ME 425 LAB #0 POST LAB REPORT

Arda Gencer

09.10.2025

Sabanci University

INTRODUCTION

In the first lab (Lab #0), an introduction to the TurtleBots (Waffle Pi 3) was made in the lab using the ROS2 development environment through MATLAB. We gained experience on how to boot up the robot, make a connection to the Raspberry Pi computer on the robot and how to set up the ROS environment in MATLAB. In our experiment, we utilized several ROS2 topics that send information from the robot and nodes that receive information from the MATLAB script. This experiment facilitated our familiarization with different sensors on the robots and taught us how to send commands to the robot.

PROCEDURE

1. Connection to the Turtlebot: First of all, the robot itself needs to be turned on. For this, the two ends of the power cable should be connected together to power the robot through its battery. Then, the robot is switched on by the switch on its arduino board. After this, a connection between the PC and turtlebot needs to be established to be able to send and receive information via nodes that will be written in ROS. To do this, the PC must be connected to the Turtlebot via its Wi-fi hotspot using the password information on the robot. Then, an ssh connection to the robot must be established by typing the following command into bash (i.e. command prompt on Windows): ssh me425-1@192.168.XX.1 (instead of XX, replace with actual ip address of the robot). After this, the password given on the robot should also be entered. After this step, a proper ssh connection to the robot will be made. Now, one must bring up the nodes on the robot to be able to publish the sensor data from the Turtlebot. For this, the command “bringup” should be used on the command prompt. After the necessary nodes on the robot start running and publishing information, MATLAB can be used to create new nodes to publish/ receive data from/ to the Turtlebot.
2. Lidar Data: For the first part of the experiment, a ROS2 node responsible for receiving the Lidar and Camera data and processing them was created. Then, a ros2subscriber under the node that was previously created that will listen for messages of type “sensor_msgs/LaserScan” from the Raspberry Pi under the topic name “/scan” was added. Then, using this node, a single read of data points containing ranges for each angle that the Lidar has read was received. These data were then plotted using polar coordinates to create a rough topographic map of the environment.

Camera Data: By creating a ros2subscriber that listens on the topic “/image_raw/compressed” for messages of type “sensor_msgs/CompressedImage” was created. By using this subscriber, the compressed image from the robot’s camera was received by the node. It was then read and displayed on the screen using the functions on MATLAB ROS Toolbox.

Note: The necessary code for these steps can be found in the Appendix section.

3. For the second part of the experiment, a separate node that is responsible for controlling the motors and receiving encoder information from the wheels was created. Then, for receiving encoder information, a subscriber that will listen for messages of type “sensor_msgs/JointState” under the topic “/joint_states” was created. A publisher sending messages of type “geometry_msgs/Twist” targeting the topic “/cmd_vel” was also created. After creating the necessary subscribers and publishers; the robot was sent commands prompting the robot to move forward with a speed of 0.2 m/s, backwards with a speed of 0.2 m/s and to rotate with an angular velocity of +1 rad/s. All three of these commands were run for 5 seconds each and encoder data was collected, formed into a vector and plotted for each wheel during the entire operation.

   Note: The necessary code for these steps can be found in the Appendix section.

