# ME 425 Lab #3 Post Lab Report

Arda Gencer
34124

Fall 2025 - 2026

# 1 Introduction

In this lab, an introduction to perception and sensors for mobile robots were made with the LiDAR sensor on the Turtlebot3. During the lab, the LiDAR was used for several purposes such as getting an instantaneous and live reading of the Turtlebot's environment and plotting these readings into a MATLAB plot. A more practical application was also developed by making a collision avoiding algorithm for the Turtlebot. With this algorithm, the Turtlebot would stop if it sensed any solid object in its path during operation. It would than change its trajectory if the object didn't move.

# 2 Procedure

At the start of the lab, the connection to the Turtlebot ( ssh connection and inside MATLAB) was made as similar to the previous labs and a node was created. Then a LiDAR subscriber was created.

Then, the LiDarScan.m was implemented as follows: A single message was received from the LiDar subscriber. After this, the range information for ach angle and the angles themselves were extracted as seperate vectors. Then, the polar to cartesian conversion was made by using the element-wise multiplication operator ".*". After this, the coordinates will be in cartesian form. They were then plotted vs. each other on a MATLAB plot and a title, axis labels and a point showing the location of the Turtlebot was also added. The program was then tested with 2 different environments. One with an open environment with almost no obstacles around and one with a more closed environment with tables, chairs etc. around.

After the first task, the code was copied and modified to implement the LiveLiDAR.m program. To do this, the code piece that gets the lidar message, extracts the information and converts these data to cartesian coordinates were copied. They were then ran inside a infinite while loop so that these tasks would be performed once every 0.05 seconds. The 0.05 seconds time was selected manually as the Lidar sampling time. Inside the code, a line was added to make the script wait for this amount of time before running the loop again. As for plotting, a single plot was created at the start of the program without actually plotting anything. This plot was then updated at each iteration of the while loop using the most recent data received from the Lidar. The program was then tested by manually controlling the Turtlebot with the keyboard via the powershell terminal. A live demonstration of this program where the plot rotates when the robot is given a angular velocity can be found in the "Results" section of the report.

For the last task, a collision avoiding mechanism was implemented inside SafeDistance.m. To do this, an additional velocity publisher was used along with the Lidar. The mechanism works as follows: The robot will have a forward linear velocity of 0.1 m/s as default. It will continuously scan the environment as it goes forward. If the Lidar detects any object that is closer than 0.55 m in the front 90 degrees periphery of the Turtlebot, an if condition inside the loop is activated and the robot stops. After this point, an internal clock begins ticking. If at any point, the obstacle is removed, the timer resets and the robot will continue its course as normal. However, if this clock reaches approximately 5 seconds, then the robot will turn 90 degrees counter-clockwise, the clock will reset, and the robot will try to move forward again if no obstacle blocks its path. The algorithm goes on like this until terminated. A live demonstration of this can be found via the link in the "Results" section.

# 3 Results

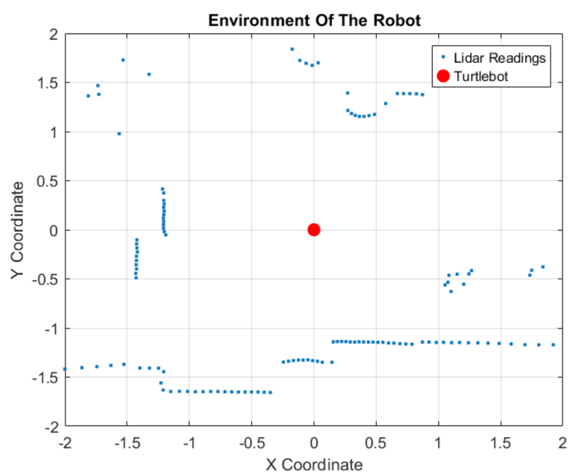


Figure 1: The LiDAR scan of the environemnt with the Turtlebot placed in an open environment

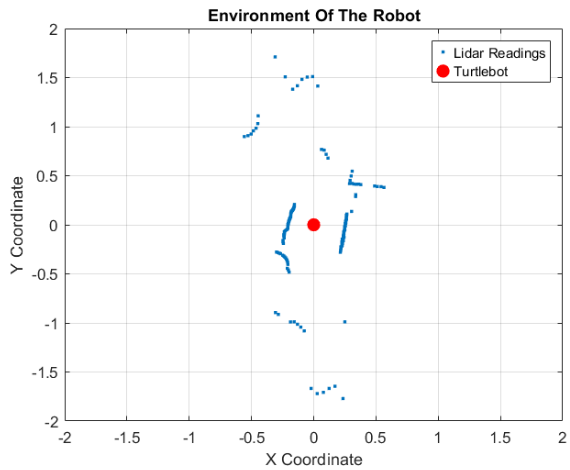


Figure 2: The LiDAR scan of the environemnt with the Turtlebot placed in an environment with obstacles

The link to the demonstration of LiveLiDAR.m: https://drive.google.com/file/d/1Sl_6zO-NVGmXevIHCiTwWBRxbXnzL2D-/view?usp=sharing The link to the demonstration of SafeDistance.m: https://drive.google.com/file/d/1zANaaLWd-CWhrdayO3fTz4urlWbdi_qd/view?usp=sharing

# 4 Conclusion

In this lab, a solid introduction to the basic concepts of the Lidar sensor were made. The simple usage of the sensor and its possible applications combined with the other components of the Turtlebot were also demonstrated via certain applications such as the collision avoidance system.

