第二章作业

1.作业说明

- 运行环境：ROS2 Humble
- 代码详见`src`目录，修改较大，将原始代码移植到ROS2中，`astar_path_finder.hpp/cpp`为A*算法实现，`jps_path_finder.hpp/cpp`为JPS算法实现。

遇到的问题及解决方法：

问题：生成的路径会与地图方块接触，甚至穿过。

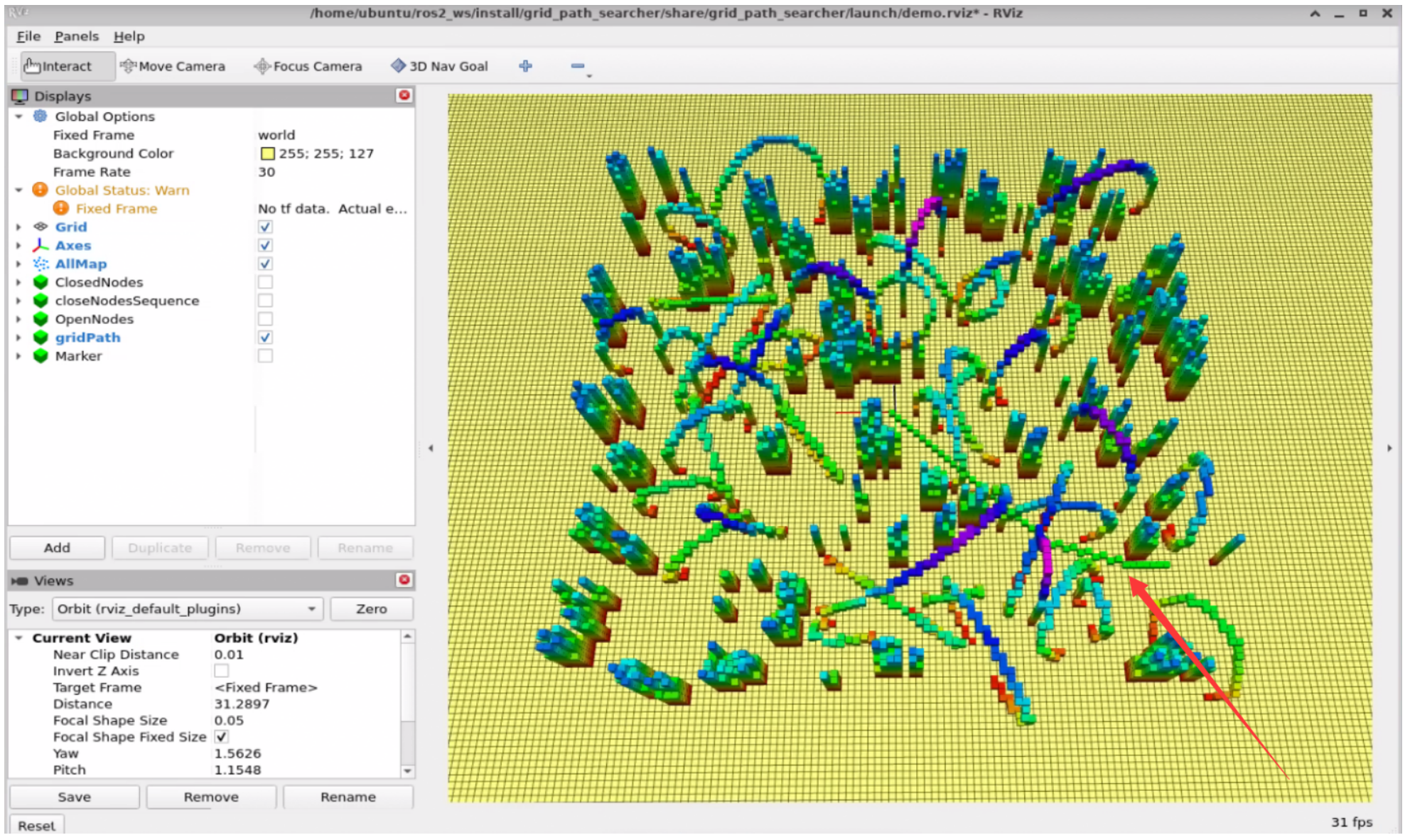
- 解决方法：这是因为rviz可视化的地图与astar算法进行搜索的网格地图不同导致的， 修改生成地图的代码`random_complex_generator.cpp`，对生成的点云地图进行了网格对齐。

2.A*算法流程及运行结果

运行命令

```
ros2 launch grid_path_searcher demo.launch.py test_case:=astar
```

随机在地图上选点，得到rviz可视化运行结果如下图（路径见箭头符号）：



算法流程

- 略

3.启发式函数对A*运行效率的影响

运行命令

```
ros2 launch grid_path_searcher demo.launch.py test_case:=astar_heuristic_function
```

