佛山科学技术学院

Foshan University

本科生毕业论文

四轴飞行器遥感平台的实现方案

学　　院：环境与土木建筑学院

专　　业：地理信息系统

学　　号：2009764134

学生姓名：褚永俊

指导教师：伍兆强（讲师）

（职称）

二〇一三年六月

摘　要

四轴飞行器作为低空低成本的遥感平台，在各个领域应用广泛。相比其他类型的飞行器，四轴飞行器硬件结构简单紧凑，而软件复杂。本文介绍四轴飞行器的一个实现方案，重点讲软件算法，包括加速度计校正、姿态计算和姿态控制三部分。校正加速度计采用最小二乘法。计算姿态采用姿态插值法、梯度下降法或互补滤波法，需要对比这三种方法然后选出一种来应用。控制姿态采用欧拉角控制或四元数控制。最后比较各种方法的效果，并附上 C 语言的算法实现代码。

关键词：四轴飞行器；姿态；控制

One Method to Make a Quadcopter

XUAN Yong-jun

Abstract

Quadcopter is a low-cost low-altitude remote sensing platforms, which widely used in various fields. Compared to other types of aircraft, the quadcopter has simpler hardware, result in more complex software. This paper describes an implementation of the quadcopter, focusing on software algorithms, including the calibration of accelerometer, estimation of attitude, and control of attitude. It uses the Least Squares Method to calibration the accelerometer. It uses Attitude Interpolation Method, Gradient Descent Method or Complementary Filter Method to estimation the attitude. It uses Euler angles or quaternion to control the attitude. Finally, there are comparisons of the methods. The C-language implementation of the methods is appended.

Key words: quadcopter; attitude; control

目　录

1. 引言

四轴飞行器最开始是由军方研发的一种新式飞行器[1]。随着 MEMS 传感器、单片机、电机和电池技术的发展和普及，四轴飞行器成为航模界的新锐力量。到今天，四轴飞行器已经应用到各个领域，如军事打击、公安追捕、灾害搜救、农林业调查、输电线巡查、广告宣传航拍、航模玩具等，已经成为重要的遥感平台[2] [3] [4] [5] [6] [7]。

以农业调查为例，传统的调查方式为到现场抽样调查或用航空航天遥感。抽样的方式工作量大，而且准确性受主观因素影响；而遥感的方式可以大范围同时调查，时效性和准确性都有保证，但只能得到大型作物的宏观的指标，而且成本很高。不连续的地块、小种作物等很难用上遥感调查。因此，低空低成本遥感技术显得相当重要，而四轴飞行器正符合低空低成本遥感平台的要求。

目前应用广泛的飞行器有：固定翼飞行器和单轴的直升机。与固定翼飞行器相比，四轴飞行器机动性好，动作灵活，可以垂直起飞降落和悬停，缺点是续航时间短得多、飞行速度不快；而与单轴直升机比，四轴飞行器的机械简单，无需尾桨抵消反力矩，成本低[8]。

本文就小型电动四轴飞行器，介绍四轴飞行器的一种实现方案，重点讲解四轴飞行器的原理和用到的算法，并对几种姿态算法进行比较。

2. 飞行器的构成

四轴飞行器的实现可以分为硬件和软件两部分。比起其他类型的飞行器，四轴飞行器的硬件比较简单，而把系统的复杂性转移到软件上，所以本文的主要内容是软件的实现，特别是算法、公式的推导。

2.1. 硬件构成

飞行器由机架、电机、螺旋桨和控制电路构成。

2.1.1. 机械构成

机架呈十字状，是固定其他部件的平台，本项目采用的是尼龙材料的机架。电机采用无刷直流电机，固定在机架的四个端点上，而螺旋桨固定在电机转子上，迎风面垂直向下。螺旋桨按旋转方向分正桨和反桨，从迎风面看逆时针转的为正桨，四个桨的中心连成的正方形，正桨反桨交错安装。整体如图 2-1。

图 2-1 四轴飞行器整机

2.1.2. 电气构成

电气部分包括：控制电路板、电子调速器、电池，和一些外接的通讯、传感器模块。控制电路板是电气部分的核心，上面包含 MCU、陀螺仪、加速度计、电子罗盘、气压计等芯片，负责计算姿态、处理通信命令和输出控制信号到电子调速器。电子调速器简称电调，用于控制无刷直流电机。电气连接如图 2-2 所示。

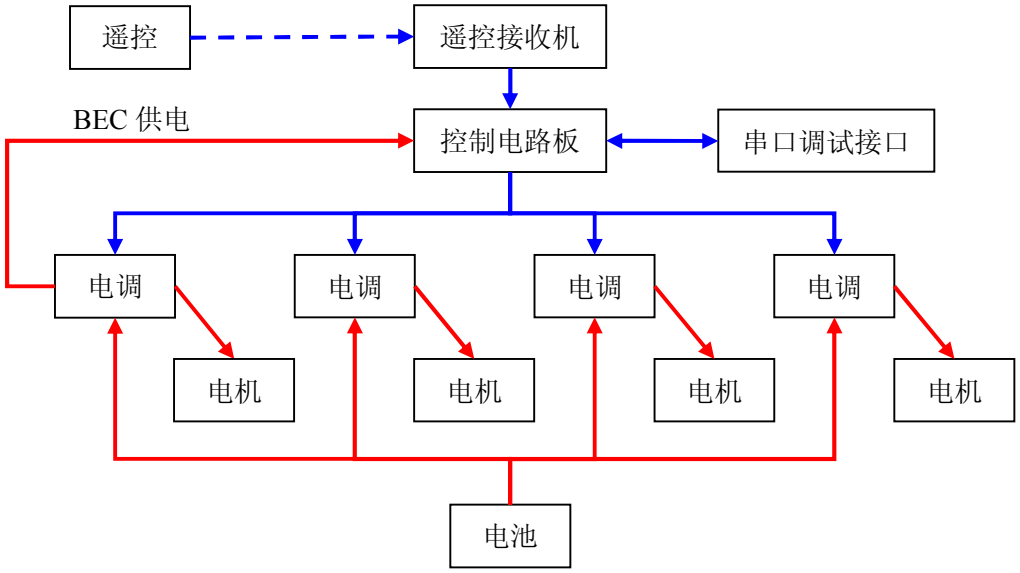

图 2-2 四轴飞行器电气连接图

硬件清单如表 2-1。

表 2-1 四轴飞行器硬件清单

器件	型号	主要参数
机架	风火轮 Z450	桨距 0.45m，尼龙材料，重量 241g。
电机	新西达 A2212	13 极，1000KV。
螺旋桨	1045	直径 10 英寸，桨叶角 45°。
电子调速器	新西达 HW-30A	额定电流 30A。
电池	Lion Power	11.1V，2200mAh，30C，重量 179g。
MCU	STM32F405RGT6	主频 168MHz。
陀螺仪	MPU6050	量程±2000dps，16 位分辨率。
加速度计	MPU6050	量程±8G，16 位分辨率。
电子罗盘	HMC5883	量程±8Gass，12 位分辨率。
气压计	BMP085	0.25m 分辨率。
遥控及其接收机	天地飞 WFT06X-A2.4G	2.4GHz 信号，5 比例通道+1 开关通道。

2.2. 软件构成

2.2.1. 上位机

上位机是针对飞行器的需要，在 Qt SDK 上写的一个桌面程序，可以通过串口与飞行器相连，具备传感器校正、显示姿态、测试电机、查看电量、设置参数等功能，主界面如图(2-3)。

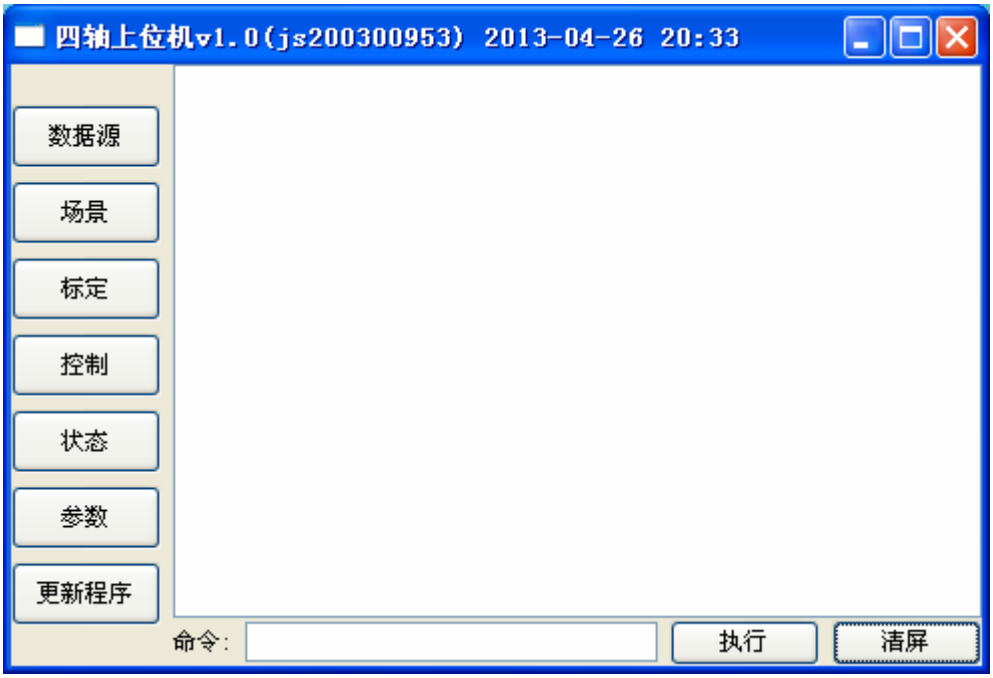

图 2-3 上位机主界面

2.2.2. 下位机

下位机为飞行器上 MCU 里的程序，主要有三个任务：计算姿态、接受命令和输出控制。下位机直接控制电机功率，飞行器的安全性、稳定性、可操纵性都取决于它。下位机的三个任务实时性都

要求很高，所以计算姿态的频率设为 200Hz，输出控制的频率为 100Hz，而接收到命令后，立即处理。因为电子调速器接受的信号为 PWM 信号，高电平时间在 1ms~2ms 之间，所以控制信号输出频率也不能太高。

3. 飞行原理

3.1. 坐标系统

飞行器涉及两个空间直角坐标系统：地理坐标系和机体坐标系。地理坐标系是固连在地面的坐标系，机体坐标系是固连在飞行器上的坐标系。四轴飞行器运动范围小，可以不考虑地面曲率，且假设地面为惯性系。地理坐标系采用“东北天坐标系”，X 轴指向东，为方便罗盘的使用，Y 轴指向地磁北，Z 轴指向天顶。机体坐标系原点在飞行器中心，xy 平面为电机所在平面，电机分布在{|x|=|y|,z=0}的直线上，第一象限的电机带正桨，z 轴指向飞行器上方。如图 3-1 所示。

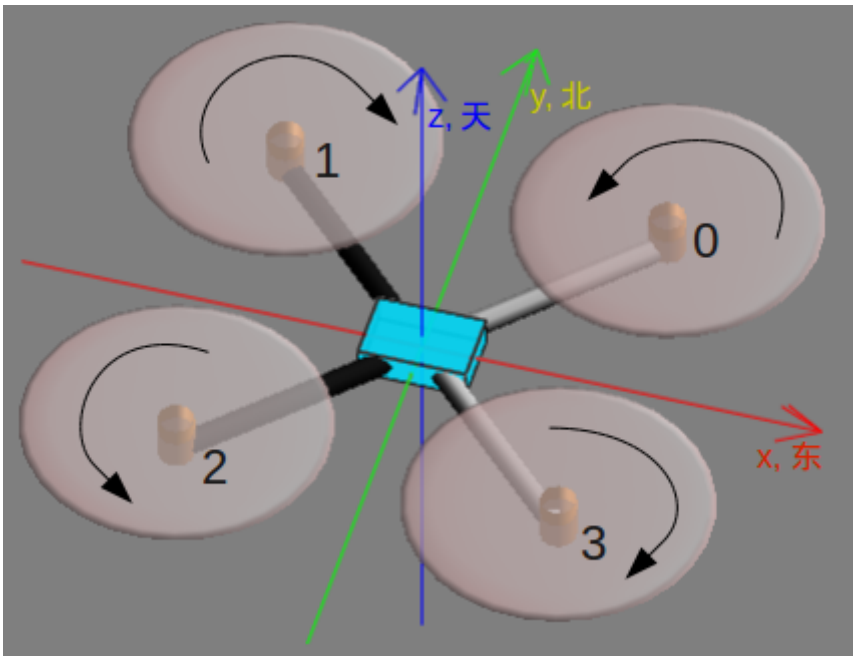

图 3-1 地理坐标系和机体坐标系图示（坐标系方向重合时）

3.2. 姿态的表示和运算

飞行器的姿态，是指飞行器的指向，一般用三个姿态角表示，包括偏航角(yaw)、俯仰角(pitch)和滚转角(roll)。更深一层，姿态其实是一个旋转变换，表示机体坐标系与地理坐标系的旋转关系，这里定义姿态为机体坐标系向地理坐标系的转换。旋转变换有多种表示方式，包括变换矩阵、姿态角、转轴转角、四元数等。

