EECS 376
Andrew Wood
aew61
Homework 3

This assignment did not involve as much thinking as the previous one. This assignment involved mostly finding a way to convert a list of waypoints into a list of (theta, dist) pairs that could then be used to turn and move the robot. The most difficult part of this assignment was organizing the amount of pairs that were necessary given (x, y, phi) and in the specific order with which they are to be executed. The algorithm I settled upon is as follows:

```
generate_commands(list of (x, y, phi) waypioints):
    current_pose = (position = (0, 0, 0), orientation = (0, 0, 0, 1))

    generated_cmds = ()  // empty list
    for each waypoint:
        angle_to_rotate = point_at_next_coords(current_pose, waypoint.position)
        generated_cmds += (angle_to_rotate, 0)

        distance_to_travel = distance(current_pose.position, waypoint.position)
        generated_cmds += (0, distance_to_travel)

        // MUST set this to reflect pointing at coords when spinning for final pose!
        current_pose.orientation = angle_2_quat(angle_to_rotate +
            quat_2_angle(current_pose.orientation))

        // assume we are now at point (more or less)...spin to desired pose
        angle_to_rotate = compute_spin(current_pose, waypoint.orientation)
        generated_cmds += (angle_to_rotate, 0)

        current_pose = waypoint
```

This algorithm is (besides “greedy”) another blind algorithm. Its only saving grace is that there is the smallest feedback loop built in, and that is taking into account final poses for waypoints. Technically, we could just ignore that pose, but in so doing, we are helping our algorithm converge a little bit more because we can sort of “correct” drift since we cannot stop exactly at the orientations that we want. This algorithm is supported by vector math, pointing towards a new set of coordinates required a small amount of vector math (computing the angle between two vectors) that was a little tedious for me to write (I did not use the tf package).

When running, I noticed that my robot tended to get stuck every once in a while. This is because I am only checking movement updates every 10Hz and am moving slightly too fast for that to be effective all of the time. Theoretically, I could lower the yaw rate and linear velocity, but that would just make my video longer and it still converges most of the time.