ELEE 4200

Autonomous Mobile Robots

Homework No:7

Wall Following

Author 1 :
Name; Wang Jingya
T.No.: 02157119

Author 2:
Name: Mi Liang
T.No.: 02157118

Instructor:
Dr. Mohan Krishnan

November 11, 2019

Grading Rubric - Information for Students

Grading rubric that will broadly apply to assessing performance. That is, assessment exercises that are associated with a predominantly qualitative rather than quantitative character. If, for a particular exercise, there is substantial deviation from the scheme outlined below, I will let you know! Note: Be aware that if your "Quality of Presentation" is poor, it could impact scores assigned for the other two categories! (DO NOT EDIT THIS PAGE)

		Level of accomplishment					
		0	**1**	**2**	**3**	**4**	**5**
Aspect of effort	**Extent of Completion of Technical Requirements**	Achieved none of the objectives or did not submit	Achieved very few of the objectives	Achieved some of the objectives	Achieved most of the objectives	Achieved almost all the objectives	Demonstrated additional initiative beyond what was required.
	Quality of Analysis & Conclusions	Inadequate	Poor	Below average	Average	Good	Insightful!
	Quality of Presentation	Inadequate	Poor	Below average	Average	Good	Exceptional!

Figure 1: Level of accomplishments

- **Student's overall Level/Score (To be entered by Professor/GTA):**.

Contents

List of Figures

List of Tables

1 Abstract

Our goal is controlling the robot to run following the wall. Additionally, making the robot drive outside the wall and inside the wall to the different directions to let the wall standing on the right or on the left. Whenever the robot is inside the wall or outside the wall, making the robot to stay in a fixed range apart from the wall.

2 Introduction

- The following are the specific requirements:

- Task A: Driving the robot outside the wall:

- Task A (1):
  1. Let the robot run around the given "island" wall. Additionally, make the robot stay away from the robot for 2 meters.
  2. During the running process, keeping the wall at the robot right side.
  3. As the robot running, plotting the robot path.
  4. After running around the "island" wall one time, output the distance the robot has run.

- Task A (2):
  1. Let the robot run around the given "island" wall. Additionally, make the robot stay away from the robot for 2 meters.
  2. During the running process, keeping the wall at the robot left side.
  3. As the robot running, plotting the robot path.
  4. After running around the "island" wall one time, output the distance the robot has run.

- Task B: Driving the robot inside the wall:

- Task B (1):
  1. Let the robot run around the given "island" wall. Additionally, make the robot stay away from the robot for 2 meters.
  2. During the running process, keeping the wall at the robot right side.
  3. As the robot running, plotting the robot path.
  4. After running around the "island" wall one time, output the distance the robot has run.

- Task B (2):
  1. Let the robot run around the given "island" wall. Additionally, make the robot stay away from the robot for 2 meters.
  2. During the running process, keeping the wall at the robot left side.
  3. As the robot running, plotting the robot path.
  4. After running around the "island" wall one time, output the distance the robot has run.

3 Methodology

- **The basic idea:**

- How to make the robot alongside the wall?
  Firstly, calculate the front distance and the lateral distance.
  As the robot running, the Hokuyo LIDAR will give back the obstacles distances which it has detected. And in the Homework 6, we know that the Hokuyo LIDAR has 270 degrees range and 1080 LIDAR lines. So, we can use

$$\text{The interval between to LIDAR a} = \frac{270}{1080}$$

  We know that the LIDAR line's number which detecting the front conditions is 540(According to last time's homework)Then, because the lateral side detecting line is 90° rotating from the 540 line. The left side detecting line is 540 line rotating clockwise 90°. The left side detecting line is 540 line rotating counterclockwise 90°.
  Through the interval, we can calculate the left LIDAR line which we want to detect is

$$\text{The left detecting LIDAR line number} = \frac{90^\circ}{a} + 540 \quad ,$$

  and the result is 900.

The right LIDAR line which we want to detect is

$$\text{The right detecting LIDAR line number} = \frac{-90^\circ}{a} + 540 \quad ,$$

and the result is 180.
Then, because if we just use one line to judge the front conditions or the side conditions, the detecting result will change quickly, and it might make the running not very stable. So, enlarging the detecting line range, which means that calculating the mean value of the feedback values from index-n to index +n, and using this method to make the robot moviang more fluently. Specifically, The front detecting range is [540-5,540+5], which is [535,545]. The left side detecting range is [900-3,900+3], which is [897,903]. The right side detecting range is [180-3,180+3],which is [177,183]. Later, using the following formula to calculate the front distance apart from the obstacles and the side distance apart from the obstacles.

$$\text{Front distance} = \frac{\sum_{535<i<545} data}{11}$$
$$\text{Left distance} = \frac{\sum_{897<i<903} data}{7}$$
$$\text{Right distance} = \frac{\sum_{177<i<183} data}{7}$$

Finally, Using the distance to judging the conditions and adjust the robot actions alongsdie the wall.

