BSc (Hons) Computer Game Applications Development

CMP 304: ARTIFICIAL INTELLIGENCE

Joseph Roper
UNIVERSITY OF ABERTAY

Contents

Introduction

This assignment will compare a finite state machine against a fuzzy inference system to see which is more effective for the task of a car following a road. The red car is operating under the finite state machine and the blue car is controlled by the fuzzy engine. The aim is to test which AI (Artificial Intelligence) is more appropriate to be integrated into a racing game, based off the car's behaviours and efficiency.

Within the report and program, Finite State Machine will be referred to as FSM and Fuzzy Logic State Machine as FLSM

Methodology

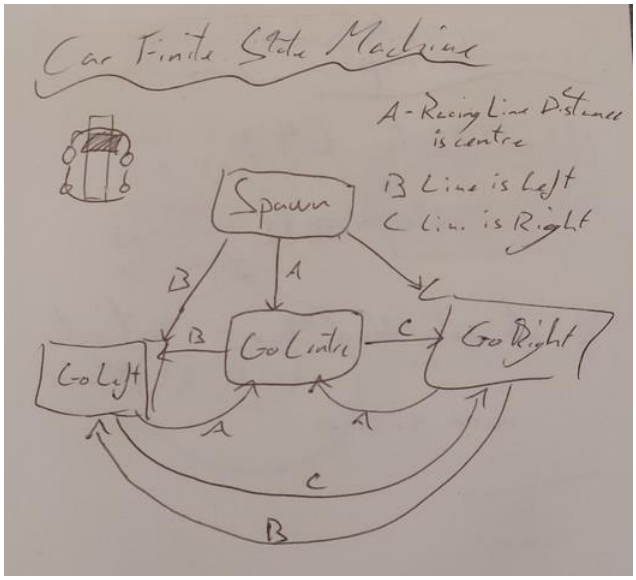
Finite State Machine

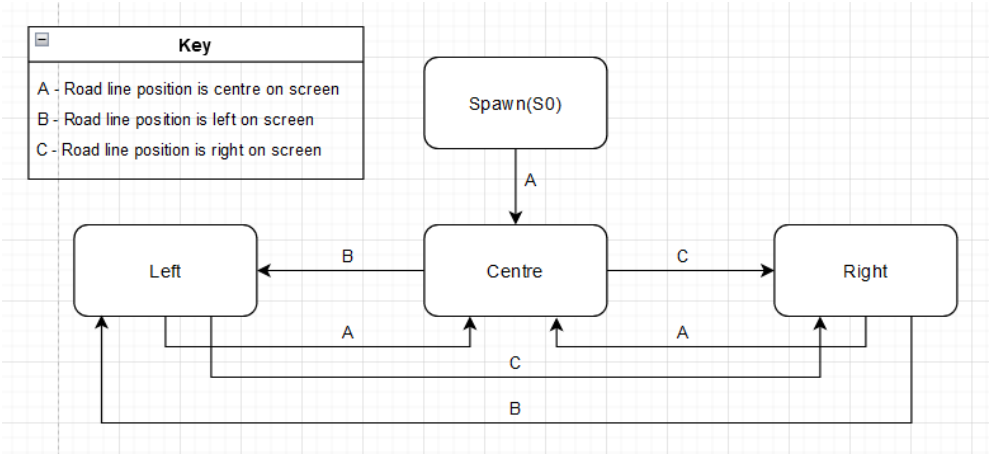
A finite state machine is a simple modelling structure to map defined states. A state diagram and subsequent implementation into code will be guided by pre-existing ideas on how the car and driver should behave.

I decided to use an FSM as one of my AI techniques, due to my program having a simple structure, a car moving left and right, it fit the functionality of a finite state machine.

