A*算法

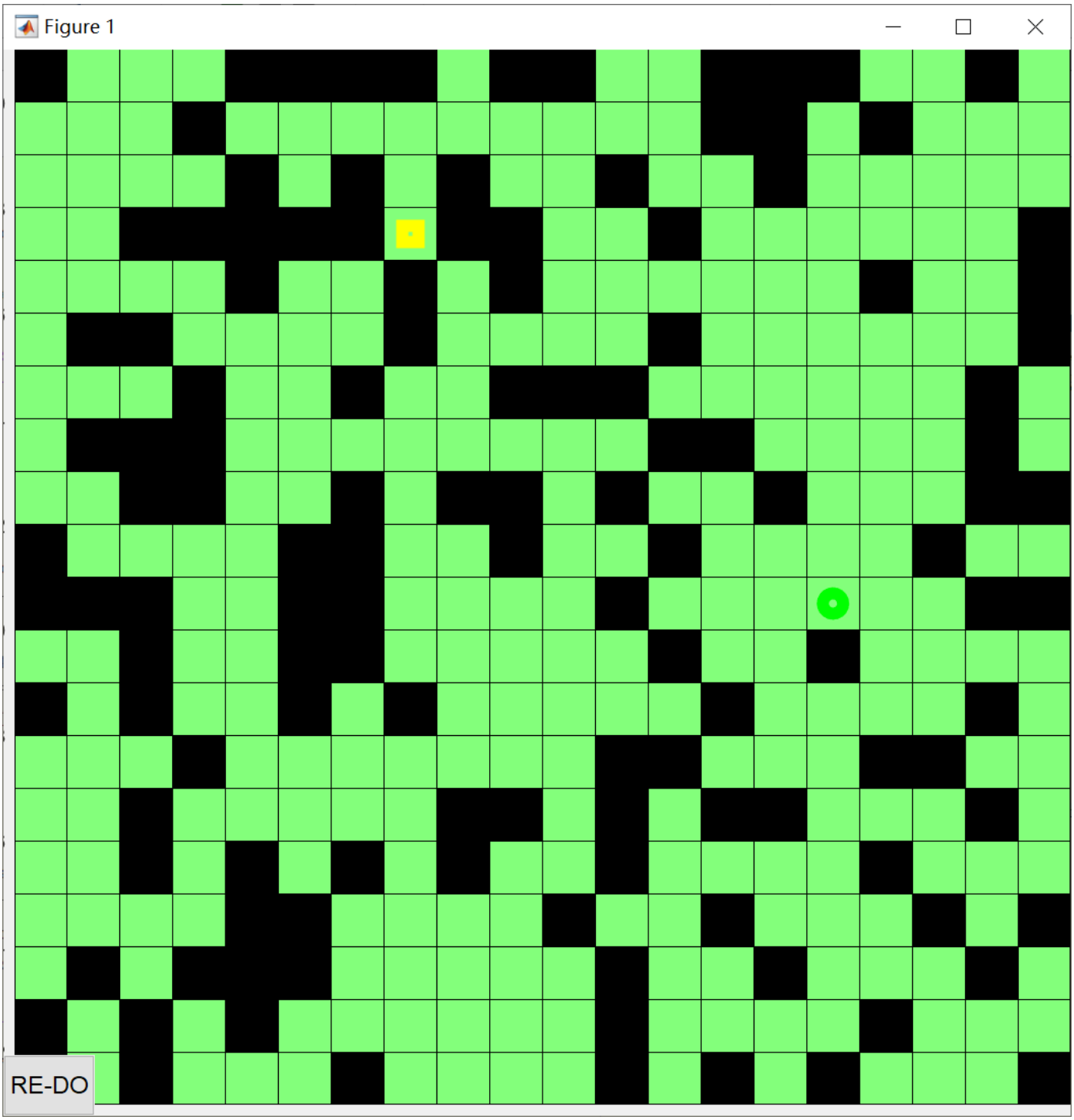
总结：我根据网上的开源A*算法初步跑了一下，了解了A*算法的建立过程。通过随机生成的2D栅格图，实现了动态衡量A*算法，在此基础上进行了拐角优化。 因为对与matlab语法使用还是不太熟悉，对于论文中的改进点还没有完全复现。在未来一周会再次学习matlab使用，复现简单的论文算法。 （参考论文：无人机路径规划与目标检测算法研究）

一、随机栅格图

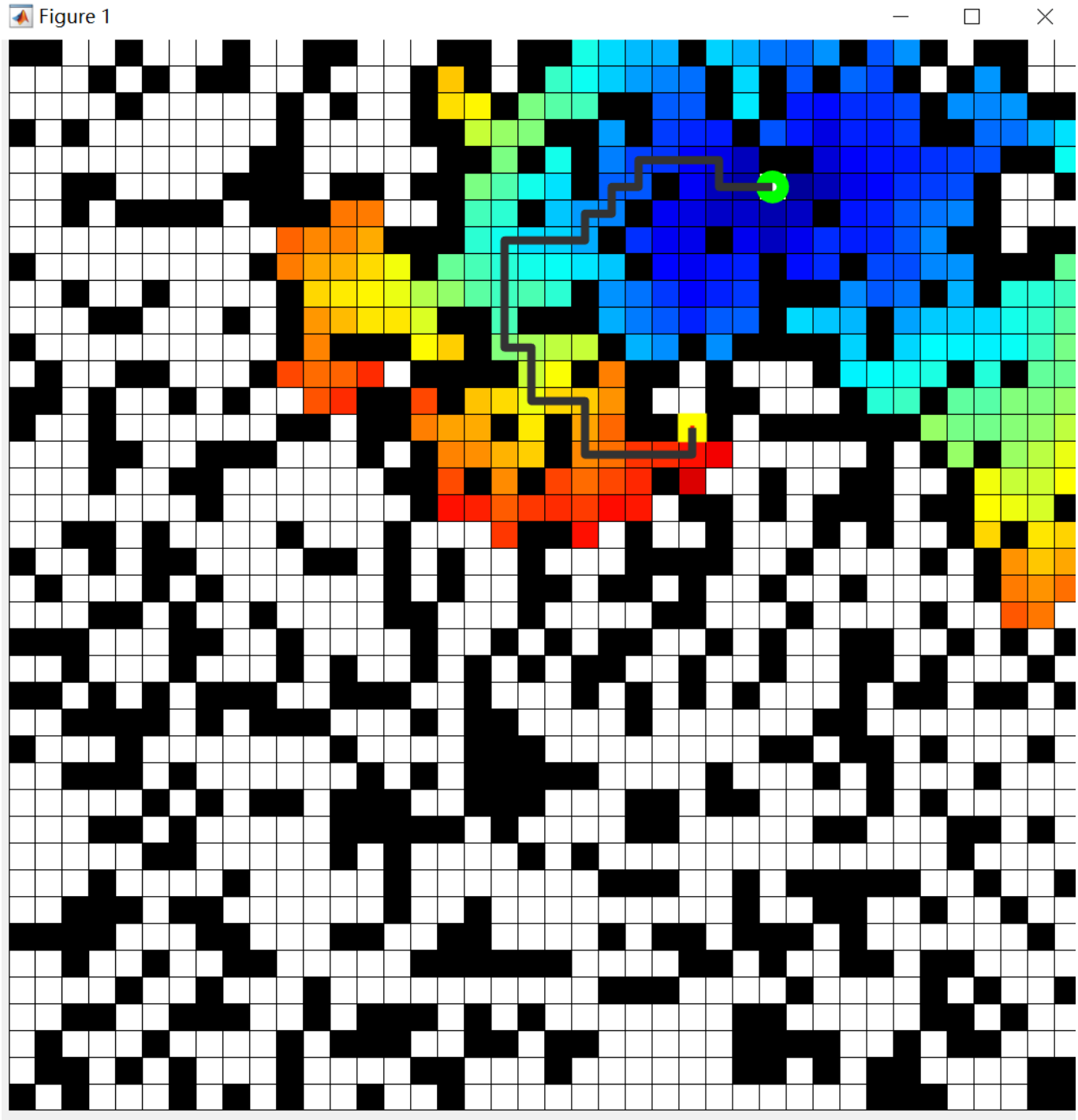
1.随机数建立栅格图

- Inf-有障碍物
- 1-11的随机数-无障碍

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inf	2.1516	Inf	2.8243	10.0563	2.5024	Inf	7.5055	7.5622	9.5869	1.9755	Inf	6.6842	Inf	9.5430	Inf	10.0143	3.4785	4.0371	Inf
Inf	7.4102	Inf	3.7515	Inf	8.4990	10.9492	4.9984	2.9757	3.9987	8.6344	Inf	8.6243	2.1240	1.8418	10.8766	Inf	8.7029	7.7480	2.5869
9.5240	Inf	10.7303	Inf	Inf	Inf	2.0897	5.1535	1.9194	9.2455	4.8226	Inf	6.9641	3.0952	Inf	6.1845	7.5072	6.1564	Inf	3.7729
7.2526	1.4807	2.1428	2.5947	Inf	Inf	9.2878	2.5219	5.5033	6.0087	Inf	10.1107	5.0980	Inf	10.5740	7.5758	4.8354	Inf	10.0818	Inf
8.0880	7.8410	Inf	4.5901	Inf	1.8779	Inf	1.2499	Inf	4.3443	6.6171	Inf	10.1257	7.2859	10.4643	2.2239	Inf	5.8012	9.7900	7.6489
2.1456	5.2518	Inf	4.3144	2.3514	7.6309	9.3452	1.8219	Inf	Inf	2.1424	Inf	2.6318	Inf	Inf	7.2662	3.2169	4.3467	Inf	2.6025
10.2393	8.6757	10.7349	Inf	5.1602	4.8803	2.6380	2.5621	7.2113	3.6277	1.9077	Inf	Inf	2.7192	3.0010	2.0718	Inf	Inf	7.8739	1.2860
Inf	4.6151	Inf	2.5473	8.1104	Inf	2.8816	Inf	3.9427	7.4976	6.4769	5.2758	7.0570	Inf	4.4435	1.3717	10.0042	7.1386	Inf	10.2746
7.6264	2.9863	Inf	6.3013	8.1057	Inf	Inf	2.6761	4.3009	1.7881	1.9500	2.2928	Inf	6.5072	8.8136	Inf	5.8911	7.2009	9.4374	8.4447
Inf	Inf	Inf	4.8319	7.0941	Inf	Inf	8.2780	1.0005	9.1102	2.4760	Inf	4.8264	10.3353	9.6536	0	4.5921	7.1089	Inf	Inf
Inf	2.0582	8.1803	10.4103	8.5956	Inf	Inf	3.2062	1.8662	Inf	9.9559	8.6981	Inf	6.9004	10.5861	8.8893	3.2137	Inf	7.6908	4.5963
10.9550	6.7434	Inf	Inf	6.8446	9.7894	Inf	3.5629	Inf	Inf	3.6138	Inf	3.6069	10.4002	Inf	2.3587	3.2703	7.1408	Inf	Inf
2.6289	Inf	Inf	Inf	4.9183	10.0417	10.3393	5.9780	2.8661	7.5059	9.5191	7.1790	Inf	Inf	9.2075	8.8231	3.5044	8.4309	Inf	5.0386
