ECE8743 Advanced Robotics
Fall 2021

Milestone Project 1

Improvement of Potential Field Algorithm for Robot Path Planning

Student: Meshaal Mouawad

Student NetID: mm4922

Student E-mail: mm4922@msstate.edu

Instructor: Dr. Chaomin Luo

Department of Electrical and Computer Engineering
Bagley College of Engineering
Mississippi State University

Date: September 13, 2021

Table of Contents

IMPROVEMENT OF POTENTIAL FIELD ALGORITHM FOR ROBOT PATH PLANNING

Meshaal. Mouawad
Department of Electrical and Computer Engineering, Mississippi State University
This technical report is to satisfy the requirement of Advanced Robotics Course, 2021.
Technical Report
Mm4922@msstate.edu

Abstract—**In this project, we improved and implemented the Artificial Potential Field Algorithms (APF) for path planning, then the robot uses the path planning to navigate from the starting position to the goal position with avoiding the obstacles collision. The path planning generated by the APF is an optimal path, shortest distance to the goal. We have investigated three scenarios to create the path planning. Forst scenario, generating path planning with one obstacle. The second scenario, generating path planning with two obstacles. The third scenario, generating the path planning with two obstacles but with different parameters of the spread of attraction, the spread of repulsive, the strength of attraction, and the strength of repulsive, then, the optimal path planning among those parameters founded in scenario 3. We tested two local minima and simulated with APF algorithm then we compare the results for the better path planning. An improvement APF has been tested and simulated for the local minima problem.**

Index Terms: Artificial potential field algorithms; local minima path planning; navigation; Ant Colony Algorithm (ACA); optimal path planning

I. PROJECT OBJECTIVE

The objective of the project is to apply the Artificial Potential Field (APF) algorithm we have learned in the class for robot path planning to improve and resolve some issues such as local minima of an autonomous mobile robot.

II. INTRODUCTION

ROBOTICS and autonomous mobile in order to move and navigate with free collision, it requires path planning such that the robot can move from the starting point to the goal point in a short distant with a condition of avoiding the obstacles, no collision, which is technically called *optimal path planning*. Optimal path planning is one of the most complex problems in robotics as it must meet the above conditions. There have been many methods and algorithms to solve this problem. In this project, we discussed the Artificial Potential Field (APF) algorithm and tested for robot to find the path planning. The Artificial Potential Field depends on the attraction force, and the repulsive force on the robot to find the path planning and navigate. The report is organized in the order, first we explained the Artificial Potential Field (APF) algorithms and we and expressed the formula of *attritive force*, and *repulsive force* then we expressed the *entire potential field.* Second section, the simulation results which we run the functions and formulas of the APF Algorithms and simulated in three scenarios, in scenario-1 the simulation included one obstacle; in scenario-2, the simulation included two obstacles; in scenario-3, the simulation included two obstacles with changing the parameters of the spread of attraction, the spread of repulsive, the strength of attraction, and the strength of repulsive. The purpose of this section is to generate reasonable path in which the path is generated between obstacles (not too close to anyone obstacle). The results of all the scenarios are recorded in this section. Third section, we have used another map and apply APF model to find the path planning. Local minima is the forth section, where we discussed the local minima with APF and simulate two local minima cases and plot the corresponding figures and simulate more cases on local minima and we show our findings in this section. We simulated Mobile Robot Path Planning Using Ant Colony Algorithm (ACA) and Improved Potential Field Method that proposed in [1] which was developed an improved APF algorithm then we show the results in section. A conclusion id drawn in the last section of this report.

III. ARTIFICIAL POTENTIAL FIELD

The Artificial Potential Field (APF) is an attractive and repulsive method between the target, the goal, and the obstacle. The robot is attracted to the target and repulsive from the obstacles. to the target point. Attractive field is radial distribution about the target point and the direction of attraction points to the target point [2]. The APF force F_{APF} is the sum of the attractive force F_{att}, and the repulsive force F_{rep} as described in Equation (1)

$$F_{APF} = F_{att} + F_{rep} \tag{1}$$

a) The Attractive Force

The attractive force, is the force the robot uses to navigate toward the target, and radially distributed from the target as we described mathematically in Equation (2) :

$$F_{att}(X_r) = -K_{att}(|X_r - X_g|)^{\alpha} \tag{2}$$

Meshaal Mouawad, PhD Student, Mississippi State University, Starkville, MS 39759 USA (e-mail: mm4922@msstate.edu).

where K is a constant, $\boldsymbol{X_r}$ represents the robot's current point coordinates, X_g is the target point coordinates, and α is the strength of attractive force which emitted from the target and affects the robot. Equation (2) often visulized as a vector field shuch that :

$$(x, y) \rightarrow (\Delta x, \Delta y) \tag{3}$$

In some cases the vector field is the gredient of the porential field finction in Equation (2):

$$(\Delta x, \Delta y) = \nabla K_{att}(x, y) = \left(\frac{\partial K_{att}}{\partial x}, \frac{\partial K_{att}}{\partial y} \right) \tag{4}$$

The potential attractive force in Equation (2) is partially derived to the x and y-axis. The equation for the potential field can be written as:

$$K_{att} = \frac{1}{2}\alpha((\delta_r - x_g)^2 + (\delta_r - y_g)^2) \tag{5}$$

Where δ_r is the robot position, (x_g, y_g) is the goal position, the target, α is the potential attractive constant.

b) The Repulsive Force

The repulsion force is the force the robot uses to avoid the obstacles. There tow conditions for the repulsive force:

1) When the distance between the robot and the obstacle is greater than the dangerous distance, the repulsion Force is zero, therefore, the robot is moved by the attraction to the target point [1].
2) When the distance between the robot and the obstacle is less than the safety distance, the repulsion force increases rapidly, and its direction is away from the obstacle [1].