在本文中，矩阵用加粗大写字母表示，如 ${}_{A}^{E}\mathbf{R}$，左上标和左下标表示从机体坐标系（Aircraft）变换到地理坐标系（Earth）；四元数用加粗小写字母表示，如 ${}_{A}^{E}\mathbf{q}$，上下标意义与变换矩阵一样；向量用带箭头加粗小写字母表示，如 ${}^{A}\vec{\mathbf{v}}$，左上标 A 表示向量的值是在机体坐标系的坐标值。

因为姿态实质是一个旋转变换，根据刚体有限转动的欧拉定理，旋转变换是可以串联的，所以一个姿态可以经过一个旋转变换，变成另一个姿态。类比点和向量的概念，姿态相当于点，旋转相当于向量，点可以通过加向量，变成另一个点。如果用矩阵表示旋转，旋转的串联由矩阵乘法来实现。如果用四元数表示旋转，则由四元数的乘法来实现旋转串联。

用四元数来表示旋转，组合旋转时比用其他方法运算量更少，所以无论在计算机图形学、飞行

器控制等涉及刚体旋转的领域，四元数都有举足轻重的地位[9]。飞行器的姿态计算是围绕姿态四元数进行的，下面简要介绍一下四元数的运算。

1 个四元数由 4 个实数组成。

$$\mathbf{q}=\begin{bmatrix} w_q & x_q & y_q & z_q \end{bmatrix}^{\mathrm{T}} \tag{3-1}$$

规范化的四元数可以表示旋转，见(3-2)式，θ 为旋转的转角，单位向量 $\begin{bmatrix} x_\omega & y_\omega & z_\omega \end{bmatrix}^{\mathrm{T}}$ 为旋转的转轴。

$$\begin{cases} w_{\mathbf{q}}=\cos\left(\dfrac{\theta}{2}\right) \\ x_{\mathbf{q}}=x_\omega\cdot\sin\left(\dfrac{\theta}{2}\right) \\ y_{\mathbf{q}}=y_\omega\cdot\sin\left(\dfrac{\theta}{2}\right) \\ z_{\mathbf{q}}=z_\omega\cdot\sin\left(\dfrac{\theta}{2}\right) \end{cases} \tag{3-2}$$

记四元数乘法的符号为 $\otimes$。四元数乘法跟矩阵一样，有结合律，没有交换律。运算过程见(3-3)式。

$$\mathbf{r}=\mathbf{p}\otimes\mathbf{q}\Leftrightarrow\begin{cases} w_{\mathbf{r}}=w_{\mathbf{p}}w_{\mathbf{q}}-x_{\mathbf{p}}x_{\mathbf{q}}-y_{\mathbf{p}}y_{\mathbf{q}}-z_{\mathbf{p}}z_{\mathbf{q}} \\ x_{\mathbf{r}}=w_{\mathbf{p}}x_{\mathbf{q}}+x_{\mathbf{p}}w_{\mathbf{q}}+y_{\mathbf{p}}z_{\mathbf{q}}-z_{\mathbf{p}}y_{\mathbf{q}} \\ y_{\mathbf{r}}=w_{\mathbf{p}}y_{\mathbf{q}}-x_{\mathbf{p}}z_{\mathbf{q}}+y_{\mathbf{p}}w_{\mathbf{q}}+z_{\mathbf{p}}x_{\mathbf{q}} \\ z_{\mathbf{r}}=w_{\mathbf{p}}z_{\mathbf{q}}+x_{\mathbf{p}}y_{\mathbf{q}}-y_{\mathbf{p}}x_{\mathbf{q}}+z_{\mathbf{p}}w_{\mathbf{q}} \end{cases} \tag{3-3}$$

四元数转成矩阵的函数记为 $\mathrm{R}(\ )$，具体过程见(3-4)式。

$$\mathrm{R}(\mathbf{q})=\begin{bmatrix} 1-2y_{\mathbf{q}}^2-2z_{\mathbf{q}}^2 & 2x_{\mathbf{q}}y_{\mathbf{q}}-2w_{\mathbf{q}}z_{\mathbf{q}} & 2x_{\mathbf{q}}z_{\mathbf{q}}+2w_{\mathbf{q}}y_{\mathbf{q}} \\ 2x_{\mathbf{q}}y_{\mathbf{q}}+2w_{\mathbf{q}}z_{\mathbf{q}} & 1-2x_{\mathbf{q}}^2-2z_{\mathbf{q}}^2 & 2y_{\mathbf{q}}z_{\mathbf{q}}-2w_{\mathbf{q}}x_{\mathbf{q}} \\ 2x_{\mathbf{q}}z_{\mathbf{q}}-2w_{\mathbf{q}}y_{\mathbf{q}} & 2y_{\mathbf{q}}z_{\mathbf{q}}+2w_{\mathbf{q}}x_{\mathbf{q}} & 1-2x_{\mathbf{q}}^2-2y_{\mathbf{q}}^2 \end{bmatrix} \tag{3-4}$$

3.3. 动力学原理

螺旋桨旋转时，把空气对螺旋桨的压力在轴向和侧向两个方向分解，得到两种力学效应：推力和转矩。当四轴飞行器悬停时，合外力为 0，螺旋桨的推力用于抵消重力，转矩则由成对的正桨反桨抵消。当飞行器运动时，因为推力只能沿轴向，所以只能通过倾斜姿态来提供水平的动力，控制运动由控制姿态来间接实现。

假设四轴为刚体，根据质点系动量矩定理，角速度和角加速度由外力矩决定[10]，通过控制四个螺旋桨，可以产生需要的力矩。首先对螺旋桨编号：第一象限的为 0 号，然后逆时针依次递增，如图(3-1)。同步增加 0 号和 1 号、减小 2 号和 3 号桨的功率，可以在不改变推力的情况下，提供 x 轴的力矩；同步增加 1 号和 2 号、减小 0 号和 3 号桨的功率，可以在不改变推力的情况下，提供 y 轴的力矩；同步增加 1 号和 3 号、减小 0 号和 2 号桨的功率，可以在不改变推力的情况下，提供 z 轴的力矩。以上“增加”和“减小”只是表明变化的方向，可以增加负数和减小负数，提供的力矩就沿对应轴的负方向了。把三个轴的力矩叠加起来，就得到各螺旋桨功率变化与提供的力矩的对应关系，可以用一个矩阵等式表示，见(3-5)式。$\Delta\mathbf{T}$ 是螺旋桨的功率变化量，为 4×1 矩阵，每行分别对

应 0 到 3 号螺旋桨；$\vec{\mathbf{m}}$ 是力矩，为 3×1 矩阵。m_x、m_y 和 m_z 是各轴的力矩系数，用于把力矩转换成功率变化量，具体数值与电机力矩特性、电机安装位置等有关。

$$\Delta\mathbf{T}=\begin{bmatrix}1 & -1 & -1\\ 1 & 1 & 1\\ -1 & 1 & -1\\ -1 & -1 & 1\end{bmatrix}\cdot\mathrm{diag}\left(\mathrm{m}_x,\mathrm{m}_y,\mathrm{m}_z\right)\cdot\vec{\mathbf{m}} \tag{3-5}$$

各个电机实际输出的功率记为 $\mathbf{T}_{output}$，推力油门对应的功率量为 $\mathbf{T}_{base}$，则有：

$$\mathbf{T}_{output}=\mathbf{T}_{base}+\Delta\mathbf{T} \tag{3-6}$$

4. 姿态测量

获取当前姿态是控制飞行器平稳飞行的基础，姿态的测量要求低噪声、高输出频率，当采用陀螺仪等需要积分的传感器时，还需要考虑积分发散等问题。近年来 MEMS 传感器越来越成熟、应用广泛，成为低成本姿态测量的首选器件[11]，因此该项目使用的传感器全部都是 MEMS 传感器，见表(2-1)。在使用传感器的值进行姿态计算之前，有必要校正传感器

4.1. 传感器校正

由于实验条件限制，传感器的校正只有两项，分别对应两种类型的传感器：陀螺仪——静止时 0 输出的传感器、加速度计与罗盘——测量某向量场强度的传感器。

4.1.1. 陀螺仪

对于陀螺仪等静止时 0 输出的传感器，可以很方便地校正零偏。把传感器固定好，这时对输出值 $\mathbf{X}_f$ 求平均，得到的 $\mathbf{A}$ 即为零偏，如(4-1)式。实际使用时，把测得的值减去零偏，得到的值就是校正值。实际应用的公式如(4-2)，$\mathbf{A}$ 为零偏值，3×1 矩阵，单位：LSB；$\mathbf{Y}_i$ 为校正好的值，3×1 矩阵，单位：rad/s；$\mathbf{X}_i$ 为测量原始值，单位：LSB； gain 为转换系数，单位：(rad/s)/LSB，由传感器的数据手册给出。

$$\mathbf{A}=\frac{1}{n}\sum_n\mathbf{X}_f \tag{4-1}$$

$$\mathbf{Y}_i=\left(\mathbf{X}_i-\mathbf{A}\right)\cdot\mathrm{gain} \tag{4-2}$$

4.1.2. 加速度计和电子罗盘

加速度计和罗盘都是测量所在点的某个向量场的值的传感器，静态时加速度计测的是等效重力加速度场，电子罗盘测的是地磁场。下面仅介绍加速度计的校正，罗盘的校正同理。

加速度计测量的对象是比力，也就是等效重力加速度和运动加速度的和，当静止时，运动加速度为 0，加速度计的测量值为等效重力加速度，可以利用这一点校正加速度计。加速度计的校正的思路为：对测量值平移和缩放，把测量值拟合到重力加速度。因此校正的任务为：寻找最佳的平移和缩放参数，使总体测量数据的更靠近重力加速度。

记测量值为 $\begin{bmatrix}x_m & y_m & z_m\end{bmatrix}^{\mathrm{T}}$，校正后的值为 $\begin{bmatrix}x_c & y_c & z_c\end{bmatrix}^{\mathrm{T}}$，平移参数为 $\begin{bmatrix}o_x & o_y & o_z\end{bmatrix}^{\mathrm{T}}$，缩放参数为 $\begin{bmatrix}g_x & g_y & g_z\end{bmatrix}^{\mathrm{T}}$，他们之间的关系为(4-3)式。

$$\begin{cases} x_c = (x_m + o_x) \cdot g_x \\ y_c = (y_m + o_y) \cdot g_y \\ z_c = (z_m + o_z) \cdot g_z \end{cases} \tag{4-3}$$

定义误差 u 为测量值长度与重力加速度常数 G 的平方差。

$$u = x_c^2 + y_c^2 + z_c^2 - G^2 \tag{4-4}$$

把(4-3)式代入(4-4)式，得：

$$u = g_x^2 x_m^2 + o_x^2 g_x^2 + 2x_m o_x g_x^2 + g_y^2 y_m^2 + o_y^2 g_y^2 + 2y_m o_y g_y^2 + g_z^2 z_m^2 + o_z^2 g_z^2 + 2z_m o_z g_z^2 - G^2 \tag{4-5}$$