10.1330	8.2123	3.3214	Inf	10.9392	8.8536	Inf	1.0323	7.9683	Inf	Inf	Inf	6.8359	4.7144	8.5746	5.1900	6.1720	7.5920	Inf	7.4857
7.0731	Inf	Inf	4.8103	7.0634	4.0958	2.7124	Inf	7.7238	10.7776	6.3293	6.9351	Inf	5.4491	4.4772	4.1582	2.7856	10.6248	10.4652	Inf
3.2275	5.5508	8.1654	7.6939	Inf	1.1640	6.7556	Inf	4.8462	Inf	8.2597	9.7281	5.3308	3.2509	5.8036	4.9194	Inf	6.0570	1.7535	Inf
7.9158	8.8329	Inf	Inf	Inf	Inf	Inf	0	Inf	Inf	1.0651	10.1737	Inf	5.9305	10.6448	10.0029	2.5214	2.6087	4.3814	Inf
8.0301	4.3768	5.0685	5.1663	Inf	3.5623	Inf	3.0619	Inf	9.1523	8.8243	Inf	7.7524	9.2556	Inf	7.6390	5.1834	5.1533	6.1721	8.2770
10.4956	2.5218	9.0957	Inf	3.8283	5.7441	3.3326	10.6540	3.1068	8.3309	5.1417	6.8865	8.6014	Inf	Inf	9.1055	Inf	5.1111	10.9900	6.2145
Inf	7.2394	1.6132	6.0860	Inf	Inf	Inf	Inf	8.8801	Inf	Inf	5.1396	6.1735	Inf	Inf	Inf	3.6507	2.3222	Inf	6.1672

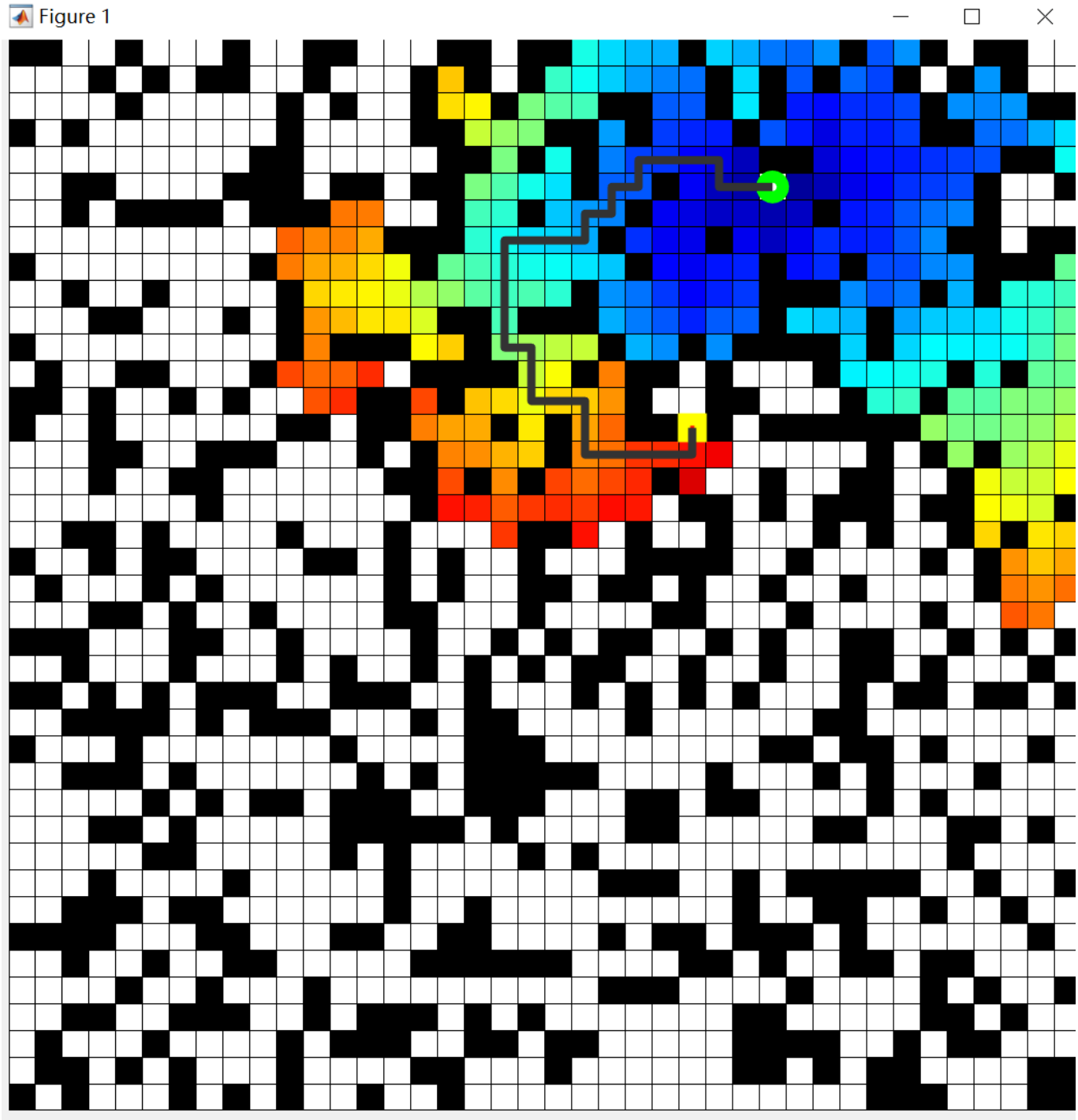


二、A*算法实验



三、拐角优化

考虑到无人机在实际飞行中的安全性，我更倾向于规划出一条转弯次数最少的路径。 为了实现这一目标，需要对路径进行进一步的优化，减少不必要的转弯，用更平滑的曲线代替多个急转弯。这样的路径不仅能够提高无人机的飞行效率，还能减少能源消耗，并提高飞行的稳定性和安全性。




```
clc;               %清除命令窗口的内容
clear all;         %清除工作空间的所有变量，函数，和MEX文件
close all;         %关闭所有的figure窗口


%功能参数的设定部分
n = 30;    % 产生一个n x n的方格，修改此值可以修改生成图片的方格数
wallpercent = 0.4;  % 这个变量代表生成的障碍物占总方格数的比例 ，如0.5 表示障碍物占总格
数的50%
Weights=1;         %动态衡量启发式A星算法中的h（n）权重系数
Corner_amend=1;   %选择是否进行拐角的修正，该变量设为0则不进行拐角修正，设为1则进行拐角修
正
```

```
%方格以及障碍物的创建
[field, startposind, goalposind, costchart, fieldpointers]
=initializeField(n,wallpercent); %随机生成包含障碍物，起始点，终止点等信息的矩阵

% 路径规划中用到的一些矩阵的初始化
setOpen = [startposind]; setOpenCosts = [0]; setOpenHeuristics = [Inf];
setClosed = []; setClosedCosts = [];
movementdirections = {'R','L','D','U'};  %移动方向

%初始化一些进行路径的修正需要用到的变量
Parent_node=0; %Parent_node初始化，否则会报错
Expected_note=0;%Expected_note初始化，否则会报错
untext_ii=0;  %未经过检验的新的ii
amend_count=0;% 记录修正的次数


% 这个函数用来随机生成环境，障碍物，起点，终点
axishandle = createFigure(field,costchart,startposind,goalposind);     %将随机生成的
方格及障碍物的数据生成图像

%%

