

# ME 425 Lab#1 Post Lab Report

Arda Gencer  
34124

October 19, 2025

## 1 Introduction

In lab #1 of ME425, certain applications concerning odometric estimation were implemented on Turtlebots. A node that allows the Turtlebot to use the equations that were derived during the lectures to integrate its position and save these values to estimate where it went were implemented. This approach was then tested using different movement patterns as indicated in the lab document. Finally, a comparison between this estimated data and desired trajectory was also made.

## 2 Procedure

### 2.1 Odometric Estimation

At the start of the experiment, the Turtlebot was connected to the computer as before following the same steps outlined in the Lab#0 document. After a connection is made, the environment can be set up in MATLAB. Inside MATLAB the environment variables are set and then a node called odometry is created.

After this, the velocity publisher (velPub) to send velocity commands to the Turtlebot and encoder subscriber to receive messages from the wheel encoders (encSub) are created.

Before the start of the movement, initial encoder values are read and recorded using the encSub (these will be the old encoder data. They will be referred to as  $w_r(k)$  and  $w_l(k)$  for right and left wheels respectively. After this, our parameters such as execution time and R (wheel radius) and L (wheel separation) values are defined. Initial  $x$ ,  $y$  and  $\theta$  values are set as 0. The vectors that will be used to record the estimated coordinates are also defined here.

After this, each movement type is initiated one by one and odometric estimation is made at each time step. In total, 7 different types of movement is made. The movement types and linear ( $v$ ) and angular ( $\omega$ ) velocities used with them is as follows:

1. Forward Motion With High Speed:  $v = 0.2 \text{ m/s}, \omega = 0$
2. Forward Motion With Low Speed:  $v = 0.1 \text{ m/s}, \omega = 0$
3. Backward Motion With High Speed:  $v = -0.2 \text{ m/s}, \omega = 0$
4. Backward Motion With Low Speed:  $v = -0.1 \text{ m/s}, \omega = 0$
5. Circular Motion Towards Left:  $v = 0.075 \text{ m/s}, \omega = 0.09 \text{ rad/s}$
6. Circular Motion Towards Right:  $v = 0.075 \text{ m/s}, \omega = -0.09 \text{ rad/s}$
7. Combined/Mixed Motion: Make a forward motion for 3 seconds ( $v = 0.1 \text{ m/s}, \omega = 0$ ), then a circular motion for 3 seconds ( $v = 0.05 \text{ m/s}, \omega = 0.1 \text{ rad/s}$ ), and finally another circular motion for 3 seconds ( $v = -0.1 \text{ m/s}, \omega = -0.05 \text{ rad/s}$ ).

To estimate the trajectory of the Turtlebot; the following operations are done during each time step:

1. The encoder information is read. This is the new encoder information. We will denote them as  $w_r(k+1)$  and  $w_l(k+1)$  for right and left wheels.
2. Then, the odometric information is made by using the following formulae:

$$\Delta E_r = w_r(k+1) - w_r(k), \quad \Delta E_l = w_l(k+1) - w_l(k)$$

$$D_r = R\Delta E_r, \quad D_l = R\Delta E_l$$

$$D_c = \frac{D_r + D_l}{2}$$

$$\Delta\theta = \frac{D_r - D_l}{2}$$

$$x(k+1) = x(k) + D_c \cos(\theta(k) + \frac{\Delta\theta}{2})$$

$$y(k+1) = y(k) + D_c \sin(\theta(k) + \frac{\Delta\theta}{2})$$

$$\theta(k+1) = \theta(k) + \Delta\theta$$

These new pose data is then recorded to the appropriate vector for each type of movement. Before ending the loop, the old encoder data information is also replaced with the new ones. This estimation is then done for each kind of motion listed above.

## 2.2 Desired Trajectory

Next up, the desired trajectory (i.e. the trajectory that would theoretically be taken with the given amount of velocity during the execution time) needs to be calculated. For this, again it is needed go through each movement type one by one and record the pose data to different arrays. The calculations for the desired trajectory are done as follows:

1. First, the needed parameters are defined as follows:  
 $x(k+1)$ ,  $y(k+1)$ ,  $\theta(k+1)$  : The pose data at k+1 th time step  
 $x(k)$ ,  $y(k)$ ,  $\theta(k)$  : The pose data at k th time step  
 $\omega$  : The angular velocity of robot  
 $v$  : The linear velocity of robot  
 $\Delta t$  : The time difference between two consecutive time steps

2. Then the desired trajectory is computed as follows:

$$x(k+1) = x(k) + v\Delta t \cos \theta$$

$$y(k+1) = y(k) + v\Delta t \sin \theta$$

$$\theta(k+1) = \theta(k) + \omega\Delta t$$

3. Before recording the new values to the vector, some data points are cut for the rotation data as to match the number of data points with those that were received from the encoders for better plotting.
4. The new pose values are then recorded to the corresponding vectors.

## 2.3 Data Plotting

The received data is then plotted into separate figures for each type of motion. Both the estimated and desired trajectory is plotted both in terms of their coordinates and also in terms of their rotation values vs time. The said plots can be found in the results section.

