

SHANGHAI JIAO TONG UNIVERSITY



多机器人系统与控制 课程大作业

小组编号: _	第	第二组	
组员姓名:	陈炜昊	王喆隆	
	易子淇	章雨悠	
_ 指导教师:	熊振华	董伟 吴建华	

2021年12月

多机器人系统与控制大作业

目录

1	Que	nestion 1		
	1.1	Modeling	1	
2	Que	nestion 2		
	2.1	Modeling and Problem Formulation	3	
	2.2	Optimal Control	4	
3	Que	Question 3		
	3.1	Optimal Control	5	
4	Que	Question 4		
	4.1	Estimation of the algebraic connectivity of the graph	5	
$\mathbf{A}_{\mathbf{J}}$	Appendices			
附	录 A	MATLAB code for solving the expression of λ_2 in Q2	6	
附	录 B	MATLAB code for solving the expression of u in Q4	7	
附	录 C	C++ code for question 2	7	
附	录 D	C++ code for question 4	11	

1 Question 1

编写可视化仿真环境,模拟上图所示主动视觉连接下的多机器人系统。同时假定系统还配置有 10Hz 通讯系统,可供机器人间进行状态交换。以下所有仿真均在此仿真环境中依照上述假定进行验证。

1.1 Modeling

在 Gazebo 中建立仿真模型如下,图中红色表示相机视角范围,白色表示相机中心线。

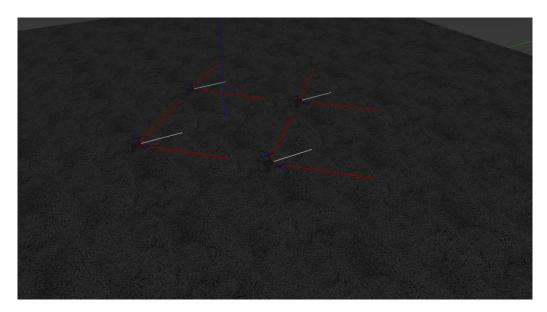


图 1: 仿真模型示意图

2 Question 2

在圆周飞行当中,若去除等间距飞行的要求,且每个相机可以观测任一协同机器人,但限定机器人飞行速度区间为 $[0,2\omega_r r]$,加速度区间为 [-g,g] ,且 4 个机器人的平均飞行速度应尽可能接近 $\omega_r r$,相机偏航角最大转速为 $\omega_{\psi_c}^m(\omega_{\psi_c}^m=10\omega_r)$,试推导保证最佳拓扑连接的运动形式。阅读多机器人拓扑保持相关文献 [1,2]。如每个机器人可通过通讯系统获知全局连通拓扑,试自行选择几种不同类型的椭圆运动,编写可保障视觉拓扑连通的分布式控制算法。若每一单体机器人可由一个单积分系统描述,试设计合理的控制器以保持上述拓扑连接。

2.1 Modeling and Problem Formulation

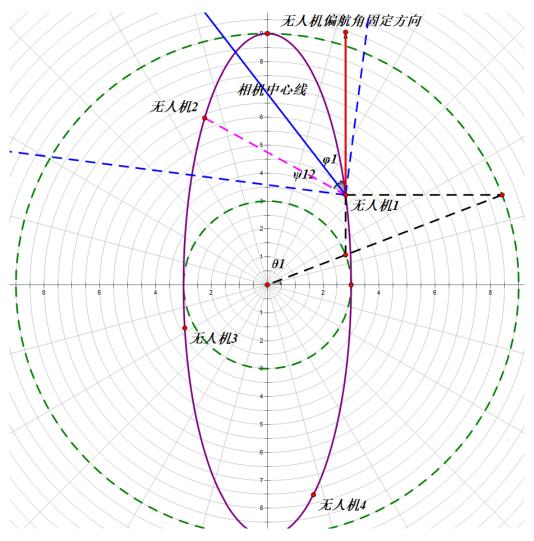


图 2: 问题二示意图

假设椭圆方程 $x^2/a^2+y^2/b^2=1$,机器人 i 在椭圆坐标 $(a\cos\theta_i,b\sin\theta_i),\theta_i\in[0,2\pi)$ 处,其相机偏航角可行范围为 $\psi_c\in(-\pi/2,\pi/2)$,机器人 j 相对于机器人 i 的偏航方向夹角为 $\psi_{ij}\in[-\pi,\pi)$

$$\psi_{ij} = \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) \tag{1}$$

机器人 j 与机器人 i 连线与机器人 i 距离为

$$l_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$
 (2)

由此设计机器人i观察机器人j的连通性权重函数为

$$w_{ij} = \exp\left[-c_1(\psi_{ij} - \varphi_i)^2 - c_2\varphi_i^2 - c_3(l_{ij} - \frac{\sqrt{a^2 + b^2}}{2})^2\right]$$
(3)

上式指数部分三项的物理含义分别为

- (1) $(\psi_{ij} \varphi_i)^2$ 代表机器人 j 相较于相机 i 中轴的偏转角度性能,机器人位于相机视野中央时观测性能最佳,偏角越大性能越差。
- (2) φ_i^2 代表相机偏航角性能,偏航角绝对值越大,相机越容易陷入"旋转死角"不利于后续的相机旋转即连通保持。
- (3) $(l_{ij} \frac{\sqrt{a^2 + b^2}}{2})^2$ 代表机器人 j 相较于相机 i 的距离性能,假设距离值等于等距分布时的距离 $2\sqrt{\frac{a^2 b^2}{a^2 + b^2}}$ 时性能最佳,距离过近时容易发生碰撞,距离过远时相机可视性不好。
- (4) c_1, c_2, c_3 分别为各分量的乘子系数。

2.2 Optimal Control

因为机器人间可以进行状态交换,彼此传递观测结果,例如机器人i观测到机器人j后将相对信息同步给机器人j后,j也可以据此反推出机器人i相对于自己的位置等信息。我们认为机器人i与机器人j的连通性为i观察j连通性权重与j观察i之和,表达式如下

$$\widetilde{w}_{ij} = w_{ij} + w_{ji}$$

$$= \exp\left[-c_1(\psi_{ij} - \varphi_i)^2 - c_2\varphi_i^2 - c_3(l_{ij} - \frac{\sqrt{a^2 + b^2}}{2})^2\right]$$

$$+ \exp\left[-c_1(\psi_{ji} - \varphi_j)^2 - c_2\varphi_j^2 - c_3(l_{ij} - \frac{\sqrt{a^2 + b^2}}{2})^2\right]$$
(4)

最终可以列出拉普拉斯矩阵为

$$L = \begin{bmatrix} \widetilde{w}_{12} + \widetilde{w}_{13} + \widetilde{w}_{14} & -\widetilde{w}_{12} & -\widetilde{w}_{13} & -\widetilde{w}_{14} \\ -\widetilde{w}_{12} & \widetilde{w}_{12} + \widetilde{w}_{23} + \widetilde{w}_{24} & -\widetilde{w}_{23} & -\widetilde{w}_{24} \\ -\widetilde{w}_{13} & -\widetilde{w}_{23} & \widetilde{w}_{13} + \widetilde{w}_{23} + \widetilde{w}_{34} & -\widetilde{w}_{34} \\ -\widetilde{w}_{14} & -\widetilde{w}_{24} & -\widetilde{w}_{34} & \widetilde{w}_{14} + \widetilde{w}_{24} + \widetilde{w}_{34} \end{bmatrix}$$
(5)

参考文献 [1] 中的梯度控制器,利用 MATLAB 求解 λ_2 的表达式(见附录代码),因结果过于复杂最终采用数值解法,设计控制器如下

$$u_{i}^{\theta} = K(\mathbf{p}) \frac{\partial \lambda_{2}}{\partial \theta_{i}}$$

$$u_{i}^{\varphi} = K(\mathbf{p}) \frac{\partial \lambda_{2}}{\partial \varphi_{i}}$$

$$s.t. \quad K(\mathbf{p}) = \operatorname{csch}^{2}(\lambda_{2} - \epsilon)$$

$$(6)$$

3 Question 3

参考上一任务的思路,但若机器人为双积分系统,试设计控制程序,使得所有机器人在作一致性椭圆运动时(长轴对应偏航角为0度,且长轴为短轴的3倍),并分析视觉系统最佳连通变化情况。

3.1 Optimal Control

在上一题的基础上,对于机器人i,设计双积分控制器表达式如下

$$\begin{cases} u_i = \dot{p}_i = \frac{\partial \lambda_2}{\partial p_i} \\ \dot{u}_i = \ddot{p}_i = \frac{\partial}{\partial t} \left(\frac{\partial \lambda_2}{\partial p_i} \right) = \frac{\partial}{\partial p_i} \left(\frac{\partial \lambda_2}{\partial p_i} \right) \cdot \frac{\partial p_i}{\partial t} = \frac{\partial}{\partial p_i} \left(\frac{\partial \lambda_2}{\partial p_i} \right) u_i = \frac{\partial u_i}{\partial p_i} u_i \end{cases}$$
(7)

