《人工智能与智能驾驶基础》期末大作业

路径规划与轨迹跟踪
源代码

学院：汽车学院

成员：贾林轩 1853688

周展辉 1851154

软件：MATLAB 2021b

时间：2021 年 12 月 28 日

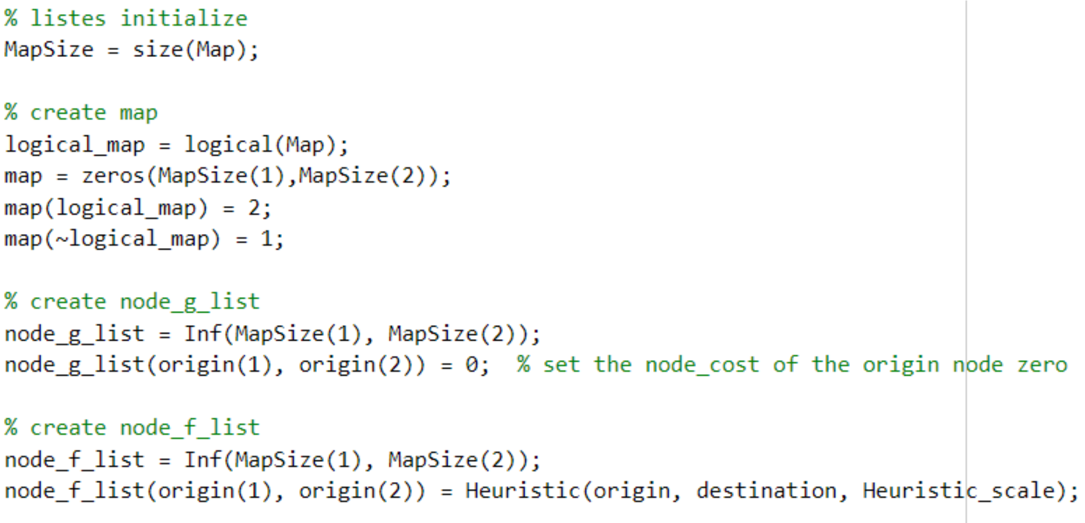
目录

1. 路径规划部分 3
[1]. A* 3
[2]. Dijkstra 5
[3]. 建立栅格地图 8
[4]. 执行路径规划 11
2. 轨迹跟踪部分 15

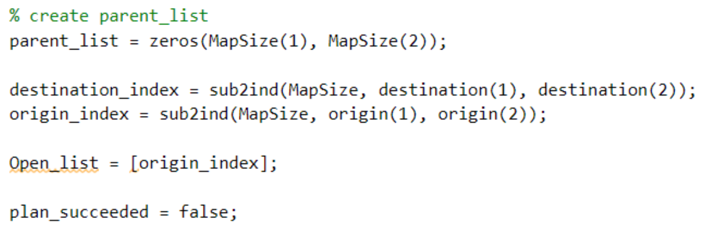
1. 路径规划部分

[1]. A*

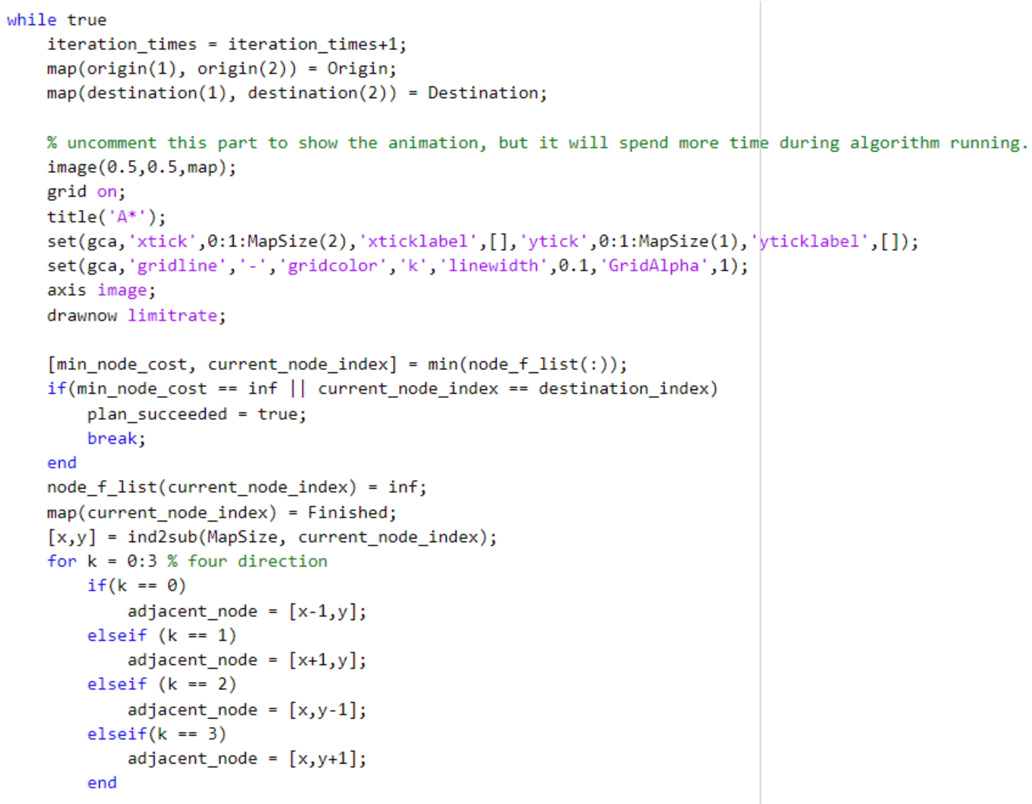
```
function [path, iteration_times,map] = Astar(Map, origin, destination)
% record the iteration times
iteration_times = 0;

%set the scale factor of Heuristic function
Heuristic_scale = 1;

% identifier setting
Obstacle = 2;
Origin = 3;
Destination = 4;
Finished = 5;
Unfinished = 6;
Path = 7;

% color setting
white = [1,1,1];
black = [0,0,0];
green = [0,1,0];
yellow = [1,1,0];
red = [1,0,0];
blue = [0,0,1];
cyan = [0,1,1];
color_list = [white; black; green; yellow; red; blue; cyan];
colormap(color_list);
```



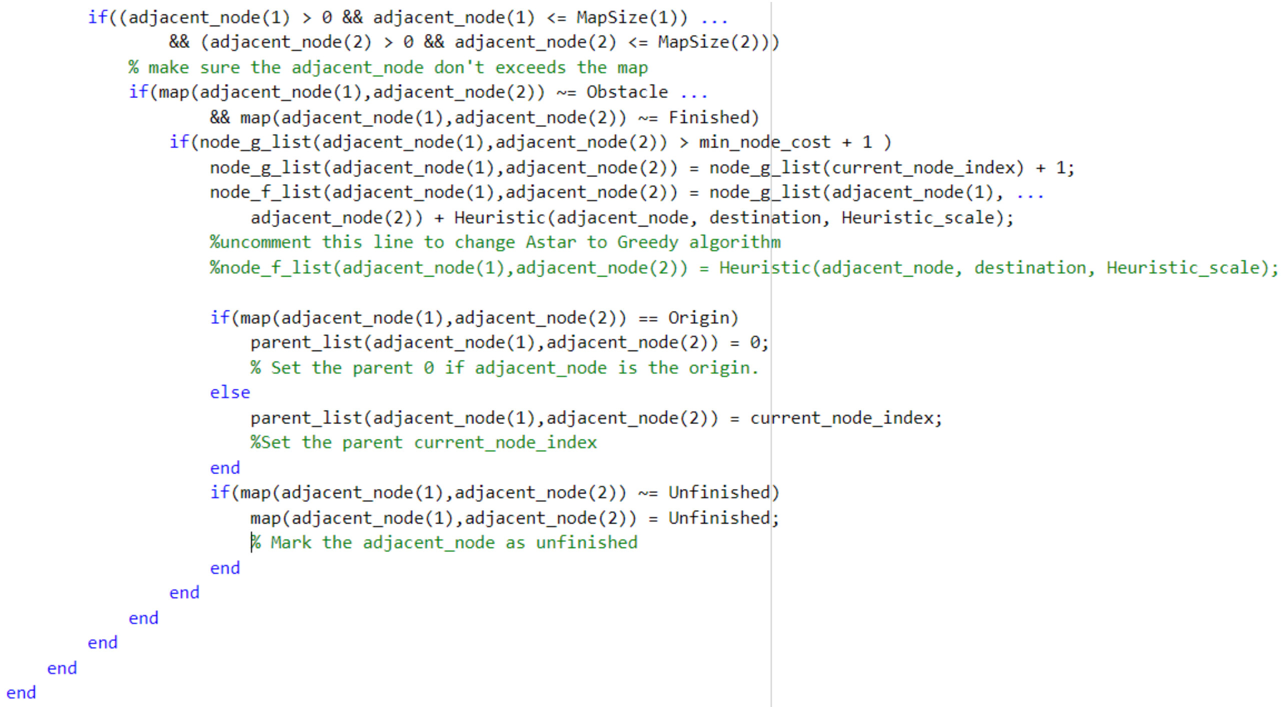
```
% listes initialize
MapSize = size(Map);

% create map
logical_map = logical(Map);
map = zeros(MapSize(1),MapSize(2));
map(logical_map) = 2;
map(~logical_map) = 1;

% create node_g_list
node_g_list = Inf(MapSize(1), MapSize(2));
node_g_list(origin(1), origin(2)) = 0;  % set the node_cost of the origin node zero

% create node_f_list
node_f_list = Inf(MapSize(1), MapSize(2));
node_f_list(origin(1), origin(2)) = Heuristic(origin, destination, Heuristic_scale);
```



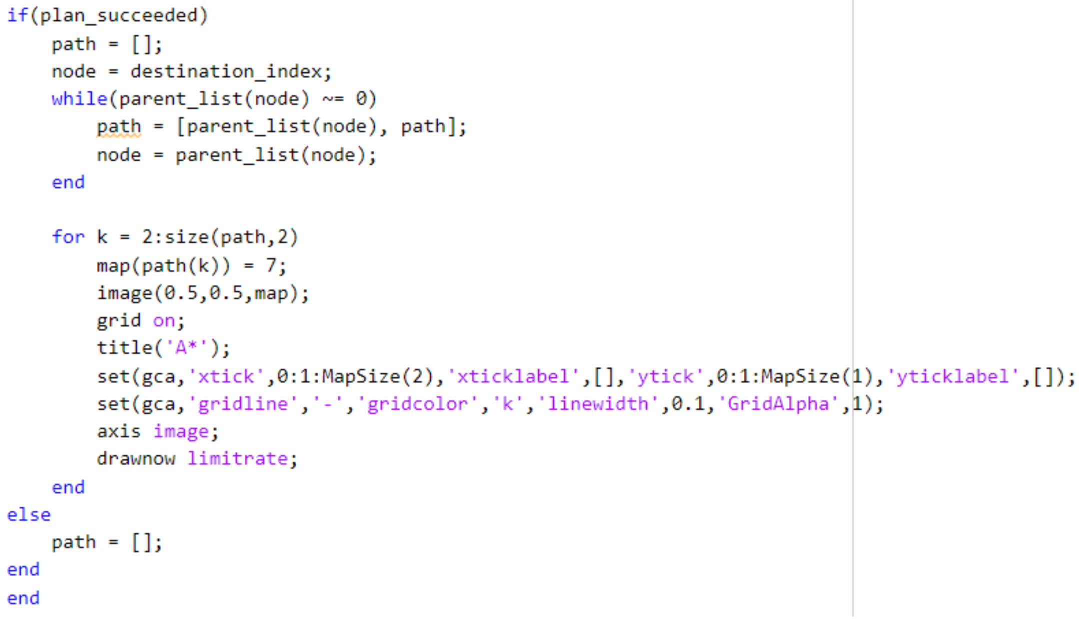
```
% create parent_list
parent_list = zeros(MapSize(1), MapSize(2));

destination_index = sub2ind(MapSize, destination(1), destination(2));
origin_index = sub2ind(MapSize, origin(1), origin(2));

Open_list = [origin_index];

plan_succeeded = false;
```



```
while true
    iteration_times = iteration_times+1;
    map(origin(1), origin(2)) = Origin;
    map(destination(1), destination(2)) = Destination;

    % uncomment this part to show the animation, but it will spend more time during algorithm running.
    image(0.5,0.5,map);
    grid on;
    title('A*');
    set(gca,'xtick',0:1:MapSize(2),'xticklabel',[],'ytick',0:1:MapSize(1),'yticklabel',[]);
    set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1);
    axis image;
    drawnow limitrate;

    [min_node_cost, current_node_index] = min(node_f_list(:));
    if(min_node_cost == inf || current_node_index == destination_index)
        plan_succeeded = true;
        break;
    end
    node_f_list(current_node_index) = inf;
    map(current_node_index) = Finished;
    [x,y] = ind2sub(MapSize, current_node_index);
    for k = 0:3 % four direction
        if(k == 0)
            adjacent_node = [x-1,y];
        elseif (k == 1)
            adjacent_node = [x+1,y];
        elseif (k == 2)
            adjacent_node = [x,y-1];
        elseif(k == 3)
            adjacent_node = [x,y+1];
        end
```