- How to plot the robot moving path?
  Creating a subscriber for model_state, and get the robot position from the subscriber.As the robot moving, plotting the position.

- How to get the robot moving distance?
  Using the Odometry feedback. Creating two arrays to record the robot Odometry coordinates. Both arrays have two elements to record the previous value and the current value. And using the following formula to calculate this section moving distance.

$$\text{distance} = \sqrt{(Xpre - Xcur)^2 + (Ypre - Ycur)^2}$$

  Finally, adding all section line distance together to get the whole distance the robot has run.

3.1 The general idea for Task A:

Firstly, setting the initial robot pose.
If keep the wall to the left, setting the position like following:
position: x: -10.0;y: -6.0;z: 0.0

orientation:x: 0.0;y: 0.0;z: -0.7;w: 0.7
If keep the wall to the right, setting the position like following:
position: x: -10.0;y: -6.0;z: 0.0 orientation:x: 0.0;y: 0.0;z: 0.7;w: 0.7 Then, giving robot a velocity to control it to move.

Secondly, judging the surrounding conditions.

Basically, Judging the front conditions first, and dividing into three basic conditions.
1. If the front distance is larger than 2.5 and less than 100(The value is simulating the Infinite value). This means the robot is running alongside a small range wall. Therefore, control the robot to make it running like a straight line.
(1) keep the wall to the right:
So, in this situation,if the robot leans to the left(the side distance is larger than fixed value), changing the angular rate to make the robot turning to the right. Because turning to clockwise, the given angular velocity should be negative. If the robot leans to the right(the side distance is less than fixed value), changing the angular velocity to make the robot turning to the left. Because turning to counterclockwise, the given angular velocity should be positive. Otherwise, running in a straight line. The judging distance is 2.01 meters, which gives the robot a 0.02 tolerance.
(1) keep the wall to the left:
So, in this situation,if the robot leans to the left(the side distance is larger than fixed value), changing the angular rate to make the robot turning to the right. Because turning to clockwise, the given angular velocity should be negative. If the robot leans to the right(the side distance is larger than fixed value), changing the angular velocity to make the robot turning to the left. Because turning to counterclockwise, the given angular velocity should be positive. Otherwise, running in a straight line. The judging distance is 2.01 meters, which gives the robot a 0.02 tolerance.

2. If the front distance is less than 2.5, the situation is that the robot are facing the wall and need to rotate a big angle to avoiding hitting on the wall.

3.The last situation is when the front distance is **Infinite**, which means there is no barrier in the front. Then, let the robot to judge the lateral conditions.
(1)If the lateral sides also doesn't have the barrier, the situation means that

the robot is at the corner, and it should turn around.
(2)But if there are also some obstacles on the side. This is means that the robot is still running along the wall, but the wall is very long. The robot should adjust its angular velocity in a small range to make itself moving in a straight line. The idea is same as the first condition.

3.2 The general idea for Task B:

The whole idea is same as Task A, but setting the initial position inside the "island" wall. If keep the wall to the left, setting the position like following:
position: x: 0.0;y: 0.0;z: 0.0
orientation:x: 0.0;y: 0.0;z: 0.0;w: 0.0
If keep the wall to the right, setting the position like following:
position: x: 0.0;y: 0.0;z: 0.0 orientation:x: 0.0;y: 0.0;z: 1.0;w: 0.0

4 Results

1. The whole running video is in the zip file.

2. The following are the path figures for the for tasks:
   The path for the Task A: Outside of the given "island"- keep the wall to the right

