计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(9):409−415 [doi: 10.15888/j.cnki.csa.008701]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Unity 物理引擎的多旋翼无人机仿真模型①

石百玉[1,2], 高　岑[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)
通信作者: 石百玉, E-mail: shibaiyu19@mails.ucas.ac.cn

摘　要: 本文主要基于 Unity 物理引擎, 以四旋翼无人机作为研究对象, 通过建立无人机运行模式的数学模型, 进行仿真模型研究.一方面, 通过将无人机受力模型直接应用在电机对应位置, 来仿真现实物体的受力状态, 从而免去了对模型的刚体数学建模, 简化了仿真建模过程; 另一方面, 通过分析无人机运动原理, 对无人机进行动力系统和控制系统建模, 其中控制系统采用串级 PID 控制算法进行姿态控制; 本文最后, 通过飞行实验和测试验证了无人机模型的稳定性、有效性, 满足了四旋翼无人机的仿真要求.

关键词: Unity; 仿真建模; 串级 PID 控制; 多旋翼无人机; 姿态控制

引用格式: 石百玉,高岑.基于 Unity 物理引擎的多旋翼无人机仿真模型.计算机系统应用,2022,31(9):409–415. http://www.c-s-a.org.cn/1003-3254/8701.html

Simulation Model of Multi-rotor UAV Based on Unity Physics Engine

SHI Bai-Yu[1,2], GAO Cen[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: This study constructs the mathematical model of the unmanned aerial vehicle (UAV) operation mode for simulation model analysis by using the Unity physics engine and taking a quad-rotor UAV as the research object. Specifically, the force state of a real object is simulated by applying the force model of the UAV directly to the corresponding position of the motor, which eliminates the mathematical rigid body modeling of the model and simplifies the simulation modeling process. Additionally, the power system and control system of the UAV are modeled by analyzing the UAV motion principle, where the control system uses the PID cascade control algorithm for attitude control. Finally, the stability and validity of the UAV model are verified by flight experiments, which meet the simulation requirements of quad-rotor UAVs.

Key words: Unity; simulation modeling; PID cascade control; multi-rotor UAV; attitude control

采用虚拟仿真技术构建现实场景仿真系统已经是各行各业应用场景中比较成熟的一种技术应用模式. 虚拟仿真技术可以通过三维建模搭建可重复过程的特性很好的解决现实物理世界中不易复现的各类场景, 从而进行相关应用问题的分析. 应用虚拟仿真技术构建的仿真系统能够在降低实物成本和保证人员安全的前提下, 在逼真的虚拟环境中为用户提供高真实感的体验, 并全程实现量化数据的跟踪, 进而有效提高用户对实际场景的应变决策能力. 随着近几年各行业中信息化建设水平的不断提高, 在软硬件配套设施及相关工程管理技术发展创新的推动下, 虚拟仿真信息化过程也不断演进变化, 从单纯的三维模型建模、三维模型展示、三维场景过程展示向实景仿真及数据实时交互反馈评估等方向发展. 在此基础上, 与各行业相匹配的虚拟仿真系统应

① 收稿时间: 2021-12-28; 修改时间: 2022-01-26; 采用时间: 2022-02-15; csa 在线出版时间: 2022-06-24

运而生, 作为评估过程、演练演习的重要手段.

Unity 是行业领先的跨平台实时 3D 开发引擎, 广泛的应用在各种专业技术领域, 可用于创作、运行实时互动的 2D 和 3D 内容. 基于 Unity3D 开发的虚拟仿真系统为用户提供了丰富的学习条件与逼真的虚拟环境. Wang 等人[1] 提出 Unity 在虚拟仿真和虚拟现实等方面具有广泛应用的价值.

目前, 随着相关领域的迅速发展, 多旋翼无人机广泛的应用在警备、城市管理、农业、地质、气象、电力、抢险救灾、视频拍摄等行业. 多旋翼无人机集成了飞行控制、惯性导航、光电侦察等多种设备于一身的复杂装备, 其价格昂贵、寿命有限, 不便用户直接使用无人机进行训练来提高操作技术[2]. 在虚拟仿真技术发展背景下, 使用虚拟仿真无人机提高用户操作能力, 能够减少训练成本, 扩展无人机应用领域, 对探索更加科学有效的训练方法具有重大意义.

