ASSIGNMENT 7

DECEMBER 29, 2021

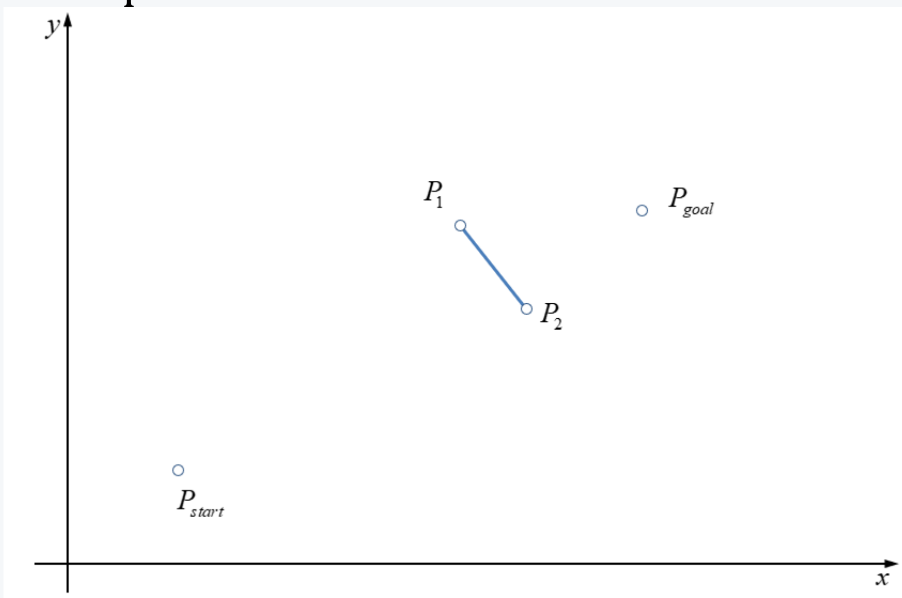
Submitted to: Prof. Dr. Kemalettin Erbatur

Name of Student: Moses Chuka Ebere
Student Number: 31491
Course: Autonomous Mobile Robotics (ME 525)

TABLE OF CONTENTS

INTRODUCTION

This is a computational assignment on the Visibility Graph. The system under consideration here is a holonomic one, so the holonomic point robot assumption is used.



The robot is required to travel from P_{start} to P_{goal} without crossing the obstacle represented by the line P1-P2.

Problem Formulation:

- It is noted that points P1, P2, P_{start}, and P_{goal} could assume **any positive real number**, so the MATLAB code is written with respect to this.
- First, the exact orientation of the lines P1-P2 and P_{start}-P_{goal} are found using:

$$\tan\theta = \frac{y_2 - y_1}{x_2 - x_1}$$

$$m = \tan\theta$$

In MATLAB, atan2(y,x) is used to find the exact orientation with respect to the horizontal axis.

- The following formula is now used to find the point of intersection of the two lines since we have obtained the thetas.

$$P_1 + \lambda_{1P_6}\begin{bmatrix}\cos(\theta_1)\\ \sin(\theta_1)\end{bmatrix} = P_2 + \lambda_{2P_6}\begin{bmatrix}\cos(\theta_2)\\ \sin(\theta_2)\end{bmatrix}$$

- The above is calculated by constructing the data matrix:

$$\begin{bmatrix} \cos(\theta_1) & -\cos(\theta_2) \\ \sin(\theta_1) & -\sin(\theta_2) \end{bmatrix}$$

- Now, the lambdas are found using:

$$\begin{bmatrix} \cos(\theta_1) & -\cos(\theta_2) \\ \sin(\theta_1) & -\sin(\theta_2) \end{bmatrix} \begin{bmatrix} \lambda_{1P_6} \\ \lambda_{2P_6} \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

- We slot the any of the above lambdas into the following equation to find the point of intersection of the two lines.

$$P = P_2 + \lambda_2 \begin{bmatrix} \cos(\theta_2) \\ \sin(\theta_2) \end{bmatrix}$$

- The P above is now slotted in the hint equation to find the overall lambda:

$$\begin{pmatrix} x \\ y \end{pmatrix} = P_1 + \lambda\left(P_2 - P_1\right)$$

- Using conditional statements, we can now check if lambda ϵ (0,1).