RESULTS

Fig. 1: The image received from the camera

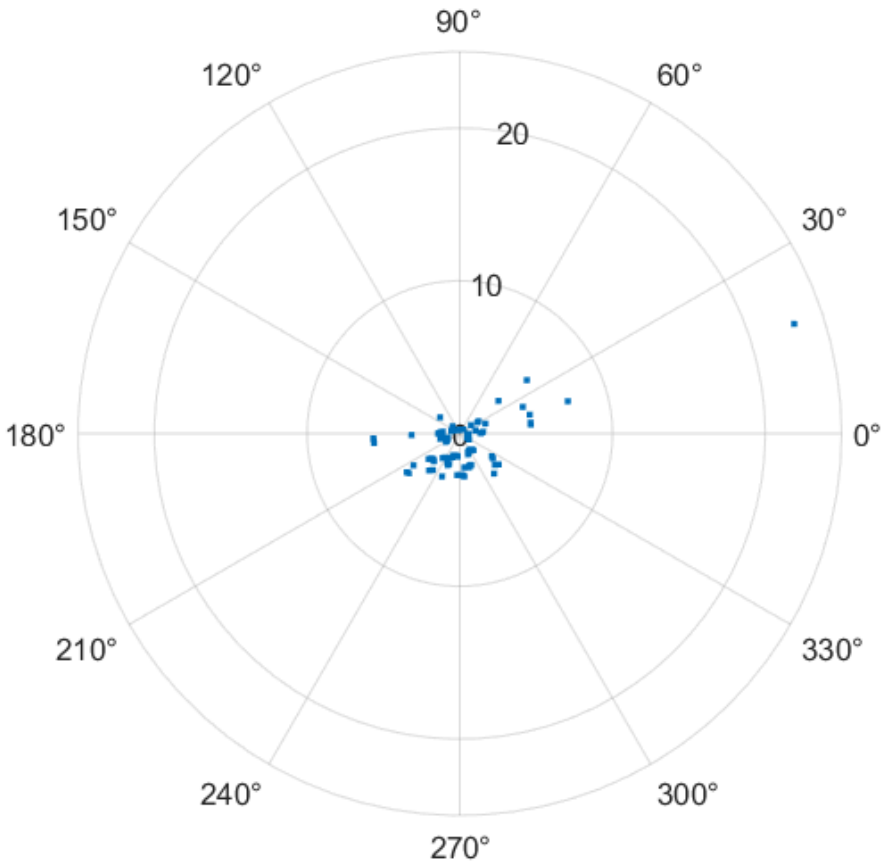

Fig. 2: The Lidar Mapping Received from the Lidar Sensor on the Turtlebot

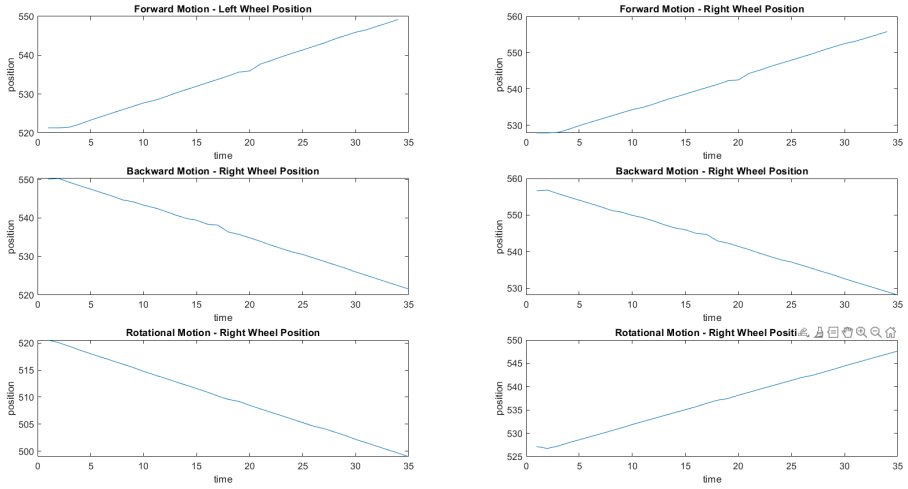

Fig. 3: The wheel positions plots for each motion type

CONCLUSION

By the end of the experiment, a solid introduction to the basics of the Turtlebot 3 Waffle Pi was made. We gained basic knowledge on how to boot up and connect the Turtlebot to the PC, how to establish a connection and boot up the nodes on the Turtlebot, how to read and use the information coming from the sensors and also how to command the Turtlebot to move around and rotate.

DISCUSSION

- The data that was collected from the Lidar can provide us a rough mapping of the environment that the robot is currently in. This is because Lidar detects the distances of solid objects around the sensor. If we plot these using polar coordinates, we can see a rough shape of the environment in sonar like mapping style.
- The camera can be used to make the robot have a clearer understanding of what is currently going on in front of itself. Using this data, combined with computer vision algorithms, we can make the robot make specific decisions based on the thing it sees in front of itself. For example, we can make an autonomous car stop when it sees a pedestrian crossing the street.
- The encoder values show that the robot moves correctly for each motion type. For forward motion, both position values are increasing at the same rate. For backward motion, they decrease at the same rate. For rotational motion, the right wheel increases while the left wheel decreases. Also, the absolute value of their rate of change is the same, meaning that it doesn't move anywhere whilst rotating.