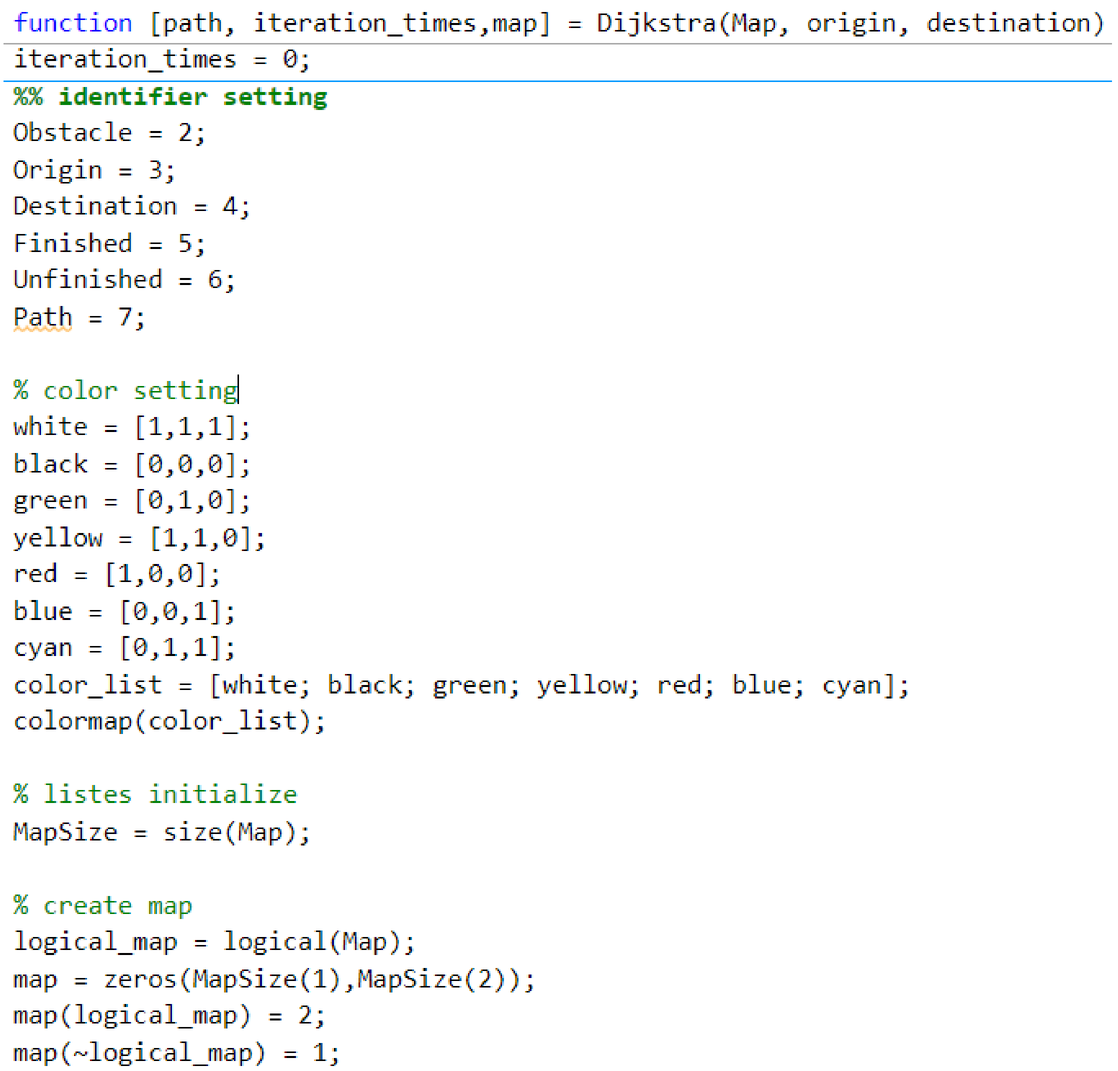


```
        if((adjacent_node(1) > 0 && adjacent_node(1) <= MapSize(1)) ...
                && (adjacent_node(2) > 0 && adjacent_node(2) <= MapSize(2)))
            % make sure the adjacent_node don't exceeds the map
            if(map(adjacent_node(1),adjacent_node(2)) ~= Obstacle ...
                    && map(adjacent_node(1),adjacent_node(2)) ~= Finished)
                if(node_g_list(adjacent_node(1),adjacent_node(2)) > min_node_cost + 1 )
                    node_g_list(adjacent_node(1),adjacent_node(2)) = node_g_list(current_node_index) + 1;
                    node_f_list(adjacent_node(1),adjacent_node(2)) = node_g_list(adjacent_node(1), ...
                        adjacent_node(2)) + Heuristic(adjacent_node, destination, Heuristic_scale);
                    %uncomment this line to change Astar to Greedy algorithm
                    %node_f_list(adjacent_node(1),adjacent_node(2)) = Heuristic(adjacent_node, destination, Heuristic_scale);

                    if(map(adjacent_node(1),adjacent_node(2)) == Origin)
                        parent_list(adjacent_node(1),adjacent_node(2)) = 0;
                        % Set the parent 0 if adjacent_node is the origin.
                    else
                        parent_list(adjacent_node(1),adjacent_node(2)) = current_node_index;
                        %Set the parent current_node_index
                    end
                    if(map(adjacent_node(1),adjacent_node(2)) ~= Unfinished)
                        map(adjacent_node(1),adjacent_node(2)) = Unfinished;
                        % Mark the adjacent_node as unfinished
                    end
                end
            end
        end
    end
end
```

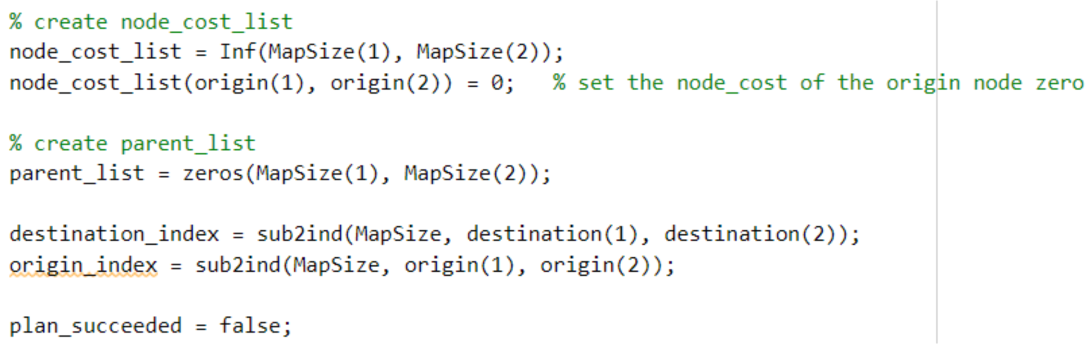


```
if(plan_succeeded)
    path = [];
    node = destination_index;
    while(parent_list(node) ~= 0)
        path = [parent_list(node), path];
        node = parent_list(node);
    end

    for k = 2:size(path,2)
        map(path(k)) = 7;
        image(0.5,0.5,map);
        grid on;
        title('A*');
        set(gca,'xtick',0:1:MapSize(2),'xticklabel',[],'ytick',0:1:MapSize(1),'yticklabel',[]);
        set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1);
        axis image;
        drawnow limitrate;
    end
else
    path = [];
end
end
```

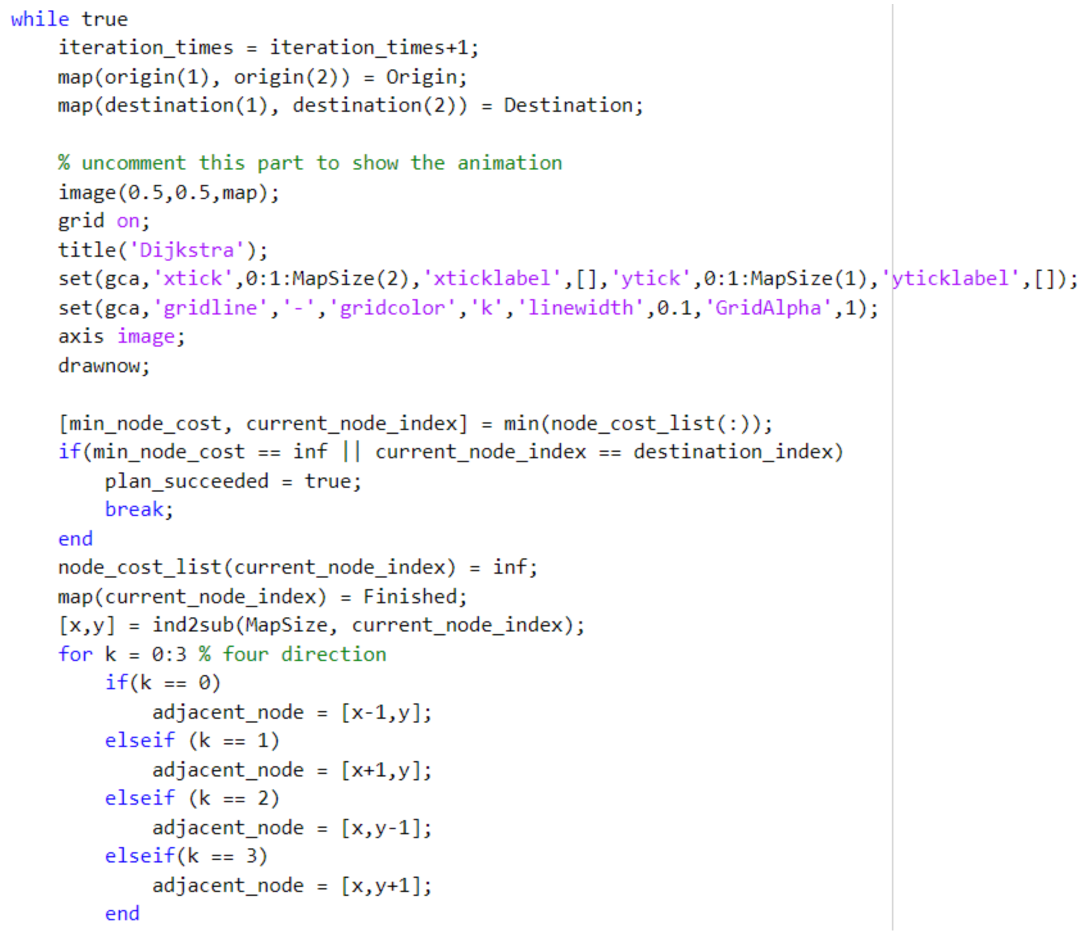
[2]. Dijkstra



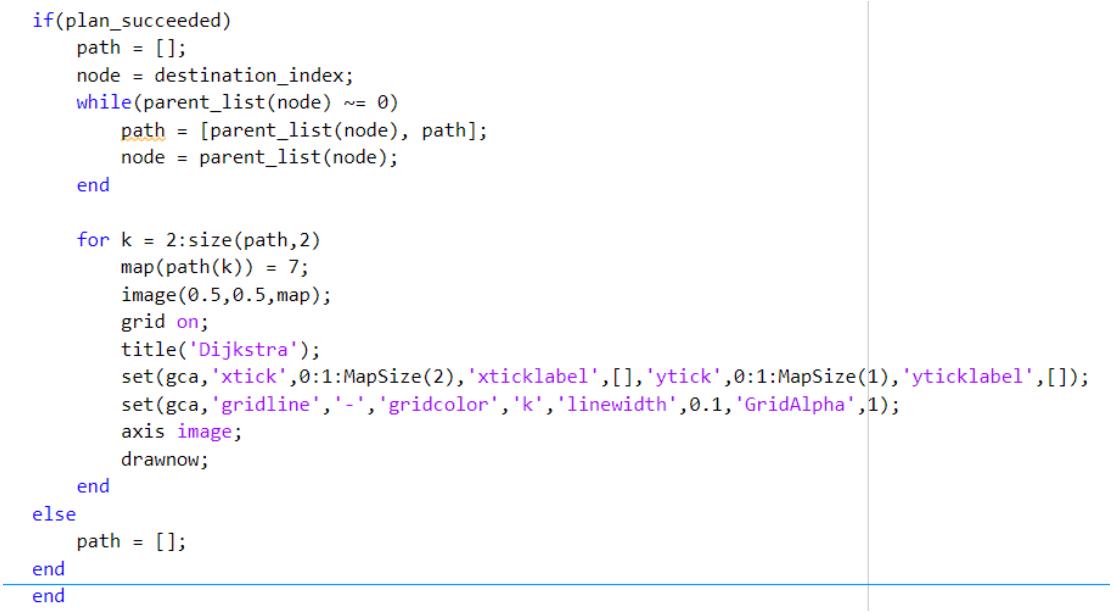
```
function [path, iteration_times,map] = Dijkstra(Map, origin, destination)
iteration_times = 0;
%% identifier setting
Obstacle = 2;
Origin = 3;
Destination = 4;
Finished = 5;
Unfinished = 6;
Path = 7;

% color setting
white = [1,1,1];
black = [0,0,0];
green = [0,1,0];
yellow = [1,1,0];
red = [1,0,0];
blue = [0,0,1];
cyan = [0,1,1];
color_list = [white; black; green; yellow; red; blue; cyan];
colormap(color_list);

% listes initialize
MapSize = size(Map);

% create map
logical_map = logical(Map);
map = zeros(MapSize(1),MapSize(2));
map(logical_map) = 2;
map(~logical_map) = 1;
```