对于虚拟仿真来说, 根据仿真应用的需求, 对应用领域的相关属性进行科学的抽象和相应的描述, 是从现实世界迈进仿真世界的重要前提. 模型是仿真活动的基础. 仿真模型不仅仅是使用数学公式对系统模型的运行规律进行建模, 而且还对虚拟环境和虚拟对象的视觉外观、物理特性进行建模.

马忠丽等人[3] 、王小青等人[4]、彭玉元等人[5] 的相关研究, 重点在于仿真平台系统的设计应用, 将重点放在系统的设计上, 取得了较好的成果, 但并没有对无人机的仿真模型进行深入的研究.

1 多旋翼无人机运动分析

研究无人机仿真模型, 首先需要分析多旋翼无人机的基本组成和飞行原理.

多旋翼无人机通常由机架、动力系统、控制系统组成. 多旋翼无人机机身常见布局如图 1 所示, 包括三旋翼、四旋翼、六旋翼和八旋翼等.

本文以 QuadX 型四旋翼无人机为例, 分析多旋翼无人机飞行原理. 无人机采用 4 个旋翼作为直接动力源, 通过改变机架上不同位置的电机转速来调整飞行器自身的姿态实现空间内 6 自由度运动, 即垂直、俯仰、滚转、偏航运动. 将机头前进方向右侧的电机命名为 M1, 以机身逆时针方向将其与电机命名为 M2, M3, M4. 通常 M1, M3 逆时针旋转、M2, M4 顺时针旋转来使相邻电机的以不同转向来抵消扭矩.

将 4 个电机的转速同时增大或减小即可实现垂直运动; 将 M1、M3 的转速增加或者将 M2、M4 的转速减小, 实现向右偏航. 反之, 实现向左偏航; 将 M1、M2 的转速减小或者将 M3、M4 转速增加时, 四旋翼会产生前倾的扭矩, 实现俯仰运动, 螺旋桨拉力的垂直分力抵消重力, 在前进方向的分力使四旋翼向前飞行, 如图 2 所示. 同理可知无人机左右飞行原理. 其他机身布局飞行原理相似.

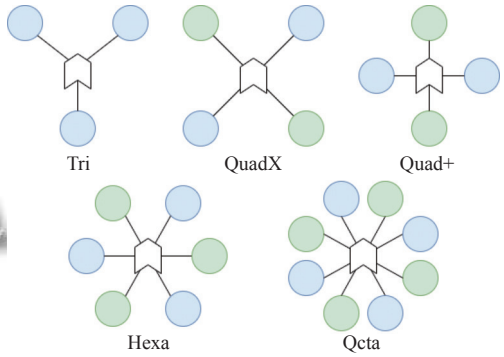

图 1 多旋翼无人机旋翼常见布局

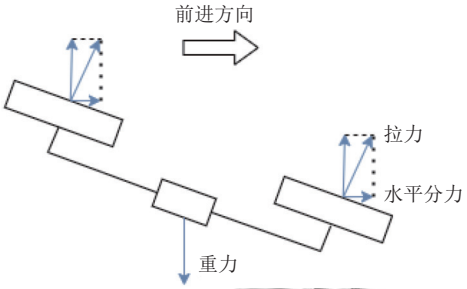

图 2 无人机前进运动

2 四旋翼无人机建模

本节就多旋翼无人机进行控制模型进行建模, 主要包含动力系统建模和控制系统建模. 其中信号传递如图 3 所示.

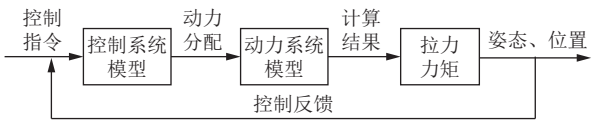

图 3 模型信号传递

2.1 无人机动力系统建模

无人机动力系统建模分为 3 部分: 螺旋桨建模、电机建模、电池建模.

多旋翼无人机常采用定桨距螺旋桨, 根据 Dai 等人[6] 的螺旋桨建模方法, 螺旋桨拉力 T (单位: N) 和转矩 M (单位: N·m) 表示如下:

$$T = C_T \rho \omega^2 D_P^4 \tag{1}$$

$$M = C_M \rho \omega^2 D_P^5 \tag{2}$$

其中, C_T 为拉力系数, C_M 为转矩系数, ω 为螺旋桨转速, D_P 为螺旋桨直径, ρ 为空气密度. 文献 [7] 给出 C_T, C_M 的计算公式为:

$$C_T = 0.25\pi^3 \lambda \xi^2 B_P K_0 \frac{\varepsilon \arctan \dfrac{H_P}{\pi D_P} - \alpha_0}{\pi A + K_0} \tag{3}$$

$$C_M = \frac{1}{4A}\pi^2 C_d \zeta^2 \lambda^2 B_P \tag{4}$$

$$C_d = C_{fd} + \frac{\pi A K_0^2}{e} \frac{\left(\varepsilon \arctan \dfrac{H_P}{\pi D_P} - \alpha_0\right)^2}{\left(\pi A + K_0\right)^2} \tag{5}$$

