



Investigation Into the Application of Neuro-Evolution to Simulated Autonomous Vehicles

6001 CEM Individual Project

By Daniel Rodrigues
9402728

Coventry University
Faculty of Computing, Engineering and Mathematics
Supervisor: Dr David Croft

Table of Contents

Contents

6001CEM Declaration of originality	4
Statement of copyright	4
Statement of ethical engagement	4
Abstract	5
Acknowledgements	5
Keywords	5
GitHub Repository	5
1 Introduction	6
Project Aim	6
Project Objectives	6
Background	6
Motivation	6
Structure	6
2 Literature Review	7
Introduction	7
Simulation	8
Polar vs Cartesian Co-ordinates	8
Vectors	8
Physics	9
Webots	9
ROS	9
Examples	10
Path Following	11
Steering Behaviour's	11
Vector Math	12
Race line	13
Neural Networks	14
Deep learning	15
Genetic Algorithms	16
Review	18
3 Research Methodology	20
Experiment Constructs	20
Independent Variables	20

Dependent Variables.....	20
Constraints	20
Simulator	21
Implementation.....	22
Hardware	22
Version control.....	22
Path Following.....	23
Autonomous Agent	24
Steering Behaviours	25
Genetic Algorithm	27
Fitness	27
Mating Pool.....	28
Heredity	28
Mutation	29
Generations.....	29
4 Results.....	30
5 Results Analysis	37
Outputs	37
Comparison	37
Implementation issues.....	38
6 Project Management.....	39
Development Methodologies	39
Social and Legal Considerations.....	43
Ethical and Professional Considerations	43
Risk Management.....	44
Meeting Records	44
Supervisor Feedback	46
Project Management Reflection	46
7 Conclusion	47
Summary.....	47
Real-World Application.....	47
Project Limitations	47
Future Research.....	48
Reflection.....	48
8 References	49
Bibliography	49

9 Appendices	51
Project Proposal (6000 CEM)	51
Research Question	51
Project Topic.....	51
Summary	52
Motivation and Outputs	52
Primary Research Plan	53
Implementation	53
Time management.....	53
Constraints	53
Literature Review	54
Paper 1	54
Paper 2	54
Paper 3	55
Paper 4	55
Conclusion.....	55
Bibliography	56
List of figures	57
List of tables.....	58

6001CEM Declaration of originality

(This form should be completed by the student and included in the project report)


I Declare that This project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Statement of copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialize products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information, please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

Statement of ethical engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (<https://ethics.coventry.ac.uk/>) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking).

Signed: 

Electronic signature is acceptable

Date: 25th April 2022

Please complete all fields.

First Name:	Daniel
Last Name:	Rodrigues
Student ID number	9402728
Ethics Application Number	P130750
Supervisor	David Croft

This form must be completed, copied, or scanned and included with your project submission to Turnitin. Failure to append these declarations may result in your project being rejected for marking.

“Can race lines be optimized in a simulated environment?”

Abstract

This paper investigates whether Neuro Evolution can be used on simulated racetracks to generate the most optimised race line. A simulated recreation of Darwin’s Theory of Evolution by Natural Selection with Autonomous Vehicles to see which agent performs the best results. Genetic Algorithms and Neural Networks which adopt similar traits to this theory are great concepts that could be applied to this experiment. This paper analyses the results of how various vehicle traits can affect a race line.

Acknowledgements

I would like to thank the contributions of my supervisor David Croft who has continuously provided advice and support throughout the development of this project.

Keywords

Autonomous Vehicles, Evolutionary programming, Race line, Optimization, Simulation, Genetic Algorithms, Neural Networks and Neuro-Evolution.

GitHub Repository

<https://github.coventry.ac.uk/rodrig75/Dissertation>

1 Introduction

Project Aim

Aims to investigate if Neuro Evolution can generate an optimal race line to improve the lap time for a simulated autonomous vehicle?

Project Objectives

- Create an autonomous vehicle
- Input and train them using Neural Networks
- Reproduce them with Genetic Algorithms
- Evaluate the most optimized Neural Networks path
- Suggest further improvements

Background

Formula Student Autonomous (FS AI) is a university challenge where teams must construct an autonomous racing car that must navigate around multiple Silverstone tracks without any manual input. The winners of this competition are the team whose car manages to get around the track consistently and with the fastest lap time. Multiple factors play into this from speed to traction but most importantly the chosen race line. The racing line is the shortest possible route around a track. According to (Cardamone, 2010) in racing games, the trajectories are pre-calculated manually by experts, this is extremely time-consuming and not allowed in the competition (Rodrigues, 2021).

Motivation

Like Formula 1, teams in FS AI don't get much testing on the official track until the event occurs (Mattocks, 2019). This also proves to be a problem for logical errors from a development standpoint. Trial and error are fundamental in understanding the system's current state and what issues are present. Without access to performance data to see what sort of consistency or lap times the vehicle will provide; brainstorming new features is entirely theoretical. Therefore, Genetic Algorithms and Simulations are a useful solution for the FS AI team (Rodrigues, 2021).

Structure

My project covers six chapters that cover, research, implementation, results, project management, reviews and evaluation. My literature review covers my research surrounding the concepts of Autonomous Agents, Simulation, Path Following, Neural Networks and Genetic Algorithms. These concepts will lead to my implementation of a system that can generate an optimised race line. This will lead to the results and analysis section where I will evaluate and form discussions surrounding the results my system has generated under different conditions. I will then explain what tools and methodologies I used surrounding project management. These tools include risk management and explanations surrounding the ethical, social, professional and legal considerations. A follow up of material relevant to my supervisor including a record of our meetings and the specific feedback he gave me and how I implemented said feedback into my project. I will then conclude this paper with a summary and reflection on the general overview of this project.

2 Literature Review

Introduction

An autonomous vehicle is an entity that uses its intelligence to form decisions based on its perception of its environment. Three components make up an autonomous vehicle. Its perception, how the information is processed, and its external forces.

Reynolds (1999) describes autonomous agents as following a three-layer procedure to determine their behaviour. **Action Selection**, **Steering**, and **Locomotion**.

Action Selection is the targets of the agent and how it plans to accomplish the said target. As I said before the autonomous agent will investigate the variables of its environment and then choose a feasible target.

Steering is the calculation of how it plans to accomplish the chosen action. If the plan is to seek the target, the calculation would be how to get from point A to B. If the plan is to flee then the calculation would be how to maintain or extend a relative distance away from point A.

Locomotion is the act of carrying out the steering calculation. This starts to go into kinematics which I will expand upon in the simulation and path following chapters, but the overview of this step is to take the required path and apply a force to have the agent follow said path until it meets the target.

Simulation

Simulations are used to represent a real-world environment in a virtual setting, acting as a sandbox for users to test their creations. Simulations create rigid bodies for agents, which allow them to interact with platforms, boundaries, and other agents in a realistic manner. It allows for scenarios to be tested to extract data on what would happen when played out. Simulations usually contain physics engines, that allow users to apply forces to an object to move them using actual locomotion. This allows for testing, analysis, and feature development. It is a commonly used technique in sports such as F1 according to mercedesamgf1 (2020).

Polar vs Cartesian Co-ordinates

Figure 1 – Polar Co-ordinates (Lumen, 2022)

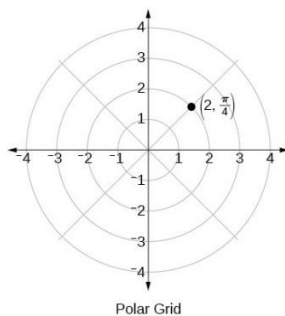
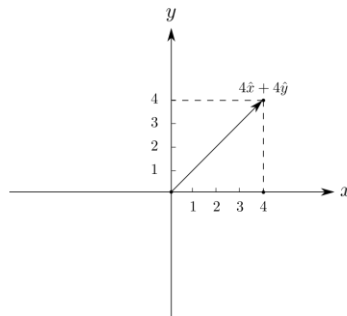


Figure 2 – Cartesian Co-ordinates (Zinka, 2022)



Every simulation has a global space with a coordinate system used for the location and navigation of an entity. Two co-ordinate systems are traditionally used in physics engines, Polar and Cartesian.

Polar uses a pole that acts as the origin. Using a radius, a circle can be drawn around the origin as the distance. Combining this with an angle, a point can be plotted onto the map. Since the distance is measured as a circle around the origin, the angle is measured as a portion of its total circumference also known as pie.

Cartesian uses an x, y, z, axis system with segments on each axis intersecting, forming a 3d grid system. This system can either use a single or all 4 quadrants using negative x y z.

Webots uses the cartesian system using all 4 quadrants with the centre of the world or (0,0,0) acting as the origin according to cyberbotics (2022). However, the heading is calculated using polar coordinates as a measurement of pie, therefore I believe a combination of the two will be required in my implementation to represent the current direction of the vehicle.

Vectors

Figure 3 - Vector diagram (scholbak, 2009)

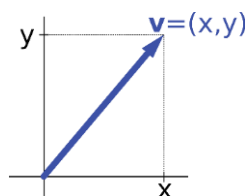
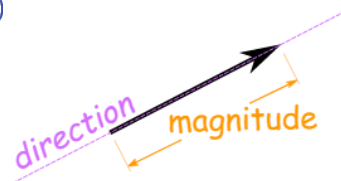


Figure 4 – Traits of a vector (mathisfun, 2021)



Vectors represent displacements between two coordinate positions, typically as the distance. A Vector is defined as having a direction and a magnitude (size). They are used as the foundation for navigating an entity around a global space. If an entity is at a location, then the vector from that location will be its future location.

Physics

Locomotion physics engines are formed out of Steering, Acceleration, Velocity, and Position. These properties depend on one another. Velocity is the rate of displacement over a period, generally noted as meters per second (m/s). The increase or decrease of velocity each second is determined by the acceleration of the vehicle combined with the steering force. The velocity stated as a vector of x y z will form the new location of that entity. This knowledge will form the foundation for the further chapters.

Webots

An example of simulations would be Webots. Webots is an open-source application used to simulate robots. Webots as a whole is an application that saves developers from reinventing the wheel, it doesn't require me to go through the complex nature of creating an entire physics engine from scratch as it has one integrated into the application according to cyberbotics (2022). It provides me with a sandbox of all the necessary toolboxes to develop a robot. This allows me in my implementation to save time, skip this level of development and focus on developing the navigation features. This is extremely beneficial from not a only development viewpoint but from a project management viewpoint by reducing the workload and density of my timeline.

ROS

Robotic Operating System (ROS) is a software framework, a set of standards used for creating robotics. These standards allow for applications on one robot to be used for every robot following ROS. It acts as a package that contains libraries and tools to assist software developers.

These tools range from packet management, hardware abstraction, distributed computing, low-level device control and many others according to Joseph (2018). Like Webots these tools aim to prevent the need to constantly reinvent the wheel. Having this framework also allows a formation of a community of developers. This can help collaborations on larger scale applications that would otherwise overwhelm a solo developer.

Examples

Joseph (2018) goes into the theory of robotics, what they are, how they operate, and what implementation methods exist. They explain the ROS framework and its most common features used in creating robotics applications.

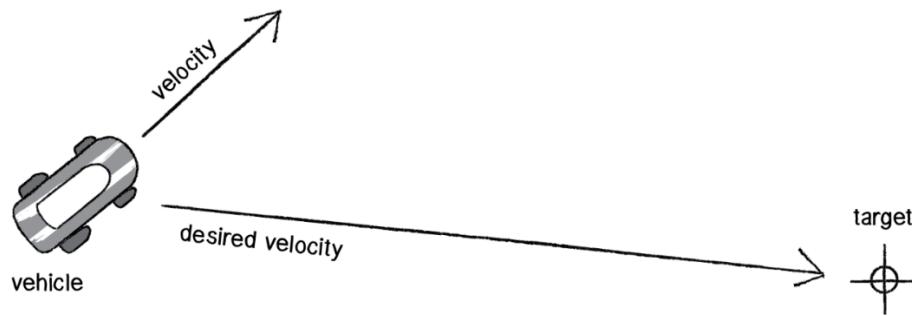
One of the most useful benefits I took from the paper that could be implemented into my project would be the Third-party integration of functionality from Webots, Integrated testing frameworks such as Rostest which should point out any potential or present flaws in my algorithms and language independence. Although I may not be integrating ROS into my project, I will be researching the potential use for ROS in the future, whether it be in my projector for the implementation of my genetic algorithms into the FS AI Car that uses ROS.

The paper also goes into a thorough explanation of the kinematics of robotics. It provided useful information and equations on how a robot is put into motion which will be carried forward into my simulation. These motions will potentially be the data for my input layers in my neural networks that will help it decide whether to accelerate or brake on which axis.

Overall, this paper explained a lot of the ROS frameworks, the rich toolset it contains then the motion of robotics. This information will be useful for my project, especially in the car controller for my simulation, having a good understanding of kinematics will help me understand the limitations of the race line with physics. The author continues to explain a lot of information largely irrelevant to my project such as the circuitry and creating a GUI. The author also never really goes into the concepts of kinematics when applied to neural networks or genetic algorithms, but they have given me enough foundational information to develop a solution for my project.

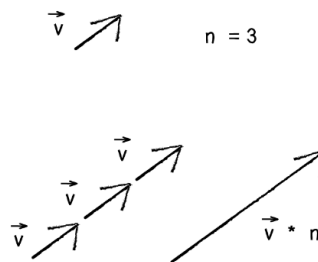
Path Following

Figure 5 – Steering force diagram (Shiffman, 2012)



As Reynolds (1999) stated, an autonomous agent must follow the steps of action selection, steering, and locomotion. We now know about the concepts of locomotion, but we don't know how locomotion is applied to steering behaviours. When explaining steering I gave an example of the seek behaviour about when an agent wants to move from its current location to its target. Now, this is quite easy to do with a vector when an object is stationary, but an agent will already be in motion most of the time. Therefore, we need to take into consideration the current velocity of the agent and the desired velocity leading to the target. **Steering force = desired velocity – current velocity**, this calculation is quite simple and will be used consistently throughout my project for tracking a target. It allows for the agent to turn towards the designated target without having to stop.

Figure 6 – Vector multiplication diagram (Shiffman, 2012)



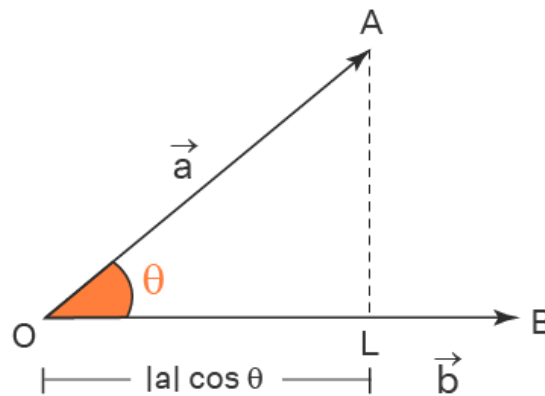
Steering Behaviour's

When it comes to steering behaviours, there are multiple types of motions. The existing seeking technique does the job, but it wouldn't work as efficiently for a moving target. Since it calculates the path based on the existing location of the target, the motion would be quite jittery when the target moves, and the agent needs to constantly course-correct. We want the steering to feel more natural and realistic.

Reynolds (1999) with his pursuit method comes up with the solution by predicting the future location of the target and setting that as the target, therefore, increasing the likelihood of the agent meeting the target with a more natural motion. This is done via vector multiplication, look at fig 6. Say we have the target current vector i.e., where it's headed. If we multiply the magnitude, we can make an accurate prediction of where it's going to be assuming it doesn't change direction.

In Path following, the path is defined as the vector or displacement between the beginning and the end. Reynolds (1999) refers to a path being a corridor with a centreline and then a radius that can be moved within freely. We don't want autonomous agents to be line followers, we want to set the radius as its environment and for the agents to have autonomy over their actions within said environment.

Figure 7 – Vector projection diagram (CUEMATH, 2022)



Vector Math

Although we want our agent to maintain as much autonomy as possible, it still needs to stay within a certain set of boundaries. In this project, I don't want my autonomous vehicle from going outside of the Silverstone track, but I want to maintain variation therefore I need a mechanism for the agent to follow the centre line when it's approaching the boundary.

