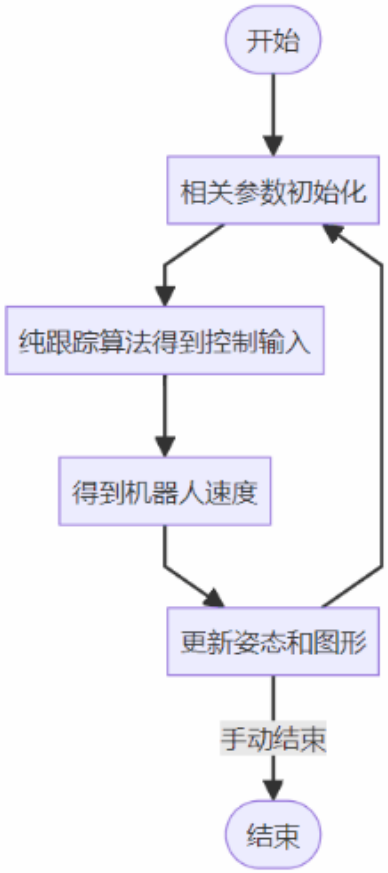
仅供参考，切勿抄袭。

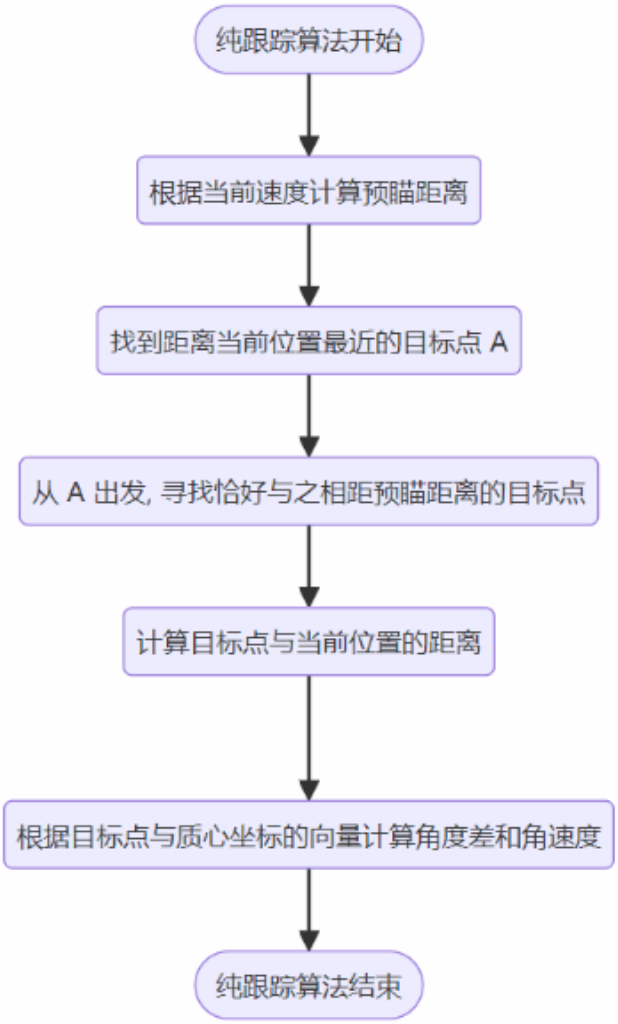
2024 实验报告要求：包含实验一二源代码（写好标注和注释）及效果、实验三四生成的地图、遇到的问题及解决方案（实验效果当堂验收无需加入）。

eai_map_imu.pgm 是生成的地图文件；PGM2png 是将生成的 pgm 格式地图转为 png 格式文件的 python 脚本，方便插入 Word 文档中。流程图是用 markdown + mermaid 绘制（见 流程图.txt，请自行修改扩展名为 md）。

