|  |  |  |  |
| --- | --- | --- | --- |
|  | | |  |
| 文档版本： | | |  |
|  | | |  |
|  | | | |
| 飞行器软件基础作业 | | | |
|  | | | |
| 研制总结报告 | | | |
|  | | | |
|  |  |  | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
| 北京航空航天大学  2023年 11 月 | | | |
|  | | | |

团队成员

|  |  |  |
| --- | --- | --- |
| 学号 | 名字 | 分工 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

更改情况记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 更改页次 | 标记 | 处数 | 更改单号 | 更改人 | 日期 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目 次

[1 概述 1](#_Toc149119888)

[1.1 需求概述 1](#_Toc149119889)

[1.3 任务分工 1](#_Toc149119890)

[2 研发内容 1](#_Toc149119891)

[2.1 开发环境 1](#_Toc149119892)

[2.2 功能定义 1](#_Toc149119893)

[3.3 逻辑流程 1](#_Toc149119894)

[4 研制过程 1](#_Toc149119895)

[4.1 软件研制过程概述 1](#_Toc149119896)

[4.1.1 软件需求分析阶段 1](#_Toc149119897)

[4.1.2 软件设计阶段 2](#_Toc149119898)

[4.1.3 软件实现和单元测试阶段 2](#_Toc149119899)

[4.1.4 系统测试阶段 2](#_Toc149119900)

[5 软件满足任务指标情况 2](#_Toc149119901)

[6 结论 2](#_Toc149119902)

[7 注释 2](#_Toc149119903)

无人机自主避障系统

1 概述

* 1. 需求概述

实现四旋翼无人机在地图3的等高自主避障飞行

1.3 任务分工

如前表格所示。

2 研发内容

2.1 开发环境

开发环境：c++编程语言，虚拟机上的ubuntu操作系统。

2.2 功能定义

软件的主要性能要求包括：

a) 完成飞行器的定高飞行。

b) 通过激光雷达反馈的距离信息调整航向角和姿态，控制飞机定高飞行的同时从场景的一侧到达另外一侧，并避开障碍。

c) 调整参数控制实现一定的飞行速度。

d) 有完整的起飞降落功能。

3.3 逻辑流程

1. 完成电机启动
2. 获取相关hector无人机的状态参数；
3. 升空，保持一定高度，大致为2m高度
4. 进行计算，输出控制参量/cmd\_vel
5. 循环执行，直到到达目的地，降落落地

4 研制过程

4.1 软件研制过程概述

ROS开源的无人机仿真软件研制经历了需求分析阶段，内容学习阶段，程序设计阶段，测试优化阶段，2023年11月转入设计定型阶段研制，本报告是对ROS开源的无人机仿真软件设计定型阶段研制过程的总结。

4.1.1 软件需求分析阶段

本阶段研制过程如下：

1. 查阅相关文献资料，分析无人机定高飞行和避障飞行的实现要求，并明确在实际设计过程中需要完成结合激光雷达的距离数据，实时调整飞机上升或下降的功能和预先探测地形走势，引导飞行器提前调整飞行姿态和飞行路径的功能。
2. 安装WMware workstation 16虚拟机，设置仿真环境，启动ROS，尝试操纵无人机飞行。
3. 学习了解ROS上开源的无人机仿真功能包hector\_quadrotor，广泛应用该功能包下的子功能包hector\_slam，包含了一个全套四旋翼可视化仿真功能包，包含了四旋翼的动力学模型、控制、姿态解算等子功能包。

4.1.2 软件设计阶段

本阶段研制过程如下：

1. 本阶段研制采用模块化设计优化方法，明确样例代码各部分所实现的功能，以及相对应参数在实际软件运行时的效果反馈。
2. 细化根据期望高度和实际高度进行高度控制的部分，实现低升高降。
3. 修改避障方面功能函数，实现无人机能在期望角前进一段距离的同时不撞到障碍物。

4.1.3 软件实现和单元测试阶段

本阶段研制及测试过程如下：

1. 初次测试迭代：
2. 主要问题：偏转角不正确引发撞机问题，sleep或者loop设置问题，subscribe更新的频率和loop的频率问题，无法降落的问题。
3. 更新：修改信息更新的时间，调整前进的距离进而和雷达探测期望角的周期相匹配；给定一个最终距离，无人机超出就会降落。
4. 二次测试迭代：
5. 主要问题：无人机起飞时没有先定高后飞行，掉高问题严重，降落不够平稳，由于存在扰动有可能撞机等问题。
6. 更新:修改落地函数参数，让无人机在既定的时间或者高度内停止电机的转动；增加无人机上升函数；修改无人机前进的距离，优化无人机前进一段距离的同时不撞到障碍物的功能；优化了起飞后由于掉高造成的扰动使航向受到干扰的问题。

