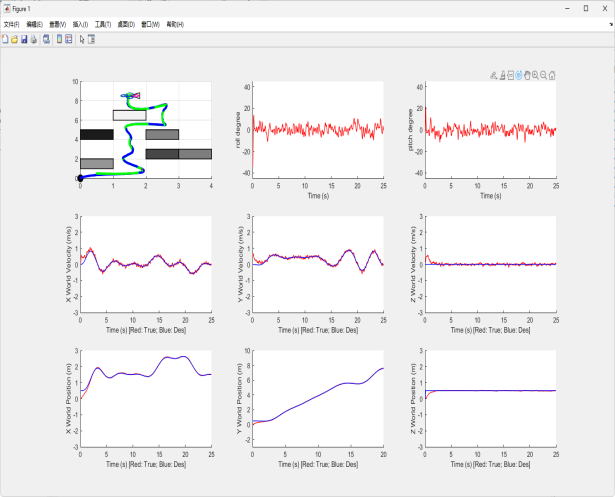
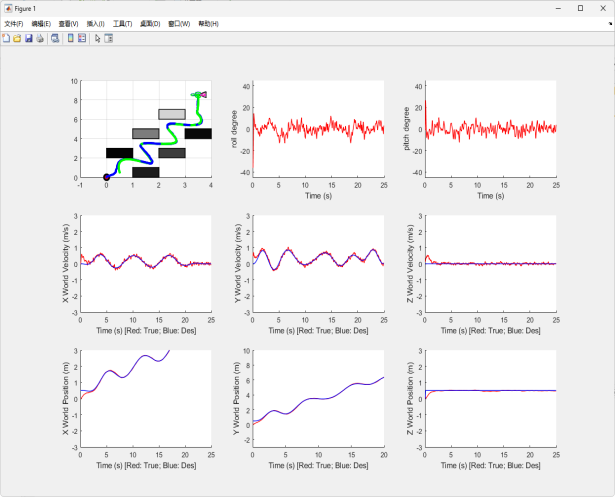
proj1phase3-徐涵-202293010207

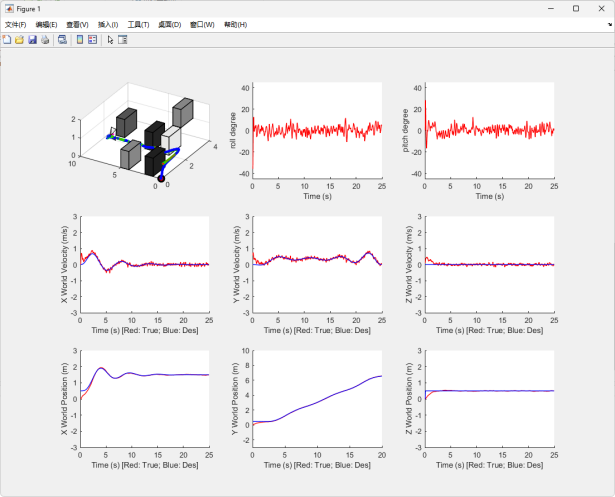
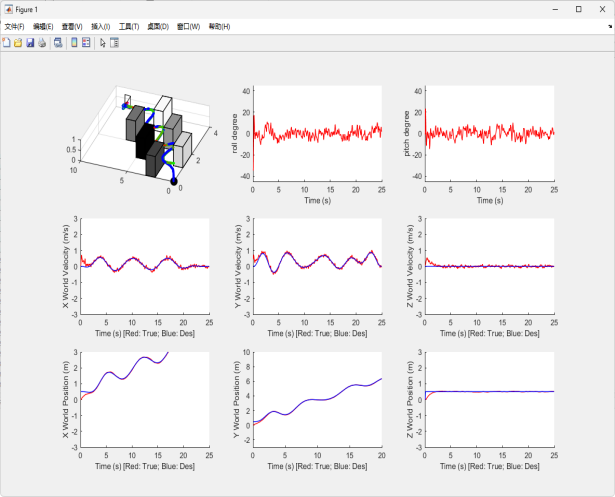
五张fig结果展示：Matlab的fig文件以及完整截图保存在assets目录中。