The repulsive Force can be represented in Equation (6)

$$F_{rep}(X_r) = \begin{cases} K_{rep} \times \left(\dfrac{D_d}{|X_O - X_r|}\right)^{\beta}, & |X_O - X_r| \le D_d \\ 0\,, & |X_O - X_r| > D_d \end{cases} \tag{6}$$

Where the K_{rep} is the strength of the repulsive field, D_d represents dangerous distance, normally is double the robot's diameter D_r, X_O is the obstacle coordinate, and β is the strength of the repulsive force. The potential attractive in Equation (6) is partially derived to the x and y-axis, therefore the equation can be written as:

$$F_{rep} = \begin{cases} K_{rep} \times \left(\dfrac{1}{\rho} - \dfrac{1}{\rho_O}\right)\dfrac{1}{\rho^2}\dfrac{\partial \rho}{\partial x}, & if\ \rho \le \rho_O \\ 0\,, & if\ \rho > \rho_O \end{cases} \tag{7}$$

Where the ρ_O is the dangerous distant, the closest distance between the robot and the obstacle, and ρ is the distant to the obstacle ρ_O

$$\rho_O = \sqrt{\Delta x_{rg}^2 - \Delta y_{rg}^2} \tag{8}$$

where Δx_{rg} , Δy_{rg} is the difference of the distance between the robot and the obstacle on the x-axis, as well for the y-axis, therefore, the equation is:

$$x_{rg} = \delta_r - x_O \tag{9}$$

$$y_{rg} = \delta_r - y_O \tag{10}$$

c) The Entire potential field.

The entire potential force filed is the magnitude of the attractive force and repulsive force with respect to the direction. The function for the entire potential field is in Equation (11):

$$F_{sum}(X_r) = F_{attr}(X_r) + F_{rep}(X_r) \tag{11}$$

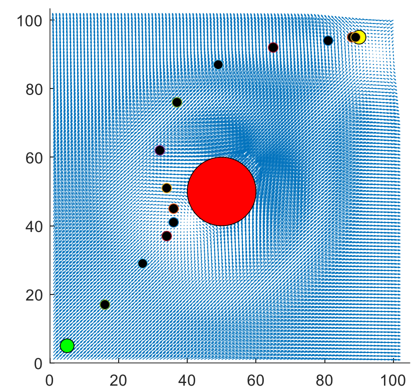

Fig. 1. The first scenario with one obstacle and the generated path planning from the user screen on MATLAB.

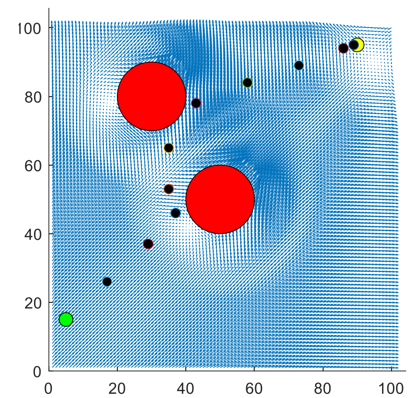

Fig. 2. The second scenario with two obstacles and the generated path planning from the user screen on MATLAB.

IV. Simulation Results

The path planning in this project was invistegated by changing the value of the parameters of the Artificial Potential Field algorithms such that a more reasnable path is generated not too colse to anyone of the obstacels. The approach was tested through simulations. The simulation enviroment and the APF algorithm were developed in MATLAB. From a simulation perspective, three secenairos the first part of the simulation was to create a scenario with one obstacle and generate a path planning using the Artificial Potential Field (APF) algorithm. The second part of the simulation was to create a scenario with two obstacles and generate a path planning. the robot was graphically displayed at the user screen as it shows in Fig. 1, and Fig. 2. The third part of the simulation was to create a scenario with two obstacles with changing the parameters of the spread of attraction, the spread of repulsive, the strength of attraction, the strength of repulsive with two obstacles and the generated path planning as planed. The result of the thrid scienario is showing in Fig. 3-7.

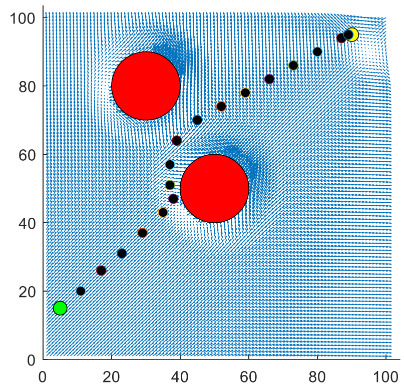

Fig. 3. The third scenario where the spread of attraction = 5%, the spread of repulsive = 5%, the strength of attraction = 0.8%, the strength of repulsive = 0.9% with two obstacles and the generated path planning from the user screen on MATLAB.

1) Scenario two

In scenario 2, there are two obstacles with the following parameters:

- the spread of attraction = 20%
- the spread of repulsive = 20%
- the strength of attraction = 0.8%
- the strength of repulsive = 0.9%

it shows the path planning is close to the Obstacle on the right. The scenario in included in Table. 1.

2) Scenario three

In scenario 3, there two obstacles similar to scenario 1, we have changed the parameters as it shows in Table. 1. Each simulation result of changing the parameters are shown in Fig. 4 – Fig. 5.