其中, H_P 为螺旋桨桨距, D_P 为螺旋桨直径, B_P 为桨叶数, 同时给出了参数的平均取值: A=5, ε=0.85, λ=0.75, ζ=0.5, e=0.83, C_{fd}=0.015, α_0=0, K_0=6.11.

目前, 多旋翼无人机使用无刷直流电机, 对于无刷直流电机转速 N 有:

$$N = K_V \cdot U \tag{6}$$

其中, K_V 为电机 KV 值, 定义为电机输入电压每增加 1 伏特电机空转转速增加的转速, 单位为 RPM/V; U 为电机输入电压. 在不考虑电机机械损耗和电机负载转矩情况下, 视电机转速 N 为螺旋桨转速.

对于电池模型, 主要目的是利用电池电压计算电机转速, 进而根据螺旋桨模型求得某电压下螺旋桨产生的拉力和转矩. 因此简化放电过程, 假设放电过程中电压保持不变, 忽略电池、电调、电机内阻的损耗, 只关注电池总电压.

考虑到模型的扩展性和易维护性, 根据模块化编程思想, 创建 Propeller 类实现螺旋桨模型的拉力、扭矩算法, 便于模型升级与扩展. 为了减少机器计算量, 在构造方法中将固定已知的参数提前计算存储. 电机类 Motor、电池类 Battery 同理. 其中 Motor 类以引用的形式调用 Propeller 类、Battery 类的方法, 以属性的形式获取或赋值当前电机的电压参数, 实现在 Unity 脚本生命周期中, 每固定 0.02 s 调用一次的 FixedUpdate() 方法, 部分代码如下:

```
var rotateSpeed = getRotateSpeed(EngineVol);
var Pull = _Propeller.getPull(rotateSpeed);
var counterTorque =
_Propeller.getTorque(rotateSpeed) * Mathf.Sign((float)Direction);
motorRigidbody.AddForce(transform.up * Pull); bodyRigidbody.
AddTorque(bodyRigidbody.transform.up * counterTorque);
```

其中 motorRigidbody、bodyRigidbody 分别是电机三维模型、无人机三维模型的 Rigidbody 组件; Direction 为枚举类型当前电机转向, 顺时针赋值 1, 逆时针赋值 −1, 调用 Mathf.Sign() 方法获得符号, rotateSpeed 的值由式 (6) 求得.

Rigidbody.AddForce()、Rigidbody.AddTorque() 方法是 Unity 脚本 API, 作用分别是向 Rigidbody 添加力、相对扭矩.

至此, 无人机动力系统建模完毕.

2.2 无人机控制系统建模

控制系统要解决机体在哪、姿态如何、如何飞行的问题, 其中姿态估计是控制模型的决策的基础, 姿态估计主要目的是估计姿态角, 测量加速度、角速度和位置等信息, 解决机体位置、机体姿态的问题.

多旋翼无人机通常使用 IMU 传感器单元进行无人机的姿态估计, 使用 GPS 传感器确定无人机的位置. 惯性测量单元 (inertial measurement unit, IMU) 包括加速度计和陀螺仪, 用来检测和测量物体加速度与旋转运动的传感器.

在 Unity, 可直接调用 Rigidbody.angularVelocity 脚本 API 即可得物体的角速度, Transform.position 可获得物体位置. 创建 Sensors 类封装该 API, 但 Unity 的脚本 API 中并没有可直接获得加速度的方法, 因此根据加速度表达式 (7) 间接求出物体加速度, 角加速度同理.

$$a = \frac{\Delta v}{\Delta t} \tag{7}$$

在 Unity 脚本生命周期中, 每固定 0.02 s 调用一次 FixedUpdate() 方法, 调用间隔时间较短, 视为 Δt. 线加速度计算代码如下:

```
previousVelocity = currentVelocity;
currentVelocity = rgb.velocity;
acceleration = new Vector3(
(currentVelocity–previousVelocity).x/FixedUpdateTime,
(currentVelocity–previousVelocity).y/FixedUpdateTime,
(currentVelocity–previousVelocity).z/ FixedUpdateTime);
```

将上述代码写入 FixedUpdate() 方法中, 即可计算出物体每一固定帧的运动线加速度, 其中 rgb 为物体刚体 Rigidbody 组件, previousVelocity、currentVelocity 为 Vector3 类型变量, FixedUpdateTime 为 float 型变量, 值为 0.02f 即 FixedUpdate() 的调用周期.