4.1.4 系统测试阶段

本阶段测试过程如下：

1. 登录虚拟机，打开场景3脚本，编译后执行hector\_ctrl文件，代码如下：

*#include <ros/ros.h>*

*#include <math.h>*

*#include <hector\_uav\_msgs/EnableMotors.h>*

*#include <hector\_uav\_msgs/YawrateCommand.h>*

*#include <hector\_uav\_msgs/ThrustCommand.h>*

*#include <hector\_uav\_msgs/AttitudeCommand.h>*

*#include <sensor\_msgs/LaserScan.h>*

*#include <geometry\_msgs/PoseStamped.h>*

*#include <geometry\_msgs/Vector3Stamped.h>*

*#include <geometry\_msgs/Twist.h>*

*float angle\_min = -2.3561899662; //　雷达数据的起始角度（最小角度）*

*float angle\_max = 2.3561899662;//　雷达数据的终止角度（最大角度）*

*float angle\_increment = 0.00436331471428; //　雷达数据的角度分辨率（角度增量）*

*//存储姿态以及位置*

*float pose\_x = 0;*

*float pose\_y = 0;*

*float pose\_z = 0;*

*float vector\_x = 0;*

*float vector\_y = 0;*

*float vector\_z = 0;*

*//期望控制高度*

*float pose\_z\_h = 2;*

*//初始化控制姿态的参数*

*float vel\_z = 0; //速度*

*float vel\_x = 0; //速度 /// 降速*

*float ac\_z = 0.2;*

*float angle\_dir = 0; //初始化定义航向期望角*

*float dis\_max = 0;*

*int ranges\_size = 0;*

*int index\_max = 0;*

*// 回调函数，获取激光扫描数据，找出其中的最大距离值，并据此计算无人机的航向期望值。*

*void scanCallback(const sensor\_msgs::LaserScan::ConstPtr& msg){*

*ranges\_size = (780>msg->ranges.size())? msg->ranges.size(): 780;*

*//这段代码的作用是将ranges\_size设置为msg->ranges.size()和780中较小的那个值。*

*//float32[] ranges　雷达数据每个点对应的在极坐标系下的距离值 [m]*

*dis\_max = 0;//dis\_max为距离最大值*

*int i = 0;*

*for (; i< ranges\_size; ++i){*

*if(isinf(msg->ranges[i]) == 0){*

*if(msg->ranges[i]> dis\_max){*

*dis\_max = msg->ranges[i];*

*index\_max = i;//选出最大值以及最大值对应的位置*

*}*

*}*

*}*

*angle\_dir = angle\_min + angle\_increment \* index\_max;//算出与0°相差的航向值*

*}*

*//ROS中用于处理机器人位姿（位置和姿态）信息的回调函数*

*//提取出x、y、z坐标值，分别存储到全局变量pose\_x、pose\_y、pose\_z*

*void PoseCallback(const geometry\_msgs::PoseStamped::ConstPtr& msg){*

*pose\_x = msg->pose.position.x;*

*pose\_y = msg->pose.position.y;*

*pose\_z = msg->pose.position.z;*

*}*

*void VectorCallback(const geometry\_msgs::Vector3Stamped::ConstPtr& msg){*

*vector\_x = msg->vector.x;*

*vector\_y = msg->vector.y;*

*vector\_z = msg->vector.z;*

*}*

*//根据期望高度和实际高度进行高度控制，低升高降*

*float PIDCtrl\_height(float pose\_z, float pose\_z\_h){*

*if (fabs(pose\_z\_h - pose\_z )<0.5){*

*return pose\_z\_h - pose\_z;*

*}*

*else if(pose\_z\_h - pose\_z>0.5){*

*return 0.5;*

*}*

*else{*

*return -0.5;*

*}*

*}*

*//根据相差角进行航向控制，dir为体轴系下角度，与头航向夹角，最后期望变成0*

*float PIDCtrl\_dir(float angle\_dir) {*

*if (fabs(angle\_dir)< angle\_increment){*

*return -angle\_dir;*

*}*

*else if(fabs(angle\_dir)<0.2){*

*return -angle\_dir;*

*}*

*else if(angle\_dir >0){*

*return -0.2;*

*}*

*else {*

*return 0.2;*

*}*

*}*

*int main(int argc, char\*\* argv){*

*ros::init(argc,argv,"hector\_ctrl\_node");//初始化节点*

*ros::NodeHandle nh;//创建节点句柄，管理节点的资源，指定全局空间*

*ros::Publisher velocity\_publisher\_;//创建话题通讯机制*

*ros::ServiceClient motor\_enable\_service\_;//创建一个服务通讯*

*ros::Rate loop\_rate(10); //设置循坏的频率是10hz*

*ros::Publisher cmd\_vel\_pub = nh.advertise<geometry\_msgs::Twist>("/cmd\_vel",1000);*