最终完整控制律表达式如下

$$u_{i}^{\theta} = \frac{\partial \lambda_{2}}{\partial \theta_{i}}$$

$$\dot{u}_{i}^{\theta} = \frac{\partial u_{i}^{\theta}}{\partial \theta_{i}} u_{i}^{\theta} = \frac{\partial}{\partial \theta_{i}} \frac{\lambda_{2}}{\partial \theta_{i}} \cdot \frac{\partial \lambda_{2}}{\partial \theta_{i}}$$

$$u_{i}^{\varphi} = \frac{\partial \lambda_{2}}{\partial \varphi_{i}}$$

$$\dot{u}_{i}^{\varphi} = \frac{\partial u_{i}^{\varphi}}{\partial \varphi_{i}} u_{i}^{\varphi} = \frac{\partial}{\partial \varphi_{i}} \frac{\lambda_{2}}{\partial \varphi_{i}} \cdot \frac{\partial \lambda_{2}}{\partial \varphi_{i}}$$

$$(8)$$

4 Question 4

如果扩大机器人数量(如 10 个机器人),且每一机器人只可获知前后方向各两个机器人的连通拓扑,试编写分布式仿真程序估计表征主动视觉连通的拉氏矩阵次小特征值及其对应特征向量。上述估计是否可保障收敛,试证明你的结论并分析保证收敛的通讯拓朴条件。

4.1 Estimation of the algebraic connectivity of the graph

任一机器人在每次通讯中只能与前后各两个,即一共四个进行通讯。对于机器人 i,无法获知全局的拉普拉斯矩阵 L 和次小特征值 λ_2 的表达式,为了实现连通性的保持,必须对估计表征主动视觉连通的拉氏矩阵次小特征值及其对应特征向量给出定义。由问题2中设计的控制器可知

$$u_i = \frac{\partial \lambda_2}{\partial p_i} = v_2^T \frac{\partial L}{\partial p_i} v_2 \tag{9}$$

参考 [2] 对 λ_2, v_2 进行估计,将 2 估计值记为 \tilde{v}_2 具体估计及迭代过程定义如下

$$\alpha_1^i = \widetilde{v}_2^i, \quad \alpha_2^i = (\widetilde{v}_2^i)^2$$

$$z_1^i = \operatorname{Ave}(\widetilde{v}_2^i), \quad z_2^i = \operatorname{Ave}((\widetilde{v}_2^i)^2)$$

$$\dot{z}^i = \gamma(\alpha^i - z^i) - K_p \sum_{j \in \mathcal{N}_i} (z^i - z^j) + K_i \sum_{j \in \mathcal{N}_i} (\omega^i - \omega^j)$$

$$\dot{\omega}^i = -K_i \sum_{j \in \mathcal{N}_i} (z^i - z^j)$$

$$\dot{\overline{v}}_2^i = -k_1 z_1^i - k_2 \sum_{j \in \mathcal{N}_i} a_{ij} (\widetilde{v}_2^i - \widetilde{v}_2^j) - k_3 (z_2^i - 1) \widetilde{v}_2^i$$

$$(10)$$

需要注意的是,由幂迭代算法 [1], \widetilde{v}_2 的初始值应当满足条件 $\mathbf{1}^T\widetilde{v}_2=0$

得到 v2 估计值后即可设计控制律如下

$$u_{i} = \widetilde{v}_{2}^{T} \frac{\partial L}{\partial p_{i}} \widetilde{v}_{2} = \sum_{j \in \mathcal{N}_{i}} \frac{\partial a_{ij}}{\partial p_{i}} (\widetilde{v}_{2} - \widetilde{v}_{j})^{2}$$

$$a_{ij} = \widetilde{w}_{ij}$$

$$= \exp \left[-c_{1}(\psi_{ij} - \varphi_{i})^{2} - c_{2}\varphi_{i}^{2} - c_{3}(l_{ij} - 2\sqrt{\frac{a^{2}b^{2}}{a^{2} + b^{2}}})^{2} \right]$$

$$+ \exp \left[-c_{1}(\psi_{ji} - \varphi_{j})^{2} - c_{2}\varphi_{j}^{2} - c_{3}(l_{ij} - 2\sqrt{\frac{a^{2}b^{2}}{a^{2} + b^{2}}})^{2} \right]$$

$$(11)$$

在本问题中 a_{ij} 最终控制律需要求解 $\frac{\partial a_{ij}}{\partial \theta_i}$ 和 $\frac{\partial a_{ij}}{\partial \varphi_i}$ 的表达式,利用附录中的 MATLAB 求解得到结果过于复杂,因此这里省略。

为了优化最终的仿真效果,将模型进行简化,修改队形轨迹为圆形,得到新的连通权重函数如下

$$\widetilde{w}_{ij} = w_{ij} + w_{ji}$$

$$= \exp\left[-c_1(\psi_{ij} - \varphi_i)^2 - c_2\varphi_i^2 - c_3(\theta_j - \theta_i)^2\right] + \exp\left[-c_1(\psi_{ji} - \varphi_j)^2 - c_2\varphi_j^2 - c_3(\theta_j - \theta_i)^2\right]$$
(12)

由此可以得到控制律为

$$u_{i}^{\theta} = \sum_{j \in \mathcal{N}_{i}} \frac{\partial a_{ij}}{\partial \theta_{i}} (\widetilde{v}_{2} - \widetilde{v}_{j})^{2}$$

$$= \sum_{j \in \mathcal{N}_{i}} \left\{ w_{ij} \left[-c_{1}(\pi + \theta_{j} + \theta_{i}) + 2(\theta_{j} - \theta_{i}) \right] + w_{ji} \left[-c_{1}(\pi + \theta_{j} + \theta_{i}) + 2(\theta_{j} - \theta_{i}) \right] \right\} (\widetilde{v}_{2} - \widetilde{v}_{j})^{2}$$

$$u_{i}^{\varphi} = \sum_{j \in \mathcal{N}_{i}} \frac{\partial a_{ij}}{\partial \varphi_{i}} (\widetilde{v}_{2} - \widetilde{v}_{j})^{2}$$

$$= \sum_{i \in \mathcal{N}_{i}} \left\{ w_{ij} \left[+c_{1}(\pi + \theta_{j} + \theta_{i}) - 2c_{2}\varphi_{i} \right] \right\} (\widetilde{v}_{2} - \widetilde{v}_{j})^{2}$$

$$(13)$$

参考文献

- [1] Peng Yang, Randy A. Freeman, Geoffrey J. Gordon, Kevin M. Lynch, Siddhartha S. Srinivasa, and Rahul Sukthankar. Decentralized estimation and control of graph connectivity in mobile sensor networks. In 2008 American Control Conference, pages 2678–2683, 2008.
- [2] Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. On decentralized connectivity maintenance for mobile robotic systems. In 2011 50th IEEE Conference on Decision and Control and European Control Conference, pages 988–993, 2011.

Appendices

附录 A MATLAB code for solving the expression of λ_2 in Q2

```
% init
      \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}
                        clc; clear;
n = 4:
                        m = 4, syms ab c1 c2 c3 theta0 theta1 theta2 theta3 phi0 phi1 phi2 phi3 theta = [theta0 theta1 theta2 theta3]; phi = [phi0 phi1 phi2 phi3];
      6
7
8
9
                       10
 \frac{11}{12}
                          end
                         % calculate psi
\frac{14}{15}
                                                 i = 1:n
for j = 1:n
                                                                         p\,s\,i\,(\,i\;,\;\;j\,)\;=\;a\,t\,a\,n\,(-(y\,(\,j\,)-y\,(\,i\,)\,)\,/\,(\,x\,(\,j\,)-x\,(\,i\,)\,)\,)\,;
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
                                                end
                          end
                        % calculate 1
                                                 for j = 1:n

for j = 1:n

j
                          for
                          end
                                                                 \begin{array}{c} \text{ulate} \ . \\ 1:n \\ j = 1:n \\ \text{if } i = j \\ L(i, j) = \text{sym}(0); \\ \text{elseif } j < i \\ L(i, j) = L(j, i); \\ \vdots \\ -\text{exp}(-\text{c1}) \\ \text{si}(j, j) \end{array}
                         % calculate w
                                                  i =
for
                                                                                                \begin{array}{l} \mathbb{L}(\texttt{i}\;,\;\texttt{j}\;) = -\exp(-\texttt{c}1*(\texttt{psi}\;(\texttt{i}\;,\;\texttt{j}) - \texttt{phi}\;(\texttt{i}))^2 - \texttt{c}2*\texttt{phi}\;(\texttt{i}\;)^2 - \texttt{c}3*(\texttt{l}\;(\texttt{i}\;,\;\texttt{j}) - 2*\texttt{sqrt}\;(\texttt{a}^2*\texttt{b}^2/(\texttt{a}^2+\texttt{b}^2)))^2) \dots \\ -\exp(-\texttt{c}1*(\texttt{psi}\;(\texttt{j}\;,\;\texttt{i}) - \texttt{phi}\;(\texttt{j}))^2 - \texttt{c}2*\texttt{phi}\;(\texttt{j})^2 - \texttt{c}3*(\texttt{l}\;(\texttt{j}\;,\;\texttt{i}) - 2*\texttt{sqrt}\;(\texttt{a}^2*\texttt{b}^2/(\texttt{a}^2+\texttt{b}^2)))^2); \end{array} 
34
35
36
37
38
39
40
41
42
                                                 end
                          end
                                                                         1:n
                                                                         \bar{j}\stackrel{\dots}{=}1\!:\!n
                                                   for
                                                                                                 continue;
                                                                          _{
m else}
  43
                                                                                                L(i, i) = L(i, i) - L(i, j);

    \begin{array}{r}
      44 \\
      45 \\
      46 \\
      47
    \end{array}

                                                                          _{\mathrm{end}}
                                                 _{
m end}
                          end
                        %% calculate lambda2
                        e = eig(L);
e = arrayfun(@char, e, 'uniform', 0);
writecell(e, 'e.txt');
  18
  49
```

