Table of Contents

Experiment 2: A* ALGORITHM

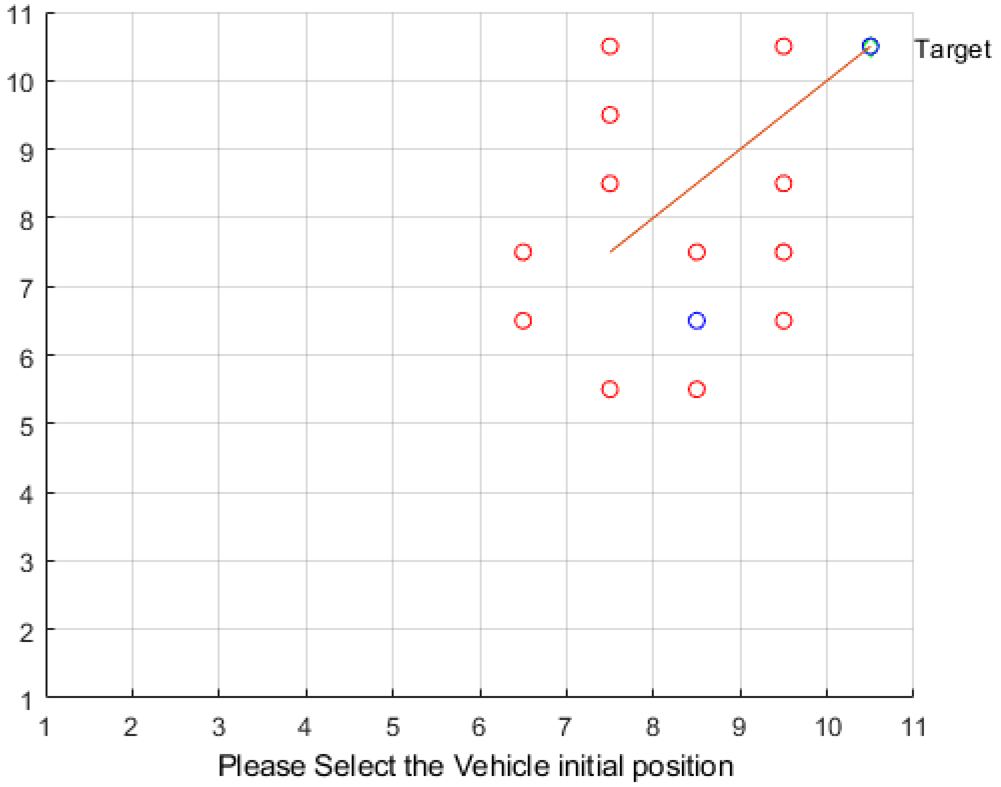
```
% Name: Ventrapragada Sai Shravani
% PRN:17070123120
% Batch: G-5 (E&TC)

% Theory: A* is a graph traversal and path search algorithm, which is
 often
% used in many fields of computer science due to its completeness,
% optimality, and optimal efficiency. One major practical drawback is
 its
% O(b^d) space complexity, as it stores all generated nodes in memory.
```

