机器人路径规划

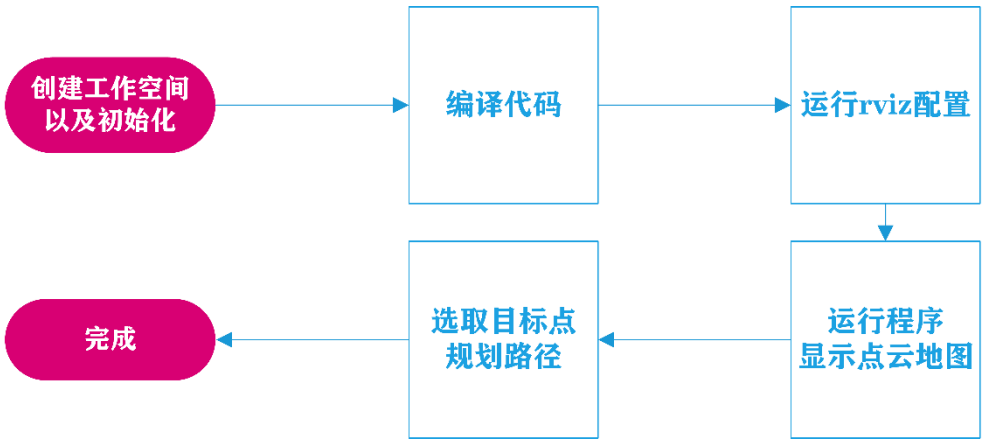
第二章作业

张 文泰

学号：21009101463 |

实验一　ROS 版本作业

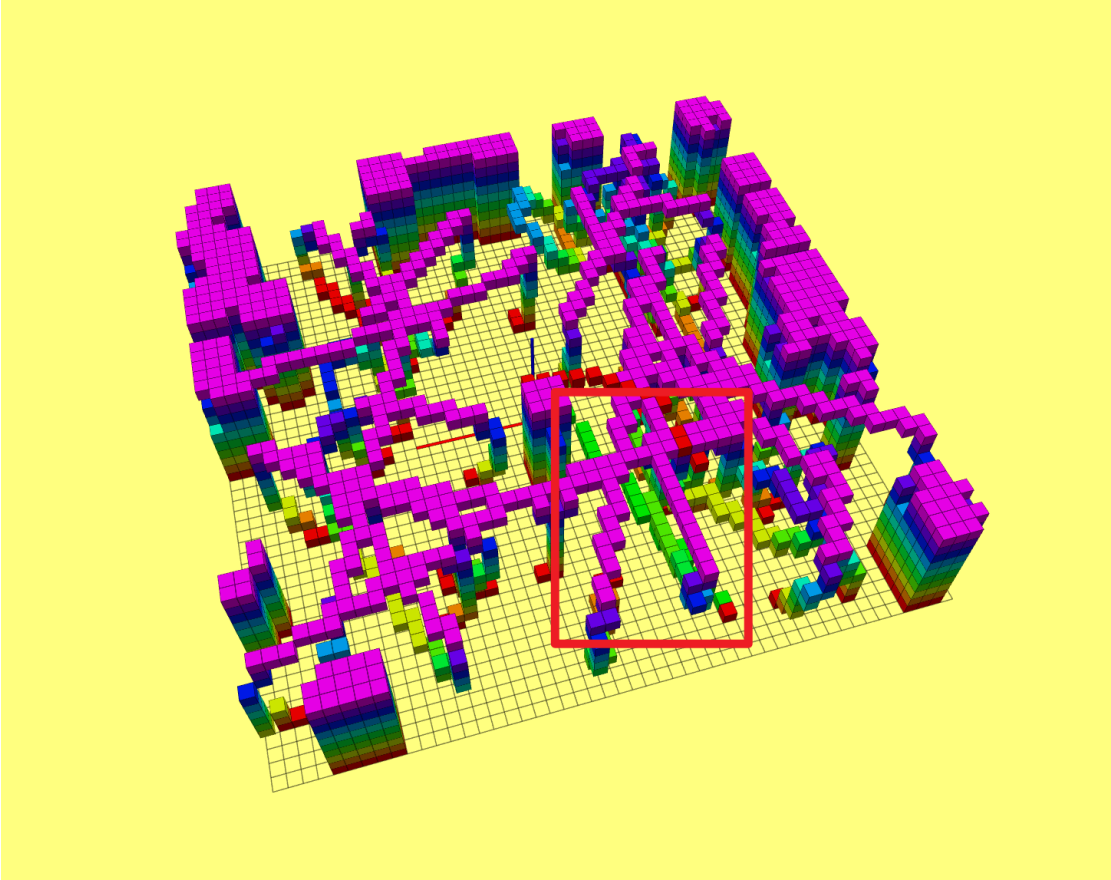
一、 实验过程



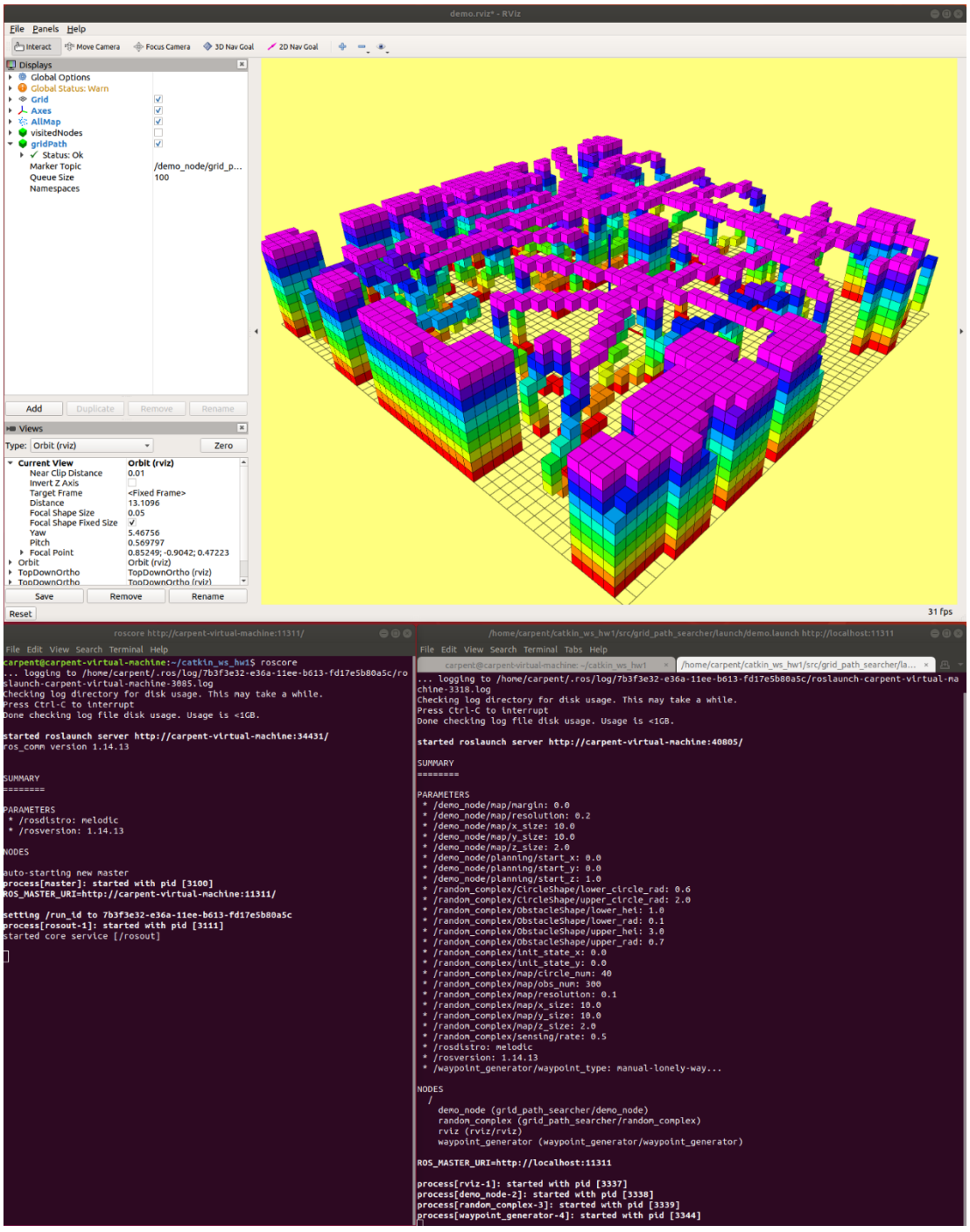
ROS 版本作业操作流程图

二、 实验结果

路径规划结果如图所示：



A*算法路径规划结果，红色框内为路径



地图运行结果及终端配置情况

实验二　MATLAB 版本作业

一、代码补充与分析

由于代码过长，因此分为两部分进行解析：

```
while((xNode ~= xTarget || yNode ~= yTarget) && NoPath == 1)
 exp_array=expand_array(xNode,yNode,path_cost,xTarget,yTarget,CLOSED,MAX_X,MAX_Y);
 exp_count=size(exp_array,1);
 for i=1:exp_count
    flag=0;
    for j=1:OPEN_COUNT
        if(exp_array(i,1) == OPEN(j,2) && exp_array(i,2) == OPEN(j,3) )
            OPEN(j,8)=min(OPEN(j,8),exp_array(i,5));
            if OPEN(j,8)== exp_array(i,5)
                OPEN(j,4)=xNode;
                OPEN(j,5)=yNode;
                OPEN(j,6)=exp_array(i,3);
                OPEN(j,7)=exp_array(i,4);
            end
            flag=1;
        end
    end
    if flag == 0
        OPEN_COUNT = OPEN_COUNT+1;
