```
 1
 2 //#include "matrix.h"
 3 #include "ahrs.h"
 4 //#include "magnet_cal.h"
 5 #include "basic_math.h"
 6 #include "flight_control.h"
 7
 8
 9 float offset[3];
10 float cor[3][3];
11
12 float q0 = 1, q1 = 0, q2 = 0, q3 = 0;
13 float gx_off, gy_off, gz_off;
14 float mx_mag, my_mag, mz_mag;
15 float wbx = 0, wby = 0, wbz = 0;
16 float by = 1, bz = 0;
17 float exInt = 0, eyInt = 0, ezInt = 0;
18
19 int count;
20 int ahrs_init_flag = 0;
21 int acc_over = 0;
22 extern int16_t gTHR;
23 float ahrs_kp;
24
25 void    ahrs_fusion_ag(AxesRaw_TypeDef_Float    *acc,    AxesRaw_TypeDef_Float    *gyro,
   AHRS_State_TypeDef *ahrs)
26 {
27   float axf,ayf,azf,gxf,gyf,gzf;
28   float norm;
29   float vx, vy, vz;
30   float ex, ey, ez;
31   float q0q0, q0q1, q0q2, q0q3, q1q1, q1q2, q1q3, q2q2, q2q3, q3q3;
32   float halfT;
33
34
35   if(gTHR<MIN_THR)
36   {
37     ahrs_kp = AHRS_KP_BIG;
38   }
39   else
40   {
41     ahrs_kp = AHRS_KP_NORM;
42   }
43
44   axf = acc->AXIS_X;
45   ayf = acc->AXIS_Y;
46   azf = acc->AXIS_Z;
47
48   // mdps convert to rad/s
49   gxf = gyro->AXIS_X * COE_MDPS_TO_RADPS;
```

機体姿勢を表すクォータニオン

スロットル・スティック操作量に応じて比例ゲインの値を切り替える。

3 軸の加速度

```
50  gyf = gyro->AXIS_Y * COE_MDPS_TO_RADPS;
51  gzf = gyro->AXIS_Z * COE_MDPS_TO_RADPS;
52
53  // auxiliary variables to reduce number of repeated operations
54  q0q0 = q0*q0;
55  q0q1 = q0*q1;
56  q0q2 = q0*q2;
57  q0q3 = q0*q3;
58  q1q1 = q1*q1;
59  q1q2 = q1*q2;
60  q1q3 = q1*q3;
61  q2q2 = q2*q2;
62  q2q3 = q2*q3;
63  q3q3 = q3*q3;
64
65  // normalise the accelerometer measurement
66  norm = invSqrt(axf*axf+ayf*ayf+azf*azf);
67
68  axf = axf * norm;
69  ayf = ayf * norm;
70  azf = azf * norm;
71
72  // estimated direction of gravity and flux (v and w)
73  vx = 2*(q1q3 - q0q2);
74  vy = 2*(q0q1 + q2q3);
75  vz = q0q0 - q1q1 - q2q2 + q3q3;
76
77  ex = (ayf*vz - azf*vy);
78  ey = (azf*vx - axf*vz);
79  ez = (axf*vy - ayf*vx);
80
81  // integral error scaled integral gain
82  exInt = exInt + ex*AHRS_KI*SENSOR_SAMPLING_TIME;
83  eyInt = eyInt + ey*AHRS_KI*SENSOR_SAMPLING_TIME;
84  ezInt = ezInt + ez*AHRS_KI*SENSOR_SAMPLING_TIME;
85
86  // adjusted gyroscope measurements
87  gxf = gxf + ahrs_kp*ex + exInt;
88  gyf = gyf + ahrs_kp*ey + eyInt;
89  gzf = gzf + ahrs_kp*ez + ezInt;
90
91  // integrate quaternion rate and normalise
92  halfT = 0.5*SENSOR_SAMPLING_TIME;
93  q0 = q0 + (-q1*gxf - q2*gyf - q3*gzf)*halfT;
94  q1 = q1 + (q0*gxf + q2*gzf - q3*gyf)*halfT;
95  q2 = q2 + (q0*gyf - q1*gzf + q3*gxf)*halfT;
96  q3 = q3 + (q0*gzf + q1*gyf - q2*gxf)*halfT;
97
98  // normalise quaternion
99  norm = invSqrt(q0 * q0 + q1 * q1 + q2 * q2 + q3 * q3);
```

（49 行目～）
3 軸の角速度

後の計算で繰り返し使う量をあらかじめ計算しておく

加速度ベクトルの長さを 1 に正規化

姿勢クォータニオンの前回値から，重力加速度ベクトルを推定する

姿勢の推定誤差，すなわち，[vx, vy, vz]を[axf, ayf, azf]へ近づけるための回転軸ベクトルを求める

[vx, vy, vz]を[axf, ayf, azf]へ近づけるため，PI 制御の要領で角速度[gxf, gyf, gzf]を補正する

姿勢クォータニオンを更新

```
100   q0 *= norm;
101   q1 *= norm;
102   q2 *= norm;
103   q3 *= norm;
104
105   ahrs->q.q0 = q0;
106   ahrs->q.q1 = q1;
107   ahrs->q.q2 = q2;
108   ahrs->q.q3 = q3;
109
110 }
```

（99 行目～）
更新したクォータニオンについて，ノルムが 1 になるように修正する