解决完机体在哪、机体姿态的问题后, 接下来需要解决如何飞行的问题, 这需要 Controller 类能够从

Sensors 类中获取机体姿态信息并根据输入的控制指令计算出达成目标姿态所需的拉力和力矩, 然后对无人机电机进行动力分配, 实现无人机的运动控制.

为了使输出能够符合期望的输入指令, 通常情况下选择 PID 控制器. PID 控制是应用最广泛的一种工业控制算法, 距今已有近百年的历史. 由于四旋翼无人机为数字控制系统, 选用离散式 PID. 离散式 PID 控制器分为位置式和增量式两种算法形式. 其中增量式计算结果只与最近 3 次的偏差有关, 解决位置式积分运算的误差积分积累问题, 因此该控制系统模型选用增量离式 PID 控制算法. 离散式 PID 控制其表达式如下:

$$u(k) = K_P \times e(k) + K_I \sum_{i=0}^{k} e(i) + K_D[e(k) - e(k-1)] \quad (8)$$

由式 (8) 得:

$$\Delta u(k) = u(k) - u(k-1) \quad (9)$$

即:

$$\Delta u(k) = K_P[e(k) - e(k-1)] + K_I e(k) + K_D[e(k) - 2e(k-1) + e(k-2)] \quad (10)$$

其中, K_P、K_I、K_D 分别为比例系数、积分系数和微分系数, $u(k)$ 是第 k 次采样时刻的计算结果, $e(k)$ 是第 k 次采样时刻控制器的偏差. 实现代码实现如下:

```
float err = pid.exp–pid.act;
float increment =
pid.kp * (err–pid.errK1) + pid.ki * err +
pid.kd * (err–2.0f * pid.errK1+pid.errK2);
pid.output+=increment;
pid.errK2=pid.errK1;
pid.errK1=err;
```

代码中, 以结构体形式定义 PID 对象, 结构体内存储 PID 控制参数、期望值、目标值、两次误差值和输出值, 以引用形式传入 PID 算法中, 在算法中读写相关参数, 即代码中结构体变量 pid.

当输出量与被控制系统呈线性关系时, 单极 PID 能获得较好的效果. 而多旋翼无人机通常可以简化为一个二阶阻尼系统[8], 是非线性系统. 螺旋桨转速和升力是平方倍关系, 单极 PID 在多旋翼无人机上很难取得好效果. 而串级 PID 控制系统提高了稳定性、可靠性和有效性, 克服了传统 PID 精度不高的问题[9].

串级 PID 控制就是两级 PID 控制串联在一起, 分为内环 PID 控制和外环 PID 控制, 外环的输出作为内环的输入, 如图 4 所示. 对无人机的 3 个姿态角做 PID 控制, 外环输入反馈的是角度数据, 内环输入反馈的是角速度数据. 这样即使外环数据剧烈变化, 而内环数据不会发生突变.

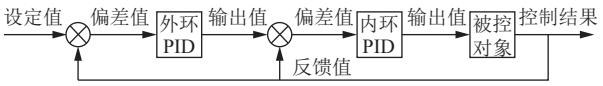

图 4 串级 PID 控制系统示意图

确定了控制算法后, 分析控制输入值、输出值和反馈值的关系. 在姿态控制中, 需要控制俯仰、横滚、偏航 3 个姿态角, 即分别需要 3 个串级 PID 来控制无人机姿态. 这 3 个串级 PID 的外环的输入值都是其期望的姿态角度, 输出值是期望的角速度; 内环的输入是外环输出的期望角速度, 输出值是期望的角加速度. 最终作用于物体后, 外环的反馈值是物体当前的实际姿态角, 内环反馈的是物体当前的实际角速度. 在模型中, 期望值由输入控制脚本提供, 测量的实际值由前文提及的 Sensors 类提供. 同时还要对无人机的飞行高度进行串级 PID 控制, 以此实现无人机的悬停运动, 高度控制 PID 外环输入是无人机实际高度, 内环输入是无人机垂直方向上的速度.