Our investigation shows that the scenario in Fig. 6 avoid both of the obstacles, therefore, it would be the optimal path planning among the other scenarios. However, the time complexity of the function in longer. The parameters for this scenario are:

- the spread of attraction = 10%
- the spread of repulsive = 10%
- the strength of attraction = 0.6%
- the strength of repulsive = 2%

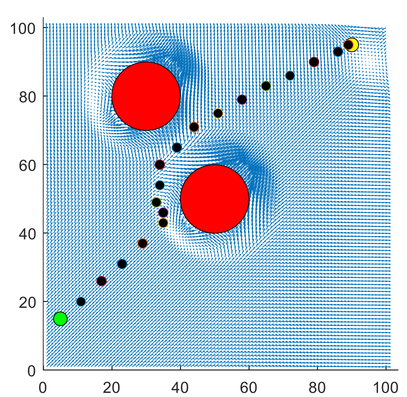

Fig. 4. The third scenario where the spread of attraction = 10%, the spread of repulsive = 10%, the strength of attraction = 0.8%, the strength of repulsive = 1.9% with two obstacles and the generated path planning from the user screen on MATLAB.

TABLE I
THE TYPICAL PARAMETER VALUES

Parameters	**Variable**	**Value**	**Time Complexity**
The spread of attraction	goalS	5.0 %	
The spread of repulsion	obsS	5.0 %	
Strength of attraction	Alpha	0.8 %	
Strength of repulsion	Beta	0.9 %	0.000197 s
Adjusted Parameters. Fig. 3			
Parameters	**Variables**	**Value**	**Time Complexity**
The spread of attraction	goalS	10 %	
The spread of repulsion	obsS	10 %	
Strength of attraction	Alpha	0.8 %	
Strength of repulsion	Beta	0.9 %	0.000181 s
Adjusted Parameters. Fig. 4			
Parameters	**Variables**	**Value**	**Time Complexity**
The spread of attraction	goalS	10 %	
The spread of repulsion	obsS	10 %	
Strength of attraction	Alpha	0.8 %	
Strength of repulsion	Beta	1.9 %	0.000218 s
Adjusted Parameters. Fig. 5			
Parameters	**Variables**	**Value**	**Time Complexity**
The spread of attraction	goalS	10 %	
The spread of repulsion	obsS	10 %	
Strength of attraction	Alpha	0.8 %	
Strength of repulsion	Beta	2.0 %	0.000197 s
Adjusted Parameters. Fig. 6			
Parameters	**Variables**	**Value**	**Time Complexity**
The spread of attraction	goalS	10 %	
The spread of repulsion	obsS	10 %	
Strength of attraction	Alpha	0.6 %	
Strength of repulsion	Beta	2.0 %	0.000486 s

S is the time in seconds. The time complexity is the Elapsed Time which MATLAB® reads the internal time at the execution of the toc function and displays the elapsed time since the most recent call to the tic function without an output.

The elapsed time is expressed in seconds.

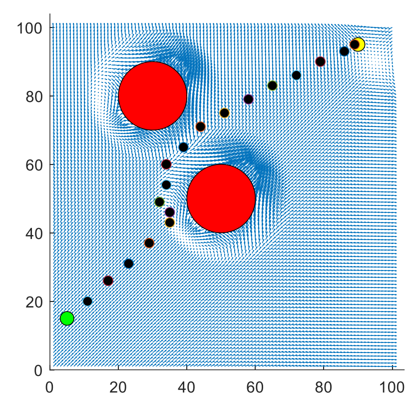

Fig. 5. The third scenario where the spread of attraction = 10%, the spread of repulsive = 10%, the strength of attraction = 0.8%, the strength of repulsive = 2% with two obstacles and the generated path planning from the user screen on MATLAB.

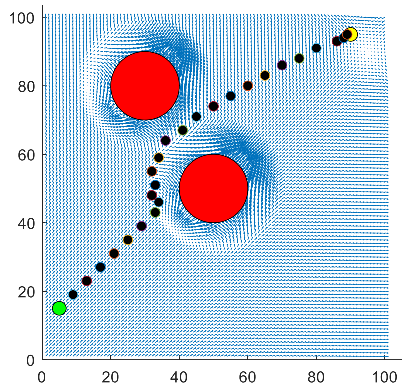

Fig. 6. The third scenario where the spread of attraction = 10%, the spread of repulsive = 10%, the strength of attraction = 0.6%, the strength of repulsive = 2% with two obstacles and the generated path planning from the user screen on MATLAB.

3) Scenario Four

We have implemented another map with 8 obstacles and simulation showing in Fig. 7

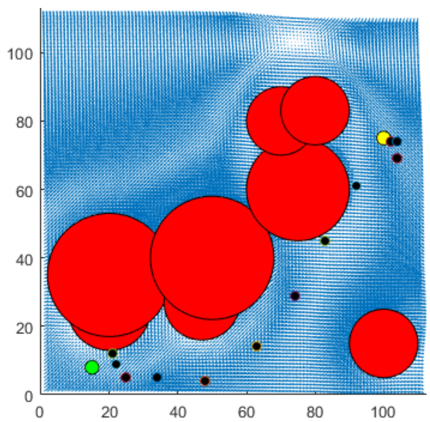

Fig. 7. Simulation of additional map for scenario-4 with 8 Obstacles.