一、 实验流程框图

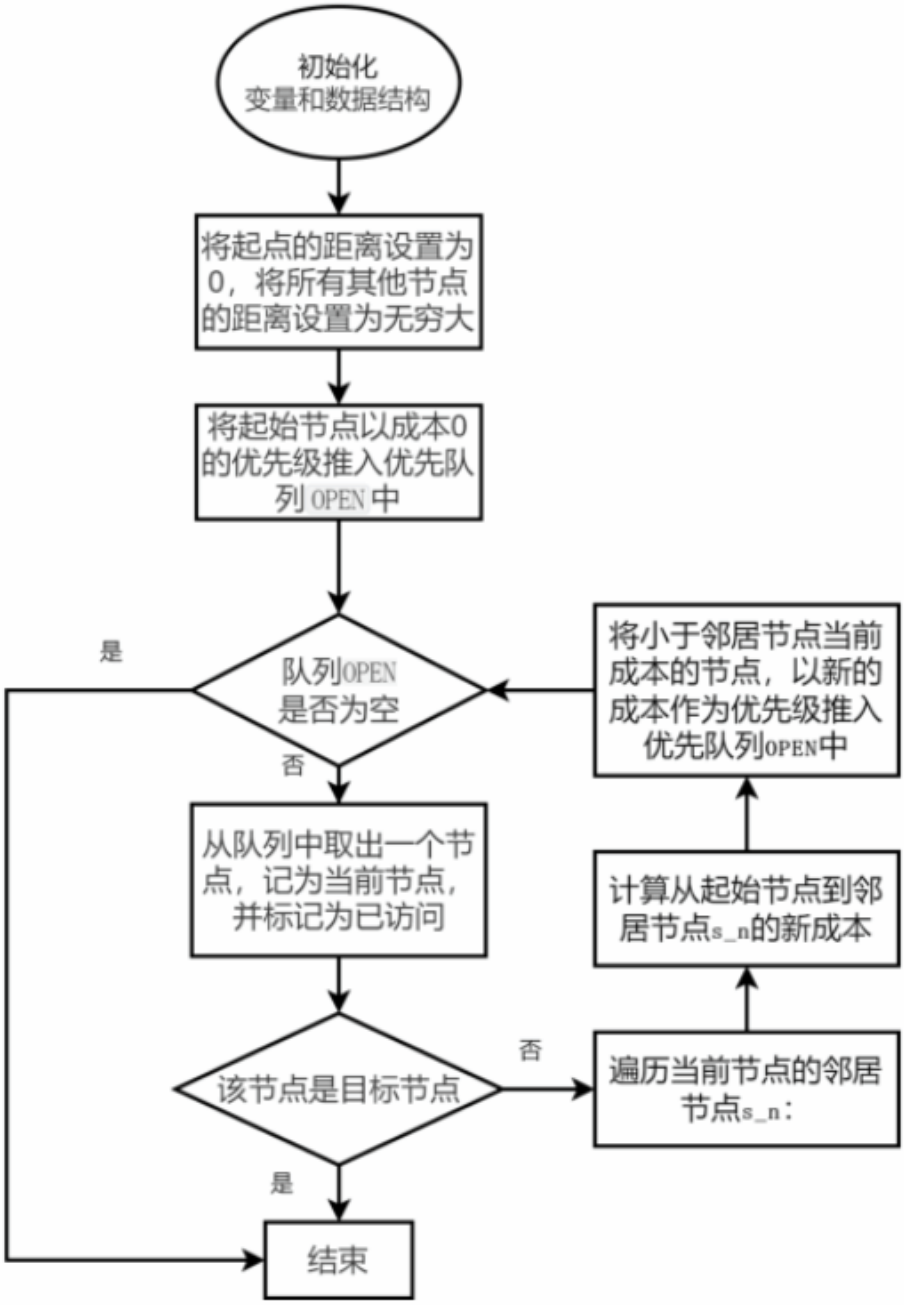
实验一：


开始
相关参数初始化
纯跟踪算法得到控制输入
得到机器人速度
更新姿态和图形
手动结束
结束

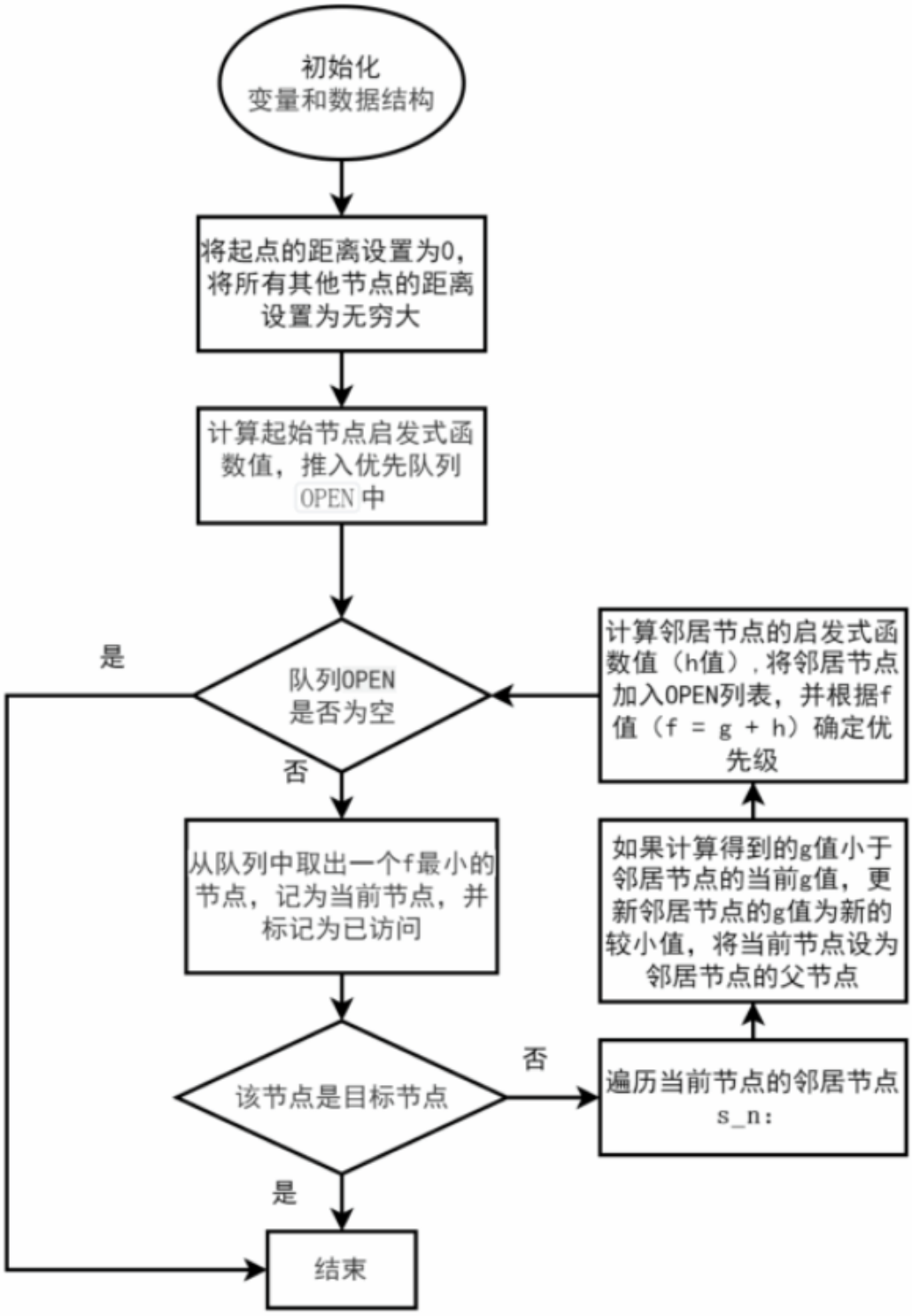

纯跟踪算法开始
根据当前速度计算预瞄距离
找到距离当前位置最近的目标点 A
从 A 出发，寻找恰好与之相距预瞄距离的目标点
计算目标点与当前位置的距离
根据目标点与质心坐标的向量计算角度差和角速度
纯跟踪算法结束