附录 B MATLAB code for solving the expression of u in Q4

```
clc; clear;
2 syms a b c1 c2 c3 theta_i theta_j phi_i phi_j
3 x_i = a*cos(theta_i); x_j = a*cos(theta_j);
4 y_i = b*sin(theta_i); y_j = b*sin(theta_j);
5 psi_ji = atan((y_j - y_i)/(x_j - x_i));
6 psi_ji = atan((y_i - y_j)/(x_i - x_j));
7 l_ij = sqrt((x_i - x_j)^2 + (y_i - y_j)^2);
8 a_ij = -exp(-c1*(psi_ji - phi_i)^2 - c2*phi_i^2 - c3*(l_ij - 2*sqrt(a^2*b^2/(a^2+b^2)))^2) ...
9 - exp(-c1*(psi_ji - phi_j)^2 - c2*phi_j^2 - c3*(l_ij - 2*sqrt(a^2*b^2/(a^2+b^2)))^2);
10 pretty(diff(a_ij, theta_i))
11 pretty(diff(a_ji, phi_i))
```

附录 C C++ code for question 2

```
#include <ros/ros.h>
#include <geometry_msgs/PoseStamped.h>
#include <geometry_msgs/Quaternion.h>
#include <mavros_msgs/CommandBool.h>
#include <mavros_msgs/SetMode.h>
#include <mavros_msgs/State.h>
#include <nav_msgs/Odometry.h>
#include <condots
#include <condo
```

```
double k_v = 0.1, k_w = 1.2; // 比例系数 double epsilon = 1e-3; double w = 0.15; //initial position offset
  23
             //initial position offset double x_ini[4] = {2.0, 0.0, -2.0, 0.0}; double y_ini[4] = {0.0, 6.0, 0.0, -6.0}; /* 定义变量 */
  25
  26
             VectorXd phi(4);
VectorXd x(4);
  29
  30
            VectorXd x(4);
VectorXd y(4);
VectorXd z(4);
VectorXd u_theta(4);
VectorXd u_phi(4);
  32
33
  \frac{34}{35}
            geometry_msgs::PoseStamped pose0; geometry_msgs::PoseStamped pose1; geometry_msgs::PoseStamped pose2; geometry_msgs::PoseStamped pose3; nav_msgs::Odometry pose_odom0; nav_msgs::Odometry pose_odom1; nav_msgs::Odometry pose_odom2; nav_msgs::Odometry pose_odom3; mavros_msgs::State current_state;
  36
37
  38
39
40
41
42
43
  44
  45
  \frac{46}{47}
            ///(>>>>>>>>>
             void update_theta();
void update_state();
  48
  49
            void update_state();
geometry_msgs::Vector3 quaternion2euler(geometry_msgs::Quaternion quater);
geometry_msgs::Quaternion euler2quaternion(geometry_msgs::Vector3 euler);
double get_lambda2(VectorXd Theta, VectorXd Phi);
void update_pub();
double get_angle(double angle1, double angle2);
  50
  51
  52
  53
  54
55
56
57
             void state_cb(const mavros_msgs::State::ConstPtr& msg){
    current state = *msg;
  58
59
60
             void plane_pos_cb0(const nav_msgs::Odometry::ConstPtr &msg){
    pose odom0 = *msg;
  61
  62
63
64
             void plane_pos_cb1(const nav_msgs::Odometry::ConstPtr &msg){
    pose_odom1 = *msg;
  65
             void plane_pos_cb2(const nav_msgs::Odometry::ConstPtr &msg){
    pose_odom2 = *msg;
  66
67
             void plane_pos_cb3(const nav_msgs::Odometry::ConstPtr &msg){
    pose_odom3 = *msg;
  68
69
70
71
72
73
74
75
76
77
78
            int main(int argc, char **argv)
                        ros::init(argc, argv, "offb_node");
                        //uav0
  79
                        ros::NodeHandle nh0;
                       ros::Subscriber state_sub0 = nh0.subscribe<mavros_msgs::State>("uav0/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub0 = nh0.subscribe<nav_msgs::Odometry>("uav0/mavros/local_position/odom", 1, plane_pos_
ros::Publisher local_pos_pub0 = nh0.advertise<geometry_msgs::PoseStamped>("/uav0/mavros/setpoint_position/local", 1
ros::ServiceClient arming_client0 = nh0.serviceClient<mavros_msgs::CommandBool>("/uav0/mavros/cmd/arming");
ros::ServiceClient set_mode_client0 = nh0.serviceClient<mavros_msgs::SetMode>("/uav0/mavros/set_mode");
  80
  81
  82
  83
84
85
86
                        //uav1
                       //uav1/ros::NodeHandle nh1;
ros::NodeHandle nh1;
ros::Subscriber state_sub1 = nh1.subscribe<mavros_msgs::State>("uav1/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub1 = nh1.subscribe<nav_msgs::Odometry>("uav1/mavros/local_position/odom", 1, plane_pos_ros::Publisher local_pos_pub1 = nh1.advertise<geometry_msgs::PoseStamped>("/uav1/mavros/setpoint_position/local", 1
ros::ServiceClient arming_client1 = nh1.serviceClient<mavros_msgs::CommandBool>("/uav1/mavros/cmd/arming");
ros::ServiceClient set_mode_client1 = nh1.serviceClient<mavros_msgs::SetMode>("/uav1/mavros/set_mode");
  87
88
  89
  90
                                                                                                                                                                                                                                                                                                                              , 10
  91
  93
   94
                        //uav2
                       //uav2/ros::NodeHandle nh2;
ros::NodeHandle nh2;
ros::Subscriber state_sub2 = nh2.subscribe<mavros_msgs::State>("uav2/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub2 = nh2.subscribe<nav_msgs::Odometry>("uav2/mavros/local_position/odom", 1, plane_pos_ros::Publisher local_pos_pub2 = nh2.advertise<geometry_msgs::PoseStamped>("/uav2/mavros/setpoint_position/local", 1
ros::ServiceClient arming_client2 = nh2.serviceClient<mavros_msgs::CommandBool>("/uav2/mavros/cmd/arming");
ros::ServiceClient set_mode_client2 = nh2.serviceClient<mavros_msgs::SetMode>("/uav2/mavros/set_mode");
  95
96
  97
                                                                                                                                                                                                                                                                                                                              , \bar{1}
  98
  99
100
101
                       //uav3/ros::NodeHandle nh3;
ros::NodeHandle nh3;
ros::Subscriber state_sub3 = nh3.subscribe<mavros_msgs::State>("uav3/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub3 = nh3.subscribe<nav_msgs::Odometry>("uav3/mavros/local_position/odom", 1, plane_pos_
ros::Publisher local_pos_pub3 = nh3.advertise<geometry_msgs::PoseStamped>("/uav3/mavros/setpoint_position/local", 1
ros::ServiceClient arming_client3 = nh3.serviceClient<mavros_msgs::CommandBool>("/uav3/mavros/cmd/arming");
ros::ServiceClient set_mode_client3 = nh3.serviceClient<mavros_msgs::SetMode>("/uav3/mavros/set_mode");
103
105
106
108
109
                        ros::Rate rate(10.0);
111
\begin{array}{c} 112 \\ 113 \end{array}
            ||/>>>>>|
                                wait for FCU connection
                        while (ros::ok() && !current_state.connected){
    ros::spinOnce();
    rate.sleep();
115
116
```

```
pose0.pose.position.x = 0;
pose0.pose.position.y = 0;
pose0.pose.position.z = z_pose1.pose.position.x = 0;
pose1.pose.position.y = 0;
120 \\ 121 \\ 122
                                                                                        lock:
\frac{123}{124}
                      \frac{125}{126}
127
128
129
130
131 \\ 132
                     //send a few setpoints before starting
for(int i = 10; ros::ok() && i > 0; -
    local_pos_pub0.publish(pose0);
    local_pos_pub1.publish(pose1);
    local_pos_pub2.publish(pose2);
    local_pos_pub3.publish(pose3);
    ros::spinOnce();
    rate_sleep():
133
                           send a few setpoints before starting
134
135
136
137
138
139
140
                                 rate.sleep();
141
\frac{142}{143}
                      \begin{array}{ll} mavros\_msgs::SetMode\ offb\_set\_mode;\\ offb\_set\_mode.request.custom\_mode =\ "OFFBOARD"; \end{array}
144
                     mavros_msgs::CommandBool arm_cmd;
arm_cmd.request.value = true;
146
147
148
149
                       ros::Time last_request = ros::Time::now();
150
151
                       while (ros::ok()) {
                                           s::ok()){
current_state.mode != "OFFBOARD" &&
(ros::Time::now() - last_request > ros::Duration(5.0))){
if(set_mode_client0.call(offb_set_mode) &&
set_mode_client1.call(offb_set_mode) &&
set_mode_client2.call(offb_set_mode) &&
set_mode_client3.call(offb_set_mode) &&
offb_set_mode.response.mode_sent){
ROS_INFO("Offboard enabled : 4");
}
152
                                 if ( current
153
154
155
156
158
159
161
                                           fast_request = ros::Time::now();
                                } else {
   if (!current_state.armed &&
162
163
                                                     !current_state.armed &&
[ros::Time::now() - last_request > ros::Duration(5.0))){

if (arming_client0.call(arm_cmd) &&
    arming_client1.call(arm_cmd) &&
    arming_client2.call(arm_cmd) &&
    arming_client3.call(arm_cmd) &&
    arming_client3.call(arm_cmd) &&
    arm_cmd.response.success){
    ROS_INFO("Vehicle armed : 4");
}
164
165
166
167
168
169
170
171
172
                                                     last_request = ros::Time::now();
