Darrien Lee, Daniel Kim

CSCI 3302 Robotics

10/20/2020

**CSCI 3302 Robotics Lab 3 Report**

1. My controller for the turn\_drive\_turn controller may fail
2. The position error is the distance between the target and the epuck robot.
3. The heading error is the difference between the front of the epuck robot and the front of the target (in other words, we want the robot to face the same direction as the target).
4. We include bearing error because we need to calculate the angle so that the epuck robot can figure out the direction of the goal.
5. First, we made sure to switch the theta that needed to be accessed (bearing error vs heading error) depending on the distance and gave it a threshold for when it should change (0.05 m). Then we calculated all of the inverse kinematics of the robot in relation to the target: phi of left and right wheels, and the normalized phi for both wheels (to get a ratio of speed the wheels must move). Then we set velocity to the max wheel speed of the left and right wheels multiplied by the normalized ratio of each wheels to move the epuck robot, synchronously reducing rotational and distance errors.
6. We had some problems with implementing all three errors at the same time. We also struggled with how to use the inverse kinematics equations to this scenario. In the end, we got Part 3 to work, even though we were getting similar simulations as Part 2 when we started.
7. Equations for the final controller:

**phi\_left** = (Xr - (theta\_change \* EPUCK\_AXLE\_DIAMETER) / 2) / EPUCK\_WHEEL\_RADIUS

**phi\_right** = (Xr + (theta\_change \* EPUCK\_AXLE\_DIAMETER) / 2) / EPUCK\_WHEEL\_RADIUS

**normalized\_phi\_left** = phi\_left / (abs(phi\_left) + abs(phi\_right))

**normalized\_phi\_right** = phi\_right / (abs(phi\_left) + abs(phi\_right))

**Xr** = min(distance\_error, 0.05)

**theta\_change** = (0.1-distance\_error)/0.1 \* heading\_error + distance\_error/0.1 \* bearing\_error

1. If we alter the gain constants, the epuck robot will move according to the weights of the different errors, and will prioritize minimizing one error over another (ie bearing and heading errors).
2. If you increase specific gain constants, the controller will prioritize reducing that specific error term that correlates with that gain constant. However, if the gain constants become too large, then the epuck may never reach the goal as the constants would outscale any of the errors and break the manipulation of the errors towards our target.
3. If there’s an obstacle between the robot and its goal, it would never reach its goal without necessary code to avoid the obstacle.
4. We could implement obstacle avoidance using light sensors to change to avoid the obstacle while minimizing the distance error as much as possible (with a threshold of current distance error plus the size of the obstacle, just for a rough estimate) so as to minimize the chances of the epuck moving way out of path of the goal and reduce the time and/or distance lost when avoiding the obstacle.
5. The obstacle avoiding robot would move around the obstacle as close as possible (whether towards north or south) to the goal and would allow the distance error to the target become larger, as long as the distance error is within the given threshold above (size/length of obstacle + distance error to target).
6. We spent roughly 6 hours total working on this lab.