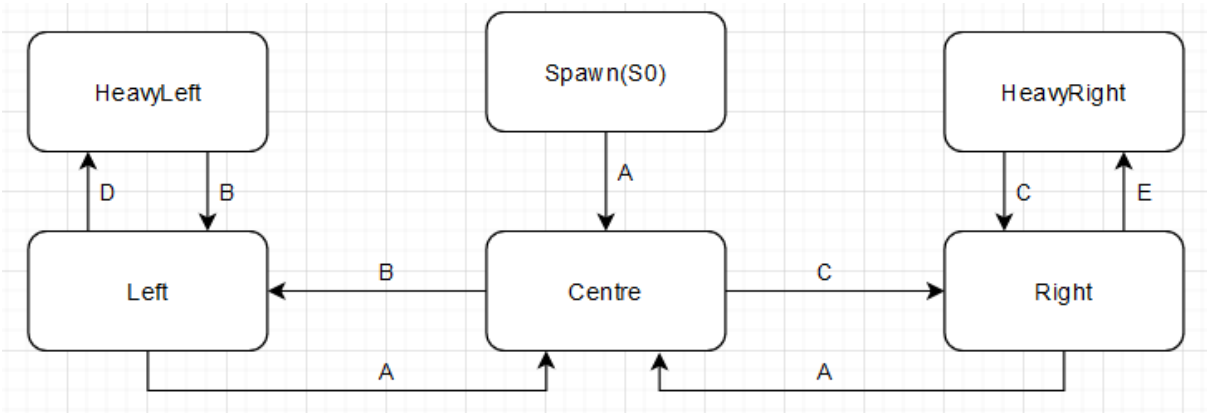
As the program has a simple structure of a car moving from left to right, this fit the functionality of a FSM.

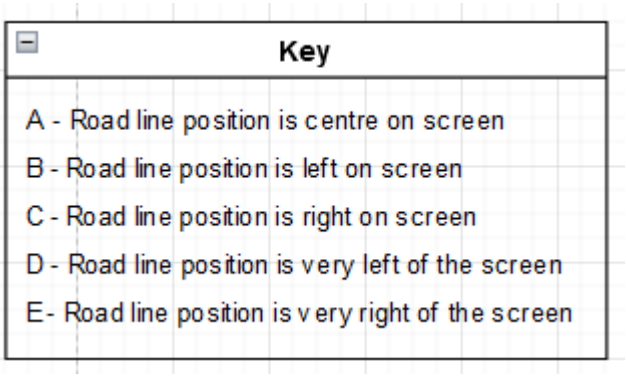
The state diagram was first planned on paper and then moved to draw.io, a useful website for designing diagrams.





When testing the first diagram within code, the functionality of the fuzzy system was limited by only having three states, so the two extra states 'HeavyLeft' and 'HeavyRight' were added to the FSM as well, due to both machines having to share the same states across both machines to test fairly.



