SimSun SimHei FangSong$_G B2312$
gb$_4 52$

本科毕业设计论文

题 目 基于视觉的无人机飞行控制

专业名称 飞行器环境与生命保障工程

学生姓名 刘 文 萌

指导教师 布 树 辉

毕业时间 2015.06

摘　要

无人机因其续航能力强，成本低廉，无人员伤亡等载人飞行器不可比拟的优势，已成为目前的研究热点。本文通过 FlightGear 搭建飞行仿真平台，研究四旋翼无人机的飞行控制系统，从而提高飞行控制系统的鲁棒性。

首先，本文进行四旋翼无人机六自由度非线性飞行动力学建模。在建模过程中，本文分别从运动学角度与力学角度两方面进行考虑。在运动学方程推导过程中，基于坐标系变换的理论基础，采用四元数法对旋转矩阵进行转换，从而建立运动学方程。在力学方程推导过程中，基于四旋翼的力学特性，不考虑四旋翼的空气动力学特性，建立六自由度非线性方程组。在模型求解的过程中，基于小扰动假设，对非线性方程进行线性化，从而求解模型。

然后，本文进行仿真平台的搭建，基于 FlightGear 建立三维视景仿真系统，进行半物理仿真。本文利用 C++ 编程实现飞行摇杆 Joystick 数据读取，将读取数据通过 UDP 编程实现对 FlightGear 通信，从而进行飞行姿态的控制。在仿真平台搭建中，三维可视化的研究是非常重要的环节，可以对无人机的运动状态及总体结构进行直观显示，从而实现实时显示无人机的飞行位置及姿态变化的效果，增加飞行仿真的真实性。飞行仿真实验平台对飞行器测试研究工作有重要的作用，不仅可以总结飞机的飞行规律，而且能够对飞机的性能进行评估，从而降低试飞的风险和成本，提高飞行的安全性。

最后，本文在搭建的仿真平台上，进行飞行仿真效果的模拟。在模拟过程中，需要对 FlightGear 软件进行配置。要对飞行器模型及飞行环境等进行部署，完成飞行器模型部署后，便可以驱动模型进行模拟飞行。同时，本文还对四旋翼飞行控制模型进行改善，提出采取基于视觉的飞行控制思想的可行性，从而减弱人在整个控制系统中的角色，减弱人的干预，实现自主飞行。

关键词：六自由度飞控模型，四元数法，三维视景仿真系统，半物理仿真

Abstract

Due to endurance UAV capability, low cost, no casualties incomparable advantages of manned spacecraft Potential, it has become a hot topic. By building FlightGear flight simulation platform to study four spin wing UAV flight control system, thereby improving the robustness of the flight control system.

First, this paper four-rotor UAV six degrees of freedom nonlinear flight dynamics modeling. In the modeling process In this paper are considered from the perspective of the kinematics and mechanical point of two ways. In the kinematic equations derivation, Based on the theoretical basis of coordinate transformation, using quaternion rotation matrix conversion, in order to establish Kinematic equations. In mechanics equation derivation, based on the mechanical properties of four rotors, irrespective of four rotors Aerodynamic characteristics, the establishment of six degrees of freedom nonlinear equations. In the process of solving the model, based on the small disturbance Move assumptions, the nonlinear equation linearization, thereby solving the model.

Then, this paper build simulation platform, the establishment of three-dimensional visual simulation system based on FlightGear,Semi-physical simulation. In this paper, the use of C ++ programming Joystick Joystick data read, reads, According to FlightGear programming through UDP communications, thereby performing flight attitude control. In the simulation platform To build the three-dimensional visualization of research is a very important part, you can exercise status and overall UAV Structure for visual display, enabling real-time display of UAV flight position and attitude change effects, increasing .Plus flight simulation authenticity. Flight simulation platform for vehicle test research has an important role,Not only the law can be summarized fly the aircraft, but also be able to assess the performance of the aircraft, reducing flight Risks and costs, improve flight safety.

Finally, on building simulation platform, simulated flight simulation effects. During the simulation,FlightGear software needs to be configured. Meanwhile, the paper also four-rotor flight .Improved control model is proposed to take feasibility flight control based on visual thinking, thus weakening people.The role of the entire control system, reduced human intervention to achieve autonomous flight.

KEY WORDS: flight control model of six degrees of freedom, quaternion, three-dimensional visual simulation system, loop simulation

目　录

绪论

1.1 研究背景

无人驾驶飞机（Unmanned Aerial Vehicle）是无人飞行器的一种，简称无人机（UAV），它是由机体内部的飞行控制系统自主控制飞行任务，或者由飞机外控制器通过传送指令完成特定飞行任务的飞行器。飞行器内部无驾驶舱，但是由飞行动力装置，自动驾驶仪及程序控制装置组成。

随着科学技术的发展，无人机也能够完成以往只能由载人飞行器完成的任务，同时，无人机因其续航能力强，成本低廉，无人员伤亡等具备的载人飞行器不可比拟的优势，从而使无人机得到了广泛的应用。在军事方面，无人机除了可以完成电子干扰，对敌侦查，无线电中继等常规任务外，在危险恶劣的环境下，无人机还可以携带武器，例如：无人直升机不仅具备了直升机的共同优势，同时，因其垂直起降，自主悬停等独特的优点，无人直升机更适合在狭小的空间内进行飞行任务，在军事方面被广泛的应用在前沿阵地，舰船等狭窄的场地内起降，完成侦察敌情，勘测等任务 [?]。

在民用方面，无人机不仅可以应用到航拍技术，地形勘测，同时，还可以被用于抢险救灾，农业森林等方面。在很多载人飞行器无法直接操纵的环境中，无人机基于自身独特的优势，在地震抢险救灾，室内勘测等方面，常常担任空中机器人的角色。比如，新型的小型无人机因其成本低廉，结构简单，形状新颖，性能卓越以及独特的飞行控制方式 [?], 特别适合执行各种近地面环境的特殊任务。

1.2 研究内容及论文结构

本文研究目的是希望通过搭建飞行仿真平台，研究四旋翼无人机和固定翼无人机的飞行运动状态与姿态，从而提高飞行控制模型的鲁棒性。但是，现有的飞行仿真平台一般基于数值形态的计算，并不能够完全提供真实的三维视景仿真系统，从而不能够全面的反映出飞行控制系统的缺陷，具有一定的局限性。因此，本文基于开源的 FlightGear 飞行模拟软件，提供三维视景仿真系统，对飞机的飞行参数进行实时显示，同时，结合 C++ 编程实现飞行摇杆设备对无人机飞行控制，直观真实的模拟飞行过程，有利于发现飞行控制系统的缺陷，从而使飞行仿真更加逼近真实的飞行情况。

在仿真平台搭建中，三维可视化的研究是非常重要的环节，可以对无人机的运动状态及总体结构进行直观显示 [?]，从而实现实时显示无人机的飞行位置及姿态变化的效果，增加飞行仿真的真实性。飞行仿真实验平台对飞行器测

试研究工作有重要的作用，不仅可以总结飞机的飞行规律，而且能够对飞机的性能进行评估，从而降低试飞的风险和成本，提高飞行的安全性。

基于以上内容，本文的论文结构主要由六部分组成，如图**??**所示。

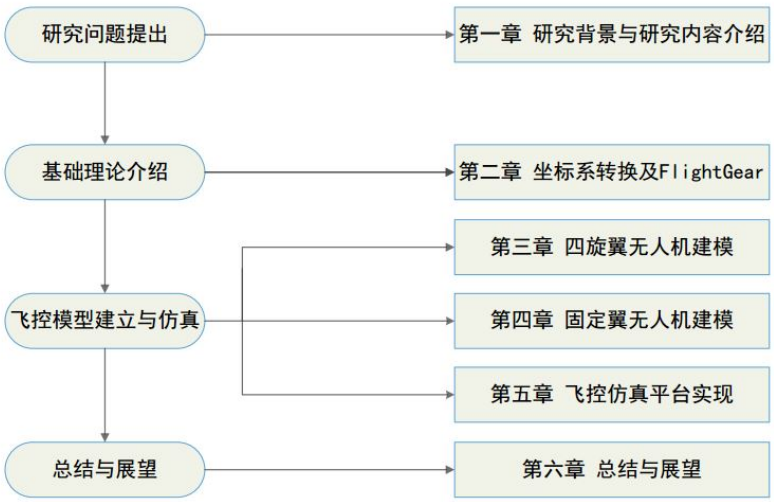


Figure 1.1: 论文结构

第**??**章：主要介绍问题提出的研究背景，研究内容及研究目的，从宏观的角度对本文进行概括，对问题进行概述。

第**??**章：推导四旋翼无人机坐标系变换公式与介绍 FlightGear 软件基础，，概述 FlightGear 软件的运作流程。

第**??**章：详细分析四旋翼无人机的飞控建模思路，首先，要通过编程实现飞行摇杆数据到飞控模型的数据传入，然后，通过建立的飞控模型，求解四旋翼无人机的飞行姿态参数，最后，传递给 FlightGear 进行视景模拟与仿真。

第**??**章：讲述固定翼无人机的飞控建模思路，第一步都是需要实现飞行摇杆的数据读入过程，固定翼无人机的飞控模型采用 JSBSim 飞行动力学模型。通过模型求解，将飞机舵面参数，飞行姿态参数传递给 FlightGear 进行视景模拟与仿真。

第**??**章：通过前两章的建模与模型求解，在 FlightGear 飞行仿真平台上模拟仿真，实现飞行摇杆对无人机的飞行控制。同时，介绍 FlightGear 三维视景仿真系统的理论基础。

第**??**章：总结建模过程中的创新点与改进之处，提出进一步研究的方向与进度，对未来的研究工作提出展望。

坐标系转换与 FlightGear 介绍

2.1 坐标系转换 [?]

本文这一节主要介绍用于描述飞机姿态位置的坐标系与坐标系之间的转换，用来推导四旋翼飞控建模过程中需要用的旋转矩阵与转换方程。飞机的姿态描述需要利用各种坐标系表示的原因如下：

1. 空气动力学的力和力矩是在机体坐标系下表示的。
2. 牛顿运动方程利用四元数法来表示，需要指明相关的坐标系。
3. 在机体坐标系下，飞机传感器读取飞机姿态参数信息，比如加速度；相反，飞机上 GPS 系统读取的飞机位置姿态参数需要在惯性坐标系下（即地面坐标系）。
4. 大多数情况下，需要在惯性坐标系下，对飞机的飞行轨迹进行描述。另外，地图所采用的位置坐标也是在惯性坐标系下表示的。

要想完成一个坐标系到另一个坐标系之间的转换，需要完成两个步骤，一个是旋转，另一个是转换。在本章的**??**节部分用来描述坐标旋转矩阵和旋转矩阵应用于坐标系变化；在本章的**??**节部分用来介绍在四旋翼无人机系统下所采用的特定坐标系，推导四旋翼旋转矩阵表达式；在本章的**??**节部分用来推导坐标系转换过程中转换关系，即 Corlios 方程。最终，推导完成坐标系转换过程的两个步骤：旋转和转换。

2.1.1 旋转矩阵理论基础

Figure 2.1: 坐标系二维旋转 [?]

