Differential Drive Tracking Controller

Autonomous Robot Navigation (E160) Lab 3 Report

Sean Mahre
Department of Engineering
Harvey Mudd College
Claremont, CA
smahre@hmc.edu

Eyassu Shimelis
Department of Engineering
Harvey Mudd College
Claremont, CA
eshimelis@hmc.edu

***Abstract*—The goal of this lab is to design and test a point controller for a differential drive robot. Differential drive robots have fewer controllable degrees of freedom than the state space, which adds a nonholonomic constraint. This lab details the design and implementation of a point tracker in both a simulated and physical differential drive robot. The controller can effectively track points both in front of and behind the robot. The final controller was tested on a series of eleven heading waypoints and thirteen position-heading waypoints. Both the simulated and physical robot successfully tracked all of the desired waypoints. The robot, unlike the simulation, exhibited an absolute steady-state error of $0.11 \text{ rad} \pm 0.01$ and $0.03 \text{ m} \pm 0.01$ in its heading and euclidean distance respectively.**

I. Introduction

An important component of robotic systems is the ability to navigate a robot towards a desired state. These states may be physical (e.g. three-dimensional position of an underwater position), or even more abstract (e.g. a utility function that an agent is attempting to maximize). It is important to effectively navigate through a state-space efficiently. This problem becomes especially more complex when mobility is limited.

The purpose of this lab is to design and test a point tracker for a differential drive robot.

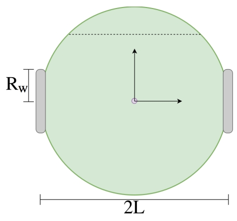
II. Background

A. Differential Drive

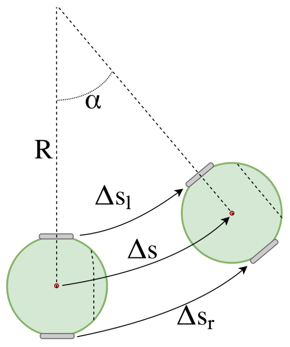
A common drive type for small, wheeled robots is a differential drive system. Differential drive robots are equipped with two wheels, each spaced a distance L from the center of the robot. Figure 1(a) shows the two important components of a differential drive system: wheel radius and wheel separation. Although a differential drive robot is capable of controlling two degrees of motion: v and ω, we are interested in controlling it in a higher dimensional state-space; namely, ξ. The robot state in the inertial frame ξ_i is dependent on its position and bearing, as shown in Eq. 1.

$$\xi_i = [x, y, \theta]^T \tag{1}$$

The robot's motion in three dimensional space can be modeled as traveling along the circular arcs with varying radii, as shown in Figure 1(b).



(a) Differential drive diagram.



(b) State estimation problem.

Fig. 1: Diagram of a differential drive robot. Each wheel has a radius R_w, separated by a distance of $2L$.

This type of motion can then be projected into the inertial frame using the following kinematic model:

$$\Delta\xi_i = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \frac{\Delta s_l + \Delta s_r}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_r}{2L}\right) \\ \frac{\Delta s_l + \Delta s_r}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_r}{2L}\right) \\ \frac{\Delta s_r - \Delta s_r}{2L} \end{bmatrix}, \tag{2}$$

where

$$\begin{aligned} \Delta s_l &= R\alpha \\ \Delta s_r &= (R + 2L)\alpha \\ \Delta s &= (R + L), \alpha \end{aligned}$$

ΔS_L and ΔS_R in Eq. 2 are the distances traveled by the left and right wheel, respectively. Although this drive type is limited in its local mobility, it still retains access to the general state-space. The robot used in this report has a wheel radius of $R_w = 3.478 \text{ cm} \pm 0.001$ and a wheel separation radius of $L = 7.075 \text{ cm} \pm 0.001$.

III. Problem Definition

The goal of this lab is to design a closed loop controller, which will drive the robot to any desired state $\xi_d = [x_d, y_d, \theta_d]^T$.

Differential drive systems are nonholonomically constrained; that is, the total number of controllable degrees of freedom is less than the dimension of the state-space. Designing a closed loop controller with nonholonomic constraint; therefore, is less intuitive.

A diagram of the problem statement is illustrated in Figure 2. The controller must use the two controllable degrees of freedom—v (forward velocity) and ω (angular velocity)—to navigate to an arbitrary desired state.