MATLAB CODE

```
clear all; close all; clc

% Initialize the points P1, P2, Pstart, and Pgoal using
random integers
% from 1 to 10. Note that any positive real number could be
used.
P1 = [randi(10)*abs(randn); randi(10)*abs(randn)];
P2 = [randi(10)*abs(randn); randi(10)*abs(randn)];

Pstart = [randi(10)*abs(randn); randi(10)*abs(randn)];
Pgoal = [randi(10)*abs(randn); randi(10)*abs(randn)];

% Find the orientation of the lines formed with the
horizontal axis.
y2_minus_y1 = P2(2) - P1(2);
x2_minus_x1 = P2(1) - P1(1);
theta_12 = atan2(y2_minus_y1, x2_minus_x1);

ygoal_minus_ystart = Pgoal(2) - Pstart(2);
xgoal_minus_xstart = Pgoal(1) - Pstart(1);
theta_sg = atan2(ygoal_minus_ystart, xgoal_minus_xstart);

% Find the point of intersection between the two lines.
data_matrix = [cos(theta_sg) -cos(theta_12); sin(theta_sg)
-sin(theta_12)];
lambda_P = (data_matrix)\(P1 - Pstart);
lambdaS_P = lambda_P(1);
lambda1_P = lambda_P(2);
% The following P is the point of intersection.
P = Pstart + lambdaS_P*[cos(theta_sg); sin(theta_sg)];

% Using the equation in the hint, we can obtain lambda. P =
P1 + lambda(P2-P1)
% If lamba is in the range (0,1), the intersection point,
P, is between P1 and P2
lambda = (P2 - P1)\(P - P1);

% This is used to be sure an intersection occurs between
the two points.
if norm(Pstart - P) >  norm(Pgoal - P)
    Pcheck = Pstart;
else
```

```
    Pcheck = Pgoal;
end
% Check if lambda lies in the range (0,1)
if (lambda > 0 && lambda < 1) && (norm(Pgoal - Pstart) >
norm(Pcheck - P))
    result = 'The line segment between the start and goal
points intersects the obstacle P1-P2';
    prompt_message = 'Please find two new paths and choose
the shorter one.';
    disp(result);
    figure
    plot(P1(1), P1(2), 'bo')
    hold
    plot(P2(1), P2(2), 'bo')
    plot(Pstart(1), Pstart(2), 'mo')
    plot(Pgoal(1), Pgoal(2), 'go')
    plot([P1(1) P2(1)], [P1(2) P2(2)], 'b')
    plot([Pstart(1) Pgoal(1)], [Pstart(2) Pgoal(2)], 'r--')
    text(Pstart(1)+0.05, Pstart(2)+0.05, 'Pstart', 'color',
'm')
    text(Pgoal(1)+0.05, Pgoal(2)+0.05, 'Pgoal', 'color',
'g')
    text(P1(1)+0.05, P1(2)+0.05, 'P1', 'color', 'b')
    text(P2(1)+0.05, P2(2)+0.05, 'P2', 'color', 'b')
    title('Paths cross; find alternative paths!')
    axis([0 max([P2(1);P1(1);Pstart(1);Pgoal(1)])+1 0
max([P2(2);P1(2);Pstart(2);Pgoal(2)])+1])
    xlabel('x [m]'), ylabel('y [m]'), grid
    disp(prompt_message);

    % Alternative Path 1
    a_path1_a = norm(Pstart - P1);
    a_path1_b = norm(P1 - Pgoal);
    a_path1 = a_path1_a + a_path1_b;

    % Alternative Path 2
    a_path2_a = norm(Pstart - P2);
    a_path2_b = norm(P2 - Pgoal);
    a_path2 = a_path2_a + a_path2_b;
```

