Running Rock Autonomous Robot
Project Summary

Mark Faingold

Georgia Institute of Technology
School of Electrical and Computer Engineering

Submitted
April 23, 2019

Introduction

This document examines the strategy used by team Running Rock for the ECE 2031 final project. The design problem was to implement a program on the DE2Bot platform to travel between 2 checkered areas within predefined boundaries shown in Figure 1 as many times as possible in two minutes.

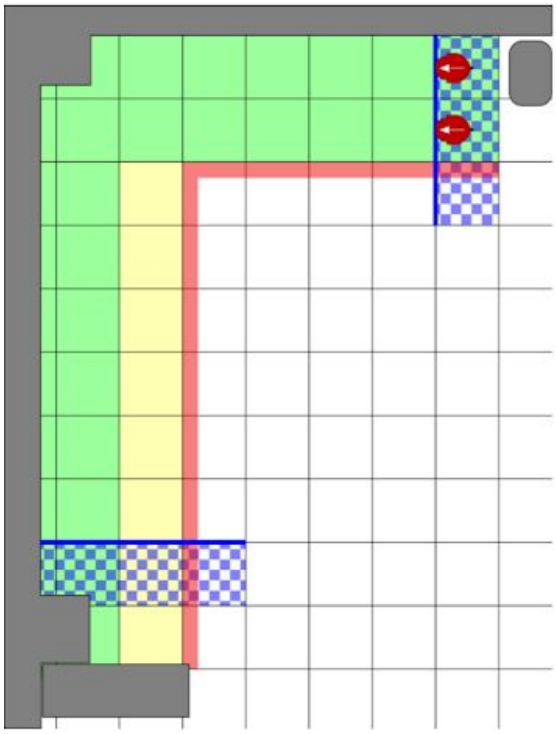

Figure 1. Layout of the robot demo area in the Digital Design Laboratory.

Our solution was to develop a wall-following algorithm using the movement API aided by sonars and a wall-collision recovery technique. The benefits of this approach include its feasibility within time constraints, robustness due to simplicity and ability to correct unsystematic odometry errors. Despite enjoying relative success in the final demonstration, our implementation had ample room for improvement and could benefit from additional testing.

General Methodology

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

Management

Because we prioritized the ability to iterate through designs quickly, we used GitHub to organize our codebase and strived to write modular code. As a result, various components could be completed independently and easily integrated.

Design

The strategy we proposed and implemented to solve the problem consisted of a state machine keeping track of the robot's position combined with a wall-following algorithm accounting for odometry drift and a collision detection routine.

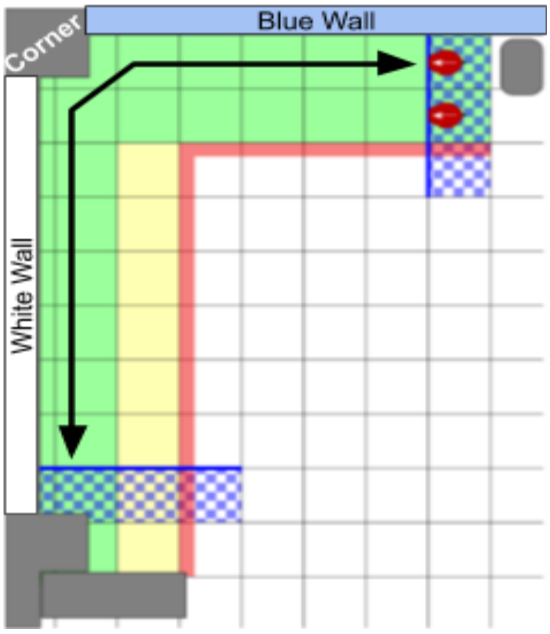

Figure 2. The course followed by the robot.

The robot had to know its current location on the course to make appropriate turns and activate correct sonars. We used a state machine to keep track of that information.

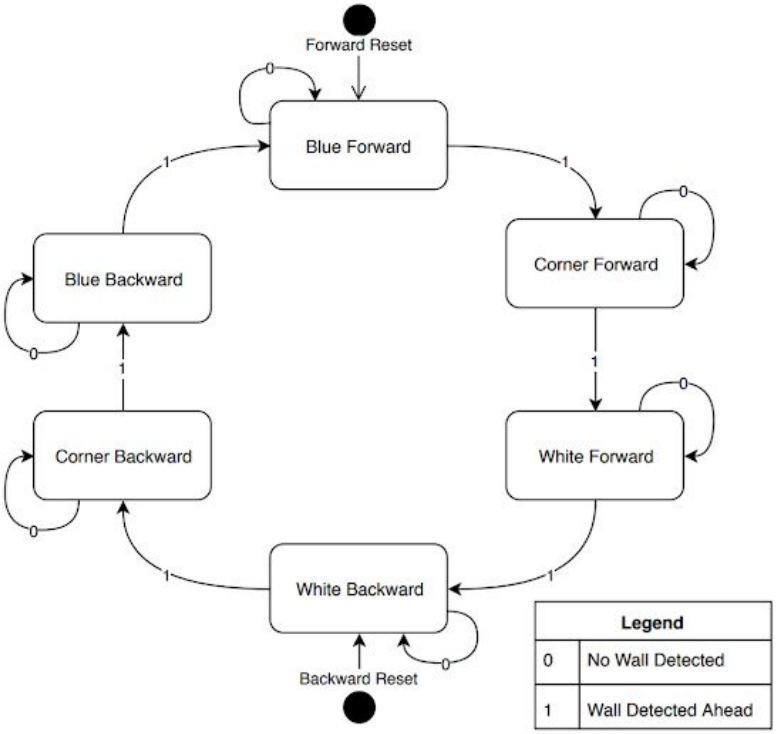

Figure 3. The states of this state machine correspond to different sections of the track.

