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 seperation) values are defined. Initial x, y and θ 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 (ω) velocities used with them is as follows:

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

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:

$$\Delta E_r = w_r(k+1) - w_r(k),\ \Delta E_l = w_l(k+1) - w_l(k)$$
$$D_l = R\Delta E_r,\ D_l = R\Delta E_l$$
$$D_c = \frac{D_r + D_l}{2}$$
$$\Delta\theta = \frac{D_r - D_l}{2}$$
$$x(k+1) = x(k) + D_c cos(\theta(k) + \frac{\Delta\theta}{2})$$
$$y(k+1) = y(k) + D_c sin(\theta(k) + \frac{\Delta\theta}{2})$$
$$\theta(k+1) = \theta(k) + \Delta\theta$$

   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 replaed 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
   ω : The angular velocity of robot
   v : The linear velocity of robot
   Δt : 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 recieved from the encoders for better plotting.
4. The new pose values are then recorded to the corresponding vectors.

2.3 Data Plotting

The recieved data is then plotted into seperate 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

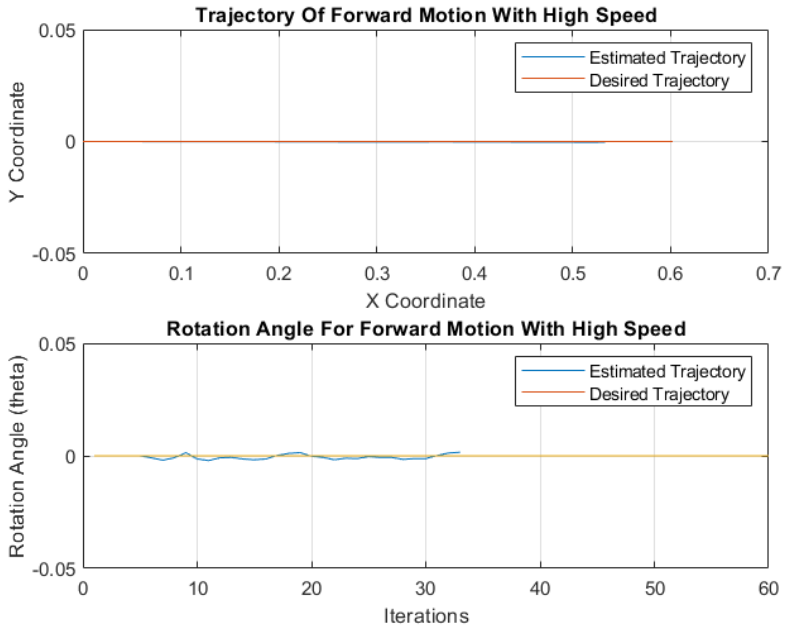

Figure 1: Position And Rotation Plot For Forward Motion With High Speed

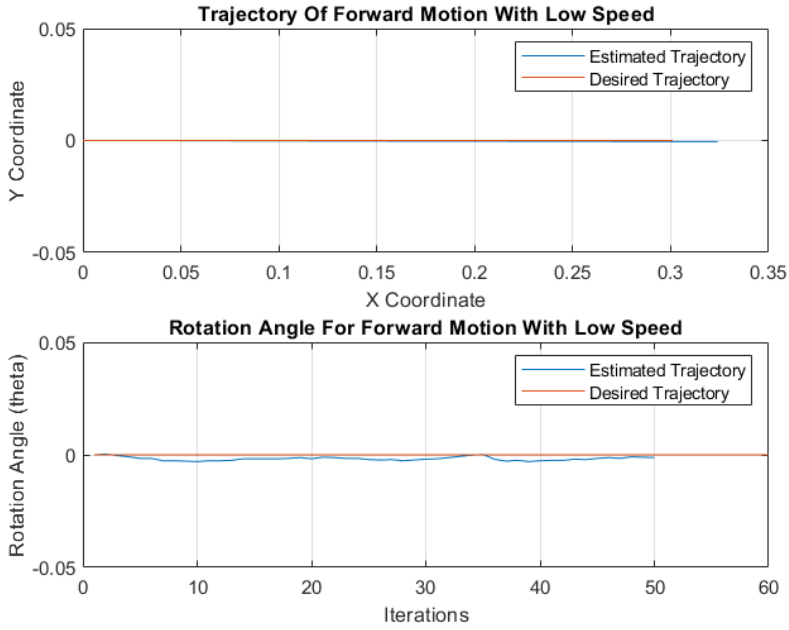

Figure 2: Position And Rotation Plot For Forward Motion With Low Speed

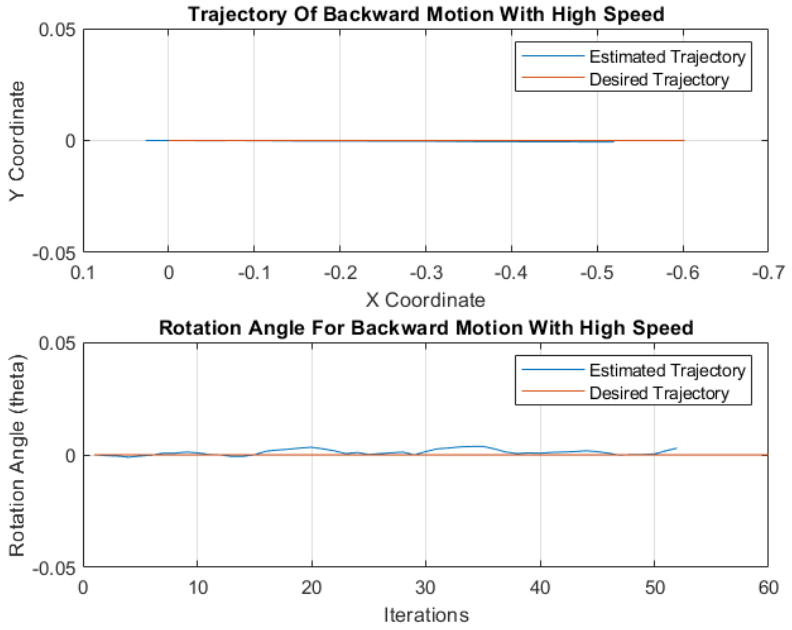

Figure 3: Position And Rotation Plot For Backward Motion With High Speed

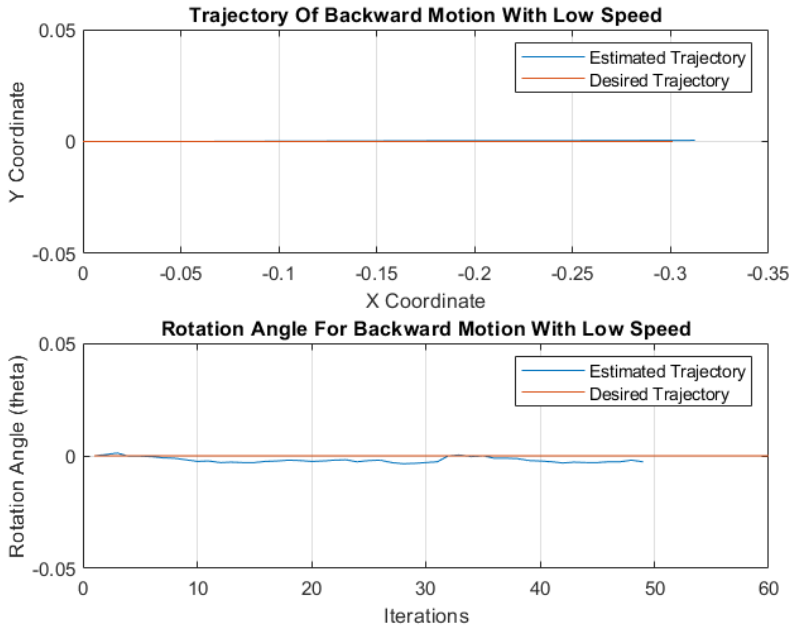

Figure 4: Position And Rotation Plot For Backward Motion With Low Speed

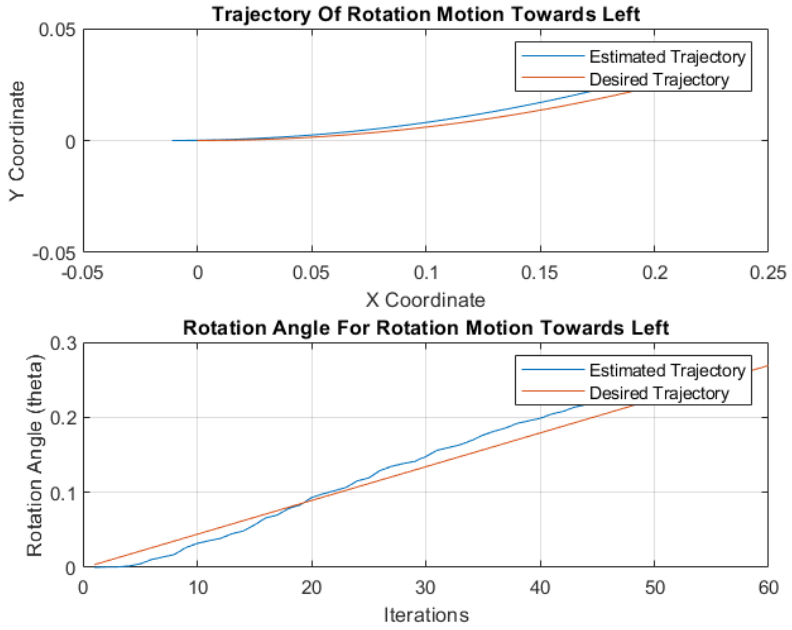

Figure 5: Position And Rotation Plot For Left Circular Motion

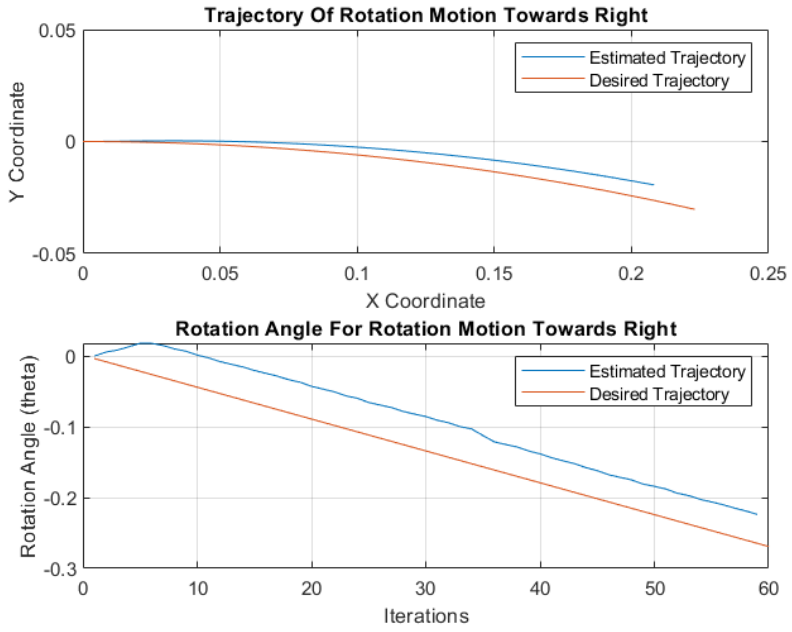

Figure 6: Position And Rotation Plot For Right Circular Motion