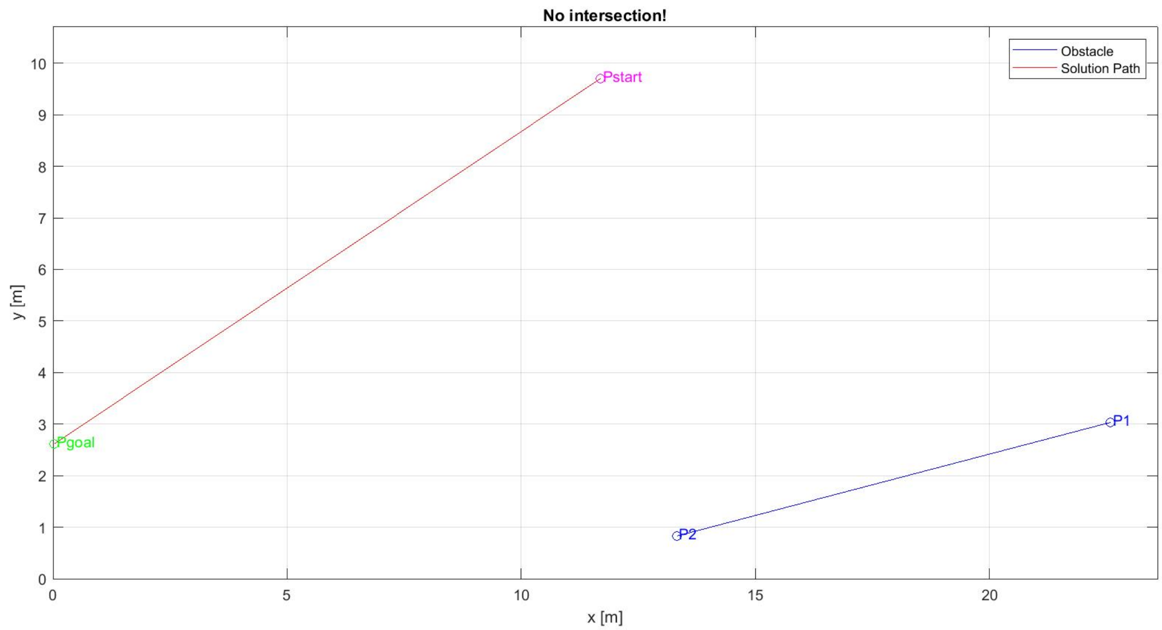
```
    % The following identifies the solution path. If both
paths are of the same length, it will identify both as
plausible solution paths

    if a_path1 < a_path2
        figure
        plot(P1(1), P1(2), 'bo')
        hold
        plot(P2(1), P2(2), 'bo')
        plot(Pstart(1), Pstart(2), 'mo')
        plot(Pgoal(1), Pgoal(2), 'go')
        h0 = plot([P1(1) P2(1)], [P1(2) P2(2)], 'b');
        h1 = plot([Pstart(1) P1(1)], [Pstart(2) P1(2)],
'r');
        h2 = plot([P1(1) Pgoal(1)], [P1(2) Pgoal(2)], 'r');

        h3 = plot([Pstart(1) P2(1)], [Pstart(2) P2(2)],
'k');
        h4 = plot([P2(1) Pgoal(1)], [P2(2) Pgoal(2)], 'k');

        text(Pstart(1)+0.05, Pstart(2)+0.05, 'Pstart',
'color', 'm')
        text(Pgoal(1)+0.05, Pgoal(2)+0.05, 'Pgoal',
'color', 'g')
        text(P1(1)+0.05, P1(2)+0.05, 'P1', 'color', 'b')
        text(P2(1)+0.05, P2(2)+0.05, 'P2', 'color', 'b')

        legend([h0 h1 h3],{'Obstacle', 'Path 1: Solution
Path','Path 2'})
        title('New Alternative Paths')
        axis([0 max([P2(1);P1(1);Pstart(1);Pgoal(1)])+1 0
max([P2(2);P1(2);Pstart(2);Pgoal(2)])+1])
        xlabel('x [m]'), ylabel('y [m]'), grid
    elseif a_path1 > a_path2
        figure
        plot(P1(1), P1(2), 'bo')
        hold
        plot(P2(1), P2(2), 'bo')
        plot(Pstart(1), Pstart(2), 'mo')
        plot(Pgoal(1), Pgoal(2), 'go')
        h0 = plot([P1(1) P2(1)], [P1(2) P2(2)], 'b');
```