V. Local Minima

Problem statement: In the APF algorithm, the local minima is a problem where where the area of the attractive potential field F_{att} and the repulsive potential field F_{rep} are both have the same value which the resulatant force cancels each other leading to the value equal to zero of the artificial potential field F_{APF} which will make the robot not to be attractive nor replusive (stopped moving). Fig. 11 shows the case of two kinds of gird traps. Fig. 11(a), the attraction and repulsive force are all along the horizontal direction, and the repulsive force is greater than the attracting force. therefore, the resultant force, the net force is opposite to the goal, and the robot will fall into a dead-end cycle of transverse reciprocating motion [1]. In Fig. 11(b), the robot is under the attraction force in the upper right of 45° and the repulsive force in the lower right of 45°, and the repulsion is greater than the attraction. therefore, the robot will move away from the obstacle and the goal point [1]. A smulation of this case is shown in Fig. 10. The local minima in Fig. 10, stopped the robot from moving.

The author in [1], proposed improvement to APF by employing the global environment information of the grid map. The proposed improvement is in the attractive force and the resultant force.

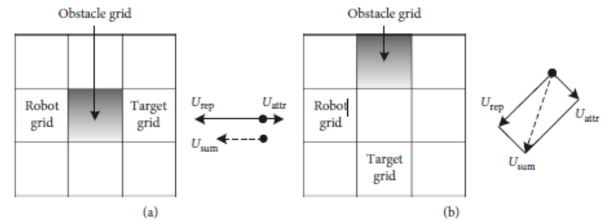

Fig. 11. Environment with local minima.: Two kinds of grid traps. (a) the obstacle grid is between the robot grid and target grid. (b) the obstacle grid is not between the robot grid and target grid. [1]

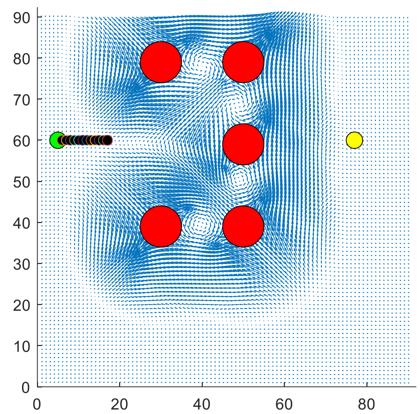

Fig. 10 local minima simulation result.

a) Improving the Attractive Function

For the grid rap shown in Fig. 11, the attractive field distribution opposite to that of the conventional potential field is used, which is described in Equation (12)

$$U_{att}(X_r)=\begin{cases} K\times\left(\dfrac{S}{|X_r-X_g|}\right)^{\alpha}, & if\ |X_r-X_g|<S \\ S\ , & if\ |X_r-X_g|\geq S \end{cases} \quad (12)$$

where S is the threshold of the attractive distance. S is taken to 1/10 of the grid map length and width. If the value of S is 3 and the coordinates of the target grid $\mathbf{X}_g$ is (c, d), then attractive area R_{att} is:

$$R_{att}=\{(i,j)\,|\,i\geq c-3,\ i\leq c+3,\ j\geq d-3,\ j\leq d+3\} \quad (13)$$

where (i, j) are the coordinates of those grids within the attractive distance.

b) Improving the Resultant Force of APF

To transfer directions in the grid map, there are only eight cases for the robot, the resultant force has eight standard directions. The direction of vertically upward is set to 0° and clockwise is the positive direction. the angular distribution of eight standard directions in the grid map is shown in Fig. 12.

315°	0°	45°
270°	Robot	90°
225°	180°	135°

Fig. 12. the angle of eight grid neighbors of robot. As explained in improving the attraction force.

the angle difference between the resultant force direction and the eight standard directions is calculated respectively.
the standard direction with the smallest absolute value of the angle difference is the direction of the resultant force. If the minimum absolute value of the angle difference has two or more standard directions, then the smaller standard direction is adopted as the resultant force direction.

c) Additional Force

When the direction of the robot's resultant force is opposite to the direction of the robot's attraction, the robot will fall into an endless loop. To overcome this, an additional force perpendicular to the resultant force is applied, then the robot will be forced out of the endless loop as it shown in Fig. 13.

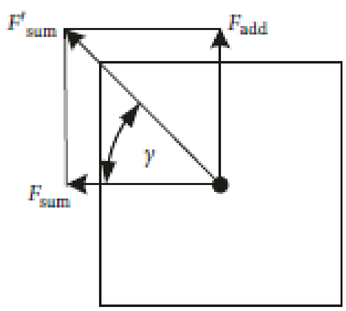

Fig. 13. The additional force proposed by [1]

In Fig. 13, F_{sum} is the original resultant force, F_{add} is the additional force, F'_{sum} is the new resultant force, and γ is the angle between the new resultant force and the original resultant force. In order for the robot to move from the loop and overcome the problem there must be $\gamma > 22°$ and $F_{add} > \tan 22.5° \times F_{sum}$. Fig. 14, shows the improvement of the APF.

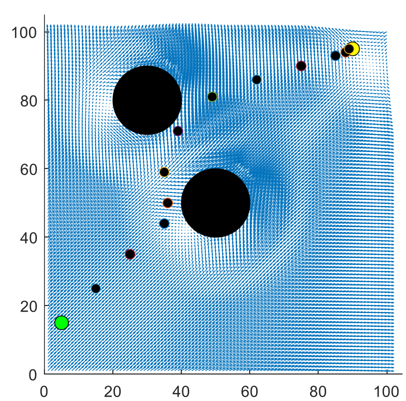

Fig. 14. The improved module of the APF

VI. Conclusion

We have shown the benefits of the Artificial Potential Field Algorithms (APF) for path planning. The path planning was generated by the APF with the shortest distance to the goal. We have tested three scenarios to create the path planning. First scenario, generating path planning with one obstacle. The second scenario, generating path planning with two obstacles. The third scenario, generating the path planning with two obstacles but with different parameters of the spread of attraction, the spread of repulsive, the strength of attraction, and the strength of repulsive, then, the optimal path planning among those parameters founded in scenario 3. We tested two local minima and simulated with APF algorithm then we compare the results for the better path planning. Finally, we showed the proposed improvement of the Artificial Potential Field algorithm for local minima and the simulation of that improvement.