For the agent to find its way back onto the path, it needs to seek a target relative to itself on the centre line of the path. This is a technique called vector projection, look at figure 7. CUEMATH (2022) uses the analogy of vector project as casting a shadow of vector **a** onto vector **b**. You can imagine in the context of this project **A** is the location of the vehicle; **b** is the path with **o** and **B** being the beginning and the end.

Cross product denoted as $\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \sin \theta$ provides a perpendicular vector of where the two vectors meet. θ known as theta is just a placeholder for the angle between the two vectors and $\|\mathbf{a}\| \|\mathbf{b}\|$ is the length of vectors **a** and **b** according to CUEMATH (2022).

The dot product denoted as $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$ is a scalar method of getting vector projection. It is the product of two vectors divided by the product of the two magnitudes. This can be quickly done by using the cosine between vectors **a** and **b** according to CUEMATH (2022).

Normalising the two vectors between 0 and 1 allows both vectors' magnitudes to be scaled. The direction remains the same, but it will become a unit vector. Finding the dot product between two-unit vectors can then be used to find the future location of the agent's relative point on the path. Referring to previous paragraphs this will then provide a smoother arc for the vehicle to lock onto the path without the constant need for correction.

Since we want the vectors to be scaled, we require a scalar quantity which is only a result from the dot product compared to the cross product which provides a vector quantity. For this reason, I believe the dot product will be integrated into my final implementation but both techniques will be applied.

Race line

Cardamone (2010) states multiple variations of the race line exist such as the minimum curvature or the shortest path, but neither is usually the optimal path. The optimal race line is very circumstantial to the constraints of the track and the vehicle that's being worked with. Therefore, it's crucial to have large amounts of testing data before and during the race to form a basis for producing the race line. A method is needed to be able to test multiple autonomous agents going around the same track with different configurations, continuously optimising with each iteration until the fastest but most feasible path is found.

Tattersall (2020) goes on to explain the importance of the race line in racing, explaining the various concepts such as the latest apex. This paper is about comparing various methods in deep learning that can be used to optimize the race line. The author explains the various machine learning techniques and how reinforcement learning is a potential valid path using the Markov decision processes in training his system. They explain the concepts of genetic algorithms but never really go into any incredible depth on how they could be implemented or even an outline of where to get started.

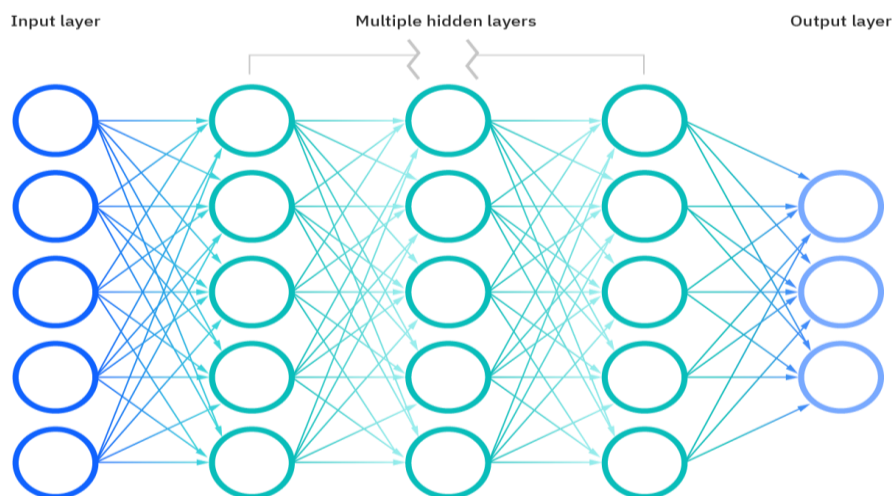
Like my vision, the author uses WeBots as his primary application in carrying out his algorithm, it's an incredibly versatile piece of software. However, it is clear from their implementation that the author started from scratch. This differentiates mine as I want my project to be implemented in the future on the FS AI car competition, I need an accurate virtual representation of Silverstone and a model of the car, and its relevant hardware being used. The author declares that his project could be useful for the team but is entirely independent of them.

Overall, it is clear the author took a different approach in his investigation, he starts with a wide variety of techniques, but his implementation seems to focus on the path of reinforcement learning. He stated that he intended to get around to genetic algorithms, mentioning it as a viable option but unfortunately, he ran out of time. He goes on to state that he wishes for someone else to continue his vision, which highlights that there is a gap in the research that my project can build from.

In my implementation, I'm going to be working with vectors, so it is critical to understand vector math to perform the natural behaviours of an autonomous agent and work with the simulation. The simulation provides the environment, and the data required such as the location but it's my job to take this data and implement the necessary operations to carry out the path following around the simulated Silverstone track.

Neural Networks

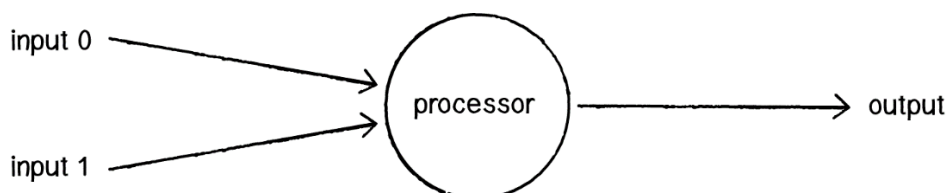
Figure 8 – Neural network diagram (IBM Education, 2020)



Neural Networks are an important algorithm in Deep Learning based on how the human mind works, explained by IBM Education (2020). This system is about the complicated relationship between neurons which act as nodes and are then divided into an **input layer**, **hidden layer**, and **output layer**, each with different responsibilities and handling the data from one another.

With every layer, there's a node and with every node, there's a connection, which forms a network. Figure 8 refers to a common neural network. Phil Kim (2017) describes the input layer as merely acting as an entry point for data to get passed to the next nodes. They don't perform any sort of calculation or function. The hidden layer is the inner network layer, it acts as sort of a block box that is not intended for the developer to access from outside the neural network. The output layer acts as a receiver from the hidden layer, outputting the weighted sum.

Figure 9 – perceptron diagram (Shiffman, 2012)



Looking more specifically at the previous figure 8 you can see a neural network is made up of figure 9. This is called the perceptron; it works on a feedforward system. Phil Kim (2017) explains the algorithm is made up of receiving the input, multiplying the input by the weighted value, summing the value and outputting the result.

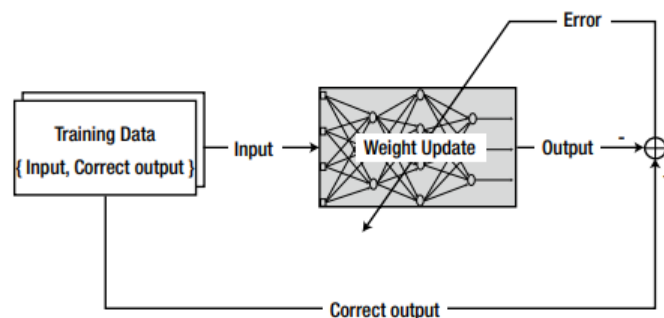
Between each neuron, there is a connection with a weight that dictates how much influence the input has over the output node and a bias that determines how far the output of the function is from the prediction. These values are usually assigned at random and then trained using the training set. The neural network will use the hidden layer to transform the data depending on the process. This is a process called supervised learning.

Deep learning

Kim (2017) states the benefits of neural networks are that they're adaptive! By changing the internal weights between the layers, they change their internal structure. As more hidden layers are added and therefore more nodes, more neural networks can be created with the structure becoming more complex. to the application. These are called deep neural networks which refer to depth.

Imran (2020) backs up that these structures can then be trained to determine logical decisions based on past experiences. Much like other networks, neural networks can then store information but when one neuron fails, it does not fail the entire network, therefore, preventing corruption.

Figure 10 – supervised learning diagram (Kim, 2017)



Supervised learning is when the training data is labelled with the input and correct output. We're essentially telling the neural network given a set of inputs, what it should be producing. For a neural network if it doesn't output the correct result then the internal structure is wrong, therefore the weights need to be shifted until it can output the correct result. We're essentially acting as the teacher, training the student. This technique is commonly used in facial recognition when matching faces to the correct name according to Shiffman (2012).

Unsupervised learning is when data isn't labelled therefore the machine needs to form categories of its own by looking for patterns, this is known as clustering. This technique won't be used for my project.

Reinforcement learning is when you award the neural network when it accomplishes an objective you like. This is typically used in simulations. Unity has a great selection of projects that use reinforcement learning, Samuel (2019) uses the technique to train an AI to park a car by grading each attempt. The AI will continuously learn from the "better attempts" and will eventually perform an optimal manoeuvre. This would be relevant to my application by training the car to follow the path, rewarding the vehicle when it follows efficiently or punishing when it's inefficient.

Kim (2017) states you can use all 3 techniques in 1 application. For my application, I believe Supervised and Reinforcement learning would be the most beneficial. Reinforcement learning sounds like a very similar concept to genetic algorithms, where you carry forward the best performing techniques or agents to optimise the system. He provides a useful book on machine learning with detailed explanations of all 3 learning methods explained above. The book also provides a detailed explanation of neural networks, what they are, how they're constructed and what benefits they provide. The book however doesn't implement neural networks with genetic algorithms, which is what I wish to use as an optimisation algorithm. This provides a gap in his research that I could build upon but will require my implementation, combining the multiple methodologies I've learnt over this literature review.

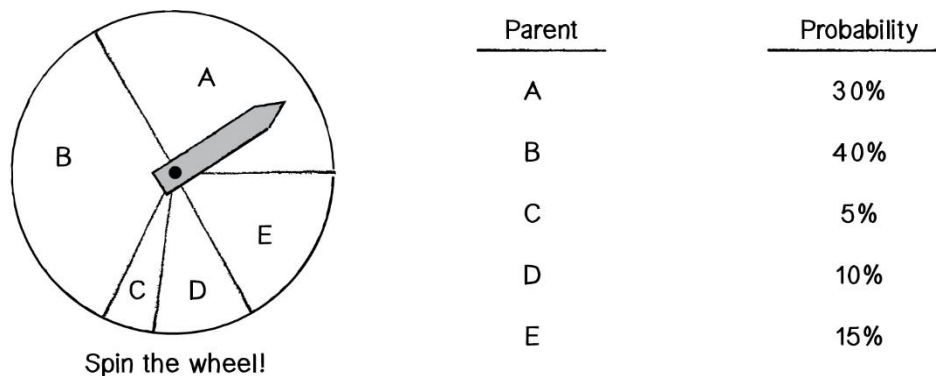
Genetic Algorithms

Genetic Algorithms are based on Darwin's theory of evolution from natural selection. It's an iterative deep learning algorithm that has each generation further adapting to its environment, optimizing the solution. Genetic algorithms are broken down into 3 steps. **Heredity**, **Variation** and **Selection**. Heredity is the act of passing down genetic information through reproduction.

In Programming, a gene is a feature. A gene is represented in an array or list which can contain bits, characters, or even vectors depending on the context of the agent and the environment. Biologically each element is considered a gene within the array, the entirety of the array would be considered a chromosome and multiple arrays would be the population.

Genetic Algorithms are largely optimisation algorithms, they take an existing dataset and provide an optimised configuration to the system. Having a high variation of data and unique genetic traits in the mating pool allows for a larger range of possibilities and a much higher likelihood of the resulting genetic trait being highly efficient in general, not just the best out of the population. It had to compete against strong candidates.

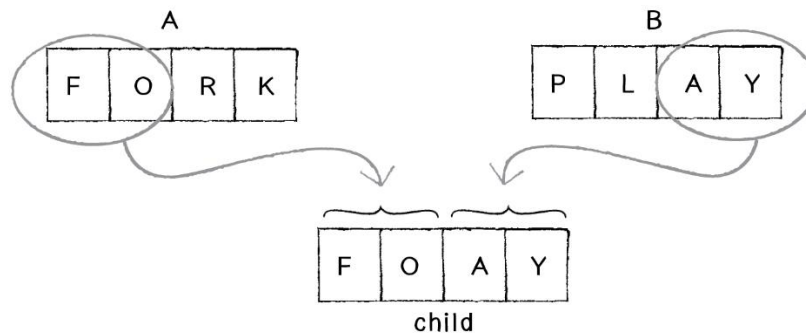
Figure 11 – choosing parent depending on probability (Shiffman, 2012)



A Fitness function is assigned to express the performance of how adapted the individual is to the objective. This score is a valid metric that can range from speed to accuracy, which is used to rank the chromosome based on its accuracy. Using this score, we form a mating pool of the top performers, then select the candidates based on probability, the higher the candidate's fitness score, the higher the chance they will be selected. This ensures not only the top two candidates get selected each generation, therefore, promoting variation in traits introduced. Sivanandam (2007) states It's important that the fitness requirements for the mating pool are fair, too high and the variation and reproduction process will be too slow, but too low then the solution will be sub-optimal.

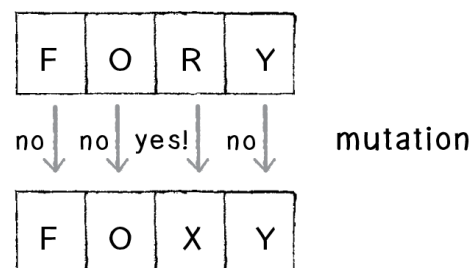
Selection states that the most adapted individuals to their respective environments reproduce more often, having a larger number of offspring that maintain their genes. With each generation, individuals must further compete to reproduce, resulting in more adapted gene pools than the prior generation. Once enough generations have passed, the parents would have reproduced with one another so much that the offspring will have closer genetic traits until they're all practically indistinguishable from one another.

Figure 12 – Crossover by mixing two elements from each array (Shiffman, 2012)



Reproduction is performed through a method called crossover. Based on fitness function probability two parents are picked to form a mating pool. Look at figure 12, we label the two parents A and B and look at their genetic arrays. We then take elements from both arrays to form the offspring array. This can be done in a multitude of methods, from taking two halves of each array to taking 3 from the parent with the higher fitness function and 1 from the other. It depends on the developer, for my system I believe I will take half from each parent to keep the system balanced.

Figure 13 – mutating an array (Shiffman, 2012)



The offspring will also go through a method called a mutation. Genetic algorithms normally don't create data, usually, they just optimise existing data. Using a mutation rate, the offspring's array will have a variable chance to have a genetic property altered. For example, the mutation rate could lead to the offspring having a 5% chance of having their top speed increased by 10%. This mechanism keeps an impending pressure to maintain variance, introducing new genes into the genetic pool which could end up becoming the dominant gene. These will then form the genetic pool for a new generation.

Review

Shiffman (2012) provides a large background on the concepts of processing data. Most of the chapters were largely irrelevant to what I was intending to accomplish, but the last two chapters highlight key concepts of how I could go around implementing my project. The last two chapters gave thorough research and explanation of evolutionary programming which is the category of what genetic algorithms fit. It goes into the 3 main steps of what makes up a genetic algorithm and what constraints are needed to be aware of such as a variation of traits. This book highlights the importance of introducing unique traits for variation.

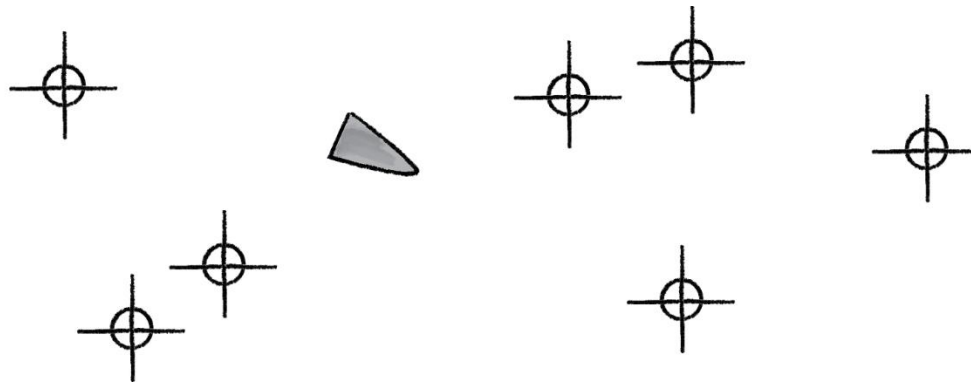
If each giraffe in a field has the same size neck and legs, therefore, can reach the same tree for food then there aren't any unique traits that allow evolutionary jumps to be made, everyone's the same! It also introduced to me that genetic algorithms don't construct genetic information out of binary data in arrays but also vectors which for a racing simulation is crucial.

