Odometry Calibration and Error Characterization for a Differential Drive Robot

Autonomous Robot Navigation (E160) Lab 2 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*—This report documents the odometery calibration and testing of a differential drive robot. The robot was tested by driving it to ten different distances, from $50 - 500$ cm. At each point, the estimated distance was compared to the measured distance, yielding an absolute mean error of 0.68 cm $\pm$ 0.2. From each point, the robot was driven back to the origin, and both the (x, y) displacements were compared to the measured displacement from the origin. The results showed an absolute mean error of $(4.4, 6.5)$ cm.**

I. Introduction

The field robotics of robotics has a diverse range of locomotion techniques. From motors to legged motion, different robots require different types of locomotion to traverse their environment. In order to localize themselves, many robotic systems rely on odometry as a means of using varying types of motion sensors to estimate how the robot is moving.

II. Background

A. Differential Drive

The most common drive type for wheeled robots is a differential drive system. These types of robots are driven by two separate wheels, often placed on either side of the robot's chassis. This allows differential drive robots two degrees of freedom; and their state in an inertial, two dimensional space, ξ_i, is

$$\xi_i = [x, y, \theta]^T,$$

where (x, y) is the robot's position, and θ is the robot's heading relative to a global reference frame.

The wheels on a differential drive system are mounted on a common axis and can be driven both forwards and backwards. Since the number of controllable degrees of freedom is lower than the dimension of our state space, this system is nonholonomic. Although this kinematic constraint will limit the local mobility, the general accessibility to the state space remains unaffected [1].

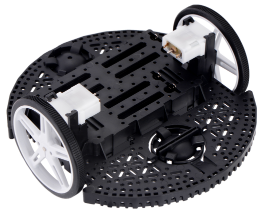
Fig. 1: The E160 robot chassis.

III. Differential Drive Kinematic Model

The robot used in this report is a small, two-pound differential drive robot, shown in Figure 1. The diameter of each wheel is 6.95 cm $\pm$ 0.01 and the distance between the center of the two wheels is 14.15 cm $\pm$ 0.01. Each wheel also has an encoder that is capable of measuring the relative rotation of each wheel with a resolution of 4.36×10^{-3} rad/tick.

A diagram of the differential drive system is shown in Figure 2. The overall movement of the differential drive robot will depend on Δs_l and Δs_r, the distance traveled by each wheel. This model assumes that the wheels do not slip.

Given the radius of each wheel, R_w, the number of measured ticks, d, and the conversion between number of ticks and radians, $\gamma = 4.36 \times 10^{-3}$ rad/tick, we can calculate the distance each wheel travels on the ground using Eq. 1.

$$\Delta s_{l,r} = R_w d \gamma \tag{1}$$

In order to convert the relative motion of each wheel into the desired state space of the robot, $\xi_i = [x, y, \theta]$, we will need to use $\Delta s_{l,r}$ to specify the displacement and rotation of the robot. The robot's motion can be described as motion along circular arcs, where the axis of rotation is called the instantaneous center of curvature [2]. In Figure 3, we see that the robot's new motion is described using Δs and α, both of which can be defined in the global coordinate frame.

We can now define Δs, Δs_l, and Δs_r using the formula

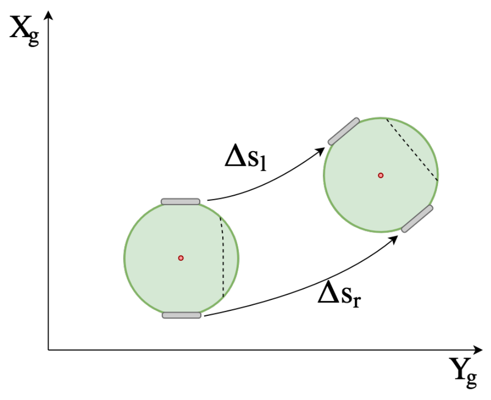

Fig. 2: Differential drive system overview. The robot resides in a global coordinate frame, (X_g, Y_g). As each wheel is spun, the robot's motion will be dictated by the distance that each wheel travels along the ground without slipping.