% 这个while循环是本程序的核心，利用循环进行迭代来寻找终止点
while ~max(ismember(setOpen,goalposind)) && ~isempty(setOpen)

    [temp, ii] = min(setOpenCosts +Weights*setOpenHeuristics);      %寻找拓展出来的
最小值

    if ((setOpen(ii)~=startposind) && (Corner_amend==1))
    [new_ii,amend_count_1]=Path_optimization(temp,
ii,fieldpointers,setOpen,setOpenCosts,startposind,Weights,setOpenHeuristics,Parent
_node,Expected_note,untext_ii,amend_count); %进行路径的修正，在保证不增加距离的基础
上，使其减少转弯的次数
    ii=new_ii;
    amend_count=amend_count_1;
    end

    %这个函数的作用就是把输入的点作为父节点，然后进行拓展找到子节点，并且找到子节点的代
价，并且把子节点距离终点的代价找到
    [costs,heuristics,posinds] = findFValue(setOpen(ii),setOpenCosts(ii),
field,goalposind,'euclidean');

  setClosed = [setClosed; setOpen(ii)];      % 将找出来的拓展出来的点中代价最小的那个
点串到矩阵setClosed 中
  setClosedCosts = [setClosedCosts; setOpenCosts(ii)];     % 将拓展出来的点中代价最小
的那个点的代价串到矩阵setClosedCosts 中

  % 从setOpen中删除刚才放到矩阵setClosed中的那个点
  %如果这个点位于矩阵的内部
  if (ii > 1 && ii < length(setOpen))
    setOpen = [setOpen(1:ii-1); setOpen(ii+1:end)];
    setOpenCosts = [setOpenCosts(1:ii-1); setOpenCosts(ii+1:end)];
    setOpenHeuristics = [setOpenHeuristics(1:ii-1); setOpenHeuristics(ii+1:end)];
```

```
    %如果这个点位于矩阵第一行
    elseif (ii == 1)
      setOpen = setOpen(2:end);
      setOpenCosts = setOpenCosts(2:end);
      setOpenHeuristics = setOpenHeuristics(2:end);

    %如果这个点位于矩阵的最后一行
    else
      setOpen = setOpen(1:end-1);
      setOpenCosts = setOpenCosts(1:end-1);
      setOpenHeuristics = setOpenHeuristics(1:end-1);
    end

  %%
    % 把拓展出来的点中符合要求的点放到setOpen 矩阵中，作为待选点
    for jj=1:length(posinds)

      if ~isinf(costs(jj))    % 判断该点（方格）处没有障碍物

        % 判断一下该点是否 已经存在于setOpen 矩阵或者setClosed 矩阵中
        % 如果我们要处理的拓展点既不在setOpen 矩阵，也不在setClosed 矩阵中
        if ~max([setClosed; setOpen] == posinds(jj))
          fieldpointers(posinds(jj)) = movementdirections(jj);
          costchart(posinds(jj)) = costs(jj);
          setOpen = [setOpen; posinds(jj)];
          setOpenCosts = [setOpenCosts; costs(jj)];