记

$$\mathbf{V} = \begin{bmatrix} x_m^2 & y_m^2 & z_m^2 & x_m & y_m & z_m & 1 \end{bmatrix}^{\mathrm{T}} \tag{4-6}$$

$$\mathbf{P} = \begin{bmatrix} a & b & c & d & e & f & g \end{bmatrix}^{\mathbf{T}} \tag{4-7}$$

$$\begin{cases} a = g_x^2 \\ b = g_y^2 \\ c = g_z^2 \\ d = 2o_x g_x^2 \\ e = 2o_y g_y^2 \\ f = 2o_z g_z^2 \\ g = o_x^2 g_x^2 + o_y^2 g_y^2 + o_z^2 g_z^2 - G^2 \end{cases} \tag{4-8}$$

则 u 可以表示成

$$u = ax_m^2 + by_m^2 + cz_m^2 + dx_m + ey_m + fz_m + g = \mathbf{V}^{\mathrm{T}} \times \mathbf{P} \tag{4-9}$$

设目标函数 U，用来衡量整体误差，这里用单个误差的平方和。

$$\mathrm{U} = \sum u^2 \tag{4-10}$$

校正的任务具体为：寻找参数$\{a,b,c,d,e,f,g\}$，使 U 最小。因为 U 是$\{a,b,c,d,e,f,g\}$的多项式函数，使 U 最小的点必然为极值点，一阶偏导为 0，得(4-11)式。

$$\begin{cases}\dfrac{\partial \mathrm{U}}{\partial a}=\sum 2u\dfrac{\partial u}{\partial a}=\sum 2ux_m^2=0\\ \dfrac{\partial \mathrm{U}}{\partial b}=\sum 2u\dfrac{\partial u}{\partial b}=\sum 2uy_m^2=0\\ \dfrac{\partial \mathrm{U}}{\partial c}=\sum 2u\dfrac{\partial u}{\partial c}=\sum 2uz_m^2=0\\ \dfrac{\partial \mathrm{U}}{\partial d}=\sum 2u\dfrac{\partial u}{\partial d}=\sum 2ux_m=0\\ \dfrac{\partial \mathrm{U}}{\partial e}=\sum 2u\dfrac{\partial u}{\partial e}=\sum 2uy_m=0\\ \dfrac{\partial \mathrm{U}}{\partial f}=\sum 2u\dfrac{\partial u}{\partial f}=\sum 2uz_m=0\\ \dfrac{\partial \mathrm{U}}{\partial g}=\sum 2u\dfrac{\partial u}{\partial g}=\sum 2u=0\end{cases} \tag{4-11}$$

记

$$\mathbf{B}=\sum \mathbf{V}\times\mathbf{V}^{\mathrm{T}} \tag{4-12}$$

由(4-6)、(4-7)和(4-11)式：

$$\sum \mathbf{V}\times u=\sum \mathbf{V}\times\mathbf{V}^{\mathrm{T}}\times\mathbf{P}=(\sum \mathbf{V}\times\mathbf{V}^{\mathrm{T}})\times\mathbf{P}=\mathbf{B}\times\mathbf{P}=\mathbf{0} \tag{4-13}$$

为了求出 $\mathbf{P}$，需要解式(4-13)的齐次线性方程组。由于数据噪声、运算精度限制等原因，方程(4-13)极少会出现非零解，所以转向求近似解，解法为经典的高斯消元法。为了减小误差，行的处理顺序有讲究，先处理待消变量系数最大的行。当处理到最后一个变量时，系数已经变得很小了，强制令为 0，以释放一个自由度，这样就有非零解了。解出的解带一个任意常数，设为 C，$\mathbf{P}_b$ 为某基础解系，见(4-14)式。

$$\mathbf{P}=C\cdot\mathbf{P}_b=\begin{bmatrix}C\cdot a_b & C\cdot b_b & C\cdot c_b & C\cdot d_b & C\cdot e_b & C\cdot f_b & C\cdot g_b\end{bmatrix}^{\mathrm{T}} \tag{4-14}$$

由(4-7)、(4-8)和(4-14)得到 6 个校正参数：

$$\begin{cases}o_x=\dfrac{d_b}{2a_b}\\ o_y=\dfrac{e_b}{2b_b}\\ o_z=\dfrac{f_b}{2c_b}\\ g_x=\sqrt{C\cdot a_b}\\ g_y=\sqrt{C\cdot b_b}\\ g_z=\sqrt{C\cdot c_b}\\ C=\dfrac{4G^2}{\dfrac{d_b^2}{a_b}+\dfrac{e_b^2}{b_b}+\dfrac{f_b^2}{c_b}-g_b}\end{cases} \tag{4-15}$$

所以校正加速度计的整体流程为：测量一批静态数据，要尽可能在球面上分布均匀，然后用这批数据根据(4-12)式生成方阵 $\mathbf{B}$，然后求方程(4-13)的近似解 $\mathbf{P}$，再代入式(4-15)，得到校正参数。如果遇到 a_b、b_b、c_b 或 C 小于 0 的情况，说明采集的数据不够典型，应该重新采集。

4.2. 数据融合

4.2.1. 概述

有了传感器的数据，就可以用来计算姿态了。计算姿态主要用到 3 个传感器：陀螺仪、加速度计、电子罗盘。加速度计测量对象为比力，受运动加速度影响大，特别是受飞行器机架的振动的影响，振动振幅高达 2G。电子罗盘测磁场强度，受环境影响也很大，特别是受电子调速器、电机等大电流器件影响。而陀螺仪则受外部影响弱，稳定性好，但输出量为角速度，需要积分才能得到姿态，无法避免误差的累积。为了得到稳定的、近实时的姿态，对各传感器的数据取长补短，需要研究各种数据融合方法。

4.2.2. 姿态插值法

首先介绍自己构思的“姿态插值法”，过程分成两部分：一、如果已知初始姿态，可以利用陀螺积分，不断推算下一个姿态，这部分动态性能好，但误差会累积；二、利用加速度计和罗盘，可以直接算出一个姿态，这个姿态的期望是正确的，但正如上文所述，这个姿态噪声大，不稳定。为了把两个系统的优点结合起来，对两个系统的结果进行插值，得到的值作为当前的姿态。

下面讲“姿态插值法”的计算过程。

首先是第一个部分——陀螺积分算姿态。陀螺的输出为时间离散的角速度，要对时间积分才得到角度。记陀螺输出的角速度为 $\vec{\mathbf{g}}=\begin{bmatrix} x_{\vec{g}} & y_{\vec{g}} & z_{\vec{g}} \end{bmatrix}^{\mathrm{T}}$,单位为 rad/s，采样间隔为 Δt。假定采样间隔足够短，在一个采样时间间隔里，角速度不变，且转角足够小，忽略三角函数高阶项，各轴间相互的影响忽略不计，根据(3-2)式，该间隔里的旋转可以用四元数表示成

$$ {}^{A_{n-1}}_{A_n}\mathbf{r}_n=\begin{bmatrix} \sqrt{1-\dfrac{\Delta t^2}{4}\left(x_{\vec{g}}^2+y_{\vec{g}}^2+z_{\vec{g}}^2\right)} & \dfrac{\Delta t\cdot x_{\vec{g}}}{2} & \dfrac{\Delta t\cdot y_{\vec{g}}}{2} & \dfrac{\Delta t\cdot z_{\vec{g}}}{2} \end{bmatrix}^{\mathrm{T}} \tag{4-16} $$

更新前的姿态为 ${}^{E}_{A_{n-1}}\mathbf{q}$，${}^{A_{n-1}}_{A_n}\mathbf{r}_n$ 为在 ${}^{E}_{A_{n-1}}\mathbf{q}$ 的基础上的旋转，则把他们相乘，即得到新的当前的姿态。

$$ {}^{E}_{A_n}\mathbf{q}_{gyr}={}^{E}_{A_{n-1}}\mathbf{q}\otimes{}^{A_{n-1}}_{A_n}\mathbf{r}_n \tag{4-17} $$

接着讲第二个部分。姿态的定义为两个坐标系统的旋转变换，因此只要知道两对在两个坐标系对应的向量，就可以把姿态求出来。两对变量为加速度和磁场强度，测量出的加速度和磁场强度是在机体坐标系的，而地理坐标系的加速度和磁场是常量，存在一个旋转，可以把机体坐标系的加速度和磁场强度转换到与地理坐标系的对应的常量重合，这个旋转就是所求的姿态。由于传感器系统误差、噪声等影响，测量出的加速度和磁场的夹角不是恒定值，不可能精确旋转到与常量一样，只能求最接近的旋转。最接近的原则可定为：一、旋转后四个向量共面；二、加速度和磁场成一定角度，旋转后的角平分线与常量的角平分线重合。

为了达到最接近原则，把处理的量由加速度和磁场强度，转成它们的平面法线和角平分线。记加速度计输出为 ${}^{A}\vec{\mathbf{a}}_m$，罗盘输出为 ${}^{A}\vec{\mathbf{h}}_m$，规范化的等效重力加速度为 ${}^{E}\mathbf{a}_c=\begin{bmatrix}0 & 0 & 1\end{bmatrix}^{\mathrm{T}}$，规范化的

地磁场强度为 ${}^{E}\vec{\mathbf{h}}_c = \begin{bmatrix} 0 & y_{E_{\vec{\mathbf{h}}}} & z_{E_{\vec{\mathbf{h}}}} \end{bmatrix}$。

测量的加速度和磁场强度的平面法线向量和对角线向量为

$$ {}^{A}\vec{\mathbf{c}}_m = \frac{{}^{A}\vec{\mathbf{a}}_m}{\left|{}^{A}\vec{\mathbf{a}}_m\right|} \times \frac{{}^{A}\vec{\mathbf{h}}_m}{\left|{}^{A}\vec{\mathbf{h}}_m\right|} \tag{4-18} $$

$$ {}^{A}\vec{\mathbf{d}}_m = \frac{{}^{A}\vec{\mathbf{a}}_m}{\left|{}^{A}\vec{\mathbf{a}}_m\right|} + \frac{{}^{A}\vec{\mathbf{h}}_m}{\left|{}^{A}\vec{\mathbf{h}}_m\right|} \tag{4-19} $$

等效重力加速度常量和地磁场强度常量的平面法线向量和对角线为：

$$ {}^{E}\vec{\mathbf{c}}_c = {}^{E}\vec{\mathbf{a}}_c \times {}^{E}\vec{\mathbf{h}}_c \tag{4-20} $$

$$ {}^{E}\vec{\mathbf{d}}_c = {}^{E}\vec{\mathbf{a}}_c + {}^{E}\vec{\mathbf{h}}_c \tag{4-21} $$

定义一个函数，用于把旋转轴和转角转换成四元数：

$$ \mathbf{r} = \text{quaternion}\left(\vec{\omega}, \theta\right), \begin{cases} w_r = \cos\left(\dfrac{\theta}{2}\right) \\ x_r = \dfrac{x_{\vec{\omega}}}{\left|\vec{\omega}\right|}\sin\left(\dfrac{\theta}{2}\right) \\ y_r = \dfrac{y_{\vec{\omega}}}{\left|\vec{\omega}\right|}\sin\left(\dfrac{\theta}{2}\right) \\ z_r = \dfrac{z_{\vec{\omega}}}{\left|\vec{\omega}\right|}\sin\left(\dfrac{\theta}{2}\right) \end{cases} \tag{4-22} $$

再定义一个函数，用于获取把一个向量旋转到另一个向量的旋转四元数：

$$ \mathbf{r} = \text{rotate}\left(\vec{\mathbf{f}}, \vec{\mathbf{t}}\right) = \text{quaternion}\left(\vec{\mathbf{f}} \times \vec{\mathbf{t}}, \arctan\left(\left|\vec{\mathbf{f}} \times \vec{\mathbf{t}}\right|, \vec{\mathbf{f}} \cdot \vec{\mathbf{t}}\right)\right) \tag{4-23} $$

第一次旋转，把平面法线旋转到重合：

$$ \mathbf{q}_1 = \text{rotate}\left({}^{A}\vec{\mathbf{c}}_m, {}^{E}\vec{\mathbf{c}}_c\right) \tag{4-24} $$

经过第一次旋转，四个向量已经共面，测量的加速度和磁场强度的对角线向量变成：

$$ {}^{A}\vec{\mathbf{d}}_{\mathrm{m}}{}' = \mathrm{R}\left(\mathbf{q}_1\right) \cdot {}^{A}\vec{\mathbf{d}}_{\mathrm{m}} \tag{4-25} $$