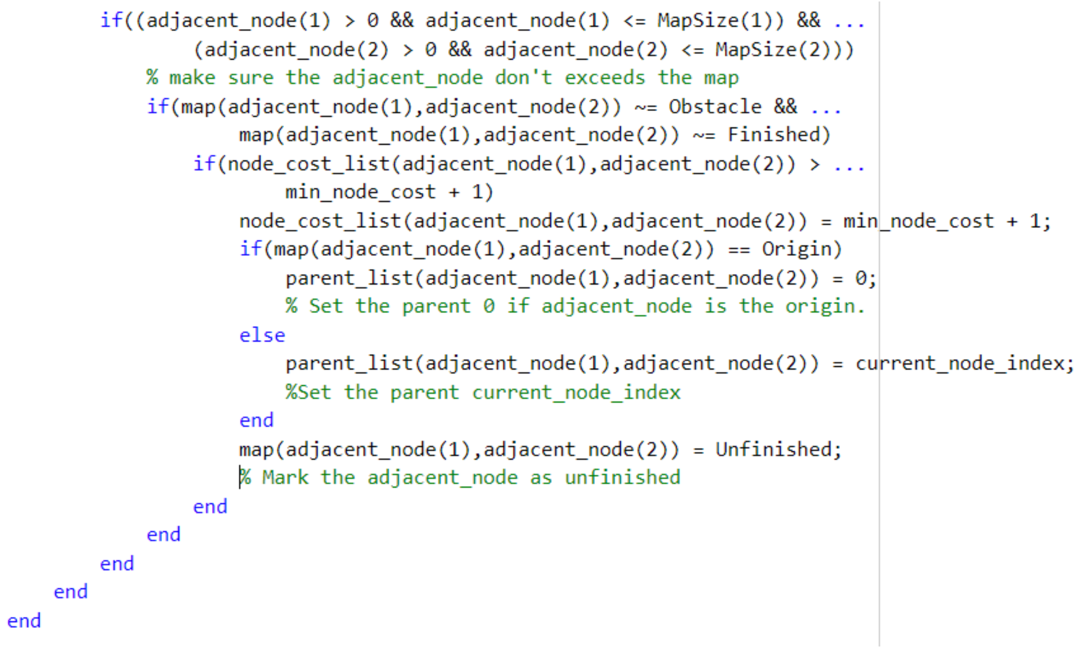
```
% create node_cost_list
node_cost_list = Inf(MapSize(1), MapSize(2));
node_cost_list(origin(1), origin(2)) = 0;   % set the node_cost of the origin node zero

% create parent_list
parent_list = zeros(MapSize(1), MapSize(2));

destination_index = sub2ind(MapSize, destination(1), destination(2));
origin_index = sub2ind(MapSize, origin(1), origin(2));

plan_succeeded = false;
```



```
while true
    iteration_times = iteration_times+1;
    map(origin(1), origin(2)) = Origin;
    map(destination(1), destination(2)) = Destination;

    % uncomment this part to show the animation
    image(0.5,0.5,map);
    grid on;
    title('Dijkstra');
    set(gca,'xtick',0:1:MapSize(2),'xticklabel',[],'ytick',0:1:MapSize(1),'yticklabel',[]);
    set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1);
    axis image;
    drawnow;

    [min_node_cost, current_node_index] = min(node_cost_list(:));
    if(min_node_cost == inf || current_node_index == destination_index)
        plan_succeeded = true;
        break;
    end
    node_cost_list(current_node_index) = inf;
    map(current_node_index) = Finished;
    [x,y] = ind2sub(MapSize, current_node_index);
    for k = 0:3 % four direction
        if(k == 0)
            adjacent_node = [x-1,y];
        elseif (k == 1)
            adjacent_node = [x+1,y];
        elseif (k == 2)
            adjacent_node = [x,y-1];
        elseif(k == 3)
            adjacent_node = [x,y+1];
        end
```

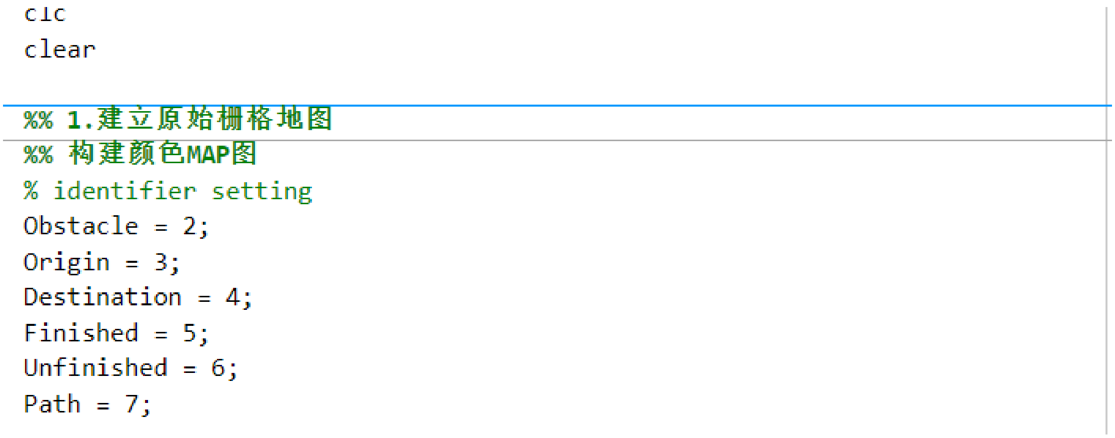


```
if(plan_succeeded)
    path = [];
    node = destination_index;
    while(parent_list(node) ~= 0)
        path = [parent_list(node), path];
        node = parent_list(node);
    end

    for k = 2:size(path,2)
        map(path(k)) = 7;
        image(0.5,0.5,map);
        grid on;
        title('Dijkstra');
        set(gca,'xtick',0:1:MapSize(2),'xticklabel',[],'ytick',0:1:MapSize(1),'yticklabel',[]);
        set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1);
        axis image;
        drawnow;
    end
else
    path = [];
end
end
```

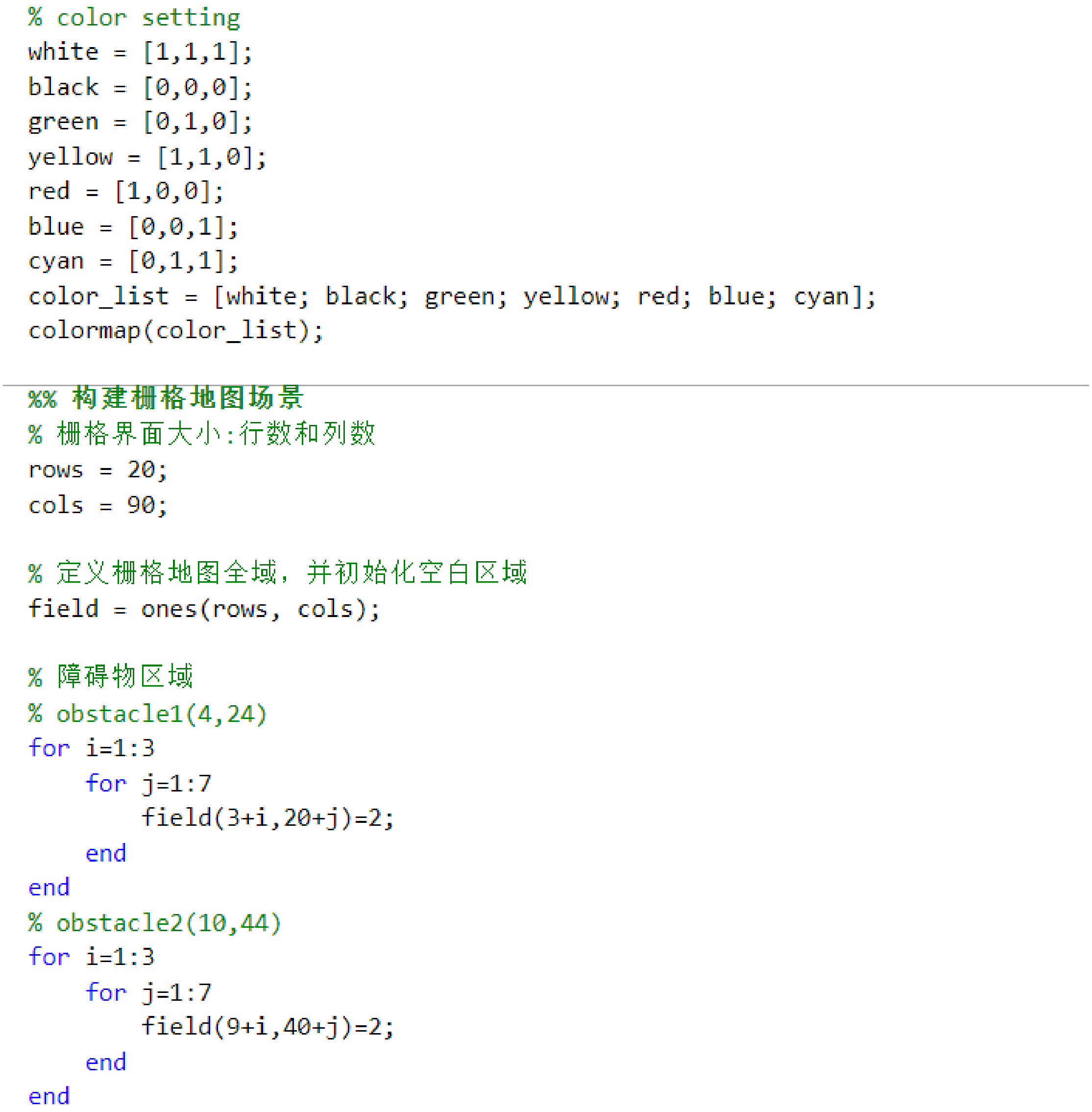


```
        if((adjacent_node(1) > 0 && adjacent_node(1) <= MapSize(1)) && ...
                (adjacent_node(2) > 0 && adjacent_node(2) <= MapSize(2)))
            % make sure the adjacent_node don't exceeds the map
            if(map(adjacent_node(1),adjacent_node(2)) ~= Obstacle && ...
                    map(adjacent_node(1),adjacent_node(2)) ~= Finished)
                if(node_cost_list(adjacent_node(1),adjacent_node(2)) > ...
                        min_node_cost + 1)
                    node_cost_list(adjacent_node(1),adjacent_node(2)) = min_node_cost + 1;
                    if(map(adjacent_node(1),adjacent_node(2)) == Origin)
                        parent_list(adjacent_node(1),adjacent_node(2)) = 0;
                        % Set the parent 0 if adjacent_node is the origin.
                    else
                        parent_list(adjacent_node(1),adjacent_node(2)) = current_node_index;
                        %Set the parent current_node_index
                    end
                    map(adjacent_node(1),adjacent_node(2)) = Unfinished;
                    % Mark the adjacent_node as unfinished
                end
            end
        end
    end
end
```

