加速度计的椭球校准

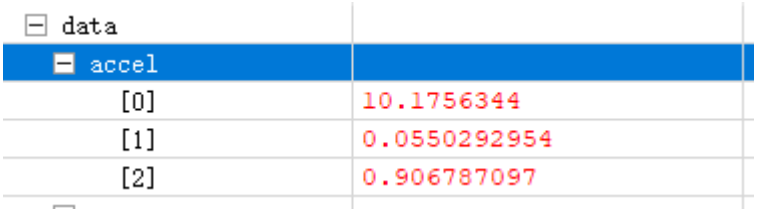
ZYC 2022.9.9

- 目录

问题的提出

IMU是一种常用的用于测量物体加速度值、角速度值的惯性传感器，作为物体姿态测量中的重要传感器，我们对其精度具有严格要求。由于IMU型号的差异乃至同一型号由于加工制造误差所造成的差异，使得不同IMU在同一套算法下计算的结果会有所不同，所以IMU在使用前往往会进行校准。对于陀螺仪的校准，我们常用的方法是消除静态误差的方法，即静止状态下多次测量取平均值得到偏移误差，将测量的陀螺仪数据减去静态误差便可得到校准数据。但加速度计却很少使用校准，一方面是因为对于高精度的IMU芯片，其内置的加速度计往往精度很高，不太需要校准，另一方面是加速度计不能使用陀螺仪消除静态误差的方法那样简单地获得误差，所以见到校准方法。但对于低精度的陀螺仪，例如MPU6050，加速度计校准是非常有必要的。下面我举个例子说明。

以我手中的MPU6050（正点原子版）为例，在静止状态下两个姿态所读取的值为



data	
accel	
[0]	3.18571782
[1]	0.988134742
[2]	8.303442

显然 $\sqrt{10.176^2+0.055^2+0.907^2}=10.216$ $\sqrt{3.186^2+0.988^2+8.303^2}=8.948$

与实际中的9.8相差较大,而且随着陀螺仪的姿态不同读取的数据也不同，所以进行加速度计校准是十分必要的。经过阅读相关文献以及查阅资料，现找到一种较为简单易懂的校准方法————**加速度计椭球校准。**

加速度计的误差模型

公式:

$$accx = a_1 * accx_m + b_1$$

$$accy = a_2 * accy_m + b_2$$

$$accz = a_3 * accz_m + b_3$$

- $accx, accy, accz$: 校准后的加速度值
- $accx_m, accy_m, accz_m$: 校准前的加速度值
- a_1, a_2, a_3 : 比例系数
- b_1, b_2, b_3 : 零偏

所以我们共需要求取6个系数才能够完成校准

椭球模型推导

静止状态下

$$accx^2 + accy^2 + accz^2 = g^2$$

将误差模型代入

$$(a_1 \cdot accx_m + b_1)^2 + (a_2 \cdot accy_m + b_2)^2 + (a_3 \cdot accz_m + b_3)^2 = g^2$$

令 $accx_m = x, accy_m = y, accz_m = z$,可化为

$$\frac{(x + \frac{b_1}{a_1})^2}{\frac{g}{a_1}} + \frac{(y + \frac{b_2}{a_2})^2}{\frac{g}{a_2}} + \frac{(z + \frac{b_3}{a_3})^2}{\frac{g}{a_3}} = 1$$