### 3 Results

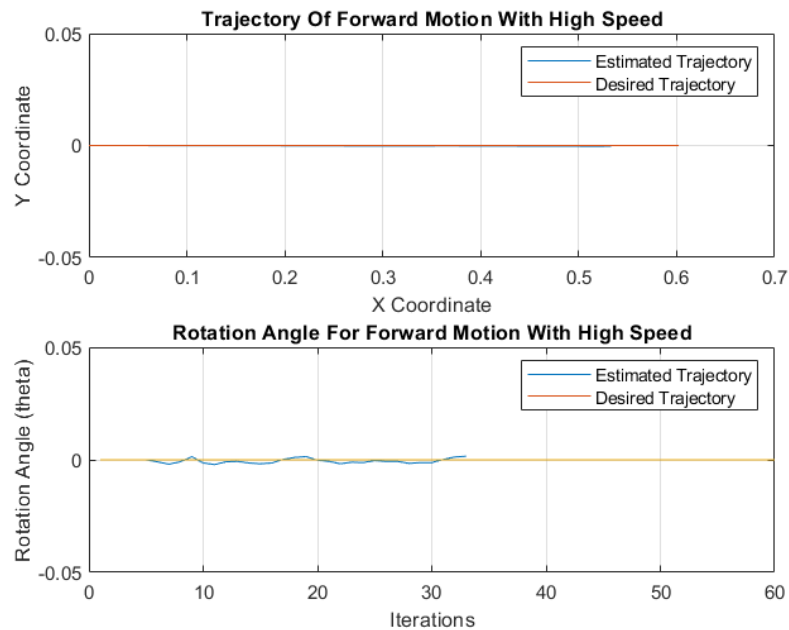


Figure 1: Position And Rotation Plot For Forward Motion With High Speed

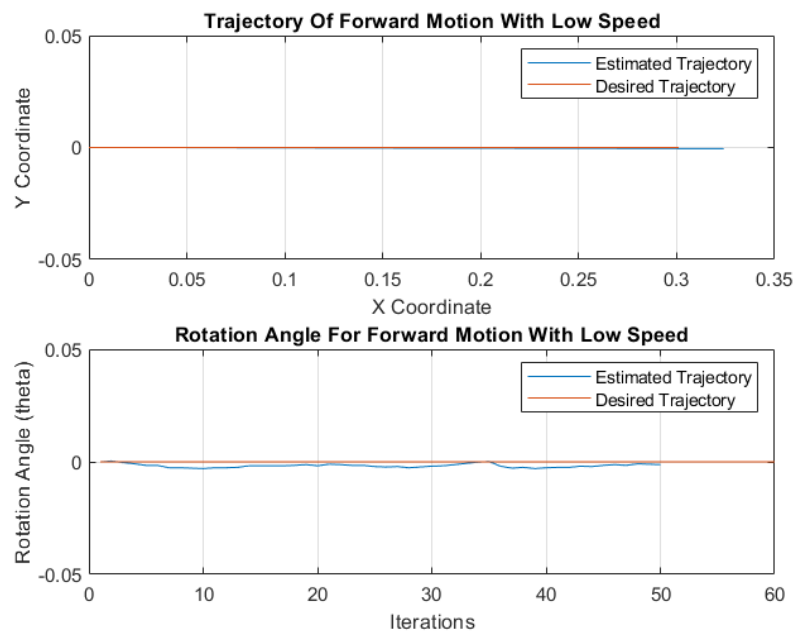


Figure 2: Position And Rotation Plot For Forward Motion With Low Speed

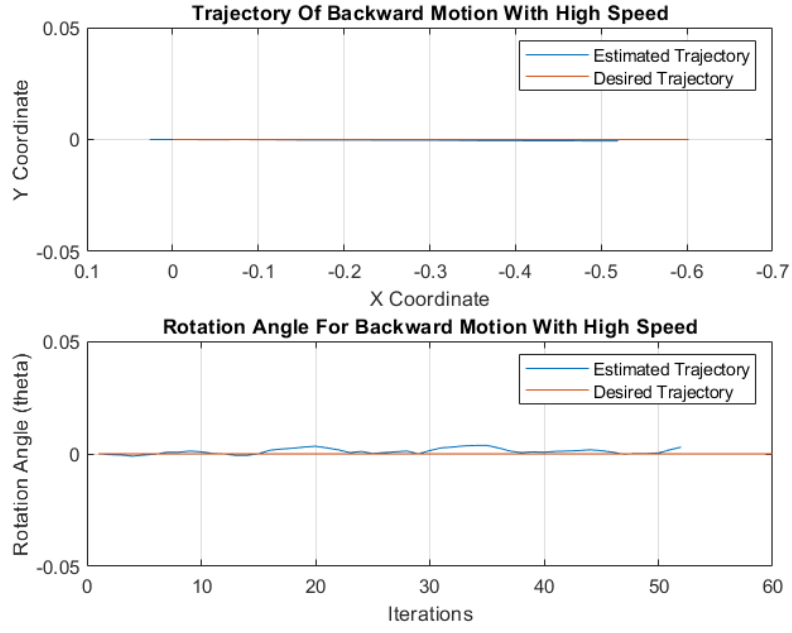


Figure 3: Position And Rotation Plot For Backward Motion With High Speed

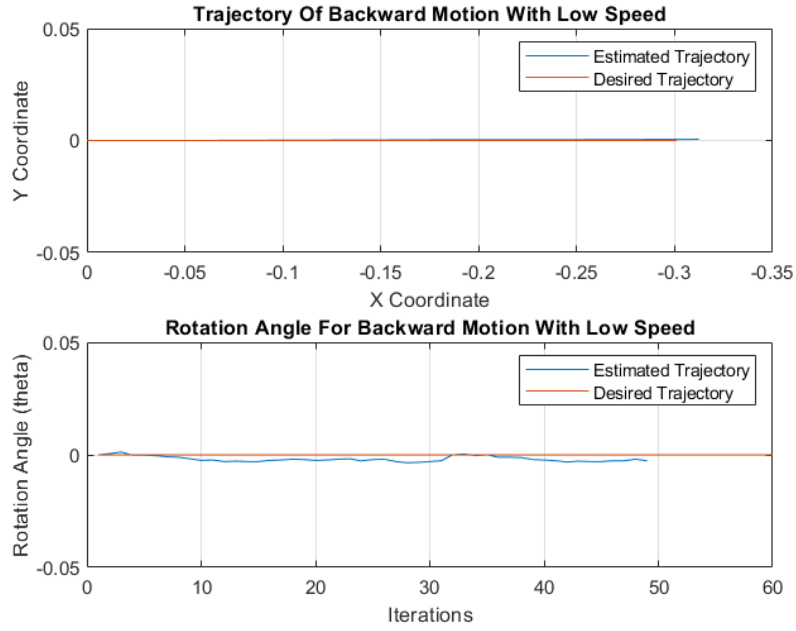


Figure 4: Position And Rotation Plot For Backward Motion With Low Speed

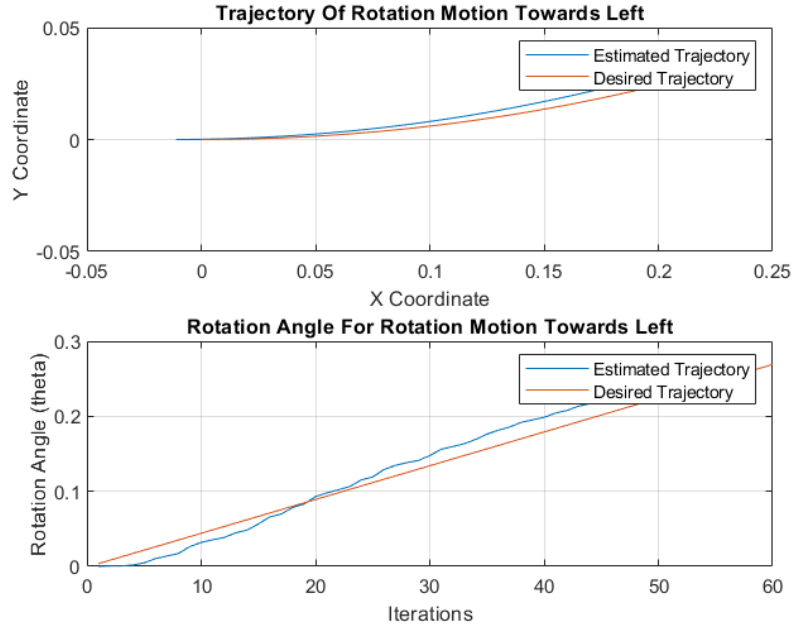


Figure 5: Position And Rotation Plot For Left Circular Motion

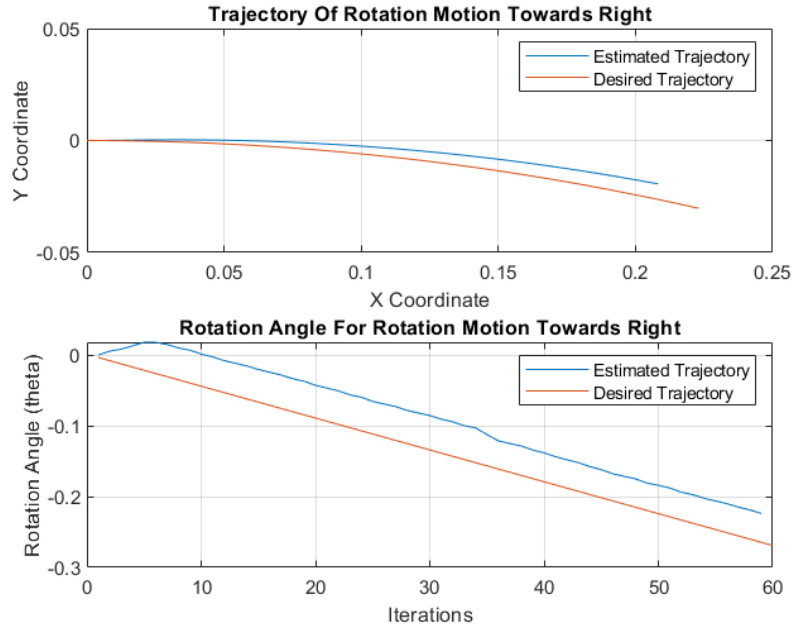


Figure 6: Position And Rotation Plot For Right Circular Motion

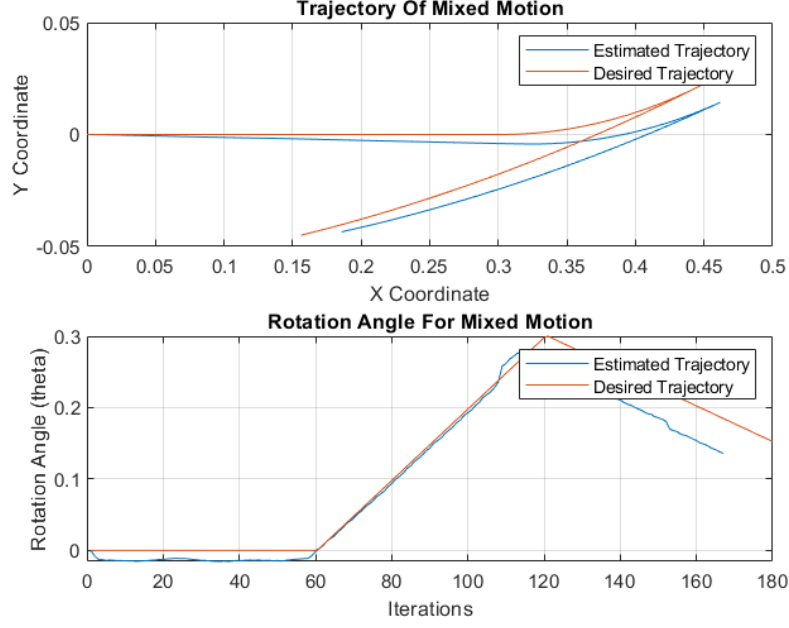


Figure 7: Position And Rotation Plot For Combined Motion

## 4 Conclusion

By making such odometric estimations, the way that the mobile robots make such estimations and their practical use was observed. By comparing the estimations with the actual trajectories, the effect of different kinds of motion and/or different sampling times etc. on the error of the estimation is also observed.

## 5 Discussion

### 5.1 Linear Motion

For linear motion, there was almost no error between the estimated and desired trajectories. Only a small error of about  $3 \times 10^{-4} m$  was observed. There was a similar result in terms of rotations as well. The error that was observed was about  $2 \times 10^{-3} m$ . Overall, it can be said that no noticeable changes occur when the speed is changed for linear motion.