        OPEN(OPEN_COUNT,:)=insert_open(exp_array(i,1),exp_array(i,2),xNode,yNode,exp_array(i,3),exp_array(i,4),exp_array(i,5));
    end
 end
 index_min_node = min_fn(OPEN,OPEN_COUNT,xTarget,yTarget);
  if (index_min_node ~= -1)

   xNode=OPEN(index_min_node,2);
   yNode=OPEN(index_min_node,3);
   path_cost=OPEN(index_min_node,6);

  CLOSED_COUNT=CLOSED_COUNT+1;
  CLOSED(CLOSED_COUNT,1)=xNode;
  CLOSED(CLOSED_COUNT,2)=yNode;
  OPEN(index_min_node,1)=0;
  else
      NoPath=0;
  end
end
```

第一部分补充代码

整体来看，该部分代码通过一个迭代的搜索过程，不断更新当前节点和路径成本，直到找到从起点到终点的最短路径或者没有可行路径为止。下面是对该部分算法的逻辑框架梳理：

1. 初始化：循环开始前，xNode 和 yNode 代表当前节点的坐标，xTarget 和 yTarget 代表目标节点的坐标，NoPath 作为是否找到路径的标志，为 1 表示还未找到，0 则表示找到。

2. 循环条件：当前节点不是目标节点，并且 NoPath = 1。

3. 扩展节点：使用 expand_array 函数来获取当前节点所有可能的下一步移动，并将结果存储在 exp_array 列表中。

4. 更新 OPEN 列表：OPEN 列表用来存储被发现但还没被评估的节点，遍历 exp_array 中的每个节点，检查它是否已经在 OPEN 列表中，如果在则更新该节点的成本和路径，如果不在则将其添加到 OPEN 列表。

5. 选择下一个节点：使用 min_fn 函数从 OPEN 列表中选择成本最低的节点作为下一个节点。