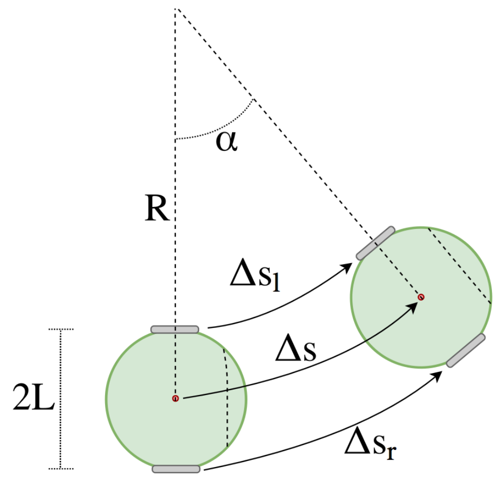

Fig. 3: Drive kinematics of a differential system expressed in terms of motion along a curvature. The robot's path, Δs, is a function of the distance from the center of the curvature, R, and the displaced angle, α.

for calculating arc lengths:

$$\Delta s_l = R\alpha$$
$$\Delta s_r = (R + 2L)\alpha$$
$$\Delta s = (R + L)\alpha$$

The first two equations above can be used to solve for Δs in terms of Δs_l and Δs_r.

$$\Delta s = \frac{\Delta s_l + \Delta s_r}{2} \tag{2}$$

The same set of equations can be manipulated to find α in Eq. 3, which is also equal to the robot's heading displacement $\Delta\theta$ [3].

$$\Delta\theta = \frac{\Delta s_r - \Delta s_r}{2L} \tag{3}$$

For very small motions, we can approximate Δs as a straight line and calculate the displacement in the global coordinate frame (Eq. 4-5):

$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2) \tag{4}$$
$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2) \tag{5}$$

The final motion of the robot in terms of Δs_l and Δs_r, in the inertial frame, can be modeled in Equation 6.

$$\Delta\xi_i = \begin{bmatrix}\Delta x\\ \Delta y\\ \Delta\theta\end{bmatrix} = \begin{bmatrix}\frac{\Delta s_l + \Delta s_r}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_r}{2L}\right)\\ \frac{\Delta s_l + \Delta s_r}{2}\sin\left(\theta + \frac{\Delta s_r - \Delta s_r}{2L}\right)\\ \frac{\Delta s_r - \Delta s_r}{2L}\end{bmatrix} \tag{6}$$

IV. Control Design

A PI (Proportional-Integral) controller was designed to drive the robot to a desired distance, using the odometry estimates. The controller equation (Eq. 7) outputs a control effort, u, which is proportional to the current and integrated distance error, e.

$$u = K_p e + K_i \int e\ dt \tag{7}$$

Although this will move the robot to the desired distance, it will not necessarily travel in a straight line. Small differences in the wheel diameter or power output will cause heading errors that will remain uncorrected. To prevent this, a PID (Proportional-Interal-Derivative) controller was designed for the heading. This controller ensured that the robot maintained its heading as it traveled forward. This controller outputs a control effort that is also proportional to the rate of change of the error, and is shown in Eq. 8.

$$u = K_p e + K_i \int e\ dt + K_d \left(\frac{d}{dt} e\right) \tag{8}$$

The gains for both the distance and heading controllers were adjusted until the performance was within an acceptable range; this will be discussed further in the results section.

V. Software Implementation

The state estimation software for the E160 robot is implemented in Python and runs on a separate computer. The main control loop in Figure 4 runs at a rate of 10Hz.

```
def main_control_loop():
    update_sensor_measurements()
    localize()
    update_control()
    wait100ms
```

Fig. 4: Main control loop for the E160 robot.

VI. Experiment

In order to characterize the errors in odometry, the robot was driven to a set of distances, ranging from $50-500$ cm±2. The experimental setup is shown in Figure 5. For each trial, two sets of measurements were collected. The first measurement is the y error after the robot has moved to the desired distance. The x error isn't measured at this time because it will depend on the original orientation of the robot at the start. After this measurement, the robot was driven backwards with the same controller. After reaching the origin, the x and y displacements were measured. It is important to note that even though the robot did not always return to the origin, the odometry error is the difference between the estimated state, $\xi_{i,est}$, and the actual state, ξ_i. This will be discussed further in the results section.

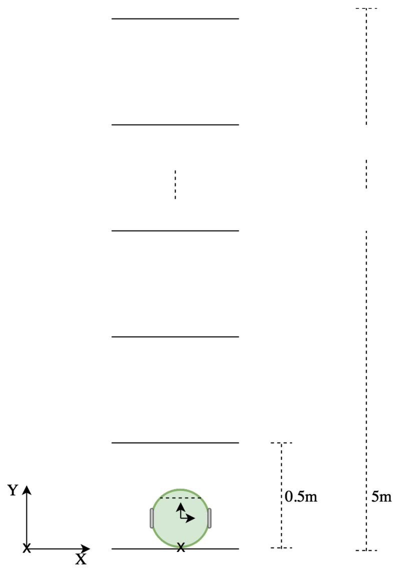

Fig. 5: Experimental setup for odometry error characterization. The robot was driven to each marker and back to the origin. The error was calculated by the final displacement of the estimated and actual states.