### 5.2 Circular Motion

For rotational motion, error was significantly higher compared to linear motion. For trajectories, the estimated trajectory had much smaller diversions on the y coordinate compared to the desired trajectory. Which means that the robot recorded its rotation a bit smaller compared to its real rotational value. In terms of magnitude, the error is about  $3 \times 10^{-3} m$ . For rotation angle plots, in left rotation plot, the rotation amount was initially underestimated but after some time it was overestimated. For the rightwards rotation, it was underestimated all the way through. As for the amount of error, it was around  $0.05/0.1 rad/s$  in terms of magnitude.

Although I didn't get a chance to test the trajectories for different samples during the lab, since we get the estimated trajectory by integrating over time the position values, we can say that we will get a more accurate result when we decrease the sampling time. If we increase it, the result we get will be farther away from the desired trajectory.

### 5.3 Combined Motion

The plots for the combined motion can be found in the results section. The same pattern as in linear and circular motion can be seen here too. For linear motion both trajectories went kind of the same. For rotation the y axis of estimated trajectory changed less compared to desired one, leading to the robot to take less curvy turns.

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

Listing 1: odometry.m, Used To Estimate The Trajectory Of The Turtlebot

```
1  clc; close; clear all;
2
3  clear node; % clear previous node handle if it exists
4
5  % Environment Setup
6  setenv('ROS_DOMAIN_ID','36'); % Set to your robot's ROS_DOMAIN_ID
7  setenv('ROS_LOCALHOST_ONLY','0'); % 0 implies multi-host communication
8  setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware
9
10 node = ros2node('odometry'); % Create node
11
12 % Publishers & Subscribers
13 velPub = ros2publisher(node, "/cmd_vel", "geometry_msgs/Twist", ...
14 "Reliability", "reliable", "Durability", "volatile", "Depth", 10);
15
16 velMsg = ros2message(velPub);
17
18 encSub = ros2subscriber(node, "/joint_states", "sensor_msgs/JointState", ...
19 "Reliability", "besteffort", "Durability", "volatile", "Depth", 10);
20
21 % Initial Encoder Reading
22 encMsg = receive(encSub, 10);
23 jointNames = string(encMsg.name);
24 idxL = find(jointNames=="wheel_left_joint" | jointNames=="left_wheel", 1);
25 idxR = find(jointNames=="wheel_right_joint" | jointNames=="right_wheel", 1);
26
27 % Parameters Initialization
28 exeTime = 3;
29 period = 0.01;
30 R = 0.033;
31 L = 0.287;
32
33 % Vectors Initialization
34 xPositionsForwardHigh = [0];
35 yPositionsForwardHigh = [0];
36 rotationAnglesForwardHigh = [0];
37 xPositionsForwardLow = [0];
38 yPositionsForwardLow = [0];
39 rotationAnglesForwardLow = [0];
40 xPositionsBackwardHigh = [0];
41 yPositionsBackwardHigh = [0];
42 rotationAnglesBackwardHigh = [0];
43 xPositionsBackwardLow = [0];
44 yPositionsBackwardLow = [0];
45 rotationAnglesBackwardLow = [0];
46 xPositionsRotationRight = [0];
47 yPositionsRotationRight = [0];
48 rotationAnglesRotationRight = [0];
49 xPositionsRotationLeft = [0];
50 yPositionsRotationLeft = [0];
51 rotationAnglesRotationLeft = [0];
52 xPositionsMixed = [0];
53 yPositionsMixed = [0];
54 rotationAnglesMixed = [0];
55
56 % Forward Motion High Speed
57
```

```

58 velMsg.linear.x = 0.2;
59 velMsg.angular.z = 0;
60 send(velPub, velMsg);
61
62 encMsg = receive(encSub,10);
63 oldLeftWheelPos = encMsg.position(idxL);
64 oldRightWheelPos = encMsg.position(idxR);
65 x = 0; y = 0; theta = 0;
66 iteration = 2;
67 t0 = tic;
68
69 while toc(t0) < exeTime
70
71     % Read Encoders
72     encMsg = receive(encSub,0.5);
73     leftWheelPosition = encMsg.position(idxL);
74     rightWheelPosition = encMsg.position(idxR);
75
76     % Odometric Estimation
77     D1 = R * (leftWheelPosition - oldLeftWheelPos);
78     Dr = R * (rightWheelPosition - oldRightWheelPos);
79     Dc = (Dr + D1) / 2;
80     dTheta = (Dr - D1) / L;
81
82     % Values Recording (x,y,theta)
83     x = x + Dc * cos(theta + dTheta/2);
84     y = y + Dc * sin(theta + dTheta/2);
85     theta = theta + dTheta;
86
87     xPositionsForwardHigh(iteration) = x;
88     yPositionsForwardHigh(iteration) = y;
89     rotationAnglesForwardHigh(iteration) = theta;
90
91     oldLeftWheelPos = leftWheelPosition;
92     oldRightWheelPos = rightWheelPosition;
93     iteration = iteration + 1;
94
95     pause(period);
96 end
97
98
99
100 % Forward Motion Low Speed
101
102
103 velMsg.linear.x = 0.1;
104 velMsg.angular.z = 0;
105 send(velPub, velMsg);
106
107 encMsg = receive(encSub,10);
108 oldLeftWheelPos = encMsg.position(idxL);
109 oldRightWheelPos = encMsg.position(idxR);
110 x = 0; y = 0; theta = 0;
111 iteration = 2;
112 t0 = tic;
113
114 while toc(t0) < exeTime
115
116     % Read Encoders
117     encMsg = receive(encSub,0.5);
118     leftWheelPosition = encMsg.position(idxL);
119     rightWheelPosition = encMsg.position(idxR);
120
121     % Odometric Estimation
122     D1 = R * (leftWheelPosition - oldLeftWheelPos);
123     Dr = R * (rightWheelPosition - oldRightWheelPos);
124     Dc = (Dr + D1) / 2;
125     dTheta = (Dr - D1) / L;
126
127     % Values Recording (x,y,theta)
128     x = x + Dc * cos(theta + dTheta/2);

