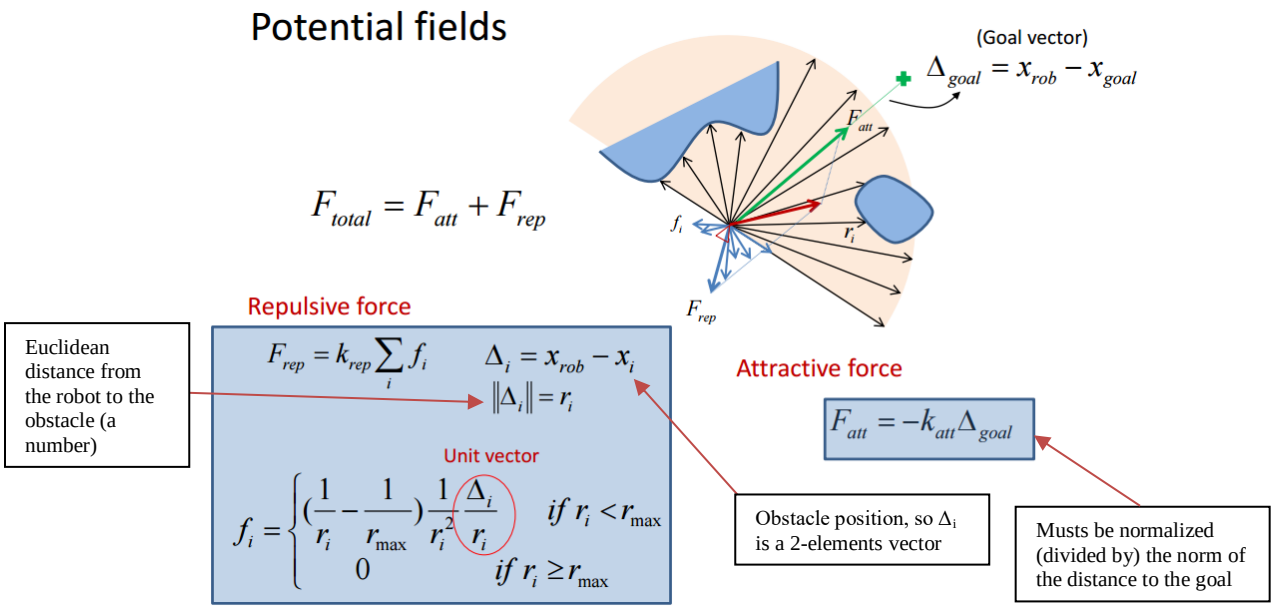
Robotics

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.



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`.

```
if(~isempty(iInfluencial))
      fi=0;
      for i = iInfluencial
          ri=Distance(i);
          rmax=RadiusOfInfluence;
          di=-Dp(:,i)/norm(Dp(:,i));
          f=(1/ri-1/rmax)*1/(ri^2)*di/ri;
          fi=fi+f;
      end
      landmarks=(Map(:,iInfluencial));
      hInfluentialObstacles=scatter(landmarks(1,:),landmarks(2,:),'y*');
      Frep=KObstacles*fi;
  else %nothing close
      Frep=0;
  end
```

1.2 – Compute the Attractive Force `FAtt`. Normalize the resultant Force by $\|\Delta_{goal}\|$.

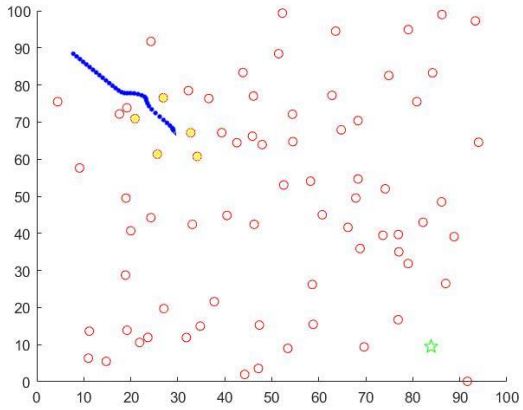
```
Fatt = -KGoal*-(GoalError/norm(GoalError));
```

1.3 – Compute the Total Force `FTotal`.

```
FTotal = Frep+Fatt;
```

2. – Understanding what's going on.

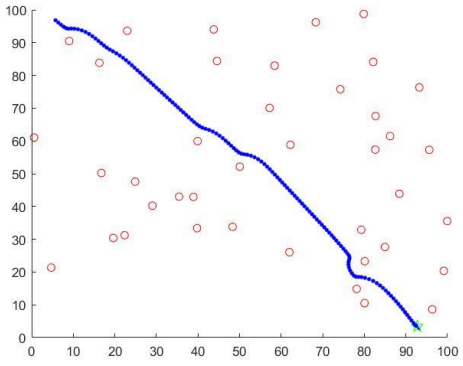
2.1 – Explain the meaning of each element appearing in the plot during the simulation of the Potential Fields reactive navigation.