173
                                          }
174
                                }
local_pos_pub0.publish(pose0);
local_pos_pub1.publish(pose1);
local_pos_pub2.publish(pose2);
local_pos_pub3.publish(pose3);
if (pose_odom0.pose.pose.position.z > 1.8) {
    ROS_INFO("plane_takeoff !");
    break.
175
176
177
178
179
180
                                           break;
182
                                 ros::spinOnce();
183
184
                                 rate.sleep();
185
            186
187
188
                               le (ros::ok()) {
  update_state();
  VectorXd theta_tmp, phi_tmp;
  for (int i = 0; i < 4; ++i){
      theta_tmp = theta;
      theta_tmp(i) += epsilon;
      u_theta(i) = (get_lambda2(theta_tmp, phi) - get_lambda2(theta, phi))/epsilon;
      phi_tmp = phi;
      phi_tmp(i) += epsilon;
      u_phi(i) = (get_lambda2(theta, phi_tmp) - get_lambda2(theta, phi))/epsilon;
      cout << "uav" << i << ' ' << u_theta(i) << ' ' << u_phi(i) << endl;
}</pre>
                       while (ros::ok())
189
190
191
192
193
194
195
197
198
199
\frac{200}{201}
                                 update_pub();
                                update_pub();
local_pos_pub0.publish(pose0);
local_pos_pub1.publish(pose1);
local_pos_pub2.publish(pose2);
local_pos_pub3.publish(pose3);
ros::spinOnce();
202
\begin{array}{c} 203 \\ 204 \end{array}
205
\begin{array}{c} 206 \\ 207 \end{array}
                                 rate.sleep();
                      }
\frac{208}{209}
                      return 0:
210
            }
\frac{210}{211}
            212
213
\frac{1}{2}14
215
216
                                {\rm else}_{\rm if}^{\{}(y(i) > 0)
217
218
```

```
219
                                                                           theta(i) = atan(a*y(i)/b/x(i)) + pi;
220
221
222
                                                                            theta(i) = atan(a*y(i)/b/x(i)) - pi;
\frac{1}{223}
                                                             }
                                             }
225
                               }
226
                }
227
228
                  void update state(){
                                \begin{array}{lll} & \text{aparacosize}(f) \\ & \text{x}(0) &= \text{pose\_odom0.pose.pose.position.x} + \text{x\_ini} \begin{bmatrix} 0 \end{bmatrix}; \\ & \text{y}(0) &= \text{pose\_odom0.pose.pose.position.y} + \text{y\_ini} \begin{bmatrix} 0 \end{bmatrix}; \\ & \text{z}(0) &= \text{pose\_odom0.pose.pose.position.z}; \\ \end{array} 
\frac{230}{231}
                               z(0) = pose_odom0.pose.pose.position.z;
phi(0) = quaternion2euler(pose_odom0.pose.pose.orientation).z;
 232
                              \frac{233}{234}
 235
236
237
 238
239
240
 241
242
                               z(3) = pose_odom3.pose.pose.position.z;
phi(3) = quaternion2euler(pose_odom3.pose.pose.orientation).z;
243
 244
245
246
                }
                geometry_msgs::Vector3 quaternion2euler(geometry_msgs::Quaternion quater){
   geometry_msgs::Vector3 temp;
   temp.x = atan2(2.0 * (quater.w * quater.x + quater.y * quater.z), 1.0 - 2.0 * (quater.x * quater.x + quater.y * quater.y + quater.y = asin(2.0 * (quater.w * quater.y - quater.z * quater.x));
   temp.z = atan2(2.0 * (quater.w * quater.z + quater.x * quater.y), 1.0 - 2.0 * (quater.y * quater.y + quater.z * quater.y + quater.z * quater.y + quater.y
248
\frac{2}{2}49
250
\frac{250}{251}
\frac{252}{253}
                                 return temp;
\frac{254}{255}
                }
\frac{1}{256}
                  geometry\_msgs::Quaternion\ euler2quaternion\ (geometry\_msgs::Vector3\ euler)
257
\frac{1}{258}
                                 geometry msgs::Quaternion temp:
                                \begin{array}{l} \text{geometry\_misgs:: Quaternion temp;} \\ \text{temp.w} = \cos\left(\text{euler.x/2}\right) * \cos\left(\text{euler.y/2}\right) * \cos\left(\text{euler.z/2}\right) \\ + \sin\left(\text{euler.x/2}\right) * \sin\left(\text{euler.y/2}\right) * \sin\left(\text{euler.z/2}\right); \\ \text{temp.x} = \sin\left(\text{euler.x/2}\right) * \cos\left(\text{euler.y/2}\right) * \cos\left(\text{euler.z/2}\right) \\ - \cos\left(\text{euler.x/2}\right) * \sin\left(\text{euler.y/2}\right) * \sin\left(\text{euler.z/2}\right); \\ \text{temp.y} = \cos\left(\text{euler.x/2}\right) * \sin\left(\text{euler.y/2}\right) * \cos\left(\text{euler.z/2}\right) \\ + \sin\left(\text{euler.x/2}\right) * \cos\left(\text{euler.y/2}\right) * \sin\left(\text{euler.z/2}\right); \\ \text{temp.z} = \cos\left(\text{euler.x/2}\right) * \cos\left(\text{euler.y/2}\right) * \sin\left(\text{euler.z/2}\right); \\ \text{softens.} \end{aligned} 
259
260
261
262
263
                                 return temp;
\frac{264}{265}
                 }
                267
 268
269
270
271
272
273
274
275
                                 for (int i = 0; i < 4; ++i){
                                               for (int j = 0; j < 4; ++j) {
    if (X(j) - X(i) > 0) {
        psi(i, j) = atan((Y(j) - Y(i))/(X(j) - X(i)));
276
\frac{1}{277}
                                                             \begin{cases} & \text{else } \{ \\ & \text{if } (Y(j) - Y(i) > 0) \{ \\ & \text{psi}(i, j) = \operatorname{atan}((Y(j) - Y(i)) / (X(j) - X(i))) + \operatorname{pi}; \end{cases} 
279
280
282
283
284
                                                                                          psi(i, j) = atan((Y(j) - Y(i))/(X(j) - X(i))) - pi;
285
286
                                                            }
288
                                              }
289
                                              291
292
204
                                for (int i = 0; i < 4; ++i){
    for (int j = 0; j < 4; ++j){
        if (i == j){
            lap(i, j) = 0;
        }
295
296
207
298
                                                            \begin{cases} & \text{else } \{ & \text{if } (j < i) \} \\ & \text{lap(i, j)} = \text{lap(j, i)}; \end{cases} 
299
300
301
302
303
304
                                                                                         \begin{array}{l} \exists \ \text{lap(i,j)} = -\exp(-c1*(psi(i,j) - Phi(i))*(psi(i,j) - Phi(i)) - c2*Phi(i)*Phi(i) \\ - c3*(l(i,j) - sqrt(a*a+b*b)/2)*(l(i,j) - sqrt(a*a+b*b)/2) \\ -\exp(-c1*(psi(j,i) - Phi(j))*(psi(j,i) - Phi(j)) - c2*Phi(j)*Phi(j) \\ - c3*(l(i,j) - sqrt(a*a+b*b)/2)*(l(i,j) - sqrt(a*a+b*b)/2)); \end{array} 
305
306
307
308
309
                                                            }
310
311
                                             }
312
                                for (int i = 0; i < 4; ++i){
for (int j = 0; j < 4; ++j){
    if (i == j){
313
314
315
```

```
continue;
318
                                          lap(i, i) -= lap(i, j);
319
320
322
                 }
EigenSolver<Eigen::MatrixXd> es(lap);
MatrixXcd evals = es.eigenvalues();
VectorXd evalsReal = evals.real();
double lambda1, lambda2;
if (evalsReal(0) < evalsReal(1)){
    lambda1 = evalsReal(0);
    lambda2 = evalsReal(1);
}</pre>
323
324
\frac{325}{326}
327
328
329
330
                  else {
331
                          lambda1 = evalsReal(1);
332
333
                          lambda2 = evalsReal(0);
334
                  fif (evalsReal(2) < lambda1){
    lambda2 = lambda1;
    lambda1 = evalsReal(2);</pre>
335
336
337
338
339
340
                               (evalsReal(2) < lambda2){
                                  lambda2 = evalsReal(2);
342
343
                 }
if (evalsReal(3) < lambda1){
    lambda2 = lambda1;
    lambda1 = evalsReal(3);
344
345
346
347
                  }
else
348
349
                               (evalsReal(3) < lambda2){
350
                                  lambda2 = evalsReal(3);
351
353
                  return lambda2:
354
         }
355
          void update_pub(){
    geometry_msgs::Vector3 euler;
    euler = quaternion2euler(pose_odom0.pose.pose.orientation);
    euler.z += k_w*u_phi(0);
356
357
358
359
                 euler . z += k_w*u_pin(0);
pose0.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom1.pose.pose.orientation);
euler . z += k_w*u_phi(1);
pose1.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom2.pose.pose.orientation);
euler . z += k_w*u_phi(2);
euler . z += k_w*u_phi(2);
360
361
363
364
365
366
                 pose2.pose.orientation = euler2quaternion(euler)
367
368
369
370
371
372
373
374
375
376
377
                 posed.pose.position.z = z_lock;
pose1.pose.position.z = z_lock;
pose2.pose.position.z = z_lock;
pose3.pose.position.z = z_lock;
378
380
```

