E206 Lab 1: Trajectory Tracking

Shivam Malpani and Nathan Sunbury

***Abstract*— This paper proposes a proportional control algorithm for trajectory tracking using a differential drive robot. The algorithm supports tracking for both forward and backward motion of the robot. The performance of the algorithm was measured by calculating the root mean square error of the robot's actual position compared with the planned trajectory as it traversed a 5.8 meter euclidean distance. Over the course of the trajectory, the robot had an RMS error in its (x, y, θ) position and orientation of $0.154m$, $0.096m$, and $0.069rad$ respectively.**

I. INTRODUCTION

In order for robots to move quickly, safely, and efficiently in the world, they must plan the path they will take through their environment. These paths, often referred to as trajectories, consist of a list of points that describe the position and orientation of the robot at every step along a path from a starting point to a desired end point. In this report, trajectories were generated from Dubins curves, the shortest path between any two configurations given a fixed nonzero turning radius and no lateral motion [1]. These curves were discretized into distinct (x, y, θ) points and assigned a time stamp based on the total amount of time allotted to follow the curve. This yielded a set of (t, x, y, θ) points that formed a trajectory for the robot to follow.

The purpose of this report is to develop and validate a proportional control algorithm that will allow a differential drive robot to follow these trajectories autonomously.

II. METHOD

A. Proportional Control for Point Tracking

In order to track the planned trajectory, a proportional control algorithm was designed and applied. Proportional control algorithms use the difference between the desired state and the actual state to calculate the control signal. In the case of a differential drive robot, there are three independent variables (x, y, θ) which describe the state of the system. A differential drive robot cannot move freely in any of these directions, however, it can only move forward or backward with velocity (v) or can rotate with angular velocity (ω). It cannot move laterally. Thus, a different coordinate system was chosen to correspond better with the actual motion of the robot. Equations 1-3 show how the new coordinate system (ρ, α, β) was derived from the robot's (x, y, θ) coordinates.

$$\rho = \Delta x^2 + \Delta y^2 \tag{1}$$

$$\alpha = -\theta + atan2(\Delta y, \Delta x) \tag{2}$$

$$\beta = -\theta - \alpha + \theta_{desired} \tag{3}$$

Figure 1 shows graphically how the (ρ, α, β) coordinate system relates to the forward/backward and rotational motion of the robot (v, ω).

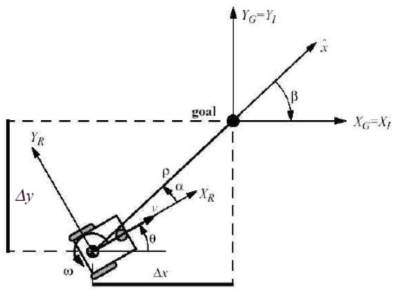

Fig. 1. Diagram of (x, y, θ) and (ρ, α, β) coordinate systems [2]

By applying the kinematics of the differential drive robot, a system of differential equations was created to describe how the robot's linear and angular velocity (v, ω) affect the state of the robot (ρ, α, β) (equation 4).

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -cos(\alpha) & 0 \\ \sin(\alpha)/\rho & -1 \\ -\sin(\alpha)/\rho & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4}$$

To control this system, the control law shown below was proposed (equation 5).

$$v = k_\rho \rho, \qquad \omega = k_\alpha \alpha + k_\beta \beta \tag{5}$$

Applying this control law and solving for the eigenvalues of the state matrix shows that the system will be stable if the conditions in equation 6 are met.

$$k_\rho > 0, \quad k_\beta < 0, \quad k_\alpha > k_\rho \tag{6}$$

In order to actually control the robot, the robot's overall (v, ω) velocities had to be converted to right and left wheel velocities (w_r, w_l). The relationship between the wheel's angular velocity (w), velocity (v), and the distance between wheels (L) is shown below:

$$w_r = \frac{v_r}{2 \cdot L}, \quad w_l = -\frac{v_l}{2 \cdot L} \tag{7}$$

A system of equations can then be created using equation 7 and the overall kinematics of the robot (equations 8,9).

$$v_{robot} = (w_r - w_l) = L\left(\frac{v_r}{2 \cdot L} + \frac{v_l}{2 \cdot L}\right) = \frac{v_r + v_l}{2} \tag{8}$$

$$\omega_{robot} = w_r + w_l = \frac{v_r - v_l}{2 \cdot L} \tag{9}$$

The calculation of (ρ, α, β) described previously assumes that the robot will drive forwards towards the desired point.

Because the differential drive robot has the capability to move both backwards and forwards in the direction it is facing, the equations for calculating (ρ, α, β) can be adjusted to shift the robot's reference frame accordingly.