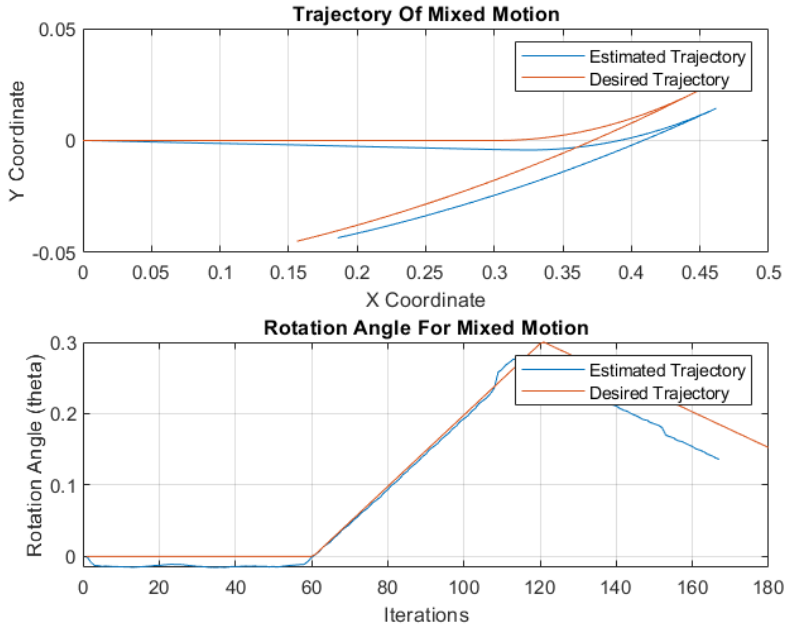

Figure 7: Position And Rotation Plot For Combined Motion

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 noticable 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 samlples during the lab, since we get the estimated trajectory by integrateing 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 rotationi 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

```
clc; close; clear all;

clear node; % clear previous node handle if it exists

% Environment Setup
setenv('ROS_DOMAIN_ID','36'); % Set to your robot's ROS_DOMAIN_ID
setenv('ROS_LOCALHOST_ONLY', '0'); % 0 implies multi-host communication
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware

node = ros2node('odometry');  % Create node

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

velMsg = ros2message(velPub);

encSub = ros2subscriber(node,"/joint_states","sensor_msgs/JointState", ...
"Reliability","besteffort","Durability","volatile","Depth",10);

% Initial Encoder Reading
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);

% Parameters Initialization
exeTime = 3;
period = 0.01;
R = 0.033;
L = 0.287;

% Vectors Initialization
xPositionsForwardHigh = [0];
yPositionsForwardHigh = [0];
rotationAnglesForwardHigh = [0];
xPositionsForwardLow = [0];
yPositionsForwardLow = [0];
rotationAnglesForwardLow = [0];
xPositionsBackwardHigh = [0];
yPositionsBackwardHigh = [0];
rotationAnglesBackwardHigh = [0];
xPositionsBackwardLow = [0];
yPositionsBackwardLow = [0];
rotationAnglesBackwardLow = [0];
xPositionsRotationRight = [0];
yPositionsRotationRight = [0];
rotationAnglesRotationRight = [0];
xPositionsRotationLeft = [0];
yPositionsRotationLeft = [0];
rotationAnglesRotationLeft = [0];
xPositionsMixed = [0];
yPositionsMixed = [0];
rotationAnglesMixed = [0];

% Forward Motion High Speed