附录 D C++ code for question 4

```
1 #include <ros/ros.h>
2 #include <geometry_msgs/PoseStamped.h>
3 #include <geometry_msgs/Quaternion.h>
4 #include <mavros_msgs/CommandBool.h>
5 #include <mavros_msgs/SetMode.h>
6 #include <mavros_msgs/State.h>
7 #include <nav_msgs/State.h>
8 #include <eigen3/Eigen/Eigen>
9 #include <cimath>
10 #include <cimath>
11 #include <iostream>
12 #include <iostream>
13 #define pi 3.1415926535
15 using namespace std;
17 using namespace Eigen;
18 ///>
19 /* 参数定义 */
19 double r = 5;
20 double cl = 16/pi/pi, c2 = 16/pi/pi, c3 = 16/pi/pi/3;
```

```
double gama = 100, Kp = 50, Ki = 200, k1 = 18, k2 = 3, k3 = 60; double z lock = 2: // 飞行高度
            double z_lock = 2; // 飞行高度
double k_v = 0.1, k_w = 1; // 比例系数
  \frac{1}{24}
           double k_v = 0.1, k_w = 1
double epsilon = 1e-3;
double w = 0.2;
//initial position offset
double x_ini[8];
double y_ini[8];
/* 定义变量 */
VectorXd theta(8);
VectorXd y(8);
VectorXd y(8):
  \frac{26}{27}
  28
  \frac{1}{29}
  30
  31
  32
  33
  \frac{34}{35}
            VectorXd y(8);
VectorXd z(8);
  36
  37
            VectorXd u_theta(8);
           VectorXd u_phi(8);

MatrixXd v2(8, 8);

MatrixXd z1(8, 8);

VectorXd z2(8);

MatrixXd omega1(8,
  38
  39
  40
  41
  43
            VectorXd omega2(8);
           MatrixXd psi(8, 8);
MatrixXd lap(8, 8);
  \frac{44}{45}\frac{46}{46}
          geometry_msgs::PoseStamped pose0;
geometry_msgs::PoseStamped pose1;
geometry_msgs::PoseStamped pose2;
geometry_msgs::PoseStamped pose2;
geometry_msgs::PoseStamped pose3;
geometry_msgs::PoseStamped pose4;
geometry_msgs::PoseStamped pose5;
geometry_msgs::PoseStamped pose6;
geometry_msgs::PoseStamped pose6;
geometry_msgs::Odometry pose_odom0;
nav_msgs::Odometry pose_odom1;
nav_msgs::Odometry pose_odom2;
nav_msgs::Odometry pose_odom3;
nav_msgs::Odometry pose_odom4;
nav_msgs::Odometry pose_odom6;
nav_msgs::Odometry pose_odom6;
nav_msgs::Odometry pose_odom6;
nav_msgs::Odometry pose_odom7;
mavros_msgs::State curren_state;
  48
  49
  \frac{50}{51}
  52
53
54
  55
56
57
58
59
60
61
  63
64
65
66
            void update_theta(); // 更新机器人在队形中位置角
void update_state(); // 更新机器人状态, 位置+姿态
geometry_msgs::Vector3 quaternion2euler(geometry_msgs::Quaternion quater); // 四元数转欧拉角
geometry_msgs::Quaternion euler2quaternion(geometry_msgs::Vector3 euler); // 欧拉角转四元数
  67
  68
  69
  70
71
72
73
74
75
76
77
78
79
80
81
            void update_pub();
           void state_cb(const mavros_msgs::State::ConstPtr& msg){
    current_state = *msg;
            void plane_pos_cb0(const nav_msgs::Odometry::ConstPtr &msg){
   pose_odom0 = *msg;
            void plane_pos_cb1(const nav_msgs::Odometry::ConstPtr &msg){
    pose_odom1 = *msg;
            void plane_pos_cb2(const nav_msgs::Odometry::ConstPtr &msg){
    pose_odom2 = *msg;
  82
83
  84
85
86
                     l plane_pos_cb3(const nav_msgs::Odometry::ConstPtr &msg){
pose_odom3 = *msg;
            void plane
  87
88
                     l plane_pos_cb4(const nav_msgs::Odometry::ConstPtr &msg){    pose_odom4 = *msg;
            void plane
  89
90
  91
                     l plane_pos_cb5(const nav_msgs::Odometry::ConstPtr &msg){
    pose_odom5 = *msg;
            void plane
  92
  9\bar{3}
  94
95
                                              _{cb6}(const\ nav\_msgs::Odometry::ConstPtr\ \&msg){}
                     pose\_odom6 = *msg;
  97
98
            void plane_pos_cb7(const nav_msgs::Odometry::ConstPtr &msg){
    pose odom7 = *msg;
  99
100
           ///>>>>>
102
103
            int main(int argc, char **argv)
104
105
                      ros::init(argc, argv, "offb_node");
106
                       //uav0
107
                     //uav0
ros::NodeHandle nh0;
ros::Subscriber state_sub0 = nh0.subscribe<mavros_msgs::State>("uav0/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub0 = nh0.subscribe<nav_msgs::Odometry>("uav0/mavros/local_position/odom", 1, plane_pos_cos::Publisher local_pos_pub0 = nh0.advertise<geometry_msgs::PoseStamped>("/uav0/mavros/setpoint_position/local", 10, state_cb);
ros::ServiceClient arming_client0 = nh0.serviceClient<mavros_msgs::CommandBool>("/uav0/mavros/cmd/arming");
ros::ServiceClient set_mode_client0 = nh0.serviceClient<mavros_msgs::SetMode>("/uav0/mavros/set_mode");
108
109
110
111
112
114
                      //uav1
                     //uavl

ros::NodeHandle nh1;

ros::Subscriber state_sub1 = nh1.subscribe<mavros_msgs::State>("uavl/mavros/state", 10, state_cb);

ros::Subscriber local_pos_sub1 = nh1.subscribe<nav_msgs::Odometry>("uavl/mavros/local_position/odom", 1, plane_pos_

ros::Publisher local_pos_pub1 = nh1.advertise<geometry_msgs::PoseStamped>("/uavl/mavros/setpoint_position/local", 1

ros::ServiceClient arming_client1 = nh1.serviceClient<mavros_msgs::CommandBool>("/uavl/mavros/cmd/arming");

ros::ServiceClient set_mode_client1 = nh1.serviceClient<mavros_msgs::SetMode>("/uavl/mavros/set_mode");
116
117
119
120
```

```
//uav2
                                       //uav2/mavros/state nh2;
ros::NodeHandle nh2;
ros::Subscriber state_sub2 = nh2.subscribe<mavros_msgs::State>("uav2/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub2 = nh2.subscribe<nav_msgs::Odometry>("uav2/mavros/local_position/odom", 1, plane_pos_cos::Publisher local_pos_pub2 = nh2.advertise<geometry_msgs::PoseStamped>("/uav2/mavros/setpoint_position/local", 10, ros::ServiceClient arming_client2 = nh2.serviceClient<mavros_msgs::CommandBool>("/uav2/mavros/cmd/arming");
ros::ServiceClient set_mode_client2 = nh2.serviceClient<mavros_msgs::SetMode>("/uav2/mavros/set_mode");
\frac{124}{125}
127
128
130
                                       //uav3/ros::NodeHandle nh3;
ros::NodeHandle nh3;
ros::Subscriber state_sub3 = nh3.subscribe<mavros_msgs::State>("uav3/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub3 = nh3.subscribe<nav_msgs::Odometry>("uav3/mavros/local_position/odom", 1, plane_pos_ros::Publisher local_pos_pub3 = nh3.advertise<geometry_msgs::PoseStamped>("/uav3/mavros/setpoint_position/local", 1
ros::ServiceClient arming_client3 = nh3.serviceClient<mavros_msgs::CommandBool>("/uav3/mavros/cmd/arming");
ros::ServiceClient set_mode_client3 = nh3.serviceClient<mavros_msgs::SetMode>("/uav3/mavros/set_mode");
132
133
135
 136
\frac{137}{138}
                                       //uav4
ros::NodeHandle nh4;
ros::Subscriber state_sub4 = nh4.subscribe<mavros_msgs::State>("uav4/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub4 = nh4.subscribe<nav_msgs::Odometry>("uav4/mavros/local_position/odom", 1, plane_pos_cos::Publisher local_pos_pub4 = nh4.advertise<geometry_msgs::PoseStamped>("/uav4/mavros/setpoint_position/local", 10 ros::ServiceClient arming_client4 = nh4.serviceClient<mavros_msgs::CommandBool>("/uav4/mavros/cmd/arming");
ros::ServiceClient set_mode_client4 = nh4.serviceClient<mavros_msgs::SetMode>("/uav4/mavros/set_mode");
 139
 140
141
142
 143
144
145
 146
                                      //uav5
ros::NodeHandle nh5;
ros::Subscriber state_sub5 = nh5.subscribe<mavros_msgs::State>("uav5/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub5 = nh5.subscribe<nav_msgs::Odometry>("uav5/mavros/local_position/odom", 1, plane_pos_cos::Publisher local_pos_pub5 = nh5.advertise<geometry_nesgs::PoseStamped>("uav5/mavros/setpoint_position/local", 10 ros::ServiceClient arming_client5 = nh5.serviceClient<navros_msgs::CommandBool>("/uav5/mavros/cmd/arming");
ros::ServiceClient set_mode_client5 = nh5.serviceClient<navros_msgs::SetMode>("/uav5/mavros/set_mode");
147
148
 149
150
151