6. 更新节点状态：如果找到了下一个节点，更新 xNode 和 yNode 为该节点的坐标，更新路径成本，并将该节点移动到 CLOSED 列表。如果没有找到下一个

节点，设置 NoPath 为 0，表示没有路径可找。

7. 结束循环：当前节点是目标节点或者 NoPath 为 0 时，循环结束。

```
i=size(CLOSED,1);
Optimal_path=[];
xval=CLOSED(i,1);
yval=CLOSED(i,2);
i=1;
Optimal_path(i,1)=xval;
Optimal_path(i,2)=yval;
i=i+1;
if ( (xval == xTarget) && (yval == yTarget))
   inode=0;
   parent_x=OPEN(node_index(OPEN,xval,yval),4);
   parent_y=OPEN(node_index(OPEN,xval,yval),5);
   while( parent_x ~= xStart || parent_y ~= yStart)
           Optimal_path(i,1) = parent_x;
           Optimal_path(i,2) = parent_y;
           inode=node_index(OPEN,parent_x,parent_y);
           parent_x=OPEN(inode,4);
           parent_y=OPEN(inode,5);
           i=i+1;
   end
   Optimal_path = [Optimal_path;xStart,yStart];
   path = Optimal_path;
end
```

第二部分补充代码

第二部分代码目的是通过路径回溯，从 CLOSED 列表中逐步构建出最优路径，下面是该部分代码的逻辑架构：

1. 初始化变量：i 设置为 CLOSED 列表的长度，即最后一个节点的索引。Optimal_path 数组用于存储最优路径。xval 和 yval 设置为 CLOSED 列表最后一个节点的坐标，即目标节点。