如图**??**所示，向量 **p** 即在坐标系 R^0 下同时还在坐标系 R^1 下，R^0 坐标系与 R^1 坐标系的坐标原点相同，故向量 **p** 从坐标系 R^0 到坐标系 R^1 的旋转，属于二维空间的旋转。在坐标系 R^0 下，单位向量用 $\left(\hat{i}^0, \hat{j}^0, \hat{k}^0\right)$ 表示，在坐标系 R^1 下, 单位向量用 $\left(\hat{i}^1, \hat{j}^1, \hat{k}^1\right)$ 表示。在坐标系 R^0 下，向量 **p** 可以表示为：

$$\mathbf{p} = p_x^0 \hat{i}^0 + p_y^0 \hat{j}^0 + p_z^0 \hat{k}^0$$

同样，在坐标系系 R^1 下, 向量 $\mathbf{p}$ 可以表示为：

$$\mathbf{p} = p_x^1\,\hat{i}^1 + p_y^1\,\hat{j}^1 + p_z^1\,\hat{k}^1$$

联立上面两式，可以得到如下的关系：

$$p_x^0\,\hat{i}^0 + p_y^0\,\hat{j}^0 + p_z^0\,\hat{k}^0 = p_x^1\,\hat{i}^1 + p_y^1\,\hat{j}^1 + p_z^1\,\hat{k}^1$$

上式两端分别点乘向量 $\left(\hat{i}^1, \hat{j}^1, \hat{k}^1\right)$，通过化简可以表示为如下的形式：

$$\mathbf{p^1} \triangleq \begin{pmatrix} p_x^1 \\ p_y^1 \\ p_z^1 \end{pmatrix} = \begin{pmatrix} \hat{i}^1\cdot\hat{i}^0 & \hat{i}^1\cdot\hat{j}^0 & \hat{i}^1\cdot\hat{k}^0 \\ \hat{j}^1\cdot\hat{i}^0 & \hat{j}^1\cdot\hat{j}^0 & \hat{j}^1\cdot\hat{k}^0 \\ \hat{k}^1\cdot\hat{i}^0 & \hat{k}^1\cdot\hat{j}^0 & \hat{k}^1\cdot\hat{k}^0 \end{pmatrix} \begin{pmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{pmatrix}$$

通过上式，结合图**??**所示的几何关系，可以表示为如下的表达式：

$$\mathbf{P^1} = F_0^1\,\mathbf{P^0} \tag{2.1}$$

其中，矩阵表示为从坐标系 R^0 到坐标系 R^1 的坐标旋转矩阵,θ 表示从坐标系 R^0 的 x 轴逆时针旋转到坐标系 R^1 的角度，旋转矩阵 F_0^1 用 θ 角来表示，如下所示：

$$F_0^1 = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

同理，若坐标系 R^0 的 y 轴，按右手旋转法则转 θ 角得到坐标系 R^1，旋转矩阵 F_0^1 用 θ 角来表示:

$$F_0^1 \triangleq \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

同理，若坐标系 R^0 的 y 轴，按右手旋转法则转 θ 角得到坐标系 R^1，旋转

矩阵 F_0^1 用 θ 角来表示:

$$F_0^1 \triangleq \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

通过上面公式，我们可以得到旋转矩阵 F_0^1 重要的性质，如下所示：

- $\left(F_a^b\right)^{-1} = \left(F_a^b\right)^T = F_a^b$
- $F_b^c\, F_a^b = F_a^c$
- $\det F_a^b = 1$

Figure 2.2: 旋转变换 [?]

下面我们需要推导旋转公式，采用左手法则，如图**??**所示，向量 **p** 顺时针旋转 μ 得到向量 **q**,$\hat{n}$ 为单位向量。

$$q = O\vec{N} + N\vec{W} + W\vec{Q} \tag{2.2}$$

向量 $O\vec{N}$ 可以用向量 **p** 和单位向量 **n** 来表示，向量 **p** 在单位向量 **n** 方向投影，得到向量 $O\vec{N}$。

$$O\vec{N} = (\mathbf{p}\cdot\hat{n})\,\hat{n}$$

向量 $N\vec{W}$ 方向沿 $\vec{p} - O\vec{N}$, 长度由 $NQ\cos\mu$ 表示。由图**??**可知，NQ 长度与 NP 长度相同，可以用 $\left\|\vec{p} - O\vec{N}\right\|$ 来表示。

$$\begin{aligned} N\vec{W} &= \frac{\mathbf{p} - (\mathbf{p}\cdot\hat{n})\,\hat{n}}{\|\mathbf{p} - (\mathbf{p}\cdot\hat{n})\,\hat{n}\|} NQ\cos\mu \\ &= (\mathbf{p} - (\mathbf{p}\cdot\hat{n})\,\hat{n})\cos\mu \end{aligned}$$

向量 $W\vec{Q}$ 方向垂直向量 **p** 和 **n**，长度由 $NQ\sin\mu$ 表示。由图**??**可知，NQ 长度也可以用 $\|\vec{p}\|\sin\phi$ 表示。

$$\begin{aligned} W\vec{Q} &= \frac{\mathbf{p}\times\hat{n}}{\|\mathbf{p}\|\sin\phi} NQ\sin\mu \\ &= -\hat{n}\times\mathbf{p}\sin\mu \end{aligned}$$

因此，分别将向量 $O\vec{N}$ ，向量 $N\vec{W}$ 和向量 $W\vec{Q}$ 表达式代入向量 $\mathbf{q}$ 的表达式，得到旋转向量表达式，如下所示：

$$\mathbf{q} = (1 - \cos\mu)(\mathbf{p} \cdot \hat{n})\hat{n} + \cos\mu\mathbf{p} - \sin\mu(\hat{n} \times \mathbf{p}) \tag{2.3}$$

2.1.2 四旋翼旋转矩阵

在四旋翼飞行控制时，需要实现不同坐标系之间的转换，下面介绍常用的坐标系。

2.1.2.1 惯性坐标系 R^i

Figure 2.3: 惯性坐标系 [?]

惯性坐标系是指坐标原点位于地心，如图**??**所示，向量 $\hat{i}$ 指向北，向量 $\hat{j}$ 指向东，向量 $\hat{k}$ 指向地心。

2.1.2.2 速度坐标系 R^v

Figure 2.4: 速度坐标系 [?]

坐标系的原点位于四旋翼的重心位置，R^v 坐标系各轴的方向与惯性坐标系 R^i 方向保持一致，如图**??**所示。

2.1.2.3 偏航坐标系 R^{v1}

Figure 2.5: 偏航坐标系 [?]

R^{v1} 坐标系的原点位于机体重心位置，R^{v1} 坐标系是机体绕 $\hat{k}^v$ 轴偏航 ψ 角，从而得到的偏航坐标系，即 $\hat{i}$ 轴指向机头方向，$\hat{j}$ 轴指向右机翼方向，$\hat{k}$ 轴指向地心，如图**??**所示。

由坐标系 R^v 到坐标系 R^{v1} 的转换，旋转角为 ψ，利用前面推导的旋转矩阵公式可以得到：

$$\mathbf{p}^{v1} = F_v^{v1}(\psi)\,\mathbf{p}^v$$

其中，

$$F_v^{v1}(\psi) = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2.1.2.4 俯仰坐标系 R^{v2}

Figure 2.6: 俯仰坐标系 [?]

R^{v2} 坐标系原点位于机体重心位置，R^{v2} 坐标系是 R^{v} 坐标系绕 $\hat{j}^{v1}$ 轴俯仰 θ 角，从而得到的俯仰坐标系，即 $\hat{i}$ 轴指向机头方向，$\hat{j}$ 轴指向右机翼方向，$\hat{k}$ 轴指向机身下部。

由坐标系 R^{v1} 到坐标系 R^{v2} 的转换，旋转角为 θ, 可得：

$$\mathbf{p}^{v2} = F_{v1}^{v2}(\theta)\,\mathbf{p}^{v1}$$

其中，

$$F_{v1}^{v2}(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}$$

2.1.2.5 机体坐标系 R^{b}

Figure 2.7: 机体坐标系 [?]

机体坐标系通过 R^{v2} 绕 $\hat{i}^{v2}$ 轴滚转 ϕ 得到 R^{b}，如图**??**所示。

由坐标系 R^{v2} 到坐标系 R^{b} 的转换，旋转角为 ϕ，可得：

$$\mathbf{p}^{b} = F_{v2}^{b}(\phi)\,\mathbf{p}^{v2}$$

其中，

$$F_{v2}^{b}(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}$$

因此，由 R^v 到机体坐标系 R^b 转换，得到的坐标系旋转矩阵。

$$
\begin{aligned}
F_v^b(\phi,\theta,\psi) &= F_{v2}^b(\phi)\,F_{v1}^{v2}(\theta)\,F_v^{v1}(\psi) \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}
\begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}
\begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\psi c\theta \end{pmatrix}
\end{aligned}
$$

其中，$c\phi \triangleq \cos\phi$，$s\phi \triangleq \sin\phi$。

2.1.3 Coriolis 公式

在这一节，本文打算推导 Coriolis 数学关系式为后面飞控动力学关系作铺垫，Coriolis 公式在坐标系变换过程中，起到非常重要的作用。假设，我们有两个坐标系，机体坐标系 R^b 和惯性坐标系 R^i，如图**??** 所示，向量 $\mathbf{p}$ 相对坐标系 R^i 的运动，可以分解为两个运动，即：有向量 $\mathbf{p}$ 随着机体坐标系 R^b 的平动，和向量 $\mathbf{p}$ 相对 R^i 坐标系转动。下面总共两步推导 Coriolis 公式 [?]：

Figure 2.8: Coriolis 公式推导图 [?]