We have iterated through several wall-following algorithms. Our initial naive approaches used hard-coded angles to adjust the robot's direction if it got too close or far from the wall. Due to the limitations of hard-coded values in a non-deterministic environment, we moved to a more robust differential correction algorithm which adjusted DE2Bot's heading proportionally to the difference between the target and measured wall distances.

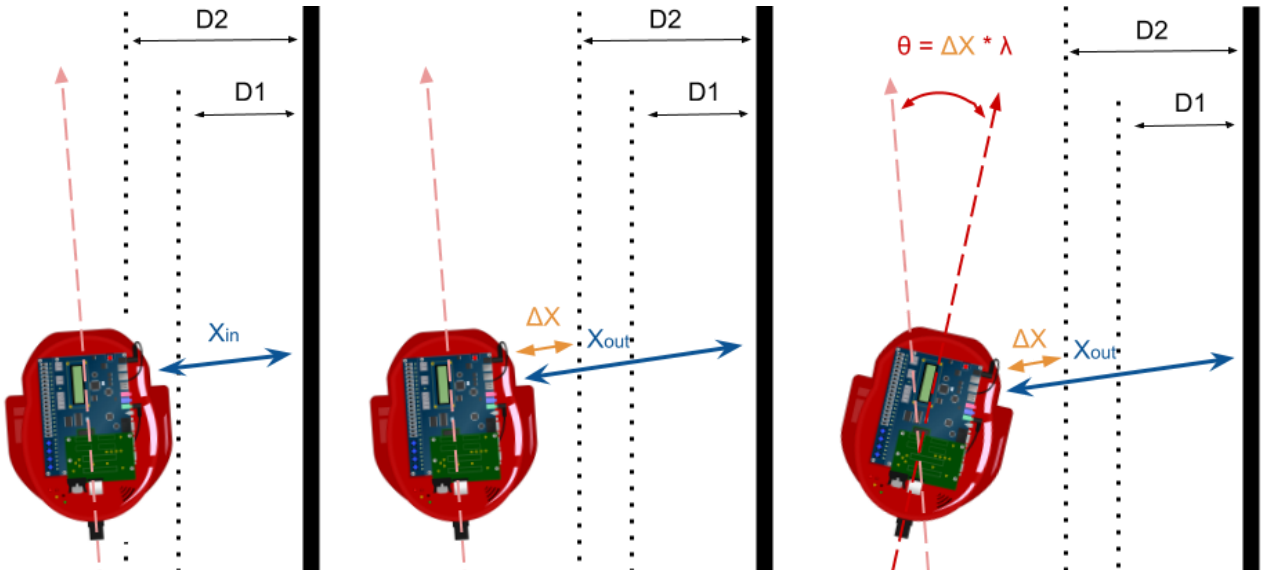

Figure 4. Overview of the differential correction algorithm.

Some robots experienced such inconsistencies in their odometry that our wall-following failed to correct the accumulated error which oftentimes resulted in wall collisions. Therefore, we introduced collision recovery—a way to reverse the DE2Bot and get it back on track after getting stuck.

Testing

Throughout the implementation process, we tested and optimized our code. We discovered that to reduce wobble in differential correction, two thresholds had to be used to create a corridor in which the robot continued straight. Most of the testing time was used to tune these and other constant values.

Technical Results

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

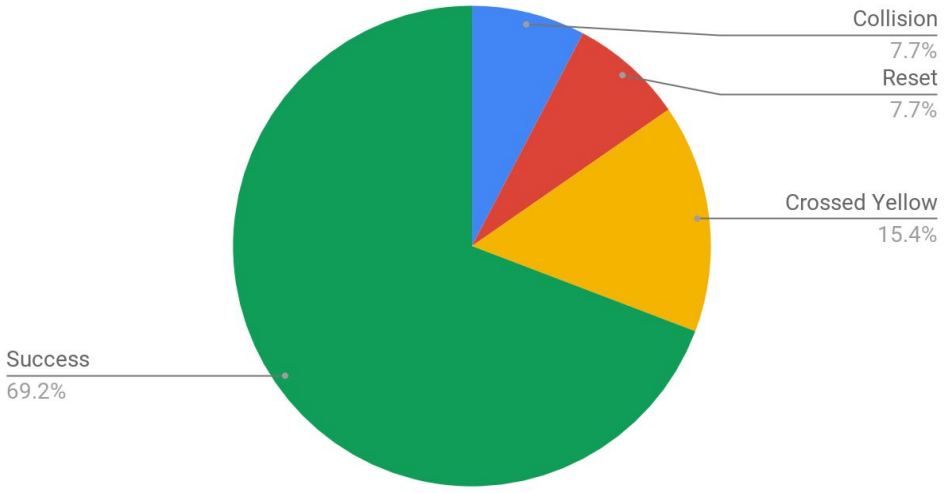

Figure 5. Performance results from the thirteen legs completed during demonstration.

Conclusions

Our design has successfully achieved the initial objective of the project since the DE2Bot was able to consistently rapidly move between the two checkered areas with little to no mistakes. Nonetheless, it had potential to perform better. During the demo, we encountered an unusually high number of errors which were most probably due to insufficient testing of the final changes to the codebase.

One of the most significant disadvantages of our implementation was the fact that we optimized it for medium velocity values because we preemptively dismissed high velocity as being unstable. In retrospect, we should have conducted more thorough testing before drawing any conclusions. Additionally, if we had to revisit this project in the future, we would incorporate a realignment routine to reset the odometry after turns. Finally, our design had numerous strengths such as robust wall-following and powerful position tracking.