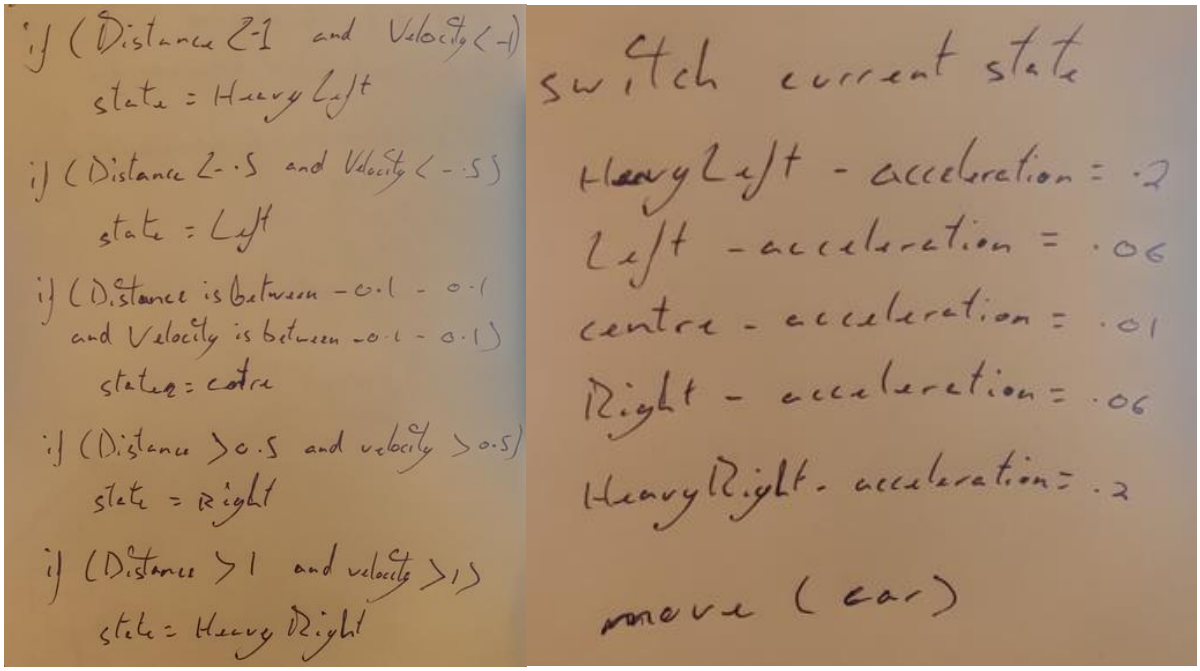


From the diagram, it can be inferred that the core of the logic is if the car is not on top of the road line, it must follow it.

As shown above, the FSM diagram is quite simple, however as it is to be tested against a FLSM the functionality must also be the same. In the future obstacle avoidance could be added for better comparison between the FSM and FLSM.

To implement the FSM diagram into code, The car's position was subtracted from the road's position to calculate the car's distance from the line. This variable was then used to calculate the velocity in which to move the car.

The velocity and distance from the line was then used within 'if' statements to set the state of the car, as shown in the pseudocode.



A switch statement is then used to move the car based on the state selected from the switch statement. Each state changes the acceleration variable to a different value, so that if the car is further away it will move to the road quickly as compared to being slightly off centre of the road. I believe this adds realism to the state machine when operating within the game. The car is moved based on the velocity, acceleration and speed being multiplied together.

Fuzzy Logic State Machine

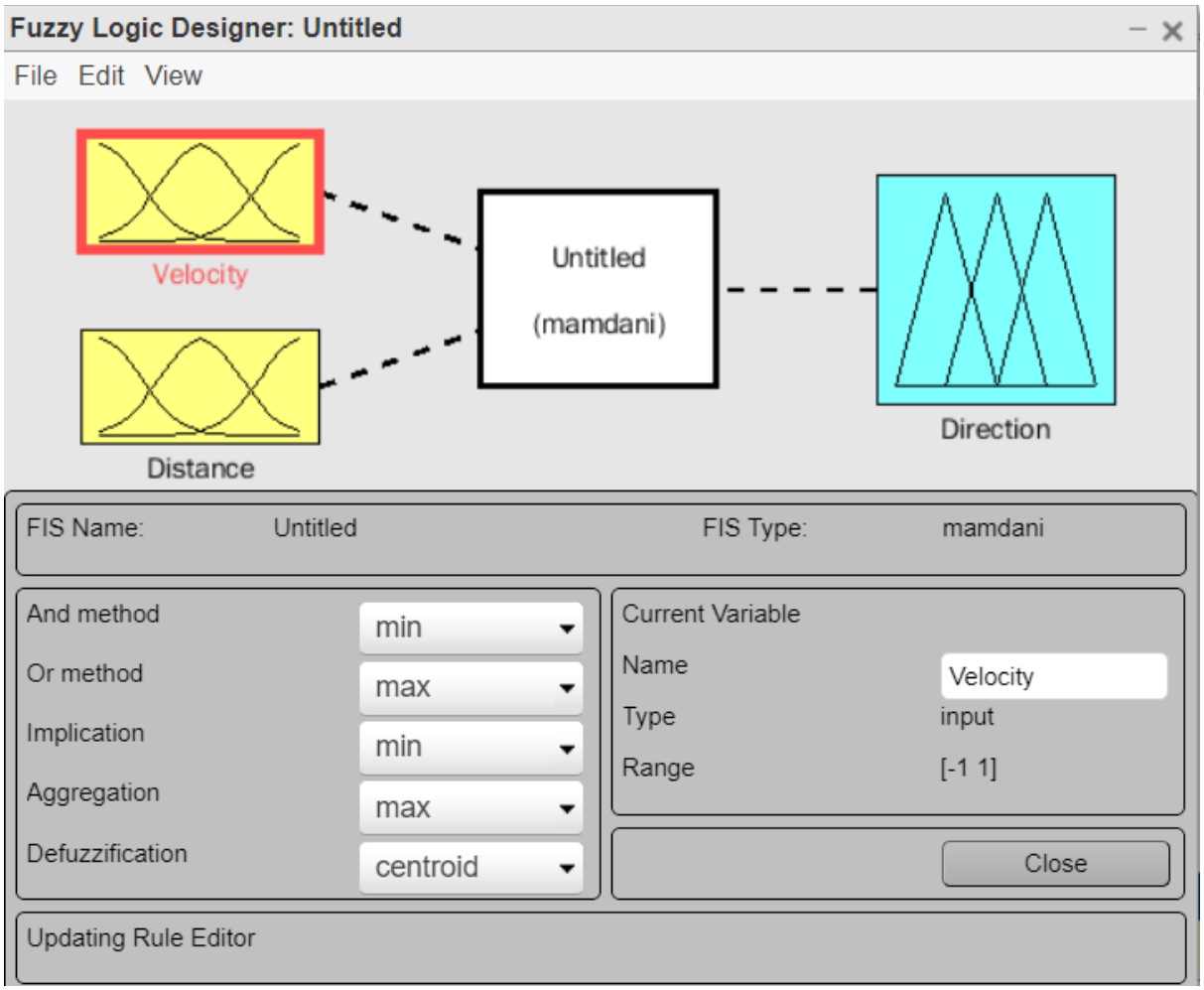
Fuzzy logic allows for multiple possible values for a variable, as opposed to a Boolean which is either true or false. For example, describing the temperature, Boolean would only allow for hot or cold, whereas fuzzy logic would say it is mostly hot but a little cold.