```



```

129     y = y + Dc * sin(theta + dTheta/2);
130     theta = theta + dTheta;
131
132     xPositionsForwardLow(iteration) = x;
133     yPositionsForwardLow(iteration) = y;
134     rotationAnglesForwardLow(iteration) = theta;
135
136     oldLeftWheelPos = leftWheelPosition;
137     oldRightWheelPos = rightWheelPosition;
138     iteration = iteration + 1;
139
140     pause(period);
141 end
142
143
144
145 % Backward Motion High Speed
146
147
148 velMsg.linear.x = -0.2;
149 velMsg.angular.z = 0;
150 send(velPub, velMsg);
151
152 encMsg = receive(encSub,10);
153 oldLeftWheelPos = encMsg.position(idXL);
154 oldRightWheelPos = encMsg.position(idXR);
155 x = 0; y = 0; theta = 0;
156 iteration = 2;
157 t0 = tic;
158
159 while toc(t0) < exeTime
160
161     % Read Encoders
162     encMsg = receive(encSub,0.5);
163     leftWheelPosition = encMsg.position(idXL);
164     rightWheelPosition = encMsg.position(idXR);
165
166     % Odometric Estimation
167     D1 = R * (leftWheelPosition - oldLeftWheelPos);
168     Dr = R * (rightWheelPosition - oldRightWheelPos);
169     Dc = (Dr + D1) / 2;
170     dTheta = (Dr - D1)/L;
171
172     % Values Recording (x,y,theta)
173     x = x + Dc * cos(theta + dTheta/2);
174     y = y + Dc * sin(theta + dTheta/2);
175     theta = theta + dTheta;
176
177     xPositionsBackwardHigh(iteration) = x;
178     yPositionsBackwardHigh(iteration) = y;
179     rotationAnglesBackwardHigh(iteration) = theta;
180
181     oldLeftWheelPos = leftWheelPosition;
182     oldRightWheelPos = rightWheelPosition;
183     iteration = iteration + 1;
184
185     pause(period);
186 end
187
188
189
190 % Backward Motion Low Speed
191
192
193 velMsg.linear.x = -0.1;
194 velMsg.angular.z = 0;
195 send(velPub, velMsg);
196
197 encMsg = receive(encSub,10);
198 oldLeftWheelPos = encMsg.position(idXL);
199 oldRightWheelPos = encMsg.position(idXR);

```

```

200 x = 0; y = 0; theta = 0;
201 iteration = 2;
202 t0 = tic;
203
204 while toc(t0) < exeTime
205
206     % Read Encoders
207     encMsg = receive(encSub,0.5);
208     leftWheelPosition = encMsg.position(idXL);
209     rightWheelPosition = encMsg.position(idXR);
210
211     % Odometric Estimation
212     Dl = R * (leftWheelPosition - oldLeftWheelPos);
213     Dr = R * (rightWheelPosition - oldRightWheelPos);
214     Dc = (Dr + Dl) / 2;
215     dTheta = (Dr - Dl)/L;
216
217     % Values Recording (x,y,theta)
218     x = x + Dc * cos(theta + dTheta/2);
219     y = y + Dc * sin(theta + dTheta/2);
220     theta = theta + dTheta;
221
222     xPositionsBackwardLow(iteration) = x;
223     yPositionsBackwardLow(iteration) = y;
224     rotationAnglesBackwardLow(iteration) = theta;
225
226     oldLeftWheelPos = leftWheelPosition;
227     oldRightWheelPos = rightWheelPosition;
228     iteration = iteration + 1;
229
230     pause(period);
231 end
232
233
234
235 % Left Rotation
236
237
238 velMsg.linear.x = 0.075;
239 velMsg.angular.z = 0.09;
240 send(velPub, velMsg);
241
242 encMsg = receive(encSub,10);
243 oldLeftWheelPos = encMsg.position(idXL);
244 oldRightWheelPos = encMsg.position(idXR);
245 x = 0; y = 0; theta = 0;
246 iteration = 2;
247 t0 = tic;
248
249 while toc(t0) < exeTime
250
251     % Read Encoders
252     encMsg = receive(encSub,0.5);
253     leftWheelPosition = encMsg.position(idXL);
254     rightWheelPosition = encMsg.position(idXR);
255
256     % Odometric Estimation
257     Dl = R * (leftWheelPosition - oldLeftWheelPos);
258     Dr = R * (rightWheelPosition - oldRightWheelPos);
259     Dc = (Dr + Dl)/2;
260     dTheta = (Dr - Dl)/L;
261
262     % Values Recording (x,y,theta)
263     x = x + Dc * cos(theta + dTheta/2);
264     y = y + Dc * sin(theta + dTheta/2);
265     theta = theta + dTheta;
266
267     xPositionsRotationLeft(iteration) = x;
268     yPositionsRotationLeft(iteration) = y;
269     rotationAnglesRotationLeft(iteration) = theta;
270

```

```

271     oldLeftWheelPos = leftWheelPosition;
272     oldRightWheelPos = rightWheelPosition;
273     iteration = iteration + 1;
274
275     pause(period);
276 end
277
278
279
280 % Right Rotation
281
282
283 velMsg.linear.x = 0.075;
284 velMsg.angular.z = -0.09;
285 send(velPub, velMsg);
286
287 encMsg = receive(encSub,10);
288 oldLeftWheelPos = encMsg.position(idXL);
289 oldRightWheelPos = encMsg.position(idXR);
290 x = 0; y = 0; theta = 0;
291 iteration = 2;
292 t0 = tic;
293
294 while toc(t0) < exeTime
295
296     % Read Encoders
297     encMsg = receive(encSub,0.5);
298     leftWheelPosition = encMsg.position(idXL);
299     rightWheelPosition = encMsg.position(idXR);
300
301     % Odometric Estimation
302     D1 = R * (leftWheelPosition - oldLeftWheelPos);
303     Dr = R * (rightWheelPosition - oldRightWheelPos);
304     Dc = (Dr + D1)/2;
305     dTheta = (Dr - D1)/L;
306
307     % Values Recording (x,y,theta)
308     x = x + Dc * cos(theta + dTheta/2);
309     y = y + Dc * sin(theta + dTheta/2);
310     theta = theta + dTheta;
311
312     xPositionsRotationRight(iteration) = x;
313     yPositionsRotationRight(iteration) = y;
314     rotationAnglesRotationRight(iteration) = theta;
315
316     oldLeftWheelPos = leftWheelPosition;
317     oldRightWheelPos = rightWheelPosition;
318     iteration = iteration + 1;
319
320     pause(period);
321 end
322
323
324
325 % Mixed Motion (three phases)
326
327
328 encMsg = receive(encSub,10);
329 oldLeftWheelPos = encMsg.position(idXL);
330 oldRightWheelPos = encMsg.position(idXR);
331 x = 0; y = 0; theta = 0;
332 iteration = 2;
333
334 % Phase 1
335 velMsg.linear.x = 0.1;
336 velMsg.angular.z = 0;
337 send(velPub, velMsg);
338 t0 = tic;
339 while toc(t0) < exeTime
340
341     % Read Encoders