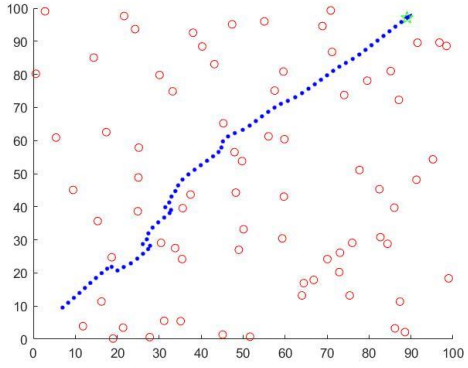
Plot 1. the intermediate execution of the program

As we see in the Plot 1, the blue circles and triangle represent the path of the robot and the actual position of the robot, respectively. The red circles represents the obstacles, the green star, the goal which the robot try to reach and the yellow asterisks are the object which the robot captures in his FOV.

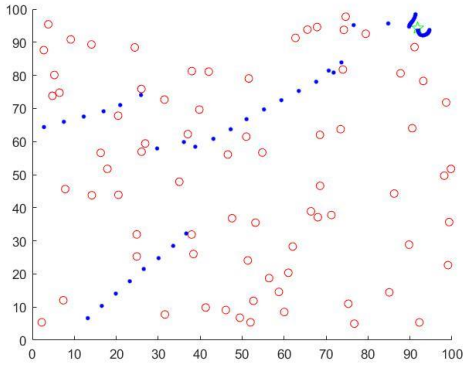
2.2 – Run the program setting different start and goal positions. Now change the values of the goal and obstacle gains (`KGoal` and `KObstacles`). How does this affect the paths followed by the robot?



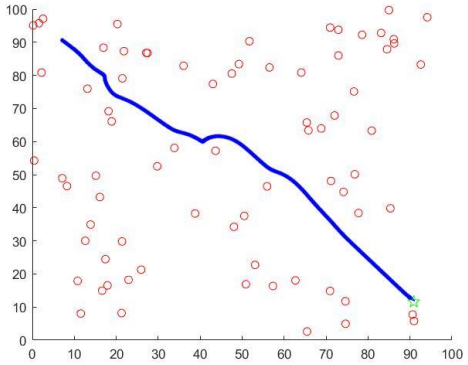
Plot 2. Original parameters



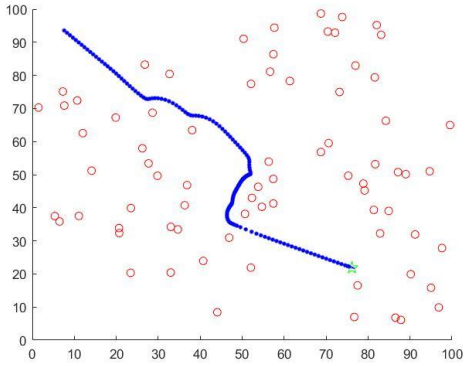
Plot 3. With KGoal=2



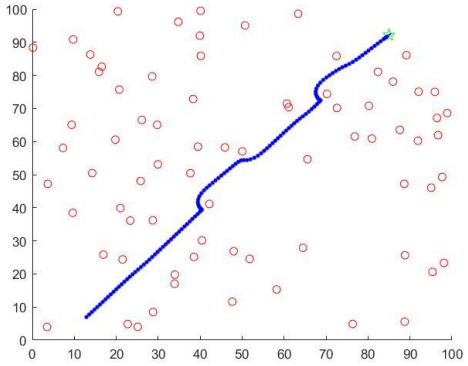
Plot 4. With KGoal = 5



Plot 5. With KGoal = 0.5



Plot 6. With KObstacles=500

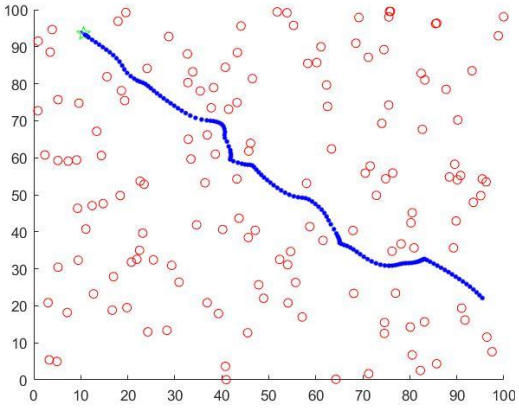


Plot 7. With KObstacles=50

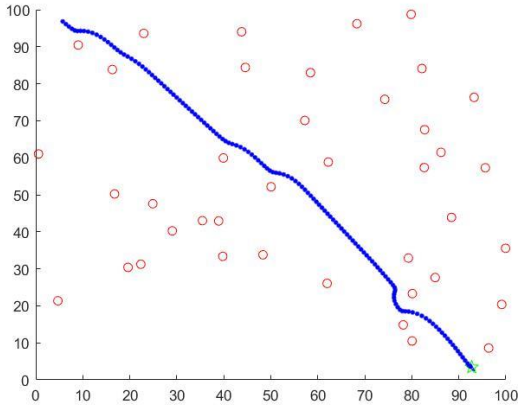
From Plot 3 to Plot 5 shows how changing KGoal change the speed which the robot goes. However, if this value is enough high, the robot will move too speedy that can lead to the malfunction of the program, such as in the Plot 4.