此时令

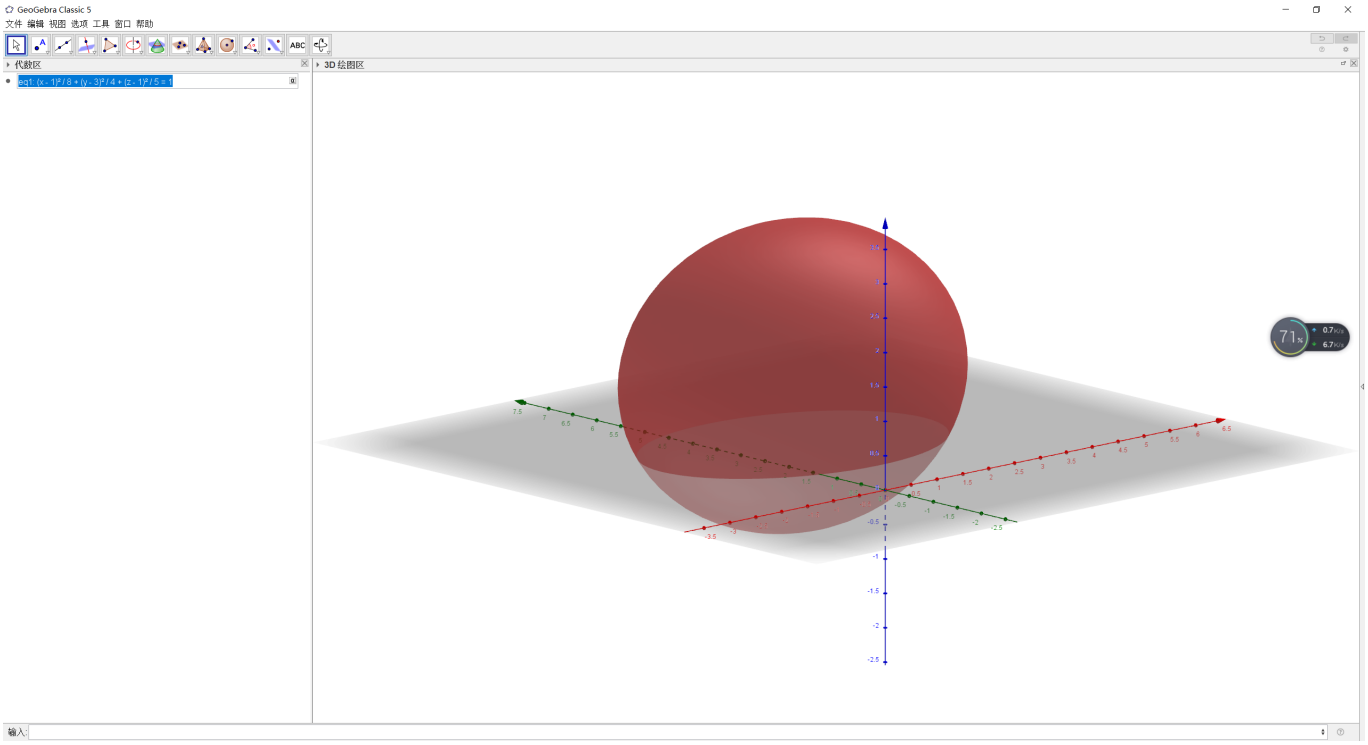
$$O_x = -\frac{b_1}{a_1}, O_y = -\frac{b_2}{a_2}, O_z = -\frac{b_3}{a_3}$$

$$R_x = -\frac{g}{a_1}, R_y = -\frac{g}{a_2}, R_z = -\frac{g}{a_3}$$

公式化为

$$(\frac{x-O_x}{R_x})^2+(\frac{y-O_y}{R_y})^2+(\frac{z-O_z}{R_z})^2=1$$

这便是常见的轴对齐椭球标准方程



也就是说加速度计测量的三个数值满足上述方程，但我们要如何获得椭球方程？这就需要读取不同数据，拟合出椭球方程。下面使用空间二次曲面数据拟合的方法（基于最小二乘法）。

椭球拟合

以下公式的推导主要参考以下博客

空间二次曲面数据拟合算法推导及仿真分析_Inspire-CSDN博客

最小二乘估计及证明_野生猿-群号1025127672-CSDN博客_最小二乘估计

三维空间中的椭球拟合+磁力计校准算法+加速度计校准算法

展开

$$(\frac{x-O_x}{R_x})^2+(\frac{y-O_y}{R_y})^2+(\frac{z-O_z}{R_z})^2=1$$

有

$$\frac{x^2}{R_x^2}-\frac{2xO_x}{R_x^2}+\frac{O_x^2}{R_x^2}+\frac{y^2}{R_y^2}-\frac{2yO_y}{R_y^2}+\frac{O_y^2}{R_y^2}+\frac{z^2}{R_z^2}-\frac{2zO_z}{R_z^2}+\frac{O_z^2}{R_z^2}=1$$

两边同时乘R_x^2

$$x^2+\frac{R_x^2}{R_y^2}y^2+\frac{R_x^2}{R_z^2}z^2+(-2O_x)x+(-2\frac{R_x^2O_y}{R_y^2})y+(-2\frac{R_x^2O_z}{R_z^2})z+(O_x^2+\frac{R_x^2O_y^2}{R_y^2}+\frac{R_x^2O_z^2}{R_z^2}-R_x^2)=0$$

令

$$A=\frac{R_x^2}{R_y^2},B=\frac{R_x^2}{R_z^2},C=-2O_x,D=-2\frac{R_x^2O_y}{R_y^2}$$

$$E=-2\frac{R_x^2O_z}{R_z^2},F=O_x^2+\frac{R_x^2O_y^2}{R_y^2}+\frac{R_x^2O_z^2}{R_z^2}-R_x^2$$