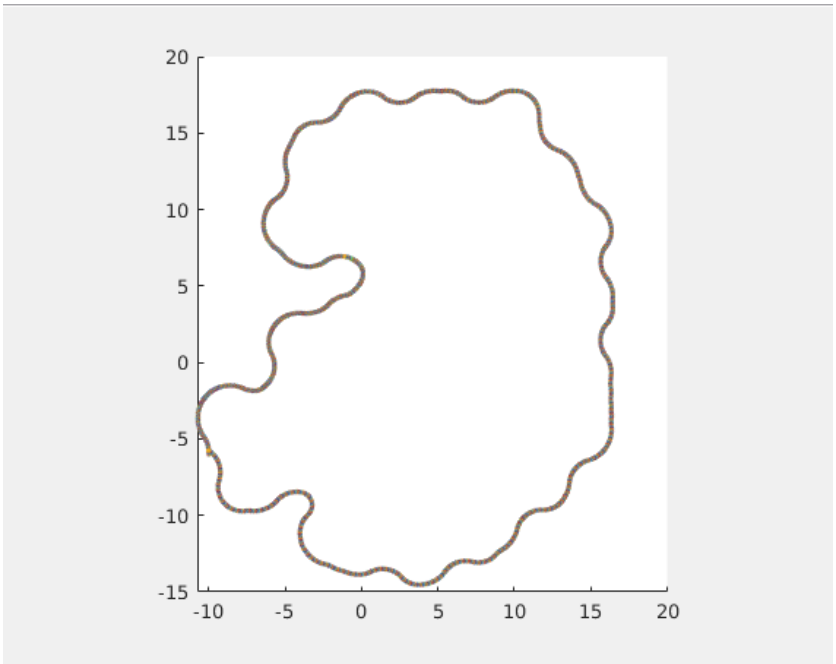

Figure 2: The path of outside the wall and keeping the wall to the right

The length of this task:

```
The robot have run 5.093276 meters.
```

Figure 3: The distance of keeping running outside of the wall and keeping to the right

The path for the Task A: Outside of the given “island”- keep the wall to the left

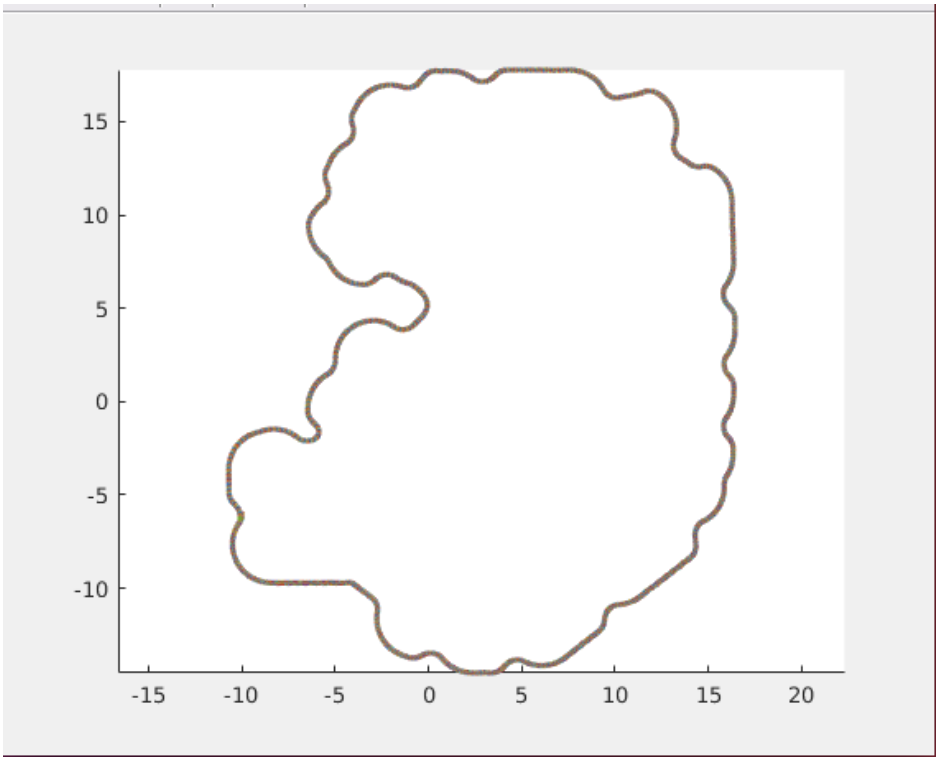

Figure 4: The path of outside the wall and keeping the wall to the left

The length of this task:

```
The robot have run 3.205700 meters.
```

Figure 5: The distance of keeping running outside of the wall and keeping to the left

The path for the Task B: Inside of the given “island”- keep the wall to the right