再进行第二次旋转，把对角线重叠在一起：

$$ \mathbf{q}_2 = \text{rotate}\left({}^{A}\vec{\mathbf{d}}_m{}', {}^{E}\vec{\mathbf{d}}_c\right) \tag{4-26} $$

把两次旋转组合起来，就是第二部分所求的姿态了。

$$ {}^{E}_{A}\mathbf{q}_{acc\&mag} = \mathbf{q}_2 \otimes \mathbf{q}_1 \tag{4-27} $$

把两个部分算出的姿态进行插值，就可以得到当前的姿态。因为两个部分算出的姿态相差小，用普通线性插值即可。α 为插值系数，范围在[0,1]，越接近 0，第一部分的姿态占的权重就越大，一般取接近 0 的值。

$$ {}^{E}_{A_n}\mathbf{q} = \frac{\mathbf{t}}{\left|\mathbf{t}\right|} \quad , \quad \mathbf{t} = \alpha \cdot {}^{E}_{A}\mathbf{q}_{acc\&mag} + \left(1-\alpha\right) \cdot {}^{E}_{A_n}\mathbf{q}_{gyr} \tag{4-28} $$

${}^{E}_{A_n}\mathbf{q}$ 即为输出的姿态。姿态插值法的流程如图 4-1。

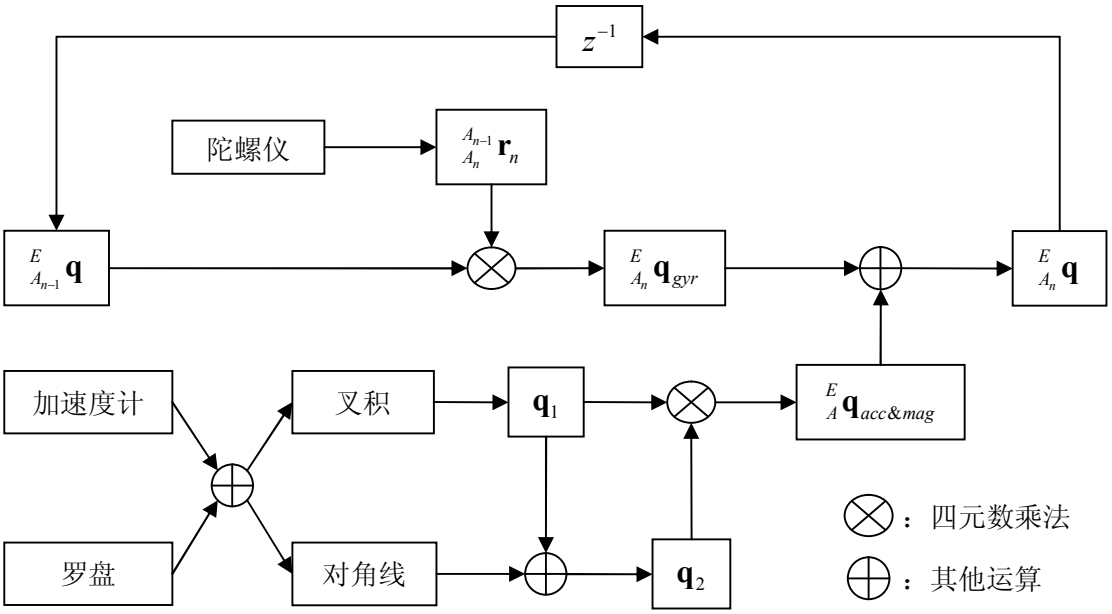

图 4-1 姿态插值法数据流程图

4.2.3. 梯度下降法

从实际的使用效果来说，姿态插值法已经满足要求，但用了三角函数，即使经过优化，运算量还是很大，不方便应用到低性能 MCU 或提高运算频率。Sebastian O.H. Madgwick 提出了更加有效的方法——用梯度下降法计算姿态[12]。其思路为：陀螺仪不断积分姿态；同时把加速度和磁场强度的误差表示成姿态的函数，令误差最小的点为极值点，然后用梯度下降法逼近误差最小的点，即纠正了陀螺的积分误差。这个方法，陀螺计算的姿态占主要，加速度计和罗盘只是辅助纠正。

用陀螺积分姿态的方法跟“姿态插值法”的第一部分一样，关键是如何用加速度和磁场强度纠正误差。首先要定义误差，最直观就是定义为测量值与常量值之差，这样加速度和磁场强度就分别有一个误差了。因为只考虑方向，相减前要规范化。为了减少运算量、利用常量值里的 0 和 1 分量，相减之前统一到机体坐标系，而不是地理坐标系。旋转矩阵的逆等于其转置矩阵。

$$\Delta\vec{\mathbf{a}} = \mathrm{R}^{-1}\left({}^{E}_{A}\mathbf{q}\right)\cdot{}^{E}\vec{\mathbf{a}}_c - \frac{{}^{A}\vec{\mathbf{a}}_m}{\left|{}^{A}\vec{\mathbf{a}}_m\right|}$$

$$= \begin{bmatrix} x_{\Delta\vec{\mathbf{a}}} \\ y_{\Delta\vec{\mathbf{a}}} \\ z_{\Delta\vec{\mathbf{a}}} \end{bmatrix} = \begin{bmatrix} 2x_{\mathbf{q}}z_{\mathbf{q}} - 2w_{\mathbf{q}}y_{\mathbf{q}} - \dfrac{x_{{}^{A}\vec{\mathbf{a}}_m}}{\left|{}^{A}\vec{\mathbf{a}}_m\right|} \\ 2y_{\mathbf{q}}z_{\mathbf{q}} + 2w_{\mathbf{q}}x_{\mathbf{q}} - \dfrac{y_{{}^{A}\vec{\mathbf{a}}_m}}{\left|{}^{A}\vec{\mathbf{a}}_m\right|} \\ 1 - 2x_{\mathbf{q}}^2 - 2y_{\mathbf{q}}^2 - \dfrac{z_{{}^{A}\vec{\mathbf{a}}_m}}{\left|{}^{A}\vec{\mathbf{a}}_m\right|} \end{bmatrix} \tag{4-29}$$

$$\Delta\vec{\mathbf{h}} = \mathrm{R}^{-1}\left({}^{E}_{A}\mathbf{q}\right)\cdot {}^{E}\vec{\mathbf{h}}_{c} - \frac{{}^{A}\vec{\mathbf{h}}_{m}}{\left|{}^{A}\vec{\mathbf{h}}_{m}\right|}$$

$$= \begin{bmatrix} x_{\Delta\vec{\mathbf{h}}} \\ y_{\Delta\vec{\mathbf{h}}} \\ z_{\Delta\vec{\mathbf{h}}} \end{bmatrix} = \begin{bmatrix} y_{E_{\vec{\mathbf{h}}}}\cdot\left(2x_{\mathbf{q}}z_{\mathbf{q}} - 2w_{\mathbf{q}}y_{\mathbf{q}}\right) + z_{E_{\vec{\mathbf{h}}}}\cdot\left(2x_{\mathbf{q}}y_{\mathbf{q}} + 2w_{\mathbf{q}}z_{\mathbf{q}}\right) - \dfrac{x_{A_{\vec{\mathbf{h}}_m}}}{\left|{}^{A}\vec{\mathbf{h}}_{m}\right|} \\ y_{E_{\vec{\mathbf{h}}}}\cdot\left(2y_{\mathbf{q}}z_{\mathbf{q}} + 2w_{\mathbf{q}}x_{\mathbf{q}}\right) + z_{E_{\vec{\mathbf{h}}}}\cdot\left(1 - 2x_{\mathbf{q}}^{2} - 2z_{\mathbf{q}}^{2}\right) - \dfrac{y_{A_{\vec{\mathbf{h}}_m}}}{\left|{}^{A}\vec{\mathbf{h}}_{m}\right|} \\ y_{E_{\vec{\mathbf{h}}}}\cdot\left(1 - 2x_{\mathbf{q}}^{2} - 2y_{\mathbf{q}}^{2}\right) + z_{E_{\vec{\mathbf{h}}}}\cdot\left(2y_{\mathbf{q}}z_{\mathbf{q}} - 2w_{\mathbf{q}}x_{\mathbf{q}}\right) - \dfrac{z_{A_{\vec{\mathbf{h}}_m}}}{\left|{}^{A}\vec{\mathbf{h}}_{m}\right|} \end{bmatrix} \tag{4-30}$$

加速度和磁场的误差都是向量，需要把他们合并，而且要符合误差最小的点在极小点。最终的误差定义为它们模的平方和。

$$F = \left|\Delta\vec{\mathbf{a}}\right|^{2} + \left|\Delta\vec{\mathbf{h}}\right|^{2} \tag{4-31}$$

有了误差 F，就对姿态求偏导。

$$\nabla F = \frac{\partial F}{\partial \mathbf{q}} = \begin{bmatrix} \dfrac{\partial F}{\partial w_{\mathbf{q}}} & \dfrac{\partial F}{\partial x_{\mathbf{q}}} & \dfrac{\partial F}{\partial y_{\mathbf{q}}} & \dfrac{\partial F}{\partial z_{\mathbf{q}}} \end{bmatrix}^{\mathrm{T}} \tag{4-32}$$

$$\begin{cases} \dfrac{\partial F}{\partial w_{\mathbf{q}}} = -4x_{\Delta\vec{\mathbf{a}}}y_{\mathbf{q}} + 4y_{\Delta\vec{\mathbf{a}}}x_{\mathbf{q}} + 4x_{\Delta\vec{\mathbf{h}}}\left(y_{E_{\vec{\mathbf{h}}}}z_{\mathbf{q}} - z_{E_{\vec{\mathbf{h}}}}y_{\mathbf{q}}\right) \\ \qquad\qquad + 4y_{\Delta\vec{\mathbf{h}}}z_{E_{\vec{\mathbf{h}}}}x_{\mathbf{q}} - 4z_{\Delta\vec{\mathbf{h}}}y_{E_{\vec{\mathbf{h}}}}x_{\mathbf{q}} \\ \dfrac{\partial F}{\partial x_{\mathbf{q}}} = 4x_{\Delta\vec{\mathbf{a}}}z_{\mathbf{q}} + 4y_{\Delta\vec{\mathbf{a}}}w_{\mathbf{q}} - 4z_{\Delta\vec{\mathbf{a}}}x_{\mathbf{q}} + 4x_{\Delta\vec{\mathbf{h}}}\left(y_{E_{\vec{\mathbf{h}}}}y_{\mathbf{q}} + z_{E_{\vec{\mathbf{h}}}}z_{\mathbf{q}}\right) \\ \qquad\qquad + 4y_{\Delta\vec{\mathbf{h}}}\left(z_{E_{\vec{\mathbf{h}}}}w_{\mathbf{q}} - y_{E_{\vec{\mathbf{h}}}}x_{\mathbf{q}}\right) - 4z_{\Delta\vec{\mathbf{h}}}\left(y_{E_{\vec{\mathbf{h}}}}w_{\mathbf{q}} + z_{E_{\vec{\mathbf{h}}}}x_{\mathbf{q}}\right) \\ \dfrac{\partial F}{\partial y_{\mathbf{q}}} = -4x_{\Delta\vec{\mathbf{a}}}w_{\mathbf{q}} + 4y_{\Delta\vec{\mathbf{a}}}z_{\mathbf{q}} - 4z_{\Delta\vec{\mathbf{a}}}y_{\mathbf{q}} + 4x_{\Delta\vec{\mathbf{h}}}\left(y_{E_{\vec{\mathbf{h}}}}x_{\mathbf{q}} - z_{E_{\vec{\mathbf{h}}}}w_{\mathbf{q}}\right) \\ \qquad\qquad + 4y_{\Delta\vec{\mathbf{h}}}z_{E_{\vec{\mathbf{h}}}}z_{\mathbf{q}} + 4z_{\Delta\vec{\mathbf{h}}}\left(y_{E_{\vec{\mathbf{h}}}}z_{\mathbf{q}} - z_{E_{\vec{\mathbf{h}}}}y_{\mathbf{q}}\right) \\ \dfrac{\partial F}{\partial z_{\mathbf{q}}} = 4x_{\Delta\vec{\mathbf{a}}}x_{\mathbf{q}} + 4y_{\Delta\vec{\mathbf{a}}}y_{\mathbf{q}} + 4x_{\Delta\vec{\mathbf{h}}}\left(y_{E_{\vec{\mathbf{h}}}}w_{\mathbf{q}} + z_{E_{\vec{\mathbf{h}}}}x_{\mathbf{q}}\right) \\ \qquad\qquad + 4y_{\Delta\vec{\mathbf{h}}}\left(z_{E_{\vec{\mathbf{h}}}}y_{\mathbf{q}} - y_{E_{\vec{\mathbf{h}}}}z_{\mathbf{q}}\right) + 4z_{\Delta\vec{\mathbf{h}}}y_{E_{\vec{\mathbf{h}}}}y_{\mathbf{q}} \end{cases}$$

