DE2Bot Wall Following Design/Performance Summary

Matthew Liu

Georgia Institute of Technology

School of Electrical and Computer Engineering

Submitted

April 23, 2019

# Introduction

Within a 2-minute time constraint, our problem entails completing laps between the checkered areas while staying inside the green and avoiding walls (grey regions) and red/yellow boundaries (see figure 1). Our approach implements a modular wall following algorithm with performance improvement subroutines. Further explained in methodology, the codebase operates like a state machine. Visually, the DE2bot performs two 45 degree turns at the corner and a 180 degree turn at the end zones. The base code corrects differentially towards and away from the wall (proportional to left and right sonar measurements). With respect to this, our demonstration meets more than the specifications of our original proposal. During final testing, the robot was able to travel between checkered areas nearly autonomously with rare occurrences of penalties. This document will continue to examine our design process, our technical results in the field, and our final observations.

**Figure 1.** The Robot’s Course

# General Methodology

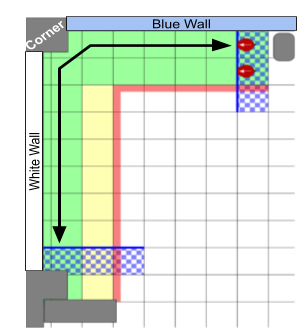
This section presents the process we followed to solve the problem.

## Management

An example of a subsection might be something like “Testing and Debugging” or “Project Progression.” Always keep your audience in mind when deciding what to include and how to describe it. Read the assignment sheet for tips.

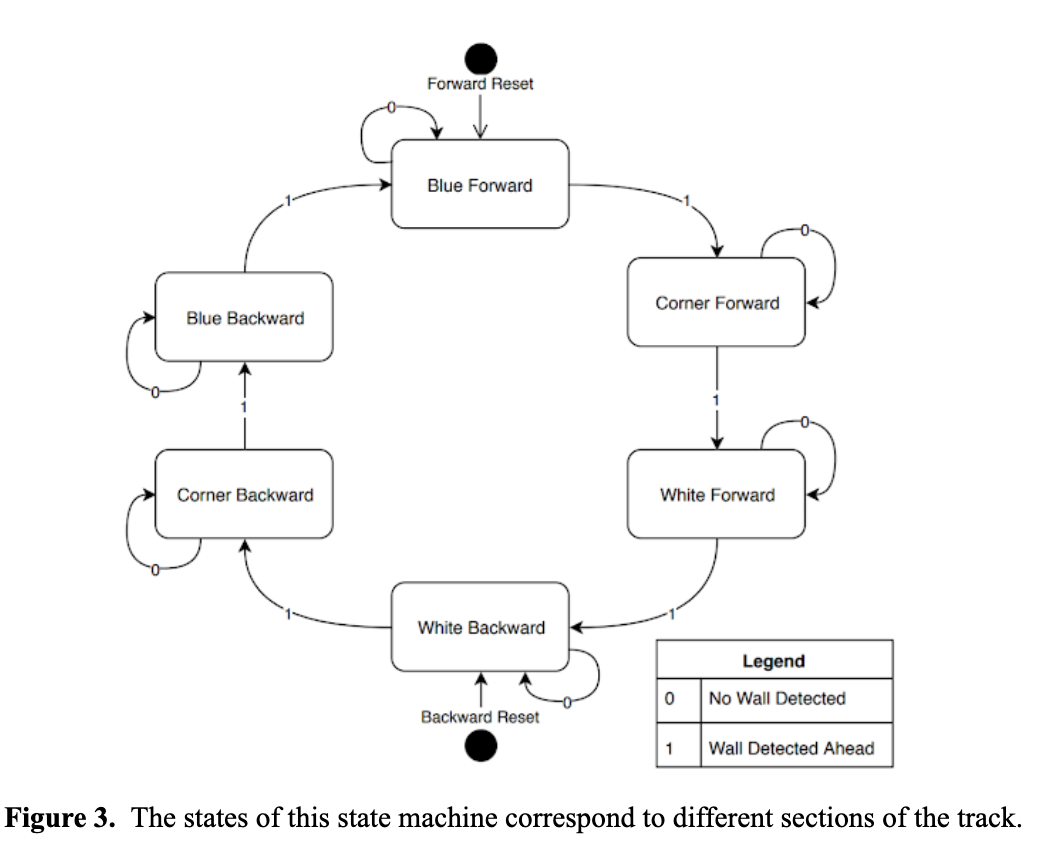
## Design

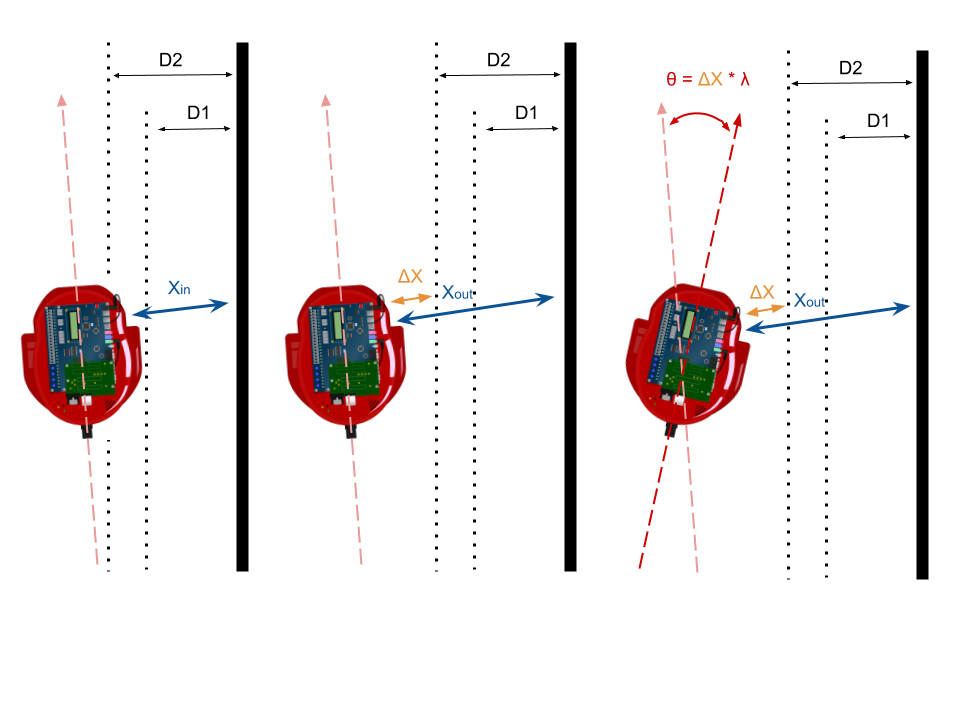
Similar to our proposal, the strategy we implemented models a state machine that keeps track of the robot’s position. This combines with a wall-following algorithm accounting for odometry drift and a collision detection routine.