What this book did best was using terms other research papers never used. It consciously made an extra effort to apply biological metaphors to explain the programming concepts, such as explaining the data within a feature as a genotype and then the expression of that as phenotypes. These metaphors made understanding the paper from a reader's perspective a much easier task and a kind of writing I will hope to apply to my dissertation.

The paper also introduced me to neural networks which got me thinking about how both chapters could work together to achieve better results by using genetic algorithms to tune the parameters of a neural network. Later I found out this was already an established concept called NeuroEvolution. This solidified for me the scope of how I would go about implementing my idea for the simulation.

Neuro Evolution

Figure 14 – Deciding which target to pursue (Shiffman, 2012)



I've now explained genetic algorithms and neural networks, the term NeuroEvolution is the combination of the two. As explained Genetic algorithms are optimisation techniques applied to a given population to see which genetic traits accomplish certain criteria the best. What if each agent in the population had a brain? What if instead of having static genetic traits, each agent is continuously learning within its generation to help itself gain the best performance and by extension fitness score.

We know from locomotion, that the agent has steering, acceleration, velocity and position. Shiffman (2012) suggests if we use those variables as the input layer, we can form a neural network that can decide those traits. This will allow the agent to decide which traits to put more emphasis on given its circumstances like a true autonomous agent. For example, if the agent was close to the target but facing the wrong way, it would place more emphasis on steering than velocity to align the heading. Now when the agent accomplishes the target, we need to tell the neural network it has performed the correct action. This will train the network using reinforcement learning, shifting its weights, and encourage the agent to perform similar actions in the future.

What we're essentially doing is reproducing a population of neural networks. Having deep neural networks providing short term learning on steering behaviours should theoretically speed up the genetic algorithm, therefore, reducing the overall generational count required to reach the most optimised configuration.

This correlates to my project. If I'm able to input the linear velocity, the angular velocity of the vehicle and distance from the corner, the neural network could potentially decide the correct traits when approaching said corner. This could optimise eual vehicle instead of having the optimisation take place between each generation.

3 Research Methodology

Experiment Constructs

Independent Variables

When conducting a study, independent variables are the values you change to explore how the results differ. It's not dictated by any other variables and therefore independent.

In my implementation, the independent variables would be the number of generations, cars per generation and mutation rate the genetic algorithm conducts. Changing these variables has a huge influence on the outcome of the system. Having a high quantity of cars allows for a higher variation, which as explained is one of the three pillars of a genetic algorithm. Having a high number of generations allows more heredity to take place, therefore allowing more of the mating pools to reproduce and therefore a higher offspring with genetically superior traits. A high chance for the offspring to mutate will allow for a higher variation outside of the initialisation of the vehicles.

Although having both the first two variables set too high extends the overall duration of the genetic algorithm. Having too many cars increases the likelihood of having a slow car which bottlenecks the generation since all cars need to either be finished or DNF to progress to the next generation. The program accomplishes the set number of generations no matter what, even if there are no performance gains to be made by reproducing.

For each vehicle, they will have independent variables as their genetic traits. This will set them as the maximum angle, speed and vision. As stated in the genetic algorithms, each of these will be balanced per vehicle to provide variation and will influence their performance in navigating around the track to the finish line.

Dependent Variables

Values that are influenced by other variables are called dependent variables. We don't care about changing these variables, just the outcome they produce, they tend to be the variables that hold static calculations or the results of the application. In my project, my dependent variables will be the fastest lap time and average lap time per generation and overall. Creating a graph from these findings will theoretically show where most of the performance gains were made and started to decrease.

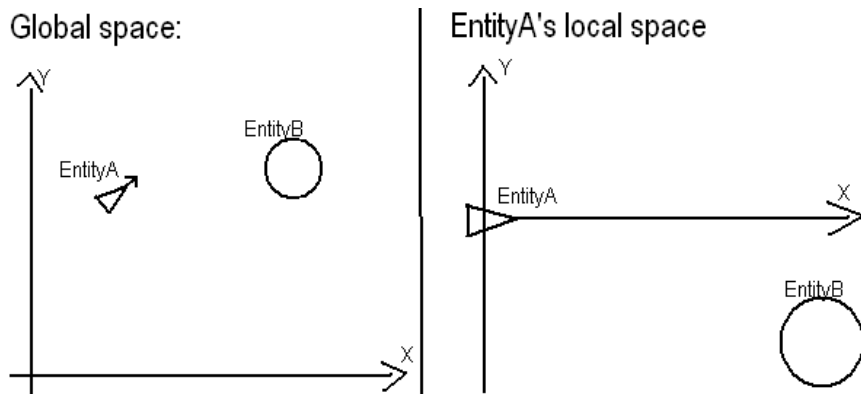
Constraints

The primary constraints of this project are the 3 relevant rules of the FS AI competition for Automated Driving Systems. IMechE (2022) states:

- The car will not be driven manually at any time.
- Path planning to determine the route through the cones.
- Respecting the track limits.

Simulator

Figure 15 – local & global space (Cohn, 2014)



I originally intended at the start of my project and throughout my literature review to use Webots. This project is intended to be a tool for the Formula Student team therefore I wanted to use the software and track environment of Silverstone that is currently being used. Midway through the development of the car controller I struggled in managing the Webots coordinate system in global space, so I decided to transition to a familiar tool called p5.js.

P5.js is a JavaScript library used for designing visual applications. I was better able to work with this system due to it having a single quadrant cartesian coordinate system compared to Webots which uses all 4 according to cyberbotics (2022). I was able to replicate all the locomotion physics required for an autonomous agent thus making it like Webots. The global space is also much easier to use due to being only in 2d space therefore I only need to work with x and y for the vectors compared to Webots 3d space, x, y and z.

P5.js has an extensive native library for handling vectors, with my attempt at websites I would either integrate my mathematical functions to find the dot product or use the NumPy library. P5.js has that integrated into the IDE (integrated development environment) preventing the need for me to reinvent the wheel.

I firmly believe this was the right call, especially since Webots doesn't have many examples to work from, making almost everything trial and error. P5.js has a large and thriving community surrounding it, with plenty of answered queries and examples. Since this isn't the tool that the main team uses, I declared this project to be a proof of concept, that would eventually be adapted into Webots.

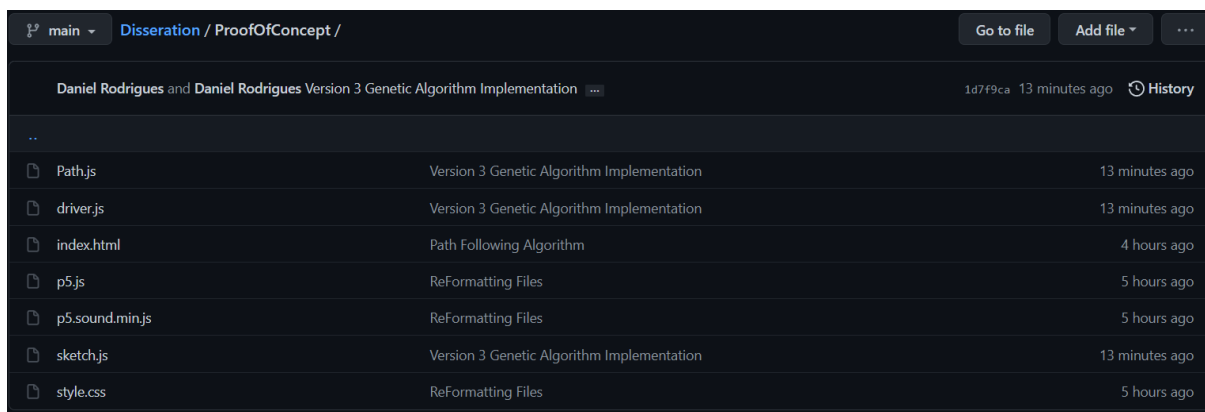
Implementation

Hardware

The system used for this project has an AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx, 2100 Mhz, 4 Core(s), 8 Logical Processor(s) and 8.00 GB of Installed Physical Memory (RAM). The code is written on P5.js which uses JavaScript 1.4.0. The code was then uploaded to GitHub using GitHub Desktop. On p5.js the only libraries that were used were the ones native to that IDE. In my proposal I intended to use C++ due to performance reasons, this then changed to python on Webots due to its simplicity but when I left Webots It changed to JavaScript. I'm familiar with JavaScript so it wasn't much of a difficult transition, only minor syntax research was required.

Version control

Figure 16 – GitHub repo



The screenshot shows a GitHub repository interface for 'Disseration / ProofOfConcept /'. The commit history table lists the following files and their commit details:

File	Commit Message	Time Ago
Pathjs	Version 3 Genetic Algorithm Implementation	13 minutes ago
driver.js	Version 3 Genetic Algorithm Implementation	13 minutes ago
index.html	Path Following Algorithm	4 hours ago
p5.js	ReFormatting Files	5 hours ago
p5.sound.min.js	ReFormatting Files	5 hours ago
sketch.js	Version 3 Genetic Algorithm Implementation	13 minutes ago
style.css	ReFormatting Files	5 hours ago

In software development, version control is used for incremental integration throughout a product's lifecycle. It allows for each iteration of an application to be compared to the previous version to show the changes made. This allows for rollbacks and comparisons which is useful in locating a bug that has occurred in the latest version.

When I started my original implementation on Webots, I required the Silverstone simulation used by the FS-AI team. Therefore, I cloned all the relevant files and created a new repository for me to create my iteration. I then started committing and pushing to my repository with detailed commits on what each push was implementing, fixing or updating. This was done using the SWW format (Summary, What, Why) to keep notes consistent and easier to read throughout the entire project.

When I swapped over to p5.js I created a new file labelled "ProofOfConcept" to store my new implementation. All navigation, path following, and genetic algorithms were done in the main branch. Development for neural networks was also going to be done in the main branch, this will be expanded upon in further chapters.

Path Following

Figure 17 – Simulated Track



Since I'm using p5.js I remade a 2d representation of the Silverstone circuit for the competition, look at figure 17. Following on from Reynolds's (1999) path following in the literature review, my path has a series of vectors connecting end to end as the centre line. From this centre line, a radius has been set, this is to act as a visual representation of the boundary that the autonomous agents must stay within.

Figure 18

```
class Path {
  constructor() {
    this.radius = 30; //
    this.nodes = []
  }
  /*
   * Initialises new node to path
   * @param {int} x
   * @param {int} y
   */
  set_point(x,y){
    let node = createVector(x,y);
    this.nodes.push(node);
  }
  /* Draws a line end to end between each node with a surrounding path line the size of the radius */
  output() {
    noFill();
    strokeJoin(ROUND);
    stroke(255);
    strokeWeight(1);
    beginShape();
    for(let element of this.nodes) {
      vertex(element.x, element.y)
    }
    endShape(CLOSE)

    noFill();
    strokeJoin(ROUND);
    stroke(255, 100);
    strokeWeight(this.radius*2);
    beginShape();
    for(let element of this.nodes){
      vertex(element.x, element.y)
    }
    endShape(CLOSE);
  }
}
```

Figure 19

```
function Initialise_Path() {
  path = new Path();
  let offset = 60;
  path.set_point(offset, height/2);
  path.set_point(offset, offset);
  path.set_point(width - offset * 11, offset);
  path.set_point(width - offset * 8, offset * 2.5);
  path.set_point(width - offset * 5, offset);
  path.set_point(width - offset, offset);
  path.set_point(width - offset * 9, offset * 7);
  path.set_point(width - offset * 11, offset * 11);
  path.set_point(width - offset, offset * 11);
  path.set_point(width - offset, height - offset);
  path.set_point(width - offset * 13, height - offset);
  path.set_point(offset, height - offset*5);
}
```

The formula student team must partake in multiple courses for different events at the competition. So, I wanted to ensure this program was adaptable to any track layout. Using the path class, I'm able to design, initialise and display any path. I have two properties set in the constructor, the radius for the centre line and the nodes for the turning points of the track. Each node is made up of an x, y coordinate so when iterating, I'm able to generate a vertex connecting each point stored.

Since the path is generated on the canvas pixel-based system p5.js works on, having static coordinates when initialising would look different depending on the resolution the system is working on. Therefore, I tried to make the dynamic of the position by making each node relative to the resolution. I haven't tested this on other screens, but I theorise this would make the track layout consistent.

Autonomous Agent

Figure 20 – Vehicle class

```
class Vehicle {
  constructor(id, dna) {
    this.id = id;
    this.position = createVector(60, height/2);
    this.velocity = createVector(10, 0);
    this.acceleration = createVector(0, 0);
    this.r = 15;
    this.score = 0;
    this.frames = 0; // timer
    this.finished = false;
    this.trail = [];

    this.dna = dna; // maxSpeed, maxAngle, vision
    this.maxSpeed = this.dna[0]
    this.maxAngle = this.dna[1]
    this.vision = this.dna[2]
  }
}
```

Figure 21 – update function

```
update() {
  if (this.finished == false){
    this.frames += 1;
    append(this.trail, this.position.copy());
  }
  this.velocity.add(this.acceleration);
  this.velocity.limit(this.maxSpeed);
  this.position.add(this.velocity);
  this.acceleration.set(0, 0);
}
```

P5.js doesn't have a native locomotion system therefore I created one based on Reynolds (1999). When constructing a vehicle, it has set default values. Its position is set by default for every vehicle to be the racing line. The main variables are velocity and acceleration. These are set initially as vectors as both contain x and y coordinates.

These values get updated every frame of the simulation therefore an update function is run. As per the literature review, velocity is the displacement of the location therefore we increase the position by the velocity at a constant rate, displaying momentum. Acceleration is the increase and decrease of velocity therefore we add, a positive or negative acceleration figure from velocity.

Steering Behaviours

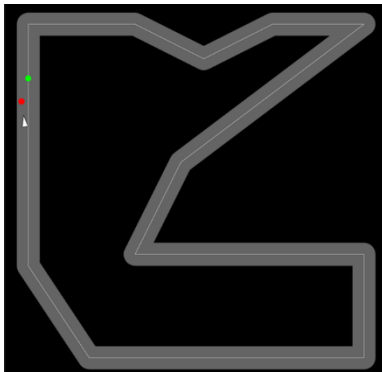


Figure 24 – Single vehicle

Figure 22 – Tracking code

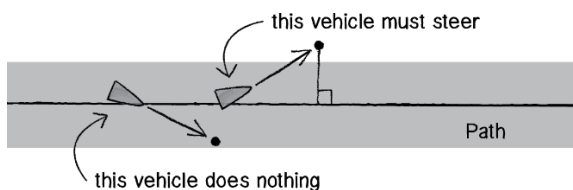
```
track(target) {  
  let desired_velocity = p5.Vector.sub(target, this.position);  
  desired_velocity.setMag(this.maxSpeed);  
  let steering = p5.Vector.sub(desired_velocity, this.velocity);  
  steering.limit(this.maxAngle); // caps magnitude  
  this.set_steer_angle(steering);  
}
```

Figure 23 – Vector projection code

```
vectorProjection(start, future, finish){  
  let future_start = p5.Vector.sub(future, start);  
  let path = p5.Vector.sub(finish, start).normalize();  
  let scalar_projection = future_start.dot(path);  
  return path.mult(scalar_projection).add(start);  
}
```

A vehicle in my simulation always acknowledges where the centre line is to the path and its position relative to it. Using the dot product, I calculated the projection of the current position onto the line. From this projection, we can create two vectors, our desired velocity and current velocity. Subtracting these two properties gives us a steering force, allowing the vehicle to move towards or track the object.

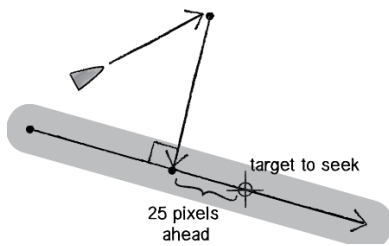
Figure 25 – If the line is off the path, steer back to path (Shiffman, 2012)



Each vehicle has a vision property, this can either be defined by how far the vehicle can see or as in the literature review, the vehicle's future location. In the constructor, this is defined as one of the three genetic traits for each vehicle in my population. In figure 25, its represented as the red dot.