计算出了梯度，然后要确定步长，这里步长定为梯度的模的 β 倍。最后把梯度乘以步长，融合到陀螺运算的姿态里了。

$${}^{E}_{A_n}\mathbf{q} = \frac{\mathbf{t}}{|\mathbf{t}|} \quad , \quad \mathbf{t} = {}^{E}_{A_n}\mathbf{q}_{gyr} - \beta\cdot\frac{\partial F}{\partial \mathbf{q}} \tag{4-33}$$

梯度下降法的流程如图 4-2。

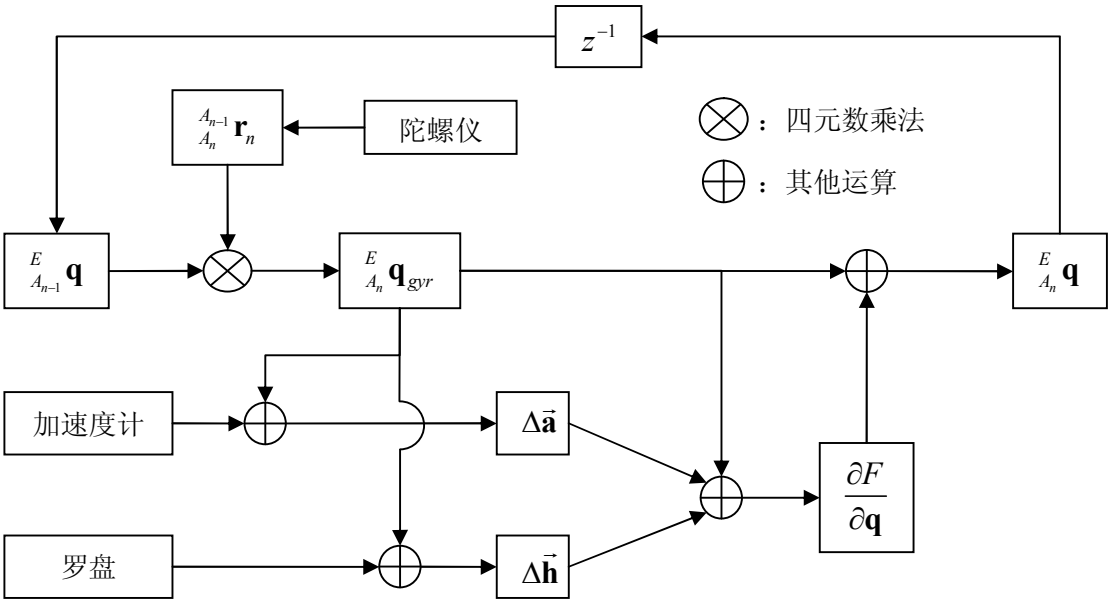

图 4-2 梯度下降法数据流程图

4.2.4. 互补滤波法

Mark Euston 提出了运算量比梯度下降法更小的互补滤波法[13]。相比梯度下降法用加速度和磁场强度计算姿态的梯度，互补滤波法是把加速度和磁场强度的误差构造成纠正旋转，叠加到陀螺测出的角增量上，实现高效的数据融合。

统一到同一坐标系后，规范化的加速度和磁场强度的测量值和常量值可以作一个叉积，叉积的模为角度误差的正弦，小角情况下认为正比于角度，方向根据右手法则，可以作为纠正的旋转轴，乘上一个系数后，可以与陀螺算出的角增量叠加。

首先算出两个叉积：

$$\vec{\mathbf{c}}_a = \frac{\overset{A}{}\vec{\mathbf{a}}_m}{\left|\overset{A}{}\vec{\mathbf{a}}_m\right|} \times \mathrm{R}^{-1}\left(\overset{E}{A}\mathbf{q}\right) \cdot \overset{E}{}\vec{\mathbf{a}}_c \tag{4-34}$$

$$\vec{\mathbf{c}}_h = \frac{\overset{A}{}\vec{\mathbf{h}}_m}{\left|\overset{A}{}\vec{\mathbf{h}}_m\right|} \times \mathrm{R}^{-1}\left(\overset{E}{A}\mathbf{q}\right) \cdot \overset{E}{}\vec{\mathbf{h}}_c \tag{4-35}$$

然后把叉积乘以一个系数，加到陀螺的角增量上，γ 和 λ 分别为加速度和磁场的纠正系数，一般取接近 0 的数：

$$\vec{\boldsymbol{\varphi}} = \vec{\mathbf{g}} \cdot \Delta t + \vec{\mathbf{c}}_a \cdot \gamma + \vec{\mathbf{c}}_h \cdot \lambda \tag{4-36}$$

然后类似(4-16)式，构造旋转增量，更新姿态：

$$\overset{A_{n-1}}{A_n}\mathbf{r}_n = \left[\sqrt{1-\frac{1}{4}\left(x_{\vec{\varphi}}^2 + y_{\vec{\varphi}}^2 + z_{\vec{\varphi}}^2\right)} \quad \frac{1}{2}x_{\vec{\varphi}} \quad \frac{1}{2}y_{\vec{\varphi}} \quad \frac{1}{2}z_{\vec{\varphi}}\right]^{\mathrm{T}} \tag{4-37}$$

$$\overset{E}{A_n}\mathbf{q} = \overset{E}{A_{n-1}}\mathbf{q} \otimes \overset{A_{n-1}}{A_n}\mathbf{r}_n \tag{4-38}$$

可以看出这个方法的复杂性比前两个简单，运算量比前两个小。互补滤波法的流程如图 4-3。

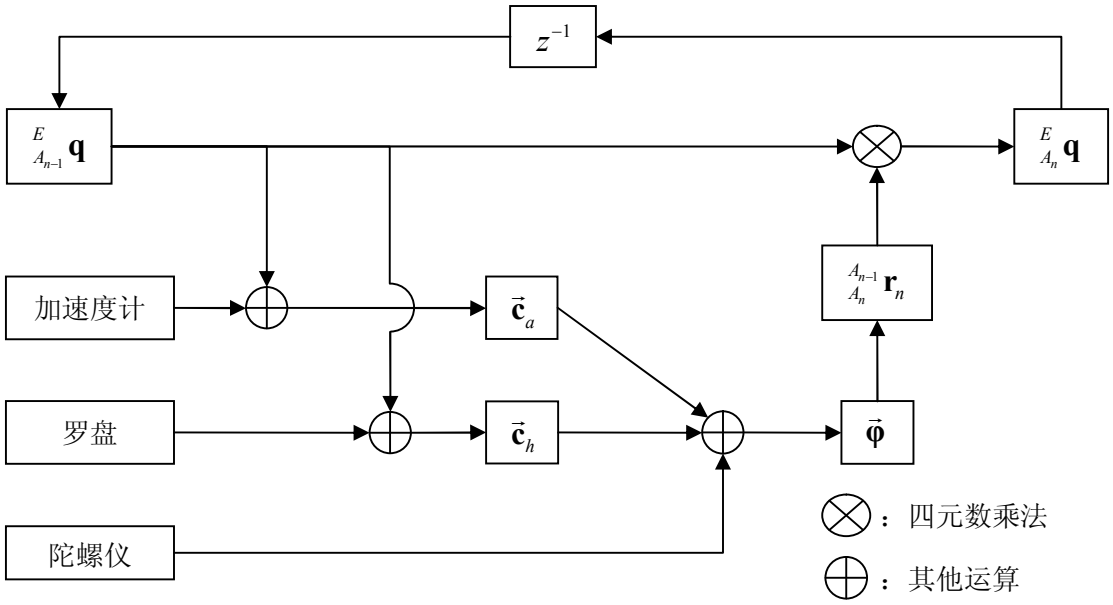

图 4-3 互补滤波法数据流程图

5. 姿态控制

姿态计算出来后，就可以输出控制了。根据被控姿态的表示方式，分为欧拉角控制和四元数控制。控制的思路为：设定一个目标姿态，调整螺旋桨，使测量出的姿态变为目标姿态。为了避免复杂的精确动力学建模[14]，选用 PID 控制器。

5.1. 欧拉角控制

由于欧拉角对应 3 个轴的旋转，当前姿态和目标姿态的差值可以作为控制输入量，角度的误差直接可以对应力矩的输出。如果当前姿态和目标姿态相差不大，可以忽略旋转顺序的影响。

首先把姿态四元数转成欧拉角。欧拉角有多种定义方式，这里定义 x 轴正方向为航向，绕 z 轴转的角度为偏航角(yaw)，绕 y 轴转的角度为俯仰角(pitch)，绕 x 轴转的角度为滚转角(roll)，旋转顺序为 ZYX。具体转换公式见(5-1)。

$$\begin{cases} \text{yaw}: \quad y_a = \arctan 2\left(2w_{\mathbf{q}}z_{\mathbf{q}} + 2x_{\mathbf{q}}y_{\mathbf{q}}, 1 - 2y_{\mathbf{q}}^2 - 2z_{\mathbf{q}}^2\right) \\ \text{pitch}: p_a = \arcsin\left(2w_{\mathbf{q}}y_{\mathbf{q}} - 2z_{\mathbf{q}}x_{\mathbf{q}}\right) \\ \text{roll}: \quad r_a = \arctan 2\left(2w_{\mathbf{q}}x_{\mathbf{q}} + y_{\mathbf{q}}z_{\mathbf{q}}, 1 - 2x_{\mathbf{q}}^2 - 2y_{\mathbf{q}}^2\right) \end{cases} \tag{5-1}$$

目标姿态由控制信号确定。初始状态时，三个角度都为 0，当飞行器需要运动时，就根据飞行的方向和速度给出对应的俯仰角 p_c 和滚转角 r_c；当需要旋转机体时，给出对应的航偏角 y_c。

确定好当前姿态和目标姿态后，就对两者作差，得到 3 个角度差：

$$\begin{cases} \Delta y = y_c - y_a \\ \Delta p = p_c - p_a \\ \Delta r = r_c - r_a \end{cases} \tag{5-2}$$

3 个角度差都是小角，分别对应 3 个欧拉角需要的增量，也对应 3 个轴需要的力矩，可以用 PID 控制器把角增量和力矩联系起来，如(5-3)式。PID 控制器只使用了 PD 部分，K_p 和 K_d 分别为 P 和 D 的参数。

$$\begin{cases} z_{\vec{\mathbf{m}}} = K_p \cdot \Delta y_n - K_d \cdot \left(\Delta y_n - \Delta y_{n-1} \right) \\ y_{\vec{\mathbf{m}}} = K_p \cdot \Delta p_n - K_d \cdot \left(\Delta p_n - \Delta p_{n-1} \right) \\ x_{\vec{\mathbf{m}}} = K_p \cdot \Delta r_n - K_d \cdot \left(\Delta r_n - \Delta r_{n-1} \right) \end{cases} \tag{5-3}$$

计算出力矩后，就可以利用(3-5)式调整螺旋桨的功率了。因为 PID 控制器需要调试参数，力矩系数可以包含在参数里面，(3-5)式中的力矩系数可以忽略，。

5.2. 四元数控制

用欧拉角来控制姿态，每次控制都要算 3 次三角函数，运算量很大。为了避免三角函数，可以直接用姿态四元数来控制。思路跟欧拉角控制一样，先求姿态差，再把姿态差输入到 PID 控制器，来输出油门变化量。