2d:

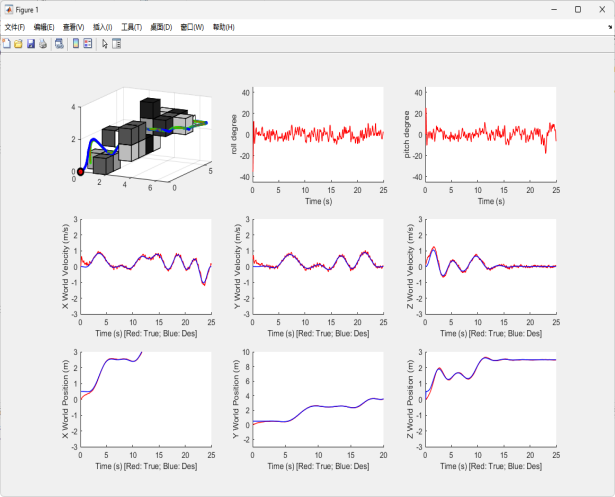
 

map1 map2

3d:

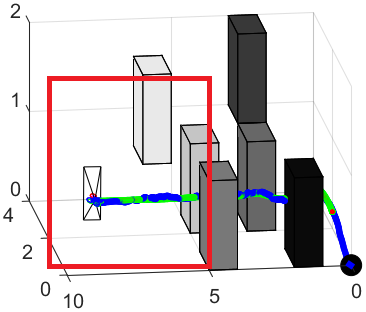
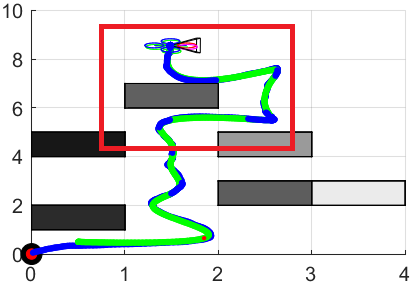
 

map1 map2



map3

2d中map1和3d中map1的障碍物的xy坐标是完全一致的，然而，2d的map1中，飞行器在最后一个障碍物处进行拐弯，在3d的map1中，飞行器在最后一个障碍物没有拐弯，这说明我的2d算法和3d算法都是切实有效的。



2d的map1（拐弯） 3d的map1（不拐弯）

代码文件解释：

path\_from\_A\_star.m和path\_from\_A\_star\_2d.m分别是2d和3d下的路径规划函数，path\_from\_A\_star\_2d.m中的代码主要依据path\_from\_A\_star.m裁剪z坐标后的代码。由于test\_trajectory.m只是针对3d点输入的轨迹生成，所以需要对path\_from\_A\_star\_2d.m生成的路点填充z值，这里选择填充z=1。代码部分如下：

*% set value==1 as the default z*

*Optimal\_path = [Optimal\_path, ones(size(Optimal\_path,1),1)];*

test\_trajectory.m和test\_trajectory.m分别可以开始2d和3d的仿真。trajectory\_generator.m在phase2的基础上，增加了图像的绘制，显示障碍物。代码部分如下：

*subplot(h);*

*hold on;*

*for i = 2: size(test\_map, 1) - 1*

*cube\_x = test\_map(i ,1) -1;*

*cube\_y = test\_map(i ,2) -1;*

*cube\_z = test\_map(i ,3) -1;*

*color\_value = rand;*

*plotcube([1,1,1], [cube\_x,cube\_y,cube\_z],1,[color\_value,color\_value,color\_value]);*

*end*

controller.m仍然遵循phase1。代码部分如下：  
*%PD控制参数*

*kp\_xyz = [10.5; 10.5; 30.5]; % Position gain  
kd\_xyz = [10.5; 10.5; 15.5]; % Velocity gain  
kp\_angle = [900; 900; 900]; % Attitude gain  
kd\_angle = [50; 50; 50]; % Angular velocity gain*