University of California, Riverside

EE/ME144/EE283A
Foundations of Robotics

Fall 2021

Lab 3 Report

Oct 22, 2021

Name	SID	Section	Group Number
Shaheriar Malik	862154387	Thursday	

1. Problem Statement

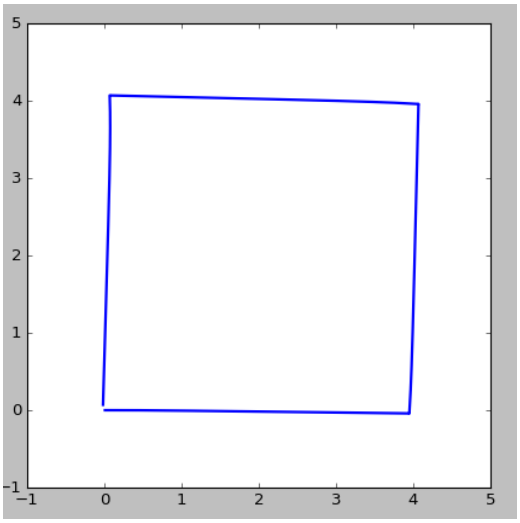
In this lab, we learned how to write a Python script to control the robot. The task was to make the robot move in a square shape using closed-loop control (i.e. letting the robot decide the orientation). The waypoints to visit were [4, 0], [4, 4], [0, 4] and [0, 0]. This meant we needed to make the robot go straight, and then after it reaches the waypoint, turn 90 degrees. This was done 4 times for 4 waypoints.

2. Design Idea

Our code contained a for loop for the target destinations, and a nested while loop to manipulate the robot's linear and angular velocities. We set the PD to 1 and 0.2 because they worked the best in our case. It was mentioned in lab that P should be closer to 1 and D closer to 0. Inside the while loop we first found the target angle using arctan of the difference between the target x and y and the current x and y, then checked if the target angle was different than the current angle. If they were then we assigned the angular velocity to update the value using vel.angular.z = ctrl.update(self.pose.theta) to turn left. If the difference between the angles was below the threshold (0.05), the linear velocity is set to 0.5 and the robot goes straight. If the robot reached its destination, then we set the linear velocity to 0 and break from the while loop to go to the next iteration of the for loop. After the for loop finishes, we send one more command to set the linear and angular velocity to 0 and end the program.

3. Results

The lab was successful where the robot reached all four waypoints within the margin of error. Since the speed was set to 0.5, the total time of completion was 36 seconds. The program is designed for the user to be able to put as many target waypoints as desired and the robot will travel to every one of them to be as modular as possible. Following is the plot of the robot's trajectory:



We can see that the robot successfully draws a square shape on its trajectory reaching waypoints [4,0], [4,4], [0,4], [0,0].

To set up the demo, we first ran the TurtleBot on Gazebo using roslaunch ee144f21 gazebo.launch, then launched the python script using python closed_loop.py. During the demo we explained our code in detail as well as showed that we achieved the desired results.

4. Appendix (optional)

Scripts

Following is the run function we used to command the robot.

```
def run(self):
    vel = Twist()
    ctrl = Controller()
    target = [[4,0],[4,4],[0,4],[0,0]]  #list of all waypoints to visit
    ctrl.setPD(1,0.2)   #setting Kp and Kd
    for i in range(len(target)):
        ctrl.setPoint(math.atan2(target[i][1]-self.pose.y, target[i][0]-self.pose.x)) #setting target point
        while(1):
            theta = math.atan2(target[i][1]-self.pose.y, target[i][0]-self.pose.x)  #calculating angle using arc tan of target - current coordinates
            if (abs(self.pose.theta-theta) > 0):
                vel.angular.z = ctrl.update(self.pose.theta)    #if current value is less than target then turn left
            self.vel_pub.publish(vel)
            self.rate.sleep()
            if (abs(self.pose.theta-theta) < 0.05):
                vel.linear.x = 0.5  #if current theta - target theta is within the threshold then go straight
            if ((abs(self.pose.y-target[i][1]) < 0.1 and abs(self.pose.x-target[i][0]) < 0.1)):
                vel.linear.x = 0
                self.vel_pub.publish(vel)   #if waypoint is reached then stop and break from while loop
                self.rate.sleep()
                break
    vel.linear.x = 0
    vel.angular.z = 0
    self.vel_pub.publish(vel)   #after the sequence is finished stop the robot
    self.rate.sleep()
    # add your code here to adjust your movement based on 2D pose feedback
    pass
```