```

```
velMsg.linear.x = 0.2;
velMsg.angular.z = 0;
send(velPub, velMsg);

encMsg = receive(encSub,10);
oldLeftWheelPos = encMsg.position(idxL);
oldRightWheelPos = encMsg.position(idxR);
x = 0; y = 0; theta = 0;
iteration = 2;
t0 = tic;

while toc(t0) < exeTime

    % Read Encoders
    encMsg = receive(encSub,0.5);
    leftWheelPosition = encMsg.position(idxL);
    rightWheelPosition = encMsg.position(idxR);

    % Odometric Estimation
    Dl = R * (leftWheelPosition - oldLeftWheelPos);
    Dr = R * (rightWheelPosition - oldRightWheelPos);
    Dc = (Dr + Dl) / 2;
    dTheta = (Dr - Dl) / L;

    % Values Recording (x,y,theta)
    x = x + Dc * cos(theta + dTheta/2);
    y = y + Dc * sin(theta + dTheta/2);
    theta = theta + dTheta;

    xPositionsForwardHigh(iteration) = x;
    yPositionsForwardHigh(iteration) = y;
    rotationAnglesForwardHigh(iteration) = theta;

    oldLeftWheelPos = leftWheelPosition;
    oldRightWheelPos = rightWheelPosition;
    iteration = iteration + 1;

    pause(period);
end



% Forward Motion Low Speed


velMsg.linear.x = 0.1;
velMsg.angular.z = 0;
send(velPub, velMsg);

encMsg = receive(encSub,10);
oldLeftWheelPos = encMsg.position(idxL);
oldRightWheelPos = encMsg.position(idxR);
x = 0; y = 0; theta = 0;
iteration = 2;
t0 = tic;

while toc(t0) < exeTime

    % Read Encoders
    encMsg = receive(encSub,0.5);
    leftWheelPosition = encMsg.position(idxL);
    rightWheelPosition = encMsg.position(idxR);

    % Odometric Estimation
    Dl = R * (leftWheelPosition - oldLeftWheelPos);
    Dr = R * (rightWheelPosition - oldRightWheelPos);
    Dc = (Dr + Dl) / 2;
    dTheta = (Dr - Dl) / L;

    % Values Recording (x,y,theta)
    x = x + Dc * cos(theta + dTheta/2);
```

```
    y = y + Dc * sin(theta + dTheta/2);
    theta = theta + dTheta;

    xPositionsForwardLow(iteration) = x;
    yPositionsForwardLow(iteration) = y;
    rotationAnglesForwardLow(iteration) = theta;

    oldLeftWheelPos = leftWheelPosition;
    oldRightWheelPos = rightWheelPosition;
    iteration = iteration + 1;

    pause(period);
end



% Backward Motion High Speed


velMsg.linear.x = -0.2;
velMsg.angular.z = 0;
send(velPub, velMsg);

encMsg = receive(encSub,10);
oldLeftWheelPos = encMsg.position(idxL);
oldRightWheelPos = encMsg.position(idxR);
x = 0; y = 0; theta = 0;
iteration = 2;
t0 = tic;

while toc(t0) < exeTime

    % Read Encoders
    encMsg = receive(encSub,0.5);
    leftWheelPosition = encMsg.position(idxL);
    rightWheelPosition = encMsg.position(idxR);

    % Odometric Estimation
    Dl = R * (leftWheelPosition - oldLeftWheelPos);
    Dr = R * (rightWheelPosition - oldRightWheelPos);
    Dc = (Dr + Dl) / 2;
    dTheta = (Dr - Dl)/L;

    % Values Recording (x,y,theta)
    x = x + Dc * cos(theta + dTheta/2);
    y = y + Dc * sin(theta + dTheta/2);
    theta = theta + dTheta;

    xPositionsBackwardHigh(iteration) = x;
    yPositionsBackwardHigh(iteration) = y;
    rotationAnglesBackwardHigh(iteration) = theta;

    oldLeftWheelPos = leftWheelPosition;
    oldRightWheelPos = rightWheelPosition;
    iteration = iteration + 1;

    pause(period);
end



% Backward Motion Low Speed


velMsg.linear.x = -0.1;
velMsg.angular.z = 0;
send(velPub, velMsg);

encMsg = receive(encSub,10);
oldLeftWheelPos = encMsg.position(idxL);
oldRightWheelPos = encMsg.position(idxR);
```

```
x = 0; y = 0; theta = 0;
iteration = 2;
t0 = tic;

while toc(t0) < exeTime

    % Read Encoders
    encMsg = receive(encSub,0.5);
    leftWheelPosition = encMsg.position(idxL);
    rightWheelPosition = encMsg.position(idxR);

    % Odometric Estimation
    Dl = R * (leftWheelPosition - oldLeftWheelPos);
    Dr = R * (rightWheelPosition - oldRightWheelPos);
    Dc = (Dr + Dl) / 2;
    dTheta = (Dr - Dl)/L;

    % Values Recording (x,y,theta)
    x = x + Dc * cos(theta + dTheta/2);
    y = y + Dc * sin(theta + dTheta/2);
    theta = theta + dTheta;

    xPositionsBackwardLow(iteration) = x;
    yPositionsBackwardLow(iteration) = y;
    rotationAnglesBackwardLow(iteration) = theta;

    oldLeftWheelPos = leftWheelPosition;
    oldRightWheelPos = rightWheelPosition;
    iteration = iteration + 1;

    pause(period);
end



% Left Rotation


velMsg.linear.x = 0.075;
velMsg.angular.z = 0.09;
send(velPub, velMsg);

encMsg = receive(encSub,10);
oldLeftWheelPos = encMsg.position(idxL);
oldRightWheelPos = encMsg.position(idxR);
x = 0; y = 0; theta = 0;
iteration = 2;
t0 = tic;

while toc(t0) < exeTime

    % Read Encoders
    encMsg = receive(encSub,0.5);
    leftWheelPosition = encMsg.position(idxL);
    rightWheelPosition = encMsg.position(idxR);

    % Odometric Estimation
    Dl = R * (leftWheelPosition - oldLeftWheelPos);
    Dr = R * (rightWheelPosition - oldRightWheelPos);
    Dc = (Dr + Dl)/2;
    dTheta = (Dr - Dl)/L;

    % Values Recording (x,y,theta)
    x = x + Dc * cos(theta + dTheta/2);
    y = y + Dc * sin(theta + dTheta/2);
    theta = theta + dTheta;

    xPositionsRotationLeft(iteration) = x;
    yPositionsRotationLeft(iteration) = y;
    rotationAnglesRotationLeft(iteration) = theta;

