DT0059
设计建议

用于传感器校准的椭球或球体拟合

作者：Andrea Vitali

主要元件*	
LSM303AGR	超紧凑高性能电子罗盘：超低功耗 3D 加速度计和 3D 磁力计
LSM6DS3	iNEMO 惯性模块：3D 加速度计和 3D 陀螺仪

目的和益处

该设计建议说明了如何通过执行球体（椭球）拟合来计算 3 轴传感器的偏移、增益和交叉轴增益。该技术通常用于校准和补偿磁力计，但它也可与其他传感器一起使用，例如加速度计。

优势：

- 添加了 MotionFX 库提供的校准功能，该库仅可为磁力计提供偏移补偿。
- 简短且必要的实现，可以让最终用户轻松实现定制和增强（osxMotionFX 仅以二进制格式提供，而非源代码）
- 易于在任意微控制器上使用（osxMotionFX 只能在 STM32 上运行，并且只有在 Open.MEMS 许可证服务器发布了适当的许可证时才能运行）。

算法描述

可对多个位置（N）进行测量，并可进行组合以找到未知数（偏移、增益和交叉轴增益）。

对于 6 点翻滚校准，需要准确布置传感器。但是，对于这里描述的椭球拟合，不需要知道传感器的真实激励，因为唯一的要求是真实激励的模数是常数（X、Y 和 Z 的平方和的平方根）。

- 对于磁力计的情况：要仅测量地磁场，不得存在任何其他寄生（通常是随时间变化的）磁异常；那么真实激励的模数就是地磁场的模数
- 对于加速度计的情况：要仅测量重力，传感器不得有任何其他加速度；那么真实激励的模数就是重力的模数

最一般的情况下，下面的等式有 9 个未知数，

$\mathbf{v = [\,a, b, c, d, e, f, g, h, i\,]^T}$，其中数据点在旋转的椭球体上。如果椭球体无旋转，则其轴将与 X、Y 和 Z 轴对齐，这里相应方程只有 6 个未知数 $\mathbf{v = [\,a, b, c, g, h, i\,]^T}$。如果轴的长度都相等，则它是一个球体，相应方程只有 4 个未知数 $\mathbf{v = [\,a{+}b{+}c, g, h, i\,]^T}$。一般方程如下：

$$\mathbf{a\,X^2 + b\,Y^2 + c\,Z^2 + d\,2XY + e\,2XZ + f\,2YZ + g\,2X + h\,2Y + i\,2Z = 1}$$

使用 N 个数据点的集合来构建数据矩阵 D，其中数据点不能是共面的：

- 旋转椭球:**D 行 = [X^2, Y^2, Z^2, 2XY,2XZ, 2YZ, 2X, 2Y, 2Z]**，其中 D 为[Nx9]。至少需要 9 个数据点来计算偏移、增益和交叉轴增益
- 非旋转椭球：**D 行 = [X^2, Y^2, Z^2, 2X, 2Y, 2Z]**，其中 D 为[Nx6]，至少需要 6 个数据点来计算偏移和增益
- 球体：**D 行 = [$X^2+Y^2+Z^2$, 2X, 2Y, 2Z]**，其中 D 为[Nx4]。至少需要 4 个数据点来计算偏移

旋转椭球拟合

现在，可以通过使用非方矩阵的伪逆来计算 **v** 中未知数的最小二乘误差近似。首先，两边都乘以转置 $\mathbf{D^T}$。其次，两边都乘以方 $\mathbf{D\,D^T}$ 的逆。可能存在 9、6 或 4 个未知数，这取决于前述约束条件。对于一般情况：

$$\mathbf{D[Nx9]\ v[9x1] = 1\ [Nx1] \rightarrow D^T[9xN]\ D[Nx9]\ v[9x1] = D^T[9xN]\ 1[Nx1] \rightarrow}$$

$$\mathbf{(D^T\,D)[9x9]\ v[9x1] = (D^T\,1)[9x1] \rightarrow v[9x1] = inv(D^T\,D)[9x9]\ (D^T\,1)[9x1]}$$