Code:

```
clc;
clear all;
close all;
%DEFINE THE 2-D MAP ARRAY
MAX_X=10;
MAX_Y=10;
MAX_VAL=10;
%This array stores the coordinates of the map and the
%Objects in each coordinate
MAP=2*(ones(MAX_X,MAX_Y));

% Obtain Obstacle, Target and Robot Position
% Initialize the MAP with input values
% Obstacle=-1,Target = 0,Robot=1,Space=2
j=0;
x_val = 1;
y_val = 1;
axis([1 MAX_X+1 1 MAX_Y+1])
grid on;
hold on;
n=0;%Number of Obstacles

% BEGIN Interactive Obstacle, Target, Start Location selection
pause(1);
h=msgbox('Please Select the Target using the Left Mouse button');
uiwait(h,5);
if ishandle(h) == 1
    delete(h);
```

```
end
xlabel('Please Select the Target using the Left Mouse
 button','Color','black');
but=0;
while (but ~= 1) %Repeat until the Left button is not clicked
    [xval,yval,but]=ginput(1);
end
xval=floor(xval);
yval=floor(yval);
xTarget=xval;%X Coordinate of the Target
yTarget=yval;%Y Coordinate of the Target

MAP(xval,yval)=0;%Initialize MAP with location of the target
plot(xval+.5,yval+.5,'gd');
text(xval+1,yval+.5,'Target')

pause(2);
h=msgbox('Select Obstacles using the Left Mouse button,to select the
 last obstacle use the Right button');
  xlabel('Select Obstacles using the Left Mouse button,to select the
 last obstacle use the Right button','Color','blue');
uiwait(h,10);
if ishandle(h) == 1
    delete(h);
end
while but == 1
    [xval,yval,but] = ginput(1);
    xval=floor(xval);
    yval=floor(yval);
    MAP(xval,yval)=-1;%Put on the closed list as well
    plot(xval+.5,yval+.5,'ro');
 end%End of While loop

pause(1);

h=msgbox('Please Select the Vehicle initial position using the Left
 Mouse button');
uiwait(h,5);
if ishandle(h) == 1
    delete(h);
end
xlabel('Please Select the Vehicle initial position ','Color','black');
but=0;
while (but ~= 1) %Repeat until the Left button is not clicked
    [xval,yval,but]=ginput(1);
    xval=floor(xval);
    yval=floor(yval);
end
xStart=xval;%Starting Position
yStart=yval;%Starting Position
MAP(xval,yval)=1;
 plot(xval+.5,yval+.5,'bo');
%End of obstacle-Target pickup
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
%LISTS USED FOR ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
%OPEN LIST STRUCTURE
%--------------------------------------------------------------------------
%IS ON LIST 1/0 |X val |Y val |Parent X val |Parent Y val |h(n) |g(n)|
f(n)|
%--------------------------------------------------------------------------
OPEN=[];
%CLOSED LIST STRUCTURE
%--------------
%X val | Y val |
%--------------
% CLOSED=zeros(MAX_VAL,2);
CLOSED=[];

%Put all obstacles on the Closed list
k=1;%Dummy counter
for i=1:MAX_X
    for j=1:MAX_Y
        if(MAP(i,j) == -1)
            CLOSED(k,1)=i;
            CLOSED(k,2)=j;
            k=k+1;
        end
    end
end
CLOSED_COUNT=size(CLOSED,1);
%set the starting node as the first node
xNode=xval;
yNode=yval;
OPEN_COUNT=1;
path_cost=0;
goal_distance=distance(xNode,yNode,xTarget,yTarget);
OPEN(OPEN_COUNT,:)=insert_open(xNode,yNode,xNode,yNode,path_cost,goal_distance,goa
OPEN(OPEN_COUNT,1)=0;
CLOSED_COUNT=CLOSED_COUNT+1;
CLOSED(CLOSED_COUNT,1)=xNode;
CLOSED(CLOSED_COUNT,2)=yNode;
NoPath=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% START ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
while((xNode ~= xTarget || yNode ~= yTarget) && NoPath == 1)
%  plot(xNode+.5,yNode+.5,'go');
 exp_array=expand_array(xNode,yNode,path_cost,xTarget,yTarget,CLOSED,MAX_X,MAX_Y);
 exp_count=size(exp_array,1);
 %UPDATE LIST OPEN WITH THE SUCCESSOR NODES
 %OPEN LIST FORMAT
 %--------------------------------------------------------------------------
 %IS ON LIST 1/0 |X val |Y val |Parent X val |Parent Y val |h(n) |
g(n)|f(n)|
 %--------------------------------------------------------------------------
 %EXPANDED ARRAY FORMAT
 %--------------------------------
 %|X val |Y val ||h(n) |g(n)|f(n)|
 %--------------------------------
 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)); %#ok<*SAGROW>
            if OPEN(j,8)== exp_array(i,5)
                %UPDATE PARENTS,gn,hn
                OPEN(j,4)=xNode;
                OPEN(j,5)=yNode;
                OPEN(j,6)=exp_array(i,3);
                OPEN(j,7)=exp_array(i,4);
            end;%End of minimum fn check
            flag=1;
        end;%End of node check
%         if flag == 1
%             break;
    end;%End of j for
    if flag == 0
        OPEN_COUNT = OPEN_COUNT+1;

 OPEN(OPEN_COUNT,:)=insert_open(exp_array(i,1),exp_array(i,2),xNode,yNode,exp_arra
     end;%End of insert new element into the OPEN list
 end;%End of i for
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %END OF WHILE LOOP
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %Find out the node with the smallest fn
  index_min_node = min_fn(OPEN,OPEN_COUNT,xTarget,yTarget);
  if (index_min_node ~= -1)
   %Set xNode and yNode to the node with minimum fn
   xNode=OPEN(index_min_node,2);
   yNode=OPEN(index_min_node,3);
   path_cost=OPEN(index_min_node,6);%Update the cost of reaching the
 parent node
  %Move the Node to list CLOSED
  CLOSED_COUNT=CLOSED_COUNT+1;
  CLOSED(CLOSED_COUNT,1)=xNode;
  CLOSED(CLOSED_COUNT,2)=yNode;
  OPEN(index_min_node,1)=0;
  else
      %No path exists to the Target!!
      NoPath=0;%Exits the loop!
  end;%End of index_min_node check
end;%End of While Loop
%Once algorithm has run The optimal path is generated by starting of
 at the
%last node(if it is the target node) and then identifying its parent
 node
%until it reaches the start node.This is the optimal path

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;
   %Traverse OPEN and determine the parent nodes
   parent_x=OPEN(node_index(OPEN,xval,yval),4);%node_index returns the
 index of the node
   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;
           %Get the grandparents:-)
           inode=node_index(OPEN,parent_x,parent_y);
           parent_x=OPEN(inode,4);%node_index returns the index of the
 node
           parent_y=OPEN(inode,5);
           i=i+1;
    end;
 j=size(Optimal_path,1);
 %Plot the Optimal Path!
 p=plot(Optimal_path(j,1)+.5,Optimal_path(j,2)+.5,'bo');
 j=j-1;
 for i=j:-1:1
  pause(.25);
  set(p,'XData',Optimal_path(i,1)+.5,'YData',Optimal_path(i,2)+.5);
 drawnow ;
 end;
 plot(Optimal_path(:,1)+.5,Optimal_path(:,2)+.5);
else
 pause(1);
 h=msgbox('Sorry, No path exists to the Target!','warn');
 uiwait(h,5);
end
```



Conclusion:

Hence through this algorithm we genenate shortest path by setting a target, a source and creating obstacles.

Published with MATLAB® R2020a