实验二：

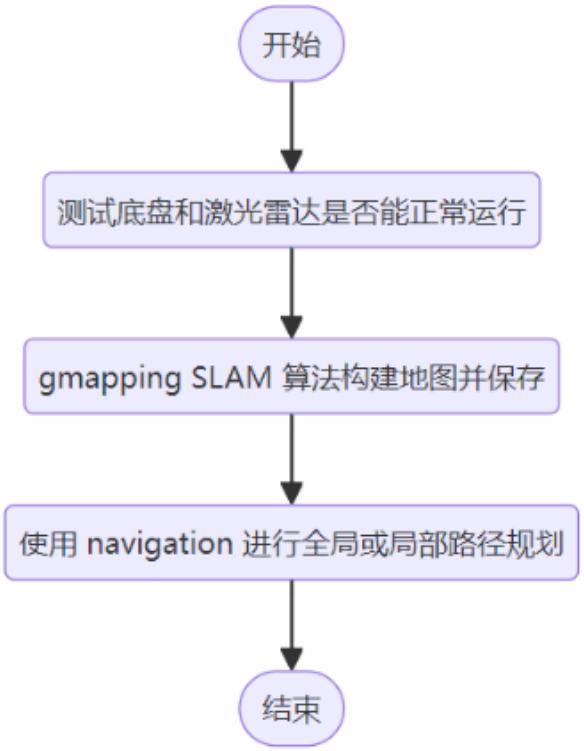
Dijkstra 算法：



A*算法：


初始化
变量和数据结构
将起点的距离设置为0，将所有其他节点的距离设置为无穷大
计算起始节点启发式函数值，推入优先队列OPEN中
队列OPEN是否为空
是
否
从队列中取出一个f最小的节点，记为当前节点，并标记为已访问
该节点是目标节点
否
是
结束
遍历当前节点的邻居节点s_n：
如果计算得到的g值小于邻居节点的当前g值，更新邻居节点的g值为新的较小值，将当前节点设为邻居节点的父节点
计算邻居节点的启发式函数值（h值），将邻居节点加入OPEN列表，并根据f值（f = g + h）确定优先级

实验三、四：



二、实验程序代码

实验一：

（仅列出pure_pursuit_control函数的源码）

```
function [v,w] = pure_pursuit_control(robotCurrentPose, path, v, vr)
    K = 0.1;
    Kp = 0.1;
    Con = 0.2;
    l = K*v+Con;  % 预瞄距离

    mindis = 1000000;
    min_id = 1;
    Point_temp = zeros([1,2]);

    for i=1:length(path) % 找到距离当前位置最近的目标点
        dis = (path(i,1) - robotCurrentPose(1))^2 + (path(i,2) -
robotCurrentPose(2))^2;
        if dis < mindis
            mindis = dis;
            min_id = i;
        end
    end
```

```
    for i=min_id:length(path) % 从最近的目标点出发，寻找恰好与之相差预瞄距离的目标点（比预瞄距离略远）
        dis = (path(i,1) - robotCurrentPose(1))^2 + (path(i,2) - robotCurrentPose(2))^2;
        if dis > l^2
            Point_lookahead = path(i,:);
            disp(i)
            break;
        end
    end

    delta = vr-v;
    v = v + Kp*delta; % P 控制器

    pos = [robotCurrentPose(1),robotCurrentPose(2)];
    Lf = norm(Point_lookahead - pos);  % 计算目标点与当前位置的距离
    theta = atan2((Point_lookahead(2) - robotCurrentPose(2)),(Point_lookahead(1) - robotCurrentPose(1)));   % 目标点与质心坐标的向量与 x 轴的夹角
    eta = robotCurrentPose(3) - theta;  % 计算机器人坐标系下的角度差异
    w = -(2*v*sin(eta))/Lf;  % 角速度控制,注意方向（逆正顺负）
end
```