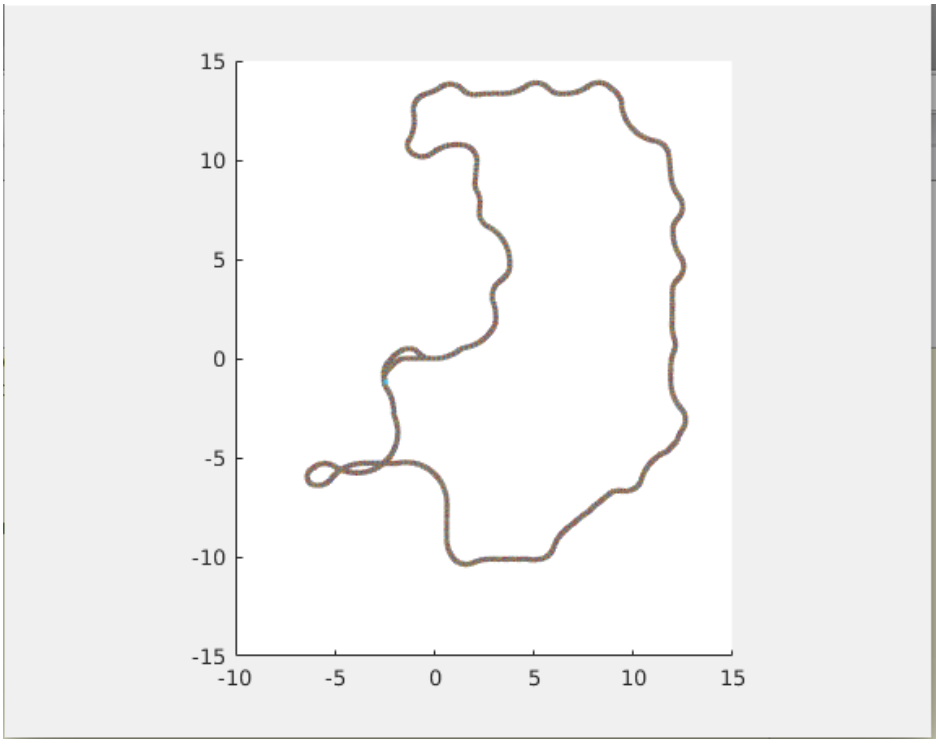

Figure 6: The path of inside the wall and keeping the wall to the right

The length of this task:

```
The robot have run 3.492700 meters.
```

Figure 7: The distance of keeping running outside of the wall and keeping to the right

The path for the Task B: Inside of the given “island”- keep the wall to the left

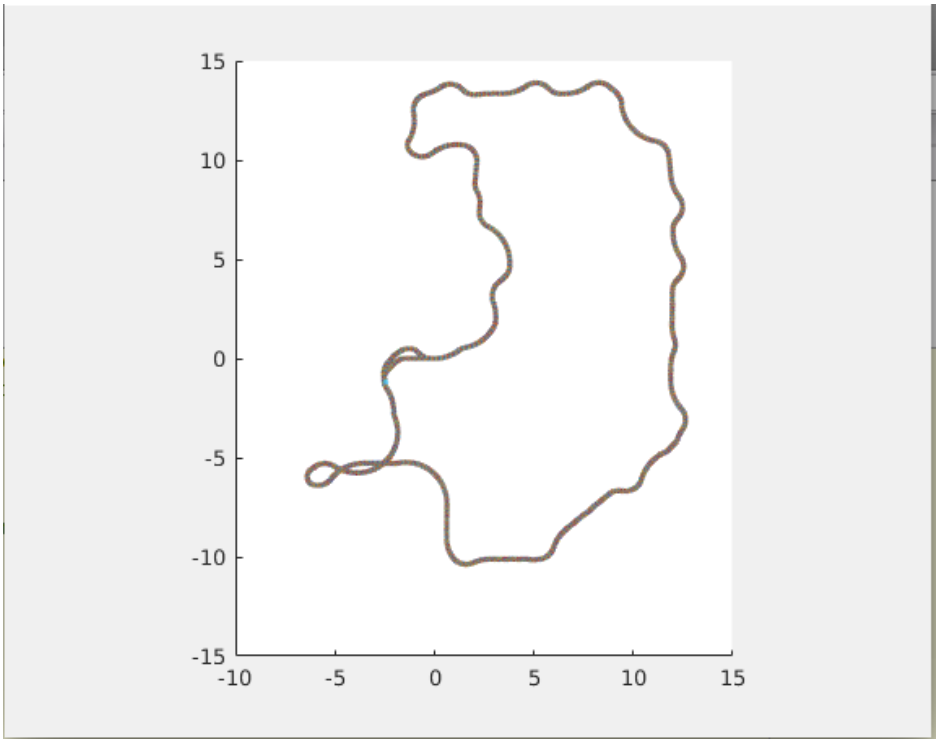

Figure 8: The path of inside the wall and keeping the wall to the left

The length of this task:

```
The robot have run 2.328372 meters.
```

Figure 9: The distance of keeping running outside of the wall and keeping to the left

5 Discussions and Conclusion

- We have successfully achieve the goal.