接下来，使用未知数 **v[9x1]**构建辅助矩阵 $\mathbf{A_4[4x4]}$ 和 $\mathbf{A_3[3x3]}$，以及辅助向量 $\mathbf{v_{ghi}[3x1]}$：

$$\mathbf{v = [\,a, b, c, d, e, f, g, h, i\,]^T,} \qquad \mathbf{v_{ghi} = [g\ h\ i]^T}$$

$$\mathbf{A_4 = [\,a\ d\ e\ g;\ \ d\ b\ f\ h;\ \ e\ f\ c\ i;\ \ g\ h\ i\ {-1}\,],} \qquad \mathbf{A_3 = [\,a\ d\ e;\ d\ b\ f;\ e\ f\ c]}$$

偏移 **o = (ox, oy, oz)**可计算如下：

$$\mathbf{A_3[3x3]\ o[3x1] = -v_{ghi}[3x1] \rightarrow o[3x1] = -inv(A_3)[3x3]\ v_{ghi}[3x1]}$$

当知道了偏移时，就可以计算另一个辅助矩阵 $\mathbf{B_4[4x4]}$，它表示转换为原点的椭球体：

$$\mathbf{T = [\,1\ 0\ 0\ 0;\ \ 0\ 1\ 0\ 0;\ 0\ 0\ 1\ 0;\ ox\ oy\ oz\ 1\,] \rightarrow B_4[4x4] = T[4x4]\ A[4x4]\ T^T[4x4],}$$

$$\mathbf{B_4[4x4] = [\,b_{11}\ b_{12}\ b_{13}\ b_{14};\ b_{21}\ b_{22}\ b_{23}\ b_{24};\ b_{31}\ b_{32}\ b_{33}\ b_{34};\ b_{41}\ b_{42}\ b_{43}\ b_{44}\,],}$$

$$\mathbf{B_3[3x3] = [\,b_{11}\ b_{12}\ b_{13};\ b_{21}\ b_{22}\ b_{23};\ b_{31}\ b_{32}\ b_{33}\,]\ /\ {-b_{44}}}$$

增益和交叉轴增益可以分别根据 $\mathbf{B_3[3x3]}$的特征值和特征向量计算得出。

- 椭球半径是 3 个特征值的倒数的平方根；这些是轴增益 $\mathbf{g = [gx, gy, gz]^T}$

- 椭球旋转矩阵 **R[3x3]** 是通过并置 3 个特征向量得到的；乘以该 3x3 矩阵可以得到增益和交叉轴增益，其中对角线包含了增益值

补偿偏移、增益和交叉轴增益以便将数据点 $\mathbf{p = [x, y, z]^T}$ 映射到单位球面上，可以分 3 步完成：

1. 减去偏移量，$\mathbf{p' = p - o = [x\text{-}ox, y\text{-}oy, z\text{-}oz]^T = [\, x', y', z' \,]^T}$
2. 乘以旋转矩阵的逆，$\mathbf{p'' = p'\, inv(R) = [\, x'', y'', z'' \,]^T}$
3. 除以增益，$\mathbf{p''' = [\, x''/gx, y''/gy, z''/gz \,]^T = [x''', y''', z''' \,]^T}$

非旋转椭球拟合

这种情况下，数据矩阵 **D[Nx6]** 仅有 6 列，只有 6 个未知数要计算 **v = [a, b, c, g, h, i]**：

$$\mathbf{D[Nx6]\, v[6x1] = 1\, [Nx1] \rightarrow D^T[6xN]\, D[Nx6]\, v[6x1] = D^T[6xN]\, 1[Nx1] \rightarrow}$$

$$\mathbf{(D^T\, D)[6x6]\, v[6x1] = (D^T\, 1)[6x1] \rightarrow v[6x1] = inv(D^T\, D)[6x6]\, (D^T\, 1)[6x1]}$$