Step1:

首先，假设机体坐标系 R^b 不相对惯性坐标系 R^i 转动，向量 $\mathbf{p}$ 随着向量机体坐标系 R^b 平动，向量 $\mathbf{p}$ 相对惯性坐标系 R^i 运动等价于向量 $\mathbf{p}$ 相对机体坐标系 R^b 运动，如下所示：

$$\frac{d}{dt_i}\mathbf{p} = \frac{d}{dt_b}\mathbf{p} \tag{2.4}$$

Step2:

向量 $\mathbf{p}$ 静止在坐标系 R^b 内，同时，坐标系 R^b 相对坐标系 R^i 转动，因此，向量 $\mathbf{p}$ 相对 R^i 坐标系转动等效于坐标系 R^b 相对坐标系 R^i 转动，利用前面推导的旋转公式**??**，可得如下关系：

$$\vec{p} + \delta\vec{p} = (1 - \cos(-\delta\phi))\,\hat{s}\,(\hat{s}\cdot\vec{p}) + \cos(-\delta\phi)\,\vec{p} - \sin(-\delta\phi)\,\hat{s}\times\vec{p}$$

利用小角度近似，同时等式两边同时除以 δt，可得如下关系：

$$\frac{\delta p}{\delta t} \approx \frac{\delta \phi}{\delta t} \times \vec{p}$$

当 $\delta t \to 0$ 时，R^b 坐标系相对 R^i 坐标系的角速度可以表示为：$\omega_{b/i} \triangleq \hat{s}\dot{\phi}$，代入上式可得：

$$\frac{d}{dt}_i \mathbf{p} = \omega_{b/i} \times \mathbf{p} \tag{2.5}$$

因为，微分方程的线性性质，可以将公式**??**和公式**??**线性相加，便可得到 Coriolis 方程：

$$\frac{d}{dt_i}\mathbf{p} = \frac{d}{dt_b}\mathbf{p} + \omega_{b/i} \times \mathbf{p} \tag{2.6}$$

2.2 FlightGear 软件介绍

FlightGear 是一个开源的飞行器模拟软件，由飞行模拟爱好者和编程爱好者共同合作开发的一款通过互联网提供大家学习的软件。FlightGear 开发的原因一方面因为商业的飞行模拟软件价格高，受众群体小；另一方面，大多数商业的飞行模拟软件由于研发资金与开发时间的限制，并不能够提供一款功能全面的逼真的飞行模拟软件 [?]。

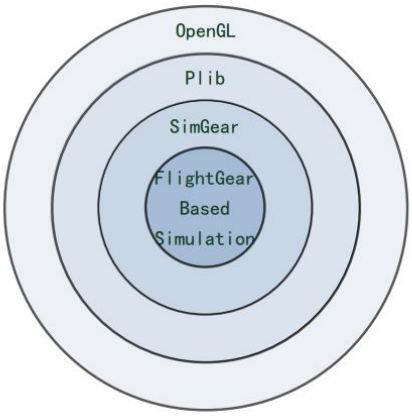


Figure 2.9: FlightGear 主要组件 [?]

FlightGear 作为一款大型的飞行模拟软件，结合了许多轻量级的开源软件。FlightGear 主要使用的软件之间的关系如图**??** 所示。下面分别介绍 OpenGL, Plib 和 SimGear 在 FlightGear 功能实现过程中的功能与作用。

- OpenGL

OpenGL 是 Open Graphics Library 的缩写，意为开放的图形库，是 FlightGear 最重要的组件。它是 SGI 公司开发的基于 FlightGear 的直升机飞行模拟系统研究的一套高性能图形处理系统，GL 代表图形库（Graphics Library），前身是 SGI 的 IRIX GL（工业标准 3D 计算机软件接口）[?]。OpenGL 是一个开放的三维图形软件包，具有强大图形处理功能，提供了基本库、实用库、辅助库，包含 100 多个函数，能够构建三维场景并对三维模型进行渲染着色，使三维模型更加逼真 [?,?]。

- SimGear
  SimGear 是仿真架构的开源工具集，是 FlightGear 的仿真引擎, 主要功能模块介绍如表**??**所示 [?]。

Table 2.1: SimGear 功能模块介绍

模块名称	功能描述
SGBucket	负责管理世界地图场景
SGDebug	用于程序调试，跟踪软件运行情况
SGEphem	管理星历表，以便准确渲染这些天体
SGIO	与底层 I/O 的接口驱动，如 socket、串行端口等
SGMagVar	利用地理坐标和时间计算磁变值和磁倾角，为飞行器助航
SGMath	提供矢量计算、坐标变换、随机数产生等数学函数
SGEnvironment	提供与天气环境相关的一些辅助类
SGMisc	提供混杂的一些工具库，如管理文件路径的 SGPath 类
SGNasal	提供 nasal“脚本”语言
SGProps	FlightGear 数据结构的核心功能
SGRoute	管理单个或一系列航路点，用于自动驾驶模块的导航
SGMaterial	管理场景图形的材质资源
SGModel	管理 3D 模型
SGSky	模拟真实飞行中的天空环境
SGSound	音效模块，对 OpenAL 进行了封装，管理所有的声音信息
SGTiming	时间分系统，计算管理各种时间参数
SGXML	解析 XML 文档

FlightGear 开发正是弥补以上的飞行模拟软件的缺陷，通过表**??**所示，强大的仿真引擎功能，从而提供更加友好的用户界面，满足飞行模拟爱好者对飞行的热爱。因此,FlightGear 具有以下的优势 [?]：

1. 跨平台性：FlightGear 支持 Linux, Windows, Mac OS X 等多个操作系

统，为了满足用户在多个操作系统或者平台上使用 FlightGear 飞行模拟软件。

2. 开源性：FlightGear 开源性，支持任何人对 FlightGear 软件的改善，同时，分享 FlightGear 软件的所有源代码文件，为科研学术研究提供更好的平台。
3. 可拓展性：FlightGear 开发者们意在提供一个可以满足任何用户需求的飞行模拟软件，与其他商业的飞行模拟软件不同，FlightGear 所有的文件都是免费提供给用户可见的，任何人都可以看 FlightGear 是如何编码开发的，目的就是让更多的用户使用这款飞行模拟软件。从设计之初开始，FlightGear 的场景地形、内部参数、API 函数和其它任何模块都对用户透明以及有文档记录。用户可以根据自身需求对 FlightGear 软件进行扩展设计，实现新的功能。

- Plib

  Plib(Portable Game Libraries) 也是一个开源库，由 Steve J. Baker 开发维护，包含 GUI、3D、音效、窗口管理等。Plib 包同样具有跨平台性，支持 Linux、Windows、Mac OS 等操作系统，各功能模块之间如表**??**所示 [?]。

2.2.1 FlightGear 程序架构

由上一节可以知道，FlightGear 以 SimGear 为仿真引擎，由 OpenGL 处理图像，Open AL 处理音频，Plib 实现跨平台，网络通信等辅助功能的飞行模拟软件。本节主要来介绍 FlightGear 的程序架构，分为两部分来介绍，一部分是框架结构，另一部分是运行流程，来了解 FlightGear 各个子系统之间的关系与功能。

2.2.1.1 FlightGear 框架结构

FlightGear 由飞行动力学系统，控制系统，飞行器模块，时间系统，场景系统等子系统构成，各个子系统之间的关系并不是相互独立，而是相互联系的，子系统之间的关系如**??**所示 [?]。

Figure 2.10: FlightGear 主要组件 [?]

2.2.2 FlightGear 运行流程

FlightGear 使用 C++ 语言开发，其运行流程如图**??**所示 [?]。由图**??**可知，FlightGear 运行流程主要有两个主循环构成，主循环 1 负责完成各个模块的初始化功能。初始化功能通过变量 $Idle\_State$ 累加来控制，随着控制参数

Table 2.3: Plib 功能模块介绍

模块名称	功能描述
UL(Utility Library)	隐藏操作系统之间的差异，使函数通用，为其他库提供辅助类或例行程序
SG(Standard Geometry Library)	为提高 OpenGL 运行效率而写的一个简单的矩阵和矢量数学库，在 Plib 其他部分中大量运用
JS(Joystick Wrappers)	封装操纵杆，支持更多操纵杆、轴、按钮
FNT(Font Library)	封装字体，将输入文本映射为纹理在 OpenGL 里绘制
SSG(Simple Scene Graph)	简单场景图形，设计为一个相对简单的、轻负载的构建在 OpenGL 层之上的场景图形 API
PUI (Picoscopic User Interface Library)	一组需要 OpenGL 和 C++ 操作的 GUI 窗口小部件（如菜单、按钮、组合框、标签、滚动条等），在 3D 硬件上的执行速度非常快
NET(Pegasus Network Library)	一个封装了联网功能的 C++ 库，在游戏中加入网络功能将变得非常容易，使用 NET 前，要首先调用 netInit 函数。

$Idle\_State$ 参数迭代累加的过程，完成各个模块的初始化，主要是助航系统初始化、三维飞行器模型、环境模型 (包括天空模型)、OpenGL 参数设置，SSG 模块的初始化。同时，主循环 1 过程中的窗口系统只要用来完成启动界面的渲染功能。

由图**??**可知，在主循环 2 的过程中，主要完成整个系统更新的功能，是 FlightGear 运行过程的核心部分，在该循环过程中的窗口系统则主要用来完成三维化可视仿真系统中的渲染功能 [?]，主要针对天空，飞行器，地面等场景的渲染。