*//创建一个Publisher，发布名为/cmd\_vel的topic，消息类型为geometry\_msgs::Twist，队列长度1000*

*motor\_enable\_service\_ = nh.serviceClient<hector\_uav\_msgs::EnableMotors>("enable\_motors");*

*if(!motor\_enable\_service\_.waitForExistence(ros::Duration(5.0))){*

*ROS\_WARN("motor enable service not found");*

*return false;*

*}*

*/\*同上，命名这个serves的名称并且规定消息类型；*

*由于Service通信的特殊性，加一个client.waitForExistence()这类阻塞式函数可以保证启动和调用的顺序\*/*

*ros::Subscriber subScan = nh.subscribe("/scan",50, scanCallback); //0.25s的更新频率*

*ros::Subscriber subPose = nh.subscribe("/ground\_truth\_to\_tf/pose",50, PoseCallback); //0.25s 的更新频率*

*ros::Subscriber subVector3 = nh.subscribe("/ground\_truth\_to\_tf/euler",50, VectorCallback); ///*

*// 订阅 /scan、 /ground\_truth\_to\_tf/pose两个话题*

*hector\_uav\_msgs::EnableMotors srv;*

*srv.request.enable = true;//导入hector\_uav\_msgs信息包，里面含有一些senor\_msg里面没有的信息*

*motor\_enable\_service\_.call(srv);//将srv调用至定义的服务通讯中*

*while(ros::ok()){//确认状态，当按下ctr+C或者别的一些异常情况时，其值才可能是FALSE*

*ros::Duration(2).sleep();//等待两秒钟无人机启动*

*geometry\_msgs::Twist cmd\_vel\_msg;//使用Twist类型的数据来控制无人机的运动*

*if(pose\_z == 0){//初始上升函数*

*ROS\_INFO("Take off");*

*int i = 1;*

*while(i<=5){*

*ros::Duration(2).sleep();*

*vel\_z = 0.5;*

*cmd\_vel\_msg.linear.z = vel\_z;*

*ac\_z = 0.1;*

*cmd\_vel\_msg.angular.z = ac\_z;*

*i++;*

*if(pose\_z >= 2){*

*ros::Duration(2).sleep();*

*i = 6;*

*}*

*cmd\_vel\_pub.publish(cmd\_vel\_msg);*

*ros::spinOnce();*

*loop\_rate.sleep();*

*}*

*}*

*vel\_z = PIDCtrl\_height(pose\_z, pose\_z\_h);//使无人机保持在2m高度*

*vel\_x = 0.065\*(dis\_max-3.3);//使无人机前进一段距离的同时不撞到障碍物*

*ac\_z = PIDCtrl\_dir(angle\_dir);*

*cmd\_vel\_msg.linear.z = vel\_z;*

*cmd\_vel\_msg.angular.z = ac\_z;*

*cmd\_vel\_msg.linear.x = vel\_x;*

*//用更新的位置、角度等参数的变化使无人机运动*

*//保持无人机稳定*

*ROS\_INFO("vel z is %f, vel x is %f, angular z is %f\n", vel\_z, vel\_x, ac\_z);//输出控制的数据信息*

*cmd\_vel\_pub.publish(cmd\_vel\_msg);//更新*

*ros::spinOnce();//将数据代入回调函数循环*

*loop\_rate.sleep();//规定循环的延迟时间*

*int i = 1;*

*float delta;*

*if(pose\_x >50){//判断是否需要降落*

*ROS\_INFO("Start landing");//提示开始降落*

*delta = pose\_z;//存储开始降落时候的高度*

*while(i<=5){*

*ros::Duration(1).sleep();*

*vel\_z = -delta/5;*

*ac\_z = PIDCtrl\_dir(angle\_dir);*

*cmd\_vel\_msg.angular.z = ac\_z;*

*vel\_x = 0.03\*(dis\_max-5);*

*cmd\_vel\_msg.linear.x = vel\_x;*

*i++;*

*if(pose\_z <= 0.5){*

*break;*

*}*

*ros::spinOnce();*

*loop\_rate.sleep();*

*}*

*break;}*

*}*

*return 0;*

*}*

运行后，成功实现无人机自主避障飞行，用时36s

5 软件满足任务指标情况

表1 功能和性能指标满足情况表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 任务书所要求的功能或性能指标 | 验证方式 | 验证结果 |
| 1 | 实现无人机等高飞行 | 运行过程中监测无人机与地面距离 | 成功 |
| 2 | 实现无人机避障飞行 | 运行过程中监测无人机与障碍物之间距离 | 成功 |

6 结论

本软件实现了无人机的等高自主避障飞行，可以进行交付使用。