Beaver Works Summer Program Technical Report

Michelle Tan

***Abstract*—This material is part of a summer boot camp at MIT Beaver Works where teams of high schoolers compete in the Mini Grand Prix Challenge, an autonomous race of cars around a racetrack. This program, which is based off of the 6.141 Robotics Science and Systems class at MIT and the Robotics IAP, meets on weekdays for eight hours for four weeks. Each day typically consists of three lab periods, a technical lecture, a seminar with guest speakers, and an occasional communications class.**

I. Introduction

THE purpose of this program is to begin to understand the algorithms and methods used today in autonomous navigation and experiment with them on the 1/10-scale racecar in order to showcase the abilities and limitations in building robust yet fast robotics systems. Some of the tasks given include using the Lidar with PID control to follow a wall, using a ZED Stereo camera to detect a blob, reactive obstacle detection, visual servoing to follow a colored blob, using a color to decide whether to take a turn, and optionally implementing localization and mapping. Much of the class dealt with how to deal with problems inherent to autonomous navigation, like sensor imperfections, inaccuracies in localization and mapping, and speed constraints. The cars used are referred to as RACECAR, which stands for Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot. The cars were pre-built with its components prior to the start of the program, which includes the NVIDIA quad-core CPU, the 192-core GPU, an Ackermann-steering system, and a large variety of sensors including a Lidar, camera, and odometer. The teams used ROS, a meta operating system for robots that focuses on modularity and interfaces.

II. Week 1

A. Technical Goal

The goal in the first week was to get experience in working with the racecar and learn about control systems by means of a wall following challenge.

B. Approach

The teams were required to use a Lidar to sense the wall but they were free to implement a state controller of their choice. [Go into details about state controllers]The simplest of the methods was using a Bang Bang controller, which would sense if the robot was too close or too far and if it was too close, it would turn away and if too far, turn towards. A slightly more sophisticated version was to implement a PID(Proportional Integral Derivative) controller which does calculations on the error value to create a smooth oscillation to correct for the robots distance. In addition, for calculating the error distance, there was an additional consideration of which laser scan data to use. Since angling of the robot could easily throw off the reading if only one measurement is taken, a two point controller could account for this with basic trigonometry.

(1)

Since there were so many options and there wasnt enough time to test all of them, we wrote code implementation of all of the separate features, like the PID controller and error calculation, so we could test each systematically. We also decided to approach the challenge by starting simple and incrementally improving it

C. Process

We started with the bang bang controller and as that worked, we slowly added a PID controller with 2-point control. For the bang bang controller, it was just a very simple algorithm that detected the amount of error in the cars distance from the wall. If the error was too big, it would turn towards the wall, and if the error was too small, it would turn away from the wall. The bang bang controller worked well to turn at a moderately steep curve.

As seen in the video, it worked well despite being a little jerky. The problem with the Bang Bang control was when we started the robot far away from the line. Since the turning rate was independent from the actual distance from the wall, when the robot was far away, it would turn back at a slower rate than it intuitively should given the distance away. Even worse was that when the robot was close to the line, it would continue to use too steep of a steering angle, resulting in inefficient oscillations and jerkiness.

We next tried implementing a P controller in order to control the oscillations more. In a P controller, the error is proportionally scaled to be the steering angle, so bigger error would result in more severe turning and on the other hand, smaller error would result in a more subtle turn. It took a while to figure out the Kp value. With higher Kp values, we found that we could adjust the rate of responsiveness. We found that Kp values near 1.0 achieved the best balance of reactiveness but not overreaction.

Afterwards, we tried moving to a PD controller. The Derivative portion would calculate the rate of change of the error and limit the change in order to attempt to smooth out the oscillations. We first found that changing the Kd value would lower the optimal Kp value. After a lot of thorough tuning of the Kp and Kd constants, we found that incorporating derivative caused jerkiness no matter what value we had it at. Although it was jerky we also found that it was necessary for adjusting when really far. When we tried the P controller and high distances, it would sometimes turn back so quickly that it would smash into the wall. With the PD controller, we found that even though the execution was shaky, it allowed the robot to not crash into the wall. Then we thought that there was no point of tuning the PD controller if we had to retune for the PID controller so we tried testing the PID controller. In the end, a large majority of the challenge had to do with finding the right constants. The setup that ended up working well was with a Kp of 1, a Kd of .05, and a Ki of 0. With more time, we could have continued perfecting the numbers, but they were sufficiently good for this challenge [research values for P, I, D]