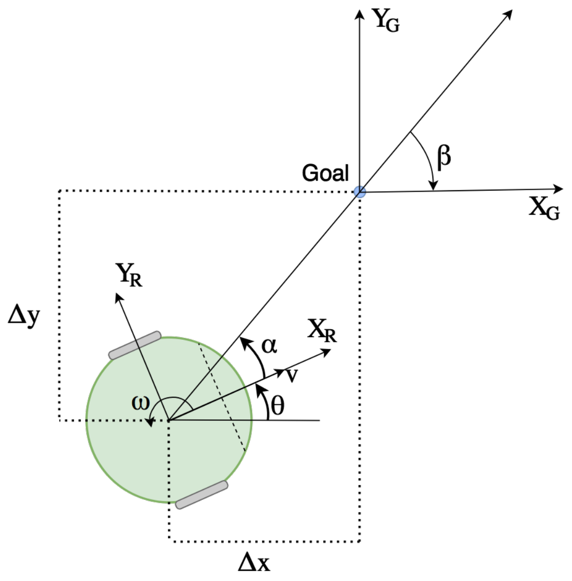


Fig. 2: Problem statement diagram. The robot must navigate to an arbitrary desired goal state.

IV. Control Design

Given a robot's current and desired state, ξ and ξ_d, respectively, we can define the error as the difference between the two, as shown in Eq. 3. Note that the robot's bearing is wrapped in the range $\theta \to (-\pi, \pi]$.

$$\mathbf{e}(t) = \xi_d - \xi = [x_d - x, y_d - y, \theta_d - \theta]^T \tag{3}$$

With this state error, the controller must implement some gain matrix K, which will drive the error to zero.

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = K\mathbf{e}(t) \tag{4}$$

The forward kinematic model of a differential drive robot is shown in Eq. 5 below.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5}$$

The same kinematic model can be expressed in the following polar coordinates, where the polar variables are illustrated in Figure 2:

$$\begin{aligned} \rho &= \sqrt{\Delta x^2 + \Delta y^2} \\ \alpha &= -\theta + \arctan 2(\Delta y, \Delta x) \\ \beta &= -\theta - \alpha, \end{aligned}$$

Ensuring that $[\rho, \alpha, \beta]^T \to [0, 0, 0]^T$, the system will drive to the desired state [1].

The new transformed kinematics—in polar coordinates—can be rewritten in terms of v and ω [1] in Eq. 6.

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \sin\alpha/\rho & -1 \\ -\sin\alpha/\rho & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \text{ for } \alpha \in (-\pi/2, \pi/2] \tag{6}$$

Following the proofs in [2], the control gain K becomes

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} k_\rho & 0 & 0 \\ 0 & k_\alpha & k_\beta \end{bmatrix} \begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix}. \tag{7}$$

As shown in [1], for a linearized version of this system, it is only stable under the following conditions:

$$\begin{aligned} k_\rho &> 0 \\ k_\beta &< 0 \\ k_\alpha - k_\rho &> 0 \end{aligned}$$

A. Path Optimization

The desired goal state may sometimes lie behind the robot. With the control law derived in Eq. 7, the robot will take path p_1, in Figure 3; however, an effective controller should allow the robot to also drive backwards, along path p_2, in order to minimize the path taken to the goal.

The controller is effective for desired points that are in front of the robot; however, the robot will always take a longer path than simply driving backwards. To allow the robot to also drive backwards, the control law must be modified for $|\alpha| > \pi/2$.

In these cases, two changes must be made to the control law [1], redefining α and reversing the velocity v.

$$\begin{aligned} \rho &= \sqrt{\Delta x^2 + \Delta y^2} \\ \alpha &= -\theta + \arctan 2(-\Delta y, -\Delta x) \\ \beta &= -\theta - \alpha, \end{aligned}$$

and

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} -k_\rho & 0 & 0 \\ 0 & k_\alpha & k_\beta \end{bmatrix} \begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix}. \tag{8}$$

In order to also track some desired angle—both forwards and backwards—simply add θ_d to β.