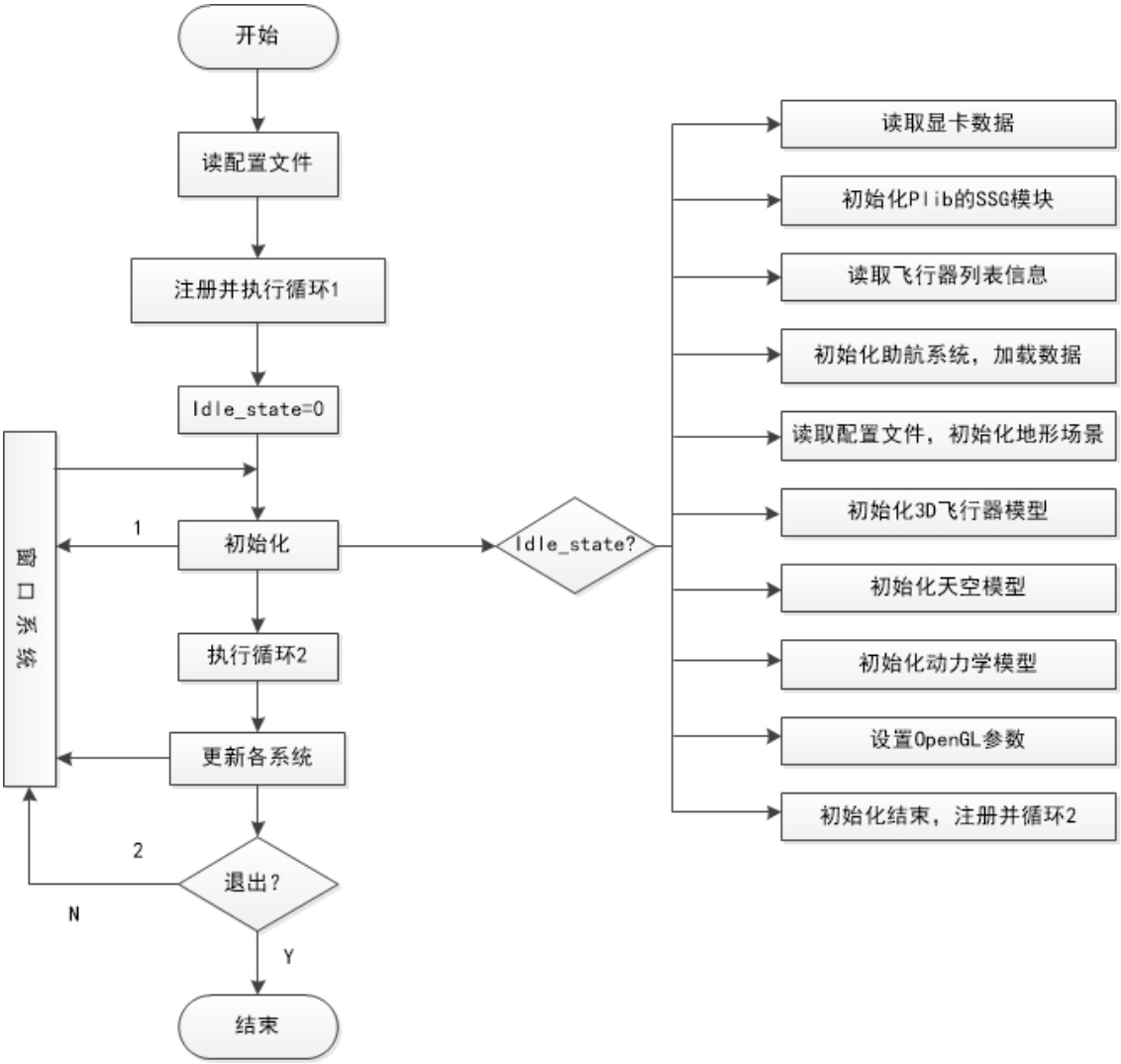


Figure 2.11: FlightGear 主要组件 [?]

四旋翼无人机飞控建模

在本章，本文分三部分介绍四旋翼的飞控建模过程，**??**节是飞行摇杆的数据传入过程，作为四旋翼无人机的姿态角的输入；**??**节是四旋翼无人机飞行动力学部分，也是本章的核心部分；**??**节是三维视景系统的设计原理。关系如图**??**所示。

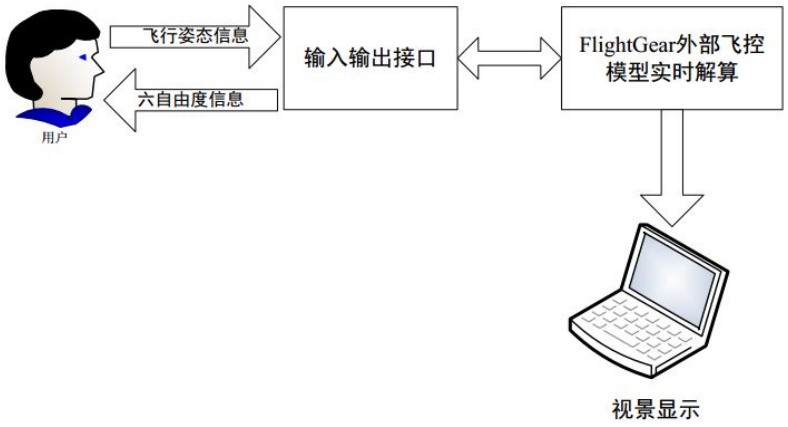


Figure 3.1: 四旋翼飞控建模过程

3.1 飞行摇杆数据传入

本文将飞行摇杆（Joystick）数据读取作为四旋翼飞控模型的输入，Joystick数据的数据主要是四旋翼的姿态角（滚转角，偏航角，俯仰角）和四旋翼的推力。因此，本文通过 C++ 编程，实现飞行摇杆数据的输入。由如下代码所示：

```
1  class HAL_JoyStick: public ZY::Thread {
2  public:
3      HAL_JoyStick(){
4          m_devID = -1;
5      }
6      HAL_JoyStick(int dev) {
7          m_devID=dev;
8      }
9      virtual ~HAL_JoyStick() {}
10     int open(int devID);
11     void run();
12     int close(void);
13     int read(JS_Val *jsv)
14     {
15         data_mutex.lock();
16         *jsv= m_JSVal;
17         m_JSVal.dataUpdated = 0;
18         data_mutex.unlock();
```

```
19          return 0;
20      }
21  public:
22      int          m_devType;
23      int          m_devID;
24      int          m_devFD;
25      char         number_of_axes;
26      char         number_of_btns;
27  protected:
28      ZY::Mutex    data_mutex;
29      JS_Val       m_JSVal;
30  }
```

3.2 四旋翼飞控建模 [?]

在本节中，分为三部分进行四旋翼飞行动力学建模的过程，第一部分是四旋翼飞行建模中需要的变量与坐标系的说明与定义；第二部分是推导六自由度四旋翼飞行动力学方程；第三部分是对模型进行化简，进行模型的求解。

3.2.1 四旋翼变量

Table 3.1: 四旋翼参数表

参数名称	参数意义
p_n	惯性坐标系 R^i 下，四旋翼 x 轴方向的位置
p_e	惯性坐标系 R^i 下，四旋翼 y 轴方向的位置
h	惯性坐标系 R^i 下，四旋翼 z 轴方向的高度
u	机体坐标系 R^b 下，四旋翼 x 轴方向的速度
v	机体坐标系 R^b 下，四旋翼 y 轴方向的速度
ω	机体坐标系 R^b 下，四旋翼 z 轴方向的速度
ϕ	滚转角
θ	俯仰角
ψ	偏航角
p	机体坐标系 R^b 下，滚转角速率
q	机体坐标系 R^b 下，俯仰角速率
r	机体坐标系 R^b 下，偏航角速率

如表**??**所示，四旋翼建模过程中所需的参数变量，将上述变量关系在图**??**中所示。四旋翼姿态位置变量 (p_n, p_e, h) 是定义在惯性坐标系（即地面坐标系）下，其中，高度变量 h 在惯性坐标系下沿着 z 轴的负半轴方向，即方向向

上。在机体坐标系下，定义四旋翼的速度变量 (u, v, ω) 和角速度变量 (p, q, r)。欧拉角（滚转角 ϕ，俯仰角 θ，ψ）分别定义与上章的滚转坐标系，俯仰坐标系和偏航坐标系下。

Figure 3.2: 坐标轴定义

3.2.2 四旋翼六自由度飞行动力学建模

3.2.2.1 四旋翼运动学方程

在地面坐标系下，定义四旋翼姿态位置变量 (p_n, p_e, h)；在机体坐标系下，定义四旋翼速度变量 (u, v, ω)。因此，需要建立位置变量与姿态变量之间的关系，结合 2.1 节坐标系转换的公式可得：

$$
\begin{aligned}
\frac{d}{dt}\begin{pmatrix} p_n \\ p_e \\ -h \end{pmatrix} &= F_b^v \begin{pmatrix} u \\ v \\ \omega \end{pmatrix} \\
&= (F_b^v)^T \begin{pmatrix} u \\ v \\ \omega \end{pmatrix} \\
&= \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\theta s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\theta c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ \omega \end{pmatrix}
\end{aligned}
$$

其中，$c = \cos$, $s = \sin$。

同样，欧拉角（滚转角 ϕ，俯仰角 θ，ψ）和角速度 (p, q, r) 之间也需要坐标系的转换。但是，它们之间坐标系的转换是复杂的，角速度变量被定义于机体坐标系，但是，欧拉角分别定义于三个不同的坐标系，即滚转角坐标系，俯仰角坐标系和偏航角坐标系（2.1 节所示)。于是，我们需要建立起角速度 (p, q, r) 和欧拉角速率 $\left(\dot{\phi}, \dot{\theta}, \dot{\psi}\right)$ 之间的关系。因为欧拉角速率均为小量，可以近似于欧拉角与角速度之间的关系。

$$
F_{v2}^b \left(\dot{\phi}\right) = F_{v1}^{v2} \left(\dot{\theta}\right) = F_v^{v1} \left(\dot{\psi}\right)
$$

因此，我们可以得到：

$$\begin{aligned}
\begin{pmatrix} p \\ q \\ r \end{pmatrix} &= R_{v2}^{b}\begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + R_{v2}^{b}(\phi) R_{v1}^{v2}\begin{pmatrix} \dot{\theta} \\ 0 \\ 0 \end{pmatrix} + R_{v2}^{b}(\phi) R_{v1}^{v2}\begin{pmatrix} \dot{\theta} \end{pmatrix} R_{v\to v1}\begin{pmatrix} \dot{\psi} \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \\
&= \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}\begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}\begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & -s\theta \\ 0 & \mathrm{c}\phi & \mathrm{s}\phi c\theta \\ 0 & -\mathrm{s}\phi & c\phi c\theta \end{pmatrix}\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}
\end{aligned} \tag{3.1}$$

通过，对上式求逆可得：

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{pmatrix}\begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3.2}$$

在惯性坐标系下，根据牛顿第二定律的微分形式，可以得到四旋翼的速度与力之间的关系，如下所示：

$$m\frac{d\mathbf{v}}{dt_i} = \mathbf{f}$$

其中，m 是四旋翼的质量，$\mathbf{f}$ 是四旋翼的合力，$\frac{d}{dt_i}$ 是四旋翼在惯性坐标系下的时间导数。根据 2.1.3 节的 Coriolis 公式可得：

$$m\frac{d\mathbf{v}}{dt_i} = m\left(\frac{d\mathbf{v}}{dt_b} + \omega_{b/i}\times\mathbf{v}\right) = \mathbf{f} \tag{3.3}$$

其中，$\omega_{b/i}$ 是在惯性坐标系下的角速度，因为四旋翼的受力是在机体坐标系下计算的，所以，ω 在机体坐标系下进行测量，利用上式，结合 $\mathbf{v}^b \triangleq (u,v,\omega)^T$ 和 $\omega_{b/i} \triangleq (p,q,r)^T$

关系式以及 $\mathbf{f}^b \triangleq (f_x, f_y, f_z)^T$ 上式可以表示为：

$$\begin{pmatrix} \dot{u} \\ \dot{\nu} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} rv - q\omega \\ p\omega - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \tag{3.4}$$

对于旋转运动而言，由牛顿第二定律可得：

$$\frac{d\mathbf{h}^b}{dt_i} = m \tag{3.5}$$

其中，h 代表角动量，m 是力矩，利用 Coriolis 公式可得：

$$\frac{d\mathbf{h}}{dt_i} = \frac{d\mathbf{h}}{dt_b} + \omega_{b/i} \times \mathbf{h} = \mathbf{m}$$

