AVC Report

ENGR101

Alexander Heffernan (Team 26)
300653128

Victoria University of Wellington
ENGR101 Project 3 report

Abstract

This project aims to develop an AVC robot for navigating a four-quadrant obstacle course. This report covers the project background, robot creation, results, and further discussion on how it could be improved in the short and long term. By being multidisciplinary in nature, i.e., integrating electrical and software engineering, the project builds practical knowledge and skills while also addressing real-world-like challenges. A compact and efficient robot was developed, able to complete most of the obstacle course. The robot proved it's navigation, obstacle detection, and decision-making ability, as it breezed through the entire course, only needing slight human interference at the very end. However, even beyond fixing this minor error, there are further improvements that could have been made. The results of this project also lead to discussion on how ideas around user-interfaces and machine learning and AI could affect such a project. In conclusion, such a project requires robust hardware and sophisticated software algorithms to produce a reliable and efficient AVC robot which can complete the obstacle course, and while it does not do it perfectly, I believe our robot achieve this.

1 Introduction

1.1 Scope

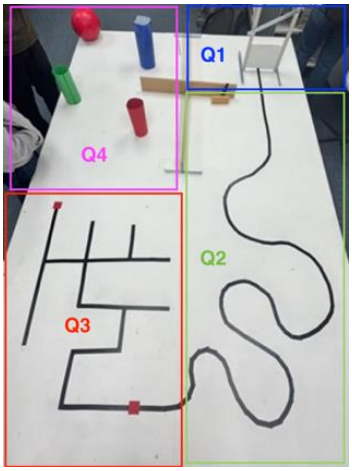
This report provides a comprehensive overview of our AVC project, including many key aspects. Firstly, background information will be provided, including the hardware provided to us and the research we conducted prior to commencing the hardware construction and software development phases. Secondly, the method behind the design and build of our hardware and software will be outlined and explained, and how steps were taken to ensure the AVC robot was able to complete most of the course. Thirdly, the results obtained via testing and marking on the prepared obstacle course will be presented, and the AVC robot's performance and effectiveness will be evaluated. Finally, discussion will explore potential areas of improvement for our project, both short and long-term. By covering these key aspects, this report offers a comprehensive analysis of our project.

1.2 Motivation

The purpose behind completing this project was to understand how AVC vehicles work, and the systems behind them. From this hands-on experience, further skills were gained in building, programming, and integrating autonomous systems. Working in random teams and with genuine hardware also provides some real-world experience, preparing us for the workforce. Such a challenge, which will be explained below, also improves our problem-solving skills, making this project very beneficial to many engineers.

1.3 Aim

The aim of this project was to create an AVC robot able to complete a complex obstacle course. An image of this course can be seen to the right. The image also shows how the course was split into four different quadrants (Q1 in blue, Q2 in green, Q3 in red, and Q4 in purple). To complete this course, the hardware and software must work together efficiently and reliably, as it makes its way through each quadrant which presents its own challenges.



1.4 Objective

The AVC robot must be able to complete each quadrant:

- **Quadrant One:** Connect to and open gate, before passing through in time for it to close behind the robot.
- **Quadrant Two:** Follow a wiggly line until red is detected on the pathway, signalling the start of the next quadrant.
- **Quadrant Three:** Reach the end of a maze, which contains sharp corners with many different routes to take.
- **Quadrant Four:** Approach three uniquely coloured cylinders, before pushing the red ball off the obstacle course.

To complete these four quadrants, a sturdy, efficient, and reliable autonomous robot is required. With the assistance of provided hardware and C++ libraries, and while using the layout and knowledge of the course, we must construct a robot, as well as write a program for it, allowing it to reach the end of the course. Four weeks were provided before project had to be complete and marked.

2 Background

2.1 Hardware

The following hardware was provided to our team at the beginning of the project to assemble our AVC robot:

- **Raspberry P (RPI):** The AVC will run on an RPI.
- **Servo Motors:** Receiving commands from the RPI, the continuous servo motors can be set to turn at different speeds and in different directions, and the non-continuous servo motors can be set to turn to a certain position and hold it.
- **Camera:** Used to send image input. This image input can be processed via a program running on the connected RPI.
- **Battery:** Used to power the RPI, and therefore power the entire AVC. Battery is connected to RPI via a USB to Micro-USB cable.