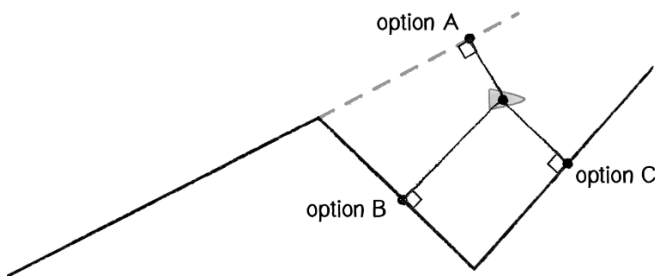
I want the vehicle to still be an autonomous agent, so although it's passively aware of where it could track, it doesn't always. One of my constraints was for the vehicle to maintain itself within the radius of the path. My balanced approach to this problem is to take the future location of the vehicle, then declare a condition to only track the centre line if that vehicle's future location is outside of the path's radius. This maintains the natural steering and locomotion of the vehicle, only rectifying when performing action selection.

Figure 26 - Vector projection (Shiffman, 2012)



One of the common problems noted to me by my supervisor, having autonomous vehicles only tracking the centre line when going outside a region tends to result in the vehicle trailing the outer boundary line due to consistently toggling between track mode and free roam. To solve this, I could either A have it consistently tracking the centre line but the aggression of the tracking changes depending on the distance from the centre line or B having the future location be what alters the tracking. Both are viable solutions, but B seemed to be more in line with my research which ended up being carried forward into my implementation. To ensure a smoother motion, I implemented the Craigs Reynolds pursuit technique, tracking the future location of the normalised vector projection.

Figure 27 – Path deciding (Shiffman, 2012)



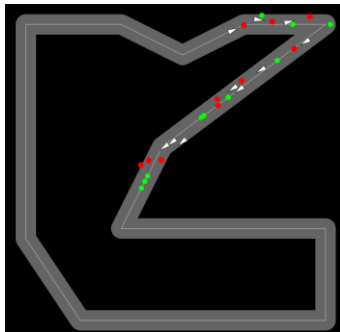
Path following is quite simple when it's just a singular line, one of the difficulties I had during my implementation was designing the vehicle to know which path to lock onto when it comes to corners. When the vehicle approaches the boundary, it attempts to find its vector projection to the line but when multiple paths exist, how do we choose?

To solve this issue, I created a system that iterates through all the paths and finds the shortest length to a valid projected point. All the path nodes are stored in an array in the path class. By initialising a max distance variable to infinite, I iterated through each path to find which path has the smaller distance through comparison. If that path's projected point has the smallest distance to the vehicle, that will be the point the vehicle pursues. This system acts as a safety mechanism to prevent navigation confusion.

Considering my constraints if the vehicle still manages to find itself outside of the path radius, it will be deactivated. These are not the results I want to carry forward into my genetic algorithm and final results.

Genetic Algorithm

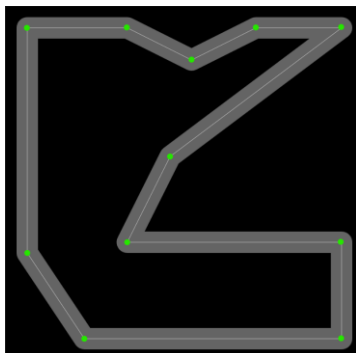
Figure 28 – Example run of my application



So far in my implementation, I have made a single autonomous vehicle that can navigate around the track. The first component I need for my genetic algorithm is population. In the sketch / main class, I create a global array to store the current population. Iterating 10 times I create 10 new objects of the vehicle class. The constructor of the vehicles requires a DNA array and an id. Since this is the initial generation of the genetic algorithm, no heredity has taken place therefore the DNA traits are set to random between a range of numbers. The id is the current iteration it got generated in so 1 to 10. Each vehicle has 3 genetic properties that affect the mechanics of how its locomotion and navigation. DNA is an array that contains the vision, max speed and max turning angle.

Fitness

Figure 29 – Point posts around the track



I wanted to create a balance of being able to create a fitness function and reinforcement learning. At the end of each path which in figure 29 is represented as a green dot, if the vehicle goes within proximity with the same radius of the path, it gains a point that adds to the score variable in the constructor. Each vehicle has a timer, which adds 1 for each frame of the game that they haven't reached the finish line.

Once every vehicle has reached the finish line, I iterate through each vehicle and created 2 separate arrays containing normalised scores and timers. Since the timer values go into the thousands but the score values range from 0 to 11, it would create an imbalance of value in the timer. Normalisation makes both values between 0 and 1 by taking the min and max of each array. This allows me to take go through each index and then create a fitness function for each vehicle by averaging their now normalised score and timer. One issue I had was once all the scores and timers were so optimised, and each vehicle was getting identical values I couldn't normalise the min and max as they were indifferent therefore, I created a condition to prevent this by setting them all to 1 if all elements were equal.

Mating Pool

Figure 30

```
function PoolSelection(fitness){
  var error = 0;
  while(true){
    var index = floor(random(vehicles.length));
    var prob = random(Math.max(...fitness));
    if (prob < fitness[index] + 0.0001){
      return vehicles[index].dna;
    }
    error++;
    if (error > 10000){
      return null;
    }
  }
}
```

This function operates by inputting the fitness array into the function. I have two variables index which gets a random index from my vehicles. I then generate a random number from 0 to the max element in the fitness array which since it's normalised is 1, this number acts as a probability figure. I declare if the number picked for probability is less than the index, return said index. The highest number 1 will almost certainly be greater than the probability as its 100% the highest number, 0.8 will be 80% likely to be greater and 0.2 will be 20% is unlikely to be picked. This loop indefinitely iterates with a while true until a successful vehicle's fitness passes the condition and then said vehicle's DNA will be returned.

Since while true is an indefinite loop, if the conditions aren't met it will run forever or until there's a memory leak and it crashes. To prevent this, I have entered a counter called error which ensures the loop runs for 10000 iterations. If the loop has to run for that many iterations, the mechanism has most likely failed thus returning null.

Heredity

Figure 31 – Heredity flow chart in my implementation

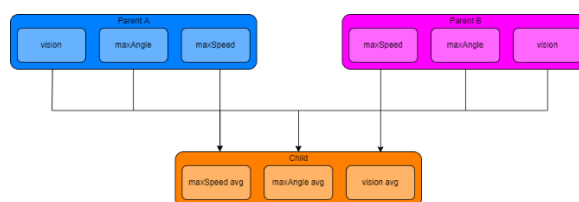


Figure 32 – Crossover code

```
function Crossover(parent1, parent2){
  var child = [];
  child[0] = (parent1[0] + parent2[0]) / 2;
  child[1] = (parent1[1] + parent2[1]) / 2;
  child[2] = (parent1[2] + parent2[2]) / 2;
  return child
}
```

One of the pillars of a genetic algorithm is to pass genetics to the offspring. From the literature review, I perform a variety of crossovers between the 2 parents. I take the DNA arrays from both parents and average out each corresponding trait therefore the offspring gains a combination of all their traits. This is stored in an array and then returned.

Mutation

Figure 33 – Mutation code

```
function Mutate(child){
  let rate = 0.1;
  for(var i = 0; i < child.length; i++){
    if (i == 0 && random(1) < rate){
      console.log("Mutated!!!")
      child[i] = random(child[i], 5)
    }
    if (i == 1 && random(1) < rate){
      console.log("Mutated!!!")
      child[i] = random(child[i], 5)
    }
    if (i == 2 && random(1) < rate){
      console.log("Mutated!!!")
      child[i] = random(child[i], 20)
    }
  }
  return child
}
```

Once the child has gone through crossover, the result gets mutated. I iterate over each element in the array and if a random number is less than the mutation rate, the element gets mutated to a random number between itself and the max value in the traits initialiser in the setup function. This makes the mutation rate an independent variable that changes the likelihood of a value being altered. The mutated child is then returned.

Generations

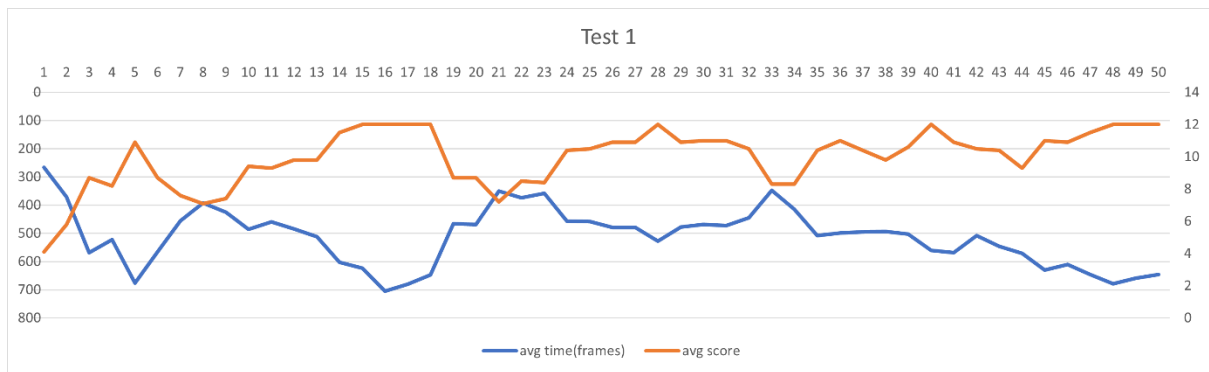
Figure 34 – Initialising generations

```
if (generation < generations){
  let completed = 0;
  for (var i = 0; i < vehicles.length; i++){
    vehicles[i].main(path);
    if (vehicles[i].finished == true){
      completed += 1;
      if (completed == 10){
        generation += 1;
        nextgen = New_Generation();
        console.log("generation: ", generation)
        for (var y = 0; y < 10; y++){
          vehicles[y] = new Vehicle(y, nextgen[y]);
        }
      }
    }
  }
}
```

Once all of the offspring DNA has been outputted, a new generation of children is created. This is done similarly to the setup by creating 10 new objects but this time instead of creating random DNA, I'm inputting the offspring's DNA into the constructor. This then begins a new race, and the cycle repeats itself until the current generation is equal to the max generation set.

4 Results

Figure 35



Configuration 1

- Max Speed: 3 to 15
- Max Angle: 3 to 15
- Vision: 5 to 20
- Generations: 50
- Mutation Rate: 10%

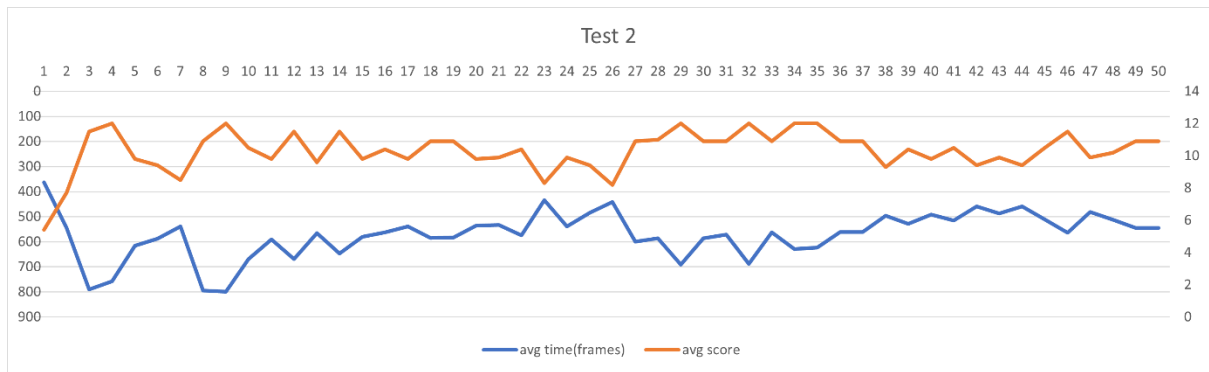
Fastest Last Lap

- 640 frames, 12 points



Figure 36

Figure 37



Configuration 2

- Max Speed: 3 to 10
- Max Angle: 3 to 10
- Vision: 5 to 20
- Generations: 50
- Mutation Rate: 10%

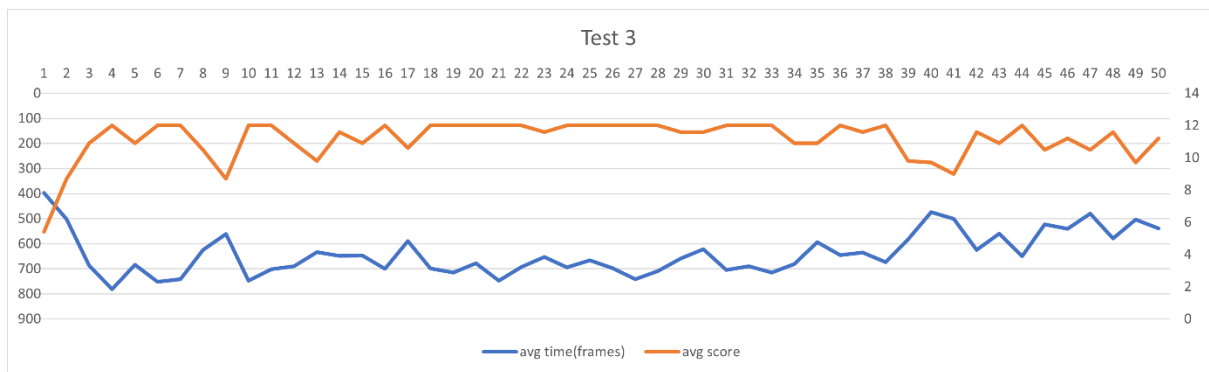
Fastest Last Lap

- 278 frames, 12 points



Figure 38

Figure 39



Configuration 3

- Max Speed: 3 to 10
- Max Angle: 3 to 10
- Vision: 5 to 20
- Generations: 50
- Mutation Rate: 20%

Fastest Last Lap

- 481 frames, 12 points

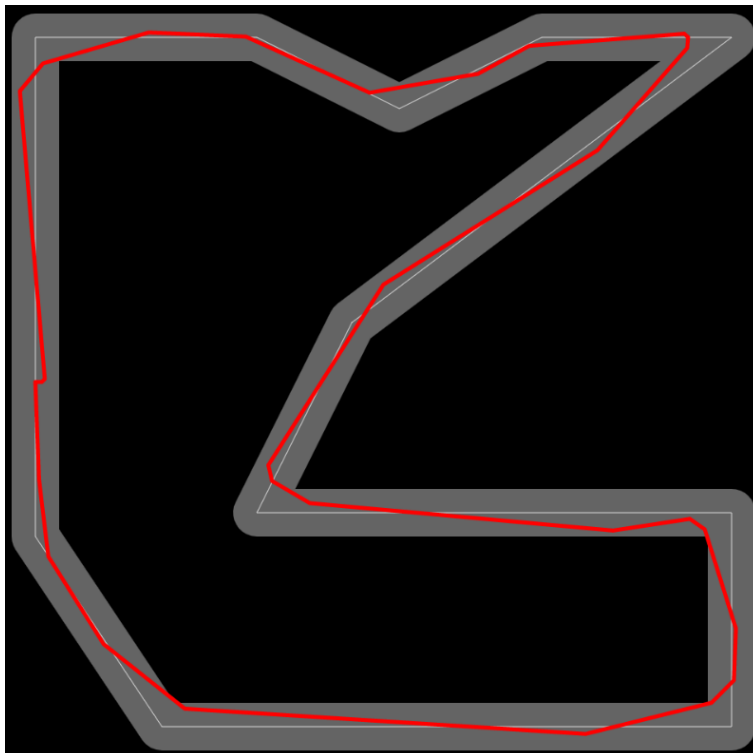
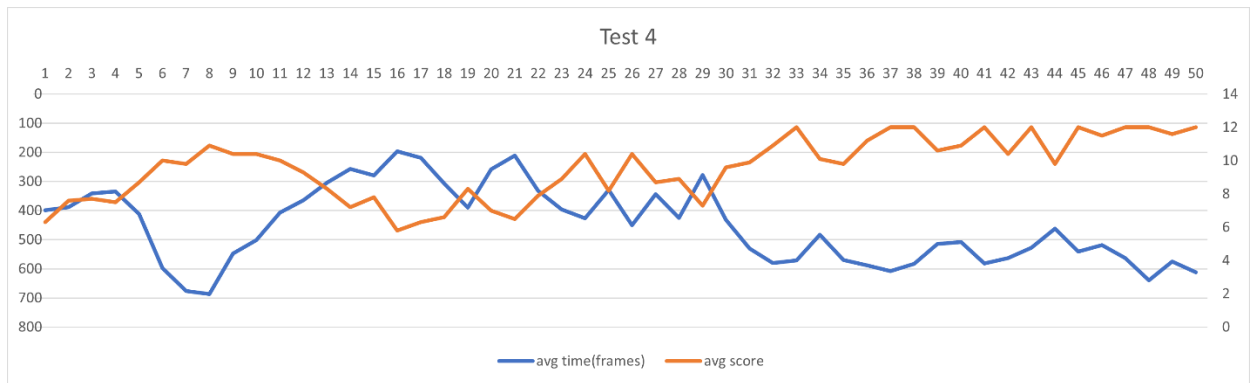