将上式在机体坐标系下，进行求解，其中，$\mathbf{h}^b = \mathbf{J}\omega_{b/i}^i$,$\mathbf{J}$ 是常量的惯性矩阵。

$$\begin{aligned} \mathbf{J} &= \begin{pmatrix} \int \left(y^2 + z^2\right) dm & -\int xydm & -\int xzdm \\ -\int xydm & \int \left(x^2 + z^2\right) dm & -\int yzdm \\ -\int xzdm & -\int yzdm & \int \left(x^2 + y^2\right) dm \end{pmatrix} \\ &\triangleq \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix} \end{aligned}$$

如图**??**所示，四旋翼关于三个坐标轴几何对称性，因此 $J_{xy} = J_{xz} = J_{yz} = 0$, 可得

Figure 3.3: 四旋翼力矩图 [?]

$$J = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}$$

因此，

$$J^{-1}=\begin{pmatrix}\frac{1}{J_x}&0&0\\0&\frac{1}{J_y}&0\\0&0&\frac{1}{J_z}\end{pmatrix}$$

如图**??**所示，球体的转动惯量为 $J=\frac{2}{5}MR^2$, 从而就可得到：

$$\begin{aligned}J_x&=\tfrac{2}{5}MR^2+2l^2m\\J_y&=\tfrac{2}{5}MR^2+2l^2m\\J_z&=\tfrac{2}{5}MR^2+4l^2m\end{aligned}$$

定义力矩 $m^b\triangleq(\tau_\phi,\tau_\theta,\tau_\psi)^T$, 代入公式（3.5）可得：

$$\begin{aligned}\begin{pmatrix}\dot{p}\\\dot{q}\\\dot{r}\end{pmatrix}&=\begin{pmatrix}\frac{1}{J_x}&0&0\\0&\frac{1}{J_y}&0\\0&0&\frac{1}{J_z}\end{pmatrix}\left[\begin{pmatrix}0&r&-q\\-r&0&p\\q&-p&0\end{pmatrix}\begin{pmatrix}J_x&0&0\\0&J_y&0\\0&0&J_z\end{pmatrix}\begin{pmatrix}p\\q\\r\end{pmatrix}+\begin{pmatrix}\tau_\phi\\\tau_\theta\\\tau_\psi\end{pmatrix}\right]\\&=\begin{pmatrix}\frac{J_y-J_z}{J_x}qr\\\frac{J_z-J_x}{J_y}pr\\\frac{J_x-J_y}{J_z}pq\end{pmatrix}+\begin{pmatrix}\frac{1}{J_x}\tau_\phi\\\frac{1}{J_y}\tau_\theta\\\frac{1}{J_z}\tau_\psi\end{pmatrix}\end{aligned}$$

最后，六自由度飞行动力学方程如下所示。

$$\begin{pmatrix}\dot{p_n}\\\dot{p_e}\\h\end{pmatrix}=\begin{pmatrix}c\theta c\psi&s\phi s\theta c\psi-c\phi s\psi&c\phi s\theta c\psi+s\phi s\psi\\c\theta s\psi&s\phi s\theta s\psi+c\phi c\psi&c\phi s\theta s\psi-s\phi c\psi\\s\theta&-s\phi c\theta&-c\phi c\theta\end{pmatrix}\begin{pmatrix}u\\v\\\omega\end{pmatrix}\tag{3.6}$$

$$\begin{pmatrix}\dot{u}\\\dot{v}\\\dot{\omega}\end{pmatrix}=\begin{pmatrix}rv-q\omega\\p\omega-ru\\qu-pv\end{pmatrix}+\frac{1}{m}\begin{pmatrix}f_x\\f_y\\f_z\end{pmatrix}\tag{3.7}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3.8}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix} \tag{3.9}$$

3.2.2.2 四旋翼力和力矩方程

对于四旋翼而言，因为没有空气动力学的升力面，所以，四旋翼的气动力和气动力矩是可以忽略的。四旋翼的力和力矩主要来自于重力和四个螺旋桨的推力。如图**??**所示。

Figure 3.4: 四旋翼力和力矩俯视图 [?]

Figure 3.5: 四旋翼力和力矩俯视图 [?]

由图**??**所示，是从四旋翼的俯视图来得到的力和力矩图。图**??**所示的四旋翼力和力矩的受力简化图，每个螺旋剑电机产生力和力矩。作用于四旋翼上的力可以表示为：

$$F = F_f + F_r + F_b + F_l$$

如图**??**所示，左右电机产生的滚转力矩可以表示为：

$$\tau_\phi = l\left(F_l - F_r\right)$$

同理，前后电机产生的俯仰力矩可以表示为：

$$\tau_\theta = l\left(F_f - F_b\right)$$

由牛顿第三定律可知，阻力产生的偏航力矩反作用于四旋翼，方向与四旋翼的旋转方向相反。因此，四旋翼的旋转矩阵可以表示为：

$$\tau_\psi = \tau_r + \tau_l - \tau_f - \tau_b$$

由螺旋桨产生的升力和阻力，与角速度的平方成线性关系，如下所示：

$$F_*=k_1\delta_*$$

$$\tau_* = k_2\delta_*$$

其中，系数 k_1 和 k_2 由实验可以测得，δ_* 电机控制的信号，$*$ 表示 f,r,b,l 电机。作用在四旋翼上力和力矩的矩阵如下所示：

$$\begin{pmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} k_1 & k_1 & k_1 & k_1 \\ 0 & -lk_1 & 0 & lk_1 \\ lk_1 & 0 & lk_1 & 0 \\ -k_2 & k_2 & -k_2 & k_2 \end{pmatrix} \begin{pmatrix} \delta_f \\ \delta_r \\ \delta_b \\ \delta_l \end{pmatrix} \triangleq M \begin{pmatrix} \delta_f \\ \delta_r \\ \delta_b \\ \delta_l \end{pmatrix}$$

四旋翼除了受电机产生的作用力外，还有自身的重力，在速度坐标系 R^v 下，重力恰好作用与坐标系的重心位置，故：

$$f_g^v = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix}$$

但是，速度是在机体坐标系下表示的，故上式需转换在机体坐标系下：

$$\begin{aligned} f_g^b &= R_v^b \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} \\ &= \begin{pmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \end{pmatrix} \end{aligned}$$

最终，四旋翼的六自由度飞行动力学模型如下：

$$\begin{pmatrix} \dot{p_n} \\ \dot{p_e} \\ h \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ \omega \end{pmatrix} \tag{3.10}$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} rv - q\omega \\ p\omega - ru \\ qu - pv \end{pmatrix} + \begin{pmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{pmatrix} + \frac{1}{m}\begin{pmatrix} 0 \\ 0 \\ -F \end{pmatrix} \tag{3.11}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3.12}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix} \tag{3.13}$$

3.2.3 模型求解

六自由度的四旋翼飞行动力学方程太复杂对于模型求解而言。本文基于小扰动假设，即果运动参数偏离基准状态的量为小量，可以略去二阶以上的运动参数变化量。在线性化的过程中，可以把无人机运动近似的分解为纵向运动和横向运动，两种运动相互独立。纵向运动即是飞机的俯仰运动；横向运动对应的是飞机的偏航与滚转运动。

假设 ϕ 和 θ 都是小量可以忽略，故公式 (3.12) 可以简化为：

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

同理，假设 Coriolis 方程中 qr,pr,pq 均为小量可以忽略，故公式（3.13) 可以简化为：

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix}$$

结合上面两式，可得：

$$\begin{pmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix} \tag{3.14}$$

对公式（3.10）求微分可得：

$$\begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi c\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix}$$

对公式（3.11）忽略 Coriolis 量，代入上式可得：

$$\begin{pmatrix} \ddot{p_n} \\ \ddot{p_e} \\ \ddot{p_d} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + \begin{pmatrix} -c\phi s\theta c\psi - s\phi s\psi \\ -c\phi s\theta s\psi + s\phi c\psi \\ -c\phi c\theta \end{pmatrix} \frac{F}{m}$$

因此，在惯性坐标系下，简化的方程如下：

$$\ddot{p_n} = (-\cos\phi\sin\theta\cos\psi - \sin\phi\sin\psi)\frac{F}{m} \tag{3.15}$$

$$\ddot{p_e} = (-\cos\phi\sin\theta sin\psi + \sin\phi\cos\psi)\frac{F}{m} \tag{3.16}$$

$$\ddot{p_d} = g - (\cos\phi\cos\theta)\frac{F}{m} \tag{3.17}$$

$$\ddot{\phi} = \frac{1}{J_x}\tau_\phi \tag{3.18}$$

$$\ddot{\theta} = \frac{1}{J_y}\tau_\theta \tag{3.19}$$

$$\ddot{\psi} = \frac{1}{J_z}\tau_\psi \tag{3.20}$$

3.3 三维视景系统

三维可视仿真系统的结构如图 **??**所示，主要包括仿真模块、飞行数据模块、通信模块、飞机模型库、纹理材质库等。

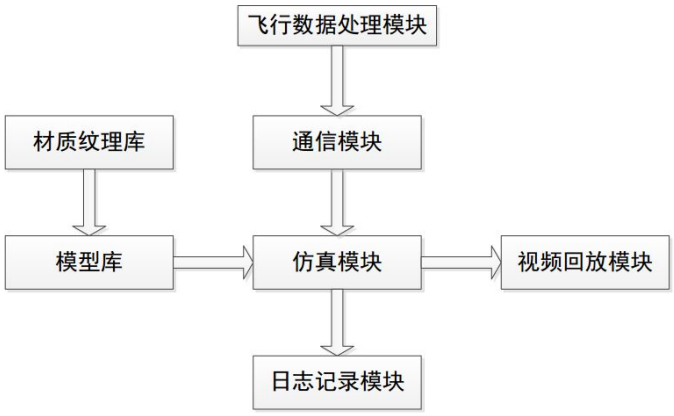


Figure 3.6: 三维可视化仿真结构图 [?]