- **The problem we have met:**

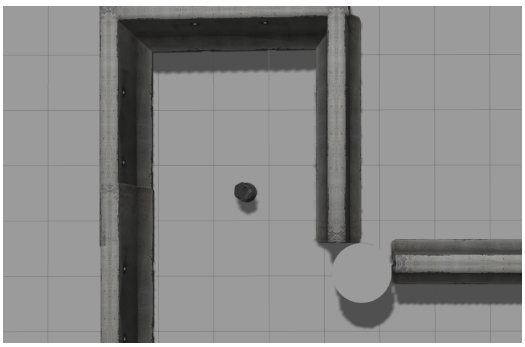

Figure 10: The robot might hit the wall at the corner

At the first time, we just dividing the situation into three conditions:

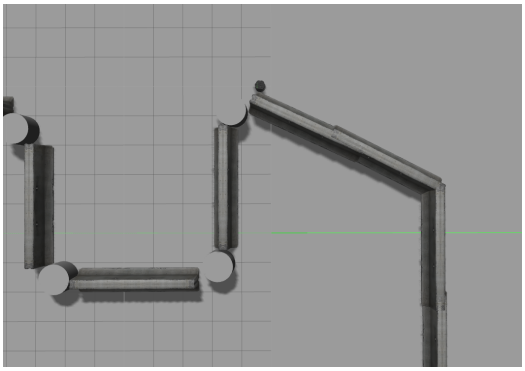

Figure 11: The robot might hit the wall at the corner

1. The front distance is larger than 2.5 and less than 100;
Adjusting the robot to run as straightly.
2. The front distance is less than 2.5; Turning at the corner
3. The last situation is when the front distance is **Infinite**;
Turning at the wall crossing corner.
And we just adjust the robot to move straight line in the first condition. But we forgot the situation: When the wall is very long, and the robot's front doesn't have any obstacles. So the robot think it is the third condition, and it will turn to inverse direction we want.

- **The solution:**
Our solution is adding a judging in the third condition:

If the side direction has obstacles, the robot is still walking alongside the wall. So, adjust the robot to run in straight line. If the side direction doesn't have any obstacles, the robot is at the big wall corner, and it should turn a big angle in order to follow the wall.

References

A Appendix

The following are guidelines. Use your judgment appropriately

A.1 List of attached files

1. The matlab code is in the Appendix.

Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Homework 6: Drive Robot to Avoid Obstacles                              %
%Auther's name: Wang Jingya                                              %
%Date:10/28/2019                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Task A: Outside of the given "island"- keep the wall to the right