在实际应用时发现, 俯仰、横滚运动中, 俯仰、横滚角取值范围在−90°~+90°控制幅度较小, 测量值即可作为输入的实际, 而偏航角取值范围在−180°~180°, 当测量值超过 180°时测量值符号会相反, 此时会造成输入误差, 需要特殊处理. 这里以向量方法求偏航角, 构造单位向量:

$$\vec{v}_{exp} = (\sin(\theta_{exp}), \cos(\theta_{exp})) 和 \vec{v}_{act} = (\sin(\theta_{act}), \cos(\theta_{act}))$$

由向量夹角公式:

$$\cos\theta = \frac{\vec{a}\,\vec{b}}{|\vec{a}||\vec{b}|} \quad (11)$$

求得向量间夹角, 再由向量叉乘确定夹角方向, 其结果作为偏航角控制的测量值.

对 3 个姿态角分别采用串级 PID 控制, 使多旋翼无人机控制系统的非线性多变量输入多变量输出控制问题简化为两变量输入单变量输出问题.

根据 QuadX 形布局的多旋翼无人机的运动原理知, 电机 M1 转速对俯仰运动有负向作用、对横滚有正向作用, 对偏航有正向作用; 电机 M2 转速对俯仰运动有负向作用、对横滚有负向作用, 对偏航有负向作用; 电机 M3 转速对俯仰运动有正向作用、对横滚有负向作用, 对偏航有正向作用; 电机 M4 转速对俯仰运动有正向作用、对横滚有正向作用, 对偏航有负向作用. 于是有:

$$\begin{cases} P_1 = P_h - P_{\text{pit}} + P_{\text{rol}} + P_{\text{yaw}} \\ P_2 = P_h - P_{\text{pit}} - P_{\text{rol}} - P_{\text{yaw}} \\ P_3 = P_h + P_{\text{pit}} - P_{\text{rol}} + P_{\text{yaw}} \\ P_4 = P_h + P_{\text{pit}} + P_{\text{rol}} - P_{\text{yaw}} \end{cases} \tag{12}$$

其中, P_1、P_2、P_3、P_4 分别代表驱动电机 M_1、M_2、M_3、M_4 所需的 PWM 值, P_h 为维持无人机飞行高度 PWM 值, P_{pit}、P_{rol}、P_{yaw} 分别为俯仰角、横滚角和偏航角的所需 PWM 值, 最后结果值限制在电机驱动器能接受的范围内. 将求得的输入电压分别驱动 4 个电机, 电机输入电压的改变引起电机转速的改变, 进而改变整个系统的受力情况, 实现对无人机的姿态控制.

在该模型中, 不考虑电机的饱和区, PWM 值为电机输入电压占电池总电压的比值. 另外, 为了保证基本面姿态控制与抗风性需求, 一般多旋翼无人机在飞行时控留有制裕度, 油门指令应该小于 0.9[7].

至此无人机控制系统建模完毕.

3 四翼无人机模型实现

3.1 Unity 脚本实现

确定了多旋翼无人机的算法模型后, 着手对模型算法的实现, 编写 Unity 脚本来实现对无人机对象的逻辑控制. 综合考虑各个脚本之间的关系, 设计 UML 时序图如图 5.

UserInput 脚本用于处理用户输入, 在多旋翼无人机中的遥控中, 需要控制 3 个姿态角和油门, 一共 4 个方向轴上的控制, 因此需要在 Unity Input Manager 中设置 4 个轴向控制来相应键盘输入. 由于无人机的姿态控制可以维持无人机的姿态, 在此设计油门输入实际是控制无人机的飞行高度. 依据航模遥控“美国手”手柄模式, 设置键盘 *w*、*s* 控制油门; 键盘 *a*、*d* 控制偏航; 方向键上下控制俯仰; 方向键左右控制横滚.