# 5 Discussion

1. **What do your results look like when you scan the environment without any obstacles? Comment on your result**

   The readings came out pretty empty, except for the walls of the environment we tested the robot, which were about 1-1.5 m away from the robot.

2. **Once you add an obstacle, how does your resultant plot change? What do the points around the obstacle look like and why?**

**Visualize the detected objects on the Cartesian plot.** Once the robot was placed in an environment with more obstacles, those obstacles showed up on the cartesian plane. The Lidar could detect almost all the solid objects like feet of people, chairs etc..

3. **How does the LiDAR detect reflective or absorptive materials? How do the shape and the colour of the obstacle affect the measurements?**

   For reflective materials, the angle that the material is compared with the robot may cause the beam of light emitted from the robot to be reflected to a different direction, causing for the object to be not detected. However, when they are correctly placed, reflective materials will return the light at a much higher percentage compared to absorptive materials. For absorptive materials, they don't have the problem of reflecting the signal away as much as reflective materials, however , they can only return a small proportion of the light back to the sensor.
   This is where geometry also plays a role, flat objects will tend to reflect the light back to the sensor much better compared to curvy objects or objects that have a tilt relative to the robot.
   The color on the other hand, doesn't have any effect on the readings since Lidar uses infrared light, which is outside of the frequency range of visible light.

4. **What is the maximum range of LiDAR? What happens when a beam returns Inf, and how would you solve this?**

   The maximum range of the Lidar on the Turtlebot (LDS-02) is 8 meters. If there is no object on a specific side of the robot, the Lidar may not return to the robot. In this case, the reading will be recorded as something like "Inf" or "NaN (Not a Number)". To prevent these readings from messing up the results, a filter is applied to the readings of the data to get rid of such readings that have Inf or NaN values.

5. **How did your implemented SafeDistance.m work? Comment on your results**

   The implemented algorithm worked quite well. By tweaking the safe distance by hand, the robot could be stopped even if something suddenly appeared in its path. The turning mechanism was also functioning quite effectively and as planned. Although in the Lab document, it was requested that the robot should only read lidar data that is directly in front of it, a range of 45 degrees to both sides of the robot was used so that the robot could see obstacles that stood slightly to the side as well.

# 6 Appendix (All MATLAB Scripts That Were Used):

Listing 1: LiDarScan.m. Used to get a single scan of the environment and then plot the results onto a cartesian plane, assuming the robot is at the origin.

```
%% Environment Setup
clear; close all; clc;

clear node; % clear previous node handle if it exists

setenv('ROS_DOMAIN_ID','43');
setenv('ROS_LOCALHOST_ONLY', '0'); % 0 implies multi-host communication
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware

% Create a unique node name for this MATLAB session
node = ros2node('Lidar');

%% Creating the Lidar Sub
%Lidar Subscriber
lidarSub = ros2subscriber(node, "/scan", "sensor_msgs/LaserScan",...
"Reliability", "besteffort", "Durability", "volatile", "Depth", 10);

%% Recieving the lidar message and extracting the distance and angle
%information
lidarMsg = receive(lidarSub, 10);
ranges = lidarMsg.ranges;
angles = lidarMsg.angle_min:lidarMsg.angle_increment:lidarMsg.angle_max;

%% Extracting the x and y coordinates from distance and angle information
xCoordinates = ranges .* transpose(cos(angles));
yCoordinates = ranges .* transpose(sin(angles));

%% Plotting
figure;
plot(xCoordinates,yCoordinates, '.','DisplayName', 'Lidar␣Readings');
hold on;
plot(0,0, 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 8, 'DisplayName', 'Turtlebot');
legend('show');
xlabel("X Coordinate");
ylabel("Y Coordinate");
title("Environment Of The Robot");
grid on;
axis([-2 2 -2 2]);
```