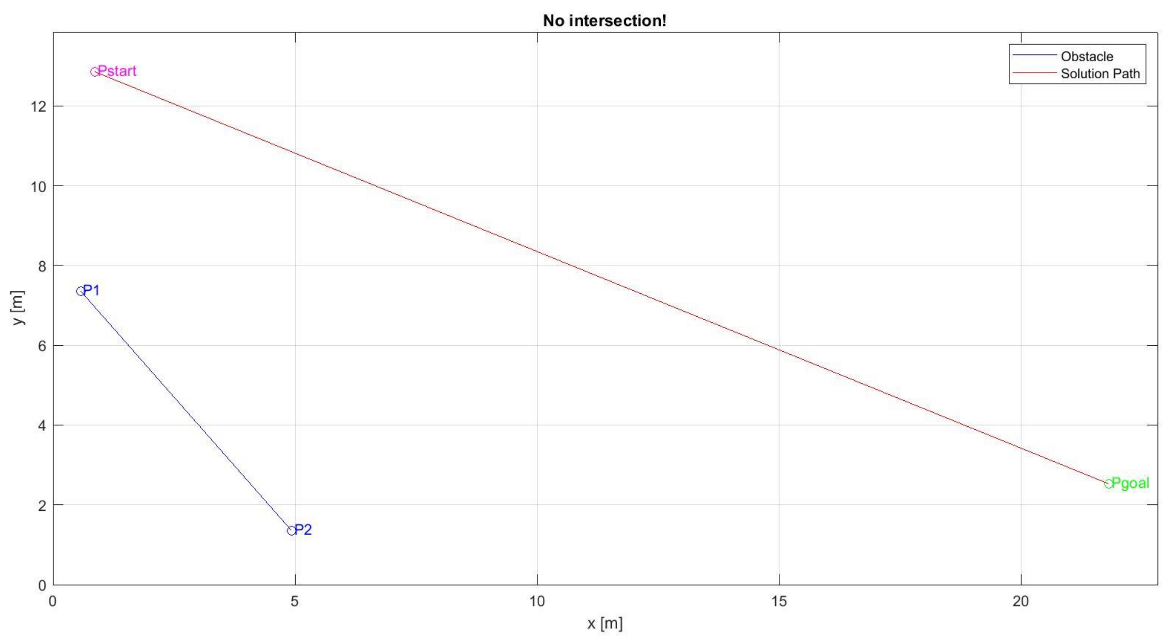
```
        h1 = plot([Pstart(1) P1(1)], [Pstart(2) P1(2)],
'k');
        h2 = plot([P1(1) Pgoal(1)], [P1(2) Pgoal(2)], 'k');

        h3 = plot([Pstart(1) P2(1)], [Pstart(2) P2(2)],
'r');
        h4 = plot([P2(1) Pgoal(1)], [P2(2) Pgoal(2)], 'r');

        text(Pstart(1)+0.05, Pstart(2)+0.05, 'Pstart',
'color', 'm')
        text(Pgoal(1)+0.05, Pgoal(2)+0.05, 'Pgoal',
'color', 'g')
        text(P1(1)+0.05, P1(2)+0.05, 'P1', 'color', 'b')
        text(P2(1)+0.05, P2(2)+0.05, 'P2', 'color', 'b')

        legend([h0 h1 h3],{'Obstacle', 'Path 1','Path 2:
Solution Path'})
        title('New Alternative Paths')
        axis([0 max([P2(1);P1(1);Pstart(1);Pgoal(1)])+1 0
max([P2(2);P1(2);Pstart(2);Pgoal(2)])+1])
        xlabel('x [m]'), ylabel('y [m]'), grid
    else % If both paths are the same length.
        figure
        plot(P1(1), P1(2), 'bo')
        hold
        plot(P2(1), P2(2), 'bo')
        plot(Pstart(1), Pstart(2), 'mo')
        plot(Pgoal(1), Pgoal(2), 'go')
        h0 = plot([P1(1) P2(1)], [P1(2) P2(2)], 'b');
        h1 = plot([Pstart(1) P1(1)], [Pstart(2) P1(2)],
'r');
        h2 = plot([P1(1) Pgoal(1)], [P1(2) Pgoal(2)], 'r');

        h3 = plot([Pstart(1) P2(1)], [Pstart(2) P2(2)],
'r');
        h4 = plot([P2(1) Pgoal(1)], [P2(2) Pgoal(2)], 'r');

        text(Pstart(1)+0.05, Pstart(2)+0.05, 'Pstart',