 153
                                        //uav6
155
 156
                                       ros::Subscriber state_sub6 = nh6.subscribe<mavros_msgs::State>("uav6/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub6 = nh6.subscribe<nav_msgs::Odometry>("uav6/mavros/local_position/odom", 1, plane_pos_cros::Publisher local_pos_pub6 = nh6.advertise<geometry_msgs::PoseStamped>("/uav6/mavros/setpoint_position/local", 10
ros::ServiceClient arming_client6 = nh6.serviceClient<mavros_msgs::CommandBool>("/uav6/mavros/cmd/arming");
ros::ServiceClient set_mode_client6 = nh6.serviceClient<mavros_msgs::SetMode>("/uav6/mavros/set_mode");
157
158
 159
160
 161
                                        //uav7
ros::NodeHandle nh7;
163
 164
                                       ros::Subscriber state_sub7 = nh7.subscribe<mavros_msgs::State>("uav7/mavros/state", 10, state_cb);
ros::Subscriber local_pos_sub7 = nh7.subscribe<nav_msgs::Odometry>("uav7/mavros/local_position/odom", 1, plane_pos_cros::Publisher local_pos_pub7 = nh7.advertise<geometry_msgs::PoseStamped>("/uav7/mavros/setpoint_position/local", 10
ros::ServiceClient arming_client7 = nh7.serviceClient<mavros_msgs::CommandBool>("/uav7/mavros/cmd/arming");
ros::ServiceClient set_mode_client7 = nh7.serviceClient<mavros_msgs::SetMode>("/uav7/mavros/set_mode");
165
166
168
 169
\frac{170}{171}
                                        ros::Rate rate(10.0):
                     for (int i = 0; i < 8; ++i){
	x_i ini[i] = r*cos(pi/4*i);
	y_i ini[i] = r*sin(pi/4*i);
175
 176
177
178
                                        // wait for FCU connection
while(ros::ok() && !current_state.connected){
    ros::spinOnce();
    rate.sleep();
 179
 180
181
 182
\begin{array}{c} 183 \\ 184 \end{array}
                                      pose0.pose.position.x = 0;
pose0.pose.position.y = 0;
pose0.pose.position.z = z_lock;
pose1.pose.position.x = 0;
pose1.pose.position.x = 0;
pose1.pose.position.y = 0;
pose1.pose.position.z = z_lock;
pose2.pose.position.x = 0;
pose2.pose.position.x = 0;
pose2.pose.position.y = 0;
pose3.pose.position.x = 0;
pose3.pose.position.x = 0;
pose3.pose.position.x = 0;
pose4.pose.position.y = 0;
pose4.pose.position.x = 0;
pose4.pose.position.x = 0;
pose5.pose.position.x = 0;
pose5.pose.position.y = 0;
pose5.pose.position.y = 0;
pose5.pose.position.x = 0;
pose5.pose.position.y = 0;
pose6.pose.position.y = 0;
pose6.pose.position.x = 0;
pose6.pose.position.y = 0;
pose6.pose.position.y = 0;
pose6.pose.position.y = 0;
pose6.pose.position.y = 0;
pose7.pose.position.y = z_lock;
 \frac{187}{188}
 190
 \frac{190}{191}
 195
 198
 199
\frac{200}{201}
\frac{201}{202}
\frac{204}{205}
208
\frac{1}{209}
                                       //send a few setpoints before starting for (int i = 10; ros::ok() && i > 0; —i) {
    local_pos_pub0.publish(pose0);
    local_pos_pub1.publish(pose1);
    local_pos_pub2.publish(pose2);
    local_pos_pub3.publish(pose3);
    local_pos_pub4.publish(pose4);
    local_pos_pub5.publish(pose5);
    local_pos_pub6.publish(pose6);
    local_pos_pub7.publish(pose7);
    ros::spinOnce();
    rate.sleep();
210
211
213
 214
216
217
219
 220
                                                          rate.sleep();
```

```
224
                                              mavros_msgs::SetMode offb_set_mode;
offb_set_mode.request.custom_mode = "OFFBOARD";
\frac{224}{225}
\frac{225}{226}
                                              mavros_msgs::CommandBool arm_cmd;
arm_cmd.request.value = true;
\frac{227}{228}
229
 \frac{230}{230}
                                                ros::Time last_request = ros::Time::now();
231
                                               \begin{array}{c} \mathrm{while}\,(\,\mathrm{ros}::\mathrm{ok}\,(\,)\,)\,\{\\ \mathrm{if}\,(\,\,\mathrm{curren}\,\underline{t}\,\_\mathrm{s}\,\\ \end{array}
 232
233
                                                                                        current_state.mode != "OFFBOARD" &&
  (ros::Time::now() - last_request > ros::Duration(5.0))){
  if( set_mode_client0.call(offb_set_mode) &&
    set_mode_client1.call(offb_set_mode) &&
    set_mode_client2.call(offb_set_mode) &&
    set_mode_client3.call(offb_set_mode) &&
    set_mode_client4.call(offb_set_mode) &&
    set_mode_client4.call(offb_set_mode) &&
    set_mode_client5.call(offb_set_mode) &&
    set_mode_client6.call(offb_set_mode) &&
    set_mode_client7.call(offb_set_mode) &&
    set_mode_client7.call(offb_set_mode) &&
    offb_set_mode.response.mode_sent){
        ROS_INFO("Offboard_enabled: 8");
}
                                                                                                                                  _state.mode != "OFFBOARD" &&
234
 235
236
237
 \frac{1}{238}
239
240
241
243
244
                                                                                          last_request = ros::Time::now();
246
247
                                                                   } else {
    if (!current_state.armed &&
                                                                                                               !current_state.armed && ros::Duration(5.0))){
if(arming_client0.call(arm_cmd) && arming_client1.call(arm_cmd) && arming_client2.call(arm_cmd) && arming_client3.call(arm_cmd) && arming_client3.call(arm_cmd) && arming_client4.call(arm_cmd) && arming_client5.call(arm_cmd) && arming_client6.call(arm_cmd) && arming_client
249
 250
251
252
 253
254
255
                                                                                                                                    arming_client6.call(arm_cmd) && arming_client7.call(arm_cmd) &&
256
257
                                                                                                                                   arm_cmd.response.success){
ROS_INFO("Vehicle armed: 8");
258
\frac{259}{259}
260
261
                                                                                                                last_request = ros::Time::now();
262
                                                                                         }
263
264
                                                                     local_pos_pub0.publish(pose0);
                                                                     local_pos_pub1.publish(pose1)
local_pos_pub2.publish(pose2)
local_pos_pub3.publish(pose3)
265
266
267
                                                                     local_pos_pub4.publish(pose4);
local_pos_pub4.publish(pose4);
local_pos_pub5.publish(pose5);
local_pos_pub6.publish(pose6);
local_pos_pub7.publish(pose7);
if (pose_odom0.pose.pose.position.z > 1.8) {
    ROS_INFO("plane_takeoff_!");
268
269
270
271
272
\frac{273}{274}
                                                                                          break:
276
                                                                     ros::spinOnce();
277
                                                                     rate.sleep();
279
                         double tmp;
srand((unsigned)time(NULL));
for (int i = 0; i < 8; ++i){
    tmp = 0;
    for (int j = 0; j < 7; ++j){
        v2(i, j) = rand() / float(RAND_MAX);
        tmp -= v2(i, j);
}</pre>
 280
281
282
283
 \frac{284}{284}
285
286
287
288
                                                                     \begin{array}{l} \label{eq:continuous_problem} \begin{cases} v2(i\,,\,\,7) = tmp; \\ v2\,.\,block(i\,,\,\,0,\,\,1,\,\,8) = v2\,.\,block(i\,,\,\,0,\,\,1,\,\,8); \\ was v2\,.\,block(i\,,\,\,0,\,\,1,\,\,8) * v2\,.\,block(i\,,\,\,0,\,\,1,\,\,8). \\ v2(i\,) = tmp(0\,,\,\,0); \\ v2(i\,) = tmp(0\,,\,\,0); \\ v3(i\,) = 0; \\ v4(i\,) = 0