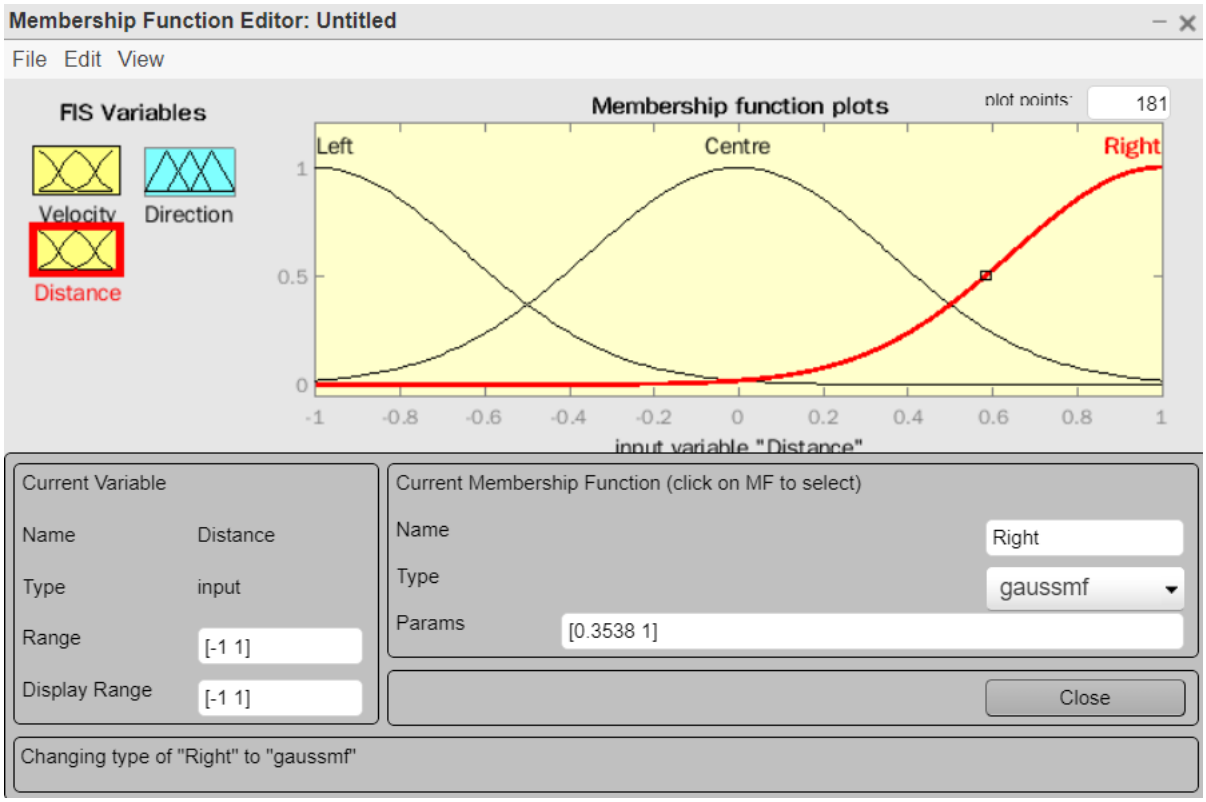
As the FLSM is an extension of an FSM, the results of the program determine the efficiency of fuzzy logic compared to a normal FSM

