some other method (another accurate compass), save the azimuth angle (and thus ‘calibrate’ it, so to say) and calculate the difference between those two vectors. However, I have the feeling this would not be a satisfying solution to you.