On top of this hardware, we were provided with various scrap materials, as well as access to a 3D printer to create custom parts, to assemble all these pieces of hardware together.

2.2 Software

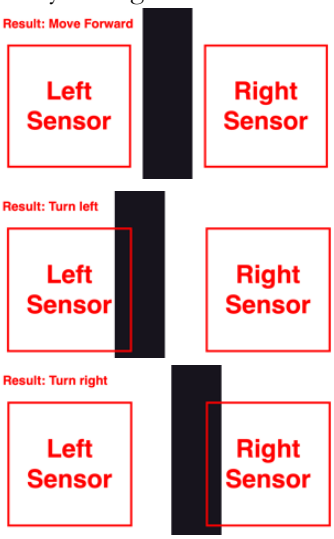
We were required to write the software for the AVC robot in C++. This program would run on the Raspberry Pi attached to the robot, allowing the program to access all the different pieces of hardware, including the servo motors and camera, to create a product that can makes its way through the course.

2.2.1 Following wiggly line.

For Quadrant 2 of the obstacle course, the AVC robot must follow a wiggly line. From my research, a report written by Siddhant Pathak [1] explains to use "two IR sensors [not available for this project but similar solutions are available, such as the camera provided] modules namely left sensor and right sensor." With these sensors, the AVC robot can:

- **Move forward:** When left and right sensor does not detect black.
- **Turn left:** When the left sensor detects black, and the right sensor does not.
- **Turn right:** When the right sensor detects black, and left sensor does not.

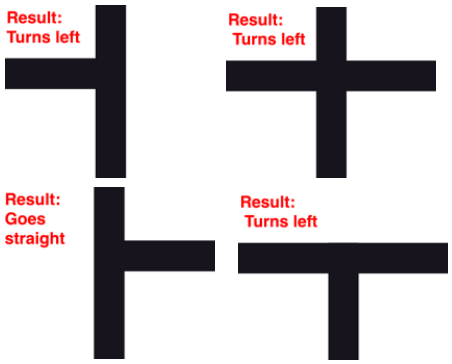
This is explained in the diagrams to the right, showing the two sensors, the black line, and the result.



Similar sensors could be used to complete Quadrant 4 as well, to drive in a straight line to approach each coloured cylinder. Rather than using IR sensors, the image processing from the provided camera can be split up to process the left and right separately, allowing for similar results.

Similarly, an article published online written by someone by the username Bot Reboot [2] explains how an error variable can be used to determine the direction to move a robot following a line. This error variable is set to the centre position of the camera (where you want the line to be) minus the position of the detected line. If the error is positive, it means the robot is to the right of the line, and if the error is negative, the robot is to the left of the line. Combining these two bits of research became the basis for completing Quadrant 2 and 4.

2.2.2 Reaching the end of a maze.



For Quadrant 3 of the obstacle course, the AVC robot must find its way out of a line maze with sharp corners. In an article written by Md Mobshshir Nayeem [3], he explains the "LSRB algorithm." The algorithm simply states:

- **Turn left:** if left is an option, take it.
- **Go straight:** if left is not an option, and straight is, go straight.
- **Turn right:** if left and straight is not an option, and right is, turn right.
- **U-turn:** if left, straight, and right is not an option, complete a U-turn and go back.

The above algorithm is guaranteed, even if the most efficient pathway is not taken, to reach the end of a line maze. All cases where two or more choices are presented, and the results of these cases, are shown above and to the right.

3 Method

Responsibilities were divvied up, with Michael Visser leading the hardware side of the project, and Alexander Heffernan and Cara Lill leading the software side. All files relating to this method are available on the Team 26 AVC Project GitLab page [4].