在随机在地图上选点，得到对比结果如下（一次）：

```
[rviz2-4] [INFO] [1667658074.573999280] [rviz2]: Setting goal: Frame:world, Position(-3.64547, -8.92343, 0.94), Orientation(0, 0,
-0.958765, 0.284199) = Angle: -2.56525
[demo_node-1] [INFO] [1667658074.574665638] [demo_node]: receive the way-points
[demo_node-1] [INFO] [1667658074.574766724] [demo_node]: A* algorithm (Manhattan):
[demo_node-1] [INFO] [1667658075.330269680] [demo_node]: path nodes size: 46, visited nodes size: 93789
[demo_node-1] [INFO] [1667658075.330320121] [demo_node]: time is 742.565684 ms, path cost if 10.828281 m
[demo_node-1] [INFO] [1667658075.363033620] [demo_node]: A* algorithm (Euclidean):
[demo_node-1] [INFO] [1667658076.408688645] [demo_node]: path nodes size: 46, visited nodes size: 127519
[demo_node-1] [INFO] [1667658076.408736467] [demo_node]: time is 1029.565868 ms, path cost if 10.828281 m
[demo_node-1] [INFO] [1667658076.446094093] [demo_node]: A* algorithm (Diagonal):
[demo_node-1] [INFO] [1667658077.382750413] [demo_node]: path nodes size: 46, visited nodes size: 120404
[demo_node-1] [INFO] [1667658077.382816087] [demo_node]: time is 922.468407 ms, path cost if 10.828281 m
[demo_node-1] [INFO] [1667658077.421066150] [demo_node]: A* algorithm (None):
[demo_node-1] [INFO] [1667658079.316295595] [demo_node]: path nodes size: 46, visited nodes size: 252986
[demo_node-1] [INFO] [1667658079.316343590] [demo_node]: time is 1870.109256 ms, path cost if 10.828281 m
```

- 经过多次选点，均得到类似效果，不同启发式函数下的运行效率的关系如下：Manhanttan> Diagonal > Euclidean > None（Dijkstra）
- 其中采用启发式函数相比于不使用启发式函数，运行效率得到大幅度明显提升。

4. Tie Breaker对A*运行效率的影响

运行命令

```
ros2 launch grid_path_searcher demo.launch.py test_case:=astar_tie_breaker
```

在随机在地图上选点，得到对比结果如下（系数为0.001时，即`h = h + cross * 0.001`）：

```
[rviz2-4] [INFO] [1667658194.143508864] [rviz2]: Setting goal: Frame:world, Position(-0.0198011, -6.31268, 0.9), Orientation(0, 0,
 -0.796766, 0.604288) = Angle: -1.84385
[demo_node-1] [INFO] [1667658194.143971103] [demo_node]: receive the way-points
[demo_node-1] [INFO] [1667658194.144067182] [demo_node]: A* algorithm (without Tie Breaker):
[demo_node-1] [INFO] [1667658194.420015391] [demo_node]: path nodes size: 33, visited nodes size: 30724
[demo_node-1] [INFO] [1667658194.420068529] [demo_node]: time is 268.231963 ms, path cost if 6.794938 m
[demo_node-1] [INFO] [1667658194.435001132] [demo_node]: A* algorithm (with Tie Breaker):
[demo_node-1] [INFO] [1667658194.770238199] [demo_node]: path nodes size: 33, visited nodes size: 30656
[demo_node-1] [INFO] [1667658194.770305001] [demo_node]: time is 328.720975 ms, path cost if 6.794938 m
```

在随机在地图上选点，得到对比结果如下（系数为1时，即`h = h + cross * 1`）：

```
[rviz2-4] [INFO] [1667658266.498238052] [rviz2]: Setting goal: Frame:world, Position(6.65412, 5.1574, 1.08), Orientation(0, 0, -0.
509372, 0.860547) = Angle: -1.06891
[demo_node-1] [INFO] [1667658266.498563939] [demo_node]: receive the way-points
[demo_node-1] [INFO] [1667658266.498591111] [demo_node]: A* algorithm (without Tie Breaker):
[demo_node-1] [INFO] [1667658267.081148753] [demo_node]: path nodes size: 34, visited nodes size: 71604
[demo_node-1] [INFO] [1667658267.081200686] [demo_node]: time is 571.873522 ms, path cost if 8.988905 m
[demo_node-1] [INFO] [1667658267.106477961] [demo_node]: A* algorithm (with Tie Breaker):
[demo_node-1] [INFO] [1667658267.230516653] [demo_node]: path nodes size: 34, visited nodes size: 9295
[demo_node-1] [INFO] [1667658267.230567651] [demo_node]: time is 116.392039 ms, path cost if 8.988905 m
```