Figure 40

Figure 41



Configuration 4

- Max Speed: 3 to 15
- Max Angle: 3 to 15
- Vision: 5 to 20
- Generations: 50
- Mutation Rate: 20%

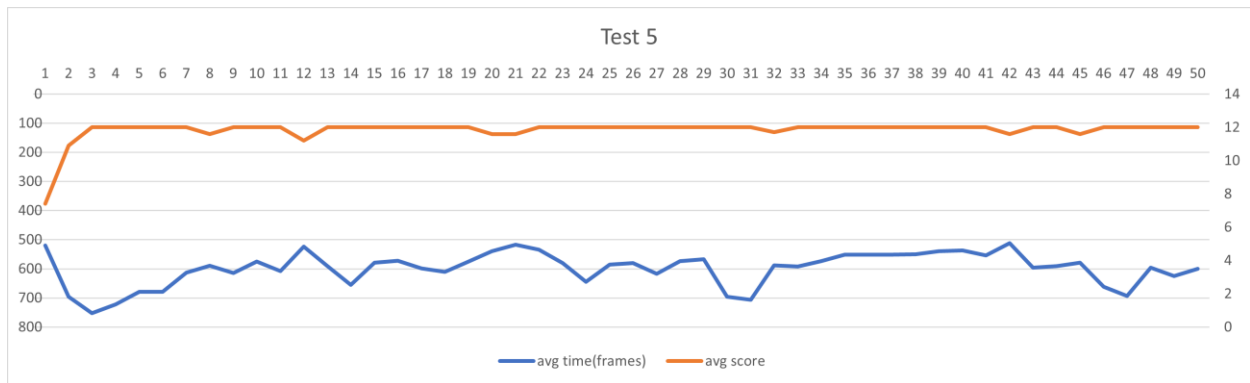
Fastest Last Lap

- 528 frames, 12 points

Figure 42



Figure 43



Configuration 5

- Max Speed: 3 to 10
- Max Angle: 3 to 10
- Vision: 5 to 30
- Generations: 50
- Mutation Rate: 10%

Fastest Last Lap

- 470 frames, 12 points

Figure 44

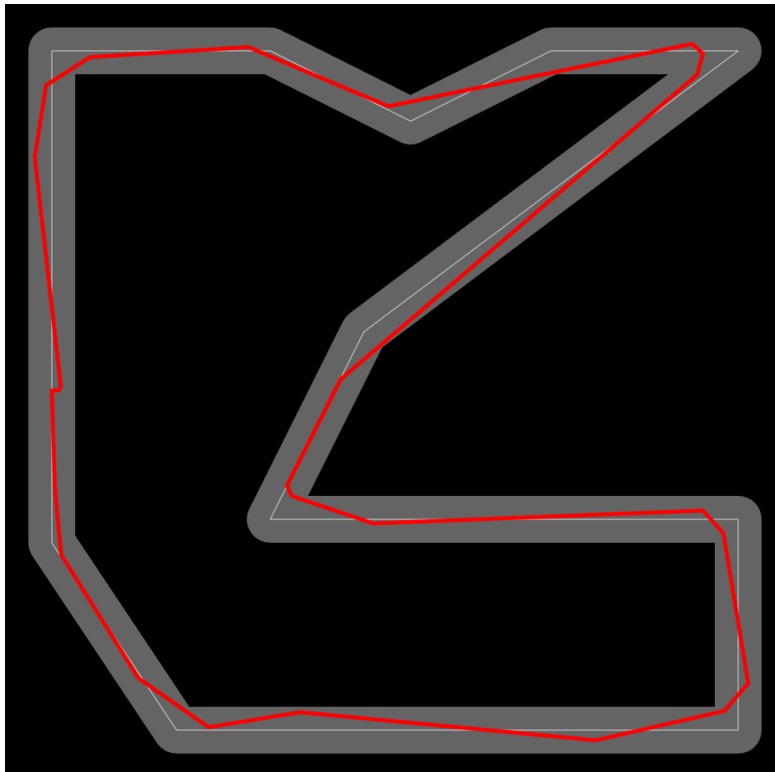
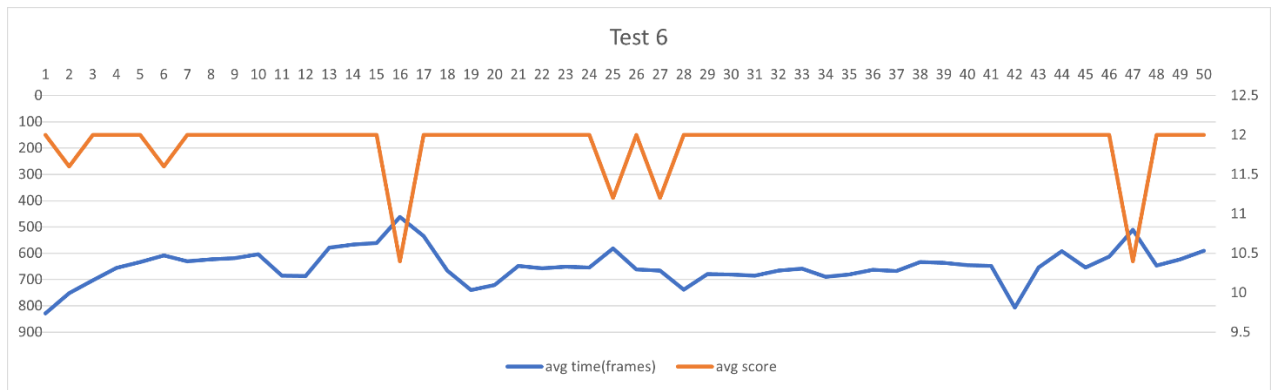


Figure 45



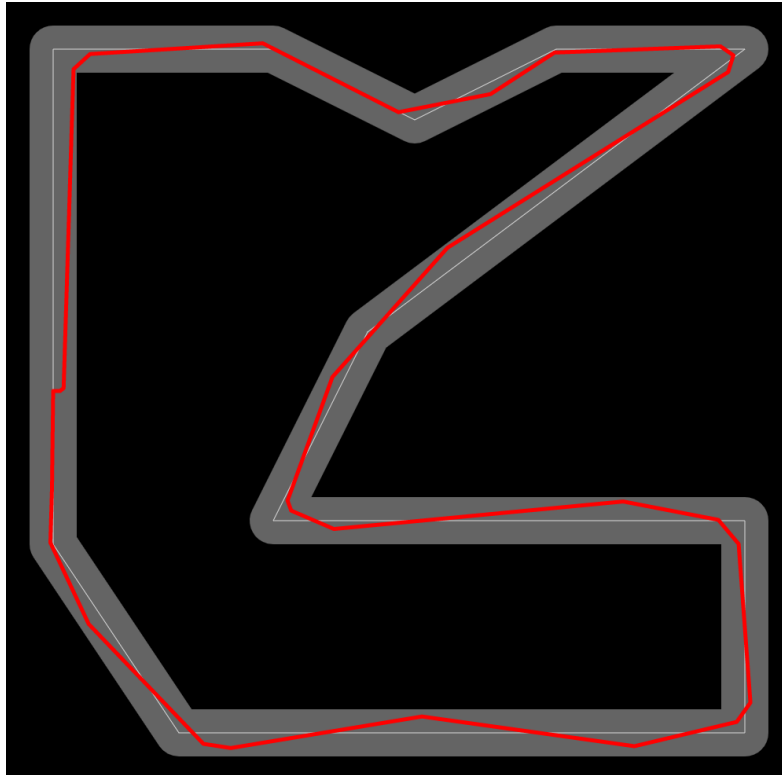
Configuration 6

- Max Speed: 3 to 10
- Max Angle: 3 to 10
- Vision: 5 to 10
- Generations: 50
- Mutation Rate: 10%

Fastest Last Lap

- 484 frames, 12 points

Figure 46



Test 7

Iteration	avg time(frames)	avg score
1	350	5.5
2	330	5.0
3	300	5.5
4	250	6.5
5	280	7.5
6	260	7.5
7	240	7.5
8	220	8.5
9	210	10.0
10	200	10.0
11	190	9.5
12	180	8.5
13	170	9.5
14	160	8.5
15	150	9.5
16	140	8.5
17	130	8.5
18	120	8.0
19	110	7.5
20	100	8.5
21	90	9.5
22	80	10.0
23	70	9.5
24	60	8.5
25	50	8.5
26	45	8.5
27	40	9.5
28	35	8.5
29	30	9.5
30	25	10.0
31	20	9.5
32	15	8.5
33	10	7.5
34	5	6.5
35	10	7.5
36	15	7.5
37	20	8.5
38	25	9.5
39	30	8.5
40	35	7.5
41	40	8.5
42	45	7.5
43	50	8.5
44	55	9.5
45	60	8.5
46	65	7.5
47	70	6.5
48	65	7.5
49	60	8.5
50	55	9.5

- Max Speed: 3 to 15
- Max Angle: 3 to 15
- Vision: 5 to 30
- Generations: 50
- Mutation Rate: 10%

- 240 frames, 12 points

5 Results Analysis

Outputs

The implementation of my genetic algorithm turned out to be a success. I managed to conduct several tests without fail and then form detailed graphs for all the results with a summary table for comparison. The several tests include changing my independent variables into various configurations to see which variables make the biggest impact on the outcome and how.

I implemented a new trait called trail within each vehicle that stores the current position in an array for every frame of the generation. In the 50th generation, I would find the vehicle with the highest fitness score and take their trail. Since the vehicle with the highest fitness score is the most optimised vehicle and the 50th generation was the last one, I declared this trail to be the most optimised path. I plotted this trail using the p5.js vertex features, similar to when initialising the path by declaring each index of the trail to be a vertex. This resulted in an accurate path plotted onto the track.

Comparison

Table 1 - Results

Test	Average (Frames)	Average (Score)	Last Gen Fastest Lap (Frames)	Last Gen Final Score
1	512.166	9.944	640	12
2	571.572	10.296	278	12
3	638.95	11.136	481	12
4	452.17	9.548	528	12
5	598.698	11.824	470	12
6	648.88	11.888	484	12
7	463.004	9.944	240	12

The results in Table 1 show the various results for the different configurations ran. Overall, what matters the most is the latest generation's score and fastest lap. This is the most optimised and what would in practice be carried forward to the FS AI car. However, I believe the averages show an extra perspective of how consistent and reliable the results are. The average score shows how well each vehicle, completed the full course. A low resulting average score but a good final gen lap time and the score could have resulted from a fluke, which would be difficult to recreate. So, when choosing a configuration before the race reliability of getting a good optimised over the best performing race line could be taken into consideration.

Purely based on perform configuration 7 wins. This is due to the vehicle having a high pace and steering force whilst also having a higher vision to support the turnings. This is proved by comparison to configurations 1 and 2 which also have the same speed and angle ranges but a lower vision range.

From figures 41 and 39, the mutation rate resulted in the genetic algorithm struggling to optimise the configuration since there was a 20% chance of a car having its traits mutated. Although this adds variance, too much variance slows down the optimised result, therefore, requiring more generations to accommodate.

Implementation issues

P5.js provided a lot of the tools and documentation surrounding vectors and navigation that Webots didn't. The disadvantage of P5.js is that it runs on a web browser, since chrome is limited to a certain amount of memory from my system, it bottlenecked performance when outputting large amounts of information to the console. Webots is a desktop application that had access to a much larger share of my systems resources and was able to handle these tasks.

P5.js also counts each vehicle's timer depending on the framerate of the system. This means the results would widely vary depending on the refresh rate of your system. Whenever I had another application open in the background, it would cause p5.js to stutter if my genetic algorithm was running which would skew the frames/timer results. Due to this whenever I ran the genetic algorithm for all my tests, I had to ensure no other application was open to getting accurate and fair results.

In my implementation I normalised each array for score and time, I did this so when calculating the fitnesses, it was fair. Each vehicle if they go outside the path radius, stops and is finished with the race. Through development, I had an issue with vehicles hitting the radius the side of the path early on and although they get a poor fitness for the score, their time would be amazing therefore boosting their overall fitness. To compensate for this, I made the weight for score double that of time so getting further in the track is valued more than getting to the first corner the quickest. Then when all the vehicles are getting the max score (12) then the better fitness is who gets around the track the quickest. This was a quick fix but creates a bias in how the system is rigged to value one trait over the other rather than both score and time equally.

6 Project Management

Development Methodologies

When beginning development on a project, you need a structure of principles and practices that set a certain standard of quality. Development methodologies are effective in providing a clear view of the tasks, barriers, goals and overall scope of a project. Cohen (2019) states there are two variations of project management incremental and iterative. An example of iterative development would be Scrum.

Scrum places emphasis on providing value to the customer through effective communication and progress throughout the entire project. Scrum begins with a product backlog; this is the owner's wish list of features they want to be incorporated into the final deliverables. Digite (2022) states during development, scrum operates on a process called sprints, usually lasting 2 to 4 weeks. It involves rapid development cycles where each cycle has a set of prioritised tasks, that need to be developed to the customer's standard. The start of the cycle has a daily scrum meeting laying this all out and by the end of the cycle, these features are reviewed and reflected upon with the client, typically with a prototype. If the customer is satisfied the next cycle or iteration begins with new features. Scrum has a large overhead of communication, which is usually required for teams that can accommodate scrum masters and multiple team members but not for an individual project like this one.

I have decided to use Agile Kanban boards which is an incremental methodology. A Kanban board contains cards that represent features. These cards are placed onto columns named backlog, planned, in progress, developed, tested and finished. These represent the stages a feature goes through in development before being delivered to the customer. All the features are independent and can be developed at any point. The con of this methodology is that to ensure development is manageable, each column has a set capacity on how many features can be developed at any given moment. Inflectra (2022) claims to do this forces the developer to take an incremental approach to the project.

Kanban and Scrum are both agile methodologies. Agile is a framework consisting of 12 principles that contribute to cycles of feedback and continuous integration of high-quality features according to Agile Alliance, 2021 (2021). The main principle I tried to integrate into this project was principle 12. This is the principle of always being adaptable depending on the current landscape. I found this useful when I had Covid 19 as it allowed me to readjust my schedule and tasks when I was unable to focus and get them completed by my original deadlines.

Where Kanban differs from Scrum is that its more suited to a single developer, with a more simplistic approach to development due to less communication overhead. For project management, I decided to use a tool called click up, Kanban board. The Kanban board originally had a backlog of all the features I need to accomplish for my project. Each task had an estimated time to achieve with a deadline it needed to be done by. Figures 49 to 61 show the backlog, with figure 62 showing the current state of the Kanban board.