A. Results

The robot's desired and estimated states were logged at each time step, and two sets of results are plotted in Figures 6-7. The left-hand side of the plots compare the desired and estimated x, y, and θ. The right-hand side of the plots illustrate the errors.

As expected, we see that the y position of the robot approaches the desired distance, both forwards and backwards.

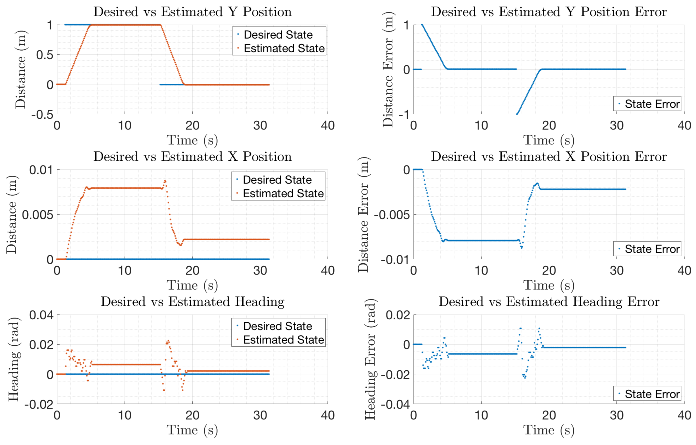

Fig. 6: Comparing the desired and estimated states for 1m run.

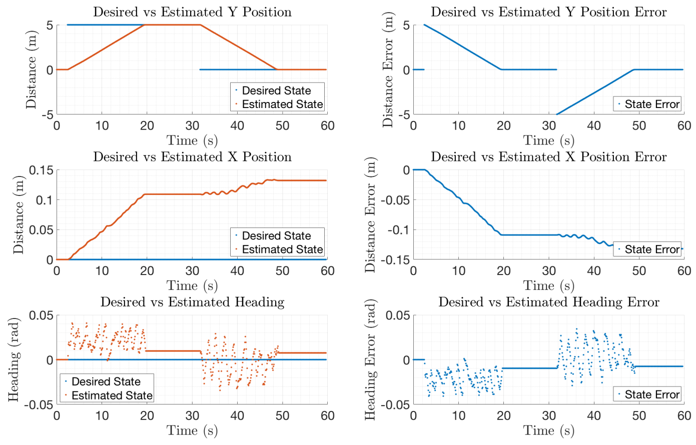

Fig. 7: Comparing the desired and estimated states for 5m run.

As the robot moves from one desired state to another, the heading oscillates around the desired angle, as it attempts to keep the robot moving straight.

Over the course of all ten distances, the overall errors are characterized in Table **??**. The error is defined as the difference between the measured state and the estimated state. The measurements in the table below have an uncertainty of ±0.2 cm

Distance (m)	**Forward y Error** (cm)	**Origin (x, y) Error (cm)**
0.5	0.1	$(-0.7, 2.0)$
1	0.3	$(-1.22, 2.8)$
1.5	0.1	$(-1.4, 4.2)$
2	-0.5	$(0.1, 3.3)$
2.5	0.6	$(-0.2, 7.5)$
3	1.3	$(-2.7, 7.1)$
3.5	1.2	$(-6.6, 1.2)$
4	1	$(-9.2, 8.5)$
4.5	0.5	$(-10.5, 11.9)$
5	1.2	$(-11.6, 16.6)$

The results were only collected for one set of trials. The mean absolute error in the forward odometry distance is $0.68\ \mathrm{cm} \pm 0.2$; however, when the robot drives back to the origin, the mean error is $(4.4, 6.5)\ \mathrm{cm} \pm 0.2$.

VII. Conclusion

From the table above, we see that the origin-displacement error grows with the distance that the robot travels. The forward error remains relatively consistent, not growing more than $2\ \mathrm{cm}$. In the future, this experiment could be improved by additional, repeated testing at each distance, in order to characterize the error in terms of a distribution.

References

[1] I. Anvari, "Non-holonomic differential drive mobile robot control & design," Master's thesis, Arizona State University, 2013.
[2] "Cs w4733 notes - differential drive robots," 2018.
[3] C. Clark, "E160 lecture 3 - odometry," 2018.