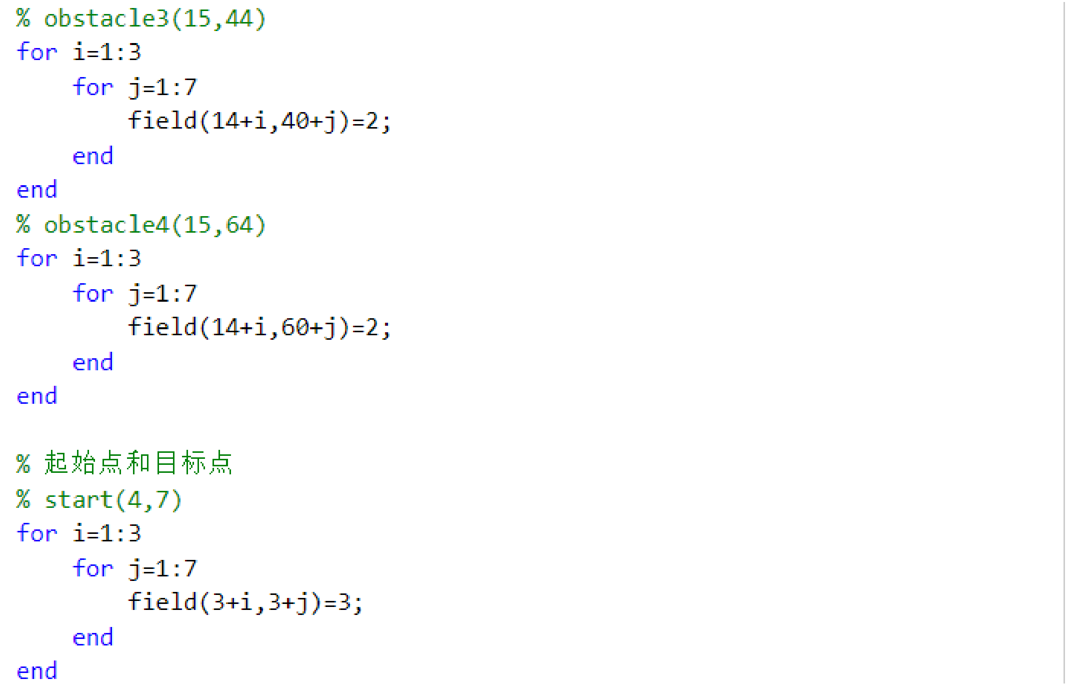
[3]. 建立栅格地图



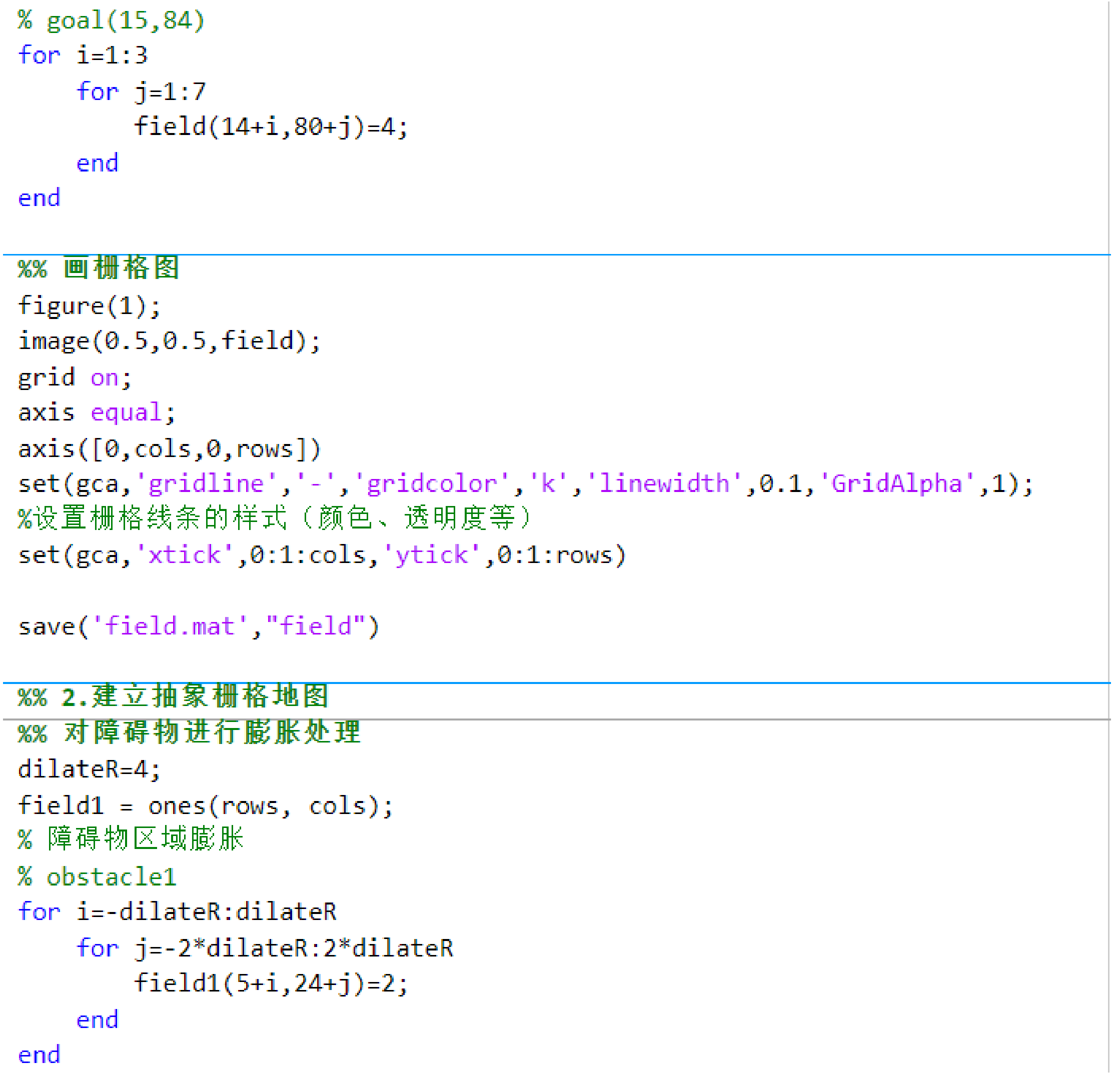
```
clc
clear

%% 1.建立原始栅格地图
%% 构建颜色MAP图
% identifier setting
Obstacle = 2;
Origin = 3;
Destination = 4;
Finished = 5;
Unfinished = 6;
Path = 7;
```



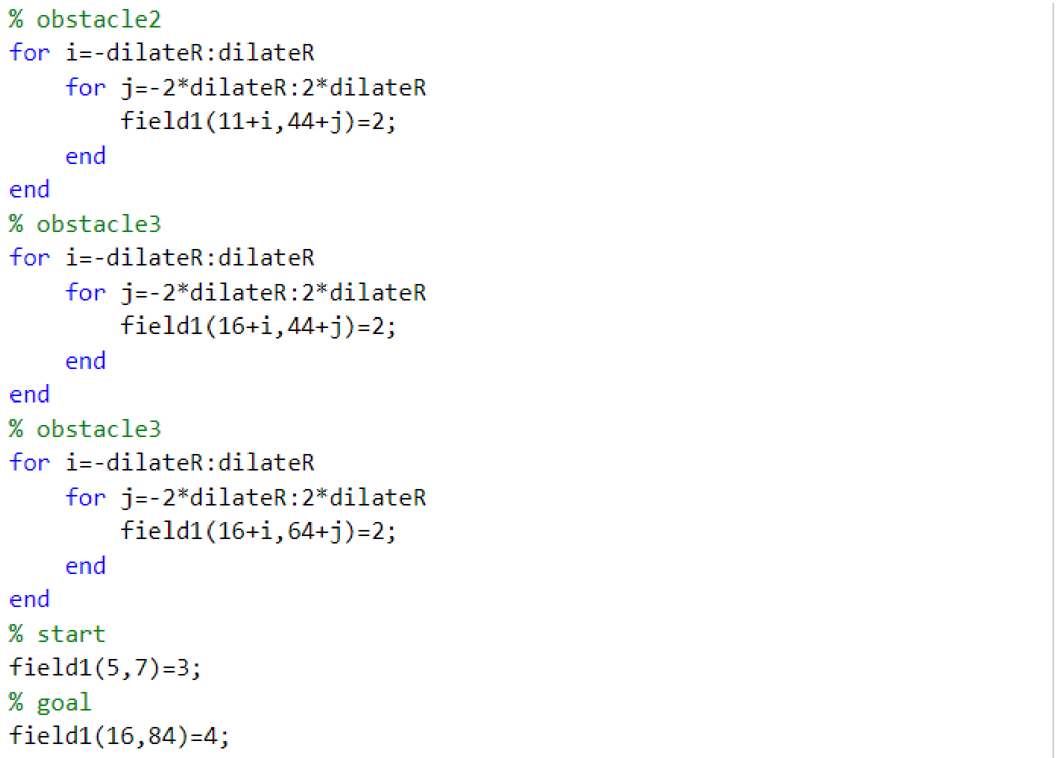
```
% color setting
white = [1,1,1];
black = [0,0,0];
green = [0,1,0];
yellow = [1,1,0];
red = [1,0,0];
blue = [0,0,1];
cyan = [0,1,1];
color_list = [white; black; green; yellow; red; blue; cyan];
colormap(color_list);

%% 构建栅格地图场景
% 栅格界面大小:行数和列数
rows = 20;
cols = 90;

% 定义栅格地图全域，并初始化空白区域
field = ones(rows, cols);

% 障碍物区域
% obstacle1(4,24)
for i=1:3
    for j=1:7
        field(3+i,20+j)=2;
    end
end
% obstacle2(10,44)
for i=1:3
    for j=1:7
        field(9+i,40+j)=2;
    end
end
```



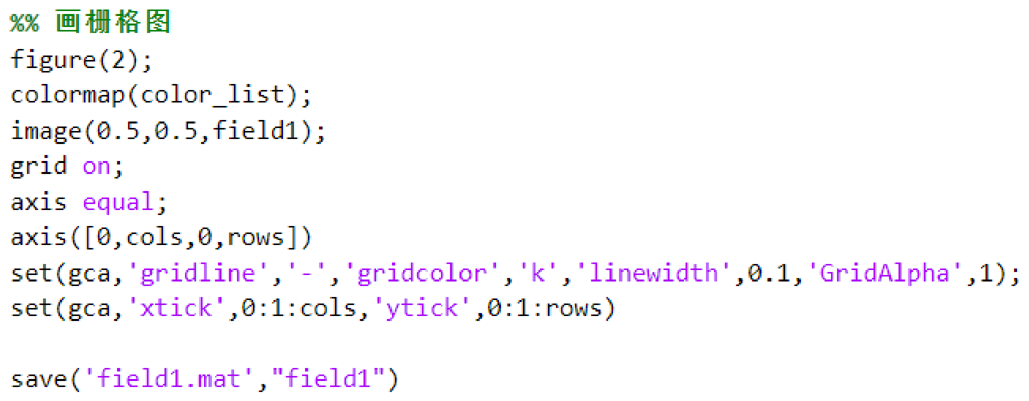
```
% obstacle3(15,44)
for i=1:3
    for j=1:7
        field(14+i,40+j)=2;
    end
end
% obstacle4(15,64)
for i=1:3
    for j=1:7
        field(14+i,60+j)=2;
    end
end

% 起始点和目标点
% start(4,7)
for i=1:3
    for j=1:7
        field(3+i,3+j)=3;
    end
end
```



```
% goal(15,84)
for i=1:3
    for j=1:7
        field(14+i,80+j)=4;
    end
end

%% 画栅格图
figure(1);
image(0.5,0.5,field);
grid on;
axis equal;
axis([0,cols,0,rows])
set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1);
%设置栅格线条的样式（颜色、透明度等）
set(gca,'xtick',0:1:cols,'ytick',0:1:rows)

save('field.mat',"field")

%% 2.建立抽象栅格地图
%% 对障碍物进行膨胀处理
dilateR=4;
field1 = ones(rows, cols);
% 障碍物区域膨胀
% obstacle1
for i=-dilateR:dilateR
    for j=-2*dilateR:2*dilateR
        field1(5+i,24+j)=2;
    end
end
```

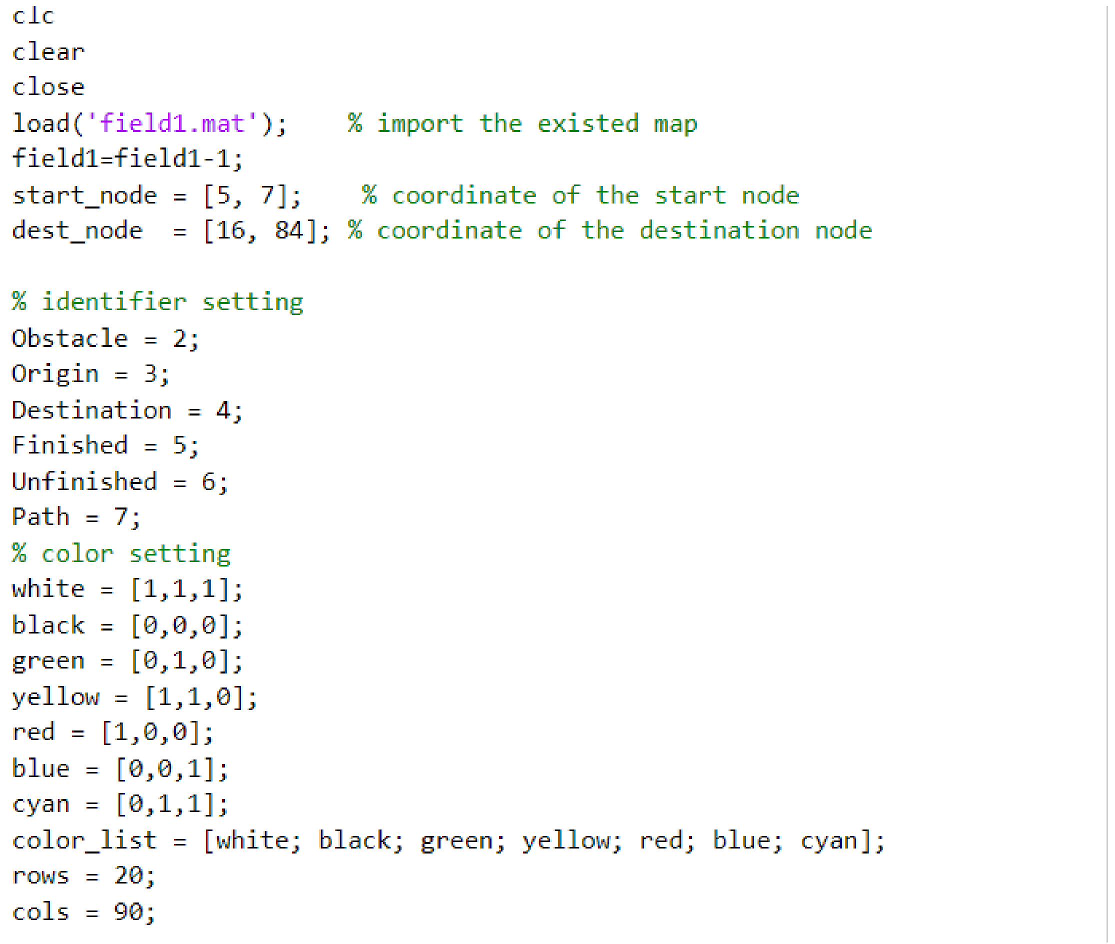


```
% obstacle2
for i=-dilateR:dilateR
    for j=-2*dilateR:2*dilateR
        field1(11+i,44+j)=2;
    end
end
% obstacle3
for i=-dilateR:dilateR
    for j=-2*dilateR:2*dilateR
        field1(16+i,44+j)=2;
    end
end
% obstacle3
for i=-dilateR:dilateR
    for j=-2*dilateR:2*dilateR
        field1(16+i,64+j)=2;
    end
end
% start
field1(5,7)=3;
% goal
field1(16,84)=4;
```

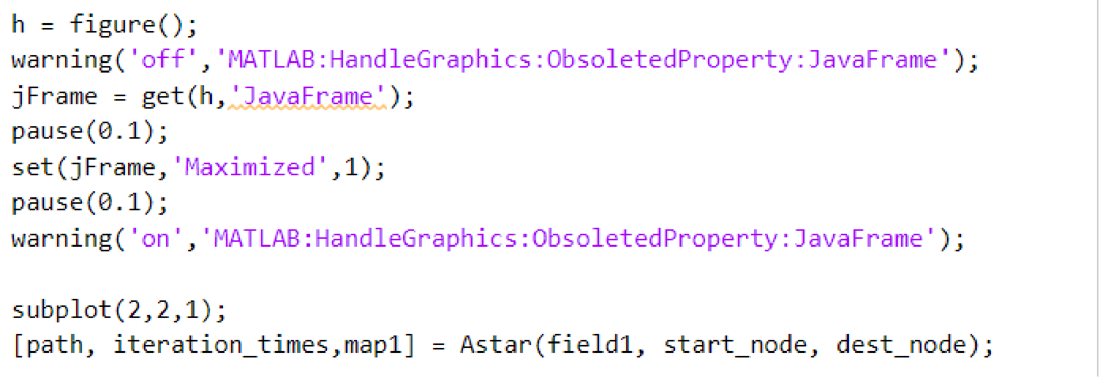


```
%% 画栅格图
figure(2);
colormap(color_list);
image(0.5,0.5,field1);
grid on;
axis equal;
axis([0,cols,0,rows])
set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1);
set(gca,'xtick',0:1:cols,'ytick',0:1:rows)

save('field1.mat',"field1")
```