VII. References

[1] G. C. a. J. Liu, "Mobile Robot Path Planning Using Ant Colony Algorithm and Improved Potential Field Method," *Computational Intelligence and Neuroscience,* vol. Volume 2019, 2019.

[2] A. M. a. Iswanto, Oyas Wahyunggoro, Adha Imam Cahyadi, "Artificial Potential Field Algorithm Implementation for Quadrotor Path Planning," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 10, 2019.

VIII. Appendix

Here is the code for the simulation which wascompleted on MATLAB.

File #1 ECE8743_PotentialFields_Obstacle_1.m
This file for the simulation of one obatacles

```
%%% Southfield,Michigan
%%% May 23, 2016
%%% Potential Fields for Robot Path
Planning
%
%
% Initially proposed for real-time
collision avoidance [Khatib 1986].
% Hundreds of papers published on APF
% A potential field is a scalar function
over the free space.
% To navigate, the robot applies a force
proportional to the
% negated gradient of the potential
field.
% A navigation function is an ideal
potential field

clc
close all
clear
%% Defining environment variables
startPos = [5,5];    % robot Start Point
goalPos = [90, 95];  % target coordinate
obs1Pos = [50, 50];  % the obstacle
coordinate
obsRad = 10;         % the obstacle
radius
goalR = 0.2; % The radius of the goal
goalS = 20;  % The spread of attraction
of the goal
obsS = 30;   % The spread of repulsion of
the obstacle
alpha = 0.8; % Strength of attraction
beta = 0.6;  % Strength of repulsion
%% Carry out the Potential Field Math as
follows:
u = zeros(100, 100);
v = zeros(100, 100);
testu = zeros(100, 100);
testv = zeros(100, 100);
for x = 1:1:100
    for y = 1:1:100
        [uG, vG] = GoalDelta(x, y,
goalPos(1), goalPos(2), goalR, goalS,
alpha);
        [uO, vO] = ObsDelta(x, y,
obs1Pos(2), obs1Pos(1), obsRad, obsS,
beta);
        xnet = uG + uO; % the resultant
on x
        ynet = vG + vO; % the resultant
on y
        vspeed = sqrt(xnet^2 + ynet^2);
% the speed
        theta = atan2(ynet,xnet);
        u(x,y) = vspeed*cos(theta);
        v(x,y) = vspeed*sin(theta);
%          hold on

    end
end
%%
[X,Y] = meshgrid(1:1:100,1:1:100);
figure
quiver(X, Y, u, v, 3)



%% Defining the grid

% Plotting the obstacles
circles(obs1Pos(1),obs1Pos(2),obsRad,
'facecolor','red')
axis square

hold on % Plotting start position
circles(startPos(1),startPos(2),2,
'facecolor','green')

hold on % Plotting goal position
circles(goalPos(1),goalPos(2),2,
'facecolor','yellow')

%% Priting of the path
currentPos = startPos;
x = 0;

while sqrt((goalPos(1)-currentPos(1))^2 +
(goalPos(2)-currentPos(2))^2) > 1
    tempPos = currentPos +
[u(currentPos(1),currentPos(2)),
v(currentPos(1),currentPos(2))]
    currentPos = round(tempPos)
    hold on
    plot(currentPos(1),currentPos(2),'-
o', 'MarkerFaceColor', 'black')
    pause(0.5)
end
```