          setOpenHeuristics = [setOpenHeuristics; heuristics(jj)];

        % 如果我们要处理的拓展点已经在setOpen 矩阵中
        elseif max(setOpen == posinds(jj))
          I = find(setOpen == posinds(jj));
          % 如果通过目前的方法找到的这个点，比之前的方法好（代价小）就更新这个点
          if setOpenCosts(I) > costs(jj)
            costchart(setOpen(I)) = costs(jj);
            setOpenCosts(I) = costs(jj);
            setOpenHeuristics(I) = heuristics(jj);
            fieldpointers(setOpen(I)) = movementdirections(jj);
          end

          % 如果我们要处理的拓展点已经在setClosed 矩阵中
        else
          I = find(setClosed == posinds(jj));
          % 如果通过目前的方法找到的这个点，比之前的方法好（代价小）就更新这个点
          if setClosedCosts(I) > costs(jj)
            costchart(setClosed(I)) = costs(jj);
            setClosedCosts(I) = costs(jj);
            fieldpointers(setClosed(I)) = movementdirections(jj);
          end
        end
      end
    end

  %%
```

```
  if isempty(setOpen) break; end
  set(axishandle,'CData',[costchart costchart(:,end); costchart(end,:)
costchart(end,end)]);
  set(gca,'CLim',[0 1.1*max(costchart(find(costchart < Inf)))]);
  drawnow;
end

%%

%调用findWayBack函数进行路径回溯，并绘制出路径曲线
if max(ismember(setOpen,goalposind))
  disp('Solution found!');
  p = findWayBack(goalposind,fieldpointers); % 调用findWayBack函数进行路径回溯，将回
溯结果放于矩阵P中
  plot(p(:,2)+0.5,p(:,1)+0.5,'Color',0.2*ones(3,1),'LineWidth',4);  %用 plot函数绘
制路径曲线
  drawnow;
  drawnow;
  clear sound
  [y,Fs] = audioread('002.wav'); sound(y,Fs); % 播放名为000的音乐，注意该文件需要跟
matlab文件位于同一文件夹下
elseif isempty(setOpen)
  disp('No Solution!');
  clear sound
  [y,Fs] = audioread('000.wav');
  sound(y,Fs);
end

%%

%findWayBack函数用来进行路径回溯，这个函数的输入参数是终止点goalposind和矩阵
fieldpointers，输出参数是P
function p = findWayBack(goalposind,fieldpointers)

    n = length(fieldpointers);  % 获取环境的长度也就是n
    posind = goalposind;
    [py,px] = ind2sub([n,n],posind); % 将索引值posind转换为坐标值 [py,px]
    p = [py px];

    %利用while循环进行回溯，当我们回溯到起始点的时候停止，也就是在矩阵fieldpointers中找
到S时停止