Setting the initial position as position: x: -10.0 y: -6.0 z: 0.0 orientation: x: 0.0 y: 0.0 z: 0.7 w: 0.7

```
clear all
close all
clc

% Shut down the executing ros
rosshutdown;

% Initialize ROS
ipaddress='localhost';
rosinit(ipaddress);

% Reset Odometry
reset_odom=rospublisher('/mobile_base/commands/reset_odometry');
reset_msg=rosmessage(reset_odom);
send(reset_odom,reset_msg);

% Create the publishers and subscribers
robot = rospublisher('/mobile_base/commands/velocity');
odom_subs = rossubscriber('/odom');
model_subs=rossubscriber('/gazebo/model_states');
laser=rossubscriber('/laserscan');

% Create the message with the right format for the topic
velmsg = rosmessage(robot);
pose = rosmessage(odom_subs);
model=rosmessage(model_subs);
laser_msg=rosmessage(laser);

% Calculate the degrees between two LIDAR lines
index_incre = 270/1080;
% Claculate the RIDAR lines number we want
index1 = -90/index_incre + 540;

% Setting a number to start running the robot
m = 1;

% Initializing the running distance value which used to record the ditance
% between the starting position and the current position
```

```
running_distance = 0;

% Used to make the distance recording continue
j=1;

% Setting two arraies to recording the odometry in order to calculate the
% distance which the robot has run
odom_pose_X(1,j)=0;
odom_pose_Y(1,j)=0;

while m==1
 % This section is used to receive the position information from the
 % model_state, in order to plot the trajectory figures.
        model=receive(model_subs);

        position_X = model.Pose(34,1).Position.X;
        position_Y = model.Pose(34,1).Position.Y;

        hold on
        axis equal
        plot(position_X,position_Y,'.');

 % This section is used to  receive the coordinates information from
 % the Odometry, which is used to calculate the distance the robot has run,
 % Finally, output the distance
        j=j+1;
        pose=receive(odom_subs);
        odom_pose_X(1,j) = pose.Pose.Pose.Position.X;
        odom_pose_Y(1,j) = pose.Pose.Pose.Position.Y;

        delta_x = odom_pose_X(1,j)-odom_pose_X(1,j-1);
        delta_y = odom_pose_Y(1,j)-odom_pose_Y(1,j-1);
        delta_distance = delta_x^2+delta_y^2;
        running_distance = running_distance + delta_distance;
        fprintf(" The robot have run %f meters. \n", running_distance);

 % This section is used to receive the RIDAR information and control the
 % robot.My idea is dividing task into three basic conditions, and in each
```

```
% conditions, adding some subconditions
       laser_msg=receive(laser);
       laser_msg.Ranges(laser_msg.Ranges==-Inf)=Inf;
       distance = laser_msg.Ranges;

% This part is used to calculate the mean value about 11 LIDAR lines
% feedback value, which are from 535 to 545(The front conditions),
% preparing for the later controling. Using the mean value
% will make the running more smoothly.
       sum1 = 0;
       for i = (540-5):(540+5)
           sum1 = sum1 + distance(i);
           length1 = sum1 / 11;
       end

% This part is used to calculate the mean value about 7 LIDAR lines
% feedback value, which are from 897 to 903 (The right side conditions),
% preparing for the later controling. Using the mean value
% will make the running more smoothly.
       sum2 = 0;
       for i = (index1-3):(index1+3)
             sum2 = sum2 + distance(i);
             length2 = sum2 / 7;
       end

% This part is used to control the robot moving; My mean idea is
% 1. Judging the front condition, if the distance is larger than 2.5 and
% less than 100(The value is simulating the Infinite value). This means
% the robot is running along a straight line. So, if the robot leans to the
% left, changing the angular rate to make the robot turning to the right.
% If the robot leans to the right, changing the angular velocity to make
% the robot turning to the left. Otherwise, running in a stright line. The
% judging distance is 2.01 meters, which gives the robot a 0.02 tolerance.
       if length1 >3 && length1 <= 100
           1
               if length2 > 2.01
                   11
               velmsg.Linear.X = 0.3;
```

```
                velmsg.Angular.Z = -0.18;
                send(robot,velmsg);
                elseif length2 < 2.01
                    12
                velmsg.Linear.X = 0.3;
                velmsg.Angular.Z = 0.18;
                send(robot,velmsg);
                else
                    13
                velmsg.Linear.X = 0.3;
                velmsg.Angular.Z = 0;
                send(robot,velmsg);
                end

% 2. If the front distance is less than 2.5, the situation is that the
% robot are facing the wall and need to rotate a big angle to avoiding
% hit on the wall.
      elseif length1 <= 3
          2
              velmsg.Linear.X = 0.3;
              velmsg.Angular.Z = 0.35;
              send(robot,velmsg);

% 3. The last situation is when the front distance is Infinite, which
% means there is no barrier in the front. Then let the robot to judge the
% lateral conditions. If the lateral sides also doesn't have the barrier,
% the situation means that the robot is at the corner, and it should turn
% around. But if there are also some obstacles on the side. This is means
% that the robot is still running along the wall, but the wall is very
% long, the robot should adjust its angular velocity in a small range.
      else
          if length2 == Inf
              31
              velmsg.Linear.X = 0.3;
              velmsg.Angular.Z = -0.35;
              send(robot,velmsg);
          else
              32
```

```
                if length2 > 2.01
                    321
                velmsg.Linear.X = 0.3;
                velmsg.Angular.Z = -0.18;
                send(robot,velmsg);
                elseif length2 < 2.01
                    322
                velmsg.Linear.X = 0.3;
                velmsg.Angular.Z = 0.18;
                send(robot,velmsg);
                else
                    323
                velmsg.Linear.X = 0.3;
                velmsg.Angular.Z = 0;
                send(robot,velmsg);
                end
            end
        end
end
```