Listing 2: LiveLiDAR.m. Used to get lidar scans of the environment about every 0.05 seconds, and then plots these results onto the cartesian plane in real time.

```
clear; close all; clc;
clear node; % clear previous node handle if it exists
%% Environment Setup
setenv('ROS_DOMAIN_ID','43');
setenv('ROS_LOCALHOST_ONLY', '0');
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware

% Create a unique node name for this MATLAB session
node = ros2node('LiveLidar');

%% Creating the Lidar Sub
lidarSub = ros2subscriber(node, "/scan", "sensor_msgs/LaserScan",...
"Reliability", "besteffort", "Durability", "volatile", "Depth", 10);

%% Getting Live Lidar and Plotting
T = 0.05;   %Sampling Period
%Creating the figure only once
figure;
hScatter = scatter(nan,nan,'.');
hold on;
plot(0,0, 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 8, 'DisplayName', 'Turtlebot');
legend('show');
xlabel("X Coordinate");
ylabel("Y Coordinate");
title("Environment Of The Robot");
grid on;
axis([-2 2 -2 2]);

while true  %This loop will run continously
lidarMsg = receive(lidarSub, 10);
ranges = lidarMsg.ranges;
angles = lidarMsg.angle_min:lidarMsg.angle_increment:lidarMsg.angle_max;

%Filtering the Lidar Messages
valid = ~isnan(ranges) & ~isinf(ranges) & (ranges > 0);
ranges = ranges(valid);
angles = angles(valid);

%Extracting the x and y coordinates from distance and angle information
xCoordinates = ranges .* transpose(cos(angles));
yCoordinates = ranges .* transpose(sin(angles));

%Updating the plot
set(hScatter, 'XData', xCoordinates, 'YData', yCoordinates, 'DisplayName', 'Lidar Readings');
drawnow limitrate;   % Efficient plotting update
pause(T);   %Pause for sampling time
end
```

Listing 3: SafeDistance.m. Implements a collision avoider algorithm which makes the robot stop and wait for 5 seconds for the obstacle to get out of the way. If the obstacle is still in the way, the robot rotates 90 degrees counter clockwise and resumes motion.

```
%% Environment Setup
clear; close all; clc;

clear node; % clear previous node handle if it exists

setenv('ROS_DOMAIN_ID','43');
setenv('ROS_LOCALHOST_ONLY', '0'); % 0 implies multi-host communication
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware

% Create a unique node name for this MATLAB session
node = ros2node('DistanceChecker');

%% Creating the Lidar and velocity Sub
%Lidar Subscriber
lidarSub = ros2subscriber(node, "/scan", "sensor_msgs/LaserScan",...
"Reliability", "besteffort", "Durability", "volatile", "Depth", 10);

velPub = ros2publisher(node,"/cmd_vel","geometry_msgs/Twist", ...
"Reliability","reliable","Durability","volatile","Depth",10);


%% Creating Lidar Plot
figure;
hScatter = scatter(nan,nan,'.');
hold on;
plot(0,0, 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 8, 'DisplayName', 'Turtlebot');
legend('show');
xlabel("X Coordinate");
ylabel("Y Coordinate");
title("Environment Of The Robot");
grid on;
axis([-2 2 -2 2]);
%% Running the loop for object detection
T = 0.05;   %Sampling Period
timeWaited = 0; %The amount of time waited on current collision


velMsg = ros2message(velPub);
velMsg.linear.x = 0.1;
send(velPub, velMsg);
while true
    %Getting the Lidar Message
    lidarMsg = receive(lidarSub, 10);
    ranges = lidarMsg.ranges;
    angles = lidarMsg.angle_min:lidarMsg.angle_increment:lidarMsg.angle_max;

    %Filtering the Lidar Messages
```

```
    valid = ~isnan(ranges) & ~isinf(ranges) & (ranges > 0);
    ranges = ranges(valid);
    angles = angles(valid);

    %Getting the data that is in front of the robot
    size = length(angles);
    oneEigth = floor(size/8);
    forwardDistances = [ranges(1:oneEigth) ranges(end-oneEigth+1:end)];

    %Extracting the x and y coordinates from distance and angle information

    xCoordinates = ranges .* transpose(cos(angles));
    yCoordinates = ranges .* transpose(sin(angles));

    %Updating the plot
    set(hScatter, 'XData', xCoordinates, 'YData', yCoordinates, 'DisplayName', 'Lidar Readings');
    drawnow limitrate;   % Efficient plotting update

    %Collision Check
    if any(forwardDistances < 0.55) %If any object that is too close is detected
        velMsg.linear.x = 0;
        send(velPub, velMsg);
        timeWaited = timeWaited + T;
        if timeWaited >=1.8   %If the robot was idle for at least 5 seconds
            %Rotate The Robot Here
            velMsg.linear.x = 0;
            velMsg.angular.z = 1;
            send(velPub,velMsg);
            pause(pi/2);
            velMsg.angular.z = 0;
            velMsg.linear.x = 0.1;
            send(velPub, velMsg);
            timeWaited = 0;
        end
    elseif timeWaited ~= 0 %If the robot has been waiting for some time
        timeWaited = 0; %Reset the counter
        %Resend the message
        velMsg.linear.x = 0.1;
        send(velPub, velMsg);
    end

    pause(T);   %Wait for the sampling period
end
```