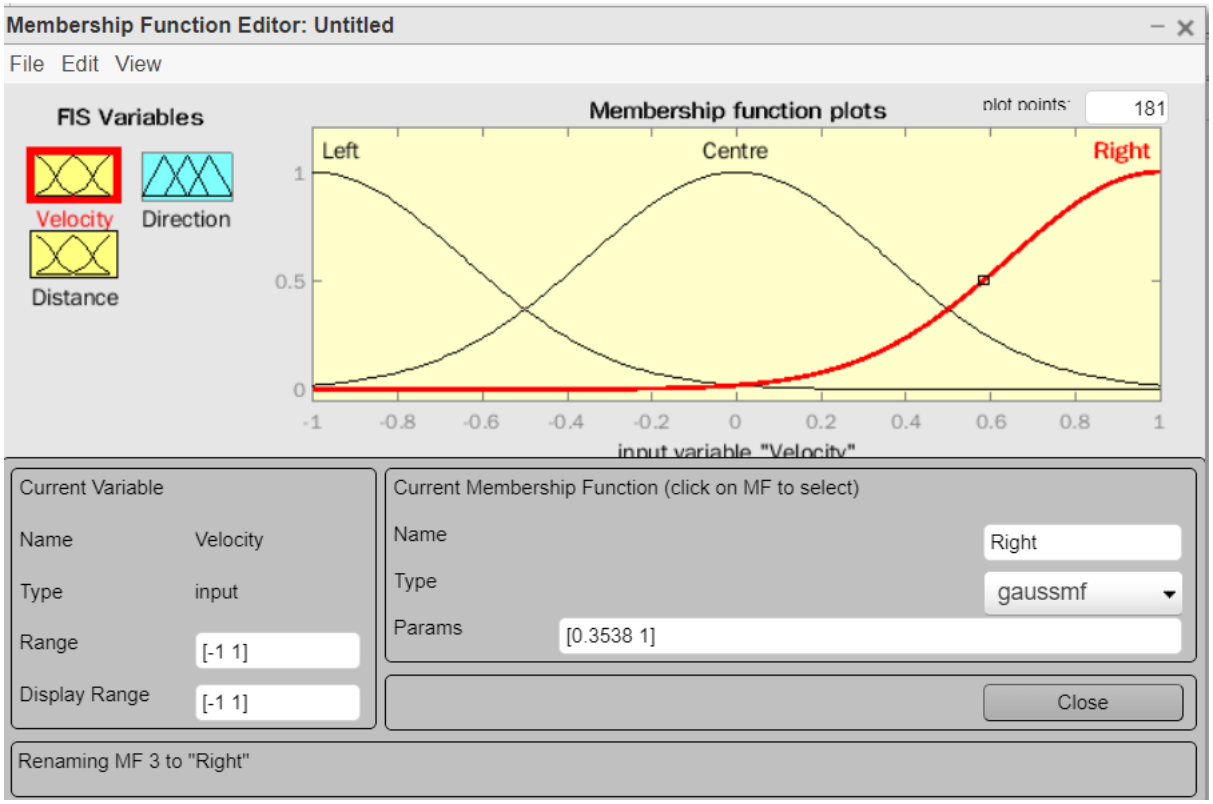
Two inputs were created in the fuzzy inference system, velocity and distance of the car from the road, which were also used in the finite state machine. Velocity will be used to adjust the speed of the car based off the distance away from the road line, if the car is further away it should accelerate towards the line. The goal of the system is to calculate the direction the car should move in based off the input variables.