APPENDIX

1. `%cameraLidar.m`
2. `clear; close all; clc;`
3. `clear node; % clear previous node handle if it exists`
4. `setenv('ROS_DOMAIN_ID','39'); % Set to your robots ROS_DOMAIN_ID (check the robot's ID card)`

```
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('matlabNode');
lidarSub = ros2subscriber(node, "/scan", "sensor_msgs/LaserScan",...
"Reliability", "besteffort", "Durability", "volatile", "Depth", 10);
camSub =
ros2subscriber(node,"/image_raw/compressed","sensor_msgs/CompressedImage"
, ...
"Reliability","besteffort","Durability","volatile","Depth",10);
lidarMsg = receive(lidarSub, 10);
ranges = lidarMsg.ranges;
angles = lidarMsg.angle_min:lidarMsg.angle_increment:lidarMsg.angle_max;
imgMsg = receive(camSub, 10);
img  = rosReadImage(imgMsg);
imshow(img);
polarplot(ranges,angles, '.');
```

```
%controlledMotion.m
clear node;
setenv('ROS_DOMAIN_ID','39'); % Set to your robots ROS_DOMAIN_ID (check
the robot's ID card)
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("MotorNode");
exeTime = 5; % Execution time in seconds
period = 0.1; % Sampling period in seconds
velPub = ros2publisher(node,"/cmd_vel","geometry_msgs/Twist", ...
"Reliability","reliable","Durability","volatile","Depth",10);
encSub = ros2subscriber(node,"/joint_states","sensor_msgs/JointState",
...
"Reliability","besteffort","Durability","volatile","Depth",10);
velMsg = ros2message(velPub);
t0 = tic;
```

```
iteration = 1;
leftPosF = [];
rightPosF = [];
leftPosB = [];
rightPosB = [];
leftPosR = [];
rightPosR = [];
while toc(t0) < exeTime
velMsg.linear.x = 0.2;
velMsg.angular.z = 0;
send(velPub, velMsg);
%Forward
encMsg = receive(encSub,10);
jointNames = string(encMsg.name);
idxL = find(jointNames=="wheel_left_joint" | jointNames=="left_wheel",1);
idxR = find(jointNames=="wheel_right_joint" |
jointNames=="right_wheel",1);
leftWheelPos = encMsg.position(idxL);
rightWheelPos = encMsg.position(idxR);
leftWheelVel = encMsg.velocity(idxL);
rightWheelVel = encMsg.velocity(idxR);
leftPosF(iteration) = leftWheelPos;
rightPosF(iteration) = rightWheelPos;
iteration = iteration + 1;
pause(period); % Wait before next command
end
%Backwards
iteration = 1;
t0 = tic;
while toc(t0) < exeTime
velMsg.linear.x = -0.2;
velMsg.angular.z = 0;
send(velPub, velMsg);
encMsg = receive(encSub,10);
jointNames = string(encMsg.name);
idxL = find(jointNames=="wheel_left_joint" | jointNames=="left_wheel",1);
idxR = find(jointNames=="wheel_right_joint" |
```

```
    jointNames=="right_wheel",1);
52. leftWheelPos = encMsg.position(idxL);
53. rightWheelPos = encMsg.position(idxR);
54. leftWheelVel = encMsg.velocity(idxL);
55. rightWheelVel = encMsg.velocity(idxR);
56. leftPosB(iteration) = leftWheelPos;
57. rightPosB(iteration) = rightWheelPos;
58. iteration = iteration + 1;
59. pause(period); % Wait before next command
60. end
61. %Rotate
62. iteration = 1;
63. t0 = tic;
64. while toc(t0) < exeTime
65. velMsg.linear.x = 0;
66. velMsg.angular.z = 1;
67. send(velPub, velMsg);
68. encMsg = receive(encSub,10);
69. jointNames = string(encMsg.name);
70. idxL = find(jointNames=="wheel_left_joint" | jointNames=="left_wheel",1);
71. idxR = find(jointNames=="wheel_right_joint" |
    jointNames=="right_wheel",1);
72. leftWheelPos = encMsg.position(idxL);
73. rightWheelPos = encMsg.position(idxR);
74. leftWheelVel = encMsg.velocity(idxL);
75. rightWheelVel = encMsg.velocity(idxR);
76. leftPosR(iteration) = leftWheelPos;
77. rightPosR(iteration) = rightWheelPos;
78. iteration = iteration + 1;
79. pause(period); % Wait before next command
80. end
81. velMsg.linear.x = 0;
82. velMsg.angular.z = 0;
83. send(velPub, velMsg);
84.
85. figure;
86. %Forward Motion
```

```
subplot(3,2,1);
plot(leftPosF);
title("Forward Motion - Left Wheel Position");
ylabel("position");
xlabel("time");
subplot(3,2,2);
plot(rightPosF);
title("Forward Motion - Right Wheel Position");
ylabel("position");
xlabel("time");
%Backward Motion
subplot(3,2,3);
plot(leftPosB);
title("Backward Motion - Right Wheel Position");
ylabel("position");
xlabel("time");
subplot(3,2,4);
plot(rightPosB);
title("Backward Motion - Right Wheel Position");
ylabel("position");
xlabel("time");
%Rotational Motion
subplot(3,2,5);
plot(leftPosR);
title("Rotational Motion - Right Wheel Position");
ylabel("position");
xlabel("time");
subplot(3,2,6);
plot(rightPosR);
title("Rotational Motion - Right Wheel Position");
ylabel("position");
xlabel("time");
```