仿真模块是三维可视仿真系统的核心模块。一般使用高级程序语言（如 C++）编写，也可以调用第三方三维可视仿真软件实现。仿真模块主要完成以下功能 [?]：

- 接收通信模块传来的飞行数据。
- 从模型库中导入飞行器模型、场景模型、声音模型，并对飞行器进行姿态和位置调整。
- 驱动飞行器模型按照飞行数据在场景中进行模拟飞行。

模型库为仿真系统提供飞行模型，包括飞行器模型、场景模型、建筑物模型、声音模型等，其中飞行器模型最为重要。并从材质纹理库读取纹理介质，将纹理介质贴于飞行器或场景表面，使模型更加美观、逼真 [?]。

材质纹理库为模型库中的各种模型的表面提供纹理介质，主要起美化作用。

飞行数据模块中的数据可以来自实时的飞行数据，也可以来自飞行动力学模型的模拟数据（如使用 matlab/simlink 构建动力学模型进行模拟），或者离线的外部数据 [?]。

通信模块负责飞行数据模块与仿真模块之间的通信，一般使用 socket 编程实现，飞行数据模块作为客户端，仿真模块作为服务端。在数据链路层还可以使用循环冗余校验码对飞行数据进行检验 [?]。

日志记录/回放模块用来记录模拟飞行时飞行器的位置和姿态信息，进行仿真过程回放，以便对某次或者某段仿真结果进行分析，查找可能存在的问题 [?]。

固定翼无人机飞控建模

本文进行固定翼无人机飞控建模，选择 c172p 飞机。固定翼飞机与四旋翼飞机在空气动力学特性等方面存在着很多不同。本文采用 FlightGear 自带的 JSBSim 飞行动力学模型，进行固定翼飞机的飞行控制，如图**??**所示。

Figure 4.1: 固定翼飞控建模过程

4.1 JSBSim 介绍

JSBsim 是一个通用的 6 自由度动态模型，模拟飞行工具的运动。它使用 C++ 语言编写，可以运行在单机方式下，也可以驱动有视觉子系统的大型程序。YAsim 是由 Andrew Ross 开发的新动力学模型，是 Flightgear 的集成部分，它通过模拟飞行器不同部分的气流来实现，这点不同于 JSBsim。UIUC 飞行动态模型是基于 LaRCsim 的，最初是 NASA 开发的，通过使用飞行器配置文件来扩充代码 [?]。目前 JSBsim [?] 和 UIUC 是比较流行的动力学仿真系统，并且均是开源项目。

4.2 JSBSim 飞行动力学模型

根据 FlightGear 自带的 JSBSim 飞行控制模型，进行飞行控制。包括飞机舵面的控制，飞机姿态的控制。

- 升降舵控制

  升降舵主要控制飞机的俯仰姿态运动，在 JSBSim 中对升降舵控制的代码如下：

```
<flight_control name="FCS:c172p">
<channel name="All">
    <summer name="Pitch Trim Sum">
        <input>fcs/elevator-cmd-norm</input>
        <input>fcs/pitch-trim-cmd-norm</input>
        <clipto>
            <min>-1</min>
            <max>1</max>
        </clipto>
</summer>
<aerosurface_scale name="Elevator Control">
    <input>fcs/pitch-trim-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.3</max>
        </range>
    <output>fcs/elevator-pos-rad</output>
</aerosurface_scale>
```

- 方向舵控制

  方向舵主要控制飞机的偏航运动，在小飞机转弯的过程中，起着重要的作用。JSBSim 的代码如下：

```
<summer name="Rudder Command Sum">
    <input>fcs/rudder-cmd-norm</input>
    <input>fcs/yaw-trim-cmd-norm</input>
    <clipto>
        min>-1</min>
        <max>1</max>
    </clipto>
</summer>
<aerosurface_scale name="Rudder Control">
    <input>fcs/rudder-command-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.35</max>
        </range>
    <output>fcs/rudder-pos-rad</output>
</aerosurface_scale>
```

- 副翼控制

  副翼主要控制飞机的滚转运动，副翼分为左右副翼，JSBSim 控制代码如下：

```
<summer name="Roll Trim Sum">
    <input>fcs/aileron-cmd-norm</input>
    <input>fcs/roll-trim-cmd-norm</input>
        <clipto>
            <min>-1</min>
            <max>1</max>
        </clipto>
</summer>
<aerosurface_scale name="Left Aileron Control">
    <input>fcs/roll-trim-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.35</max>
        </range>
    <output>fcs/left-aileron-pos-rad</output>
</aerosurface_scale>
<aerosurface_scale name="Right Aileron Control">
    <input>-fcs/roll-trim-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.35</max>
        </range>
    <output>fcs/right-aileron-pos-rad</output>
</aerosurface_scale>
```

本文选择 c172p 飞机，如图**??**所示，结合上面 JSBsim 模型对 c172p 进行飞行控制。

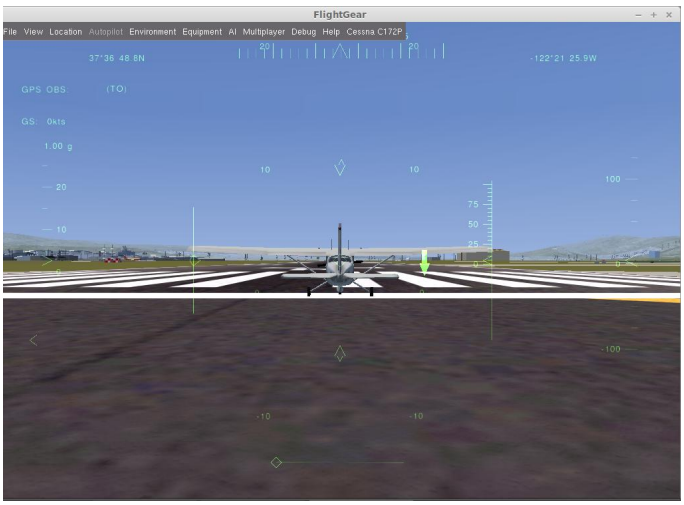


Figure 4.2: 固定翼飞控建模过程

FlightGear 仿真效果

本章主要介绍 FLightGear 仿真效果，首先，是介绍 FlightGear 驱动飞行器模拟的步骤；然后，是介绍 FlightGear 系统实施的软硬件环境；最后，是展示四旋翼飞行仿真效果。

5.1 FlightGear 飞行模拟器驱动

FlightGear 实现对飞行模拟器的驱动，需要经过飞行模拟器载入及配置的过程，然后，需要设置 FlightGear 通信模块，最后，进行飞行数据的处理。FlightGear 使用 xml 文件对模型进行配置，包括飞行器的位置，动作及姿态 [?]。

1. 载入飞行器模型，主要是通过-set.xml 文件指定使用的飞行器模型、场景模型和声音模型等;
2. 根据载入效果，通过配置文件调整飞行器位置及姿态；
3. 配置飞行器需要完成的动作，对于四旋翼无人机来说，主要是螺旋桨的旋转。

5.1.1 飞行模拟器载入及配置

基于 FlightGear 良好的三维图像兼容特性，可以方便导入三维飞行器模型。在 FlightGear 中 FlightGear 飞行模型的主配置文件是 XML 类型文件，命名规则为“飞行器模型名-set.xml”，因此每个飞行器模型都必须有一个独一无二的名字，本系统中的四旋翼无人机模型命名为 Quarotor，主配置文件命名为 quarotor-set.xml，该配置文件主要完成以下配置：

1. 指定飞行器模型 [?]

   使用/sim/model/path 参数指定飞行器模型的配置文件，该配置文件也是 XML 格式，一般习惯以飞行器模型的名称直接命名，但不是必须的。相关配置语句如下：

```
<sim>
  <model>
   <path>Aircraft/arducopter/Models/arducopter.xml</path>
  </model>
</sim>
```

   quadrotor.xml 指定用到的飞行器模型，如本系统使用的四旋翼无人机模型 quadrotor.ac 文件，该文件与 quadrotor.xml 位于同一目录下。quadrotor.xml 最主要的作用其实是配置飞行器模型需要完成的各项动作, 相关配置语句如下：

```
<PropertyList>
   <path>quadrotor.ac</path>
</PropertyList>
```

2. 指定声音模型 [?]

使用/sim/sound/path 参数指定仿真飞行时使用的声音配置文件，四旋翼无人机飞行声音主要是螺旋桨旋转的轰鸣声、风声，没有特殊声音，可以使用 FlightGear 提供的基本声音配置文件，具体配置代码如下：

```
<sound>
      <path>Aircraft/Generic/generic-sound.xml</path>
</sound>
```

3. 指定飞行动力学模型 [?]

本系统使用 FlightGear 进行虚拟现实的显示，飞行器模型的飞行通过动力学模型控制。动力学系统通过解算动力学模型，将飞行模型的姿态和位置等信息参数通过通信模块发送到飞行控制系统，飞行控制系统将信息参数转换成控制命令传递到 FlightGear 进行视景更新。

使用哪种飞行动力学模型可以使用两种方式进行指定，一种仍然是通过主配置文件，使用/sim/flight-model 参数，比如指定使用 JSBsim 动力学模型的代码如下：

```
<sim>
    <flight-model>jsb</flight-model>
</sim>
```

另一种方法是在启动 FlightGear 时直接使用命令行参数指定，格式如“–fdm=jsb”。除了可以选用 FlightGear 提供的动力学模型，FlightGear 也支持用户自己编写动力学模型，此时使用“–fdm=external”命令，此命令通知 FlightGear 接受外部数据。

4. 指定飞行器动作参数

为了增加仿真效果的逼真度，飞行器的某些部位应该能完成相应的动作，如螺旋桨旋转、起落架的放下/收起、驾驶舱门的开启/关闭、风向舵的偏转等。这些动作的完成都需要配置相关的动作参数。四旋翼无人机的主要动作是四个螺旋桨的旋转，需要配置螺旋桨的转速，配置代码如下：

```
<engines>
<engine>
   <rpm type="double">350</rpm>
</engine>
</engines>
```

代码中 rpm 代表每分钟旋转的次数，其值被设置为 350，类型为 double（双精度数字）。此配置方法在使用“–fdm=external”命令时无效，此时如果需要使用 rpm 参数则需要在 FlightGear 通信模型中设置相应变量。

飞行器模型的动作需要使用此参数时，使用“/engines/engine[0]/rpm”字符串，具体配置方式如下：

```
<property>/engines/engine[0]/rpm</property>
```

5.1.2 飞行器模型的动作

飞行器模型需要在仿真飞行时完成一定的动作，如螺旋桨旋转。FlightGear 允许用户控制飞行器的任意部位完成相应的工作，唯一要求就是这一部位在三维模型中被命名过，即设置了对象名称（object name)。

FlightGear 现在主要支持以下几种类型的动作：none、billboard、rotate、scale、blend、select、spin、timed、translate、texrotate、textranslate、textmultiple、material、range、alpha-test、noshadow。四旋翼无人机模型主要用的了 noshadow、select、spin、spin 四种类型。下面以其中一个旋翼的动作为例结合实际代码介绍这四种动作类型的含义及用法，其他三个旋翼与之类似：

- Noshadow:

  无阴影动作类型，使该对象屏蔽投射阴影的命令，如螺旋桨的阴影本身已经是阴影，不能再投射出阴影。具体用法：使用 type 标签指定动作类型 noshadow，使用 object-name 标签指定该动作的作用对象，该标签的属性值必须与三维模型中设置的对象名称完全一致。具体代码如下：

```
<animation>
  <type>noshadow</type>
  <object-name>Propeller1.Spinning</object-name>
</animation>
```