实验二：

（DijkstraGrid.m，仅列出从主循环前一行到函数结尾的代码[其余部分未作改动]）

```
function [route,numExpanded,I,J] = DijkstraGrid(input_map, start_coords, dest_coords)
% 前半部分未作改动，略去
% 记录已扩展的节点数
numExpanded = 0;
% 第一次标志，因为第一次要访问的起始节点被标记为 5，不在标记为 4 的 OPEN 队列里
once = true;
% Main Loop
while true

    % 显示起点和终点
    map(start_node) = 5;
    map(dest_node) = 6;

    if (drawMapEveryTime)
        map1 = imrotate(map,90);
```

```
        image(0.5, 0.5, map1);
        axis image;  % 设置坐标轴的宽高比为 1，使地图绘制时不发生形变
        drawnow;  % 立即更新图形窗口
    end

    min_dist = Inf;
    current = 0;
    % 在 OPEN 队列里寻找距离最小的点，并返回最小值 min_dist 和对应的索引 current
    for i=1:size(distanceFromStart,1)
        for j=1:size(distanceFromStart,2)
            temp_point = sub2ind(size(map), i, j);
            if map(temp_point) == 4 || once
                dis = distanceFromStart(temp_point);
                if dis < min_dist
                    min_dist = dis;
                    current = temp_point;
                end
            end
        end
    end

    if once == true
        once = false;
    end
%     [min_dist, current] = min(distanceFromStart(:));

    % 判断条件是否满足其中之一，说明已找最短路径或无法到达终点
    % 跳出循环，结束路径搜索
    if ((current == dest_node) || isinf(min_dist))
        break;
    end

    % 更新地图
    map(current) = 3;   % 将当前节点标记为已访问状态

    numExpanded = numExpanded + 1;

    % 计算当前节点的行列坐标
    [i, j] = ind2sub(size(distanceFromStart), current);

   % **********************************************************************
    % 访问当前节点的每个相邻节点
    neighbors = [i-1, j; i+1, j; i, j-1; i, j+1]; % 上下左右四个邻居
    for k = 1:size(neighbors, 1)
```

```
        ni = neighbors(k, 1);
        nj = neighbors(k, 2);
        if ni > 0 && ni <= nrows && nj > 0 && nj <= ncols
            neighbor = sub2ind(size(map), ni, nj);
            if map(neighbor) ~= 2 && map(neighbor) ~= 3  % 不是障碍物或已访问节
点
                tentative_dist = distanceFromStart(current) + 1;  % 距离增加 1
                if tentative_dist < distanceFromStart(neighbor)
                    distanceFromStart(neighbor) = tentative_dist;
                    parent(neighbor) = current;  % 记录父节点
                    map(neighbor) = 4;  % 将邻居标记为正在考虑的节点
                end
            end
        end
    end
    % ************************************************************************
end

%% 构建从起点到终点的路径，并进行可视化
if (isinf(distanceFromStart(dest_node))) % 判断是否存在路径
    route = [];
else
    % ************************************************************************
    % 构建从终点到起点回溯路径
    route = dest_node;
    while route(1) ~= start_node
        route = [parent(route(1)), route]; % 回溯路径
    end

    I = zeros(1,size(route,2));
    J = zeros(1,size(route,2));
    disp(size(route))
    for i=1:size(route,2)
        [I(i),J(i)] = ind2sub(size(map), route(i));  % 提取路径点横纵坐标
    end
    disp(I)
    disp(J)

    % ************************************************************************

    % 用于可视化地图和路径的代码片段
    for k = 2:length(route) - 1
        map(route(k)) = 7;
        pause(0.1);
```

```
        map1 = imrotate(map,90);
        image(0.5, 0.5, map1);
        grid on;
        axis image;
    end
end

end
```