    while ~strcmp(fieldpointers{posind},'S')
      switch fieldpointers{posind}

        case 'L' % ’L’ 表示当前的点是由左边的点拓展出来的
          px = px - 1;
        case 'R' % ’R’ 表示当前的点是由右边的点拓展出来的
          px = px + 1;
        case 'U' % ’U’ 表示当前的点是由上面的点拓展出来的
          py = py - 1;
        case 'D' % ’D’ 表示当前的点是由下边的点拓展出来的
          py = py + 1;
      end
      p = [p; py px];
```

```
      posind = sub2ind([n n],py,px);% 将坐标值转换为索引值
    end
end

%%
%这个函数的作用就是把输入的点作为父节点，然后进行拓展找到子节点，并且找到子节点的代价，并且把子节点距离终点的代价找到。
%函数的输出量中costs表示拓展的子节点到起始点的代价，heuristics表示拓展出来的点到终止点的距离大约是多少，posinds表示拓展出来的子节点
function [cost,heuristic,posinds] =
findFValue(posind,costsofar,field,goalind,heuristicmethod)
    n = length(field);  % 获取矩阵的长度
    [currentpos(1) currentpos(2)] = ind2sub([n n],posind);    %将要进行拓展的点（也就是父节点）的索引值拓展成坐标值
    [goalpos(1) goalpos(2)] = ind2sub([n n],goalind);         %将终止点的索引值拓展成坐标值
    cost = Inf*ones(4,1); heuristic = Inf*ones(4,1); pos = ones(4,2); %将矩阵cost和heuristic初始化为4x1的无穷大值的矩阵，pos初始化为4x2的值为1的矩阵

    % 拓展方向一
    newx = currentpos(2) - 1; newy = currentpos(1);
    if newx > 0
      pos(1,:) = [newy newx];
      switch lower(heuristicmethod)
        case 'euclidean'
          heuristic(1) = 10*abs(goalpos(2)-newx) + 10*abs(goalpos(1)-newy);
        case 'taxicab'
          heuristic(1) = 10*abs(goalpos(2)-newx) +10*abs(goalpos(1)-newy);
      end
      cost(1) = costsofar + field(newy,newx);
    end

    % 拓展方向二
    newx = currentpos(2) + 1; newy = currentpos(1);
    if newx <= n
      pos(2,:) = [newy newx];
      switch lower(heuristicmethod)
        case 'euclidean'
          heuristic(2) = 10*abs(goalpos(2)-newx) +10*abs(goalpos(1)-newy);
        case 'taxicab'
          heuristic(2) = 10*abs(goalpos(2)-newx) + 10*abs(goalpos(1)-newy);
      end
      cost(2) = costsofar + field(newy,newx);
    end

    % 拓展方向三
    newx = currentpos(2); newy = currentpos(1)-1;
    if newy > 0
      pos(3,:) = [newy newx];
      switch lower(heuristicmethod)