3.1 Hardware

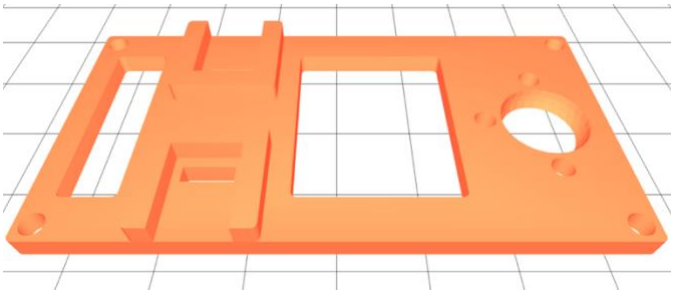
The AVC robot was designed to get through each quadrant of the obstacle course. From the start, it was important that the robot was as compact as possible, to allow for easy passing through the gate in Quadrant One. For consistency between quadrants, it was important that the robot's weight was well distributed and balance. Finally, the camera needs to be able to look up and down, allowing for the robot to see down for Quadrants One, Two, and Three, and look up for Quadrant Four.

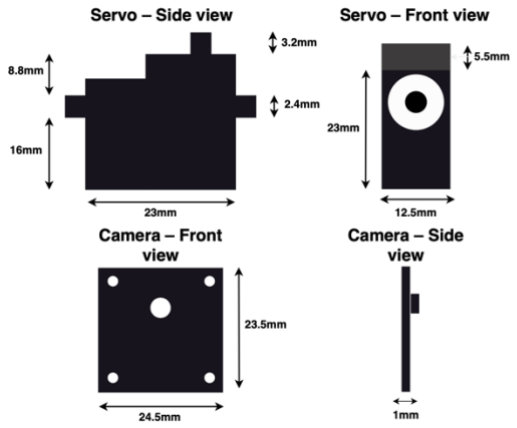
3.1.1 Components Used

- Raspberry Pi 0W
- Custom printed circuit board (PCB)
- Raspberry Pi Camera
- Battery
- USB to Micro-USB cable
- 2x Continuous Micro Servo Motors (FS90R)
- 1x Micro Servo Motor (MG90S)
- 2x small wheels
- 1x acrylic plate
- 1x 3D printed baseplate
- 4x 3D printed poles
- 1x marble
- 1x 3D printed camera mount
- 1x 3D printed camera servo holder
- 4x 40mm M3 bolts and nuts
- 4x M3 screws
- 2x 10mm M2 bolts and nuts
- 2x M2 screws

3.1.2 3D Printing

Early in development, diagrams (which can be seen below) of parts were created to ensure exact measurements were confirmed before the modelling phase, allowing for more precision. Most additional structural components were designed in FreeCad and created with a 3D printer. While there were early errors with the 3D printer, simple recalibration fixed most issues. A model of the 3D printed baseplate can be seen below:



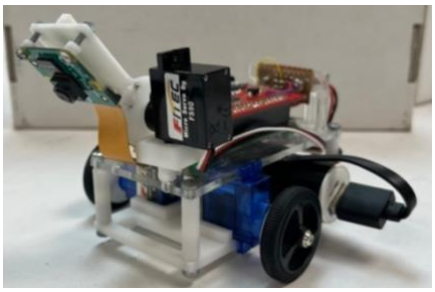


3.1.2 Assembly

The AVC robot is made up of two main plates. The plates are connected with the four 3D printed poles, as the 3D printed baseplate serves as a raising platform for the acrylic plate. This design meant we didn't need extra mounting hardware and allowed the motors to be securely placed between the plates. Each plate held different parts of the final AVC robot's hardware:

- **Acrylic plate:** Holds the RPI, custom PCB, camera servo, and the camera.
- **3D printed baseplate:** Holds the marble, wheel servos, wheels, and the battery.

The marble is stored in a cut-out at the back of the 3D printed baseplate, allowing the back of the robot to slide along the surface when moving. It is further secured with an additional plate to prevent it from falling out. The battery was then placed between the motors and marble, wedging between the two plates very perfectly when paired with the USB to Micro-USB cable (used to connect battery to Raspberry Pi). This battery placement allowed for good balance, and easy setup and removal.