```

```
    oldLeftWheelPos = leftWheelPosition;
    oldRightWheelPos = rightWheelPosition;
    iteration = iteration + 1;

    pause(period);
end



% Right Rotation


velMsg.linear.x = 0.075;
velMsg.angular.z = -0.09;
send(velPub, velMsg);

encMsg = receive(encSub,10);
oldLeftWheelPos = encMsg.position(idxL);
oldRightWheelPos = encMsg.position(idxR);
x = 0; y = 0; theta = 0;
iteration = 2;
t0 = tic;

while toc(t0) < exeTime

    % Read Encoders
    encMsg = receive(encSub,0.5);
    leftWheelPosition = encMsg.position(idxL);
    rightWheelPosition = encMsg.position(idxR);

    % Odometric Estimation
    Dl = R * (leftWheelPosition - oldLeftWheelPos);
    Dr = R * (rightWheelPosition - oldRightWheelPos);
    Dc = (Dr + Dl)/2;
    dTheta = (Dr - Dl)/L;

    % Values Recording (x,y,theta)
    x = x + Dc * cos(theta + dTheta/2);
    y = y + Dc * sin(theta + dTheta/2);
    theta = theta + dTheta;

    xPositionsRotationRight(iteration) = x;
    yPositionsRotationRight(iteration) = y;
    rotationAnglesRotationRight(iteration) = theta;

    oldLeftWheelPos = leftWheelPosition;
    oldRightWheelPos = rightWheelPosition;
    iteration = iteration + 1;

    pause(period);
end



% Mixed Motion (three phases)


encMsg = receive(encSub,10);
oldLeftWheelPos = encMsg.position(idxL);
oldRightWheelPos = encMsg.position(idxR);
x = 0; y = 0; theta = 0;
iteration = 2;

% Phase 1
velMsg.linear.x = 0.1;
velMsg.angular.z = 0;
send(velPub, velMsg);
t0 = tic;
while toc(t0) < exeTime

    % 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