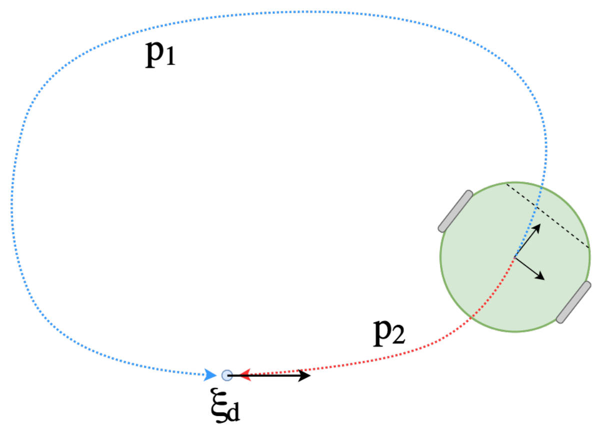


Fig. 3: The control law must be able to handle desired points that lie behind the robot, with the goal of minimizing the path to the goal.

V. Implementation

The motion control described above was implemented in a small differential drive robot, see Figure 4. Each wheel is equipped with a Hall Effect wheel encoder that has a resolution of approximately $r_e = 1440$ ticks/rev ± 10. For this lab, the robot was localized using only the odometry readings. Previous work with this robot [3] has demonstrated that odometry-only localization is very accurate, with an absolute positional error of 0.3 cm $\pm\, 0.1$ per meter traveled.

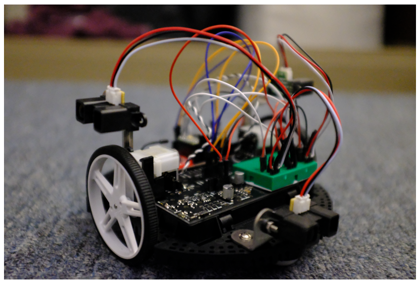


Fig. 4: The E160 robot, equipped with three IR distance sensors and two wheel encoders.

The controller was implemented in Python, running on a separate computer. The E160 robot houses an Arduino mini and XBee wireless microcontroller. All of the computation is done off-board. The results, which are motor speeds for each wheel, are transmitted wirelessly to the robot at a rate of 10 Hz. The main control loop is outlined in Alg. 1, below.

Further expanding on the control update step, the point tracker is outline in Alg. 2, below.

As seen in Alg. 2, ρ and α are set to zero when the robot is close enough to the desired point. This is done because ρ and α help the robot approach the desired point in the right direction, but act sporadically because small values around zero for Δx and Δy in will cause α to jump around because $\arctan 2(\Delta y, \Delta x) = \arctan(\Delta y/\Delta x)$.

Algorithm 1 Main Control Loop

```
1: function MAIN
2:     $z, r \leftarrow$ GetSensorMeasurements()    ▷ update sensor readings and encoder counts
3:     $\xi_t \leftarrow$ Localize($\xi_{t-1}, z$)    ▷ localize robot
4:     $R, L \leftarrow$ UpdateControl($\xi_t, \xi_d$)    ▷ set left and right wheel speeds wait until 0.1 seconds have elapsed since start of loops
5: end function
```

Algorithm 2 Point Tracker

```
1: function UPDATECONTROL($\xi_t, \xi_d$)
2:     $\xi_e = \xi_d - \xi_t$    ▷ Calculate state error
3:     if $\xi_e >$ minDistThreshold then
4:         $\alpha = \theta_t - \arctan 2(y_e, x_e)$
5:         if $-\pi/2 < \alpha \leq pi/2$ then
6:             $\rho, \alpha, \beta \leftarrow$ ForwardsControlLaw()
7:         else
8:             $\rho, \alpha, \beta \leftarrow$ BackwardsControlLaw()
9:         end if
10:    else▷ Robot has reached desired point, tracking angle
11:        $\beta = -\theta_e$
12:        $\rho, \alpha = 0$
13:    end if
14:    $v, \omega \leftarrow$ UpdateVelocities($\rho, \alpha, \beta$)
15:    return $R, L \leftarrow$ UpdateWheelVelocities($v, \omega$)
16: end function
```

VI. Results

The point tracking control algorithm was tested in both simulation and hardware. The robots were tested by driving to a series of waypoints, both forwards and backwards. The main interface for the E160 robot is a Python graphical user interface (GUI), shown in Figure 5.

Figure 6 shows an overhead path multiple starting positions, all tracking the origin. Each robot starts at the edge of a circle of radius 0.25 m, with its starting heading tangent to the circle.