        case 'euclidean'
          heuristic(3) = 10*abs(goalpos(2)-newx) + 10*abs(goalpos(1)-newy);
        case 'taxicab'
          heuristic(3) = 10*abs(goalpos(2)-newx) + 10*abs(goalpos(1)-newy);
```

```
      end
      cost(3) = costsofar + field(newy,newx);
    end

    % 拓展方向四
    newx = currentpos(2); newy = currentpos(1)+1;
    if newy <= n
      pos(4,:) = [newy newx];
      switch lower(heuristicmethod)
        case 'euclidean'
          heuristic(4) = 10*abs(goalpos(2)-newx) + 10*abs(goalpos(1)-newy);
        case 'taxicab'
          heuristic(4) = 10*abs(goalpos(2)-newx) + 10*abs(goalpos(1)-newy);
      end
      cost(4) = costsofar + field(newy,newx);
    end
     posinds = sub2ind([n n],pos(:,1),pos(:,2)); % 将拓展出来的子节点的坐标值转换为索
引值
end

%%
%这个矩阵的作用就是随机生成环境，障碍物，起始点，终止点等
function [field, startposind, goalposind, costchart, fieldpointers] =
initializeField(n,wallpercent)
    field = 10*ones(n,n);%设置任意两方格间的距离为10
    field(ind2sub([n n],ceil(n^2.*rand(n*n*wallpercent,1)))) = Inf;%向上取整
    % 随机生成起始点和终止点
    startposind = sub2ind([n,n],ceil(n.*rand),ceil(n.*rand));  %随机生成起始点的索
引值
    goalposind = sub2ind([n,n],ceil(n.*rand),ceil(n.*rand));   %随机生成终止点的索
引值
    field(startposind) = 0; field(goalposind) = 0;  %把矩阵中起始点和终止点处的值设为
0

    costchart = NaN*ones(n,n);%生成一个nxn的矩阵costchart，每个元素都设为NaN。就是矩
阵初始NaN无效数据
    costchart(startposind) = 0;%在矩阵costchart中将起始点位置处的值设为0

    % 生成元胞数组
    fieldpointers = cell(n,n);%生成元胞数组n*n
    fieldpointers(:)= {'1'};
    fieldpointers{startposind} = 'S'; fieldpointers{goalposind} = 'G'; %将元胞数组
的起始点的位置处设为 'S'，终止点处设为'G'
    fieldpointers(field==inf)={'0'};


end
% end of this function

%%
%利用随机生成的环境数据来进行环境的绘制
function axishandle = createFigure(field,costchart,startposind,goalposind)

      % 这个if..else结构的作用是判断如果没有打开的figure图，则按照相关设置创建一个
```

```
figure图
      if isempty(gcbf)                                          %gcbf是当前返回图像的句柄，isempty(gcbf)假如gcbf为空的话，返回的值是1，假如gcbf为非空的话，返回的值是0