当前姿态记为 $\mathbf{c}$，目标姿态记为 $\mathbf{t}$，从当前姿态转到目标姿态的旋转，即姿态差，记为 $\mathbf{d}$，则有：

$$\begin{aligned} & \mathbf{c} \otimes \mathbf{d} = \mathbf{t} \\ \Leftrightarrow\ & \mathbf{c}^{-1} \otimes \mathbf{c} \otimes \mathbf{d} = \mathbf{c}^{-1} \otimes \mathbf{t} \\ \Leftrightarrow\ & \mathbf{d} = \mathbf{c}^{-1} \otimes \mathbf{t} \end{aligned} \tag{5-4}$$

假定姿态差为小量，三角函数可以用小角替换，根据四元数表示姿态的意义，见(3-2)式，$x_{\mathbf{d}}$、$y_{\mathbf{d}}$ 和 $z_{\mathbf{d}}$ 为各轴旋转角的一半，可以作为 PID 的输入：

$$\begin{cases} z_{\vec{\mathbf{m}}} = K_p \cdot z_{d,n} - K_d \cdot \left(z_{d,n} - z_{d,n-1} \right) \\ y_{\vec{\mathbf{m}}} = K_p \cdot y_{d,n} - K_d \cdot \left(y_{d,n} - y_{d,n-1} \right) \\ x_{\vec{\mathbf{m}}} = K_p \cdot x_{d,n} - K_d \cdot \left(x_{d,n} - x_{d,n-1} \right) \end{cases} \tag{5-5}$$

最后根据(3-5)式得到油门变化量。因为用的是 PID 控制器，(3-5)式中的力矩系数可以忽略。

6. 算法效果

6.1. 加速度计校正

按(4-12)式计算矩阵 $\mathbf{B}$，结果如表 6-1：

表 6-1 用测量数据得到的矩阵 $\mathbf{B}$

9.21183E+15	4.49429E+13	1.00577E+13	1.95005E+11	4.11950E+10	4.85747E+09	5.43845E+08
4.49429E+13	9.15746E+15	1.32965E+12	4.51877E+10	6.07766E+10	7.76086E+09	5.42682E+08
1.00577E+13	1.32965E+12	9.29868E+15	6.56837E+10	9.62601E+09	6.45181E+10	5.45455E+08
1.95005E+11	4.51877E+10	6.56837E+10	5.43845E+08	-1.48676E+06	7.34192E+06	1.04500E+04
4.11950E+10	6.07766E+10	9.62601E+09	-1.48676E+06	5.42682E+08	-9.14325E+05	4.29700E+03
4.85747E+09	7.76086E+09	6.45181E+10	7.34192E+06	-9.14325E+05	5.45455E+08	1.91600E+03
5.43845E+08	5.42682E+08	5.45455E+08	1.04500E+04	4.29700E+03	1.91600E+03	9.60000E+01

注：数据在三个轴正负半轴附近采集，各采 16 组、共 96 组数据。

然后解出一组 $\mathbf{P}_b$，如式(6-1)。

$$\mathbf{P}_b = \begin{bmatrix} -5.89997e-08 \\ -5.90601e-08 \\ -5.87218e-08 \\ 1.38898e-05 \\ 4.26284e-06 \\ 4.61911e-06 \\ 1 \end{bmatrix}^{\mathrm{T}} \tag{6-1}$$

代入(4-15)式即得参数。

$$\begin{cases} o_x = -117.711 \\ o_y = -36.089 \\ o_z = -39.3305 \\ g_x = 0.00237639 \\ g_y = 0.00237761 \\ g_z = 0.00237079 \end{cases} \tag{6-2}$$

根据 MPU6050 的数据手册，$g = \dfrac{9.7883}{4096} \approx 0.00238972$，与(6-2)式中的 g 相差不超过 1%，结果是比较合理的。

再比较校正前后的长度误差分布，校正前的相对误差分布如图 6-1，校正后的相对误差分布如图 6-2。

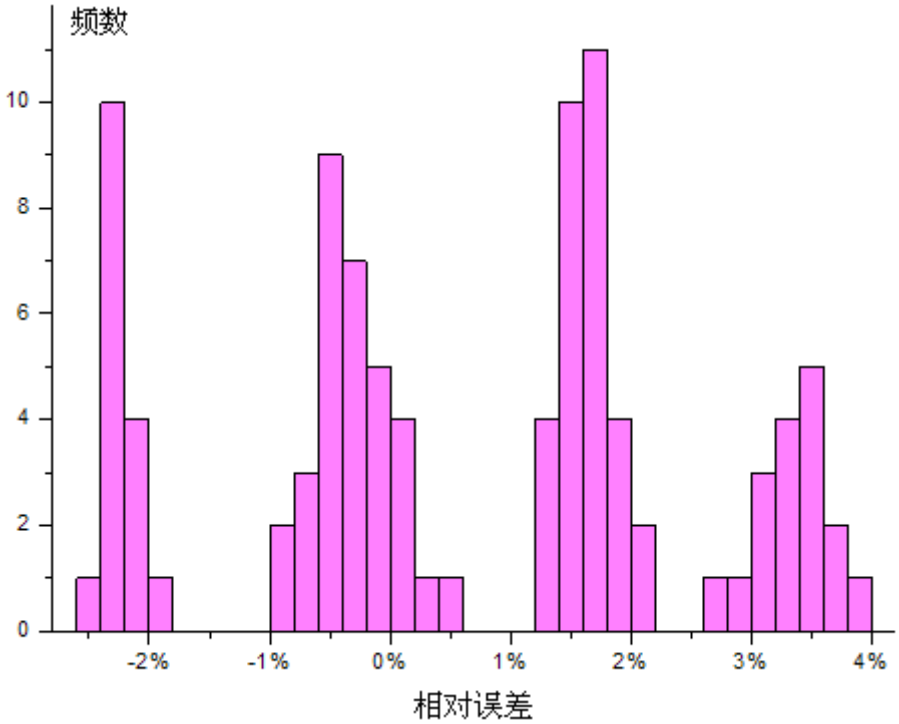

图 6-1 校正前的长度相对误差分布

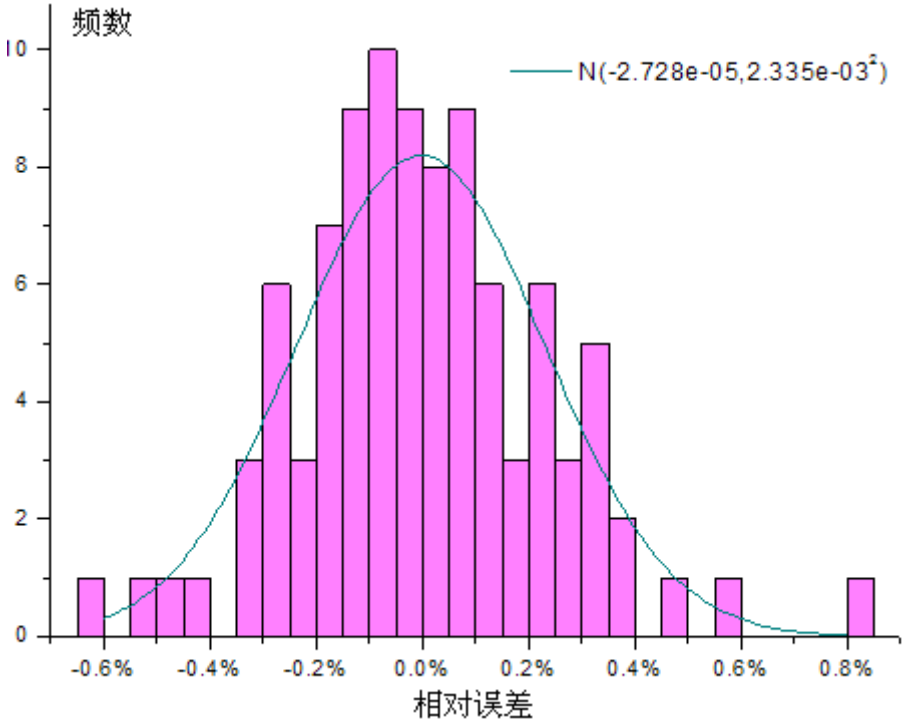

图 6-2 校正后的长度相对误差分布

假设校正后的长度误差服从正态分布，进行一次拟合优度检验。数据最大似然期望为-2.728e-05，最大似然标准差为 2.335e-03，则实际的频数和理论的频数如表(6-2)。

表 6-2 校正后的加速度的长度相对误差的分布

序号	范围	实际频数	理论频数
1	-∞ ~ -0.3%	7	9.75
2	-0.3% ~ -0.2%	9	9.37
3	-0.2% ~ -0.15%	7	6.24
4	-0.15% ~ -0.1%	9	7.14
5	-0.1% ~ -0.05%	10	7.8
6	-0.05% ~ 0%	9	8.15
7	0% ~ 0.05%	8	8.13
8	0.05% ~ 0.1%	9	7.74
9	0.1% ~ 0.15%	6	7.05
10	0.15% ~ 0.25%	9	11.23
11	0.25% ~ 0.35%	8	7.11
12	0.35% ~ ∞	5	6.29

计算得到 Pearson 拟合优度 χ^2 统计量为 3.25。如果假设成立，χ^2 符合自由度为 12-2-1=9 的 χ^2 分布。因为 $\chi^2 < \chi_9^2(0.05)=16.919$，表明在显著性水平为 0.05 的情况下接受假设[15]，可以认为误差服从正态分布，误差是随机噪声造成的，校正减小了系统误差。误差最大值也从 4%降到了 0.85%。

6.2. 姿态计算

为了比较几种姿态计算算法的效果，先在下位机采集数据，然后在电脑上离线处理，这样可以用相同的数据进行计算和比较。数据分两组，分别对应静态和动态的情况，测量时电机都是开的，

因此把电机振动也考虑进来了。

先比较静态的情况。因为几种姿态融合方法的思路都是：陀螺为主、加速度计和罗盘用于纠正陀螺误差，因此动态性能取决于陀螺，静态性能取决于加速度计和罗盘，所以静态的情况最能反映姿态融合算法的优劣。图 6-3 为三种方法算出的滚转角，取了其中连续的 1000 个样点，即连续 5 秒时间的数据。为了公平比较，先把参数调整到临界值，即刚好能纠正陀螺漂移的值。由图可以看出，姿态插值法和互补滤波法效果差不多，梯度下降法噪声振幅比前两者都大。

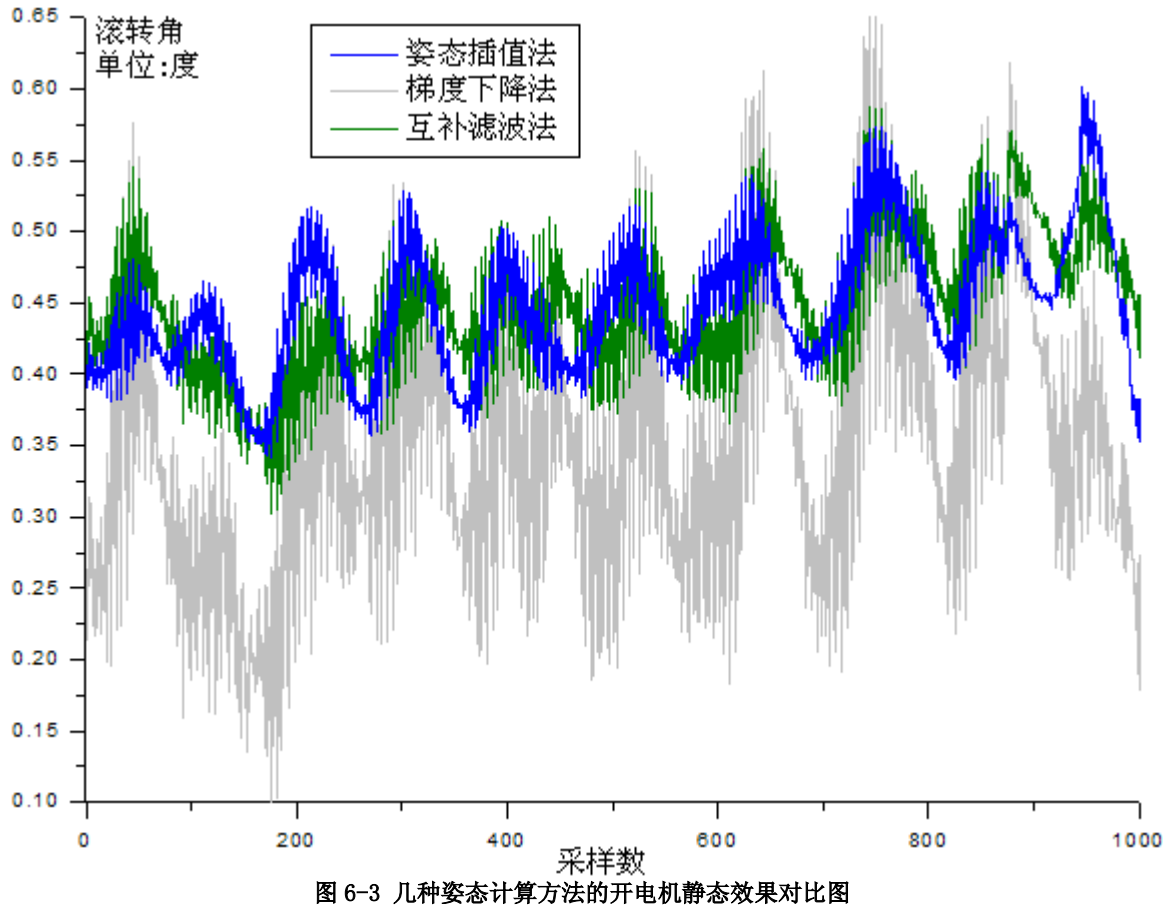

图 6-3 几种姿态计算方法的开电机静态效果对比图

注：姿态插值法的 $\alpha = 0.003$，梯度下降法的 $\beta = 0.002$，互补滤波法的 $\gamma = \lambda = 0.003$。