**Figure 2.** The course followed by the robot.

A state machine keeps track of information for the robot to know its current location on the course. This allows the bot to make correct turns and activate appropriate sonars.



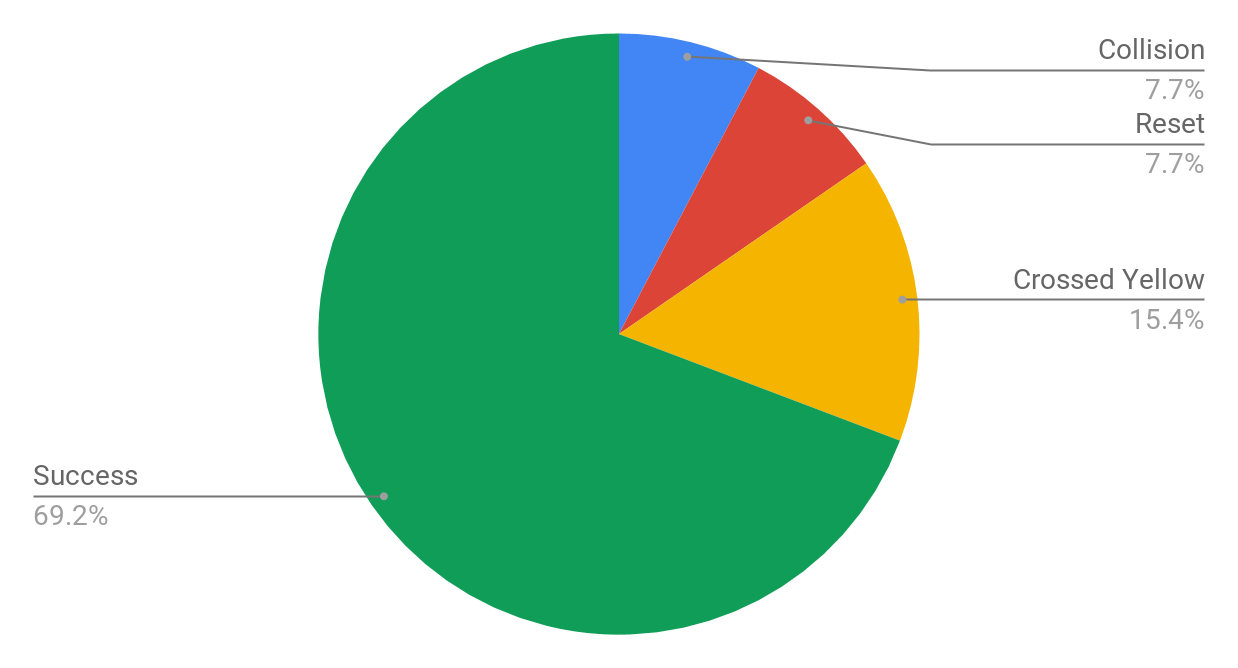
****Among several other wall-following algorithms, our initial and unsuccessful approach utilized hard-coded angles to adjust the robot with respect to the wall. The limitations of fixed values in a non-deterministic environment resulted in a shift of focus toward a more robust algorithm: differential correction. Consequently, the robot’s heading adjustments could be made in proportion to the difference between the target and measured wall distances.

**Figure 4.** Overview of differential correction algorithm

## Testing

Throughout the design process, we tested and optimized our code. And as a result, we reduced a “wobble” in the differential correction by implementing two thresholds to create a corridor for the robot to go straight. A majority of fine-tuning focused around constant values like these two thresholds.

# Technical Results

In the final testing phase, the DE2Bot consistently completed 6 legs per run while rarely crossing the yellow or colliding with the wall. However, during the demonstration, the robot only averaged 4.5 legs per run and frequently crossed yellow and even had to be reset.

**Figure 5.** Performance results from the 13 legs completed during the demonstration

# Conclusions

In retrospect, I am pleased with our robot’s capabilities during final testing; however, I am disappointed in the poor performance during demo day. We did fairly well in accomplishing the original purpose of the project, which we interpreted as designing a wall following algorithm to complete the course. For future engineers, an important recommendation is maintaining code modularity for ease of iteration between routines and codebase. Additionally, replacing fixed value correction with dynamic value correction algorithms often yielded better results due to the bot’s enhanced ability to respond to its changing environment. Based on our unexpected demo results, testing more robots in preparation also remains important. This would be one of our important considerations if we had more time - on top of optimizing values for higher robot speeds. An important weakness exists in the minimal testing of the robot at faster speeds. Additionally, in the design process, we probably could’ve benefited from more full trial runs. Overall, we did well, but should’ve done better during demo.