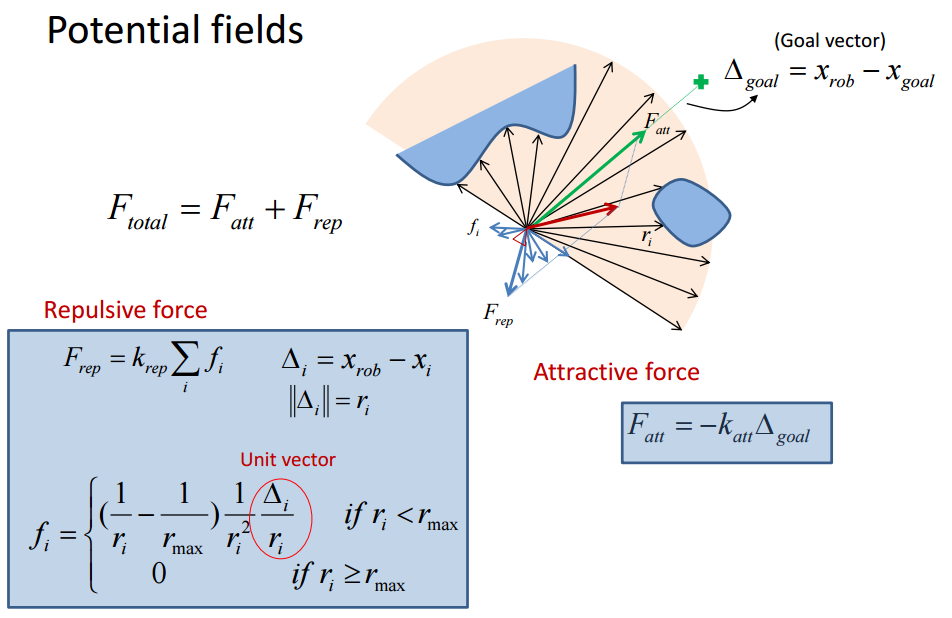
**Robotics**

**Exercise 8.1 Potential Fields**

In this exercise we are going to complete a code implementing a reactive navigation algorithm based on Potential Fields. This code generates a random map with a given number of obstacles, and requests the user to set the robot’s start and goal positions by clicking with the mouse on that map. Then you just have to enjoy watching how the robot moves!

**1. – Implementation.**

Euclidean distance from the robot to the obstacle (a number)

Musts be normalized (divided by) the norm of the distance to the goal

Obstacle position, so Δi is a 2-elements vector

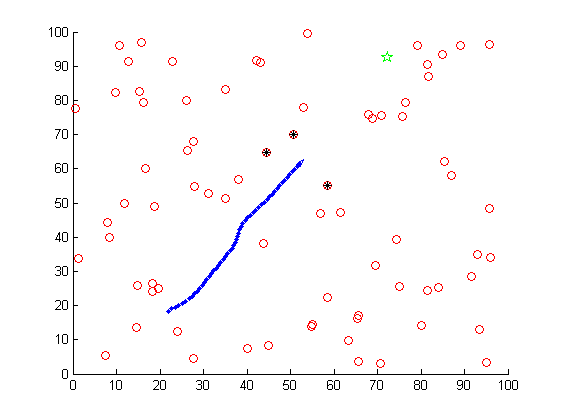
**1.1 –** Implement the computation of the Repulsive Force FRep, which is the sum of the repulsive forces yielded by each obstacle close to the object. Recall that forces are 2-elements vectors. Plot an asterisk over the obstacles that has influence on this force, and store the handler of that plot in hInfluentialObstacles.

**1.2 –** Compute the Attractive Force FAtt.Normalize the resultant Force by**.**

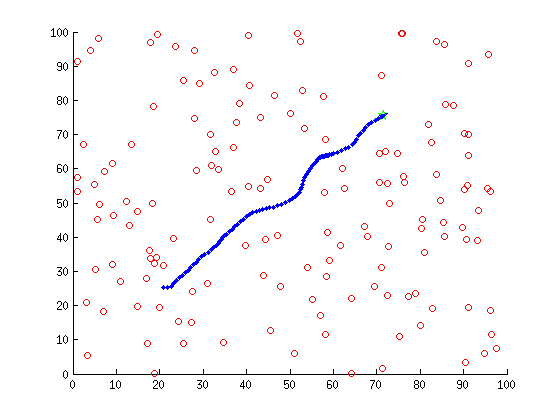
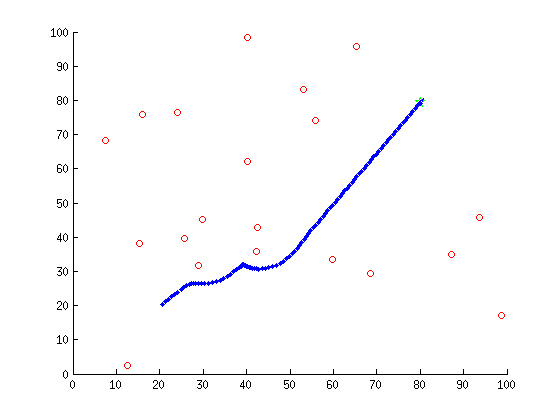
**1.3 –** Compute the Total Force FTotal.