```

```

342     encMsg = receive(encSub,0.5);
343     leftWheelPosition = encMsg.position(idxL);
344     rightWheelPosition = encMsg.position(idxR);
345
346     % Odometric Estimation
347     Dl = R * (leftWheelPosition - oldLeftWheelPos);
348     Dr = R * (rightWheelPosition - oldRightWheelPos);
349     Dc = (Dr + Dl)/2;
350     dTheta = (Dr - Dl)/L;
351
352     % Values Recording (x,y,theta)
353     x = x + Dc * cos(theta + dTheta/2);
354     y = y + Dc * sin(theta + dTheta/2);
355     theta = theta + dTheta;
356
357     xPositionsMixed(iteration) = x;
358     yPositionsMixed(iteration) = y;
359     rotationAnglesMixed(iteration) = theta;
360
361     oldLeftWheelPos = leftWheelPosition;
362     oldRightWheelPos = rightWheelPosition;
363     iteration = iteration + 1;
364
365     pause(period);
366 end
367
368 % Phase 2
369 velMsg.linear.x = 0.05;
370 velMsg.angular.z = 0.1;
371 send(velPub, velMsg);
372 t0 = tic;
373 while toc(t0) < exeTime
374
375     % Read Encoders
376     encMsg = receive(encSub,0.5);
377     leftWheelPosition = encMsg.position(idxL);
378     rightWheelPosition = encMsg.position(idxR);
379
380     % Odometric Estimation
381     Dl = R * (leftWheelPosition - oldLeftWheelPos);
382     Dr = R * (rightWheelPosition - oldRightWheelPos);
383     Dc = (Dr + Dl)/2;
384     dTheta = (Dr - Dl)/L;
385
386     % Values Recording (x,y,theta)
387     x = x + Dc * cos(theta + dTheta/2);
388     y = y + Dc * sin(theta + dTheta/2);
389     theta = theta + dTheta;
390
391     xPositionsMixed(iteration) = x;
392     yPositionsMixed(iteration) = y;
393     rotationAnglesMixed(iteration) = theta;
394
395     oldLeftWheelPos = leftWheelPosition;
396     oldRightWheelPos = rightWheelPosition;
397     iteration = iteration + 1;
398
399     pause(period);
400 end
401
402 % Phase 3
403 velMsg.linear.x = -0.1;
404 velMsg.angular.z = -0.05;
405 send(velPub, velMsg);
406 t0 = tic;
407 while toc(t0) < exeTime
408
409     % Read Encoders
410     encMsg = receive(encSub,0.5);
411     leftWheelPosition = encMsg.position(idxL);
412     rightWheelPosition = encMsg.position(idxR);

```

```

413
414 % Odometric Estimation
415 Dl = R * (leftWheelPosition - oldLeftWheelPos);
416 Dr = R * (rightWheelPosition - oldRightWheelPos);
417 Dc = (Dr + Dl)/2;
418 dTheta = (Dr - Dl)/L;
419
420 % Values Recording (x,y,theta)
421 x = x + Dc * cos(theta + dTheta/2);
422 y = y + Dc * sin(theta + dTheta/2);
423 theta = theta + dTheta;
424
425 xPositionsMixed(iteration) = x;
426 yPositionsMixed(iteration) = y;
427 rotationAnglesMixed(iteration) = theta;
428
429 oldLeftWheelPos = leftWheelPosition;
430 oldRightWheelPos = rightWheelPosition;
431 iteration = iteration + 1;
432
433 pause(period);
434 end
435
436 % Final stop
437 velMsg.linear.x = 0;
438 velMsg.angular.z = 0;
439 send(velPub, velMsg);

```

Listing 2: desiredTrajectory.m, Used To Calculate The Desired Trajectory Of The Turtlebot

```

1  %Parameter Initialization
2  exeTime = 3;
3  period = 0.01;
4  iteration = 2;
5
6  R = 0.033;
7  L = 0.287;
8
9  t0 = 0;
10 x = 0;
11 y = 0;
12 theta = 0;
13
14 % Vectors Initialization
15 xFH = [0];
16 yFH = [0];
17 tFH = [0];
18 xFL = [0];
19 yFL = [0];
20 tFL = [0];
21 xBH = [0];
22 yBH = [0];
23 tBH = [0];
24 xBL = [0];
25 yBL = [0];
26 tBL = [0];
27 xRR = [0];
28 yRR = [0];
29 tRR = [0];
30 xRL = [0];
31 yRL = [0];
32 tRL = [0];
33 xM = [0];
34 yM = [0];
35 tM = [0];
36
37 %Forward Motion High Speed
38 v = 0.2;
39 w = 0;
40 while t0 < exeTime
41     %x, y, theta Calculations
42     x = x + v * period * cos(theta);
43     y = y + v * period * sin(theta);
44     theta = theta + w * period;
45     % Values Recording (x,y,theta)
46     xFH(iteration) = x;
47     yFH(iteration) = y;
48     if rem(iteration,5) == 0
49         tFH(floor(iteration / 5)) = theta;
50     end
51
52     %Update Parameters
53     iteration = iteration + 1;
54
55     t0 = t0 + period;
56 end
57
58 t0 = 0;
59 x = 0;
60 y = 0;
61 theta = 0;
62 iteration = 2;
63
64 v = 0.1;
65 w = 0;
66 %Forward Motion Low Speed
67 while t0 < exeTime
68     %x, y, theta Calculations
69     x = x + v * period * cos(theta);
70     y = y + v * period * sin(theta);