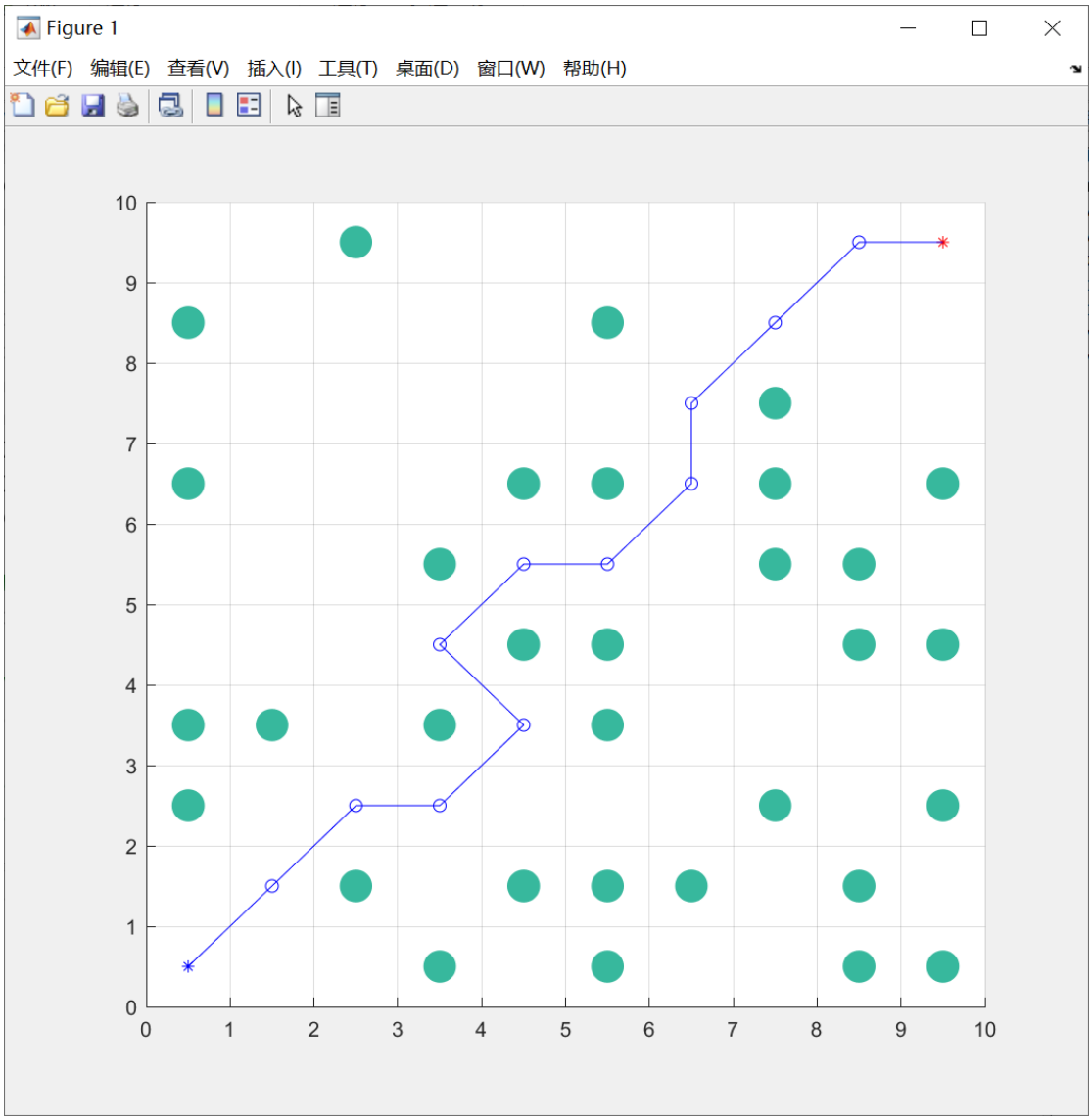
2. 构建最优路径：将目标节点的坐标添加到 Optimal_path 数组的第一行。

3. 检查目标节点：如果当前节点坐标与目标节点坐标相同，则开始回溯过程。

4. 回溯父节点：使用 node_index 函数在 OPEN 列表中找到当前节点的父节点坐标，随后将父节点坐标添加到 Optimal_path 数组，并更新 inode 为父节点在 OPEN 列表中的索引。重复此过程直到父节点坐标为起始节点坐标。

5. 完成最优路径：将起始节点坐标添加到 Optimal_path 数组的末尾，将最优路径存储在 path 变量中，用于在主程序 visualize_map 函数进行绘图时调用。