偏移 **o = (oX, oY, oZ)**可计算如下：

$$\mathbf{o = [\, a/g, b/h, c/i \,]^T}$$

增益 $\mathbf{g = [\, gx, gy, gz \,]^T}$ 可计算如下：

$$\mathbf{G = 1 + g^2/a + h^2/b + i^2/c \rightarrow g = [\, sqrt(a/G)\ sqrt(b/G)\ sqrt(c/G) \,]^T}$$

球体拟合

这种情况下，数据矩阵 **D[Nx4]** 仅有 4 列，只有 4 个未知数要计算 $\mathbf{v = [a+b+c, g, h, i\,]^T = [\, a'', g, h, i]^T}$：

$$\mathbf{D[Nx4]\, v[4x1] = 1\, [Nx1] \rightarrow D^T[4xN]\, D[Nx4]\, v[4x1] = D^T[4xN]\, 1[Nx1] \rightarrow}$$

$$\mathbf{(D^T\, D)[4x4]\, v[4x1] = (D^T\, 1)[4x1] \rightarrow v[4x1] = inv(D^T\, D)[4x4]\, (D^T\, 1)[4x1]}$$

偏移 **o = (oX, oY, oZ)**可计算如下：

$$\mathbf{o = [\, g/a'', h/a'', i/a'' \,]^T}$$

增益 $\mathbf{g = [\, gx, gy, gz \,]^T}$ 可计算如下：

$$\mathbf{G = 1 + g^2/a'' + h^2/a'' + i^2/a'' \rightarrow g = [\, sqrt(a''/G)\ sqrt(a''/G)\ sqrt(a''/G) \,]^T}$$

注释

微控制器上紧凑实时实现的提示：

- 只有乘积 $\mathbf{D^T[MxN]\ D[NxM]}$需要保存在存储器中，这是一个 MxM 矩阵，M=9, 6 或 4；最坏情况是 9x9=81 个元素要存储在存储器中

- 只有乘积 $\mathbf{D^T[MxN]\ 1[Nx1]}$需要保存在存储器中，这是一个 Mx1 向量，M=9, 6 或 4；最坏情况是 9 个元素要存储在存储器中

- 当已经采集了足够的数据点（至少 M 个）时，可以执行 Gaussian 消元法来计算前述 MxM 矩阵的逆

- 对于旋转椭球拟合的情况，可以通过使用闭合公式计算 3x3 矩阵的特征值和特征向量

- 对于无旋转或很少旋转的旋转椭球的情况，系统不容易收敛到正确的解；如果数据点受噪声影响，则尤其如此。如果预期很少或没有旋转（矩阵 R 在对角线外的值很小）和/或如果数据点受到显著噪声的影响，则建议使用以下备用方程系统：

  $$\mathbf{D[Nx9] = [\ X^2+Y^2-2Z^2,\ X^2-2Y^2+Z^2,\ 4XY, 2XZ,\ 2YZ,\ 2X,\ 2Y,\ 2Z,\ 1\ ]}$$

  $$\mathbf{E[Nx1] = [\ X^2+Y^2+Z^2\ ]}$$

  $$\mathbf{D[Nx9]\ u[9x1] = E[Nx1] \rightarrow D^T[9xN]\ D[Nx9]\ u[9x1] = D^T[9xN]\ E[Nx1] \rightarrow}$$

  $$\mathbf{(D^T\ D)[9x9]\ u[9x1] = (D^T\ 1)[9x1] \rightarrow u[9x1] = inv(D^T\ D)[9x9]\ (D^T\ E)[9x1]}$$

  $$\mathbf{S'[3x3] = [\ 3,\ 1,\ 1;\ 3,\ 1,\ -2;\ 3,\ -2,\ 1\ ]}$$

  $\mathbf{S[10x10] = [\ S'[3x3],\ 0[3x7];\ 0[7x3]\ eye[7x7]\ ]}$ 然后设 $\mathbf{s_{44} = 2}$

  $$\mathbf{v' = S[10x10]\ [\ -1/3;\ u[9x1]\ ] = [\ a',\ b',\ c',\ d',\ e',\ f',\ g',\ h',\ i',\ j'\ ]^T}$$

  $$\mathbf{v = -\ [\ a',\ b',\ c',\ d',\ e',\ f',\ g',\ h',\ i'\ ]^T / j' = [\ a,\ b,\ c,\ d,\ e,\ f,\ g,\ h,\ i\ ]^T}$$