'color', 'm')
        text(Pgoal(1)+0.05, Pgoal(2)+0.05, 'Pgoal',
'color', 'g')
        text(P1(1)+0.05, P1(2)+0.05, 'P1', 'color', 'b')
```

```
        text(P2(1)+0.05, P2(2)+0.05, 'P2', 'color', 'b')

        legend([h0 h1 h3],{'Obstacle', 'Path 1','Path 2'})
        title('New Alternative Paths - The Paths are of the
Same Length')
        axis([0 max([P2(1);P1(1);Pstart(1);Pgoal(1)])+1 0
max([P2(2);P1(2);Pstart(2);Pgoal(2)])+1])
        xlabel('x [m]'), ylabel('y [m]'), grid
    end
else % If there's no intersection
    result1 = 'The robot encounters no obstacle from the
start to the goal point!';
    disp(result1);
    figure
    plot(P1(1), P1(2), 'bo')
    hold
    plot(P2(1), P2(2), 'bo')
    plot(Pstart(1), Pstart(2), 'mo')
    plot(Pgoal(1), Pgoal(2), 'go')
    plot([P1(1) P2(1)], [P1(2) P2(2)], 'b')
    h00 = plot([Pstart(1) Pgoal(1)], [Pstart(2) Pgoal(2)],
'r');
    legend(h00,{'Solution Path'})
    text(Pstart(1)+0.05, Pstart(2)+0.05, 'Pstart', 'color',
'm')
    text(Pgoal(1)+0.05, Pgoal(2)+0.05, 'Pgoal', 'color',
'g')
    text(P1(1)+0.05, P1(2)+0.05, 'P1', 'color', 'b')
    text(P2(1)+0.05, P2(2)+0.05, 'P2', 'color', 'b')
    title('No intersection!')
    axis([0 max([P2(1);P1(1);Pstart(1);Pgoal(1)])+1 0
max([P2(2);P1(2);Pstart(2);Pgoal(2)])+1])
    xlabel('x [m]'), ylabel('y [m]'), grid
end
```