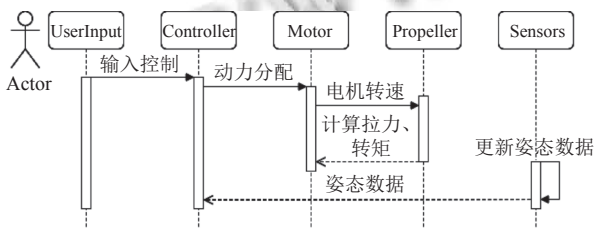

图 5 UML 时序图

UserInput 脚本的输入即为无人机的期望高度和期望姿态, Controller 脚本获得无人机期望姿态后, 调用 PID 算法计算期望输出, 并将动力分配给 Motor 脚本, 而后将目标转速送入 Propeller 脚本中计算单个电机所产生的拉力和转矩, 直接作用在无人机对象的刚体上产生力的作用, Sensors 脚本实时监控无人机对象的状态, 反馈给 Controller 脚本实现闭环控制. Controller 脚本流程如图 6.

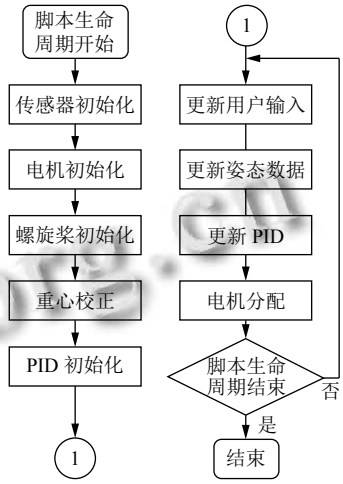

图 6 Controller 脚本流程图

3.2 Unity 无人机对象组件设置

在 Unity 中使用包围盒碰撞检测, 在游戏对象上挂载 Collider 组件实现, 同时可以在脚本中实现碰撞后的受力效果, 无人机对象添加 Collider 效果如图 7.

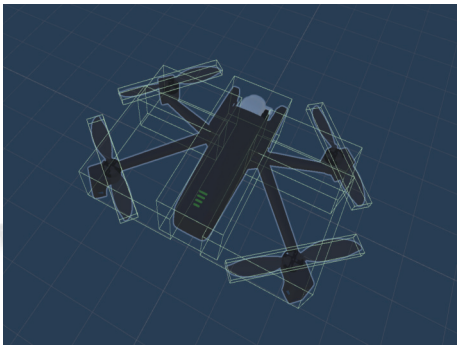

图 7 无人机对象

在多旋翼无人机的运动中, 仅有机体位置和螺旋桨旋转变化, 设置螺旋桨旋转动画会让仿真更加逼真, 给予用户沉浸体验. 创建 Animator 脚本, 该脚本获取 Motor 脚本的电机转速, 将其应用到螺旋桨上, 使螺旋桨产生旋转效果. 然而在现实中, 螺旋桨旋转由于视觉暂留而产生模糊效果, 转速越快模糊效果越明显, 而在 Unity 的旋转是在每渲染帧刷新对象状态, 使对象“瞬移”看起来不够真实, 对此可以使用模糊纹理贴图的代替三模模型旋转, 当螺旋桨低速旋转时使用真实三维模型旋转, 当转速逐渐提高时, 隐藏三维模型, 使用平

面网格模型, 其材质使用提前绘制出模糊纹理的透明图片 (图 8), 实际旋转效果如图 9, 其中 M1, M2 没有使用模糊纹理替换, 看出有很明显差别.

图 8 模糊纹理

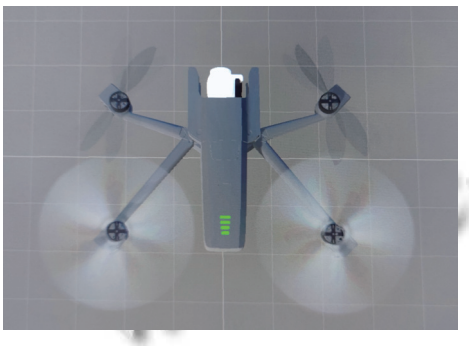

图 9 模糊纹理替换旋转效果

为了实现 Unity 中无人机的控制逻辑, 需要对无人机对象挂载所需的脚本及相关组件, 挂载组件方式如图 10.

Rigidbody 刚体组件是实现游戏对象物理特性必不可少的组件, 刚体为游戏对象赋予物理属性, 使游戏对象在 Unity 物理引擎的驱动下产生力的作用, 实现真实世界中的运动效果. 本文将无人机受力模型直接应用在电机对应位置, 来仿真现实物体的受力状态, 从而免去了对模型的刚体数学建模.