Matlab was used to create the fuzzy inference system; Matlab is a tool that helps the creation process for mathematical systems and analysing data.

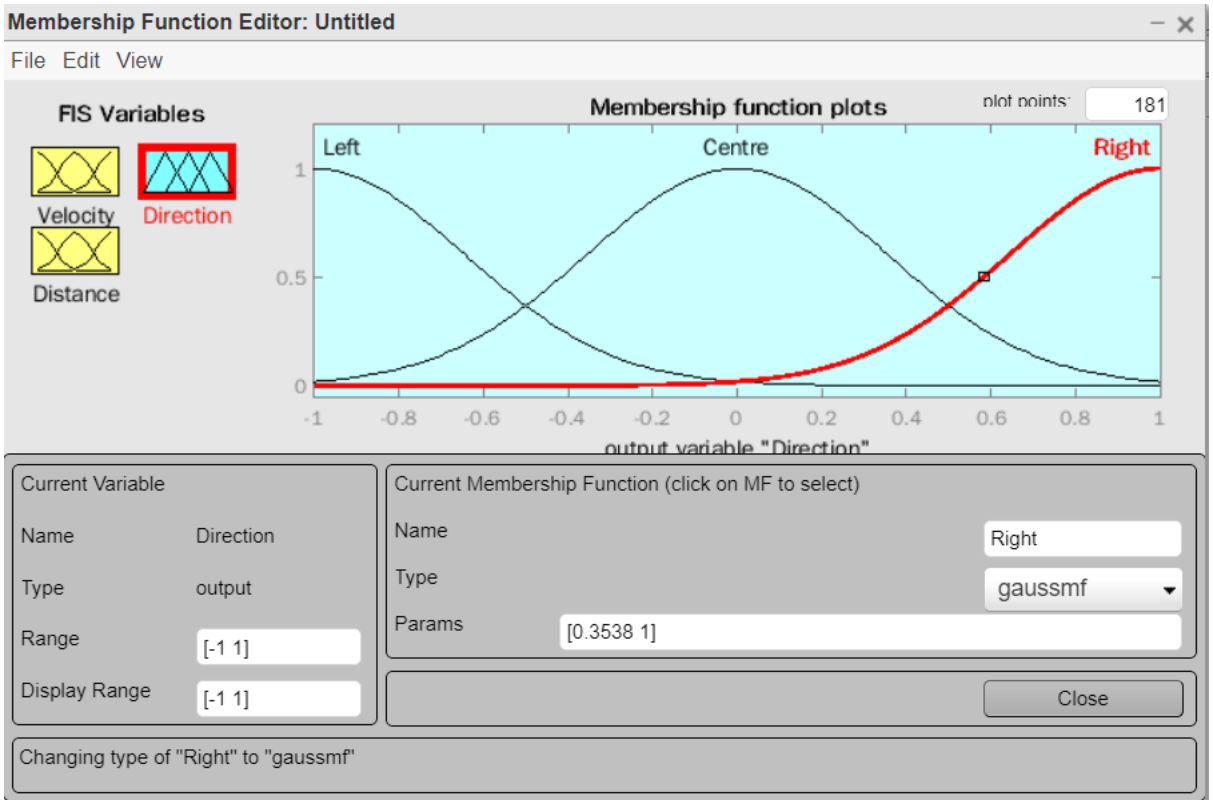
As discussed, the two inputs are velocity and distance, and the output is direction. These were added to the Matlab interface. 3 states were added to the velocity and direction inputs, left, centre and right. The system will calculate the value of the input based off the centroid curve of the graph.