File #2 ECE8743_PotentialFields_Obstacles_2.m
This file for the simulation of 2 obatacles

```
%%% Southfield,Michigan
%%% May 23, 2016
%%% Potential Fields for Robot Path
Planning
%
%
% Initially proposed for real-time
collision avoidance [Khatib 1986].
% Hundreds of papers published on APF
```

```
% A potential field is a scalar function
over the free space.
% To navigate, the robot applies a force
proportional to the
% negated gradient of the potential
field.
% A navigation function is an ideal
potential field

clc
close all
clear
%% Defining environment variables
startPos = [5,15];
goalPos = [90, 95];
obs1Pos = [50, 50];
obs2Pos = [30, 80];
obs3Pos = [20,25];
obs4Pos= [25,35];
obs5Pos = [47,27];
obs6Pos = [50,40] ;
obs7Pos = [75,50] ;
obs8Pos = [70,65];
obs9Pos = [80,70];
obs10Pos = [90,15];
obsRad = 10;
goalR = 0.2; % The radius of the goal
goalS = 10;  % The spread of attraction
of the goal
obsS = 10;   % The spread of repulsion of
the obstacle
alpha = 0.8; % Strength of attraction
beta = 2;  % Strength of repulsion


%% To perform the Potential Field Math as
follows:
u = zeros(100, 100);
v = zeros(100, 100);
testu = zeros(100, 100);
testv = zeros(100, 100);

for x = 1:1:100
    for y = 1:1:100
        [uG, vG] = GoalDelta(x, y,
goalPos(1), goalPos(2), goalR, goalS,
alpha);
        [uO, vO] = ObsDelta(x, y,
obs1Pos(2), obs1Pos(1), obsRad, obsS,
beta);
        [uO2, vO2] = ObsDelta(x, y,
obs2Pos(2), obs2Pos(1), obsRad, obsS,
beta);
        xnet = uG + uO + uO2; % the
resultant on x
        ynet = vG + vO + vO2; % the
resultant on y
        vspeed = sqrt(xnet^2 + ynet^2);
        theta = atan2(ynet,xnet);
        u(x,y) = vspeed*cos(theta);
        v(x,y) = vspeed*sin(theta);
%          hold on

    end
end
%%
[X,Y] = meshgrid(1:1:100,1:1:100);
figure
quiver(X, Y, u, v, 3)


%% Defining the grid

% Plotting the obstacles
circles(obs1Pos(1),obs1Pos(2),obsRad,
'facecolor','red')
axis square

hold on
circles(obs2Pos(1),obs2Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs3Pos(1),obs3Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs4Pos(1),obs4Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs5Pos(1),obs5Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs6Pos(1),obs6Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs7Pos(1),obs7Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs8Pos(1),obs8Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs9Pos(1),obs9Pos(2),obsRad,
'facecolor','red')
hold on
circles(obs10Pos(1),obs10Pos(2),obsRad,
'facecolor','red')

hold on % Plotting start position
circles(startPos(1),startPos(2),2,
'facecolor','green')

hold on % Plotting goal position
circles(goalPos(1),goalPos(2),2,
'facecolor','yellow')

%% Priting of the path
currentPos = startPos;
x = 0;
```

```
while sqrt((goalPos(1)-currentPos(1))^2 +
(goalPos(2)-currentPos(2))^2) > 1
    tempPos = currentPos +
[u(currentPos(1),currentPos(2)),
v(currentPos(1),currentPos(2))]
    currentPos = round(tempPos)
    hold on
    plot(currentPos(1),currentPos(2),'o',
'MarkerFaceColor', 'black')
    pause(0.5)
end
tic
   % The program section to time.
toc
```

file: section_5.m
this is the code for the 8 obstacles.

```
%%% Southfield,Michigan
%%% May 23, 2016
%%% Potential Fields for Robot Path
Planning
%
%
% Initially proposed for real-time
collision avoidance [Khatib 1986].
% Hundreds of papers published on APF
% A potential field is a scalar function
over the free space.
% To navigate, the robot applies a force
proportional to the
% negated gradient of the potential
field.
% A navigation function is an ideal
potential field
%
% code was edited by Meshaal Mouawad
09/13/2021
% email: mm4922@msstate.edu
% Mississippi State University
% ECE Department

clc
close all
clear
%% Defining environment variables
startPos = [15,8];
goalPos = [100, 75];
obs1Pos = [20, 25];
obs2Pos = [20, 35];
obs3Pos = [47, 27];
obs4Pos = [50, 40];
obs5Pos = [100, 15];
obs6Pos = [75, 60];
obs7Pos = [70, 80];
obs8Pos = [80, 83];

goalR = 0.2;                % The radius of
the goal
goalS = 20;                 % The spread of
attraction of the goal
obsS = 15 ;                 % The spread of
repulsion of the obstacle
alpha = 0.9;                % Strength of
attraction
beta = 0.5;                 % Strength of
repulsion
map_s = 110;
%% To perform the Potential Field Math as
follows:
u = zeros(map_s, map_s);
v = zeros(map_s, map_s);
testu = zeros(map_s, map_s);
testv = zeros(map_s, map_s);


for x = 1:1:map_s
    for y = 1:1:map_s  % tells MATLAB to
create a vector of values from 1 to 100,
with a spacing of 1. Similarly,
        [uG, vG] = GoalDelta(x, y,
goalPos(1), goalPos(2), goalR, goalS,
alpha); % Calculates the Attractive force
cause by the Goal

        [uO, vO] = ObsDelta(x, y,
obs1Pos(2), obs1Pos(1), 12, obsS, beta);
% find the Repulsive force
        [uO2, vO2] = ObsDelta(x, y,
obs2Pos(2), obs2Pos(1), 18, obsS, beta);%
find the Repulsive force by the second
obstacle
        [uO3, vO3] = ObsDelta(x, y,
obs3Pos(2), obs3Pos(1), 11, obsS, beta);%
find the Repulsive force by the third
obstacle
        [uO4, vO4] = ObsDelta(x, y,
obs4Pos(2), obs4Pos(1), 18, obsS, beta);%
find the Repulsive force by forth
obstacle
        [uO5, vO5] = ObsDelta(x, y,
obs5Pos(2), obs5Pos(1), 10, obsS, beta);%
find the Repulsive force by the fifth
obstacle
        [uO6, vO6] = ObsDelta(x, y,
obs6Pos(2), obs6Pos(1), 15, obsS, beta);%
Calculate the Repulsive force by the
sixth obstacle
        [uO7, vO7] = ObsDelta(x, y,
obs7Pos(2), obs7Pos(1), 10, obsS, beta);%
Calculate the Repulsive force by the
seventh obstacle
        [uO8, vO8] = ObsDelta(x, y,
obs8Pos(2), obs8Pos(1), 10, obsS, beta);%
Calculate the Repulsive force by the
eigth obstacle

        xnet = uG + uO + uO2+ uO3 +  uO4
+  uO5 +  uO6  +  uO7 +  uO8;
        ynet = vG + vO + vO2+  vO3 +
vO4+  vO5 +  vO6 +  vO7 +  vO8;
```

```
        vspeed = sqrt(xnet^2 + ynet^2);
        theta = atan2(ynet,xnet);
        u(x,y) = vspeed*cos(theta);
        v(x,y) = vspeed*sin(theta);

    end
end
%%
[X,Y] = meshgrid(1:1:map_s,1:1:map_s);
figure
quiver(X, Y, u, v, 3)

%% Defining the grid
% Plotting the obstacles
circles(obs1Pos(1),obs1Pos(2),12,
'facecolor','red')
axis square

hold on
circles(obs2Pos(1),obs2Pos(2),18,
'facecolor','red')
hold on
circles(obs3Pos(1),obs3Pos(2),11,
'facecolor','red')
hold on
circles(obs4Pos(1),obs4Pos(2),18,
'facecolor','red')
hold on
circles(obs5Pos(1),obs5Pos(2),10,
'facecolor','red')
hold on
circles(obs6Pos(1),obs6Pos(2),15,
'facecolor','red')
hold on
circles(obs7Pos(1),obs7Pos(2),10,
'facecolor','red')
hold on
circles(obs8Pos(1),obs8Pos(2),10,
'facecolor','red')
hold on % Plotting start position
circles(startPos(1),startPos(2),2,
'facecolor','green')

hold on % Plotting goal position
circles(goalPos(1),goalPos(2),2,
'facecolor','yellow')



%% Priting of the path
currentPos = startPos;
x = 0;



while sqrt((goalPos(1)-currentPos(1))^2 +
(goalPos(2)-currentPos(2))^2) > 1
    tempPos = currentPos +
[u(currentPos(1),currentPos(2)),
v(currentPos(1),currentPos(2))]
    currentPos = round(tempPos)
    hold on
    plot(currentPos(1),currentPos(2),'o',
'MarkerFaceColor', 'black')
    pause(0.5)
end


tempPos
```