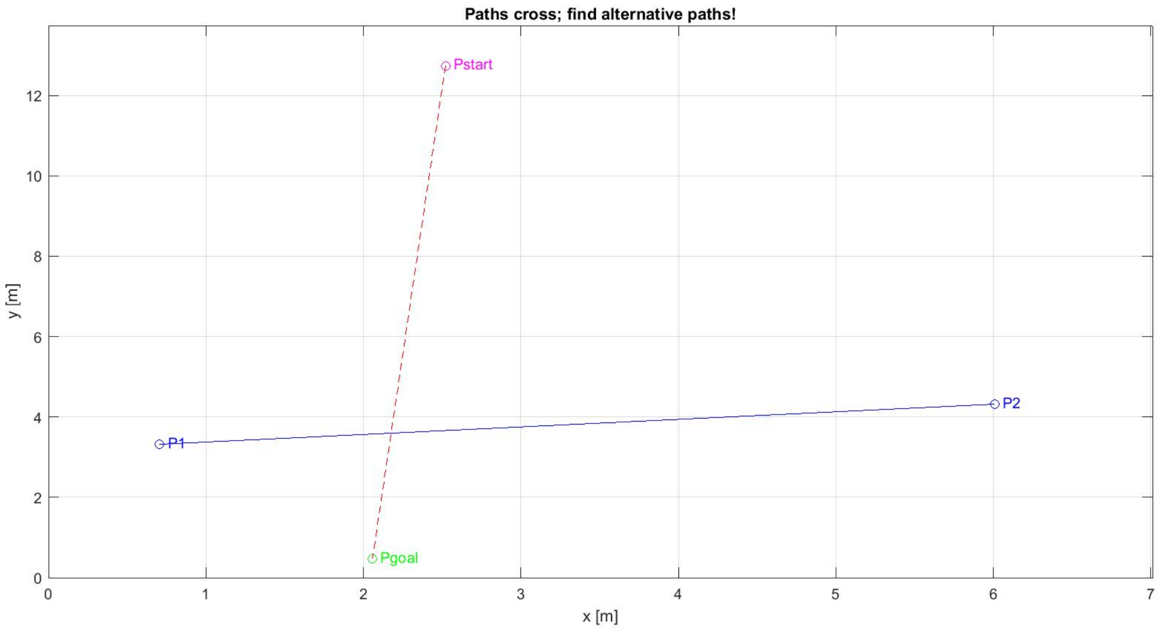
PLOTS – SAMPLE RESUTLS

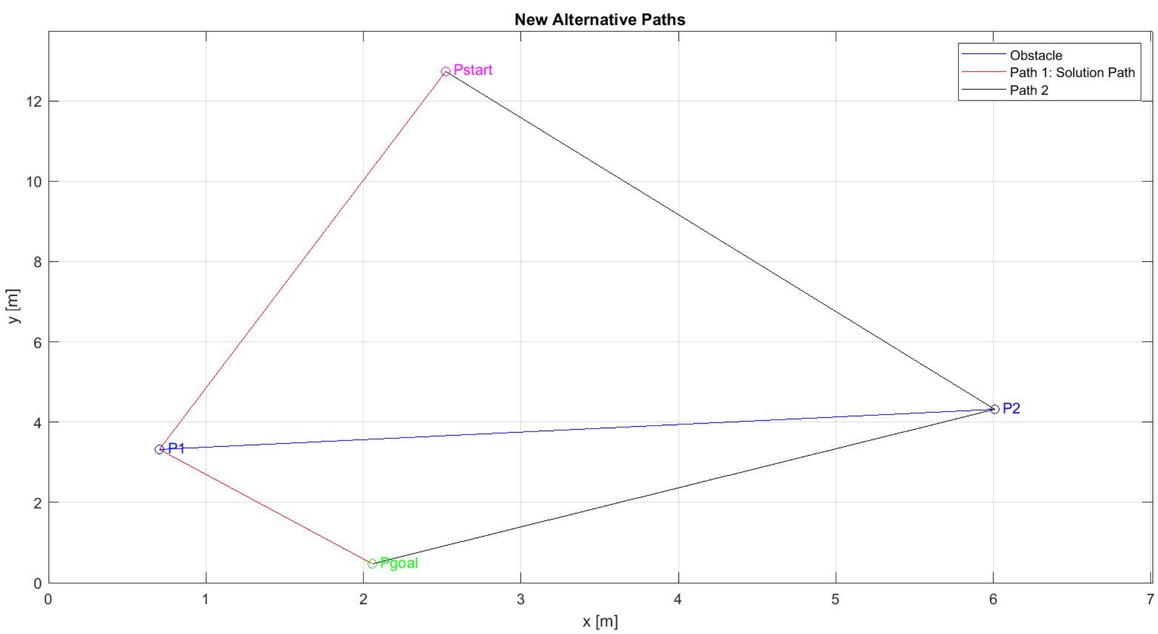
When there’s no intersection