Task A: Outside of the given "island"- keep the wall to the left

Setting the initial position as position: x: -10.0 y: -6.0 z: 0.0 orientation: x: 0.0 y: 0.0 z: -0.7 w: 0.7

```
% The second task used the same idea as the first one. But the codes should
% change the angular velocity directioins in different situations.
clear all
close all
clc

%shut down the executing ros
rosshutdown;
%Initialize ROS
ipaddress='localhost';
```

```
rosinit(ipaddress);

%Reset Odometry
reset_odom=rospublisher('/mobile_base/commands/reset_odometry');
reset_msg=rosmessage(reset_odom);
send(reset_odom,reset_msg);

%Create the publishers and subscribers
robot = rospublisher('/mobile_base/commands/velocity');
odom_subs = rossubscriber('/odom');
model_subs=rossubscriber('/gazebo/model_states');
laser=rossubscriber('/laserscan');

%Create the message with the right format for the topic
velmsg = rosmessage(robot);
pose = rosmessage(odom_subs);
model=rosmessage(model_subs);
laser_msg=rosmessage(laser);

index_incre = 270/1080;
index1 = 90/index_incre + 540;

running_distance = 0;
j=1;
odom_pose_X(1,j)=0;
odom_pose_Y(1,j)=0;

m = 1;
while m==1
        model=receive(model_subs);
        pose=receive(odom_subs);

        position_X = model.Pose(34,1).Position.X;
        position_Y = model.Pose(34,1).Position.Y;

        hold on
        axis equal
        plot(position_X,position_Y,'.');
```

```
j=j+1;
pose=receive(odom_subs);
odom_pose_X(1,j) = pose.Pose.Pose.Position.X;
odom_pose_Y(1,j) = pose.Pose.Pose.Position.Y;

delta_x = odom_pose_X(1,j)-odom_pose_X(1,j-1);
delta_y = odom_pose_Y(1,j)-odom_pose_Y(1,j-1);
delta_distance = delta_x^2+delta_y^2;
running_distance = running_distance + delta_distance;
fprintf(" The robot have run %f meters. \n", running_distance);

laser_msg=receive(laser);
laser_msg.Ranges(laser_msg.Ranges==-Inf)=Inf;
distance = laser_msg.Ranges;

sum1 = 0;
for i = (540-5):(540+5)
    sum1 = sum1 + distance(i);
    length1 = sum1 / 11;
end
sum2 = 0;
for i = (index1-3):(index1+3)
      sum2 = sum2 + distance(i);
      length2 = sum2 / 7;
end
if length1 >2.5 && length1 <= 100
        if length2 > 2.01
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = 0.18;
        send(robot,velmsg);
        elseif length2 < 2.01
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = -0.18;
        send(robot,velmsg);
        else
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = 0;
```

```
            send(robot,velmsg);
            end
        elseif length1 <= 2.5
            velmsg.Linear.X = 0.15;
            velmsg.Angular.Z = -0.35;
            send(robot,velmsg);
        else
            if length2 == Inf
                velmsg.Linear.X = 0.15;
                velmsg.Angular.Z = 0.35;
                send(robot,velmsg);
            else
                if length2 > 2.01
                velmsg.Linear.X = 0.15;
                velmsg.Angular.Z = 0.18;
                send(robot,velmsg);
                elseif length2 < 2.01
                velmsg.Linear.X = 0.15;
                velmsg.Angular.Z = -0.18;
                send(robot,velmsg);
                else
                velmsg.Linear.X = 0.15;
                velmsg.Angular.Z = 0;
                send(robot,velmsg);
                end
            end
        end

end
```

Task B: Inside of the given "island"- keep the wall to the left

Setting the initial position as position: x: 0.0 y: 0.0 z: 0.0 orientation: x: 0.0 y: 0.0 z: 0.0 w: 0.0

```
% The third task used the same idea as the first one. But the codes should
% change the angular velocity directioins in different situations.
clear all
close all
clc

%shut down the executing ros
rosshutdown;
%Initialize ROS
ipaddress='localhost';
rosinit(ipaddress);

%Reset Odometry
reset_odom=rospublisher('/mobile_base/commands/reset_odometry');
reset_msg=rosmessage(reset_odom);
send(reset_odom,reset_msg);

%create the publishers and subscribers
robot = rospublisher('/mobile_base/commands/velocity');
odom_subs = rossubscriber('/odom');
model_subs=rossubscriber('/gazebo/model_states');
laser=rossubscriber('/laserscan');

%create the message with the right format for the topic
velmsg = rosmessage(robot);
pose = rosmessage(odom_subs);
model=rosmessage(model_subs);
laser_msg=rosmessage(laser);

index_incre = 270/1080;
index1 = 90/index_incre + 540;

running_distance = 0;
j=1;
odom_pose_X(1,j)=0;
odom_pose_Y(1,j)=0;

m = 1;
```

```
while m==1
        model=receive(model_subs);
        pose=receive(odom_subs);

        position_X = model.Pose(34,1).Position.X;
        position_Y = model.Pose(34,1).Position.Y;

        hold on
        axis equal
        plot(position_X,position_Y,'.');

        j=j+1;
        pose=receive(odom_subs);
        odom_pose_X(1,j) = pose.Pose.Pose.Position.X;
        odom_pose_Y(1,j) = pose.Pose.Pose.Position.Y;

        delta_x = odom_pose_X(1,j)-odom_pose_X(1,j-1);
        delta_y = odom_pose_Y(1,j)-odom_pose_Y(1,j-1);
        delta_distance = delta_x^2+delta_y^2;
        running_distance = running_distance + delta_distance;
        fprintf(" The robot have run %f meters. \n", running_distance);

        laser_msg=receive(laser);
        laser_msg.Ranges(laser_msg.Ranges==-Inf)=Inf;
        distance = laser_msg.Ranges;

        sum1 = 0;
        for i = (540-5):(540+5)
            sum1 = sum1 + distance(i);
            length1 = sum1 / 11;
        end
        sum2 = 0;
        for i = (index1-3):(index1+3)
              sum2 = sum2 + distance(i);
              length2 = sum2 / 7;
        end
        if length1 >2.5 && length1 <= 100
                if length2 > 2.01
```

```
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = 0.18;
        send(robot,velmsg);
        elseif length2 < 2.01
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = -0.18;
        send(robot,velmsg);
        else
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = 0;
        send(robot,velmsg);
        end
elseif length1 <= 2.5
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = -0.35;
        send(robot,velmsg);
else
    if length2 == Inf
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = 0.35;
        send(robot,velmsg);
    else
        if length2 > 2.01
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = 0.18;
        send(robot,velmsg);
        elseif length2 < 2.01
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = -0.18;
        send(robot,velmsg);
        else
        velmsg.Linear.X = 0.15;
        velmsg.Angular.Z = 0;
        send(robot,velmsg);
        end
    end
end
```

```
end
```