289
\frac{1}{290}
291
292
293
294
295
                                                                     omega2(i) = 0;
 296
297
\frac{1}{298}
299
                                                while (ros::ok()) {
300
                                                                     update_state();
                                                                    for (int i = 0; i < 8; ++i){ for (int j = 0; j < 8; ++j){ if (x(j) - x(i) > 0){ psi(i, j) = atan((y(j) - y(i))/(x(j) - x(i))); }}
302
303
304
305
                                                                                                             306
307
308
309
310
311
                                                                                                                                                          psi(i, j) = atan((y(j) - y(i))/(x(j) - x(i))) - pi;
312
314
                                                                                                               }
                                                                                          }
315
316
                                                                                          318
```

```
\begin{array}{l} \mathrm{if} \ (j = \mathrm{i} - 1 \ || \ j = \mathrm{i} - 2 \ || \ j = \mathrm{i} + 1 \ || \ j = \mathrm{i} + 2 \ || \ j = \mathrm{i} + 7 \ || \ j = \mathrm{i} + 6 \ || \ j = \mathrm{i} - 7 \ || \ j \\ \mathrm{lap}(\mathrm{i}, \ j) = -\mathrm{exp}(-\mathrm{c1}*(\mathrm{psi}(\mathrm{i}, \ j) - \mathrm{phi}(\mathrm{i}))*(\mathrm{psi}(\mathrm{i}, \ j) - \mathrm{phi}(\mathrm{i})) - \mathrm{c2}*\mathrm{phi}(\mathrm{i})*\mathrm{phi}(\mathrm{i}) \backslash \\ - \mathrm{c3}*(\mathrm{theta}(\mathrm{j}) - \mathrm{theta}(\mathrm{i}))*(\mathrm{theta}(\mathrm{j}) - \mathrm{theta}(\mathrm{i}))) \backslash \\ -\mathrm{exp}(-\mathrm{c1}*(\mathrm{psi}(\mathrm{j}, \ \mathrm{i}) - \mathrm{phi}(\mathrm{j}))*(\mathrm{psi}(\mathrm{j}, \ \mathrm{i}) - \mathrm{phi}(\mathrm{j})) - \mathrm{c2}*\mathrm{phi}(\mathrm{j})*\mathrm{phi}(\mathrm{j}) \backslash \\ - \mathrm{c3}*(\mathrm{theta}(\mathrm{j}) - \mathrm{theta}(\mathrm{i}))*(\mathrm{theta}(\mathrm{j}) - \mathrm{theta}(\mathrm{i}))); \end{array}
319
320
321
322
323
324

\begin{cases}
else & \{ \\
lap(i, j) = 0;
\end{cases}

325
326
328
329
                                                         \begin{cases} \text{for (int i = 0; i < 8; ++i)} \{ \\ \text{for (int j = 0; j < 8; ++j)} \{ \\ \text{if (i = j)} \{ \\ \text{continue:} \end{cases} 
330
331
332
333
334
335
                                                                                            else {
336
                                                                                                            lap(i, i) = lap(i, j);
337
338
                                                                         }
\frac{339}{340}
                                                        }
                                                      for (int i = 0; i < 8; ++i){
    z1. block(i, 0, 1, 8) += gama*(v2.block(i, 0, 1, 8) - z1.block(i, 0, 1, 8))\
    - Kp*(4*z1.block(i, 0, 1, 8) - z1.block(((i-1>=0) ? i-1 : 7+i) , 0, 1, 8)\
    - z1.block(((i-2>=0) ? i-2 : 6+i) , 0, 1, 8)\
    - z1.block(((i+2 < 8) ? i+1 : i-7) , 0, 1, 8)\
    - z1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\
    + Ki*(4*emegal.block(i, 0, 1, 8) - omegal.block(((i-1>=0) ? i-1 : 7+i) , 0, 1, 8)\
    - omegal.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\
    - omegal.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\
    - omegal.block(((i+2 < 8) ? i+1 : i-7) , 0, 1, 8)\
    - omegal.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\;
    omegal.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\;
    - z1.block(((i-2>=0) ? i-2 : 6+i) , 0, 1, 8)\;
    - z1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - z1.block(((i+2 < 8) ? i+1 : i-7) , 0, 1, 8)\\
    - z1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - z1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - z1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0, 1, 8)\\
    - x1.block(((i+2 < 8) ? i+2 : i-6) , 0,
341
342
343
345
346
348
349
                                                                                                                                                                                                                                                                                                             (8) - z1.block(((i-1 >= 0) ? i-1 : 7+i) , 0, 1, 8)
351
352
354
355
356
\begin{array}{c} 357 \\ 358 \end{array}
359
\frac{360}{361}
                                                                          362
363
364
\frac{365}{366}
                                                        }
                                                                        367
                                                         for (int i = 0; i < 8; ++i){
368
 369
\begin{array}{c} 370 \\ 371 \end{array}
\begin{array}{c} 373 \\ 374 \end{array}
376
377
                                                                         }
\frac{379}{380}
                                                        }
381
                                                         update_pub();
                                                         local_pos_pub0.publish(pose0);
local_pos_pub1.publish(pose1);
382
383
                                                         local_pos_pub2.publish(pose2);
local_pos_pub3.publish(pose3);
local_pos_pub3.publish(pose4);
384
385
386
                                                        local_pos_pub3.publish(pose5);
local_pos_pub3.publish(pose6);
local_pos_pub3.publish(pose7);
387
388
389
                                                        ros::spinOnce();
rate.sleep();
390
391
392
                                      }
\frac{393}{394}
                                      return 0;
\frac{395}{396}
                    \begin{array}{c} \mbox{void update\_theta()} \{ \\ \mbox{for (int } i = 0; \ i < 8; \ +\!\!\!+\!\!\!i) \{ \\ \mbox{if } (x(i) > 0) \{ \\ \mbox{theta(i)} = \mbox{atan(y(i)/x(i))}; \end{array}
397
398
399
\begin{array}{c} 400 \\ 401 \end{array}
                                                          \begin{array}{c} \text{felse } \{ \\ \text{if } (y(\texttt{i}) > 0) \{ \\ \text{theta}(\texttt{i}) = \operatorname{atan}(y(\texttt{i})/x(\texttt{i})) \, + \, \operatorname{pi}; \end{array} 
\begin{array}{c} 403 \\ 404 \end{array}
405
                                                                          \begin{cases} else & \{ \\ theta(i) = atan(y(i)/x(i)) - pi; \end{cases}
406
407
408
409
                                      }
410
                    }
412
                      \begin{array}{lll} void & update\_state() \{ \\ & x(0) = pose\_odom0.pose.pose.position.x + x\_ini \begin{bmatrix} 0 \\ y(0) = pose\_odom0.pose.pose.position.y + y\_ini \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \end{array}
```

```
z(0) = pose_odom0.pose.pose.position.z;
phi(0) = quaternion2euler(pose_odom0.pose.pose.orientation).z;
x(1) = pose_odom1.pose.pose.position.x + x_ini[1];
y(1) = pose_odom1.pose.pose.position.y + y_ini[1];
z(1) = pose_odom1.pose.pose.position.z;
phi(1) = quaternion2euler(pose_odom1.pose.pose.orientation).z;
y(2) = pose_odom2.pose_pose_position.y + x_ini[2];
416
417
419
420