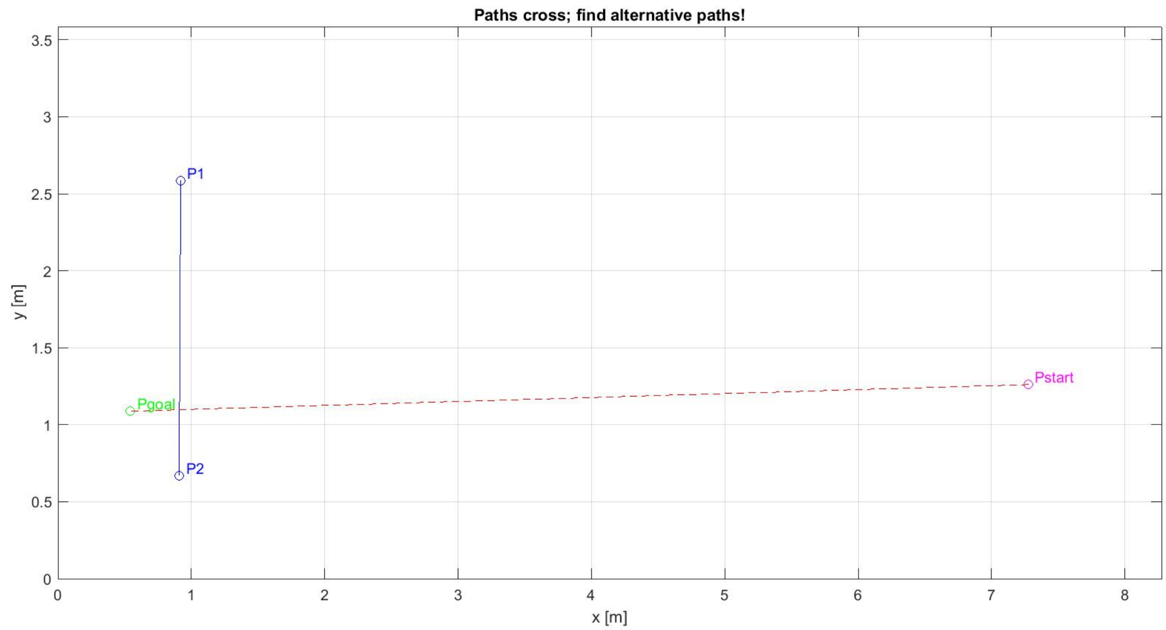


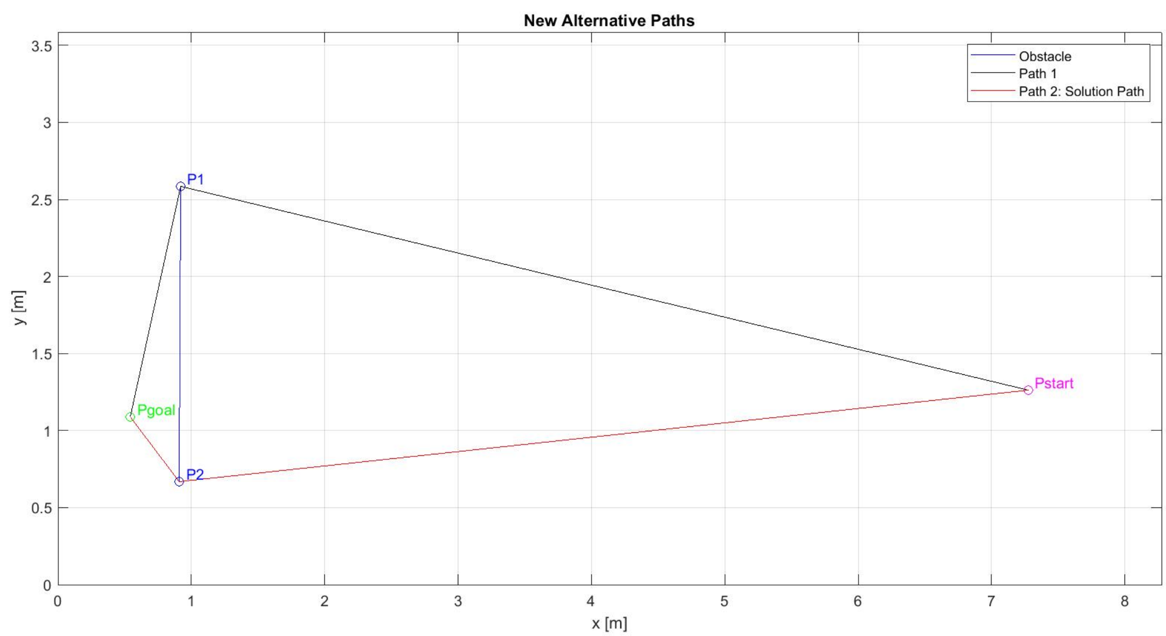


There's an intersection; find two alternative paths and choose the shorter one:



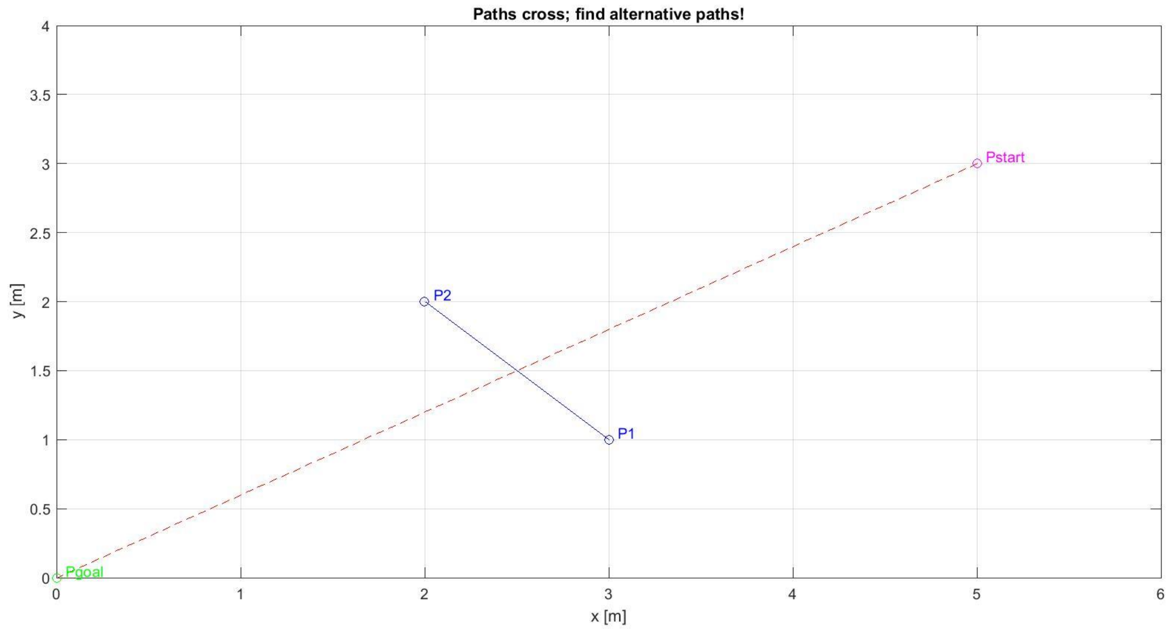


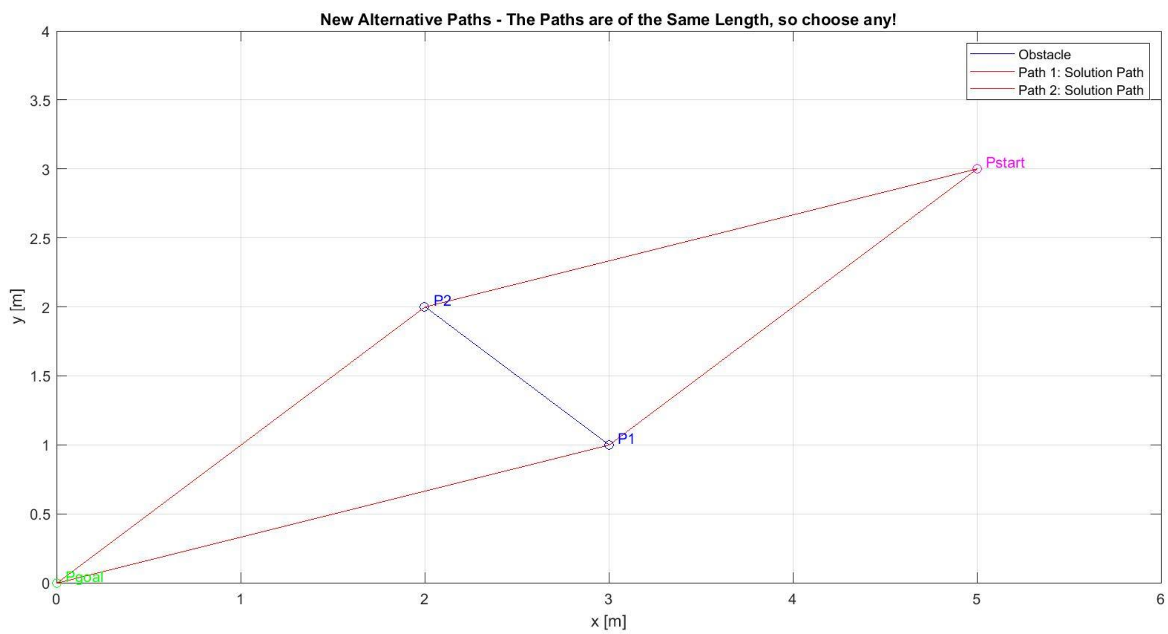




DISCUSSION

- The code is written to perform very well for any positive real number.
- When the path crosses the obstacle, it is plotted using dotted lines.
- The code identifies both alternative paths as plausible solution paths **when they are exactly of the same length** as in the following case:


Paths cross; find alternative paths!
Pstart
P2
P1
Pgoal
y [m]
x [m]


New Alternative Paths - The Paths are of the Same Length, so choose any!
Obstacle
Path 1: Solution Path
Path 2: Solution Path
Pstart
P2
P1
Pgoal
y [m]
x [m]