**2. – Understanding what’s going on.**

**2.1 –** Explain the meaning of each element appearing in the plot during the simulation of the Potential Fields reactive navigation.

****

**2.2 –** Run the program setting different start and goal positions. Now change the values of the goal and obstacle gains (KGoal and KObstacles). How does this affect the paths followed by the robot?

**2.3 –** Play with different numbers of obstacles and discuss the obtained results.

**2.4 –** Illustrate a navigation where the robot doesn’t reach the goal position in the specified number of steps. Why did that happen? Could the robot have reached the goal with more iterations of the algorithm? Hint: take a look at the FTotalvariable.

**Appendix: Exercise’s code to analyze/extend**

function Potential\_Fields

clear all;

close all;

% Visualization modes

%mode = 'step\_by\_step';

mode = 'non\_stop';

% Configuration of the map

nObstacles = 75;

MapSize = 100;

Map = MapSize\*rand(2,nObstacles);

% Drawings

figure(1); hold on; set(gcf,'doublebuffer','on');

plot(Map(1,:),Map(2,:),'ro');

disp('Mark the starting point')

xStart = ginput(1)'

disp('Mark the destination point')

xGoal = ginput(1)'

plot(xGoal(1),xGoal(2),'gp','MarkerSize',10);

% Initialization of useful vbles

xRobot = xStart;

GoalError = xGoal - xRobot;

Hr = DrawRobot([xRobot;0],'b',[]);

% Algorithm configuration

RadiusOfInfluence = 10; % Objects far away of this radius does not influence

KGoal= 1; % Gain for the goal (attractive) field

KObstacles = 250; % Gain for the obstacles (repulsive) field

% Simulation configuration

k = 0;

nMaxSteps = 300;

while(norm(GoalError)>1 && k<nMaxSteps)

%find distance to all entities

Dp = Map-repmat(xRobot,1,nObstacles);

Distance = sqrt(sum(Dp.^2));

iInfluencial = find(Distance<RadiusOfInfluence);

%

% Compute repulsive (obstacles) potential field

%

if(~isempty(iInfluencial))

%

% Point 1.1

%

Else %nothing close

%

% Point 1.1

%

end

%

% Compute attractive (goal) potential field

%

%

% Point 1.2

%

%

% Compute total (attractive+repulsive) potential field

%

%

% Point 1.3

%

% New vehicle pose

xRobot = xRobot + FTotal;

Theta = atan2(FTotal(2),FTotal(1));

% Drawings

DrawRobot([xRobot;Theta],'k',Hr);

drawnow;

if strcmp(mode,'non\_stop')

pause(0.1); % for visibility purposes

else

pause;

end

if (~isempty(iInfluencial))

set(hInfluentialObstacles,'Visible','off'); % handler of a plot showing a mark over the obstacles that are within the radius of influence of the robot. Do this plot at Point 1.1

end

% Update termination conditions

GoalError = xGoal - xRobot;

k = k+1;

end

%-------------------------------------------------------------------------%

function H = DrawRobot(Xr,col,H)

p=0.005; % percentage of axes size

a=axis;

l1=(a(2)-a(1))\*p;

l2=(a(4)-a(3))\*p;

P=[-1 1 0 -1; -1 -1 3 -1];%basic triangle

theta = Xr(3)-pi/2;%rotate to point along x axis (theta = 0)

c=cos(theta);

s=sin(theta);

P=[c -s; s c]\*P; %rotate by theta

P(1,:)=P(1,:)\*l1+Xr(1); %scale and shift to x

P(2,:)=P(2,:)\*l2+Xr(2);

if(isempty(H))

H = plot(P(1,:),P(2,:),col,'LineWidth',0.1);% draw

else

set(H,'XData',P(1,:));

set(H,'YData',P(2,:));

plot(Xr(1),Xr(2),'b.','MarkerSize',10);

end