Figure 49

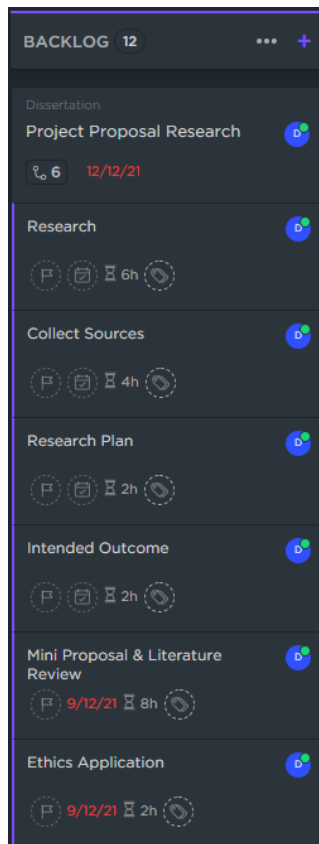


Figure 50

Figure 51

Figure 52

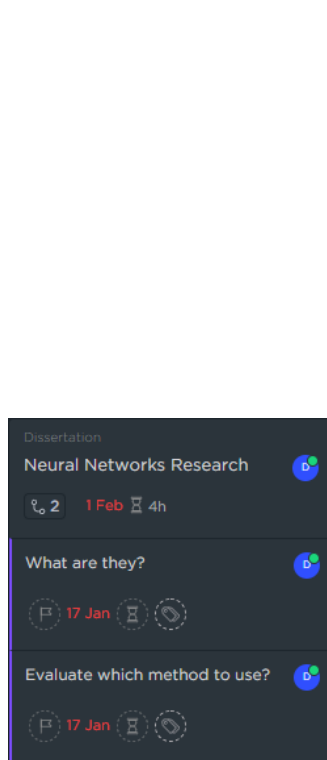


Figure 53

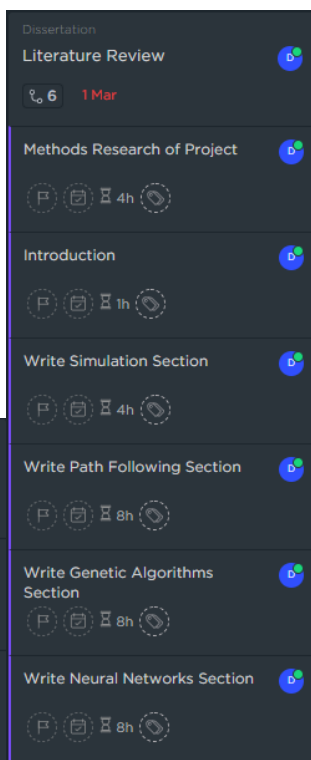


Figure 54

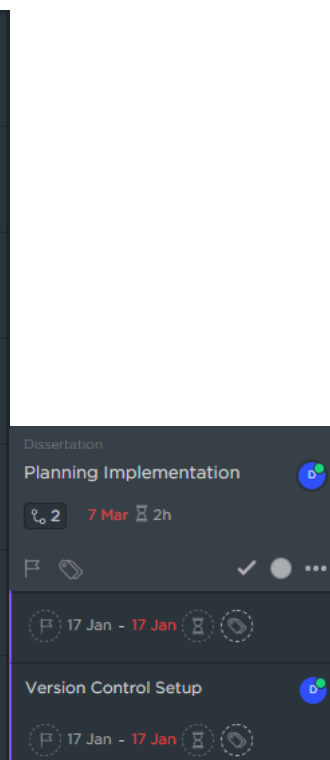


Figure 55

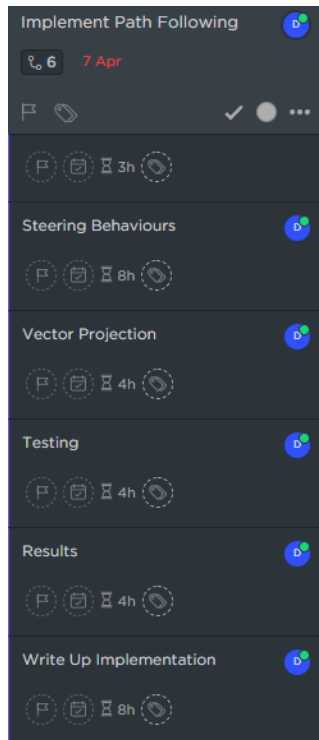


Figure 56



Figure 57

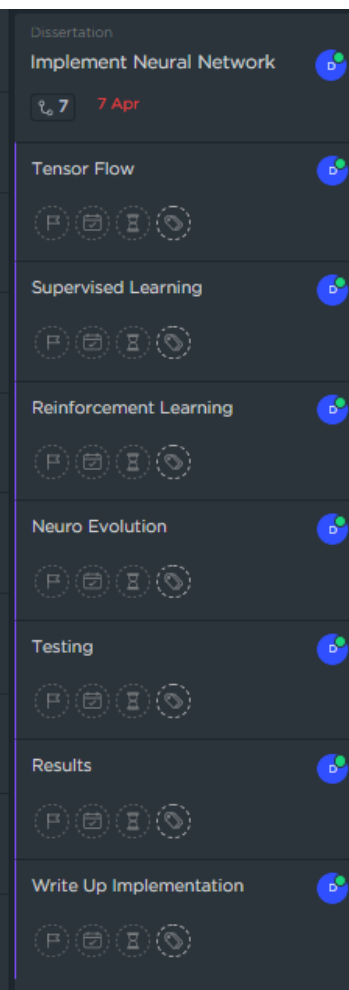


Figure 58

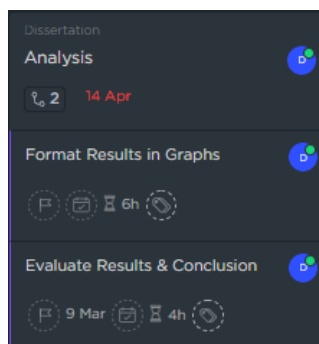


Figure 59



Figure 60

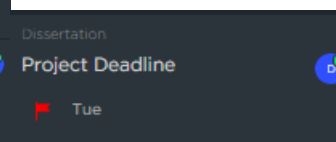
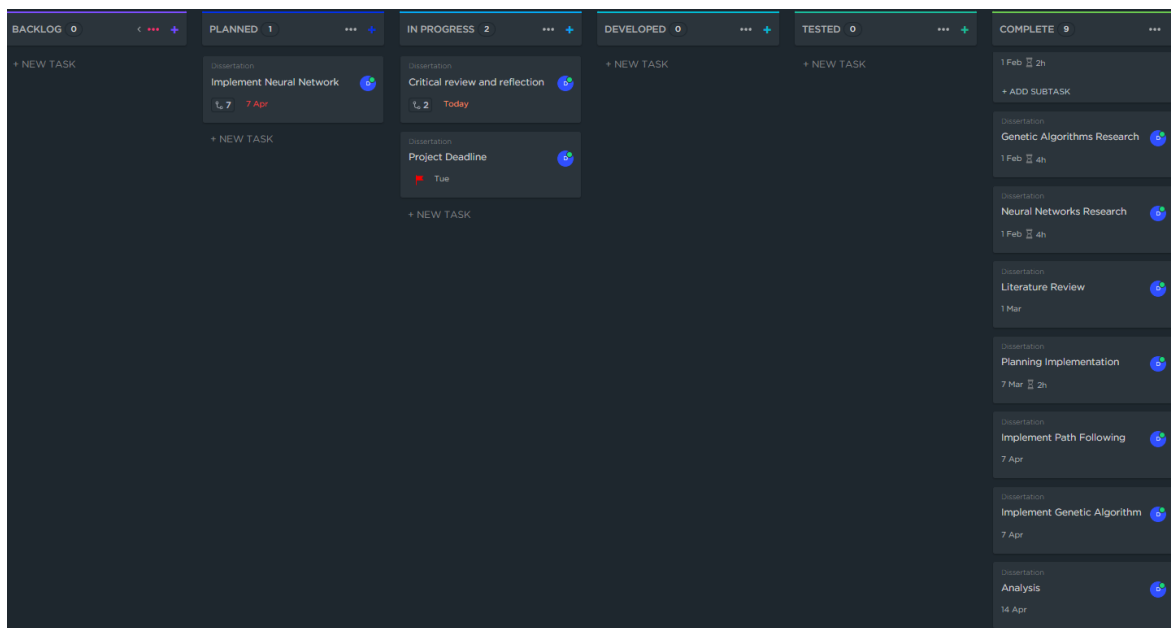


Figure 61 – Kanban board



Social and Legal Considerations

The copyright act of 1988 gives creative control to content creators over their work according to the Intellectual property office (2011). I ensured all work that has been published by other sources has been correctly referenced using the Coventry University Harvard format. This goes for the research, code and figures used in this paper. All other content has been signed in a letter of originality to be my own.

Communications act of 2003 controls how I can handle communication services such as messages and emails. gov.uk (2003) states the act protects against crimes by sending indecent or grossly offensive messages. This act was applied throughout my project when communicating with my supervisor and other members of his group.

Computer misuse act of 1990 covers by gov.uk (1990) in section 1, unauthorised access to computer material, section 2 explains the further intent to commit offences. All material used was publicly available, and no private information was used in the making of this project. All written and practical work created was done on my personal computer.

Ethical and Professional Considerations

British Computer Society (2021) has 4 core pillars in its code of conduct that all members should abide by. **“Public interest”**, this project is for everyone to learn from and is addressing issues from within the FS AI Team. **“Professional Competency and Integrity”**, this project has only pushed me forward within my competency, I’ve only tried to make claims with evidence to back them up and within my knowledge made no malicious attempt at discrediting anyone. **“Duty to the relevant authority”**, I accept responsibility for every non-credited work as my own in this project. **“Duty to the profession”**, this project was designed to provide a tool to improve the standards of my colleagues on the FS AI Team. The project has filled in gaps from researched projects in my literature review, which has only ever pushed the profession forward. I encourage anyone conducting their project on a similar subject to do the same with mine.

Version control has been used to back up my code but also to see the differences in the incremental developments of the application. If a newly developed function starts to bug out, version control allows me to see what was working before the function and how the developed function could be incompatible therefore providing a solution.

All my code on GitHub has been clearly explained using correctly formatted docstrings and individual comments. This is to ensure anyone looking to recreate or expand upon my work, can understand the internal functions and mechanism that makes the application operate. The code is open source for a reason, it is my duty to the public interest for anyone wishing to use my material to be able to do so. To be able to improve my work, a developer needs to understand my work therefore comments are crucial in providing this.

All Commit notes were done in a Summary What Why for a detailed easy understanding of incremental developments of my project. I have tried to consistently list all the new functions that have gone into each version of my project and what their purpose is.

Risk Management

Table 2 – Risk management

Asset at Risk	Hazard	Scenario	Contingency	Probability	Impact
Project Development	Development problems	Development barrier	Research solutions	Medium	High
Workload	Burnout	Slower development	Extend development time	High	Low
Schedule	Lack of time	Unfinished features	Focus on core features	Medium	Medium
Mental Health	Stress & Burnout	Slower development	Extend leisure time	High	High
Physical Health	Covid 19	Delays development	Wear a mask in public space	High	High

Meeting Records

Table 3 – Meeting Records

Date	Subject	Topics	Feedback
03/11/2022	Project Proposal	Proposal Subject	Create an idea
04/11/2022	Supervisor Swap	Carey Pridgeon to David Croft	Start attending new meetings.
11/11/2022	Supervisor Swap	Misunderstandings	Confirm to Carey the swap.
11/11/2022	Project Proposal	Proposal Subject	Find Resource materials. Use reference management software.
25/11/2022	Project Proposal	Proposal Literature review	Research methodologies on machine learning techniques to do with race line optimisation and start literature review.
06/12/2022	Project Proposal	Submission	Submit to David Croft for review and submit ethics application.
27/01/2022	Individual Project	Webots setup	Setup difficulties on Webots 2022 with FS AI worlds use 2020 version
17/02/2022	Individual Project	Webots difficulties	Look at the difference between global space and local space as a

			solution for navigation difficulties.
24/02/2022	Individual Project	Webots difficulties	Look at a PID system for tracking intensity. Autonomous Agents on or off track tend to trail the boundary.
03/03/2022	Individual Project	IDE Change to p5.js	Continue using p5.js if you're having more development progression.
10/03/2022	Individual Project	Progress update	The pathing algorithm is finished, move on to the genetic algorithm.
17/03/2022	N/A	N/A	I was unavailable but I notified David. Try to make small incremental updates to the literature review.
24/03/2022	Individual Project	Progress update. Submit sections for review and feedback	The genetic algorithm is almost Finished. Try to make small incremental updates to write up every day.
31/03/2022	N/A	N/A	I was unavailable but notified David.
07/04/2022	Individual Project	Progress update. Deadlines.	Approaching deadline, continue progress pace.
14/04/2022	Individual Project	Progress update Submit for review	Submit the project by the end of this week for review as David is on holiday after.

Supervisor Feedback

On 22/04/2022, I submitted a draft copy of this report to my Supervisor David Croft. The feedback has been taken into consideration and adjustments have already been made. I will provide in italics the notes he made.

So, it's a good start but the material needs to be found in this project more. Make it clear why it is relevant.

Also, think about ordering one of your materials. For example, you mention Webots a lot, but nothing is explaining what it is in or why it's being used.

For example, your () of types of ML (eg. Supervised/Unsupervised/reinforce) comes after you have looked at 2 types of supervised in detail.

Webots have been expanded upon to reflect an accurate definition of the toolbox available, how it's useful for developers and then how it will benefit my implementation. The neural network chapter in my literature review now comes before Genetic algorithms, giving the reader more knowledge on supervised and reinforcement learning, therefore, providing them with a better understanding of Genetic Algorithms.

Project Management Reflection

The project exceeded my expectations in the amount of technical knowledge and research required. Even with structured project management using the Kanban board, I struggled to maintain a work-life balance. All of the risks in my risk management did occur which bottlenecked development.

The original deadline for the project was the 18th but in February I tested positive for Covid 19. During that period, self-isolation rules were still a legal requirement therefore I had to self-isolate for 10 days. Isolation threw off both sides of the work-life balance, I wasn't able to enjoy my hobbies or meet up with friends or family, completely demoralising my attitude toward the project.

On top of that enduring, the illness of covid left me bedridden for 7 days of the isolation period and unable to focus for the other 3. This halted development altogether in what should have been the start of my implementation. Once recovering from my illness, I had to revise all of the research I had conducted and then start the practical. Luckily Coventry University extended my project deadline until the 26th of April, allowing me to readjust my schedule and begin development at a manageable pace using the Agile methodology.

The biggest obstacle in my project management was the shift from Webots to p5.js. I spent 2 weeks of research and development, setting up Webots and trying to get the car controller to work, I found a hurdle with every step of development throughout this project using Webots. Once I transitioned to p5.js, development was a lot easier to manage, in hindsight, I wish I made the transition sooner. Then I would have had extra development time to implement the Neural Networks and by extension Neuro Evolution for potentially more optimal results and more complex configurations.

7 Conclusion

Summary

This is how my application has met the objectives I set out in the introduction of this project:

- Met the defining traits specified in my literature review of an autonomous vehicle or agent using steering behaviours.
- I wasn't able to input and train a neural network but have laid the foundation for this to be implemented further.
- The genetic algorithm can create a mating pool and perform a form of heredity through the crossover to create a new generation
- I have outputted the most optimised path conducted through several simulation tests. I have explained what attributes affect which sections of performance the most and what the developer should consider when running certain configurations.
- I have explained why I couldn't implement neural networks but have provided the concepts surrounding them and how they could be implemented in the future.

Real-World Application

I've explained that this tool is originally intended for the FS AI team but at the core of this application, it's a path optimising algorithm around a pre-existing mapped area. This has many applications in industries such as Formula 1, if they wish to find the fastest race line given their race setup, this will provide further data outside of real-world testing. Formula 1 is a racing competition dating back to 1950 according to Manishin (2018).

Another application the program could be altered to suit would be public transport routes. Now the application would need additional constraints on the vehicles to take into consideration the highway code. The highway code is a framework of rules for motorists, cyclists and pedestrians regarding public roads according to Transport (2015).

Project Limitations

My implementation has limitations on how the fitness values are calculated. These biases ingrained in the source code create inaccuracies in my results by not balancing each trait equally. These biases could overlook many variations of solutions which could be a more optimal solution to the actual result.