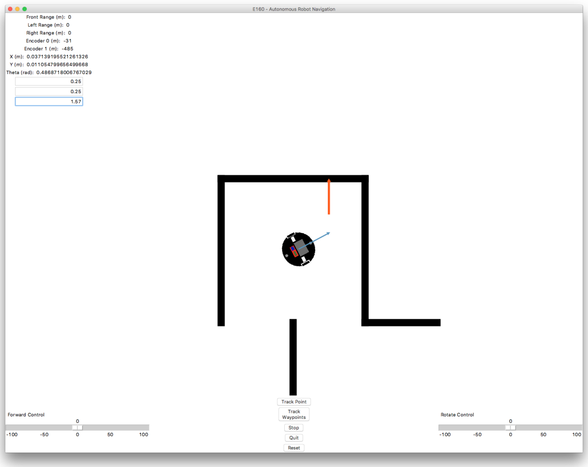
A. Simulation

Both the simulated and actual E160 robot use the very same control update laws, the only difference is the control gains used. The simulated robot is modeled to follow the same kinematics described in Section II-A. The control gains for the simulation are:

$$k_\rho = 1.2$$
$$k_\alpha = 3$$
$$k_\beta = -1.3$$

The first set of tests was tracking just a desired angle. The simulation was run on a series of desired heading waypoints ($\theta = -\pi$ to π). The results are shown in Figure 7 below.

Figure 7 shows that the controller can reliably track any desired heading, in simulation. Next, the simulation was tested



Front Range (m): 0
Left Range (m): 0
Right Range (m): 0
Encoder 0 (m): -31
Encoder 1 (m): -485
X (m): 0.037139195521261326
Y (m): 0.011054799656499668
Theta (rad): 0.4868718006767029
0.25
0.25
1.57

Fig. 5: Simulated E160 robot driving towards a desired point, marked by an orange arrow. The start of the arrow lies at the desired position, and the angle points in the direction of the desired heading.

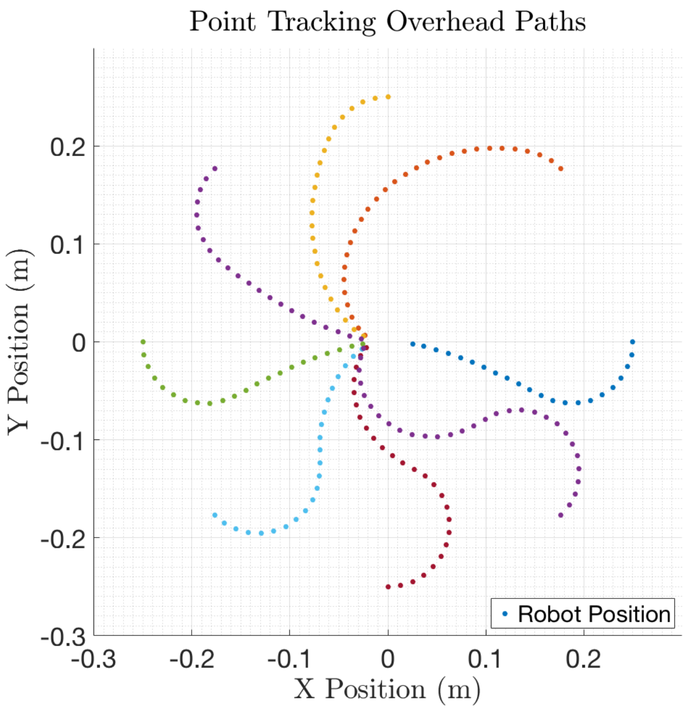


Fig. 6: Simulation results for a series of desired headings.

on a wide variety of waypoints, forcing the controller to drive both forwards and backwards. The results of this test is shown in Figure 8, below.

In every case, the simulation successfully tracked the desired positions and headings.

B. Hardware

The very same control algorithm was used to point track with a physical differential drive robot. The control gains used

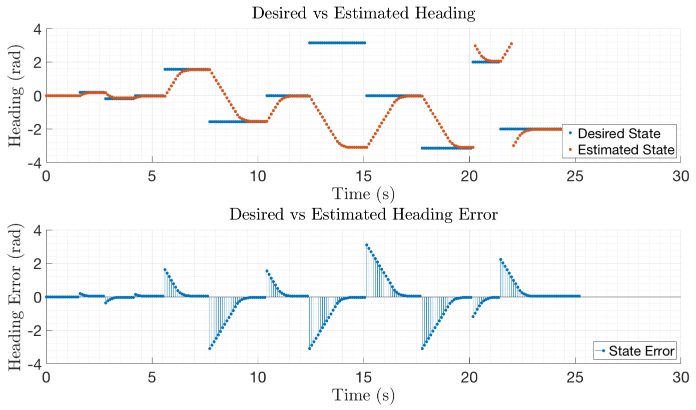


Fig. 7: Simulation results for a series of desired headings.