Finally, the camera servo is mounted on a raised 3D printed mount at the front of the AVC robot. This gave is a clear view of the line below, as well as everything in front of it when rotated to look up. Camera and servo are attached together with a 3D printed mount and screws through the back of the camera.

3.2 Software

The code was split into four sections, a section for each quadrant. The main section initiates the camera stream, toggles the camera to face downwards, before executing the beginning of Quadrant One. While most of each quadrant's code is held in its function, Quadrant Three includes the use of many additional functions, to allow faster development and testing. Unfortunately, the same is not done for Quadrants One, Two, and Four. The program does not finish execution until all four quadrants are finished.

3.2.1 Quadrant One

To open the gate, code communicates with it over the network. This communication is explained below:

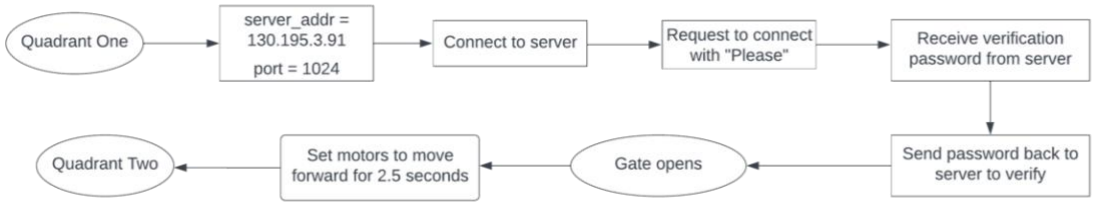


Figure 1: Opening gate

3.2.2 Quadrant Two

The line follow uses an on-off loop, turning right when the error is too high, and turning left with the error is too low. If the error is close enough to zero, the AVC robot will drive in a straight line. Error is calculated along the middle row of the outputted image from the camera, adding the pixel number of each black pixel to the error (with pixel number 0 being in the centre of the camera). If error is exactly 0, or in other words, there is not black line, the AVC robot will reverse. This process is explained below:

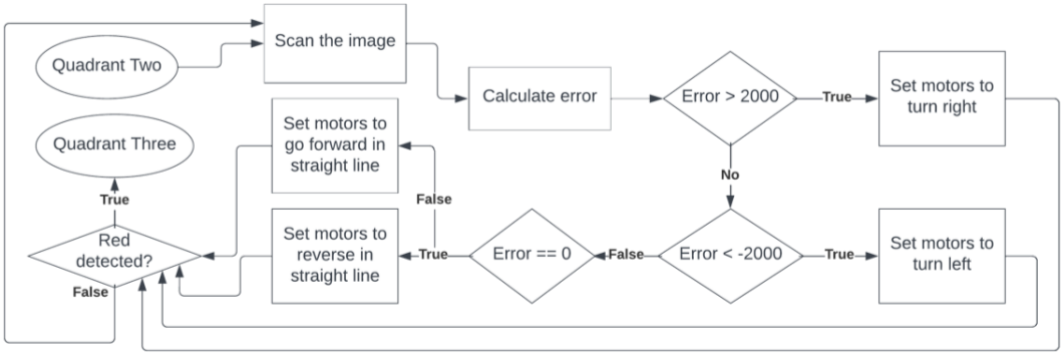


Figure 2: Following wiggly line.

3.2.3 Quadrant Three

The AVC Robot makes its way through this quadrant following the LSRB algorithm. By scanning the image produced by the camera, top, left, and right errors and paths are calculated, allowing for the algorithm to decide the next move for the robot. It does this following the below process:

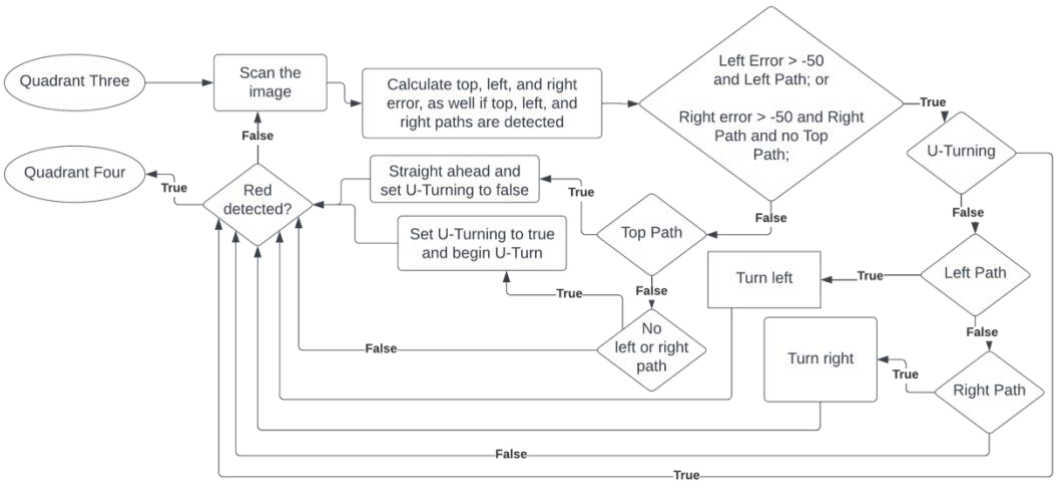


Figure 3: Reaching end of maze.

3.2.4 Quadrant 4

The AVC robot makes its way through this quadrant, using a similar on-off system as described in Figure 2, except instead of looking for the colour black, it's looking for the colours red, green, and blue. To approach a coloured cylinder, it will turn until it is visible. Once visible, it will approach the cylinder until it is close to the robot's camera (i.e., coloured pixel count extremely high). At this point, it will reverse away from the cylinder and prepare for the next stage. The pushing ball process is very similar, except once the ball is close to the camera, it will continue to move forward until the ball is gone, immediately stopping the cameras. These processes are explained below:

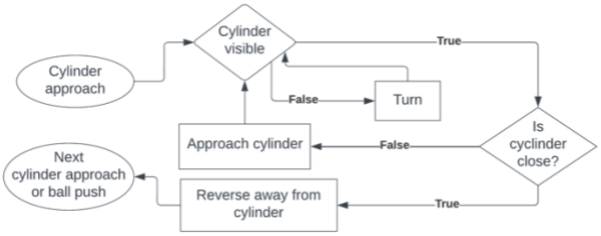


Figure 4: Approach cylinders

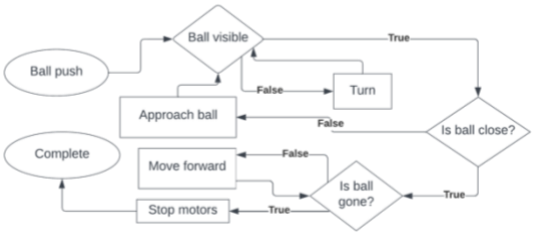


Figure 5: Push ball

4 Results

4.1 Hardware

The result for the AVC's hardware was compact, lightweight, and effective. However, the hardware was not perfect. Below outlines the strengths and weaknesses of the final hardware:

Strengths:

- **Compact:** Especially important in terms of getting through the gate in Quadrant One.
- **Lightweight:** Due to this factor, the robot it manoeuvrable, which was critical during Quadrants Two and Three, especially with the tight turns.
- **Tightly wound wiring:** Wiring connecting servo motors to Raspberry Pi were tightly wound through the chassis, preventing anything from getting caught or causing drag on the surface.
- **Precise design:** Due to most of the parts being custom made with modelling and 3D printing, everything fitted together perfectly in assembly, meaning no later modifications were required.

Weaknesses:

- **Low hanging camera:** Made line detection sensitive, as the line covered large portion of image output. Caused incorrect turns during Quadrant Three testing.
- **Inaccurate servo motors:** Provided motors needed constant adjustments for consistent movement. Left unattended for too long and the robot was unable to drive in a straight line or in reverse.
- **Tight marble holder:** Marble was held in place, but unable to rotate at the back of the vehicle. Marble's presence however was still enough for the back of the robot to glide along the surface with the wheels.
- **Low chassis:** If terrain was rough, robot would quickly get stuck during testing. Robot is however perfectly effective when surface is flat.