可得

$$Ay^2+Bz^2+Cx+Dy+Ez+F=-x^2$$

$$[y^2\quad z^2\quad x\quad y\quad z\quad 1]\cdot[A\quad B\quad C\quad D\quad E\quad F]^T=-x^2$$

令

$$k=[y^2\quad z^2\quad x\quad y\quad z\quad 1]$$

$$X=[A\quad B\quad C\quad D\quad E\quad F]$$

$$Y=-x^2$$

所以

$$KX=Y$$

故X的最小二乘解为

$$X=(K^TK)^{-1}K^TY$$

CSDN最小二乘法的矩阵表达---AHU-丁少侠

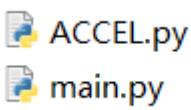
所以

$$O_x=-\frac{C}{2},O_y=-\frac{D}{2A},O_y=-\frac{E}{2B}$$

$$R_x=\sqrt{O_x^2+AO_y^2+BO_z^2-F},R_y=\sqrt{\frac{R_x^2}{A}},R_z=\sqrt{\frac{R_x^2}{B}}$$

应用

打开文件夹中的python文件ACCEL.py



python文件主要实现电脑和STM32板的UART通信，并将读取的数据处理得到结果,该文件包含的包有

```
import serial
import numpy as np
import struct
```

类的初始化,port表示端口，我的是"COM4"，bsp表示波特率，num表示一个点的采样个数

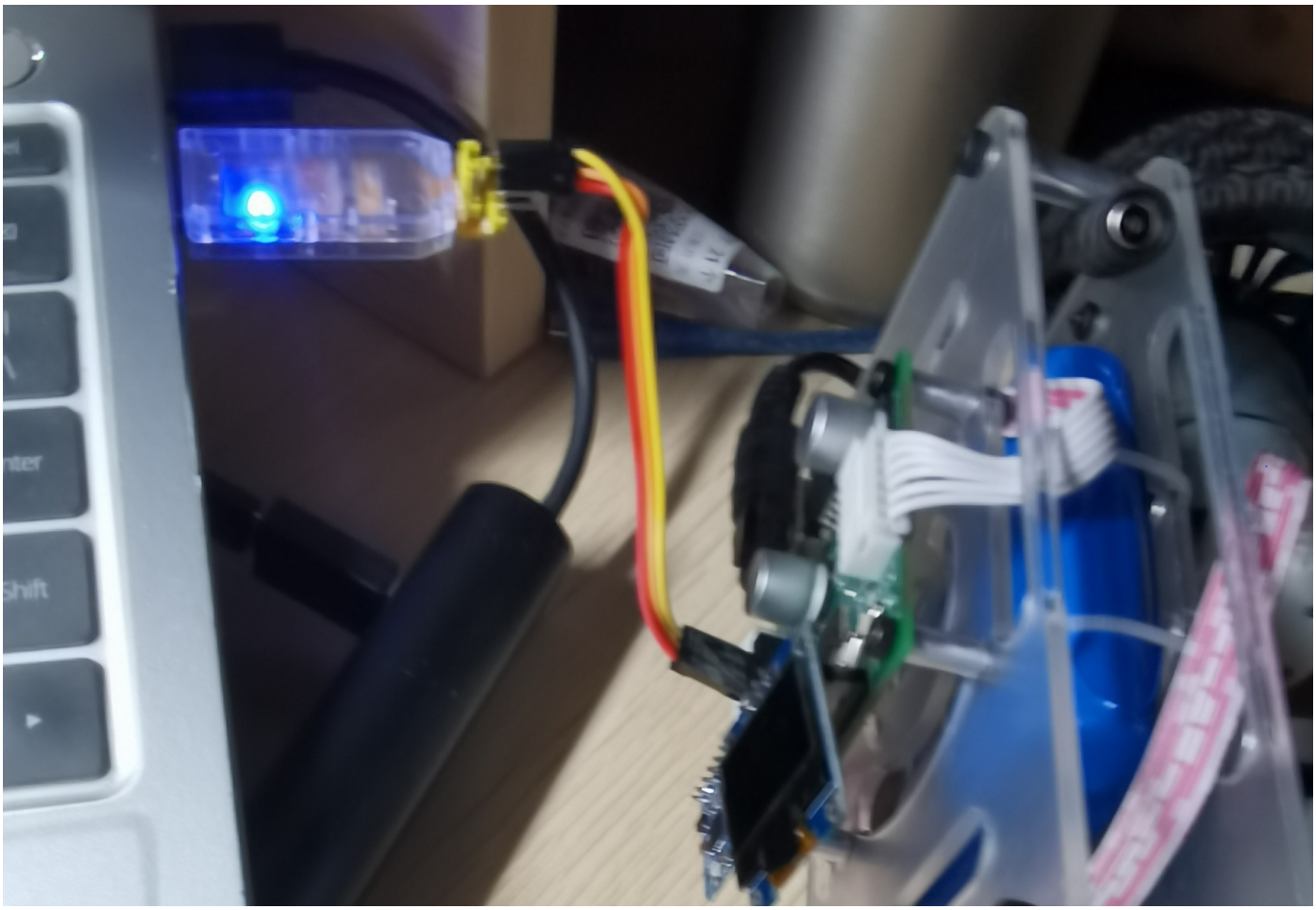
```
    def __init__(self,port,bsp,num)
```