二、运行结果与感受



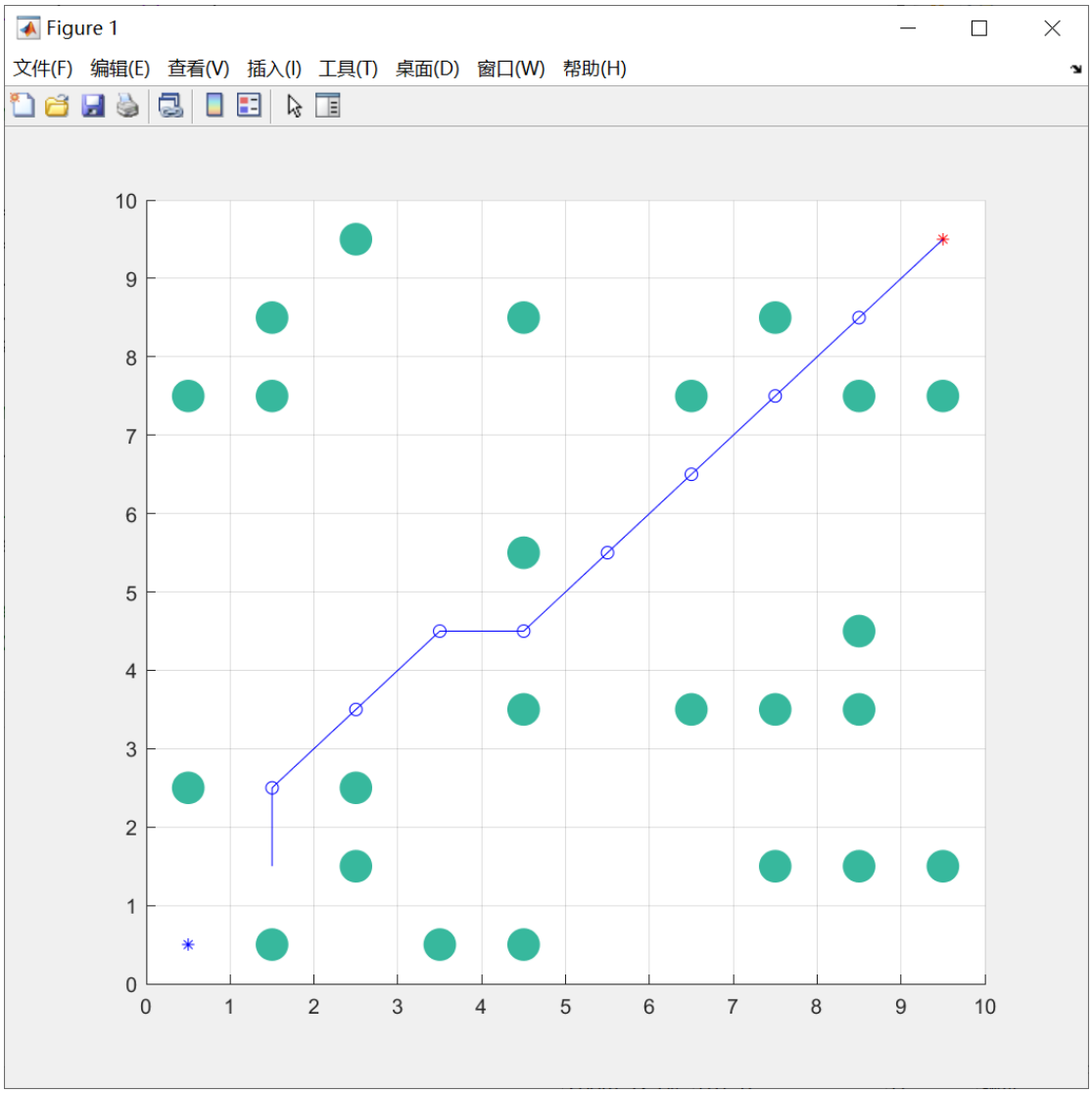
MATLAB 版本作业运行结果

代码完成后运行多次，结果均正确，成功通过 A*算法实现了最优路径规划，本次实验也加深了我对 A*算法的认识与理解，锻炼了我的 MATLAB 代码能力与算法逻辑能力，让我对机器人路径规划课程知识有了新的感悟。

三、问题与解决方案

问题：在初步完成作业代码补充之后，运行主程序，发现路径无法与起始点相连，如下图所示。

解决方案：在多次梳理代码结构，尝试改动代码之后发现，问题在于补充的算法代码在进行回溯时，最后没有将起始点包含在内，即起始点没有被添加到 Optimal_path 数组中，因此在绘图时调用的 path 缺少了起始点坐标。随后在 A* 算法最后添加了代码 `Optimal_path = [Optimal_path;xStart,yStart];` 成功解决了问题。



路径无法与起始点相连