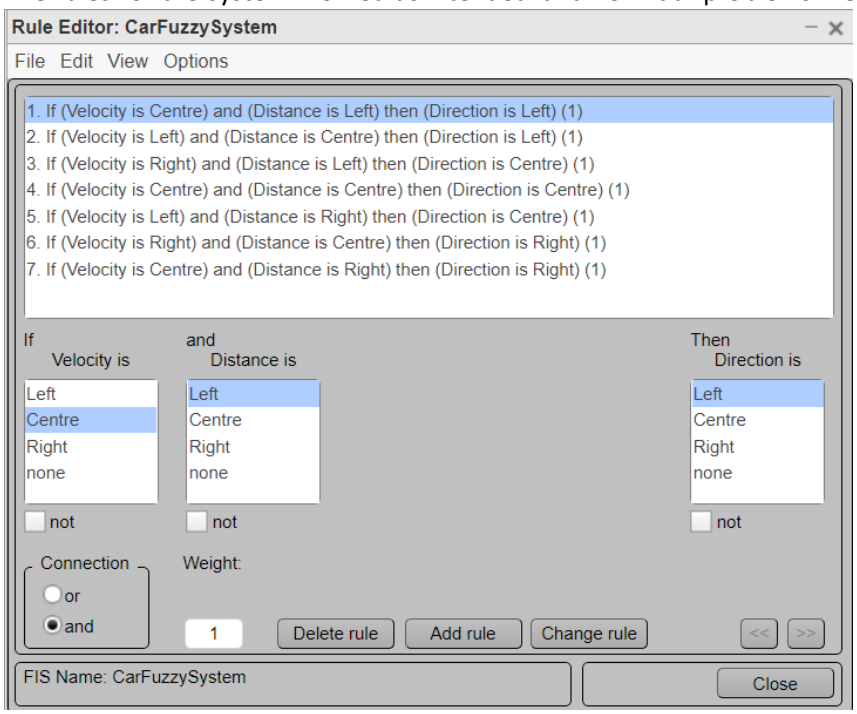




The output for the system was predetermined by Matlab and was not changed for the first attempt, as seen above it has the same states as the inputs.



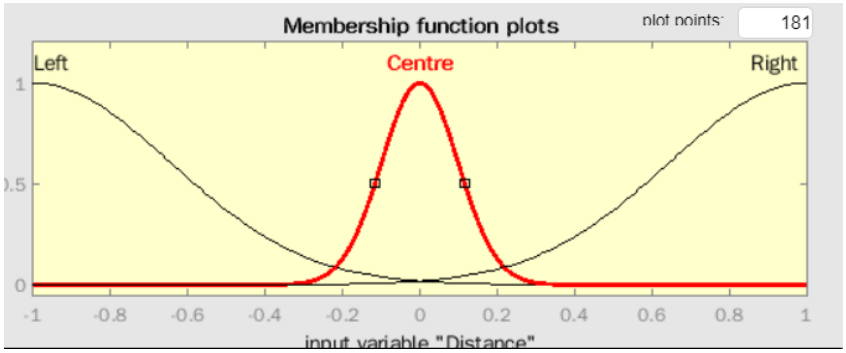
The rules for the system worked as intended and no initial problems were found.