---

file: section_6.m
this file has the code for the local minima.

```
%%% Southfield,Michigan
%%% May 23, 2016
%%% Potential Fields for Robot Path
Planning
%
%
% Initially proposed for real-time
collision avoidance [Khatib 1986].
% Hundreds of papers published on APF
% A potential field is a scalar function
over the free space.
% To navigate, the robot applies a force
proportional to the
% negated gradient of the potential
field.
% A navigation function is an ideal
potential field
%
% code was edited by Meshaal Mouawad
09/13/2021
% email: mm4922@msstate.edu
% Mississippi State University
% ECE Department

clc
close all
clear
%% Defining environment variables
startPos = [5,60];
goalPos = [77, 60];
obsRad = 5;          % obstical radius
obs1Pos = [30, 39];
obs2Pos = [50, 39];
obs3Pos = [50, 59];
obs4Pos = [50, 79];
obs5Pos = [30, 79];

%% Initial Parameters

goalR = 0.2;         % The radius of the
goal
goalS = 10           % The spread of
attraction of the goal
```

```
obsS = 20;           % The spread of
repulsion of the obstacle
alpha = 0.1          % Strength of
attraction
beta = 0.7;          % Strength of
repulsion


%% To perform the Potential Field Math as
follows:
u = zeros(90, 90);
v = zeros(90, 90);
testu = zeros(90, 90); % can not seem to
find their prupose
testv = zeros(90, 90);% can not see to
find their purpose

% Responsible for completing the
calculation need to properly run the
% neural activity.
for x = 1:1:90
    for y = 1:1:90
        [uG, vG] = GoalDelta(x, y,
goalPos(1), goalPos(2), goalR, goalS,
alpha); % Calculates the Attractive force
cause by the Goal
        [uO, vO] = ObsDelta(x, y,
obs1Pos(2), obs1Pos(1), obsRad, obsS,
beta);   % Calculates the Repulsive force
by 1st obstacle
        [uO2, vO2] = ObsDelta(x, y,
obs2Pos(2), obs2Pos(1), obsRad, obsS,
beta); % Calculate the Repulsive force by
the 2nd obstacle
        [uO3, vO3] = ObsDelta(x, y,
obs3Pos(2), obs3Pos(1), obsRad, obsS,
beta); % Calculate the Repulsive force by
the 3rd obstacle
        [uO4, vO4] = ObsDelta(x, y,
obs4Pos(2), obs4Pos(1), obsRad, obsS,
beta); % Calculate the Repulsive force by
the 4th obstacle
        [uO5, vO5] = ObsDelta(x, y,
obs5Pos(2), obs5Pos(1), obsRad, obsS,
beta); % Calculate the Repulsive force by
the 5th obstacle
%
        xnet = uG + uO + uO2 + uO3 + uO4
+ uO5 ;
        ynet = vG + vO + vO2 + vO3 + vO4
+ vO5  ;
        vspeed = sqrt(xnet^2 + ynet^2);
        theta = atan2(ynet,xnet);
        u(x,y) = vspeed*cos(theta);
        v(x,y) = vspeed*sin(theta);
%         hold on

    end
end
%%
[X,Y] = meshgrid(1:1:90,1:1:90);
figure
quiver(X, Y, u, v, 3) % plots th
directional arrows seen in the graph


%% Defining the grid
% Plotting the Circle obstacles

circles(obs1Pos(1),obs1Pos(2),obsRad,
'facecolor','red')
axis square

hold on
circles(obs2Pos(1),obs2Pos(2),obsRad,
'facecolor','red')

hold on
circles(obs3Pos(1),obs3Pos(2),obsRad,
'facecolor','red')

hold on
circles(obs4Pos(1),obs4Pos(2),obsRad,
'facecolor','red')

hold on
circles(obs5Pos(1),obs5Pos(2),obsRad,
'facecolor','red')

hold on % Plotting start position
circles(startPos(1),startPos(2),2,
'facecolor','green')

hold on % Plotting goal position
circles(goalPos(1),goalPos(2),2,
'facecolor','yellow')

%% Priting of the path
currentPos = startPos;
x = 0;

while sqrt((goalPos(1)-currentPos(1))^2 +
(goalPos(2)-currentPos(2))^2) > 1
    tempPos = currentPos +
[u(currentPos(1),currentPos(2)),
v(currentPos(1),currentPos(2))]
    currentPos = round(tempPos)
    hold on
    plot(currentPos(1),currentPos(2),'o',
'MarkerFaceColor', 'black')
    pause(0.5)
end
```