然后比较动态的情况。如图 6-4，“无陀螺姿态”是指仅用加速度计和罗盘计算的姿态，相当于姿态插值法第二部分得到的姿态，可以看到 3 种算法光滑程度差别不大，因为动态时，性能由陀螺决定，而且相对于几十度的运动角度，零点几度的噪声几乎忽略不计。但是不同算法不同参数运算的结果相差比较大，由于没有专业的测量仪器，哪种算法的结果更接近实际值有待以后的研究。

比较运算量，姿态插值法远大于梯度下降法，梯度下降法又稍大于互补滤波法。比较结果效果，姿态插值法跟互补滤波法差不多，都比梯度下降法好一点。最终方案是选择互补滤波法。

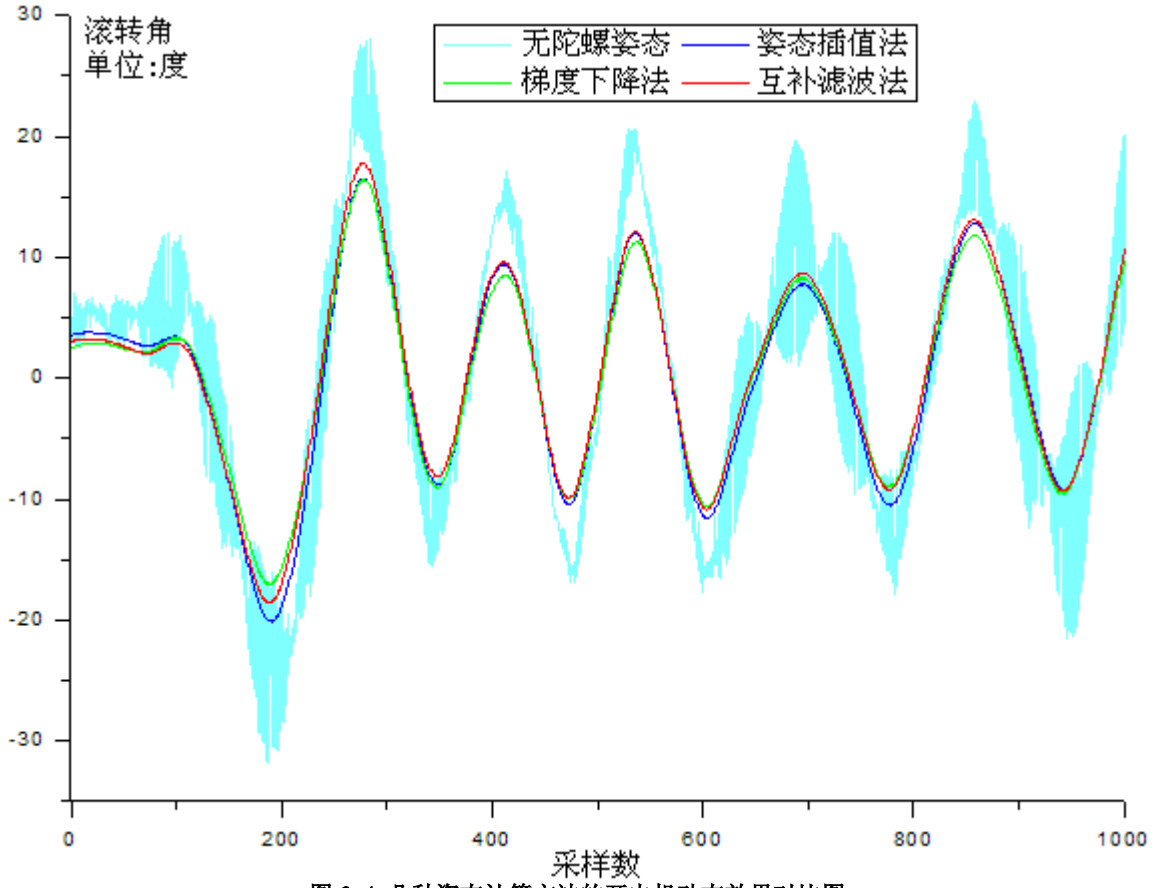

图 6-4 几种姿态计算方法的开电机动态效果对比图

注：姿态插值法的$\alpha = 0.003$，梯度下降法的$\beta = 0.002$，互补滤波法的$\gamma = \lambda = 0.003$。

7. 结论及存在的问题

虽然通过使用上面介绍的各种方法，已经成功让飞行器飞起来了，但为了把飞行器应用到遥感，至少还需要三个改进：一、改进控制算法，使机体更平稳；二、增加卫星导航定位，实现预定航线飞行；三、要用增稳云台，遥感传感器要装在云台上。

参考文献：

[1] 彭军桥.非共轴式碟形飞行器研究[D].上海大学 2001 级硕士研究生学位毕业论文:1-68.

[2] 吴东国.基于四旋翼飞行器平台的低空遥感技术在公路环境调查中的应用[J].公路交通技术.2012.第 6 期:137-138.

[3] 姬江涛，扈菲菲，贺智涛，杜新武，刘剑君.四旋翼无人机在农田信息获取中的应用[J].农机化研究.2013.第 2 期:1-4.

[4] 赵晨,杜勇.四旋翼无人机在输电线路巡视中的应用[J].湖北电力.2012.第36卷第6期:35-36.

[5] 张鹏，程飞，曹宇强，孙来，王琪.一种新型四轴搜救飞行器设计[J].科技广场.2010 年 09 期:145-147.

[6] 王帅，周洋.用于危险区域物品清理的四旋翼飞行抓捕手[J].兵工自动化.2011-03.第 30 卷第 3 期:78-80.

[7] 刘庆元，徐柳华，沈彩莲，王小平.基于无人飞行器遥感影像的数字摄影测量关键技术研究[J].测绘科学.2010.第 35 卷第 1 期:28-30.

[8] 岳基隆，张庆杰，朱华勇.微小型四旋翼无人机研究进展及关键技术浅析[J].电光与控制.2010-10.第 17 卷第 10 期:46-52.

[9] Philip J.Schneider, David H.Eberly.计算机图形学几何工具算法详解[M].第一次印刷.北京.电子工业出版社.2005:633-641.

[10] 李俊峰，张雄.理论力学[M].第二版.北京.清华大学出版社.2010:213-216.

[11] 蒋庆仙.关于 MEMS 惯性传感器的发展及在组合导航中的应用前景[J].测绘通报.2006.09:5-8.

[12] Sebastian O.H.Madgwick, Andrew J.L.Harrison,Ravi Vaidyanathan.Estimation of IMU and MARG orientation using a gradient descent algorithm[J].IEEE International Conference on Rehabilitation Robotics.2011.1:1-7.

[13] Mark Euston, Paul Coote, Robert Mahony , Jonghyuk Kim and Tarek Hamel.A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV[J].IEEE/RSJ International Conference on Intelligent Robots and Systems.2008.Sept,22-26 :340-345.

[14] 王冬来，吕强，刘峰.小型四轴飞行器动力学参数测定方法设计[J].科技导报.2011.第 29 卷第 36 期:42-45.

[15] 王松桂，张忠占，程维虎，高旅端.概率论与数理统计[M].第二版.北京.科学出版社.2009:185-189.

