Task 1 Questions

1. The robot doesn't know about room coordinates. How does the robot attempt to establish a fixed frame? How did you specify the goal point in your code? Does the target point exist at a definite location in the room, or is it possible that the target point might subtly shift within the room from run to run?

The robot uses its initial orientation and position as the fixed frame, or the world frame. The goal point is thus specified as (2, 1) in world coordinates. The task here was to drive the robot two meters forward and one meter to the left. This target point is arbitrary. From wherever the robot is placed, and whatever direction the robot is facing, it will attempt to find a target that is two meters ahead of it and one meter to the left. This means that unless the robot is placed on the floor every time in the exact same way (same position and orientation) then the target point will always be different in the frame of the room.

2. Why is it OK for the second control law not to have a sgn(tx) term multiplying bx?

The second control law does not need to have a sgn(tx) term multiplying bx because the robot has already turned to face the target point, meaning that the target point is in front of the robot. This means that tx cannot be negative.

3. Why is it OK for the second control law for x˙ to ignore ty, the robot-frame y-coordinate of the target? What assumptions are being made here?

The second control law for x' can ignore ty, the robot-frame y-coordinate of the target because the robot has already turned itself to face toward the target. It makes an assumption that the trajectory of the robot is already pretty close to being straight towards the target point. If there is any error in ty that cannot be ignored, the control law for theta' should be able to readjust in order to make it up.

4.2 Questions

1. In the starter code, what keeps the position of the goal in the world frame fairly constant over time?

The goal position is a function of the robot's current position which comes from the odometry information, odom_listener. The goal position is saved in the world frame which should not change. The origin of the world frame is where the robot starts.

2. Why might it slowly drift?

The odom_listener probably isn't perfect so when the robot moves there can be differences between the estimates in the change in position and the actual motion of the robot. This can be due to imperfections in the sensors or even wheel slippage.

4.5 Questions

1. What problems did you run into along the way when developing your controller? How did you solve them?

The first challenge was choosing a target point. The method that made the most sense to us was to understand that the goal, ball, target should all be in a straight line. If the robot is stationary and spins around to identify the goal and the ball from where the robot is, we can come up with some world coordinates for the goal and the ball. We wanted to draw an imaginary line connecting those two points, and extend it by 0.8 meters. We solved this problem using similar triangles geometry. We subtracted the x and y coordinates for the goal and the ball to get a dx and a dy, and used the distance formula to find some hypotenuse c. The hypotenuse of the larger triangle we knew would be c+0.8m so we can multiply by a factor of (c+0.8)/c to find dx and dy of the larger triangle. This gives the target location in world coordinates, which we then convert to robot coordinates. After we managed to drive the robot to the target, scoring the goal was a matter of driving into the ball. This part was simpler…we turned the robot to face the ball, and drive straight forward. A high speed was necessary to apply enough force to the ball in the correct direction, otherwise the ball would sometimes roll sideways or get stuck to the robot. We also found that the error in the trajectory of the robot was best when we kept the turning (to the target point, and to the ball) speed very slow.

2. How did you use debugging tools (e.g. ‘print‘ statements, the plotter) to aid your development?

We used rospy.loginfo to print out all of our variable values when calculating the target point, such as ball location and goal location in the world frame and other preliminary variables related to the calculation of the target location, i.e dx, dy, c, c’, factor of c’/c. Afterwards, to verify that the calculations are correct, we plotted the ball, goal, and target locations onto symbolab. If these points lined up in a straight line, we knew the calculation was done correctly. We also used rospy.,loginfo to find a good time duration for the robot to drive forward when hitting the ball.

Part 5 Description

To do something beyond pushing the ball into the goal, we decided to score a bank shot off of the wall. The setup is the ball set up next to (not in front of) the goal, and we placed each cone and the ball a distance of 0.7 meters away from the wall. We decided to keep the distance from the wall the same for every run so we wouldn’t have to put tape on the walls or anything like that to detect another blob to get the distance. To score the bank shot, we needed to find a target point from where the robot could push the ball into the wall, so it bounces at an angle to go through the goal. This location on the wall we decided would be a point that is perpendicular to the line connecting the ball and the goal, and a distance of 1 meter away from the midpoint of the ball and the goal. This took a bit of math…the midpoint coordinates can be found from the average of the ball and goal world coordinates, and we needed to create a vector perpendicular to the vector connecting the ball and goal. Then, we find the ratio of 1 meter over the distance between the ball and the goal and multiply it by the perpendicular vector. This result is added to the midpoint, finally giving us the point on the wall where the ball should bounce. To find the target point, we needed to extend the vector connecting the wall point and the ball by 0.8 meters. We did this by creating similar triangles in a fashion similar to task 2. Hitting the ball was not so easy…the ball was not bouncy enough, or we could not hit the ball hard enough to make it bounce off of the wall. In a lot of the trials, the ball would just roll into the wall and stop because it didn't have enough momentum. The first thing we did to try and make this work better was dramatically increasing the speed of the robot, by allowing it to accelerate faster. We decreased the distance from the ball and goal to the wall to just 0.7 meters, so that the ball would not have to roll as far, giving it less time to slow down. We also increased the distance from the target point to the ball from 0.8 to 1.1 meters, so the robot would have more time to reach a furious speed and hit the ball harder. Finally, we taped a wooden block to the front of the robot to act as a bumper, so the robot would make a higher contact point with the ball, so the ball would roll better without sliding on contact.

Links to the Videos

https://drive.google.com/drive/folders/162NG1BJaMAGIWtEHBWGorlK9ZqnvbjhA?usp=sharing