B. Trajectory Tracking

Trajectory tracking is a natural extension of point tracking. Instead of moving towards a single point, the robot leverages point tracking to follow a series of points along a path towards an ultimate goal. At any given time, the desired point is set as the next chronological point in the trajectory to minimize the error between the current position and this desired point.

III. Experiments and Results

A. Experimental Setup

The performance of the control algorithm was evaluated in an open source physics simulator and test environment using a simulated differential drive robot [3].

B. Point Tracking Results

Using the previously described control algorithm, the robot was successfully able to track points in its environment. Following the path created by the stable system, the robot successfully reached the sequence of points from $(0, 0, 0)$ to $(2, 0, 0)$, $(2, 2, 0)$, $(0, 0, 2, \pi/2)$, $(0, -2, 2, 0)$, $(0, -2, 0, 0)$, $(0, 0, -2, \pi/2)$, and $(0, 2, -2, 0)$, returning to the origin after each point. The plot below shows the performance of robot.

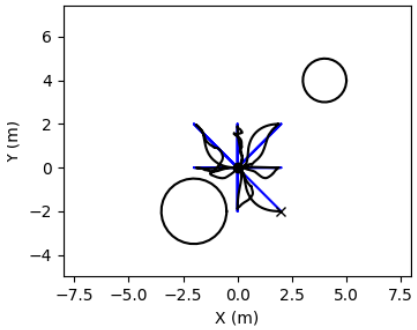

Fig. 2. Results of tracking 7 points, returning to the origin after each one

The robot successfully reached these points to within a predetermined margin of error of $0.1m$ (measured using the euclidean distance) from the point and $0.10rad$ from the desired orientation.

C. Trajectory Tracking Results

The robot was successfully able to track a trajectory in its environment. Trajectory tracking was implemented by leveraging the point tracking control algorithm to sequentially track points along the desired path by assigning the desired state of the robot based on the current timestamp and the calculated trajectory. To test the performance of the trajectory tracking algorithm, the robot was instructed to follow a Dubins trajectory from $(0, 0, 0)$ to $(5, -3, 0)$ in 20 seconds. Figure 3 shows the results of this test.

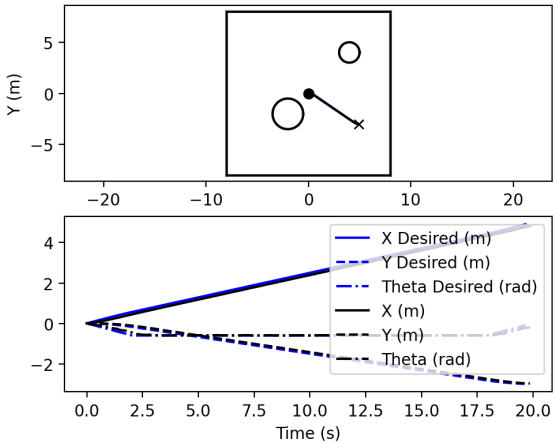

Fig. 3. Results of tracking a trajectory from $(0, 0, 0, 0)$ to $(20, 5, -3, 0)$. Top: Map of environment and Dubins path followed by robot. Bottom: Actual vs Planned (x, y, θ) coordinates over time.

Table I shows the root mean square error of the (x, y, θ) coordinates over the course of the trajectory.

Variable	RMS Error
x	0.154 m
y	0.097 m
$\boldsymbol{\theta}$	0.069 rad

TABLE I

RMS Error of (x, y, θ) over the trajectory in figure 3

IV. Conclusions

The control algorithm developed in this paper was able to successfully guide a robot along a trajectory. Future directions for this project could include improving the robustness of the trajectory tracking algorithm by adding obstacle avoidance or the ability to re-plan its trajectory on the fly.

Acknowledgment

Thanks to Prof. C. Clark for his assistance with completing and understanding this lab and to Seth Isaacson for his assistance with software environment installation and troubleshooting.

References

[1] S. LaValle, "15.3.1 Dubins Curves," *Planning Algorithms*, [Online], Available: http://planning.cs.uiuc.edu/node821.html. [Accessed Feb. 14, 2020]

[2] C. Clark, "E206 Lectures: Lecture 2A," *E206 Class Website*, [Online], Available: https://sites.google.com/g.hmc.edu/e206-motion-planning/lectures?authuser=0. [Accessed Feb. 7, 2020]

[3] "Specifications," *Fetch Robotics*, [Online], Available: https://fetchrobotics.com/. [Accessed Feb. 10, 2020]