```
%Parameter Initialization
exeTime = 3;
period = 0.01;
iteration = 2;

R = 0.033;
L = 0.287;

t0 = 0;
x = 0;
y = 0;
theta = 0;

% Vectors Initialization
xFH = [0];
yFH = [0];
tFH = [0];
xFL = [0];
yFL = [0];
tFL = [0];
xBH = [0];
yBH = [0];
tBH = [0];
xBL = [0];
yBL = [0];
tBL = [0];
xRR = [0];
yRR = [0];
tRR = [0];
xRL = [0];
yRL = [0];
tRL = [0];
xM = [0];
yM = [0];
tM = [0];

%Forward Motion High Speed
v = 0.2;
w = 0;
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xFH(iteration) = x;
    yFH(iteration) = y;
    if rem(iteration,5) == 0
        tFH(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end

t0 = 0;
x = 0;
y = 0;
theta = 0;
iteration = 2;

v = 0.1;
w = 0;
%Forward Motion Low Speed
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
```

```
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xFL(iteration) = x;
    yFL(iteration) = y;
    if rem(iteration,5) == 0
        tFL(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end

t0 = 0;
x = 0;
y = 0;
theta = 0;
iteration = 2;

v = -0.2;
w = 0;
%Backward Motion High Speed
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xBH(iteration) = x;
    yBH(iteration) = y;
    if rem(iteration,5) == 0
        tBH(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end

t0 = 0;
x = 0;
y = 0;
theta = 0;
iteration = 2;

v = -0.1;
w = 0;
%Backward Motion Low Speed
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xBL(iteration) = x;
    yBL(iteration) = y;
    if rem(iteration,5) == 0
        tBL(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end

t0 = 0;
x = 0;
y = 0;
```

```
theta = 0;
iteration = 2;

v = 0.075;
w = 0.09;
%Left Rotation
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xRL(iteration) = x;
    yRL(iteration) = y;
    if rem(iteration,5) == 0
        tRL(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end
t0 = 0;
x = 0;
y = 0;
theta = 0;
iteration = 2;

v = 0.075;
w = -0.09;
%Right Rotation
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xRR(iteration) = x;
    yRR(iteration) = y;
    if rem(iteration,5) == 0
        tRR(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end

t0 = 0;
x = 0;
y = 0;
theta = 0;
iteration = 2;

v = 0.1;
w = 0;
%Mixed Motion
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xM(iteration) = x;
    yM(iteration) = y;
    if rem(iteration,5) == 0
        tM(floor(iteration / 5)) = theta;
    end

```

```
    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end

t0 = 0;
v = 0.05;
w = 0.1;
%Mixed Motion
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xM(iteration) = x;
    yM(iteration) = y;
    if rem(iteration,5) == 0
        tM(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end

t0 = 0;
v = -0.1;
w = -0.05;
%Mixed Motion
while t0 < exeTime
    %x, y, theta Calculations
    x = x + v * period * cos(theta);
    y = y + v * period * sin(theta);
    theta = theta + w * period;
    % Values Recording (x,y,theta)
    xM(iteration) = x;
    yM(iteration) = y;
    if rem(iteration,5) == 0
        tM(floor(iteration / 5)) = theta;
    end