file: section_7.m
in section 7 folder there is the simualtion of the improved APF

```
%%% Southfield,Michigan
%%% May 23, 2016
%%% Potential Fields for Robot Path
Planning
%
%
% Initially proposed for real-time
collision avoidance [Khatib 1986].
% Hundreds of papers published on APF
% A potential field is a scalar function
over the free space.
% To navigate, the robot applies a force
proportional to the
% negated gradient of the potential
field.
% A navigation function is an ideal
potential field

clc
close all
clear
%% Defining environment variables
startPos = [5,15]; % robot Start Point
goalPos = [90, 95]; % target location
obs1Pos = [50, 50];% the obstacle
coordinate
obs2Pos = [30, 80];
obsRad = 10; % obstical radius
goalR = 0.2; % The radius of the goal
goalS = 20;  % The spread of attraction
of the goal
obsS = 20;   % The spread of repulsion of
the obstacle
alpha = 0.7; % Strength of attraction
beta = 0.80;  % Strength of repulsion
%% To perform the Potential Field Math as
follows:
u = zeros(100, 100);
v = zeros(100, 100);
testu = zeros(100, 100);
testv = zeros(100, 100);

for x = 1:1:100
    for y = 1:1:100
        [uG, vG] = GoalDelta(x, y,
goalPos(1), goalPos(2), goalR, goalS,
alpha);
        [uO, vO] = ObsDelta(x, y,
obs1Pos(2), obs1Pos(1), obsRad, obsS,
beta);
        [uO2, vO2] = ObsDelta(x, y,
obs2Pos(2), obs2Pos(1), obsRad, obsS,
beta);
% the resultant force
        xnet = uG + uO + uO2 ;
        ynet = vG + vO + vO2 ;
        vspeed = sqrt(xnet^2 + ynet^2);
        theta = atan2(ynet,xnet);
        u(x,y) = vspeed*cos(theta);
        v(x,y) = vspeed*sin(theta);
%          hold on

    end
end
[X,Y] = meshgrid(1:1:100,1:1:100);
figure
quiver(X, Y, u, v, 3)


%% Defining the grid

% Plotting the obstacles
circles(obs1Pos(1),obs1Pos(2),obsRad,
'facecolor','black')
axis square

hold on
circles(obs2Pos(1),obs2Pos(2),obsRad,
'facecolor','black')

hold on % Plotting initial position
circles(startPos(1),startPos(2),2,
'facecolor','green')

hold on % Plotting target position
circles(goalPos(1),goalPos(2),2,
'facecolor','yellow')

%% Priting of the path
currentPos = startPos;
x = 0;

while sqrt((goalPos(1)-currentPos(1))^2 +
(goalPos(2)-currentPos(2))^2) > 1
    tempPos = currentPos +
[u(currentPos(1),currentPos(2)),
v(currentPos(1),currentPos(2))]
    currentPos = round(tempPos)
    hold on
    plot(currentPos(1),currentPos(2),'o',
'MarkerFaceColor', 'black')
    pause(0.5)
end
```

Folder: section_7 > file: GoalDelta.m
This is the code for the improved APF for finsing the delta of the goal

```
function [delXG, delYG] = GoalDelta(vx,
vy, gx, gy, goalR, goalS, alpha)
% This function gives delX, delY of
attraction caused by the goal point

dGoal = sqrt((gx-vx)^2 + (gy-vy)^2); %
distance bw goal and current position
thetaG = atan2((gy-vy),(gx-vx));     %
angle between goal and current position
```

```
if dGoal<goalR
    delXG = 0; delYG = 0;

elseif ((goalS + goalR) >= dGoal) &&
(dGoal >= goalR)
    delXG = 2*((dGoal -
goalR)^alpha)*cos(thetaG);

    delYG = 2*((dGoal -
goalR)^alpha)*sin(thetaG);

else
    delXG = (goalS^alpha)*cos(thetaG);
    delYG = (goalS^alpha)*sin(thetaG);

end
```

Folder: section_7 > file: ObsDelta.m
This is the code for the improved APF for finsing the delta of the obstacle

```
function [delXO, delYO] = ObsDelta(vx,
vy, ox, oy, obsRad, obsS, beta)
% This function gives delX, delY of
repulsion caused by the obstacPF
% this the improved AFP
L = 1;
inf = 10;
dObs = sqrt((ox-vx)^2 + (oy-vy)^2); %
distance bw obstacle and current position
thetaO = atan2((oy-vy),(ox-vx));     %
angle between goal and current position

dd = 4*obsRad;
if dObs<=obsRad
    delXO = -(sign(cos(thetaO)))*inf;

    delYO = -(sign(sin(thetaO)))*inf;

elseif (dObs < dd) && (dObs>=obsRad)

    delXO = -L*((dd/ (dObs-
obsRad))^beta)*cos(thetaO);
    delYO =-L*((dd/ (dObs-
obsRad))^beta)*sin(thetaO);

else
    delXO = 0;
    delYO = 0;
end

end
```