D. Results

The challenge consisted of a straight left and right wall that the robot should be able to follow. Three time trials were recorded on the different sides starting the robot at different distances, then a challenge test was tried where the robot was put in the center of the 2-lane track, requiring agile recovery to avoid crashing into the wall. We learned that although there is a lot a theoretical research on optimal values for the PID controller, what happens in real life is vastly different. A lot of testing is required. According to Kyle from JPL, although the guess and check method was tedious, it was much easier than the alternative. By week one, we got a good idea of this problem in robotics, that what should happen or what happens in simulation is not a substitute for real life testing

III. Format

The report can be written in LaTeX or Microsoft Word, but LaTeX is definitely preferred. Its appearance should be as close to this document as possible to achieve consistency in the proceedings.

References should be cited as numbers, and should be ordered by their appearance (example: "... as shown in [1], ..."). Only references that are actually cited can be listed in the references section. The references' format should be evident from the examples in this text.

References should be of academic character and should be published and accessible. Your advisor can answer your questions regarding literature research. You must cite all used sources. Examples of good references include text books and scientific journals or conference proceedings. If possible, citing internet pages should be avoided. In particular, Wikipedia is *not* an appropriate reference in academic reports. Avoiding references in languages other than English is recommended.

Figures and tables should be labeled and numbered, such as in Table I and Fig. 1.

TABLE I
SIMULATION PARAMETERS

Information message length	$k = 16000$ bit
Radio segment size	$b = 160$ bit
Rate of component codes	$R_{cc} = 1/3$
Polynomial of component encoders	$[1, 33/37, 25/37]_8$

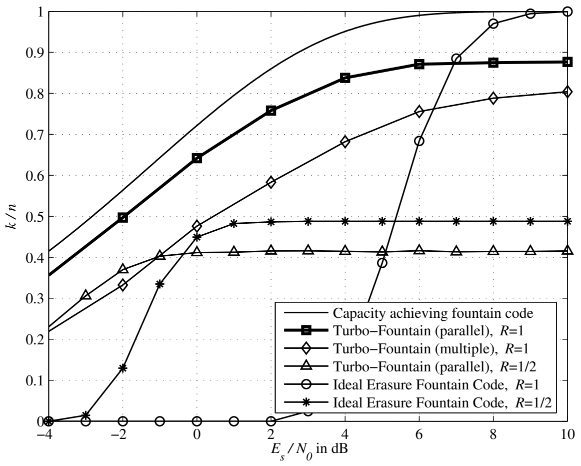

Fig. 1. Simulation results on the AWGN channel. Average throughput k/n vs E_s/N_0.

IV. Filling this page

Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. Gallos ab Aquitanis Garumna flumen, a Belgis Matrona et Sequana dividit. Horum omnium fortissimi sunt Belgae, propterea quod a cultu atque humanitate provinciae longissime absunt, minimeque ad eos mercatores saepe commeant atque ea quae ad effeminandos animos pertinent important, proximique sunt Germanis, qui trans Rhenum incolunt, quibuscum continenter bellum gerunt. Qua de causa Helvetii quoque reliquos Gallos virtute praecedunt, quod fere cotidianis proeliis cum Germanis contendunt, cum aut suis finibus eos prohibent aut ipsi in eorum finibus bellum gerunt. Eorum una, pars, quam Gallos obtinere dictum est, initium capit a flumine Rhodano, continetur Garumna flumine, Oceano, finibus Belgarum, attingit etiam ab Sequanis et Helvetiis flumen Rhenum, vergit ad septentriones. Belgae ab extremis Galliae finibus oriuntur, pertinent ad inferiorem partem fluminis Rheni, spectant in septentrionem et orientem solem.

V. Conclusion

This section summarizes the paper.

References

[1] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429-445, Mar. 1996.

[2] T. Mayer, H. Jenkac, and J. Hagenauer. Turbo base-station cooperation for intercell interference cancellation. *IEEE Int. Conf. Commun. (ICC)*, Istanbul, Turkey, pp. 356–361, June 2006.

[3] J. G. Proakis. *Digital Communications*. McGraw-Hill Book Co., New York, USA, 3rd edition, 1995.

[4] F. R. Kschischang. Giving a talk: Guidelines for the Preparation and Presentation of Technical Seminars. http://www.comm.toronto.edu/frank/guide/guide.pdf.

[5] IEEE Transactions LaTeXand Microsoft Word Style Files. http://www.ieee.org/web/publications/authors/transjnl/index.html