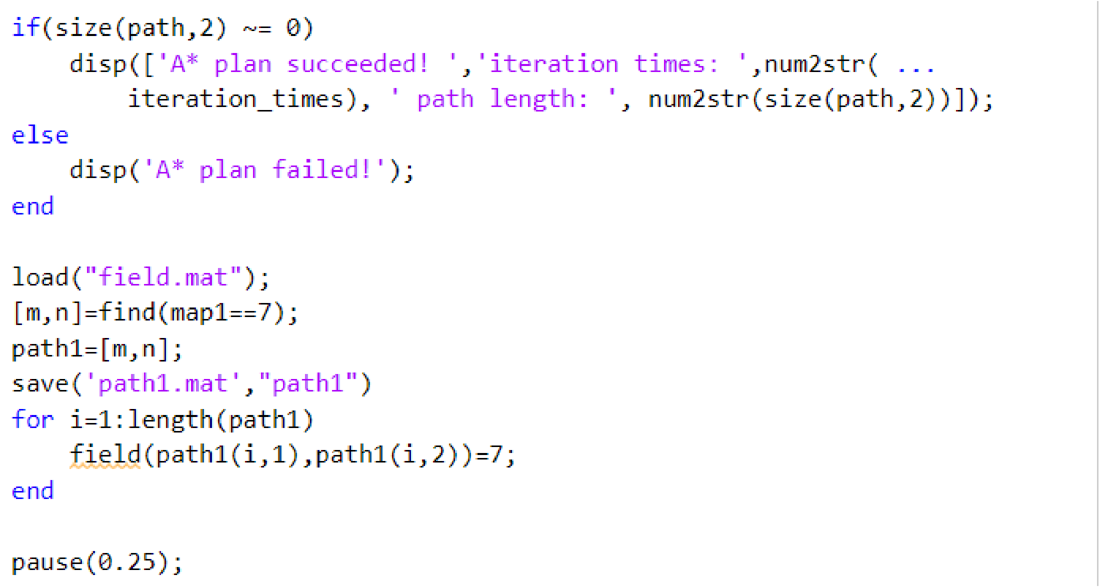
[4]. 执行路径规划



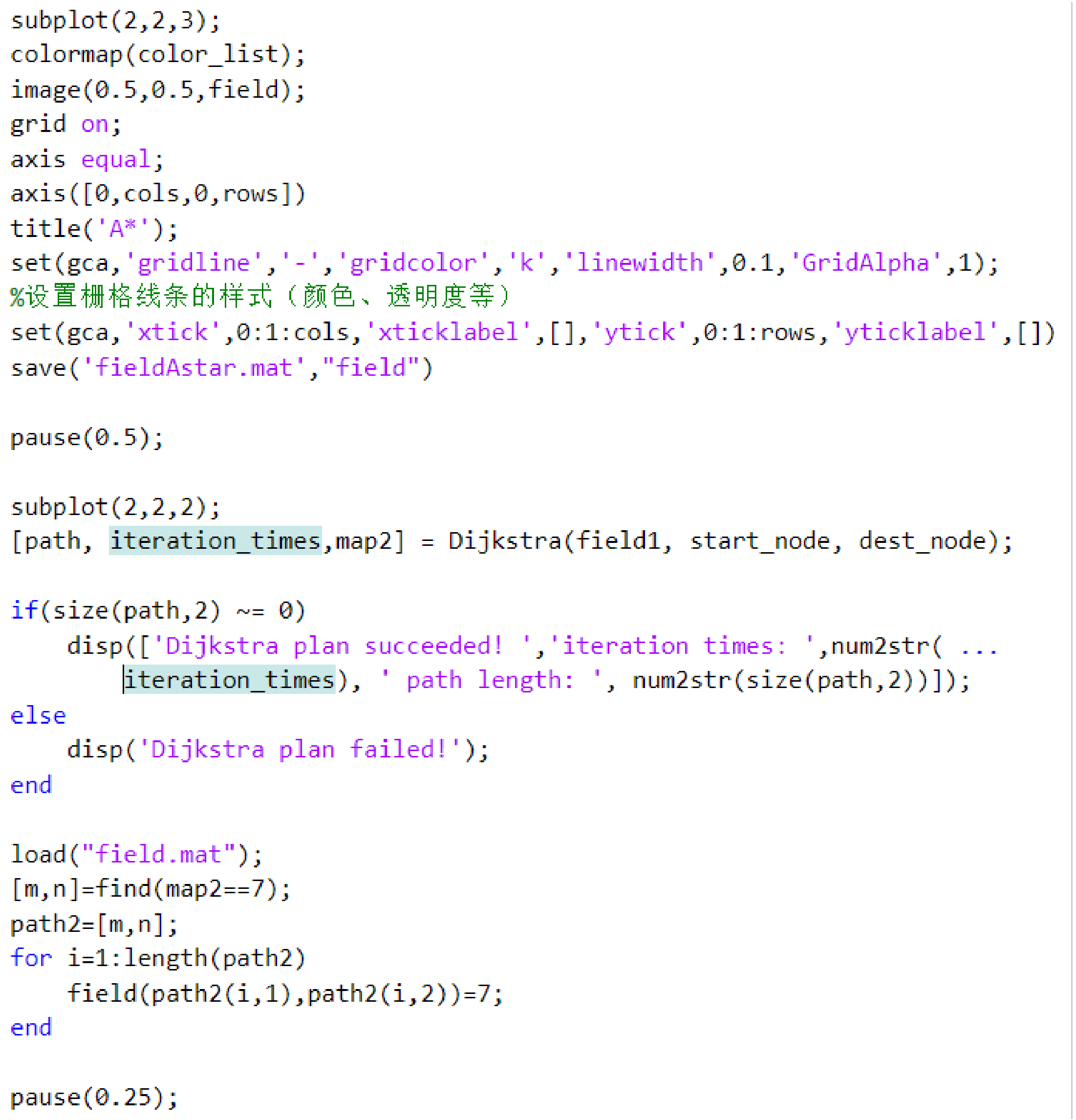
```
clc
clear
close
load('field1.mat');    % import the existed map
field1=field1-1;
start_node = [5, 7];    % coordinate of the start node
dest_node  = [16, 84]; % coordinate of the destination node

% identifier setting
Obstacle = 2;
Origin = 3;
Destination = 4;
Finished = 5;
Unfinished = 6;
Path = 7;
% color setting
white = [1,1,1];
black = [0,0,0];
green = [0,1,0];
yellow = [1,1,0];
red = [1,0,0];
blue = [0,0,1];
cyan = [0,1,1];
color_list = [white; black; green; yellow; red; blue; cyan];
rows = 20;
cols = 90;
```



```
h = figure();
warning('off','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');
jFrame = get(h,'JavaFrame');
pause(0.1);
set(jFrame,'Maximized',1);
pause(0.1);
warning('on','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');

subplot(2,2,1);
[path, iteration_times,map1] = Astar(field1, start_node, dest_node);
```



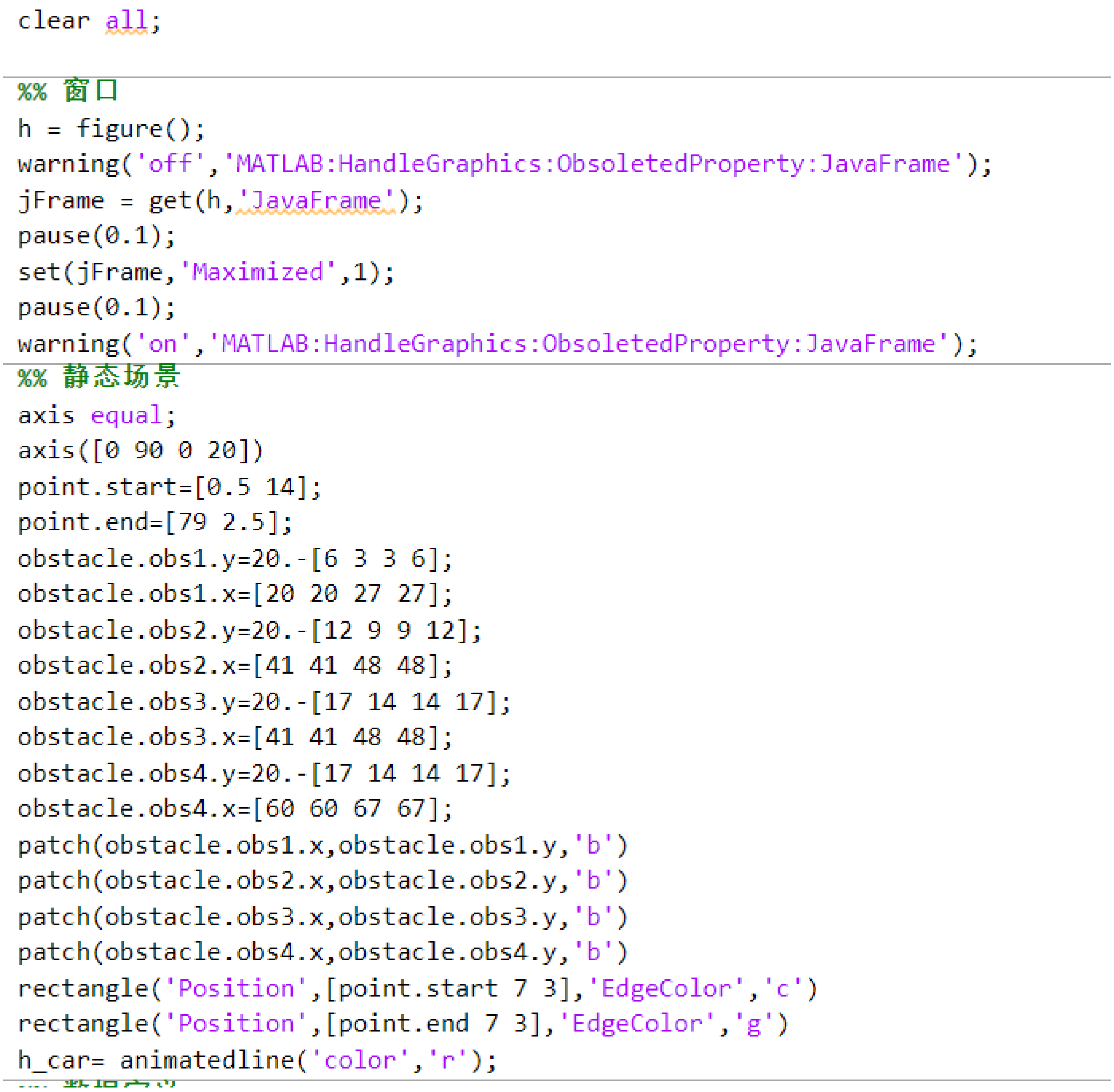
```
if(size(path,2) ~= 0)
    disp(['A* plan succeeded! ','iteration times: ',num2str( ...
        iteration_times), ' path length: ', num2str(size(path,2))]);
else
    disp('A* plan failed!');
end

load("field.mat");
[m,n]=find(map1==7);
path1=[m,n];
save('path1.mat',"path1")
for i=1:length(path1)
    field(path1(i,1),path1(i,2))=7;
end

pause(0.25);
```