  然后，对于旋转椭球的情况，计算如前所述进行。

用于椭球/球体拟合的 MatLab 代码

参考实现。

```
function [ofs,gain,rotM]=ellipsoid_fit(XYZ,varargin)
% 将（非）旋转椭球或球体拟合到一组 xyz 数据点
% XYZ:N(rows) x 3(cols)，N 个数据点的矩阵 (x,y,z)
% 可选标志 f，默认为 0（旋转椭球拟合）
x=XYZ(:,1); y=XYZ(:,2); z=XYZ(:,3); if nargin>1, f=varargin{1}; else f=0; end;
if     f==0, D=[x.*x,  y.*y,  z.*z,  2*x.*y,2*x.*z,2*y.*z, 2*x,2*y,2*z]; % 任意轴（旋转椭球）
elseif f==1, D=[x.*x,  y.*y,  z.*z,                        2*x,2*y,2*z]; % XYZ 轴（非旋转椭球）
elseif f==2, D=[x.*x+y.*y,    z.*z,                        2*x,2*y,2*z]; %  并且半径 x=y
elseif f==3, D=[x.*x+z.*z,    y.*y,                        2*x,2*y,2*z]; %  并且半径 x=z
elseif f==4, D=[y.*y+z.*z,    x.*x,                        2*x,2*y,2*z]; %  并且半径 y=z
elseif f==5, D=[x.*x+y.*y+z.*z,                            2*x,2*y,2*z]; %  并且半径 x=y=z （球体）
end;
v = (D'*D)\(D'*ones(length(x),1)); % 最小二乘拟合
if f==0, % 旋转椭球
```

```
  A = [ v(1) v(4) v(5) v(7); v(4) v(2) v(6) v(8); v(5) v(6) v(3) v(9); v(7) v(8) v(9) -1 ];
  ofs=-A(1:3,1:3)\[v(7);v(8);v(9)]; % 偏移是椭球的中心
  Tmtx=eye(4); Tmtx(4,1:3)=ofs'; AT=Tmtx*A*Tmtx'; % 椭球转化为(0,0,0)
  [rotM ev]=eig(AT(1:3,1:3)/-AT(4,4)); % 特征向量（旋转）和特征值（增益）
  gain=sqrt(1./diag(ev)); % 增益是椭球的半径
else % 非旋转椭球
  if    f==1, v = [ v(1) v(2) v(3) 0 0 0 v(4) v(5) v(6) ];
  elseif f==2, v = [ v(1) v(1) v(2) 0 0 0 v(3) v(4) v(5) ];
  elseif f==3, v = [ v(1) v(2) v(1) 0 0 0 v(3) v(4) v(5) ];
  elseif f==4, v = [ v(2) v(1) v(1) 0 0 0 v(3) v(4) v(5) ];
  elseif f==5, v = [ v(1) v(1) v(1) 0 0 0 v(2) v(3) v(4) ]; % 球体
  end;
  ofs=-(v(1:3).\v(7:9))'; % 偏移是椭球的中心
  rotM=eye(3); % 特征向量（旋转），特性 = 无旋转
  g=1+(v(7)^2/v(1)+v(8)^2/v(2)+v(9)^2/v(3));
  gain=(sqrt(g./v(1:3)))'; % 找到椭球的半径（比例）
end;
```