                                            x(2) = pose\_odom2.pose.pose.position.x + x\_ini[2];

y(2) = pose\_odom2.pose.pose.position.y + y\_ini[2];

z(2) = pose\_odom2.pose.pose.position.z;
 422
423
 424
                                            \begin{array}{lll} \text{phi}(2) &= \text{quaternion2euler(pose\_odom2.pose.pose.orientation).z;} \\ \text{x}(3) &= \text{pose\_odom3.pose.pose.position.x} + \text{x\_ini[3];} \\ \text{y}(3) &= \text{pose\_odom3.pose.pose.position.y} + \text{y\_ini[3];} \\ \end{array}
425
426
                                            y(3) = pose_odom3.pose.pose.pose.position.y + y_nnr[3],
z(3) = pose_odom3.pose.pose.position.z;
phi(3) = quaternion2euler(pose_odom3.pose.pose.orientation).z;
x(4) = pose_odom4.pose.pose.position.x + x_ini[4];
y(4) = pose_odom4.pose.pose.position.y + y_ini[4];
428
429
 430
 431
                                             z(4) = pose_odom4.pose.pose.position.z;
phi(4) = quaternion2euler(pose_odom4.pose.pose.orientation).z;
432
 433
                                            x(5) = \text{pose\_odom5.pose.pose.position.x} + x\_\text{ini} [5];

y(5) = \text{pose\_odom5.pose.pose.position.y} + y\_\text{ini} [5];

z(5) = \text{pose\_odom5.pose.pose.position.z};
 434
435
 436
                                            437
438
 439
 440
                                                                    = pose_odom6.pose.pose.position.z;
                                            phi(6) = quaternion2euler(pose_odom6.pose.pose.orientation).z;
x(7) = pose_odom7.pose.pose.position.x + x_ini[7];
y(7) = pose_odom7.pose.pose.position.y + y_ini[7];
z(7) = pose_odom7.pose.pose.position.z;
phi(7) = quaternion2euler(pose_odom7.pose.pose.orientation).z;
441
442
 443
444
 445
 446
                                               update_theta();
447
                        }
 448
449
                         geometry_msgs::Vector3 quaternion2euler(geometry_msgs::Quaternion quater){
                                            letry_insgs:: vector3 quaternionzener (geometry_insgs:: Quaternion quater) {
    geometry_msgs:: Vector3 temp;
    temp.x = atan2(2.0 * (quater.w * quater.x + quater.y * quater.z), 1.0 - 2.0 * (quater.x * quater.x + quater.y * quater.y + quater.y = asin(2.0 * (quater.w * quater.y - quater.z * quater.x));
    temp.z = atan2(2.0 * (quater.w * quater.z + quater.x * quater.y), 1.0 - 2.0 * (quater.y * quater.y + quater.z * quater.x + quater.y + quater.
 450
451
452
454
                                              return temp:
 455
                        }
456
 457
                         geometry_msgs::Quaternion euler2quaternion(geometry_msgs::Vector3 euler)
\frac{458}{459}
                                             \begin{array}{l} {\rm geometry\_msgs::Quaternion\ temp;} \\ {\rm temp.w=cos(euler.x/2)*cos(euler.y/2)*cos(euler.z/2)} + \sin({\rm euler.x/2})*\sin({\rm euler.y/2})*\sin({\rm euler.z/2}); \\ {\rm temp.x=sin(euler.x/2)*cos(euler.y/2)*cos(euler.z/2)} - \cos({\rm euler.x/2})*\sin({\rm euler.y/2})*\sin({\rm euler.z/2}); \\ {\rm temp.y=cos(euler.x/2)*sin(euler.y/2)*cos(euler.z/2)} + \sin({\rm euler.x/2})*\cos({\rm euler.z/2})*\sin({\rm euler.z/2}); \\ {\rm temp.z=cos(euler.x/2)*cos(euler.y/2)*sin(euler.z/2)} - \sin({\rm euler.x/2})*\sin({\rm euler.z/2})*\cos({\rm euler.z/2}); \\ {\rm temp.z=cos(euler.x/2)*cos(euler.y/2)*sin(euler.z/2)} - \sin({\rm euler.x/2})*\sin({\rm euler.z/2})*\cos({\rm euler.z/2}); \\ {\rm temp.z=cos(euler.x/2)*cos(euler.z/2)*sin(euler.z/2)} + \sin({\rm euler.z/2})*\sin({\rm euler.z/2})*\sin({\rm euler.z/2})*sin({\rm euler.z/2}); \\ {\rm temp.z=cos(euler.x/2)*cos(euler.z/2)*sin(euler.z/2)} + \sin({\rm euler.z/2})*sin({\rm euler.z/2}
 460
461
462
 463
464
                                              return temp;
 465
                        }
466
                         void update_pub(){
    geometry_msgs::Vector3 euler;
    euler = quaternion2euler(pose_odom0.pose.pose.orientation);
468
 469
470
                                               euler.z \stackrel{\cdot}{+}= k_w*u_phi(0);
                                              pose0.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom1.pose.pose.orientation);
471
 472
 473
                                               euler.z \stackrel{\cdot}{+}= k_w*u_phi(1);
474
                                              pose1.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom2.pose.pose.orientation);
 475
 476
                                              \texttt{euler.z} \stackrel{.}{+}= \texttt{k\_w*u\_phi(2)};
                                              pose2.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom3.pose.pose.orientation);
477
 478
 479
                                              euler.z \stackrel{\cdot}{+}= k_w*u_phi(3);
                                              pose3.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom4.pose.pose.orientation);
480
 481
                                              euler.z += k_w*u_phi(4);
 482
                                              pose4.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom5.pose.pose.orientation);
483
 484
                                              euler.z += k_w*u_phi(5);
 485
                                              pose5.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom6.pose.pose.orientation);
486
 487
                                              euler.z += k_w*u_phi(6);
 488
                                              pose6.pose.orientation = euler2quaternion(euler);
euler = quaternion2euler(pose_odom7.pose.pose.orientation);
489
 490
                                          euler = quaternion2euler (pose_odom*.pose.pose.orientation); euler .z += k_w*u_phi(7); pose7.pose.orientation = euler2quaternion(euler); pose0.pose.position.x = r*cos(theta(0) + k_v*u_theta(0) + w) - x_ini [0]; pose1.pose.position.x = r*cos(theta(1) + k_v*u_theta(1) + w) - x_ini [1]; pose2.pose.position.x = r*cos(theta(2) + k_v*u_theta(2) + w) - x_ini [2]; pose3.pose.position.x = r*cos(theta(3) + k_v*u_theta(2) + w) - x_ini [2]; pose3.pose.position.x = r*cos(theta(3) + k_v*u_theta(3) + w) - x_ini [3]; pose4.pose.position.x = r*cos(theta(4) + k_v*u_theta(4) + w) - x_ini [4]; pose5.pose.position.x = r*cos(theta(5) + k_v*u_theta(5) + w) - x_ini [5]; pose6.pose.position.x = r*cos(theta(6) + k_v*u_theta(6) + w) - x_ini [6]; pose7.pose.position.x = r*cos(theta(7) + k_v*u_theta(0) + w) - x_ini [7]; pose0.pose.position.y = r*sin(theta(0) + k_v*u_theta(0) + w) - y_ini [0]; pose1.pose.position.y = r*sin(theta(1) + k_v*u_theta(1) + w) - y_ini [1]; pose2.pose.position.y = r*sin(theta(2) + k_v*u_theta(2) + w) - y_ini [3]; pose4.pose.position.y = r*sin(theta(3) + k_v*u_theta(4) + w) - y_ini [3]; pose6.pose.position.y = r*sin(theta(4) + k_v*u_theta(4) + w) - y_ini [4]; pose5.pose.position.y = r*sin(theta(5) + k_v*u_theta(6) + w) - y_ini [5]; pose6.pose.position.y = r*sin(theta(5) + k_v*u_theta(6) + w) - y_ini [6]; pose7.pose.position.y = r*sin(theta(7) + k_v*u_theta(6) + w) - y_ini [7]; pose0.pose.position.y = r*sin(theta(7) + k_v*u_theta(7) + w) - y_ini [7]; pose0.pose.position.z = z_lock; pose2.pose.position.z = z_lock; pose2.pose.position.z = z_lock; pose3.pose.position.z = z_lock; pose4.pose5.pose4.pose5.pose5.position.z = z_lock; pose63.pose5.
                                              euler.z += k_w*u_phi(7);
 491
492
493
 494
495
496
 497
498
499
501
502
504
505
507
508
 509
510
                                              pose3.pose.position.z = z\_lock;
```