致　谢

首先必须感谢的是父母和学校，是他们提供了条件，我才可以进行相关研究，这篇论文才有可能写出来。

感谢各位一起制作四轴飞行器的网友，跟他们的交流使自己的思路更加清晰和开阔。特别感谢cctsao，论文中的方案使用的正是cctsao赠送的飞控电路板。

感谢伍兆强老师，在跟老师做项目的过程中，学到了远比学习本身更重要的东西，让我提前意识到“求知”到“适应”的转变。

感谢小学的褶显微老师，你讲的精彩的故事丰富了一个少年的生活，你的科技实践课指引了一个少年的奋斗方向。

感谢Internet和当当网。

附录 算法的 C 语言实现代码

```
// 定义四元数类型。
typedef struct
{
    float w;
    float x;
    float y;
    float z;
}quaternion;

// 规范化四元数。
void quaternion_normalize(quaternion * q)
{
    float norm_r = 1.0f / sqrtf(q->w*q->w + q->x*q->x + q->y*q->y + q->z*q->z);
    q->w *= norm_r;
    q->x *= norm_r;
    q->y *= norm_r;
    q->z *= norm_r;
}

// 四元数相乘
// left   : 被乘数，输入。
// right  : 乘数，输入。
// result : 积，输出。
void quaternion_mult(quaternion * result,const quaternion * left,const quaternion * right)
{
    result->w = left->w * right->w - left->x * right->x - left->y * right->y - left->z * right->z;
    result->x = left->x * right->w + left->w * right->x + left->y * right->z - left->z * right->y;
    result->y = left->y * right->w + left->w * right->y + left->z * right->x - left->x * right->z;
    result->z = left->z * right->w + left->w * right->z + left->x * right->y - left->y * right->x;
}

// 用四元数来旋转向量。
void quaternion_rotateVector(const quaternion * rotation,const float from[3],float to[3])
{
    float x2  = rotation->x * 2;
    float y2  = rotation->y * 2;
    float z2  = rotation->z * 2;
    float wx2 = rotation->w * x2;
    float wy2 = rotation->w * y2;
    float wz2 = rotation->w * z2;
    float xx2 = rotation->x * x2;
    float yy2 = rotation->y * y2;
    float zz2 = rotation->z * z2;
    float xy2 = rotation->x * y2;
    float yz2 = rotation->y * z2;
    float xz2 = rotation->z * x2;
    //
    to[0] = from[0]*(1 - yy2 - zz2) + from[1]*(xy2 - wz2)     + from[2]*(xz2 + wy2);
    to[1] = from[0]*(xy2 + wz2)     + from[1]*(1 - xx2 - zz2) + from[2]*(yz2 - wx2);
    to[2] = from[0]*(xz2 - wy2)     + from[1]*(yz2 + wx2)     + from[2]*(1 - xx2 - yy2);
}

//
// 两向量旋转→四元数旋转。
// 输入：from、to 两向量，长度都必须大于 0。
// 输出：从 from 方向转到 to 方向的旋转。
void quaternion_fromTwoVectorRotation(quaternion * result,const float from[3],const float to[3])
{
```

```
    float from_norm = fabsf(from[0]*from[0] + from[1]*from[1] + from[2]*from[2]);
    float to_norm = fabsf(to[0]*to[0] + to[1]*to[1] + to[2]*to[2]);
    //
    from_norm = sqrtf(from_norm);
    to_norm = sqrtf(to_norm);
    float cos_theta = (from[0]*to[0] + from[1]*to[1] + from[2]*to[2]) / (from_norm*to_norm);
    result->w = sqrtf((1.0f + cos_theta) / 2); // cos(theta/2)
    float sin_half_theta = sqrtf((1 - cos_theta) / 2);
    float cross_x = from[1]*to[2] - from[2]*to[1];
    float cross_y = from[2]*to[0] - from[0]*to[2];
    float cross_z = from[0]*to[1] - from[1]*to[0];
    //
    float sin_half_theta_div_cross_norm = sin_half_theta /
        sqrtf(cross_x*cross_x + cross_y*cross_y + cross_z*cross_z);
    result->x = cross_x * sin_half_theta_div_cross_norm;
    result->y = cross_y * sin_half_theta_div_cross_norm;
    result->z = cross_z * sin_half_theta_div_cross_norm;
}

// 姿态插值法。
void mix_AttitudeInterpolation(quaternion * attitude,const float gyr[3],
        const float acc[3],const float mag[3],float interval)
{
    //
    // 第一部分，陀螺积分。
    // 构造增量旋转。
    // 由于角增量是小角，w 会很接近 1，为了减小运算量，令其为 1。
    float delta_x = gyr[0] * interval / 2;
    float delta_y = gyr[1] * interval / 2;
    float delta_z = gyr[2] * interval / 2;
    //
    // 融合，四元数乘法。
    float q_w = attitude->w;
    float q_x = attitude->x;
    float q_y = attitude->y;
    float q_z = attitude->z;
    attitude->w = q_w         - q_x*delta_x - q_y*delta_y - q_z*delta_z;
    attitude->x = q_w*delta_x + q_x         + q_y*delta_z - q_z*delta_y;
    attitude->y = q_w*delta_y - q_x*delta_z + q_y         + q_z*delta_x;
    attitude->z = q_w*delta_z + q_x*delta_y - q_y*delta_x + q_z;
    //
    // 第二部分，加速度计和罗盘纠正积分漂移。
    // 先规范化测量的加速度和磁场强度。
    float acc_n[3],mag_n[3];
    float a_rsqrt = 1.0f / sqrtf(acc[0]*acc[0]+acc[1]*acc[1]+acc[2]*acc[2]);
    float h_rsqrt = 1.0f / sqrtf(mag[0]*mag[0]+mag[1]*mag[1]+mag[2]*mag[2]);
    for(int i=0;i<3;i++)
    {
        acc_n[i] = acc[i] * a_rsqrt;
        mag_n[i] = mag[i] * h_rsqrt;
    }
    //
    // 计算对角线和平面法线。
    const static float MAG_VECTOR[] = {0,0.743144f/*cos(42)*/,-0.669130f/*sin(-42)*/};
    const static float ACC_VECTOR[] = {0,0,1};
    float mid_from[3],mid_to[3],cross_from[3],cross_to[3];
    math_vector_cross(cross_from,acc_n,mag_n);
    math_vector_cross(cross_to,ACC_VECTOR,MAG_VECTOR);
    for(int i=0;i<3;i++)
    {
```

```
        mid_from[i] = acc_n[i] + mag_n[i];
        mid_to[i] = MIX_ACC_VECTOR[i] + MIX_MAG_VECTOR[i];
    }
    //
    // 先把 mid 转到重合。
    quaternion rotation_1;
    quaternion_fromTwoVectorRotation(&rotation_1,mid_from,mid_to);
    //
    // 然后把 cross 转到重合。
    quaternion rotation_2;
    float cross_from_1[3];
    quaternion_rotateVector(&rotation_1,cross_from,cross_from_1);
    quaternion_fromTwoVectorRotation(&rotation_2,cross_from_1,cross_to);
    //
    // 再组合两次旋转。
    quaternion target;
    quaternion_mult(&target,&rotation_2,&rotation_1);
    quaternion_normalize(&target);
    //
    // 把两部分的姿态进行插值。
    static const float ALPHA = 0.003; // 插值系数。
    attitude->w = (1-ALPHA)*attitude->w + ALPHA*target.w;
    attitude->x = (1-ALPHA)*attitude->x + ALPHA*target.x;
    attitude->y = (1-ALPHA)*attitude->y + ALPHA*target.y;
    attitude->z = (1-ALPHA)*attitude->z + ALPHA*target.z;
    //
    // 最后规范化一下姿态，防止发散。
    quaternion_normalize(attitude);
}

// 梯度下降法。
void mix_SteepestDescent(quaternion * attitude,const float gyr[3],
        const float acc[3],const float mag[3],float interval)
{
    //
    // 先处理陀螺积分。
    // 构造增量旋转。
    // 由于角增量是小角，w 会很接近 1，为了减小运算量，令其为 1。
    float delta_x = gyr[0] * interval / 2;
    float delta_y = gyr[1] * interval / 2;
    float delta_z = gyr[2] * interval / 2;
    //
    // 融合，四元数乘法。
    float q_w = attitude->w;
    float q_x = attitude->x;
    float q_y = attitude->y;
    float q_z = attitude->z;
    attitude->w = q_w         - q_x*delta_x - q_y*delta_y - q_z*delta_z;
    attitude->x = q_w*delta_x + q_x         + q_y*delta_z - q_z*delta_y;
    attitude->y = q_w*delta_y - q_x*delta_z + q_y         + q_z*delta_x;
    attitude->z = q_w*delta_z + q_x*delta_y - q_y*delta_x + q_z;
    //
    // 然后用梯度下降法纠正陀螺漂移。
    // 规范化测量的加速度和磁场强度。
    float a_rsqrt = 1.0f / sqrtf(acc[0]*acc[0]+acc[1]*acc[1]+acc[2]*acc[2]);
    float x_a = acc[0] * a_rsqrt;
    float y_a = acc[1] * a_rsqrt;
    float z_a = acc[2] * a_rsqrt;
    float h_rsqrt = 1.0f / sqrtf(mag[0]*mag[0]+mag[1]*mag[1]+mag[2]*mag[2]);
    float x_h = mag[0] * h_rsqrt;
```

```
    float y_h = mag[1] * h_rsqrt;
    float z_h = mag[2] * h_rsqrt;
    //
    // 预先计算一些重复使用的值，减少运算浪费。
    float w_q = attitude->w;
    float x_q = attitude->x;
    float y_q = attitude->y;
    float z_q = attitude->z;
    float x_q_2 = x_q * 2;
    float y_q_2 = y_q * 2;
    float z_q_2 = z_q * 2;
    //
    // 计算加速度和磁场强度的误差。
    const static float MAG_Y =  0.743144f; // cos(-42° )，实测当地磁倾角为-42° 。
    const static float MAG_Z = -0.669130f; // sin(-42° )
    float x_da = x_q*z_q_2 - w_q*y_q_2     - x_a;
    float y_da = y_q*z_q_2 + w_q*x_q_2     - y_a;
    float z_da = 1 - x_q*x_q_2 - y_q*y_q_2 - z_a;
    float x_dh = MAG_Y*(x_q*y_q_2 + w_q*z_q_2)     + MAG_Z*(x_q*z_q_2 - w_q*y_q_2)     - x_h;
    float y_dh = MAG_Y*(1 - x_q*x_q_2 - z_q*z_q_2) + MAG_Z*(y_q*z_q_2 + w_q*x_q_2)     - y_h;
    float z_dh = MAG_Y*(y_q*z_q_2 - w_q*x_q_2)     + MAG_Z*(1 - x_q*x_q_2 - y_q*y_q_2) - z_h;
    //
    // 计算误差对姿态的梯度的 1/4。
    float w_pf =  - x_da*y_q + y_da*x_q + x_dh*(MAG_Y*z_q - MAG_Z*y_q)           \
            + y_dh*MAG_Z*x_q - z_dh*MAG_Y*x_q;
    float x_pf = x_da*z_q + y_da*w_q - z_da*x_q + x_dh*(MAG_Y*y_q + MAG_Z*z_q)   \
            + y_dh*(MAG_Z*w_q - MAG_Y*x_q) - z_dh*(MAG_Y*w_q + MAG_Z*x_q);
    float y_pf = - x_da*w_q + y_da*z_q - z_da*y_q + x_dh*(MAG_Y*x_q - MAG_Z*w_q) \
            + y_dh*MAG_Z*z_q + z_dh*(MAG_Y*z_q - MAG_Z*y_q);
    float z_pf = x_da*x_q + y_da*y_q + x_dh*(MAG_Y*w_q + MAG_Z*x_q)              \
            + y_dh *(MAG_Z*y_q - MAG_Y*z_q) + z_dh*MAG_Y*y_q;
    //
    // 用梯度更新姿态。
    const static float BETA = 0.002*4;
    attitude->w -= w_pf * BETA;
    attitude->x -= x_pf * BETA;
    attitude->y -= y_pf * BETA;
    attitude->z -= z_pf * BETA;
    //
    // 最后要规范化姿态，防止发散。
    quaternion_normalize(attitude);
}

// 互补滤波法。
void mix_ComplementaryFilter(quaternion * attitude,const float gyr[3],
        const float acc[3],const float mag[3],float interval)
{
    //
    // 规范化测量的加速度和磁场强度。
    float a_rsqrt = 1.0f / sqrtf(acc[0]*acc[0]+acc[1]*acc[1]+acc[2]*acc[2]);
    float x_a = acc[0] * a_rsqrt;
    float y_a = acc[1] * a_rsqrt;
    float z_a = acc[2] * a_rsqrt;
    float h_rsqrt = 1.0f / sqrtf(mag[0]*mag[0]+mag[1]*mag[1]+mag[2]*mag[2]);
    float x_h = mag[0] * h_rsqrt;
    float y_h = mag[1] * h_rsqrt;
    float z_h = mag[2] * h_rsqrt;
    //
    // 预先计算一些重复使用的值，减少运算浪费。
    float w_q = attitude->w;
```

```
    float x_q = attitude->x;
    float y_q = attitude->y;
    float z_q = attitude->z;
    float x_q_2 = x_q * 2;
    float y_q_2 = y_q * 2;
    float z_q_2 = z_q * 2;
    //
    // 计算常量在机体坐标下的表示。
    const static float MAG_Y =  0.743144f; // cos(-42° )，实测当地磁倾角为-42° 。
    const static float MAG_Z = -0.669130f; // sin(-42° )
    float x_ae = x_q*z_q_2 - w_q*y_q_2;
    float y_ae = y_q*z_q_2 + w_q*x_q_2;
    float z_ae = 1 - x_q*x_q_2 - y_q*y_q_2;
    float x_he = MAG_Y*(x_q*y_q_2 + w_q*z_q_2)     + MAG_Z*(x_q*z_q_2 - w_q*y_q_2);
    float y_he = MAG_Y*(1 - x_q*x_q_2 - z_q*z_q_2) + MAG_Z*(y_q*z_q_2 + w_q*x_q_2);
    float z_he = MAG_Y*(y_q*z_q_2 - w_q*x_q_2)     + MAG_Z*(1 - x_q*x_q_2 - y_q*y_q_2);
    //
    // 测量值与常量的叉积。
    float x_ac = y_a * z_ae - z_a * y_ae;
    float y_ac = z_a * x_ae - x_a * z_ae;
    float z_ac = x_a * y_ae - y_a * x_ae;
    float x_hc = y_h * z_he - z_h * y_he;
    float y_hc = z_h * x_he - x_h * z_he;
    float z_hc = x_h * y_he - y_h * x_he;
    //
    // 构造增量旋转。
    const static float GAMMA = 0.003;
    const static float LAMDA = 0.003;
    float delta_x = gyr[0] * interval / 2 + x_ac * GAMMA + x_hc * LAMDA;
    float delta_y = gyr[1] * interval / 2 + y_ac * GAMMA + y_hc * LAMDA;
    float delta_z = gyr[2] * interval / 2 + z_ac * GAMMA + z_hc * LAMDA;
    //
    // 融合，四元数乘法。
    attitude->w = w_q         - x_q*delta_x - y_q*delta_y - z_q*delta_z;
    attitude->x = w_q*delta_x + x_q         + y_q*delta_z - z_q*delta_y;
    attitude->y = w_q*delta_y - x_q*delta_z + y_q         + z_q*delta_x;
    attitude->z = w_q*delta_z + x_q*delta_y - y_q*delta_x + z_q;
    //
    // 最后要规范化姿态，防止发散。
    quaternion_normalize(attitude);
}
```