初始化后直接调用类内函数

```
 def Calibration(self):
```

由于加速度计在高速中精度降低，所以读取的每一个数据都是在静止状态下，下面展示实践用法。

- 首先，需要将CH340转串口将电脑与板子的串口连接，然后确定端口号，修改代码中的端口号。我所使用的是芯片为STM32F103C8T6，IMU为MPU6050，STM32只需要读取加速度的值并一直发送就好。**注意，发送的是原数据而不是处理后的，也就是从MPU中直接读到的6个字节，不发送处理后的数据最主要是发送6个字节比发送3个float类型的数据占用带宽更少**



- 点击运行代码，输入start便可读取数据，数据个数为所设定的，数据越多则拟合越准。当记录完一个点后，便需要改变陀螺仪的姿态，继续录取其他点，输入start或next都可以，直到记录数据足够多，输入

end结束，返回a_1,a_2,a_3,b_1,b_2,b_3 。

```
请输入命令: end
[-9.18151855  0.04785156  0.11364746 ...  2.67011719  6.3546875
 -8.2244873 ]
0.9961208267027851
0.9961816912209319
0.9850846239868644
-0.5282283133798743
0.1308315497601815
1.1516887155070668

进程已结束，退出代码 0
```

- 将数据放入代码中检测结果

```
104    mpu_config.accelfactor[0] =0.9961208267027851;
105    mpu_config.accelfactor[1] =0.9961816912209319;
106    mpu_config.accelfactor[2] =0.9850846239868644;
107    mpu_config.accelbias[0] = -0.5282283133798743;
108    mpu_config.accelbias[1] = 0.1308315497601815;
109    mpu_config.accelbias[2] = 1.1516887155070668;
110    MPU6050_Create(&test,&mpu_config);
```

```
obj->calibration_data.accel[0] = obj->data.accel[0]*obj->config.accelfactor[0]+obj->config.accelbias[0];
obj->calibration_data.accel[1] = obj->data.accel[1]*obj->config.accelfactor[1]+obj->config.accelbias[1];
obj->calibration_data.accel[2] = obj->data.accel[2]*obj->config.accelfactor[2]+obj->config.accelbias[2];
```

实际效果

选取三个姿态读取数据

⊟ data	
⊟ accel	
[0]	10.1636715
[1]	0.00957031269
[2]	0.906787097
⊞ gyro	
⊞ euler	
⊞ euler_deg	
round	0
yaw	0.0154440291
⊟ calibration_data	
⊟ accel	
[0]	9.60316658
[1]	0.120106064
[2]	2.04495072
⊞	

⊟ accel	
[0]	3.41660166
[1]	0.943872094
[2]	8.28789043
⊞ gyro	
⊞ euler	
⊞ euler_deg	
round	0
yaw	-0.0263858512
⊟ calibration_data	
⊟ accel	
[0]	2.8143456
[1]	1.04488182
[2]	9.32185459
⊞ gyro	

⊟ accel	
[0]	-9.26765156
[1]	0.153125003
[2]	-0.0119628906
⊞ gyro	
⊞ euler	
⊞ euler_deg	
round	0
yaw	-1.27702999
⊟ calibration_data	
⊟ accel	
[0]	-9.75992966
[1]	0.310781479
[2]	1.14815331

分别计算(前一个为原数据，后一个为校准数据)

1. 图一

$$\sqrt{10.164^2+0.0096^2+0.907^2}=10.204, \sqrt{9.603^2+0.120^2+2.045^2}=9.810$$

2. 图二

$$\sqrt{3.417^2+0.944^2+8.288^2}=9.014, \sqrt{2.814^2+1.045^2+9.322^2}=9.793$$

3. 图三

$$\sqrt{(-9.268)^2+0.153^2+(-0.012)^2}=9.269, \sqrt{(-9.760)^2+0.310^2+1.148^2}=9.832$$

总结

从实际效果中可以看出，校准后的加速度值更贴近于真实的重力加速度。虽然在其他高精度的IMU中所测得的静态加速度计不需要校准，但懂得多一个方法对以后的应用也会有很大的帮助