Task B: Inside of the given "island"- keep the wall to the right

Setting the initial position as position: x: 0.0 y: 0.0 z: 0.0 orientation: x: 0.0 y: 0.0 z: 1.0 w: 0.0

```
% The third task used the same idea as the first one. But the codes should
% change the angular velocity directioins in different situations.
clear all
close all
clc

%shut down the executing ros
rosshutdown;
%Initialize ROS
ipaddress='localhost';
rosinit(ipaddress);

%Reset Odometry
reset_odom=rospublisher('/mobile_base/commands/reset_odometry');
reset_msg=rosmessage(reset_odom);
send(reset_odom,reset_msg);

%create the publishers and subscribers
robot = rospublisher('/mobile_base/commands/velocity');
odom_subs = rossubscriber('/odom');
model_subs=rossubscriber('/gazebo/model_states');
laser=rossubscriber('/laserscan');

%create the message with the right format for the topic
velmsg = rosmessage(robot);
pose = rosmessage(odom_subs);
model=rosmessage(model_subs);
laser_msg=rosmessage(laser);
```

```
index_incre = 270/1080;
index1 = -90/index_incre + 540;

running_distance = 0;
j=1;
odom_pose_X(1,j)=0;
odom_pose_Y(1,j)=0;

m = 1;
while m==1
        model=receive(model_subs);
        pose=receive(odom_subs);

        position_X = model.Pose(34,1).Position.X;
        position_Y = model.Pose(34,1).Position.Y;

        hold on
        axis equal
        plot(position_X,position_Y,'.');

        j=j+1;
        pose=receive(odom_subs);
        odom_pose_X(1,j) = pose.Pose.Pose.Position.X;
        odom_pose_Y(1,j) = pose.Pose.Pose.Position.Y;

        delta_x = odom_pose_X(1,j)-odom_pose_X(1,j-1);
        delta_y = odom_pose_Y(1,j)-odom_pose_Y(1,j-1);
        delta_distance = delta_x^2+delta_y^2;
        running_distance = running_distance + delta_distance;
        fprintf(" The robot have run %f meters. \n", running_distance);

        laser_msg=receive(laser);
        laser_msg.Ranges(laser_msg.Ranges==-Inf)=Inf;
        distance = laser_msg.Ranges;

        sum1 = 0;
        for i = (540-5):(540+5)
```

```
    sum1 = sum1 + distance(i);
    length1 = sum1 / 11;
end
sum2 = 0;
for i = (index1-3):(index1+3)
      sum2 = sum2 + distance(i);
      length2 = sum2 / 7;
end
if length1 >2.5 && length1 <= 100
        if length2 > 2.01
        velmsg.Linear.X = 0.3;
        velmsg.Angular.Z = -0.18;
        send(robot,velmsg);
        elseif length2 < 2.01
        velmsg.Linear.X = 0.3;
        velmsg.Angular.Z = 0.18;
        send(robot,velmsg);
        else
        velmsg.Linear.X = 0.3;
        velmsg.Angular.Z = 0;
        send(robot,velmsg);
        end
elseif length1 <= 2.5
        velmsg.Linear.X = 0.3;
        velmsg.Angular.Z = 0.35;
        send(robot,velmsg);
else
    if length2 == Inf
        velmsg.Linear.X = 0.3;
        velmsg.Angular.Z = -0.35;
        send(robot,velmsg);
    else
        if length2 > 2.01
        velmsg.Linear.X = 0.3;
        velmsg.Angular.Z = -0.18;
        send(robot,velmsg);
        elseif length2 < 2.01
        velmsg.Linear.X = 0.3;
```

```
                velmsg.Angular.Z = 0.18;
                send(robot,velmsg);
                else
                velmsg.Linear.X = 0.3;
                velmsg.Angular.Z = 0;
                send(robot,velmsg);
                end
            end
        end

end
```