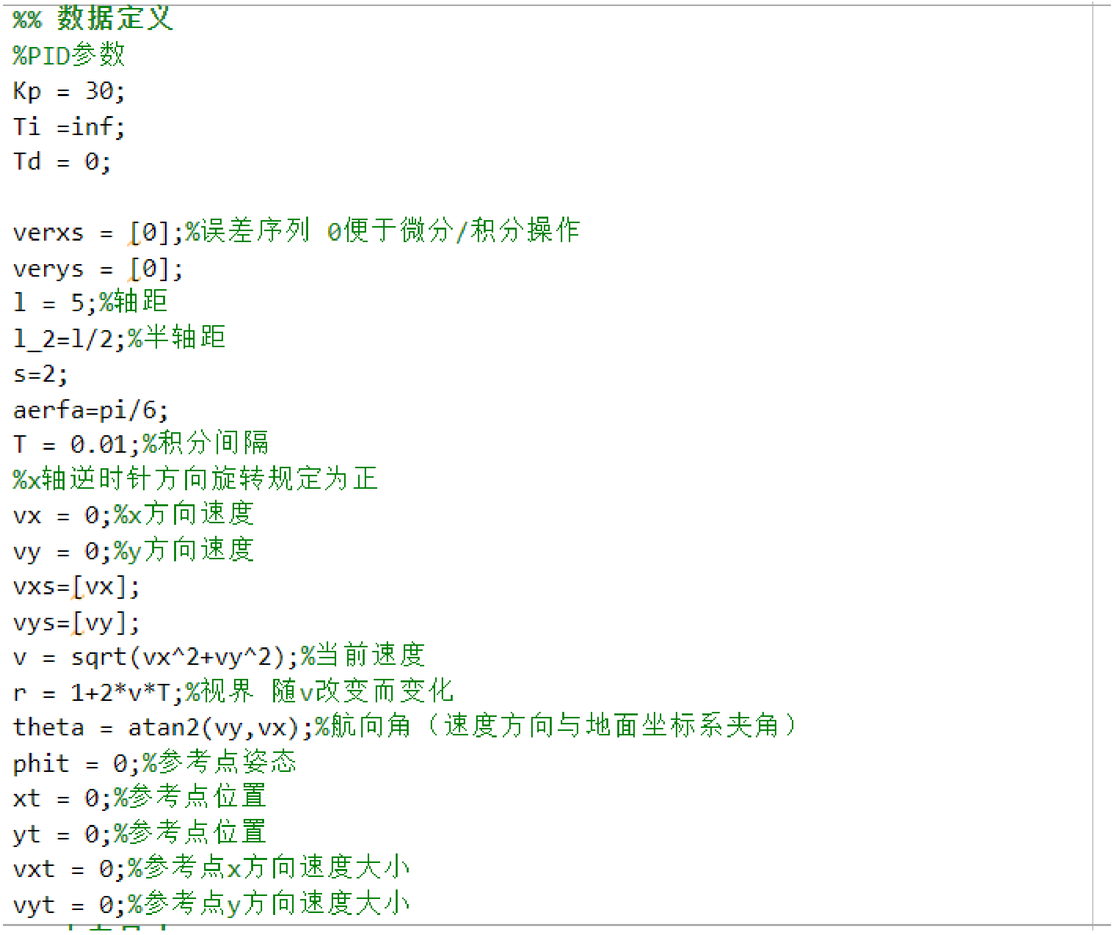
```
subplot(2,2,3);
colormap(color_list);
image(0.5,0.5,field);
grid on;
axis equal;
axis([0,cols,0,rows])
title('A*');
set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1);
%设置栅格线条的样式（颜色、透明度等）
set(gca,'xtick',0:1:cols,'xticklabel',[],'ytick',0:1:rows,'yticklabel',[])
save('fieldAstar.mat',"field")

pause(0.5);

subplot(2,2,2);
[path, iteration_times,map2] = Dijkstra(field1, start_node, dest_node);

if(size(path,2) ~= 0)
    disp(['Dijkstra plan succeeded! ','iteration times: ',num2str( ...
        iteration_times), ' path length: ', num2str(size(path,2))]);
else
    disp('Dijkstra plan failed!');
end

load("field.mat");
[m,n]=find(map2==7);
path2=[m,n];
for i=1:length(path2)
    field(path2(i,1),path2(i,2))=7;
end

pause(0.25);
```

```
subplot(2,2,4);
colormap(color_list);
image(0.5,0.5,field);
grid on;
axis equal;
axis([0,cols,0,rows])
title('Dijkstra');
set(gca,'gridline','-','gridcolor','k','linewidth',0.1,'GridAlpha',1); %设置栅格线条的样式（颜色、透明度等）
set(gca,'xtick',0:1:cols,'xticklabel',[],'ytick',0:1:rows,'yticklabel',[])
save('fieldDijkstra.mat',"field")
```

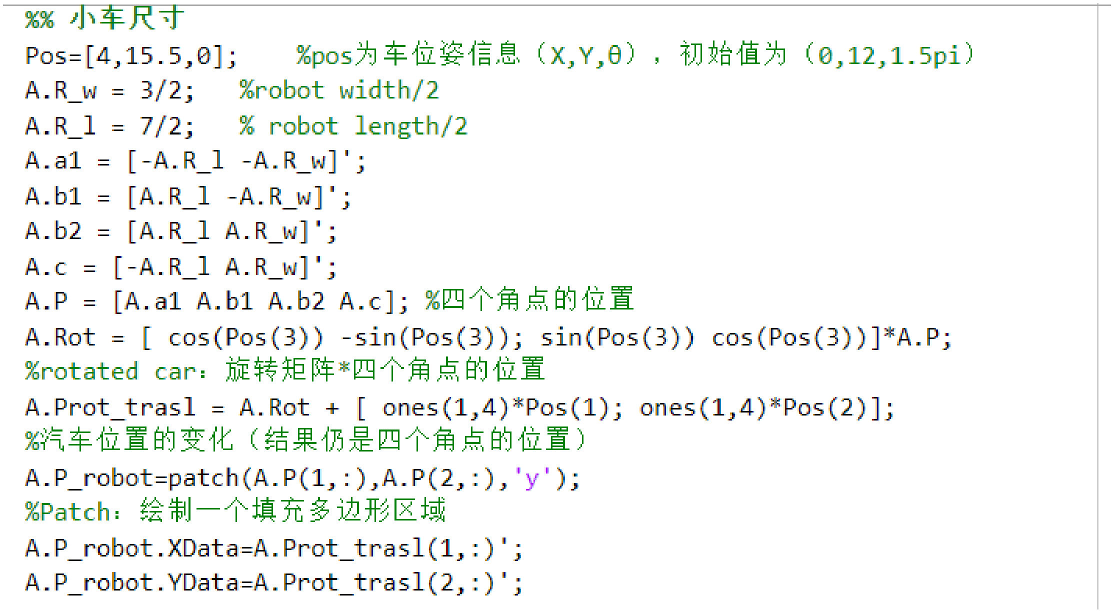
2. 轨迹跟踪部分



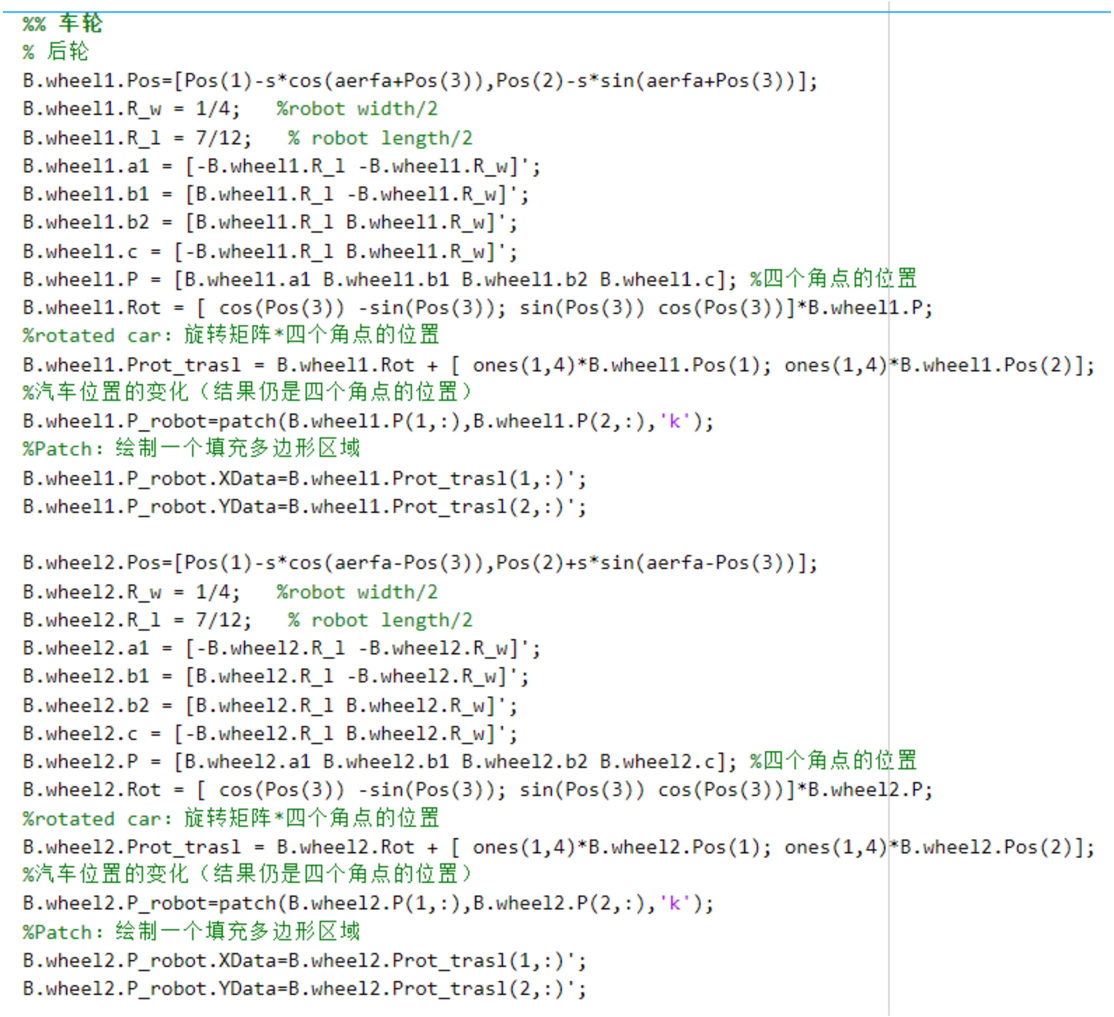
```
clear all;

%% 窗口
h = figure();
warning('off','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');
jFrame = get(h,'JavaFrame');
pause(0.1);
set(jFrame,'Maximized',1);
pause(0.1);
warning('on','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');
%% 静态场景
axis equal;
axis([0 90 0 20])
point.start=[0.5 14];
point.end=[79 2.5];
obstacle.obs1.y=20.-[6 3 3 6];
obstacle.obs1.x=[20 20 27 27];
obstacle.obs2.y=20.-[12 9 9 12];
obstacle.obs2.x=[41 41 48 48];
obstacle.obs3.y=20.-[17 14 14 17];
obstacle.obs3.x=[41 41 48 48];
obstacle.obs4.y=20.-[17 14 14 17];
obstacle.obs4.x=[60 60 67 67];
patch(obstacle.obs1.x,obstacle.obs1.y,'b')
patch(obstacle.obs2.x,obstacle.obs2.y,'b')
patch(obstacle.obs3.x,obstacle.obs3.y,'b')
patch(obstacle.obs4.x,obstacle.obs4.y,'b')
rectangle('Position',[point.start 7 3],'EdgeColor','c')
rectangle('Position',[point.end 7 3],'EdgeColor','g')
h_car= animatedline('color','r');
```



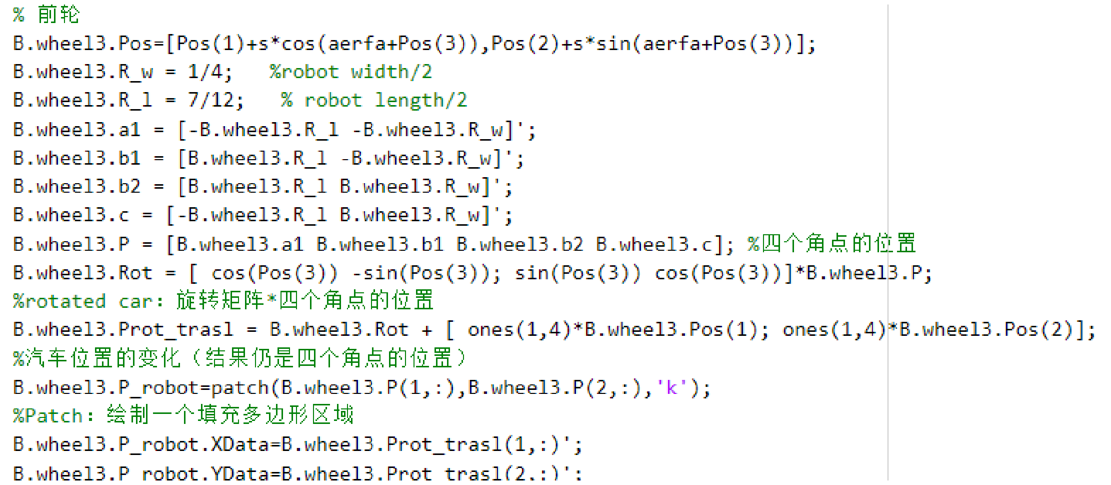
```
%% 数据定义
%PID参数
Kp = 30;
Ti =inf;
Td = 0;

verxs = [0];%误差序列 0便于微分/积分操作
verys = [0];
l = 5;%轴距
l_2=l/2;%半轴距
s=2;
aerfa=pi/6;
T = 0.01;%积分间隔
%x轴逆时针方向旋转规定为正
vx = 0;%x方向速度
vy = 0;%y方向速度
vxs=[vx];
vys=[vy];
v = sqrt(vx^2+vy^2);%当前速度
r = 1+2*v*T;%视界 随v改变而变化
theta = atan2(vy,vx);%航向角（速度方向与地面坐标系夹角）
phit = 0;%参考点姿态
xt = 0;%参考点位置
yt = 0;%参考点位置
vxt = 0;%参考点x方向速度大小
vyt = 0;%参考点y方向速度大小
```



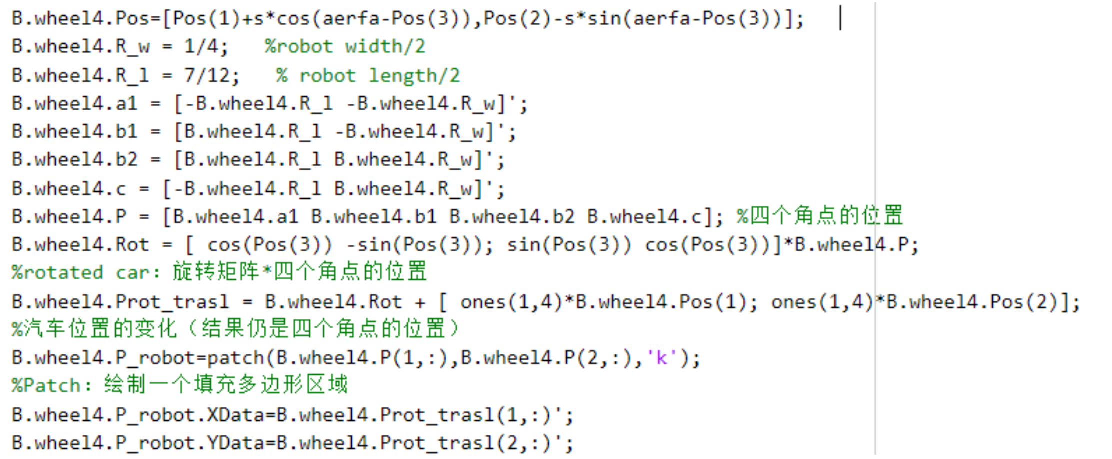
```
%% 小车尺寸
Pos=[4,15.5,0];     %pos为车位姿信息（X,Y,θ），初始值为（0,12,1.5pi）
A.R_w = 3/2;    %robot width/2
A.R_l = 7/2;    % robot length/2
A.a1 = [-A.R_l -A.R_w]';
A.b1 = [A.R_l -A.R_w]';
A.b2 = [A.R_l A.R_w]';
A.c = [-A.R_l A.R_w]';
A.P = [A.a1 A.b1 A.b2 A.c]; %四个角点的位置
A.Rot = [ cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*A.P;
%rotated car: 旋转矩阵*四个角点的位置
A.Prot_trasl = A.Rot + [ ones(1,4)*Pos(1); ones(1,4)*Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
A.P_robot=patch(A.P(1,:),A.P(2,:),'y');
%Patch: 绘制一个填充多边形区域
A.P_robot.XData=A.Prot_trasl(1,:)';
A.P_robot.YData=A.Prot_trasl(2,:)';
```