```

```

71     theta = theta + w * period;
72     % Values Recording (x,y,theta)
73     xFL(iteration) = x;
74     yFL(iteration) = y;
75     if rem(iteration,5) == 0
76         tFL(floor(iteration / 5)) = theta;
77     end
78
79     %Update Parameters
80     iteration = iteration + 1;
81
82     t0 = t0 + period;
83 end
84
85 t0 = 0;
86 x = 0;
87 y = 0;
88 theta = 0;
89 iteration = 2;
90
91 v = -0.2;
92 w = 0;
93 %Backward Motion High Speed
94 while t0 < exeTime
95     %x, y, theta Calculations
96     x = x + v * period * cos(theta);
97     y = y + v * period * sin(theta);
98     theta = theta + w * period;
99     % Values Recording (x,y,theta)
100    xBH(iteration) = x;
101    yBH(iteration) = y;
102    if rem(iteration,5) == 0
103        tBH(floor(iteration / 5)) = theta;
104    end
105
106    %Update Parameters
107    iteration = iteration + 1;
108
109    t0 = t0 + period;
110 end
111
112 t0 = 0;
113 x = 0;
114 y = 0;
115 theta = 0;
116 iteration = 2;
117
118 v = -0.1;
119 w = 0;
120 %Backward Motion Low Speed
121 while t0 < exeTime
122     %x, y, theta Calculations
123     x = x + v * period * cos(theta);
124     y = y + v * period * sin(theta);
125     theta = theta + w * period;
126     % Values Recording (x,y,theta)
127     xBL(iteration) = x;
128     yBL(iteration) = y;
129     if rem(iteration,5) == 0
130         tBL(floor(iteration / 5)) = theta;
131     end
132
133     %Update Parameters
134     iteration = iteration + 1;
135
136     t0 = t0 + period;
137 end
138
139 t0 = 0;
140 x = 0;
141 y = 0;

```

```

142 theta = 0;
143 iteration = 2;
144
145 v = 0.075;
146 w = 0.09;
147 %Left Rotation
148 while t0 < exeTime
149     %x, y, theta Calculations
150     x = x + v * period * cos(theta);
151     y = y + v * period * sin(theta);
152     theta = theta + w * period;
153     % Values Recording (x,y,theta)
154     xRL(iteration) = x;
155     yRL(iteration) = y;
156     if rem(iteration,5) == 0
157         tRL(floor(iteration / 5)) = theta;
158     end
159
160     %Update Parameters
161     iteration = iteration + 1;
162
163     t0 = t0 + period;
164 end
165 t0 = 0;
166 x = 0;
167 y = 0;
168 theta = 0;
169 iteration = 2;
170
171 v = 0.075;
172 w = -0.09;
173 %Right Rotation
174 while t0 < exeTime
175     %x, y, theta Calculations
176     x = x + v * period * cos(theta);
177     y = y + v * period * sin(theta);
178     theta = theta + w * period;
179     % Values Recording (x,y,theta)
180     xRR(iteration) = x;
181     yRR(iteration) = y;
182     if rem(iteration,5) == 0
183         tRR(floor(iteration / 5)) = theta;
184     end
185
186     %Update Parameters
187     iteration = iteration + 1;
188
189     t0 = t0 + period;
190 end
191
192 t0 = 0;
193 x = 0;
194 y = 0;
195 theta = 0;
196 iteration = 2;
197
198 v = 0.1;
199 w = 0;
200 %Mixed Motion
201 while t0 < exeTime
202     %x, y, theta Calculations
203     x = x + v * period * cos(theta);
204     y = y + v * period * sin(theta);
205     theta = theta + w * period;
206     % Values Recording (x,y,theta)
207     xM(iteration) = x;
208     yM(iteration) = y;
209     if rem(iteration,5) == 0
210         tM(floor(iteration / 5)) = theta;
211     end
212

```



```

213     %Update Parameters
214     iteration = iteration + 1;
215
216     t0 = t0 + period;
217 end
218
219 t0 = 0;
220 v = 0.05;
221 w = 0.1;
222 %Mixed Motion
223 while t0 < exeTime
224     %x, y, theta Calculations
225     x = x + v * period * cos(theta);
226     y = y + v * period * sin(theta);
227     theta = theta + w * period;
228     % Values Recording (x,y,theta)
229     xM(iteration) = x;
230     yM(iteration) = y;
231     if rem(iteration,5) == 0
232         tM(floor(iteration / 5)) = theta;
233     end
234
235     %Update Parameters
236     iteration = iteration + 1;
237
238     t0 = t0 + period;
239 end
240
241 t0 = 0;
242 v = -0.1;
243 w = -0.05;
244 %Mixed Motion
245 while t0 < exeTime
246     %x, y, theta Calculations
247     x = x + v * period * cos(theta);
248     y = y + v * period * sin(theta);
249     theta = theta + w * period;
250     % Values Recording (x,y,theta)
251     xM(iteration) = x;
252     yM(iteration) = y;
253     if rem(iteration,5) == 0
254         tM(floor(iteration / 5)) = theta;
255     end
256
257     %Update Parameters
258     iteration = iteration + 1;
259
260     t0 = t0 + period;
261 end