- Select

  选择动作类型，根据配置的条件决定选择哪个对象显示。当螺旋桨转速低于一定值的时候显示螺旋桨，当转速高于一定值后显示螺旋桨阴影，用于无人机启动或熄火时制造螺旋桨加速、减速的效果。property 标签的属性值必须是 quarotor-set.xml 主配置文件配置过的动作参数，具体代码如下：

```
<animation>
 <type>select</type>
  <object-name>Propeller1</object-name>
   <condition>
     <less-than>
       <property>/engines/engine[0]/rpm</property>
       <value>350</value>
     </less-than>
   </condition>
</animation>
```

- Spin

  旋转动作类型，是无人机模型最主要的动作类型，其控制螺旋桨旋转中心、旋转轴。property 标签指定转速，center 标签指定旋转中心，axis 标签指定旋转轴。旋转中心使用的坐标系为机体坐标系，具体坐标应是三维无人机模型中的实际坐标，具体代码如下：

```
<animation>
   <type>spin</type>
   <object-name>propeller0</object-name>
   <property>/controls/engines/engine[0]/throttle</property>
   <factor>12000</factor>
   <axis>
     <x1-m>0.000</x1-m>
     <y1-m>-0.288</y1-m>
     <z1-m>0.046</z1-m>
     <x2-m>0.000</x2-m>
     <y2-m>-0.288</y2-m>
     <z2-m>0.012</z2-m>
   </axis>
 </animation>
```

5.1.3 通信模块

在飞行器模型载入 FlightGear 并完成相应的配置之后，下一步就是驱动飞行器模型进行仿真了。本节将介绍 FlightGear 使用的通信模型，并主要介绍如何实现外部模拟数据与 FlightGear 进行通信来驱动飞行器模型。

1. FlightGear 通信模型

   本文说的通信模型指的是通信双方都认可的通信格式 [?]。要与 FlightGear 通信，必须使用 FlightGear 的通信模型 FlightGear 的通信模型为了保证通用性，包含了飞行器模型飞行时需要的所有参数信息。本文在进行四旋翼无人机的三维可视仿真时只用到部分参数，主要参数如表**??**所示。

Table 5.1: 四旋翼无人机飞行需要的主要参数

参数名称	参数意义
FG_NET_FDM_VERSION	通信模型版本号，同一版本的通信模型才能通信
longitude	经度，单位是度
latitude	纬度，单位是度
altitude	海拔，单位是英尺
ϕ	滚转角，从后向前视角的顺时针旋转为正。
θ	俯仰角，从左向右视角的顺时针旋转为正。
ψ	偏航角，从上向下视角的顺时针旋转为正。
eng_state	发动机状态（0：关闭，1：启动，2：运行）
rpm	螺旋桨转速

2. 通信模块的实现

通信模块用于连接飞行数据模块和仿真模块，完成数据的接收和发送，如图**??**。驱动飞行器模型进行仿真的方式主要有动力学模型驱动、外部仿真数据驱动、真实飞行器的实时数据驱动。不管使用哪种方式，都涉及到如何与 FlightGear 进行通信，但是如果每一种驱动方式都编写各自的通信模型，肯定不是明智的选择。本文将基于 Socket（套接字）为不同的驱动方式编写统一接口，实现对通信方式的模块化封装，使系统能够灵活的更换驱动方式。

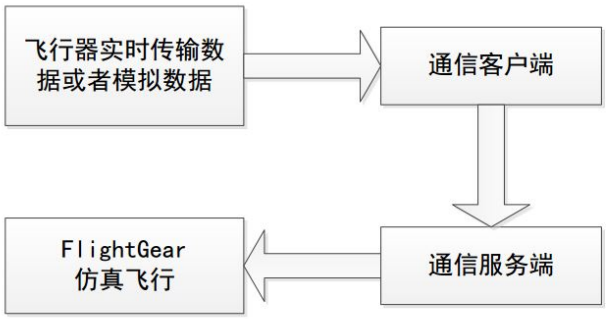


Figure 5.1: 通信模块的作用

为了实现在网络中传输数据，必须使用网络通信协议。TIP/IP 协议是当前计算机网络中使用最多的网络通信协议，TCP/IP 的核心部分由网络操作系统的内核实现，应用程序通过编程接口来访问 TCP/IP[?]。Socket 是介于网络应用层和传输层之间的编程接口，套接字接口提供了访问下层通信协议的大量系统调用函数和相应的数据结构 [?]，Socket 在通信协议中的地位**??**所示。

TCP/IP 协议的体系结构如图**??**所示 [?]，其中网络层有两种通信协议 TCP 和 UDP：

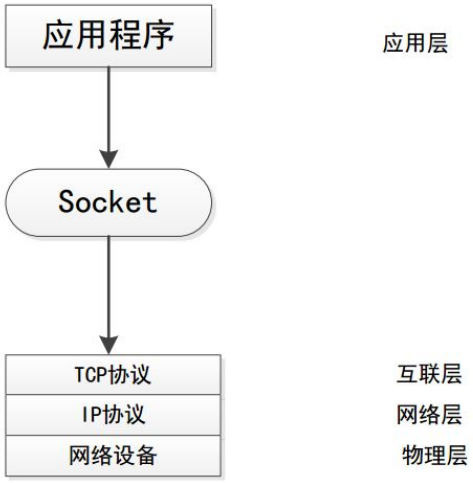


Figure 5.2: Socket 套接字在 TCP/IP 通信模型中的地位

Figure 5.3: TCP/IP 协议框架

FlightGear 支持使用 TCP 和 UDP 两种传输协议进行通信，出于以下几点考虑，本文使用基于数据报套接字的 UDP 协议开发通信模块：

- 行模拟对实时性要求较高，UDP 方式不用建立连接通道，不对数据进行校验，传输速率较快，能最大程度的保证飞行模拟的实时性；
- 飞行模拟在画面渲染和模拟数据的传输方面对系统资源的开销都很大，UDP 方式没有 TCP 方式对数据质量的控制功能，能降低系统对硬件的要求，最大程度的保证仿真效果；
- 飞行模拟对数据的可靠行要求不是非常高，由于前后帧的模拟数据相差很小，丢失一帧甚至几帧的数据不会对模拟结果产生较大影响，因此虽然 UDP 方式不保证数据无差错，但是使用 UDP 方式的差错率是可以容忍的。

通信模块分为客户端和服务器端，客户端和服务器端可以部署于同一台机器，也可以部署到不同的机器上。在网络中进行通信至少需要一对套接字，其中一个运行于客户端，称之为 ClientSocket；另一个运行于服务器端，称之为 ServerSocket[?,?]。本系统使用 UDP 传输协议实现数据传输的流程图**??**。

Figure 5.4: UDP 协议下的客户端与服务端通信流程

服务器端进程依次进行如下操作：

- 调用 socket() 方法创建一个数据报套接字 (SOCK_DGRAM)；
- 调用 bind() 方法将服务器地址绑定在该套接字上；
- 调用 recvfrom() 方法等待客户端发来的数据包，此时程序进入无限循环等待状态；
- 接受客户端传来的数据包，使用该数据包驱动飞行器模型完成相应的动作；
- 继续等待接受客户端的请求；
- 服务器端关闭，注销该套接字。

客户进程依次进行如下操作：

- 调用 socket() 方法创建一个数据报套接字 (SOCK_DGRAM)；
- 读取飞行数据，并根据服务器地址向服务器端发送数据报；
  数据来源可以是飞行器的实时飞行、动力学模型仿真，也可以是离线的外部数据，或者是 FlightGear 本身的日志记录文件。本文使用离线外部数据来测试三维可视仿真系统。以 C++ 文件流方式读取外部文件, 外部文件来源于 joystick 的数据读入部分。
- 数据处理。数据处理主要包括度量单位转换（如弧度与角度转换）、坐标系转换。FlightGear 使用测地学坐标系，如果外部数据是基于直角坐标系，则需要进行坐标转换。
- 继续发送下一条飞行数据;
  统通过基于 UDP 协议的 Socket 套接字进行发送，发送时间间隔有两种：
  一种是指定固定时间间隔进行发送，如 20 毫秒，即每隔 20 毫秒发送 1 次数据，发送时间间隔是固定的；
  另一种是根据外部数据的实际生成时间发送，即发送时间间隔不一定是固定的，此方式适合于非匀速变化。
- 模拟仿真结束，注销该套接字。

5.2 FlightGear 仿真实现

本节在前面几节的基础上，进行四旋翼的 FlightGear 飞行仿真的实施步骤及飞行仿真效果。

5.2.1 飞行仿真具体实施过程

1. 模型部署

首先在 Linux 操作系统中安装 FlightGear 操作软件，本文的安装路径是“/usr/local/share/FlightGear”。本文选择 FlightGear 自带的四旋翼模型，如图**??**所示。

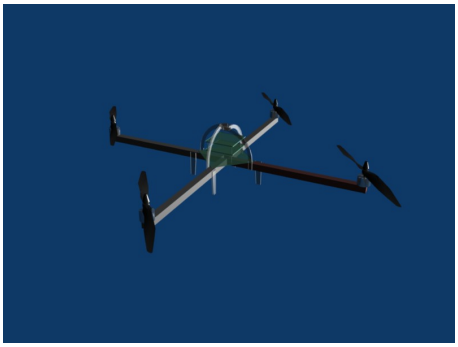
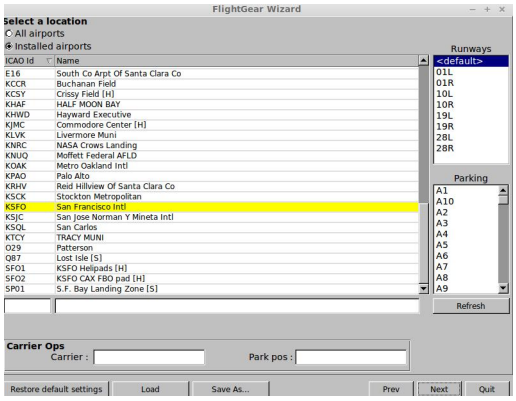


Figure 5.5: 四旋翼模型

最后启动 FlightGear 进行调试。在终端窗口内，输入’fgrun’, 启动 FlightGear 软件，选择“Quadrotor”，选择“next”设置好飞机场、飞行时间等参数后，点击“run”按钮，如图**??**所示。如果顺利进入了图 **??**界面，并且伴随螺旋桨轰鸣声，则说明四旋翼无人机飞行模型已经成功配置到了 FlightGear 中，可以使用动力学模型驱动来进行飞行模拟了。



(a) 模型选择界面　　(b) 机场选择界面

Figure 5.6: FlightGear 飞行器选择界面

2. 模型启动

在完成飞行器模型的部署后，便可以驱动模型进行仿真模拟了。在终端内，输入命令，驱动 FlightGear 进行飞行模拟的命令行如下：

```
1  fgfs --aircraft=arducopter
2       --fdm=network
3       --airport=KSFO
```