The scale of the project is limited to the web browser IDE capabilities of p5.js. The limited number of system resources not only on my hardware but available to chrome means the simulation will not optimally perform or under intense pressure crash. This limits the number of vehicles per generation which therefore limits the variation of the mating pool. Having more vehicles per generation will add more possibilities that could lead to the optimal solution or as shown in my results more variation adds more noise, therefore, increasing the total number of required generations to find the optimal solution.

Future Research

Due to all of my risks in my risk management coming true throughout this project, my development was bottlenecked meaning a lot of the features I intended for my project didn't make it into the final version of the application. I intend to work on the following features in the future:

- A possible solution to my limitations would be to add a third DNF (Did Not Finish) trait when calculating fitness. This could hopefully negate the bias by making both time and score equal in weight but a reduction in fitness if the vehicle doesn't finish the entire track.
- Other future research would be to incorporate the concepts taught in my literature review on Neural Networks. Having a Neural Network actively deciding its steering force by altering its internal weights would allow for a larger variety of traits in the genetic algorithms mating pool.
- Adapting the proof of concept onto Webots as was originally intended for this project. This can then be used on the Coventry University FS-AI car for the competition.

Reflection

The project has only increased my passion for the subject altogether and has left me wanting to expand upon my knowledge further. Although there were a ton of hurdles during the development overcoming them has only improved my skills in research, design, implementation and project management.

The greatest tool I've learnt from this project was my use of vector math and optimisation algorithms through genetic algorithms. Taking the concepts from my literature review which appeared to be so foreign to me at the time and then implementing them in a project I started from scratch has increased my confidence as a programmer and software engineer.

The main piece of advice I would give myself at the start of this project would be to give more time to account for risk management, and work-life balance and consider work for other modules in my degree. Even with my existing development time, If I managed to be more efficient by keeping morale and confidence high, I would have stood a higher chance of implementing all my features.

8 References

Bibliography

- Agile Alliance, 2021. *The 12 Principles behind the Agile Manifesto*. [Online]
Available at: <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>
[Accessed 25 04 2022].
- Berman, B., 2021. *What's the Deal With Autonomous Racing? New Races Redefine Motorsports for the Self-Driving Era*. [Online]
Available at: <https://groundtruthautonomy.com/whats-the-deal-with-autonomous-racing-new-races-redefine-motorsports-for-the-self-driving-era/>
- British Computer Society, 2021. *CODE OF CONDUCT FOR BCS MEMBERS*, 09: 06.
- Cardamone, L. a. L. D. a. L. P. L. a. B. A. P., 2010. *Searching for the Optimal Racing Line Using Genetic Algorithms*. Copenhagen, IEEE, pp. 388-394.
- Cohen, E., 2019. *The Definitive Guide to Project Management Methodologies*. [Online]
Available at: <https://www.workamajig.com/blog/project-management-methodologies>
- Cohn, A., 2014. *How do I convert from the global coordinate space to a local space?*. [Online]
Available at: <https://gamedev.stackexchange.com/questions/79765/how-do-i-convert-from-the-global-coordinate-space-to-a-local-space>
[Accessed 24 04 2022].
- CUEMATH, 2022. *Cross Product of Two Vectors*. [Online]
Available at: <https://www.cuemath.com/geometry/cross-product/>
[Accessed 23 04 2022].
- CUEMATH, 2022. *Dot Product*. [Online]
Available at: <https://www.cuemath.com/algebra/dot-product/>
[Accessed 23 04 2022].
- cyberbotics, 2022. *Webots Reference Manual*. [Online]
Available at: <https://cyberbotics.com/doc/reference/physics>
[Accessed 23 04 2022].
- digite, 2022. *What Is Scrum Methodology? &*. [Online]
Available at: <https://www.digite.com/agile/scrum-methodology/>
- gov.uk, 1990. *Computer Misuse Act 1990*. [Online]
Available at: <https://www.legislation.gov.uk/ukpga/1990/18/contents>
- gov.uk, 2003. *Communications Act 2003*. [Online]
Available at: <https://www.legislation.gov.uk/ukpga/2003/21/contents>
- IBM Education, 2020. *Neural Networks*. [Online]
Available at: <https://www.ibm.com/cloud/learn/neural-networks>
[Accessed 24 04 2022].
- IMechE, 2022. *FORMULA STUDENT AI 2022 RULES*. s.l.:Institution of Mechanical Engineers.
- Imran, M., 2020. *Advantages of Neural Networks – Benefits of AI and Deep Learning*, s.l.: folio3.

inflectra, 2022. *What is Agile Kanban Methodology?*. [Online]
Available at: <https://www.inflectra.com/Methodologies/Kanban.aspx>

Intellectual property office, 2011. *Copyright act*. [Online]
Available at: <https://www.gov.uk/government/publications/copyright-acts-and-related-laws>

Joseph, L., 2018. *Robotics using Python*. Birmingham: Packt.

Kim, P., 2017. *MATLAB Deep Learning With Machine Learning, Neural*. s.l.:Apress.

Lentin, J., 2018. *Robotics using Python*. Birmingham: Packt.

Lumen, 2022. *The Polar Coordinate System*. [Online]
Available at: <https://courses.lumenlearning.com/boundless-algebra/chapter/the-polar-coordinate-system/>
[Accessed 24 04 2022].

Manishin, G. B., 2018. *F1 Origins*, s.l.: Flipboard.

mathisfun, 2021. *Vectors*. [Online]
Available at: <https://www.mathsisfun.com/algebra/vectors.html>
[Accessed 24 04 2022].

Mattocks, H., 2019. *TESTING IN F1: WHY THE CURRENT SYSTEM IS A JOKE!*, s.l.: drivetribe.

mercedesamgf1, 2020. *How Does F1 Simulation Work*, s.l.: AMG PETRONAS FORMULA ONE TEAM.

Reynolds, C., 1999. *Steering Behaviors For Autonomous Characters*, s.l.: s.n.

Rodrigues, D., 2021. *Individual Project Preparation*, s.l.: Coventry University.

Samuel Arzt, 2019. *AI Learns to Park - Deep Reinforcement Learning*, s.l.: Youtube.

scholbak, J., 2009. *label of the vector*, s.l.: wikimedia.

Shiffman, D., 2012. *THE NATURE OF CODE*. s.l.:Free Software Foundation.

Sivanandam, 2007. *Introduction to Genetic Algorithms*. s.l.:Springer.

Tattersall, R., 2020. *INVESTIGATION INTO THE CREATION OF RACING LINES IN A SIMULATOR*, s.l.: Coventry University.

Transport, D. f., 2015. *The Highway Code*, s.l.: Gov.UK.

Zinka, 2022. *Two dimensional Cartesian coordinate system*. [Art].

9 Appendices

Project Proposal (6000 CEM)

Investigation into the application of Genetic Algorithms on Simulated Automated Vehicles

Research Question

Can Genetic Algorithms generate an optimal race line to improve the lap time for a simulated autonomous vehicle?

Project Topic

Formula Student Autonomous (FS AI) is a university challenge where teams must construct an autonomous racing car that must navigate around multiple Silverstone tracks without any manual input. The winners of this competition are the team whose car manages to get around the track consistently and with the fastest lap time. Multiple factors play into this from speed to traction but most importantly the chosen race line. The racing line is the shortest possible route around a track. According to (Cardamone, 2010) in racing games, the trajectories are pre-calculated manually by experts, this is extremely time-consuming and not allowed in the competition.

Multiple variations of the race line exist such as the minimum curvature or the shortest path, but neither is usually the optimal path (Cardamone, 2010). The optimal race line is very circumstantial to the constraints of the track and the vehicle that's being worked with. Therefore, it's crucial to have large amounts of testing data before and during the race to form a basis for producing the race line. This is where Genetic Algorithms take place.

Genetic Algorithms are based on Darwin's theory of evolution from natural selection. It's an iterative deep learning algorithm that has each generation further adapting to its environment, therefore, optimising the solution. The most adapted individuals to their respective environments reproduce more often, therefore, having a larger number of offspring that maintain their genes. With each generation, individuals must further compete to reproduce, resulting in more adapted gene pools than the prior generation.

A Fitness function is assigned to express the performance of how adapted the individual is to the objective. This score is a valid metric that can range from speed to accuracy, which is used to rank the chromosome based on its accuracy. Using this score, we form a mating pool of the top performers, then select the candidates based on probability, the higher the candidate's fitness score, the higher the chance they will be selected. This ensures not only the top two candidates get selected each generation, therefore, promoting variation in traits introduced. It is important that the fitness requirements for the mating pool are fair, too high and the variation and reproduction process will be too slow, but too low then the solution will be sub-optimal (Sivanandam, 2007).

Neural Networks are an important algorithm in Deep Learning based on the human mind. This system is about the complicated relationship between neurons which act as nodes and are then divided into an input layer, hidden layer and output layer, each with different responsibilities and handling the data from one another. Between each node, there is a connection with a weight that dictates how much influence the input has over the output node and a bias that determines how far the output of the function is from the prediction.

NeuroEvolution is the process of applying genetic algorithms to a candidate pool of neural networks. Genetic Algorithms assist Neural Networks in finding the best weights and biases by reproducing the Neural Network Candidates with the best features.

Simulations are used to represent a real-world environment in a virtual setting. It allows for scenarios to be tested to extract data on what would happen when played out. This allows for testing, analysis and feature development. It is a commonly used technique in sports such as F1.

Summary

Automated cars run on a large magnitude of machine learning algorithms, they're essential in their awareness of their surroundings, developing scenarios, therefore, allowing them to understand what protocols need to be taken. To train these algorithms for better accuracy, large amounts of relevant data are also required.

Simulations allow for various scenarios to be taken and calculate a large amount of directional data from sensors. NeuroEvolution in theory can take the data to form predictive outputs on what vectors the cars should take when detecting the track wall. Having hundreds of candidates in parallel experiencing different scenarios will allow for each generation to continuously adapt features that optimise the race line.

FS AI currently has a functioning race line method that uses a triangular system to form a central path throughout the cones of the track. The system works but is severely limited in how it's more reactive than proactive, it doesn't maintain the discovered path in its memory, therefore, requiring urgent twists and turns when the vehicle discovers a corner, creating an extremely inefficient race line. Neuroevolutionary algorithms set the weights of the candidates at random upon the first generation but Genetic Algorithms are extremely adaptable, they can start from scratch or evolve off this existing pathing system, therefore potentially increasing the speed of the optimisation significantly.

Motivation and Outputs

Like Formula 1, teams in FS AI don't get much testing on the official track until the event occurs (Mattocks, 2019). This additionally proves to be a problem, from a development standpoint for logical errors. Trial and error are fundamental in understanding the current state of the system and what issues are present. Without access to performance data to see what sort of consistency or lap times the vehicle will provide; brainstorming new features is entirely theoretical. Therefore, Genetic Algorithms and Simulations is a useful solution for the FS AI team.

Using the FS AI Team's Silverstone simulation, I want to apply a NeuroEvolution algorithm using a fitness function of a reduced lap time. The candidates each generation will race around the track, constantly reproducing and improving its consistency and speed. The result will be the optimal race line from the leading candidate which will be compared to the baseline system, with the expectation of showing a reduced lap time. The genetic algorithm must run to completion within the time it takes for the car to complete 1 lap on the existing race line. This will be iterated multiple times with each having a different selection pressure to compare results.

Primary Research Plan

Implementation

The simulation will be what is used to facilitate my genetic algorithm, populated with neural networks. As stated, before I will be using the FS AI Silverstone simulation as it contains an accurate model of the car we will be using come to the event in July, this will assist in keeping the test as accurate as possible. The system uses an application called WeBots, the system takes Python and C++, I'll be using C++ as it performs better due to the compiler and memory management features.

My genetic algorithm will look at the multiple ways of conducting reproduction such as mutation and crossover, I will need to decide what sort of selection pressure will produce the most optimal results. I will attempt to use the current triangular system for the FS AI as an evolutionary foundation to see if it can assist with my performance constraints.

The Neural Networks will take in-depth data from the sensors to determine whether the brakes or acceleration should be applied on each wheel. Various libraries exist for neural networks, but the FS AI team seem to be interested in Tensor Flow therefore I intend to work with that. From this, the genetic algorithms will either take this data for the genes as values for the intensity of these actions or just the vectors of the direction. Both are plausible approaches that I will only conclude as the most functional in development.

Time management

For effective time management, I've constructed a timeline to be used as a framework to complete my designated tasks within. This is used to assist me in balancing my workload across my time. It's unrealistic to expect me to work on this project every day of the week therefore between major accomplishments, I've given myself breathing periods as well as long coding development times so I don't need to rush. The main bottlenecks that could potentially occur would be getting feedback, supervisors have other pupils to attend to so they certainly will not be able to provide feedback immediately when I request it therefore I've given dedicated time for the feedback to come through.

Task	Start Date	End Date	Duration
Research - 1.0.1.1	11/11/19	11/11/19	1d
Research - 1.0.1.2	11/11/19	11/11/19	1d
Research - 1.0.1.3	11/11/19	11/11/19	1d
Research - 1.0.1.4	11/11/19	11/11/19	1d
Research - 1.0.1.5	11/11/19	11/11/19	1d
Research - 1.0.1.6	11/11/19	11/11/19	1d
Research - 1.0.1.7	11/11/19	11/11/19	1d
Research - 1.0.1.8	11/11/19	11/11/19	1d
Research - 1.0.1.9	11/11/19	11/11/19	1d
Research - 1.0.1.10	11/11/19	11/11/19	1d
Research - 1.0.1.11	11/11/19	11/11/19	1d
Research - 1.0.1.12	11/11/19	11/11/19	1d
Research - 1.0.1.13	11/11/19	11/11/19	1d
Research - 1.0.1.14	11/11/19	11/11/19	1d
Research - 1.0.1.15	11/11/19	11/11/19	1d
Research - 1.0.1.16	11/11/19	11/11/19	1d
Research - 1.0.1.17	11/11/19	11/11/19	1d
Research - 1.0.1.18	11/11/19	11/11/19	1d
Research - 1.0.1.19	11/11/19	11/11/19	1d
Research - 1.0.1.20	11/11/19	11/11/19	1d
Research - 1.0.1.21	11/11/19	11/11/19	1d
Research - 1.0.1.22	11/11/19	11/11/19	1d
Research - 1.0.1.23	11/11/19	11/11/19	1d
Research - 1.0.1.24	11/11/19	11/11/19	1d
Research - 1.0.1.25	11/11/19	11/11/19	1d
Research - 1.0.1.26	11/11/19	11/11/19	1d
Research - 1.0.1.27	11/11/19	11/11/19	1d
Research - 1.0.1.28	11/11/19	11/11/19	1d
Research - 1.0.1.29	11/11/19	11/11/19	1d
Research - 1.0.1.30	11/11/19	11/11/19	1d
Research - 1.0.1.31	11/11/19	11/11/19	1d
Research - 1.0.1.32	11/11/19	11/11/19	1d
Research - 1.0.1.33	11/11/19	11/11/19	1d
Research - 1.0.1.34	11/11/19	11/11/19	1d
Research - 1.0.1.35	11/11/19	11/11/19	1d
Research - 1.0.1.36	11/11/19	11/11/19	1d
Research - 1.0.1.37	11/11/19	11/11/19	1d
Research - 1.0.1.38	11/11/19	11/11/19	1d
Research - 1.0.1.39	11/11/19	11/11/19	1d
Research - 1.0.1.40	11/11/19	11/11/19	1d
Research - 1.0.1.41	11/11/19	11/11/19	1d
Research - 1.0.1.42	11/11/19	11/11/19	1d
Research - 1.0.1.43	11/11/19	11/11/19	1d
Research - 1.0.1.44	11/11/19	11/11/19	1d
Research - 1.0.1.45	11/11/19	11/11/19	1d
Research - 1.0.1.46	11/11/19	11/11/19	1d
Research - 1.0.1.47	11/11/19	11/11/19	1d
Research - 1.0.1.48	11/11/19	11/11/19	1d
Research - 1.0.1.49	11/11/19	11/11/19	1d
Research - 1.0.1.50	11/11/19	11/11/19	1d
Research - 1.0.1.51	11/11/19	11/11/19	1d
Research - 1.0.1.52	11/11/19	11/11/19	1d
Research - 1.0.1.53	11/11/19	11/11/19	1d
Research - 1.0.1.54	11/11/19	11/11/19	1d
Research - 1.0.1.55	11/11/19	11/11/19	1d
Research - 1.0.1.56	11/11/19	11/11/19	1d
Research - 1.0.1.57	11/11/19	11/11/19	1d
Research - 1.0.1.58	11/11/19	11/11/19	1d
Research - 1.0.1.59	11/11/19	11/11/19	1d
Research - 1.0.1.60	11/11/19	11/11/19	1d
Research - 1.0.1.61	11/11/19	11/11/19	1d
Research - 1.0.1.62	11/11/19	11/11/19	1d
Research - 1.0.1.63	11/11/19	11/11/19	1d
Research - 1.0.1.64	11/11/19	11/11/19	1d
Research - 1.0.1.65	11/11/19	11/11/19	1d
Research - 1.0.1.66	11/11/19	11/11/19	1d
Research - 1.0.1.67	11/11/19	11/11/19	1d
Research - 1.0.1.68	11/11/19	11/11/19	1d
Research - 1.0.1.69	11/11/19	11/11/19	1d
Research - 1.0.1.70	11/11/19	11/11/19	1d
Research - 1.0.1.71	11/11/19	11/11/19	1d
Research - 1.0.1.72	11/11/19	11/11/19	1d
Research - 1.0.1.73	11/11/19	11/11/19	1d
Research - 1.0.1.74	11/11/19	11/11/19	1d
Research - 1.0.1.75	11/11/19	11/11/19	1d
Research - 1.0.1.76	11/11/19	11/11/19	1d
Research - 1.0.1.77	11/11/19	11/11/19	1d
Research - 1.0.1.78	11/11/19	11/11/19	1d
Research - 1.0.1.79	11/11/19	11/11/19	1d
Research - 1.0.1.80	11/11/19	11/11/19	1d
Research - 1.0.1.81	11/11/19	11/11/19	1d
Research - 1.0.1.82	11/11/19	11/11/19	1d
Research - 1.0.1.83	11/11/19	11/11/19	1d
Research - 1.0.1.84	11/11/19	11/11/19	1d
Research - 1.0.1.85	11/11/19	11/11/19	1d
Research - 1.0.1.86	11/11/19	11/11/19	1d
Research - 1.0.1.87	11/11/19	11/11/19	1d
Research - 1.0.1.88	11/11/19	11/11/19	1d
Research - 1.0.1.89	11/11/19	11/11/19	1d
Research - 1.0.1.90	11/11/19	11/11/19	1d
Research - 1.0.1.91	11/11/19	11/11/19	1d
Research - 1.0.1.92	11/11/19	11/11/19	1d
Research - 1.0.1.93	11/11/19	11/11/19	1d
Research - 1.0.1.94	11/11/19	11/11/19	1d
Research - 1.0.1.95	11/11/19	11/11/19	1d
Research - 1.0.1.96	11/11/19	11/11/19	1d
Research - 1.0.1.97	11/11/19	11/11/19	1d
Research - 1.0.1.98	11/11/19	11/11/19	1d
Research - 1.0.1.99	11/11/19	11/11/19	1d
Research - 1.0.1.100	11/11/19	11/11/19	1d