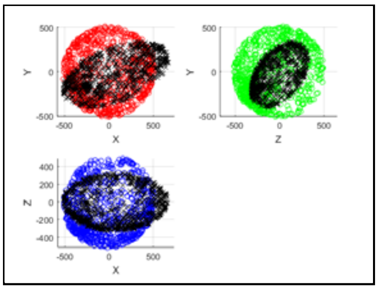
很少旋转或没有旋转的近球形数据的替代实现

```
function [ofs,gain,rotM]=ellipsoid_fit(XYZ)
% 将旋转椭球拟合到一组 xyz 数据点
% XYZ:N(rows) x 3(cols)，N 个数据点的矩阵 (x,y,z)
x=XYZ(:,1); y=XYZ(:,2); z=XYZ(:,3);
x2=x.*x; y2=y.*y; z2=z.*z;
D = [x2+y2-2*z2, x2-2*y2+z2, 4*x.*y, 2*x.*z, 2*y.*z, 2*x, 2*y, 2*z, ones(length(x),1)];
R = x2+y2+z2;
b = (D'*D)\(D'*R); % 最小二乘解
mtxref = [ 3  1  1   0  0  0   0  0  0  0; 3  1 -2   0  0  0   0  0  0  0; 3 -2  1   0  0  0   0  0  0  0; ...
           0  0  0   2  0  0   0  0  0  0; 0  0  0   0  1  0   0  0  0  0; 0  0  0   0  0  1   0  0  0  0; ...
           0  0  0   0  0  0   1  0  0  0; 0  0  0   0  0  0   0  1  0  0; 0  0  0   0  0  0   0  0  1  0; ...
           0  0  0   0  0  0   0  0  0  1];
v = mtxref*[-1/3; b]; nn=v(10); v = -v(1:9);
A = [ v(1) v(4) v(5) v(7); v(4) v(2) v(6) v(8); v(5) v(6) v(3) v(9); v(7) v(8) v(9) -nn ];
ofs=-A(1:3,1:3)\[v(7);v(8);v(9)]; % 偏移是椭球的中心
Tmtx=eye(4); Tmtx(4,1:3)=ofs'; AT=Tmtx*A*Tmtx'; % 椭球转化为(0,0,0)
[rotM ev]=eig(AT(1:3,1:3)/-AT(4,4)); % 特征向量（旋转）和特征值（增益）
gain=sqrt(1./diag(ev)); % 增益是椭球的半径
```

测试代码和示例输出

```
[ofs,gain,rotM]=ellipsoid_fit([X Y Z]);
XC=X-ofs(1); YC=Y-ofs(2); ZC=Z-ofs(3); % 转化为(0,0,0)
XYZC=[XC,YC,ZC]*rotM; % 旋转到 XYZ 轴
refr = 500; % 参考半径
XC=XYZC(:,1)/gain(1)*refr;
YC=XYZC(:,2)/gain(2)*refr;
ZC=XYZC(:,3)/gain(3)*refr; % 扩展到球体

figure;
subplot(2,2,1); hold on; plot(XC,YC,'ro'); plot(X,Y,'kx');
xlabel('X'); ylabel('Y'); axis equal; grid on;
subplot(2,2,2); hold on; plot(ZC,YC,'go'); plot(Z,Y,'kx');
xlabel('Z'); ylabel('Y'); axis equal; grid on;
subplot(2,2,3); hold on; plot(XC,ZC,'bo'); plot(X,Z,'kx');
xlabel('X'); ylabel('Z'); axis equal; grid on;
```



辅助资料

相关的设计支持材料
面向 STM32Cube 的 BlueMicrosystem1、蓝牙低功耗和传感器软件扩展
面向 STM32Cube 的 Open.MEMS、MotionFX、实时运动传感器数据融合软件扩展
文件

相关的设计支持材料
应用笔记，AN4508，low-g 3轴加速度计的参数和校准
应用笔记，AN4615，用于STM32 Nucleo的融合和罗盘校准API，使用 X-NUCLEO-IKS01A1 传感器扩展板
设计建议，DTxxxx，6 点翻滚传感器校准

版本历史

日期	版本	变更
2016 年 4 月 11 日	1	初始版本
2016 年 8 月 26 日	2	更新了方程式，添加了 Matlab 代码