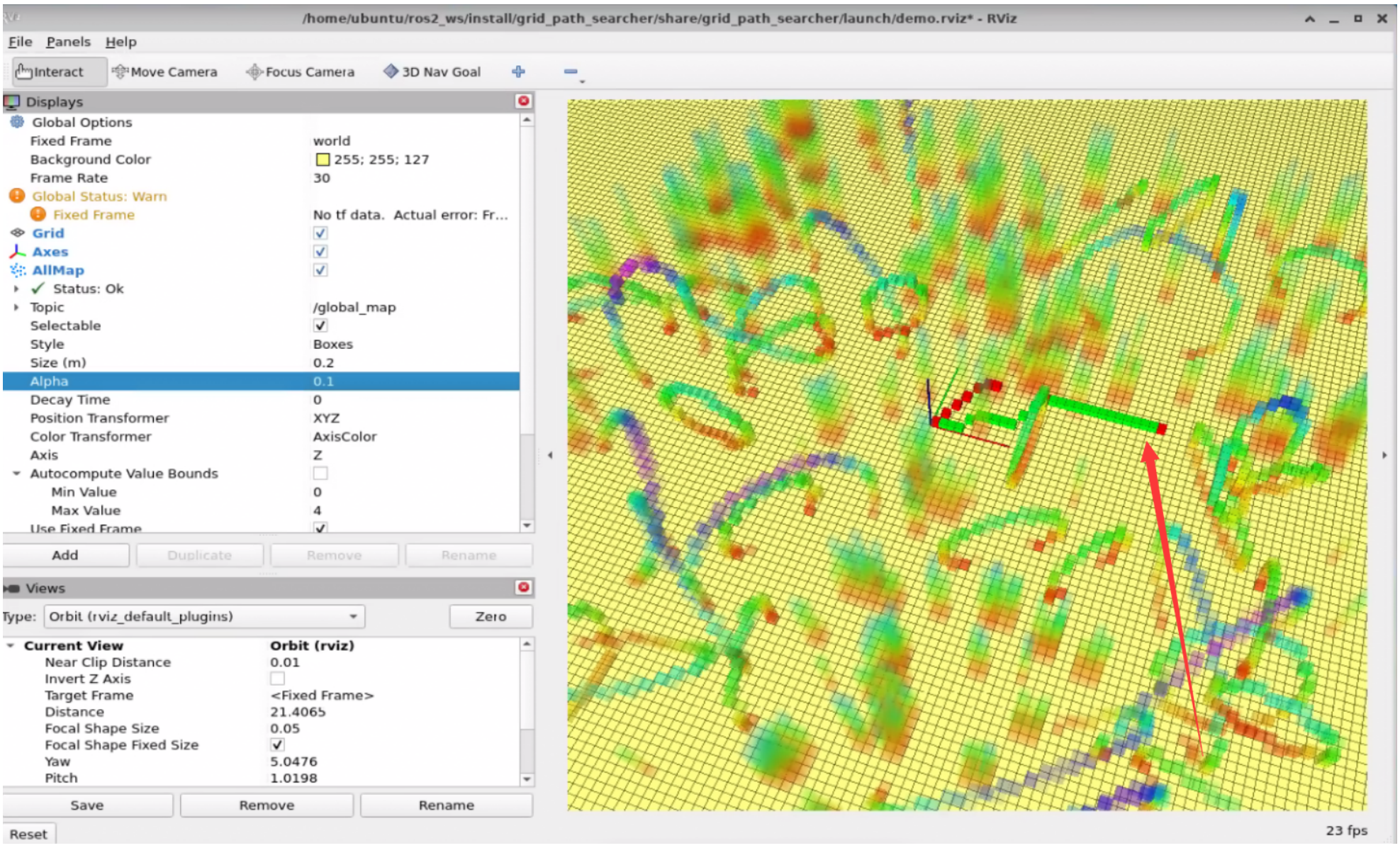
- 在系数为0.001的时候，不但没有A*的效率不但没有提升，反而下降，因为系数太小导致tie breaker并没有发挥作用，而计算tie breaker中的向量积会耗费更多的时间。
- 在系数为1的时候，tie breaker对A*的运行效率提升明显，访问节点的数量显著减少，使得搜索时间大幅度下降。

5. A*调用JPS算法效率分析

运行命令

```
ros2 launch grid_path_searcher demo.launch.py test_case:=astar_jps
```

在随机在地图上选点，rviz可视化结果如下（绿色为astar路径，红色为jps路径）：



运行效率结果如下图所示：

```
[rviz2-4] [INFO] [1667658746.512272329] [rviz2]: 3D Goal Set
[rviz2-4] [INFO] [1667658746.512403902] [rviz2]: Setting goal: Frame:world, Position(5.31256, 1.19299, 0.88), Orientation(0, 0, -0
.13464, 0.990895) = Angle: -0.270099
[demo_node-1] [INFO] [1667658746.512878349] [demo_node]: receive the way-points
[demo_node-1] [INFO] [1667658746.512931914] [demo_node]: A* algorithm:
[demo_node-1] [INFO] [1667658746.678222184] [demo_node]: path nodes size: 27, visited nodes size: 19896
[demo_node-1] [INFO] [1667658746.678290934] [demo_node]: time is 157.068672 ms, path cost if 5.868483 m
[demo_node-1] [INFO] [1667658746.689347416] [demo_node]: JPS algorithm:
[demo_node-1] [INFO] [1667658750.199916613] [demo_node]: path nodes size: 9, visited nodes size: 6230
[demo_node-1] [INFO] [1667658750.199963229] [demo_node]: time is 3506.476884 ms, path cost if 5.868483 m
```

- 不难发现jps的运行效率远低于A*方法，但是jps访问节点数量要小于A*方法，这是因为我们随机生成的地图障碍物较为稀疏，在jump过程中的搜索扩展非常耗时。