Constraints

My project is limited by the goal of having my simulation run within 1 lap of the race of the team's current baseline system. This severely limits the number of generations my genetic algorithm can undertake therefore limiting how advanced my neural networks and by extension, race lines can be. This will need to be solved by investigating parallel processing and multi-threading techniques that can be applied to my genetic algorithms to make the most of my hardware.

Literature Review

Paper 1

(Tattersall, 2020) This was a paper conducted by another student at Coventry University. Similarly, to mine it goes on to explain the importance of the race line in racing, explaining the various concepts such as the latest apex.

This paper is about comparing various methods in deep learning that can be used to optimise the race line. The author explains the various machine learning techniques and how reinforcement learning is a potential valid path using the Markov decision processes in training his system. They explain the concepts of genetic algorithms but never really goes into any incredible depth on how they could be implemented or even an outline of where to get started.

Like my vision, the author uses WeBots as his primary application in carrying out his algorithm, it's an incredibly versatile piece of software. However, it is clear from their implementation that the author started from scratch. This differentiates mine as I want my project to be implemented in the future on the FS AI car competition, I need an accurate virtual representation of Silverstone and a model of the car and its relevant hardware being used. The author declares that his project could be useful for the team but is entirely independent of them.

Overall, It is clear the author took a different approach in his investigation, he starts with a wide variety of techniques, but his implementation seems to focus on the path of reinforcement learning. He stated that he intended to get around to genetic algorithms, mentioning it as a viable option but unfortunately, he ran out of time. He goes on to state that he wishes for someone else to continue his vision, which highlights that there is a gap in the research that my project can build from.

Paper 2

(Cardamone, 2010) This was a paper that carries out a genetic algorithmic approach to optimising the race line. Again, like my vision, it states what the racing line is and goes into a lot more depth surrounding the concepts and equations for the optimal approach to corners than the prior paper, this gives the reader a more thorough understanding of the logic behind their implementation.

The author explains how genetic algorithms can benefit the race line and applies it to various tracks such as Forza. He records the performance results in seconds and compares them to traditional techniques of calculating the race line such as the Minimum curvature path (MCP). This is a similar approach to my vision with the implementation of genetic algorithms and provides valid results that can optimise the race line with the comparisons the author has set out to make.

Where my method differentiates is the scope of the project. They use genetic algorithms to tune the parameters on the calculation of the MCP until it finds the best results. My project is trying to implement using a combination of genetic algorithms and machine learning using neural networks, It asks the question of whether this approach will provide significantly better results as the generations have gone on and the neural networks learn. The circuits I want my solution to be implemented on are entirely different to those the author has chosen with mine being Silverstone. With the performance constraint of wanting the simulation to run within 1 lap of the existing race line, I'm going to need to consider different methods of increasing the runtime speed. He also uses different applications by using TORCS. So, although our methods overlap, we both have our visions, and constraints and the contexts in how our methods are implemented are entirely different.

Paper 3

(Shiffman, 2012) This book provides a large background on the concepts of processing data. Most of the chapters were largely irrelevant to what I was intending to accomplish, but the last two chapters highlight key concepts of how I could go around implementing my project.

The last two chapters gave thorough research and explanation of evolutionary programming which is the category into which genetic algorithms fit. It goes into the 3 main steps of what makes up a genetic algorithm and what constraints are needed to be aware of such as a variation of traits. This book highlights the importance of introducing unique traits. If each giraffe in a field has the same size neck and legs, therefore, can reach the same tree for food then there aren't any unique traits that allow evolutionary jumps to be made, everyone's the same! It also introduced to me that genetic algorithms don't construct genetic information out of binary data in arrays but also vectors which for a racing simulation is crucial

What this book did best was using terms other research papers never used. It consciously made an extra effort to apply biological metaphors to explain the programming concepts, such as explaining the data within a feature as a genotype and then the expression of that as phenotypes. These metaphors made understanding the paper from a reader's perspective a much easier task and a kind of writing I will hope to apply to my dissertation.

The paper also introduced me to neural networks which got me thinking about how both chapters could work together to achieve better results by using genetic algorithms to tune the parameters of a neural network. Later I found out this was already an established concept called NeuroEvolution. This solidified for me the scope of how I would go about implementing my idea for the simulation.

Paper 4

(Joseph, 2018) This paper goes into the theory of robotics, what they are, how they operate and what implementation methods exist. I learned a lot from this paper, it explains what the ROS framework is, and its most common features used in creating robotics applications. One of the most useful benefits I took from the paper that could be implemented into my project would be the Third-party integration of functionality from Webots, Integrated testing frameworks such as Rotest which should point out any potential or present flaws in my algorithms and language independence, allowing me to create a node in my intended C++ for better performance with the compiler. Although I may not be integrating ROS into my project, I will be researching the potential use for ROS in the future, whether it be in my projector for the implementation of my genetic algorithms into the FS AI Car that uses ROS.

The paper also goes into a thorough explanation of the kinematics of robotics. It provided useful information and equations on how a robot is put into motion which will be carried forward into my simulation. These motions will potentially be the data for my input layers in my neural networks that will help it decide whether to accelerate or brake on which axis.

Overall, this paper explained a lot of the ROS frameworks, the rich toolset it contains then the motion of robotics. This information will be useful for my project, especially in the car controller for my simulation, having a good understanding of kinematics will help me understand the limitations of the race line with physics. The author continues to explain a lot of information largely irrelevant to my project such as the circuitry and creating a GUI. The author also never really goes into the concepts of kinematics when applied to neural networks or genetic algorithms, but they have given me enough foundational information to develop a solution for my project.

Conclusion

This literature review has provided me with the chance to conduct thorough research surrounding the topics of genetic algorithms, neural networks, kinematics and simulation which are the relevant tools I need to optimise the race line. These articles explained thoroughly enough conceptual knowledge to conduct more research over Christmas and start developing my project in my second semester.

Bibliography

- Agile Alliance, 2021. *The 12 Principles behind the Agile Manifesto*. [Online]
Available at: <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>
[Accessed 25 04 2022].
- Berman, B., 2021. *What's the Deal With Autonomous Racing? New Races Redefine Motorsports for the Self-Driving Era*. [Online]
Available at: <https://groundtruthautonomy.com/whats-the-deal-with-autonomous-racing-new-races-redefine-motorsports-for-the-self-driving-era/>
- British Computer Society, 2021. *CODE OF CONDUCT FOR BCS MEMBERS*, 09: 06.
- Cardamone, L. a. L. D. a. L. P. L. a. B. A. P., 2010. *Searching for the Optimal Racing Line Using Genetic Algorithms*. Copenhagen, IEEE, pp. 388-394.
- Cohen, E., 2019. *The Definitive Guide to Project Management Methodologies*. [Online]
Available at: <https://www.workamajig.com/blog/project-management-methodologies>
- Cohn, A., 2014. *How do I convert from the global coordinate space to a local space?*. [Online]
Available at: <https://gamedev.stackexchange.com/questions/79765/how-do-i-convert-from-the-global-coordinate-space-to-a-local-space>
[Accessed 24 04 2022].
- CUEMATH, 2022. *Cross Product of Two Vectors*. [Online]
Available at: <https://www.cuemath.com/geometry/cross-product/>
[Accessed 23 04 2022].
- CUEMATH, 2022. *Dot Product*. [Online]
Available at: <https://www.cuemath.com/algebra/dot-product/>
[Accessed 23 04 2022].
- cyberbotics, 2022. *Webots Reference Manual*. [Online]
Available at: <https://cyberbotics.com/doc/reference/physics>
[Accessed 23 04 2022].
- digite, 2022. *What Is Scrum Methodology? &*. [Online]
Available at: <https://www.digite.com/agile/scrum-methodology/>
- gov.uk, 1990. *Computer Misuse Act 1990*. [Online]
Available at: <https://www.legislation.gov.uk/ukpga/1990/18/contents>
- gov.uk, 2003. *Communications Act 2003*. [Online]
Available at: <https://www.legislation.gov.uk/ukpga/2003/21/contents>
- IBM Education, 2020. *Neural Networks*. [Online]
Available at: <https://www.ibm.com/cloud/learn/neural-networks>
[Accessed 24 04 2022].
- IMechE, 2022. *FORMULA STUDENT AI 2022 RULES*. s.l.:Institution of Mechanical Engineers.
- Imran, M., 2020. *Advantages of Neural Networks – Benefits of AI and Deep Learning*, s.l.: folio3.
- inflectra, 2022. *What is Agile Kanban Methodology?*. [Online]
Available at: <https://www.inflectra.com/Methodologies/Kanban.aspx>

Intellectual property office, 2011. *Copyright act*. [Online]
Available at: <https://www.gov.uk/government/publications/copyright-acts-and-related-laws>

Joseph, L., 2018. *Robotics using Python*. Birmingham: Packt.

Kim, P., 2017. *MATLAB Deep Learning With Machine Learning, Neural*. s.l.:Apress.

Lentin, J., 2018. *Robotics using Python*. Birmingham: Packt.

Lumen, 2022. *The Polar Coordinate System*. [Online]
Available at: <https://courses.lumenlearning.com/boundless-algebra/chapter/the-polar-coordinate-system/>
[Accessed 24 04 2022].

Manishin, G. B., 2018. *F1 Origins*, s.l.: Flipboard.

mathisfun, 2021. *Vectors*. [Online]
Available at: <https://www.mathisfun.com/algebra/vectors.html>
[Accessed 24 04 2022].

Mattocks, H., 2019. *TESTING IN F1: WHY THE CURRENT SYSTEM IS A JOKE!*, s.l.: drivetribe.

mercedesamgf1, 2020. *How Does F1 Simulation Work*, s.l.: AMG PETRONAS FORMULA ONE TEAM.

Reynolds, C., 1999. *Steering Behaviors For Autonomous Characters*, s.l.: s.n.

Rodrigues, D., 2021. *Individual Project Preparation*, s.l.: Coventry University.

Samuel Arzt, 2019. *AI Learns to Park - Deep Reinforcement Learning*, s.l.: Youtube.

scholbak, J., 2009. *label of the vector*, s.l.: wikimedia.

Shiffman, D., 2012. *THE NATURE OF CODE*. s.l.:Free Software Foundation.

Sivanandam, 2007. *Introduction to Genetic Algorithms*. s.l.:Springer.

Tattersall, R., 2020. *INVESTIGATION INTO THE CREATION OF RACING LINES IN A SIMULATOR*, s.l.: Coventry University.

Transport, D. f., 2015. *The Highway Code*, s.l.: Gov.UK.

Zinka, 2022. *Two dimensional Cartesian coordinate system*. [Art].

List of figures

Figure 1 – Polar Co-ordinates (Lumen, 2022)	Figure 2 – Cartesian Co-ordinates (Zinka, 2022).....	8
Figure 3 - Vector diagram (scholbak, 2009)	Figure 4 – Traits of a vector (mathisfun, 2021).....	8
Figure 5 – Steering force diagram (Shiffman, 2012)		11
Figure 6 – Vector multiplication diagram (Shiffman, 2012).....		11
Figure 7 – Vector projection diagram (CUEMATH, 2022)		12
Figure 8 – Neural network diagram (IBM Education, 2020)		14
Figure 9 – perceptron diagram (Shiffman, 2012).....		14
Figure 10 – supervised learning diagram (Kim, 2017)		15
Figure 11 – choosing parent depending on probability (Shiffman, 2012)		16

Figure 12 – Crossover by mixing two elements from each array (Shiffman, 2012).....	17
Figure 13 – mutating an array (Shiffman, 2012)	17
Figure 14 – Deciding which target to pursue (Shiffman, 2012)	19
Figure 15 – local & global space (Cohn, 2014)	21
Figure 16 – GitHub repo.....	22
Figure 17 – Simulated Track.....	23
Figure 18 Figure 19	23
Figure 20 – Vehicle class Figure 21 – update function	24
Figure 22 – Tracking code	25
Figure 23 – Vector projection code.....	25
Figure 24 – Single vehicle.....	25
Figure 25 – If the line is off the path, steer backthe to path (Shiffman, 2012).....	25
Figure 26 - Vector projection (Shiffman, 2012)	26
Figure 27 – Path deciding (Shiffman, 2012)	26
Figure 28 – Example run of my application	27
Figure 29 – Point posts around the track.....	27
Figure 30	28
Figure 31 – Heredity flow chart in my implementation Figure 32 – Crossover code.....	28
Figure 33 – Mutation code.....	29
Figure 34 – Initialising generations	29
Figure 35	30
Figure 36	30
Figure 37	31
Figure 38	31
Figure 39	32
Figure 40	32
Figure 41	33
Figure 42	33
Figure 43	34
Figure 44	34
Figure 45	35
Figure 46	35
Figure 47	36
Figure 48	36
Figure 49 Figure 50 Figure 51	40
Figure 52 Figure 53 Figure 54	40
Figure 55 Figure 56 Figure 57	41
Figure 58 Figure 59 Figure 60.....	41
Figure 61 – Kanban board	42

List of tables

Table 1 - Results	37
Table 2 – Risk management	44
Table 3 – Meeting Records	44