（AStarGrid.m，仅列出从主循环前一行到函数结尾的代码[其余部分未作改动]）

```
function [route,numExpanded,I,J] = AStarGrid(input_map, start_coords,
dest_coords)
% 前半部分未作改动，略去
% 第一次标志，因为第一次要访问的起始节点被标记为 5，不在标记为 4 的 OPEN 队列里
once = true;
% Main Loop
while true

    % 显示起点和终点
    map(start_node) = 5;
    map(dest_node) = 6;

    if (drawMapEveryTime)
        map1=imrotate(map,90);
        image(0.5, 0.5, map1);
        axis image;  % 设置坐标轴的宽高比为 1，使地图绘制时不发生形变
        drawnow;  % 立即更新图形窗口
    end

    min_f = Inf;
    current = 0;
    % 寻找 f 最小的点，并返回最小值 min_f 和对应的索引 current
    for i=1:size(f,1)
        for j=1:size(f,2)
            temp_point = sub2ind(size(map), i, j);
            if map(temp_point) == 4 || once
                dis = f(temp_point);
                if dis < min_f
                    min_f = dis;
                    current = temp_point;
```

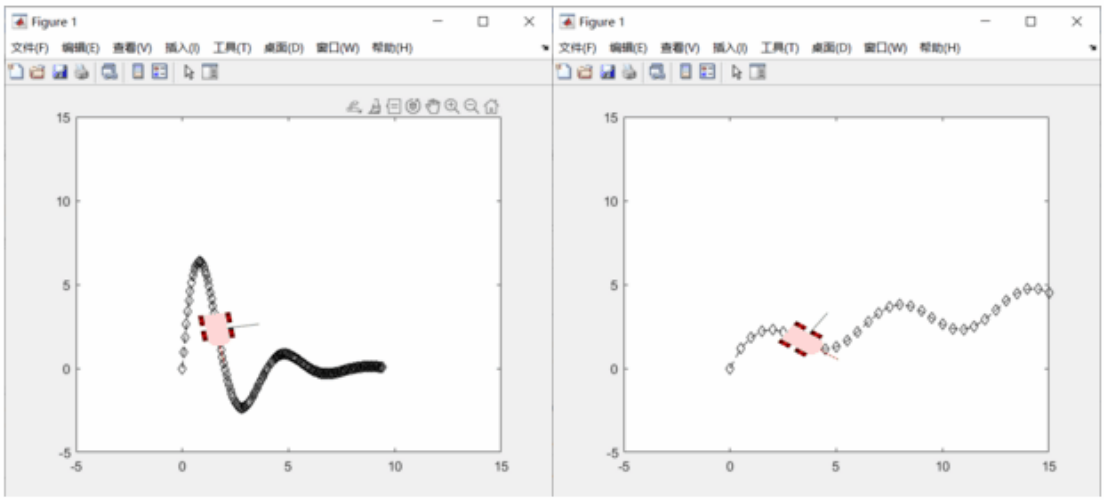
```
                end
            end
        end
    end

    if once == true
        once = false;
    end
%     [min_f, current] = min(f(:));

    % 判断条件是否满足其中之一，说明已找最短路径或无法到达终点
    % 跳出循环，结束路径搜索
    if ((current == dest_node) || isinf(min_f))
        break;
    end

    % 更新地图
    map(current) = 3;  % 将当前节点标记为已访问状态
    numExpanded = numExpanded + 1;

    % 计算当前节点的行列坐标
    [i, j] = ind2sub(size(f), current);

   % ********************************************************************
    % 访问当前节点的每个相邻节点
    % 根据访问结果更新地图、f 和 g 和父节点表
    neighbors = [i-1, j; i+1, j; i, j-1; i, j+1]; % 上下左右四个邻居
    for k = 1:size(neighbors, 1)
        ni = neighbors(k, 1);
        nj = neighbors(k, 2);
        if ni > 0 && ni <= nrows && nj > 0 && nj <= ncols
            neighbor = sub2ind(size(map), ni, nj);
            if map(neighbor) ~= 2 && map(neighbor) ~= 3  % 不是障碍物或已访问节
点
                tentative_g = g(current) + 1;
                if tentative_g < g(neighbor)
                    g(neighbor) = tentative_g;
                    f(neighbor) = g(neighbor) + H(neighbor);
                    parent(neighbor) = current;  % 记录父节点
                    map(neighbor) = 4;  % 将邻居标记为正在考虑的节点
                end
            end
        end
    end
```

```
    %*************************************************************************
end

%% 构建从起点到终点的路径，并进行可视化
if (isinf(f(dest_node))) % 判断是否存在路径
    route = [];
else
    % *************************************************************************
    % 构建从终点到起点回溯路径
    route = dest_node;
    while route(1) ~= start_node
        route = [parent(route(1)), route]; % 回溯路径
    end

    I = zeros(1,size(route,2));
    J = zeros(1,size(route,2));
    disp(size(route))
    for i=1:size(route,2)
        [I(i),J(i)] = ind2sub(size(map), route(i));  % 提取路径点横纵坐标
    end
    disp(I)
    disp(J)
    % *************************************************************************

    % 用于可视化地图和路径的代码片段
    for k = 2:length(route) - 1
        map(route(k)) = 7;
        pause(0.1);
        map1 = imrotate(map,90);
        image(0.5, 0.5, map1);
        grid on;
        axis image;
    end
end
end
```