While the AVC robot is not perfect, it's weaknesses did not get in the way of completing the obstacle course when being marked. Simply means the AVC robot is not very flexible if put to the test in other conditions.

4.2 Software

The result for the AVC's software written in C++ was very successful. The program leads the AVC robot right through the obstacle course, only requiring slight interference during final marking to reach the finish line.

4.2.1 Quadrant One

Initial testing proved that opening the gate and instantly initiating Quadrant Two, was not effective, as the Quadrant Two code was not fast enough to get through the gate in time. To fix this, during Quadrant One, the robot drives in a straight line at a faster speed for two and a half seconds, following the opening of the gate, before initiating Quadrant Two. This proved successful, assuming the robot was placed in the same direction as the path it had to follow. Placed in a slightly incorrect position, and the robot would drive fast in that direction, losing the line before Quadrant Two code could initiate.

4.2.2 Quadrant Two

While very reliable, the AVC robot is often slow at getting through Quadrant Two. Because the robot turns left and right at a constant speed, and as it is not controlled by how far the robot needs to turn to reach the middle point of the line again, the robot is never aligned closely enough to the centre of the path for it to determine it safe to move forward in a straight line. The robot is always realigning itself as it follows the path to the end of the quadrant. This constant realigning is reliable, but not efficient. However, even with this slow down period, the robot makes it through the obstacle course with plenty of time to spare.

4.2.3 Quadrant Three

After thorough testing, the AVC robot moves through Quadrant Three very efficiently and reliably. Due to the low hanging camera, it took great refinement and precision of the software to reach this point. Before, it struggled to distinguish between left, right, and forward pathways, causing errors in its decision making. However, by the end of testing, it was able to consistently distinguish between available pathways and make the correct decision as to which way to go, following the "LSRB algorithm" [3].

4.2.4 Quadrant Four

The result of Quadrant Four was not perfect, and in final marking required a little bit of interference from the marker to reach the end of the course. This was due to it failing to distinguish between the red cylinder which it was meant to target, and the red ball at the end of the course. However, the marker was able to move the red ball to the side, and the AVC robot successfully finished the quadrant. Due to the amount of time spent testing Quadrant Three, we unfortunately did not have time to remove this fault, but I believe with more time it would have been a relatively straight forward fix (refer to 5.2.3 for more information).

5 Discussion

5.1 Hardware

While the hardware for the AVC robot was very effective, below are some improvements that could have been made based on previously discussed weaknesses:

- **Low hanging camera:** Adding height to the camera mount would have removed issues surrounding sensitive line detection. Using larger wheels would have also helped add height.
- **Low chassis:** Using larger wheels would have helped raise the chassis off the ground further, lessening chances of the AVC robot catching on something.
- **Tight marble holder:** Sanding the inner sides of the holder may have been beneficial here, allowing more room for marble to spin. We could have also investigated smoother materials to make the holder out of to remove some friction between the marble and the holder.

5.2 Software

While the software for the AVC robot was able to complete most of the obstacle course without human interference, there are some improvements that could have been made to make it more efficiency and reliable. In 5.2.3, solutions to faults in Quadrant Four are described.

5.2.1 PID Controller

During my research, and in lectures, the recommendation was made to use a PID Controller for following the line with the AVC robot. While the current software, which is without such a system, does work and complete the course, it is much more inefficient than I believe using a PID controller would have been.

PID controller, or a proportional integral differential controller, is a process that prevents overshooting when trying to adjust motors to reach an error of 0. Current software constantly overshoots, causing many oscillations to the error. Using a PID controller would allow the AVC robot to "take different types of curves at different curve radius without getting off the line" [2]. It does this by adjusting based on proportional distance between AVC robot and the line, and the rate of change of the AVC robot's error. Three of the four quadrants would have benefited uniquely to such a controller:

- **Quadrant One:** Would have been able to be placed at different angles relative to the start point of the line, with it then being able to line itself up with the pathway before opening the gate and driving straight through.
- **Quadrant Two:** Rather than constantly, and drastically, adjusting to ensure it was following the line, these adjustments would have been more precise, allowing for more of a focus on moving along the line. Also, would have been more reliable, as overshoots often caused issues with the robot losing the line and needing to go into reverse before finding the line again. Overall increase to efficiency.
- **Quadrant Four:** Like Quadrant Two, a PID Controller would have made the AVC robot much more efficient. Speed would greatly increase, as it spends less time making drastic adjustments to reach an error of zero, and overshoots wouldn't cause the robot to lose sight of each coloured cylinder, causing it to spin around in a full 360° before finding the coloured cylinder again.