      figure('Position',[560 70 700 700], 'MenuBar','none');  %对创建的figure图像进行设置，设置其距离屏幕左侧的距离为450，距离屏幕下方的距离为50，长度和宽度都为700，并且关闭图像的菜单栏
      axes('position', [0.01 0.01 0.99 0.99]);                  %设置坐标轴的位置，左下角的坐标设为0.01,0.01   右上角的坐标设为0.99 0.99   (可以认为figure图的左下角坐标为0 0，右上角坐标为1 1 )
      else
      gcf; cla;   %gcf 返回当前 Figure 对象的句柄值，然后利用cla语句来清除它
      end

      n = length(field);  %获取矩阵的长度，并赋值给变量n
      field(field < Inf) = 0; %将fieid矩阵中的随机数（也就是没有障碍物的位置处）设为0
      pcolor(1:n+1,1:n+1,[field field(:,end); field(end,:) field(end,end)]);%多加了一个重复的（由n X n变为 n+1 X n+1 ）

      cmap = flipud(colormap('jet'));  %生成的cmap是一个256X3的矩阵，每一行的3个值都为0-1之间数，分别代表颜色组成的rgb值
      cmap(1,:) = zeros(3,1); cmap(end,:) = ones(3,1); %将矩阵cmap的第一行设为0 ，最后一行设为1
      colormap(flipud(cmap)); %进行颜色的倒转
      hold on;
      axishandle = pcolor([1:n+1],[1:n+1],[costchart costchart(:,end); costchart(end,:) costchart(end,end)]);  %将矩阵costchart进行拓展，插值着色后赋给axishandle
      [goalposy,goalposx] = ind2sub([n,n],goalposind);
      [startposy,startposx] = ind2sub([n,n],startposind);
       plot(goalposx+0.5,goalposy+0.5,'ys','MarkerSize',10,'LineWidth',6);
       plot(startposx+0.5,startposy+0.5,'go','MarkerSize',10,'LineWidth',6);
       %uicontrol('Style','pushbutton','String','RE-DO', 'FontSize',12, 'Position', [1 1 60 40], 'Callback','astardemo');
end
%%
function [new_ii,amend_count_1] = Path_optimization(temp, ii,fieldpointers,setOpen,setOpenCosts,startposind,Weights,setOpenHeuristics,Parent_node,Expected_note,untext_ii,amend_count)
   n = length(fieldpointers);  %获取矩阵的长度，并赋值给变量n

 %获取其父节点的索引值
  switch fieldpointers {setOpen(ii)}
        case 'L' % ’L’ 表示当前的点是由左边的点拓展出来的
          Parent_node = setOpen(ii) - n;
        case 'R' % ’R’ 表示当前的点是由右边的点拓展出来的
           Parent_node = setOpen(ii) + n;
        case 'U' % ’U’ 表示当前的点是由上面的点拓展出来的
           Parent_node = setOpen(ii) -1;
        case 'D' % ’D’ 表示当前的点是由下边的点拓展出来的
           Parent_node = setOpen(ii) + 1;
  end