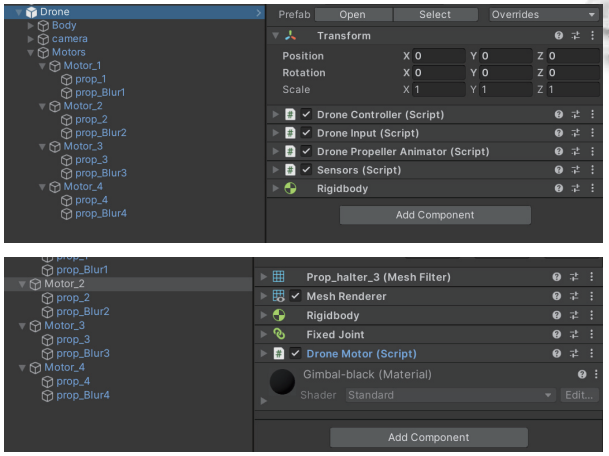

图 10 无人机 GameObject 组件挂载关系

在 Drone 对象中, Body 是无人机机体三维模型对象, Camera 是云台相机三维模型对象, Motors 是所有电机、螺旋桨三维模型对象的父物体.

Motors 对象中, Motor_1 等是无人机的电机对象, 每个电机对象中的 Fixed Joint 组件将电机刚体和父物体刚体固定在一起; 其中 Motor 脚本用于对父物体产生实际的力和扭矩.

在 Motor 脚本中, 计算螺旋桨产生的拉力扭矩并直接作用电机的刚体上, 电机模型刚体带动整个无人机刚体运动. 所有的电机模型所产生的总拉力、总扭矩的作用效果由 Unity 物理引擎实现. 值得注意的是, 在 Controller 脚本的初始化中, 根据 4 个电机的位置将无人机的重心设置在对角电机连线的交点处, 否则无人机在飞行过程中可能产生不稳定的效果.

另外 Prop_1 等是螺旋桨三维模型, 以 Motor_1 等对象作为父物体, 这样可以使螺旋桨跟随电机对象; 而 Prop_Blur1 等为上述模糊纹理螺旋桨对象.

4 运行效果

实验分别以无人机悬停、俯仰、滚转、偏航运动效果作为实验结果, 实验中无人机数据参数如表 1.

4.1 PID 参数整定

无人机的姿态控制采用串级 PID 控制算法, 内环的输出是最终结果, 所以先调内环参数, 再调外环参数. 经过实验测试, 确定该无人机姿态控制 PID 参数如表 2.

表 1 无人机数据参数

参数名称 (单位)	参数值
无人机重量 (kg)	36.5
最大俯仰角度 (°)	30
螺旋桨直径 (inch)	38
螺旋桨螺距 (inch)	20
螺旋桨桨叶数 (个)	2
电机KV值 (RPM/V)	75
电机数量 (个)	4
电池总电压 (V)	51.8
刚体线运动阻力系数	0.2
刚体角运动阻力系数	0.05

表 2 无人机 PID 参数

控制类型	内/外环	KP	KI	KD
高度控制	内环	0.5	0.001	0.1
	外环	1.65	0.1	—
俯仰控制	内环	0.009	0.001	0.005
	外环	8.5	—	—
滚转控制	内环	0.008	0.001	0.005
	外环	6.1	—	—
偏航控制	内环	0.28	0.01	0.05
	外环	1.9	—	—

在 PID 参数整定过程中, 使用 Unity 调用 Debug.Log() 方法将无人机姿态数据实时打印出来, 将打印结果使用 Matlab 描绘控制曲线如图 11.

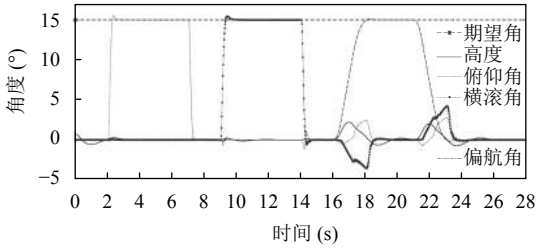

图 11 PID 姿态控制曲线

由图 11 可见, 在调整偏航角时会影响到俯仰和滚转运动, 产生较大波动. 为解决此问题, 使机体在一定误差范围内调整完俯仰、滚转运动后再调节偏航运动, 实验结果如图 12 所示, 可见限制偏航运动在一定程度上增加了稳定性, 同时限制偏航运动的俯仰滚误差与偏航响应时间呈现负向关, 限制误差值越小, 偏航响应越慢.