三、 实验结果及分析

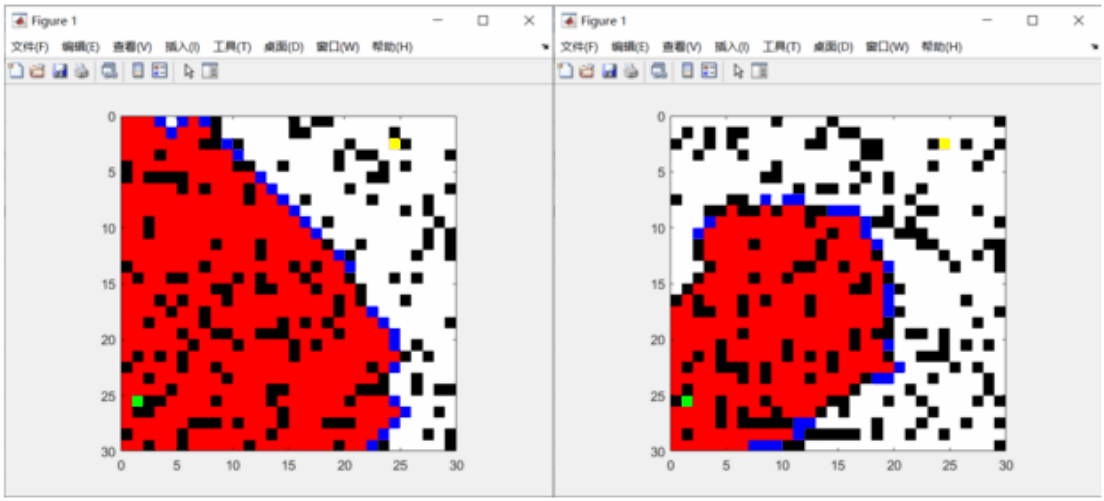
实验一：

图示（下面两张，左图为参考路径 2，右图为参考路径 3）：

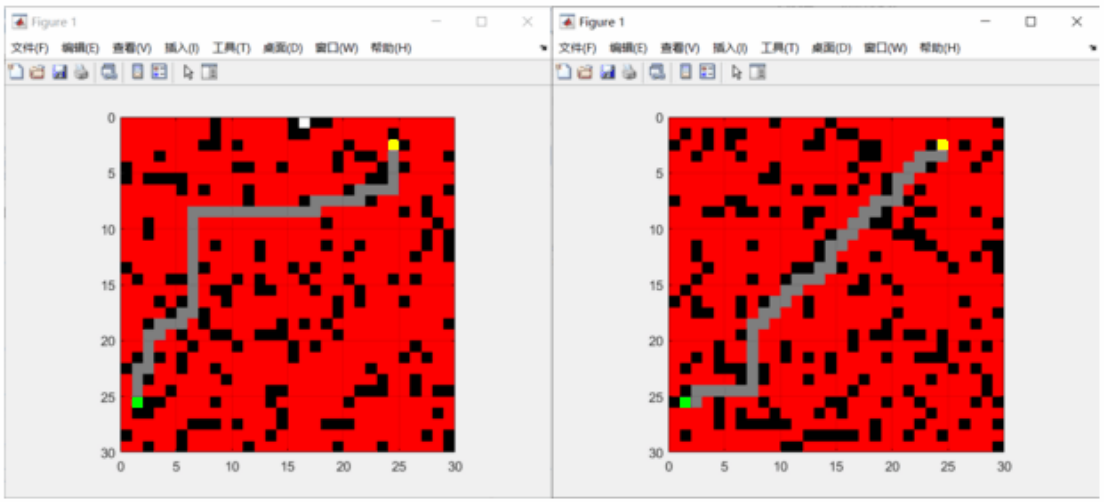


实验二：

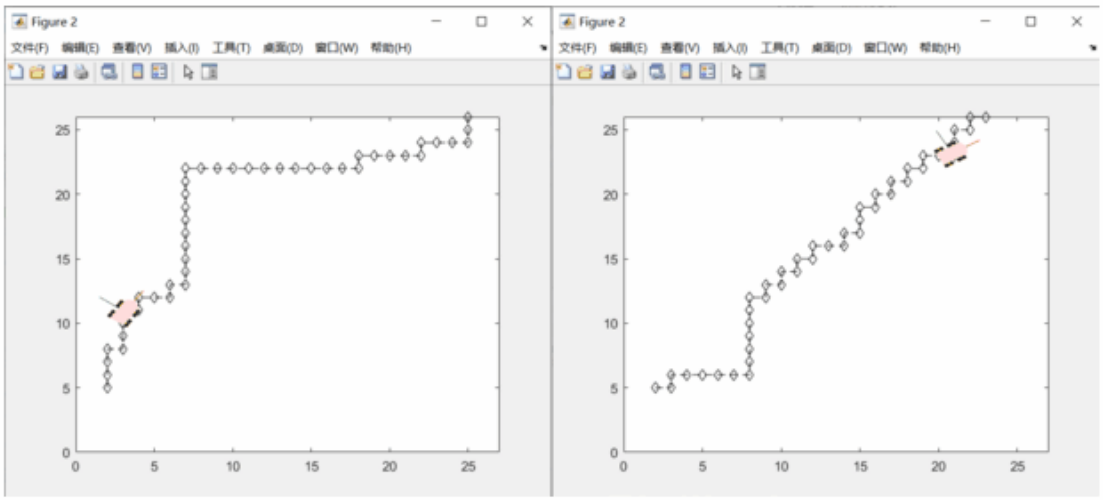
搜索过程图示（下面两张，左图为 Dijkstra 算法，右图为 A* 算法）：



路径回溯图示（下面两张，左图为 Dijkstra 算法，右图为 A* 算法）：



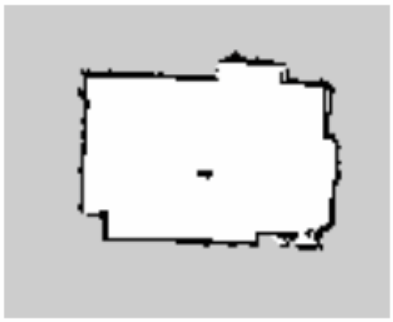
导航图示（下面两张，左图为 Dijkstra 算法，右图为 A* 算法）：



可知用上一部分中展示的代码可以完成路径搜索的任务。

实验三、四：

生成的地图文件（位置为 K321 旁电梯间，中间的障碍物由一位同学充当）：



实验效果已当堂验收，此处略去。

四、 实验结论

仿真中，实现了 Dijkstra 算法和 A*算法的路径搜索，且机器人能跟踪目标轨迹。

实验中，遇到的问题是：

① 遇到了无法识别端口的现象，首先更改 config 文件识别到 STM32，对于 LIDAR，未经特殊解决，仅仅插拔端口即自行好转。

② 室内环境动态变化，所得地图噪声较大，常出现存在路径却因建图误差导致不可达的状况。

最后选择动态因素较少的电梯间完成实验，定位导航效果较好。

关于参数的更改：减小与障碍物最大距离参数后，机器人变得更加激进，导航的路径更加平直，在同样最大速度条件下，到达目标点需时更少。