  if Parent_node==startposind  %如果这个点的父节点是起始点的话，跳过修正
     new_ii=ii;
```

```
    amend_count_1=amend_count;
  else

      %获取期望下一步要走的点的索引值
    switch fieldpointers{Parent_node}

        case 'L' % ’L’ 表示当前的点是由左边的点拓展出来的,走直线的话，我们期望要走的下一个点为此点右边的点
          Expected_note = Parent_node + n;
        case 'R' % ’R’ 表示当前的点是由右边的点拓展出来的，走直线的话，我们期望要走的下一个点为此点左边的点
           Expected_note =  Parent_node - n;
        case 'U' % ’U’ 表示当前的点是由上面的点拓展出来的，走直线的话，我们期望要走的下一个点为此点下面的点
           Expected_note = Parent_node +1;
        case 'D' % ’D’ 表示当前的点是由下边的点拓展出来的，走直线的话，我们期望要走的下一个点为此点上面的点
           Expected_note = Parent_node - 1;
    end

    if ((Expected_note<=0)||(Expected_note>n*n))    %如果我们期望的点不在待选点矩阵setOpen中，或者超出边界，跳过修正
          new_ii=ii;
          amend_count_1=amend_count;
    else

          %计算新的要进行拓展的点在setOPen中的索引值
         if fieldpointers{setOpen(ii)}==fieldpointers{Parent_node}    %如果修正之前要走的点就是我们期望的构成直线的点，跳出修正
                new_ii=ii;
                amend_count_1=amend_count;
         elseif  find(setOpen == Expected_note)  %如果我们期望要走的点在待选点矩阵setOpen中

                 untext_ii=find(setOpen == Expected_note);
                 now_cost=setOpenCosts(untext_ii)
+Weights*setOpenHeuristics(untext_ii);    %计算期望点要花费的代价

                if  temp==now_cost        %如果我们期望的点要花费的代价等于修正之前要走的点花费的代价，就进行修正（因为之前要走的点，是待选点矩阵setOPen中代价最小的一个点之一，所以期望的点的代价不可能小于该点）
                     new_ii=untext_ii;    %将新的setOPen矩阵的索引值赋值给new_ii输出
                     amend_count=amend_count+1;
                     amend_count_1=amend_count; %amend_count_1中记录了我们进行修正的次数，为了查看这个函数是否有发挥作用
                else
                     new_ii=ii;           %如果我们期望的点要花费的代价大于修正之前要走的点花费的代价，就跳过修正（A星算法要保证进行拓展的点是待选点中代价最小的，这也是导致远离终止点的哪一类拐角无法得到修正的原因）
                     amend_count_1=amend_count;
                end
         else
                 new_ii=ii;     %如果我们期望的点不在待选点矩阵setOpen中（也就是这个点是障碍物或者超出边界了），则跳过修正
```

```
                    amend_count_1=amend_count;
            end
        end
    end

end
```