    %Update Parameters
    iteration = iteration + 1;

    t0 = t0 + period;
end
```

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

```
figure;

subplot(1,2,1);
plot(xPositionsForwardHigh, yPositionsForwardHigh);
hold on;
plot(xFH, yFH);
ylim([-0.05 0.05]);
title("Trajectory Of Forward Motion With High Speed");
xlabel("X Coordinate");
ylabel("Y Coordinate");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;

subplot(1,2,2);
plot(rotationAnglesForwardHigh);
hold on;
plot(tFH);
plot(tFH);
ylim([-0.05 0.05]);
title("Rotation Angle For Forward Motion With High Speed");
xlabel("Iterations");
ylabel("Rotation Angle (theta)");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;



figure;
subplot(1,2,1);
plot(xPositionsForwardLow, yPositionsForwardLow);
hold on;
plot(xFL, yFL);
ylim([-0.05 0.05]);
title("Trajectory Of Forward Motion With Low Speed");
xlabel("X Coordinate");
ylabel("Y Coordinate");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;

subplot(1,2,2);
plot(rotationAnglesForwardLow);
hold on;
plot(tFL);
ylim([-0.05 0.05]);
title("Rotation Angle For Forward Motion With Low Speed");
xlabel("Iterations");
ylabel("Rotation Angle (theta)");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;



figure;
subplot(1,2,1);
plot(xPositionsBackwardHigh, yPositionsBackwardHigh);
hold on;
plot(xBH, yBH);
set(gca, 'XDir', 'reverse');
ylim([-0.05 0.05]);
title("Trajectory Of Backward Motion With High Speed");
xlabel("X Coordinate");
ylabel("Y Coordinate");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;

```

```
subplot(1,2,2);
plot(rotationAnglesBackwardHigh);
hold on;
plot(tBH);
ylim([-0.05 0.05]);
title("Rotation Angle For Backward Motion With High Speed");
xlabel("Iterations");
ylabel("Rotation Angle (theta)");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;


figure;
subplot(1,2,1);
plot(xPositionsBackwardLow, yPositionsBackwardLow);
hold on;
plot(xBL, yBL);
set(gca, 'XDir', 'reverse');
ylim([-0.05 0.05]);
title("Trajectory Of Backward Motion With Low Speed");
xlabel("X Coordinate");
ylabel("Y Coordinate");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;

subplot(1,2,2);
plot(rotationAnglesBackwardLow);
hold on;
plot(tBL);
ylim([-0.05 0.05]);
title("Rotation Angle For Backward Motion With Low Speed");
xlabel("Iterations");
ylabel("Rotation Angle (theta)");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;


figure;
subplot(1,2,1);
plot(xPositionsRotationLeft, yPositionsRotationLeft);
hold on;
plot(xRL, yRL);
ylim([-0.05 0.05]);
title("Trajectory Of Rotation Motion Towards Left");
xlabel("X Coordinate");
ylabel("Y Coordinate");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;

subplot(1,2,2);
plot(rotationAnglesRotationLeft);
hold on;
plot(tRL);
title("Rotation Angle For Rotation Motion Towards Left");
xlabel("Iterations");
ylabel("Rotation Angle (theta)");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;


figure;
subplot(1,2,1);
plot(xPositionsRotationRight, yPositionsRotationRight);
hold on;
plot(xRR, yRR);
ylim([-0.05 0.05]);
```

```
title("Trajectory Of Rotation Motion Towards Right");
xlabel("X Coordinate");
ylabel("Y Coordinate");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;

subplot(1,2,2);
plot(rotationAnglesRotationRight);
hold on;
plot(tRR);
title("Rotation Angle For Rotation Motion Towards Right");
xlabel("Iterations");
ylabel("Rotation Angle (theta)");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;


figure;
subplot(1,2,1);
plot(xPositionsMixed, yPositionsMixed);
hold on;
plot(xM, yM);
ylim([-0.05 0.05]);
title("Trajectory Of Mixed Motion");
xlabel("X Coordinate");
ylabel("Y Coordinate");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;

subplot(1,2,2);
plot(rotationAnglesMixed);
hold on;
plot(tM);
title("Rotation Angle For Mixed Motion");
xlabel("Iterations");
ylabel("Rotation Angle (theta)");
grid on;
legend("Estimated Trajectory", "Desired Trajectory");
hold off;
```