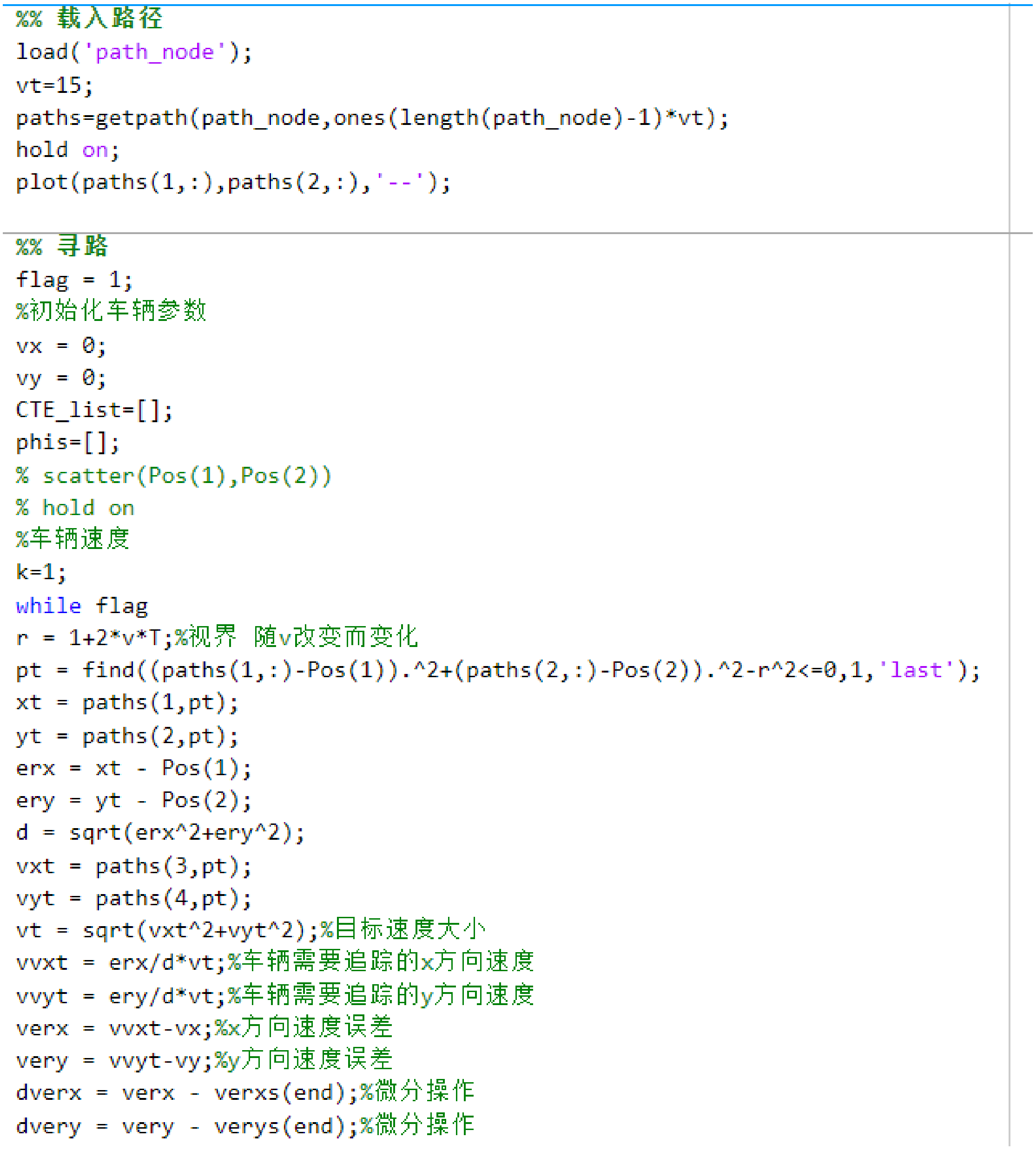
```
%% 车轮
% 后轮
B.wheel1.Pos=[Pos(1)-s*cos(aerfa+Pos(3)),Pos(2)-s*sin(aerfa+Pos(3))];
B.wheel1.R_w = 1/4;   %robot width/2
B.wheel1.R_l = 7/12;   % robot length/2
B.wheel1.a1 = [-B.wheel1.R_l -B.wheel1.R_w]';
B.wheel1.b1 = [B.wheel1.R_l -B.wheel1.R_w]';
B.wheel1.b2 = [B.wheel1.R_l B.wheel1.R_w]';
B.wheel1.c = [-B.wheel1.R_l B.wheel1.R_w]';
B.wheel1.P = [B.wheel1.a1 B.wheel1.b1 B.wheel1.b2 B.wheel1.c]; %四个角点的位置
B.wheel1.Rot = [ cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*B.wheel1.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel1.Prot_trasl = B.wheel1.Rot + [ ones(1,4)*B.wheel1.Pos(1); ones(1,4)*B.wheel1.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel1.P_robot=patch(B.wheel1.P(1,:),B.wheel1.P(2,:),'k');
%Patch: 绘制一个填充多边形区域
B.wheel1.P_robot.XData=B.wheel1.Prot_trasl(1,:)';
B.wheel1.P_robot.YData=B.wheel1.Prot_trasl(2,:)';

B.wheel2.Pos=[Pos(1)-s*cos(aerfa-Pos(3)),Pos(2)+s*sin(aerfa-Pos(3))];
B.wheel2.R_w = 1/4;   %robot width/2
B.wheel2.R_l = 7/12;   % robot length/2
B.wheel2.a1 = [-B.wheel2.R_l -B.wheel2.R_w]';
B.wheel2.b1 = [B.wheel2.R_l -B.wheel2.R_w]';
B.wheel2.b2 = [B.wheel2.R_l B.wheel2.R_w]';
B.wheel2.c = [-B.wheel2.R_l B.wheel2.R_w]';
B.wheel2.P = [B.wheel2.a1 B.wheel2.b1 B.wheel2.b2 B.wheel2.c]; %四个角点的位置
B.wheel2.Rot = [ cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*B.wheel2.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel2.Prot_trasl = B.wheel2.Rot + [ ones(1,4)*B.wheel2.Pos(1); ones(1,4)*B.wheel2.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel2.P_robot=patch(B.wheel2.P(1,:),B.wheel2.P(2,:),'k');
%Patch: 绘制一个填充多边形区域
B.wheel2.P_robot.XData=B.wheel2.Prot_trasl(1,:)';
B.wheel2.P_robot.YData=B.wheel2.Prot_trasl(2,:)';
```



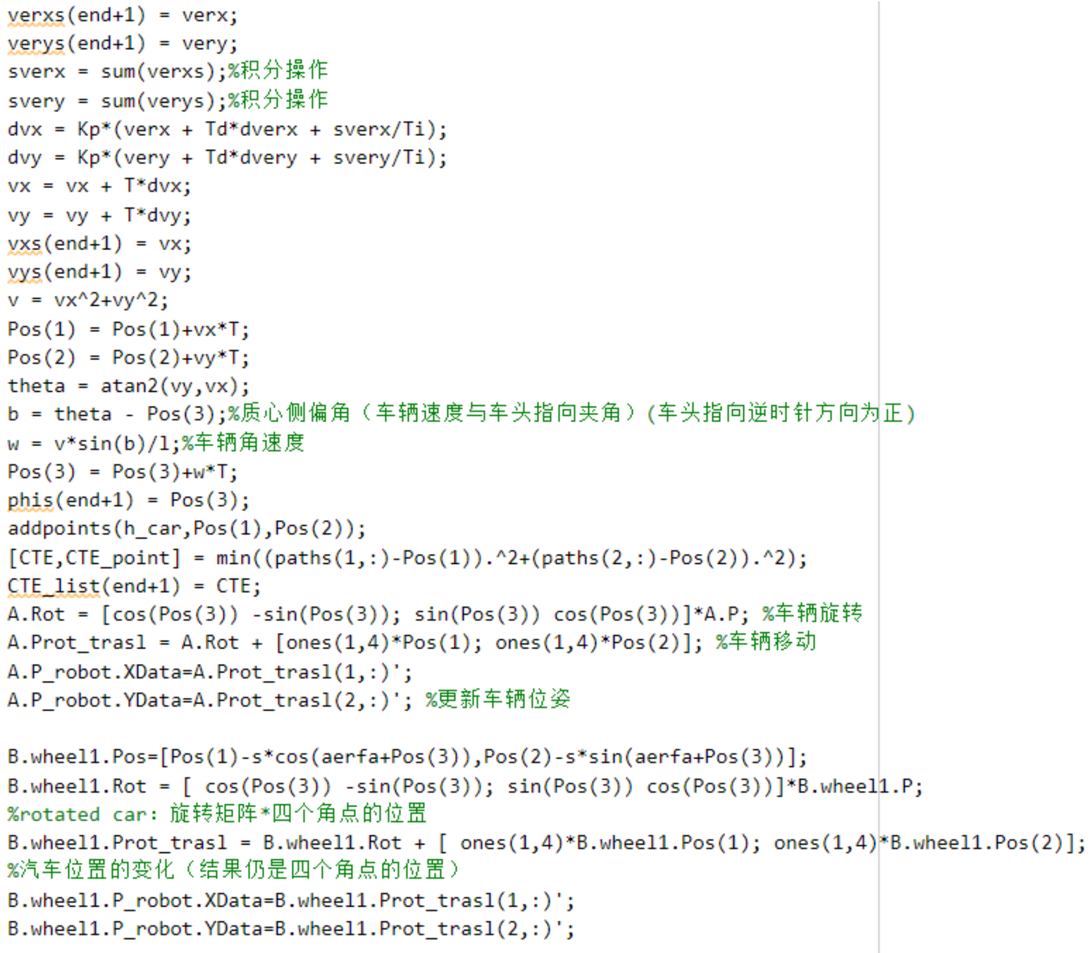
```
% 前轮
B.wheel3.Pos=[Pos(1)+s*cos(aerfa+Pos(3)),Pos(2)+s*sin(aerfa+Pos(3))];
B.wheel3.R_w = 1/4;   %robot width/2
B.wheel3.R_l = 7/12;   % robot length/2
B.wheel3.a1 = [-B.wheel3.R_l -B.wheel3.R_w]';
B.wheel3.b1 = [B.wheel3.R_l -B.wheel3.R_w]';
B.wheel3.b2 = [B.wheel3.R_l B.wheel3.R_w]';
B.wheel3.c = [-B.wheel3.R_l B.wheel3.R_w]';
B.wheel3.P = [B.wheel3.a1 B.wheel3.b1 B.wheel3.b2 B.wheel3.c]; %四个角点的位置
B.wheel3.Rot = [ cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*B.wheel3.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel3.Prot_trasl = B.wheel3.Rot + [ ones(1,4)*B.wheel3.Pos(1); ones(1,4)*B.wheel3.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel3.P_robot=patch(B.wheel3.P(1,:),B.wheel3.P(2,:),'k');
%Patch: 绘制一个填充多边形区域
B.wheel3.P_robot.XData=B.wheel3.Prot_trasl(1,:)';
B.wheel3.P robot.YData=B.wheel3.Prot trasl(2,:)';
```