Note, such a PID controller likely would not have been necessary in Quadrant Three and would not have been implemented. This is due to the lines being perfectly straight, rather than curvy like in Quadrant 2. Simple alignment rotations were implemented in Quadrant Two to ensure it never went off track.

5.2.2 Motor speeds

To improve speed of the AVC robot, more time testing with different motors speeds would have been beneficial. Early on in development, we found speeds that worked, and simply stuck with them. Alternatively, we should have spent time testing out multiple different motors speeds to find the fastest speed we could use to get through the obstacle course, without sacrificing reliability and accuracy.

5.2.3 Quadrant Four Improvements

As Quadrant Four was where our team lost 2% from our mark, I believe there are some solutions that could have been tested to prevent the need for the marker to interfere. For instance, adding a short turn to the right for the AVC robot at the start of Quadrant Four would have prevented the robot from detecting the red ball at the end of the course before the red cylinder which it is supposed to target first. Another idea would be to work on improvements to decision making. Maybe the AVC robot could have been able to detect that the red ball was too far away, via the red pixel count when looking in its general detection. This would allow the AVC robot to ignore the red ball at the end of the course, and instead continue rotating until finding its true target, the red cylinder, which would have a higher red pixel count due to its shorter distance away.

6 Conclusion

Overall, the AVC robot was very successful. Though neither the hardware nor the software was without flaws, together they came together to complete the entire obstacle course with only a little interference from the marker. This interference resulted in 2% being taken off our final mark. To have gain full marks, we should have made improvements to the software, especially around Quadrant Four. I believe implementing better object detection algorithms, or improving the simple one in place, to be able to distinguish between the far distanced red ball and the red cylinder, would have allowed for the AVC robot to finish the course, and receive full marks.

7 Future work

Beyond the successful creation and development of the AVC robot are many different pieces of research and exploration that can be done:

- **Machine learning and AI:** This would enhance AVC robot's adaptability and learning capabilities. Looking into such techniques, as well as deep learning, can lead to improvements in object recognition, line detection, decision-making, etc. This would allow the AVC robot to continuously learn and improve its performance over time.
- **User-interface:** Would make product more user-friendly, providing controls and ways to monitor the AVC robot's status, beyond the very un-user-friendly command terminal. Could provide user with a start button, view of the camera output, current progress through the obstacle course, as well as an emergency stop function.

References

[1] S. Pathak, "Line Following Robot," Dept. of Electrical Engineering, Madhav Institute of technology and science, Gwalior, India, DOI 10.17148/IJIREEICE.2021.9542

[2] B. Reboot, "Line Follower Robot (with PID controller)," hackster.io, Aug. 11, 2020. [Online]. Available: https://www.hackster.io/anova9347/line-follower-robot-with-pid-controller-cdedbd/. [Accessed Jun. 10, 2023].

[3] M. Nayeem, "Coding a Line Follower Robot for Maze using LSRB Algorithm and finding it's Shortest Path," towardinfinity.medium.com, Jul. 7, 2019. [Online]. Available: https://towardinfinity.medium.com/coding-a-line-follower-robot-using-lsrb-and-finding-the-shortest-path-d906ffec71d. [Accessed Jun. 10, 2023].

[4] M. Visser, A. Heffernan, and C. Lill (Team 26), "AVC Project T26", gitlab.ecs.vuw.ac.nz. [Online]. Available: https://gitlab.ecs.vuw.ac.nz/course-work/engr101/2023/project3/t26/avc-project-t26.git