From the other plots, it shows the change in value of KObstacule. If we increase the value, the robot will be more sensible of the presence of the object. This make the robot avoid the object more far away. In the other hand, if we decrease the value, it seems that the robot does not sense the presence until it like in front of the object. From there, it will avoid the object.

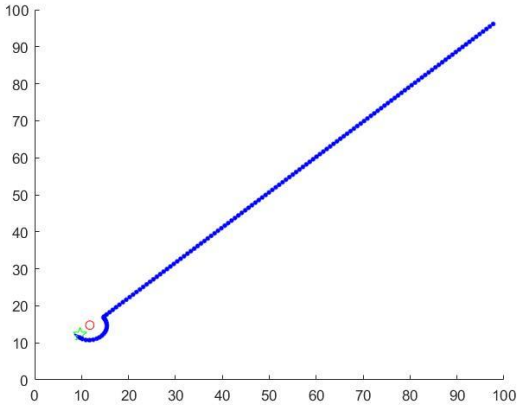
2.3 – Play with different numbers of obstacles and discuss the obtained results.



Plot 9. Map with 150 obstacles



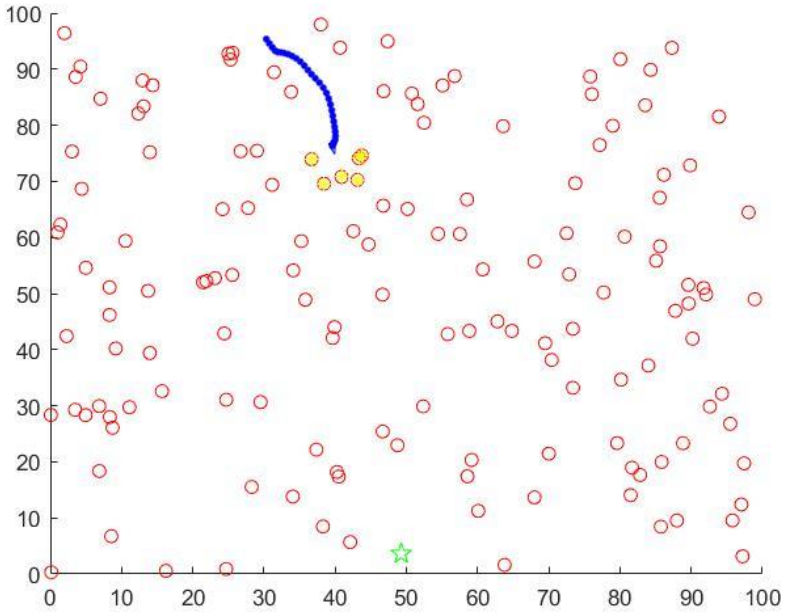
Plot 8. Map with 100 obstacles



Plot 10. Map with one obstacle

The plots show us if we have more obstacles, the robot will have more problems to reach the goal until the point can get stuck in a local minimum. However, if we have less obstacles, the robot will reach the goal with more facility. Moreover, it will have a more linear path than the other case.

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 `FTotal` variable.



Plot 11. Execution of the program with a "stuck state"

It did not reach the goal because it fell in a local minimum, ergo the obstacles produce a repulsive force which annuls or nearly annuls the attractive force. Therefore, the sum of these forces makes the robot will not move in any direction and will stay in that position.
It is difficult to get out of a local minimum, even if we give more iterations to the algorithm. Thus, it is nearly impossible to the robot to reach the goal, even though we give more iterations, if the robot reaches the condition we describe in the last paragraph.

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
          %
          fi=0;
          for i = iInfluencial
                    ri=Distance(i);
                    rmax=RadiusOfInfluence;
                    di=-Dp(:,i)/norm(Dp(:,i));
                    f=(1/ri-1/rmax)*1/(ri^2)*di/ri;
                    fi=fi+f;
          end
          landmarks=(Map(:,iInfluencial));
          hInfluentialObstacles=scatter(landmarks(1,:),landmarks(2,:),'y*');
          Frep=KObstacles*fi;

        Else %nothing close
           %
           % Point 1.1
           %
          Frep=0;
        end
```

```
        %
        % Compute attractive (goal) potential field
        %

          %
          % Point 1.2
          %
          Fatt = -KGoal*-(GoalError/norm(GoalError));

        %
        % Compute total (attractive+repulsive) potential field
        %

          %
          % Point 1.3
          %
          FTotal = Frep+Fatt;

        % 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
```