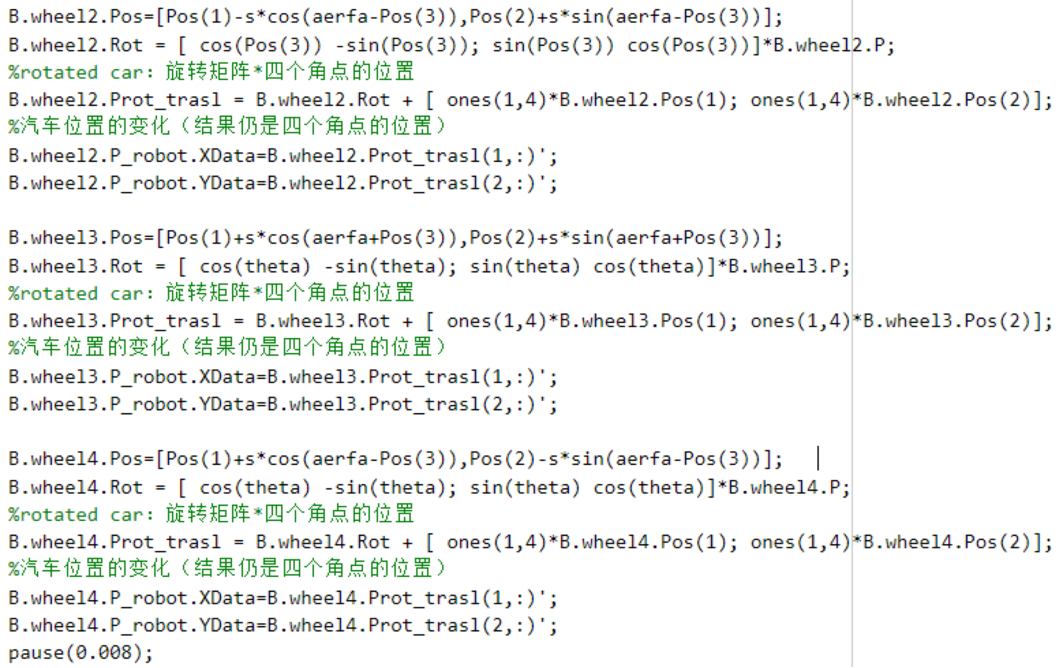
```
B.wheel4.Pos=[Pos(1)+s*cos(aerfa-Pos(3)),Pos(2)-s*sin(aerfa-Pos(3))];
B.wheel4.R_w = 1/4;   %robot width/2
B.wheel4.R_l = 7/12;   % robot length/2
B.wheel4.a1 = [-B.wheel4.R_l -B.wheel4.R_w]';
B.wheel4.b1 = [B.wheel4.R_l -B.wheel4.R_w]';
B.wheel4.b2 = [B.wheel4.R_l B.wheel4.R_w]';
B.wheel4.c = [-B.wheel4.R_l B.wheel4.R_w]';
B.wheel4.P = [B.wheel4.a1 B.wheel4.b1 B.wheel4.b2 B.wheel4.c]; %四个角点的位置
B.wheel4.Rot = [ cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*B.wheel4.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel4.Prot_trasl = B.wheel4.Rot + [ ones(1,4)*B.wheel4.Pos(1); ones(1,4)*B.wheel4.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel4.P_robot=patch(B.wheel4.P(1,:),B.wheel4.P(2,:),'k');
%Patch: 绘制一个填充多边形区域
B.wheel4.P_robot.XData=B.wheel4.Prot_trasl(1,:)';
B.wheel4.P_robot.YData=B.wheel4.Prot_trasl(2,:)';
```



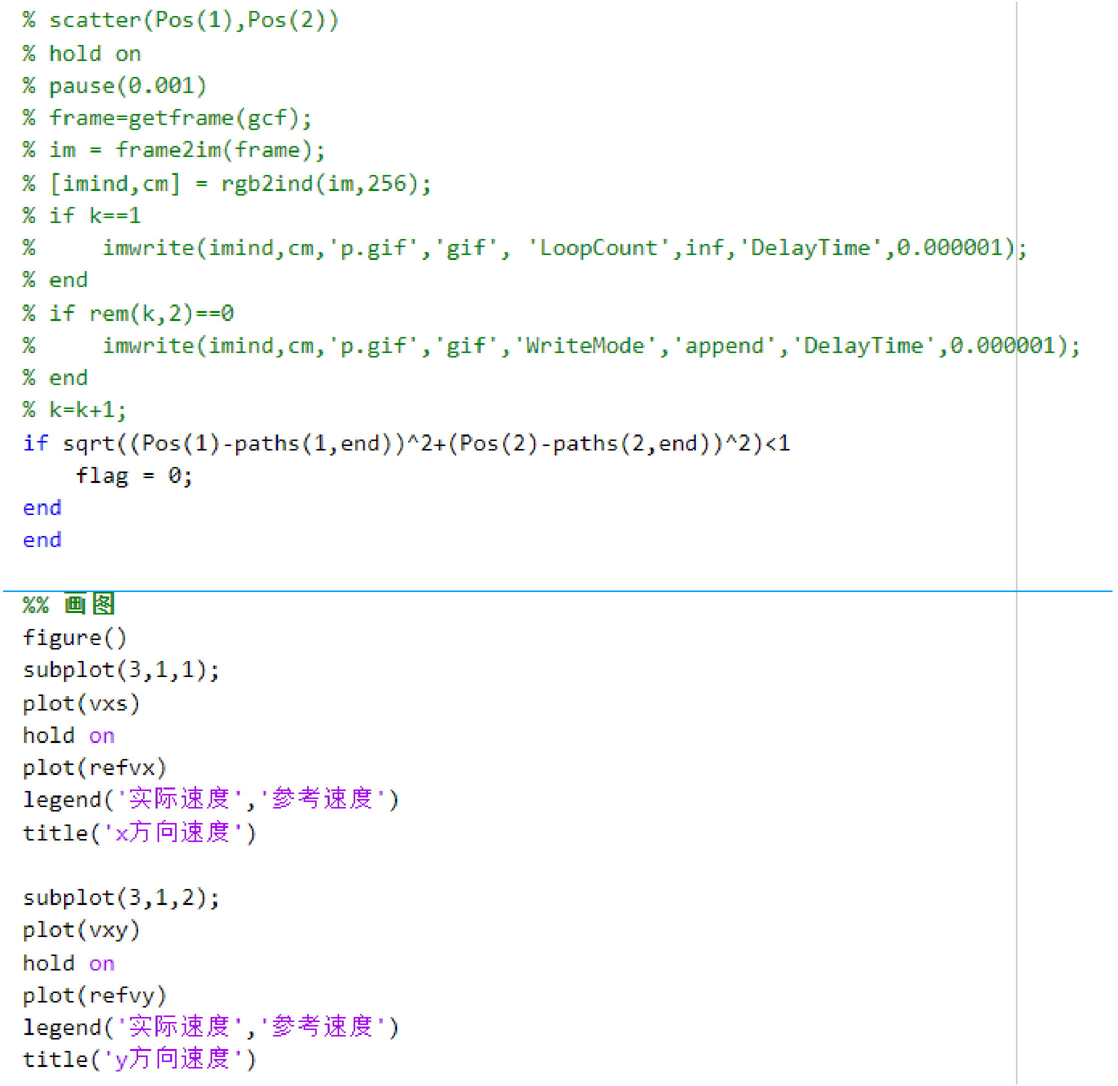
```
%% 载入路径
load('path_node');
vt=15;
paths=getpath(path_node,ones(length(path_node)-1)*vt);
hold on;
plot(paths(1,:),paths(2,:),'--');

%% 寻路
flag = 1;
%初始化车辆参数
vx = 0;
vy = 0;
CTE_list=[];
phis=[];
% scatter(Pos(1),Pos(2))
% hold on
%车辆速度
k=1;
while flag
r = 1+2*v*T;%视界 随v改变而变化
pt = find((paths(1,:)-Pos(1)).^2+(paths(2,:)-Pos(2)).^2-r^2<=0,1,'last');
xt = paths(1,pt);
yt = paths(2,pt);
erx = xt - Pos(1);
ery = yt - Pos(2);
d = sqrt(erx^2+ery^2);
vxt = paths(3,pt);
vyt = paths(4,pt);
vt = sqrt(vxt^2+vyt^2);%目标速度大小
vvxt = erx/d*vt;%车辆需要追踪的x方向速度
vvyt = ery/d*vt;%车辆需要追踪的y方向速度
verx = vvxt-vx;%x方向速度误差
very = vvyt-vy;%y方向速度误差
dverx = verx - verxs(end);%微分操作
dvery = very - verys(end);%微分操作
```



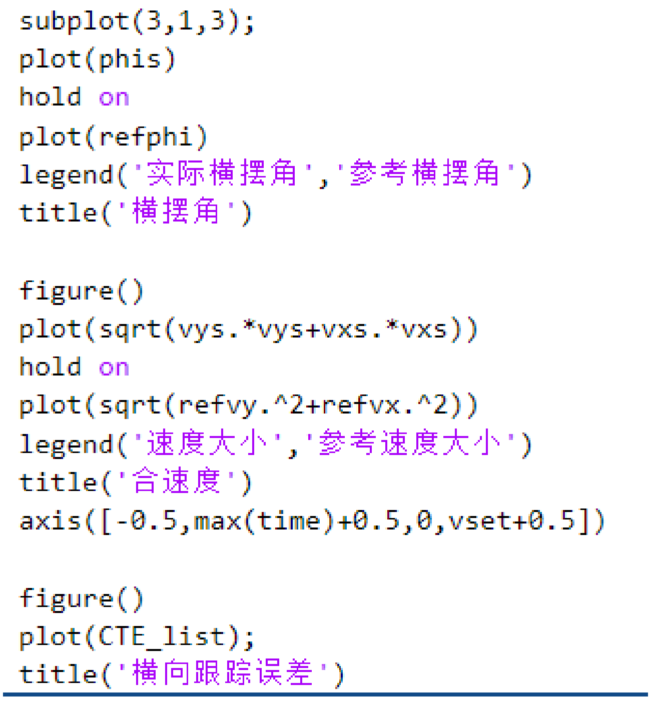
```
verxs(end+1) = verx;
verys(end+1) = very;
sverx = sum(verxs);%积分操作
svery = sum(verys);%积分操作
dvx = Kp*(verx + Td*dverx + sverx/Ti);
dvy = Kp*(very + Td*dvery + svery/Ti);
vx = vx + T*dvx;
vy = vy + T*dvy;
vxs(end+1) = vx;
vys(end+1) = vy;
v = vx^2+vy^2;
Pos(1) = Pos(1)+vx*T;
Pos(2) = Pos(2)+vy*T;
theta = atan2(vy,vx);
b = theta - Pos(3);%质心侧偏角（车辆速度与车头指向夹角）（车头指向逆时针方向为正）
w = v*sin(b)/l;%车辆角速度
Pos(3) = Pos(3)+w*T;
phis(end+1) = Pos(3);
addpoints(h_car,Pos(1),Pos(2));
[CTE,CTE_point] = min((paths(1,:)-Pos(1)).^2+(paths(2,:)-Pos(2)).^2);
CTE_list(end+1) = CTE;
A.Rot = [cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*A.P; %车辆旋转
A.Prot_trasl = A.Rot + [ones(1,4)*Pos(1); ones(1,4)*Pos(2)]; %车辆移动
A.P_robot.XData=A.Prot_trasl(1,:)';
A.P_robot.YData=A.Prot_trasl(2,:)'; %更新车辆位姿

B.wheel1.Pos=[Pos(1)-s*cos(aerfa+Pos(3)),Pos(2)-s*sin(aerfa+Pos(3))];
B.wheel1.Rot = [ cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*B.wheel1.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel1.Prot_trasl = B.wheel1.Rot + [ ones(1,4)*B.wheel1.Pos(1); ones(1,4)*B.wheel1.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel1.P_robot.XData=B.wheel1.Prot_trasl(1,:)';
B.wheel1.P_robot.YData=B.wheel1.Prot_trasl(2,:)';
```



```
B.wheel2.Pos=[Pos(1)-s*cos(aerfa-Pos(3)),Pos(2)+s*sin(aerfa-Pos(3))];
B.wheel2.Rot = [ cos(Pos(3)) -sin(Pos(3)); sin(Pos(3)) cos(Pos(3))]*B.wheel2.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel2.Prot_trasl = B.wheel2.Rot + [ ones(1,4)*B.wheel2.Pos(1); ones(1,4)*B.wheel2.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel2.P_robot.XData=B.wheel2.Prot_trasl(1,:)';
B.wheel2.P_robot.YData=B.wheel2.Prot_trasl(2,:)';

B.wheel3.Pos=[Pos(1)+s*cos(aerfa+Pos(3)),Pos(2)+s*sin(aerfa+Pos(3))];
B.wheel3.Rot = [ cos(theta) -sin(theta); sin(theta) cos(theta)]*B.wheel3.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel3.Prot_trasl = B.wheel3.Rot + [ ones(1,4)*B.wheel3.Pos(1); ones(1,4)*B.wheel3.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel3.P_robot.XData=B.wheel3.Prot_trasl(1,:)';
B.wheel3.P_robot.YData=B.wheel3.Prot_trasl(2,:)';

B.wheel4.Pos=[Pos(1)+s*cos(aerfa-Pos(3)),Pos(2)-s*sin(aerfa-Pos(3))];
B.wheel4.Rot = [ cos(theta) -sin(theta); sin(theta) cos(theta)]*B.wheel4.P;
%rotated car: 旋转矩阵*四个角点的位置
B.wheel4.Prot_trasl = B.wheel4.Rot + [ ones(1,4)*B.wheel4.Pos(1); ones(1,4)*B.wheel4.Pos(2)];
%汽车位置的变化（结果仍是四个角点的位置）
B.wheel4.P_robot.XData=B.wheel4.Prot_trasl(1,:)';
B.wheel4.P_robot.YData=B.wheel4.Prot_trasl(2,:)';
pause(0.008);
```



```
% scatter(Pos(1),Pos(2))
% hold on
% pause(0.001)
% frame=getframe(gcf);
% im = frame2im(frame);
% [imind,cm] = rgb2ind(im,256);
% if k==1
%     imwrite(imind,cm,'p.gif','gif', 'LoopCount',inf,'DelayTime',0.000001);
% end
% if rem(k,2)==0
%     imwrite(imind,cm,'p.gif','gif','WriteMode','append','DelayTime',0.000001);
% end
% k=k+1;
if sqrt((Pos(1)-paths(1,end))^2+(Pos(2)-paths(2,end))^2)<1
    flag = 0;
end
end

%% 画图
figure()
subplot(3,1,1);
plot(vxs)
hold on
plot(refvx)
legend('实际速度','参考速度')
title('x方向速度')

subplot(3,1,2);
plot(vxy)
hold on
plot(refvy)
legend('实际速度','参考速度')
title('y方向速度')
```



```
subplot(3,1,3);
plot(phis)
hold on
plot(refphi)
legend('实际横摆角','参考横摆角')
title('横摆角')

figure()
plot(sqrt(vys.*vys+vxs.*vxs))
hold on
plot(sqrt(refvy.^2+refvx.^2))
legend('速度大小','参考速度大小')
title('合速度')
axis([-0.5,max(time)+0.5,0,vset+0.5])

figure()
plot(CTE_list);
title('横向跟踪误差')
```