```

Listing 3: dataPlotting.m, Used To Plot Both Data Received From Previous Scripts

```

1  figure;
2
3  subplot(1,2,1);
4  plot(xPositionsForwardHigh, yPositionsForwardHigh);
5  hold on;
6  plot(xFH, yFH);
7  ylim([-0.05 0.05]);
8  title("Trajectory Of Forward Motion With High Speed");
9  xlabel("X Coordinate");
10 ylabel("Y Coordinate");
11 grid on;
12 legend("Estimated Trajectory", "Desired Trajectory");
13 hold off;
14
15 subplot(1,2,2);
16 plot(rotationAnglesForwardHigh);
17 hold on;
18 plot(tFH);
19 plot(tFH);
20 ylim([-0.05 0.05]);
21 title("Rotation Angle For Forward Motion With High Speed");
22 xlabel("Iterations");
23 ylabel("Rotation Angle (theta)");
24 grid on;
25 legend("Estimated Trajectory", "Desired Trajectory");
26 hold off;
27
28
29
30 figure;
31 subplot(1,2,1);
32 plot(xPositionsForwardLow, yPositionsForwardLow);
33 hold on;
34 plot(xFL, yFL);
35 ylim([-0.05 0.05]);
36 title("Trajectory Of Forward Motion With Low Speed");
37 xlabel("X Coordinate");
38 ylabel("Y Coordinate");
39 grid on;
40 legend("Estimated Trajectory", "Desired Trajectory");
41 hold off;
42
43 subplot(1,2,2);
44 plot(rotationAnglesForwardLow);
45 hold on;
46 plot(tFL);
47 ylim([-0.05 0.05]);
48 title("Rotation Angle For Forward Motion With Low Speed");
49 xlabel("Iterations");
50 ylabel("Rotation Angle (theta)");
51 grid on;
52 legend("Estimated Trajectory", "Desired Trajectory");
53 hold off;
54
55
56
57 figure;
58 subplot(1,2,1);
59 plot(xPositionsBackwardHigh, yPositionsBackwardHigh);
60 hold on;
61 plot(xBH, yBH);
62 set(gca, 'XDir', 'reverse');
63 ylim([-0.05 0.05]);
64 title("Trajectory Of Backward Motion With High Speed");
65 xlabel("X Coordinate");
66 ylabel("Y Coordinate");
67 grid on;
68 legend("Estimated Trajectory", "Desired Trajectory");
69 hold off;
70

```

```

71 subplot(1,2,2);
72 plot(rotationAnglesBackwardHigh);
73 hold on;
74 plot(tBH);
75 ylim([-0.05 0.05]);
76 title("Rotation Angle For Backward Motion With High Speed");
77 xlabel("Iterations");
78 ylabel("Rotation Angle (theta)");
79 grid on;
80 legend("Estimated Trajectory", "Desired Trajectory");
81 hold off;
82
83
84 figure;
85 subplot(1,2,1);
86 plot(xPositionsBackwardLow, yPositionsBackwardLow);
87 hold on;
88 plot(xBL, yBL);
89 set(gca, 'XDir', 'reverse');
90 ylim([-0.05 0.05]);
91 title("Trajectory Of Backward Motion With Low Speed");
92 xlabel("X Coordinate");
93 ylabel("Y Coordinate");
94 grid on;
95 legend("Estimated Trajectory", "Desired Trajectory");
96 hold off;
97
98 subplot(1,2,2);
99 plot(rotationAnglesBackwardLow);
100 hold on;
101 plot(tBL);
102 ylim([-0.05 0.05]);
103 title("Rotation Angle For Backward Motion With Low Speed");
104 xlabel("Iterations");
105 ylabel("Rotation Angle (theta)");
106 grid on;
107 legend("Estimated Trajectory", "Desired Trajectory");
108 hold off;
109
110
111 figure;
112 subplot(1,2,1);
113 plot(xPositionsRotationLeft, yPositionsRotationLeft);
114 hold on;
115 plot(xRL, yRL);
116 ylim([-0.05 0.05]);
117 title("Trajectory Of Rotation Motion Towards Left");
118 xlabel("X Coordinate");
119 ylabel("Y Coordinate");
120 grid on;
121 legend("Estimated Trajectory", "Desired Trajectory");
122 hold off;
123
124 subplot(1,2,2);
125 plot(rotationAnglesRotationLeft);
126 hold on;
127 plot(tRL);
128 title("Rotation Angle For Rotation Motion Towards Left");
129 xlabel("Iterations");
130 ylabel("Rotation Angle (theta)");
131 grid on;
132 legend("Estimated Trajectory", "Desired Trajectory");
133 hold off;
134
135
136 figure;
137 subplot(1,2,1);
138 plot(xPositionsRotationRight, yPositionsRotationRight);
139 hold on;
140 plot(xRR, yRR);
141 ylim([-0.05 0.05]);

```

```

142 title("Trajectory Of Rotation Motion Towards Right");
143 xlabel("X Coordinate");
144 ylabel("Y Coordinate");
145 grid on;
146 legend("Estimated Trajectory", "Desired Trajectory");
147 hold off;
148
149 subplot(1,2,2);
150 plot(rotationAnglesRotationRight);
151 hold on;
152 plot(tRR);
153 title("Rotation Angle For Rotation Motion Towards Right");
154 xlabel("Iterations");
155 ylabel("Rotation Angle (theta)");
156 grid on;
157 legend("Estimated Trajectory", "Desired Trajectory");
158 hold off;
159
160
161 figure;
162 subplot(1,2,1);
163 plot(xPositionsMixed, yPositionsMixed);
164 hold on;
165 plot(xM, yM);
166 ylim([-0.05 0.05]);
167 title("Trajectory Of Mixed Motion");
168 xlabel("X Coordinate");
169 ylabel("Y Coordinate");
170 grid on;
171 legend("Estimated Trajectory", "Desired Trajectory");
172 hold off;
173
174 subplot(1,2,2);
175 plot(rotationAnglesMixed);
176 hold on;
177 plot(tM);
178 title("Rotation Angle For Mixed Motion");
179 xlabel("Iterations");
180 ylabel("Rotation Angle (theta)");
181 grid on;
182 legend("Estimated Trajectory", "Desired Trajectory");
183 hold off;

```