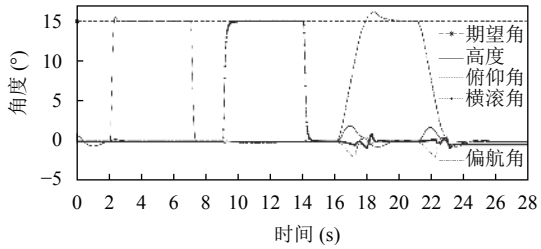

图 12 限制偏航运动的 PID 姿态控制曲线

运行场景, 使用键盘控制无人机飞行, 进行姿态控制, 无人机能够实现俯仰控制前进后退、 滚转控制左移右移、偏航实现左转右转, 如图 13 所示.

5 结论与展望

本文基于 Unity 物理引擎, 对四旋翼无人机这种典型非线性、多变量、欠驱动系统进行建模, 将无人机受力直接应用在电机对应位置, 更贴近现实物体受力状态, 免去了对模型的刚体数学建模, 使建模化繁为简; 在模型的姿态控制采用串级 PID 算法对无人机姿态进行控制, 飞行实验结果证明控制该模型的稳定性、有效性, 满足无人机仿真要求. 该模型既满足了仿真模拟的真实性又简化了建模过程, 为无人机仿真模拟提供了一条新思路. 不过本文没有考虑电机、电调、电池等物理特性、螺旋桨的气动阻力等, 有待进一步研究.

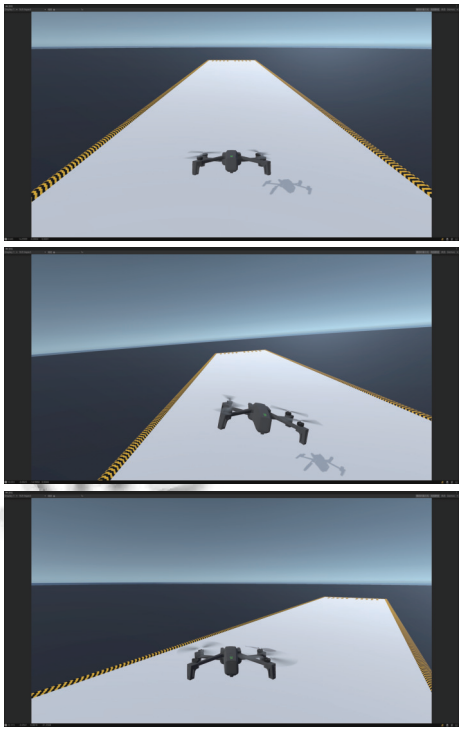

图 13 无人机模型实际运行效果

参考文献

1 Wang S, Mao ZL, Zeng CH, *et al*. A new method of virtual reality based on Unity3D. Proceedings of 2010 18th International Conference on Geoinformatics. Beijing: IEEE, 2010. 1–5.
2 褚丽娜, 李建增, 谢志刚. 基于 Unity3D 的无人机交互式课件开发. 现代计算机, 2015, (24): 39–42.
3 马忠丽, 吴丽丽, 李嘉迪, 等. 基于 Unity3D 的多旋翼飞行器模拟训练系统设计. 应用科技, 2021, 48(3): 111–117.
4 王小青, 吴兆轩, 马岩. 基于 Unity 的无人机三维视景系统设计与实现. 长江信息通信, 2021, 34(2): 12–14. [doi: 10.3969/j.issn.1673-1131.2021.02.004]
5 彭玉元, 李敏敏, 鲁家亮, 等. 无人机飞行虚拟训练平台设计应用. 大众科技, 2019, 21(5): 1–3. [doi: 10.3969/j.issn.1008-1151.2019.05.001]
6 Dai XH, Quan Q, Ren JR, *et al*. An analytical design-optimization method for electric propulsion systems of multicopter UAVs with desired hovering endurance. IEEE/ASME Transactions on Mechatronics, 2019, 24(1): 228–239. [doi: 10.1109/TMECH.2019.2890901]
7 全权. 多旋翼飞行器设计与控制. 杜光勋, 赵峙尧, 戴训华, 等, 译. 北京: 电子工业出版社, 2018: 68.
8 赵亮, 王强, 徐立攀, 等. 四旋翼飞行器的串级 PID 姿态控制. 电脑知识与技术, 2018, 14(8): 248–249.
9 张静, 刘恒, 郑采薇. 串级 PID 控制在微型无人机姿态控制中的应用. 电子世界, 2014, 16: 271–272. [doi: 10.3969/j.issn.1003-0522.2014.16.267]

(校对责编: 孙君艳)