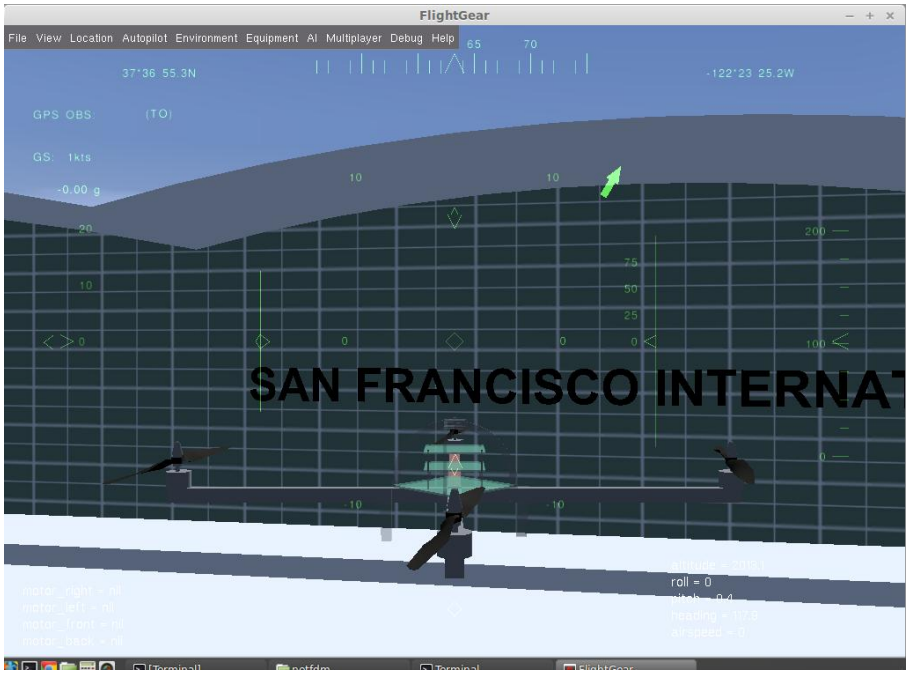


Figure 5.7: 四旋翼无人机启动界面

```
4        localhost,5501,5502,5503
5        --airport=KSFO
```

–airport=KSFO：指定仿真时使用的飞机场 ID，该飞机场必须在/FlightGear/Airports 下能够找到。

–aircraft=quadrotor：指定仿真时使用的飞行器模型

–fdm=network：指定仿真使用的动力学模型，可供选择的有：jsb，larcsim，yasim，magic，balloon，external，pipe，ada，null。默认的动力学模型是 jsb(JSBSim)。network 指通过外部数据驱动程序运行（比如通过网络发送的数据）。

localhost,5501,5502,5503 开启动力学数据连接协议, 向 fligtgear 传送数据。

5.2.2 仿真效果

系统实现了四旋翼无人机飞行的三维可视仿真。将四旋翼无人机的飞行参数（六自由度数据），通过通信模块传递给 FlightGear，在飞行参数的驱动下，四旋翼无人机模型在显示屏上可以模拟飞行器起飞、飞行、降落等各种飞行场景，画面连续，可控制播放，可变视角观察（键盘 ctrl+V 命令），并可进行飞行数据记录与视景回放，达到了设计的预期效果，满足了仿真要求，如图**??**所示。

(a) 滚转运动　　(b) 俯仰运动

(c) 偏航运动　　(d) 空中悬停

Figure 5.8: 四旋翼仿真效果

总结与展望

6.1 全文总结

飞行控制是保证飞机安全飞行的核心部分，是衡量飞机飞行品质的重要指标。飞行控制技术的高低决定了无人机的飞行品质。本文主要研究了四旋翼无人机及固定翼无人机的飞行控制，基于 FLightGear 飞行模拟软件搭建飞行仿真平台，实现半物理仿真。本文主要做的工作如下：

1. 简述飞行控制过程中坐标系转换的原理以及坐标系旋转矩阵公式推导过程。
2. 推导四旋翼旋转矩阵，为六自由度四旋翼飞控模型铺垫。
3. 概述 FlighGear 软件组件，程序框架及软件优势。
4. 进行四旋翼无人机的飞控建模，分为三个方面。第一个方面，是实现飞行摇杆的数据传入过程，对于四旋翼而言，主要是飞行姿态角传入及油门数据读取。第二部分，是建立六自由度非线性的飞行动力学模型，实现 FlightGear 外部飞控模型的实时解算，达到实时控制四旋翼无人机效果。第三部分，是基于 FlightGear 搭建的三维视景仿真系统的介绍。
5. 基于 FlightGear 内部的飞行动力学模型 JSBSim，对固定翼无人机进行控制。其过程同样分为三部分，第一部分与第三部分分别于四旋翼无人机相同，主要是第二部分的飞行动力学模型。本文对 JSBSim 模型进行配置，实现 JSBSim 与飞行摇杆数据之间的接口通信编程。
6. 详细讲述 FlightGear 飞行器驱动的步骤以及系统实施的条件。以四旋翼为对象，从飞行器模型载入到 FlightGear 通信模块的实现最后到 FlightGear 三维视景系统的仿真效果展现。介绍了在 Linux 操作系统下，如何使用 FlightGear 进行半物理仿真的操作步骤。
7. 本文对无人机飞行控制的工作作出了一些展望，尤其是基于视觉的无人机飞行控制，结合 FlightGear 飞行仿真软件，实现功能更为强大的半物理仿真的飞行控制。

6.2 对未来工作的展望

根据本文的分析，可以发现无人机飞行控制技术已经非常成熟，但也存在不少可以改进的地方，基于视觉的无人机飞行控制，可以作为未来研究的重点，概括起来主要有如下几个方面：

1. 可行性。无人机内置水平和竖直两个摄像头，可以完成对无人机所处环境图像的采集，这满足了引入计算机视觉方法的前提条件；
2. 高效性。无人机采集的图像，能够通过无人机内部自建的 wifi 无线网络实时传送至计算机，从而可在计算机上运行视觉处理算法，充分发挥计算机强大的计算能力，实现对有用信息进行解算。这是非常关键的一点，解决了计算机视觉方法数据处理量大，难以利用无人机自带的处理器芯片进行求解的问题；
3. 自主飞行性。无人机在实际飞行时，必须由人实时发出控制信号才能保证其飞行。而我们希望无人机在需要很少的人为引导，甚至是没有人干预的情况下，同样可以安全平稳飞行，即减弱人在整个控制系统中所扮演的角色。利用计算机视觉技术取代人在控制系统中的作用，就显得尤为重要。
4. 通用性。摄像机善于捕捉运动信息，而传统的传感器则较吃力，从应用的角度来看，视觉信号的抗干扰性能很好。此外，视觉导航既适用于室内环境，也适用于室外环境，通用性好。
5. 合理性。无人机设计的完整合理性，使我们不需要考虑其电子元件级的实现、气动布局、力学建模以及电机转速的控制方法，可直接通过对俯仰角、滚转角、偏航角以及竖直方向速度的控制实现对无人机的各种控制，这大大简化了我们的工作，使我们可以专注于无人机与计算机视觉方法的结合。

致　谢

首先要感谢我的导师布树辉老师。感谢布老师在整个毕设过程中的耐心指导，感谢布老师在整个论文进展过程中，提供的文献资料和实验平台，感谢布老师在整个毕设过程中的宝贵意见。与布老师交流过程中，不断的加深对问题的理解与认识，不断的提高自己解决问题的能力。对于很多女同学而言，并不会去选择编程方向作为自己的毕设，我只是因为一时的兴趣才去选择了它。回想起来，自己从一月份的一无所知到三月份的懵懵懂懂再到如今六月份的豁然开朗，从开始的压力山大到中途的排斥抵触再到现在的一往无前，在布老师的悉心教导下，让我懂得科研的路途毕竟是曲折而坎坷的，需要一份对待科研的严谨与热情，去迎风破浪，最终，必会柳岸花明。

同时要感谢教研室的赵勇师兄，在坐标系变换的编程实践中，赵勇师兄提供的四元数算法，让问题瞬间豁然开朗，感谢赵勇师兄的不吝赐教。感谢韩鹏程，程少光，王磊师兄，在我代码调试过程中，悉心的指导，在我压力很大的时候，热心的开导，分享他们的科研经历，感谢你们的鼓励与支持，让我可以一步一步慢慢的成长。

另外要感谢 Curt Olson 等飞行爱好者们，是他们创造了 FlightGear 这个功能强大的开源的飞行模拟软件；感谢为 Linux 贡献代码的程序员们，这个自由免费的平台为我完成毕设提供了不少便利；感谢清华大学王磊博士，他创作的 LaTeX 模板使我的论文的排版得以顺利完成。

最后感谢我的家人对我一如既往的关心和支持，感谢我的男朋友对我一如既往的支持与鼓励。

毕业设计小结

本次毕业设计不仅仅是对我大学四年来学习知识的一个总结，更是对我知识学习能力一次拓展与提高的过程。本次毕设主要以四旋翼无人机飞行控制为核心，学习了如何通过 C++ 编程实现坐标系的转换，旋转矩阵的运算。实践了通过编程实现半物理仿真的过程，以及提高了如何将数学公式转换为编程语言的能力。同时，了解了 FlightGear 飞行模拟软件的通信过程，学习如何实现 UDP 的通信编程。因为 FlightGear 是一款功能强大的开源软件，更适合在 Linux 操作系统下进行使用，所以，学习了如何在 Liunx 操作系统下进行 FlightGear 软件的安装，启动及 shell 文件的编写。在 Linux 系统下，更方便的使用 FlightGear 飞行模拟软件与 Joystick 摇杆结合，实现对四旋翼无人机的飞行控制。同时，本次毕设也对固定翼无人机的飞行控制进行了学习，通过 FlightGear 自带的 JSBsim 飞行动力学模型，结合 Joystick 飞行摇杆，通过 TCP/IP 编程实现对 FlightGear 的通信过程，完成对固定翼无人机的飞行控制。

本次毕业设计中印象比较深刻的有飞行控制过程中，坐标系转换的问题，必须保证坐标系选择的一致性，学习了如何通过 C++ 编程实现对向量矩阵的运算过程。同时，通过对 FlightGear 通信模块的学习，了解了 UDP,TCP/IP 协议，通过 C++ 编程实现 Joystick 飞行摇杆数据到 FlightGear 通信的过程。对六自由度非线性飞行动力学方程的推导过程，进行了十分深刻的了解。同时，对如何将数学公式变成编程语言的编程能力，有了一定提高。

经历了本次毕设，我对无人机飞行控制的过程有了一定了解，对 Linux 操作系统有了一定认识。在今后的学习过程中，会更加深入的学习飞行控制理论，提高自己的编程水平，熟练使用 FlightGear 飞行模拟软件，实现功能更强大的飞行仿真。