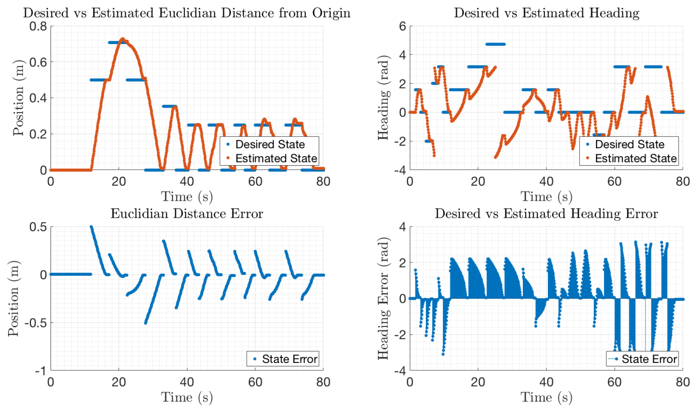


Fig. 8: Simulation results for a series of desired waypoints and headings. A total of 13 unique waypoints were tested.

for the physical robot are:

$$k_\rho = 1.0$$
$$k_\alpha = 2.7$$
$$k_\beta = -2.5$$

The very same angle test shown in Figure 7 was run in the physical robot. The results are depicted in Figure 9, below.

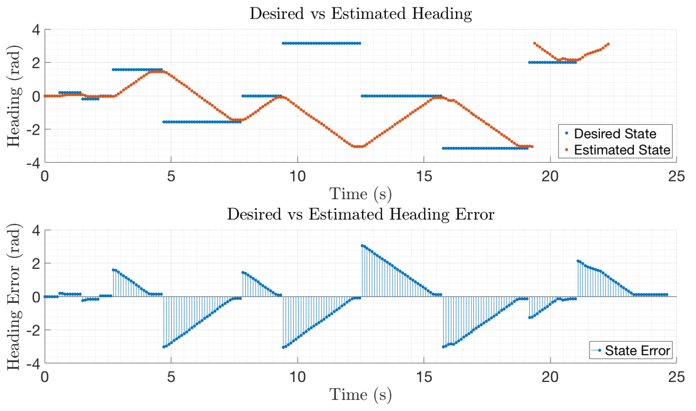


Fig. 9: Hardware results for a series of desired headings.

Like the simulation, the physical robot was able to successfully track all of the desired headings; however, there is some slight steady-state error. This can be minimized by increasing k_β, or altogether eliminated by including and integral controller.

The physical robot was also tested on the same series of waypoints used in Figure 8. Those results are shown below in Figure 10.

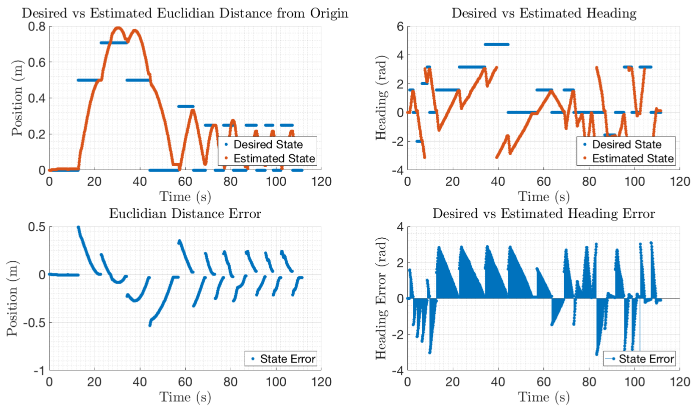


Fig. 10: Hardware results for a series of desired waypoints and headings. A total of 13 unique waypoints were tested.

VII. Conclusion

The point tracking algorithm described in this report is an effective way of efficiently driving a differential drive robot to a desired state. This, however, is not the only type of motion controller for differential drive robots. There are trivial motion controlling algorithms for traversing the state space; for example: turning to face the goal, driving straight to the goal, then turning to face the desired heading.

References

[1] C. Clark, "E160 lecture 4 - state space control & point tracking," 2018.
[2] K. Daniilidis and V. Kumar, "Cis 390 - control intro and appcliation to differential drive vehicles."
[3] S. Mahre and E. Shimelis, "Odometry Calibration and Error Characterization for a Differential Drive Robot," Harvey Mudd College, Tech. Rep., 02 2018.