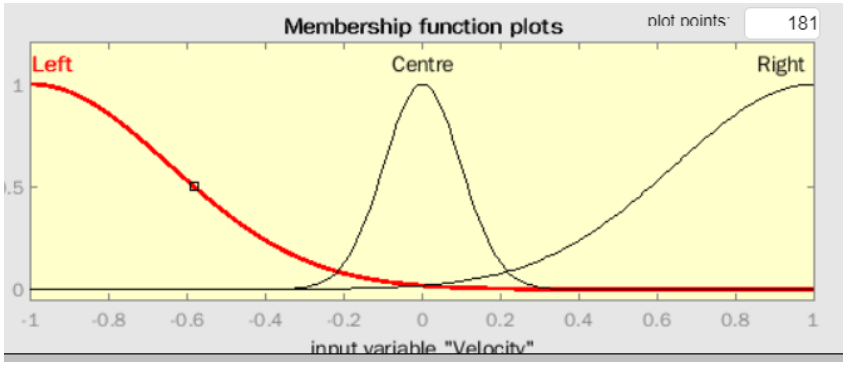


Overall, the first iteration of the fuzzy system worked, however the results needed to be developed further. The car had no urgency to get towards the road, and it did not look realistic when compared to what a human driver would do. To solve this, the machine was made more complex by adding two more states into the program, heavy left and heavy right. Also, if the input graphs were changed so that the centre curve was tighter, it would cause the car to quickly shift into the middle of the racing line.

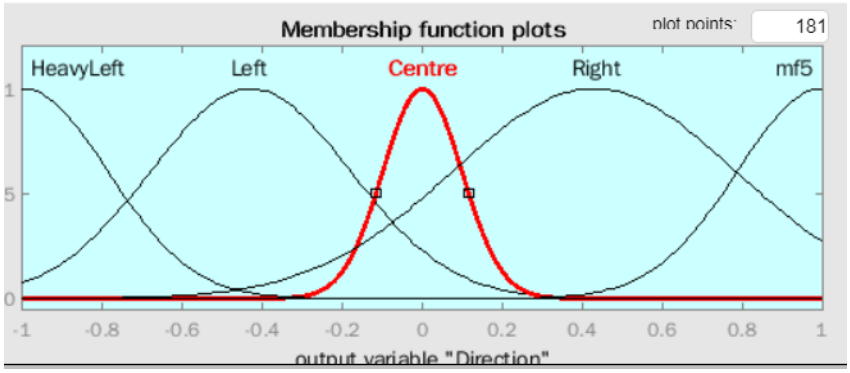
Fuzzy Engine Changes

First the input graphs were changed so that the centre states values are closer together meaning the car will quickly cross into the centre when used in the program.





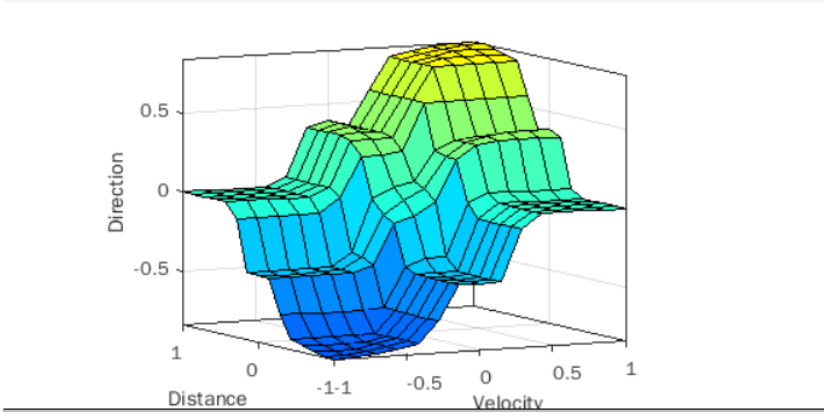
New states were added onto the direction output and the centre curve was changed to match the inputs.

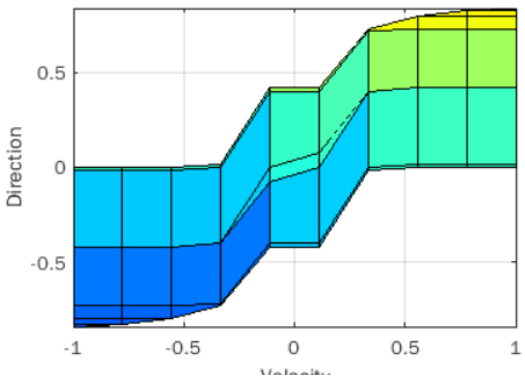


The rules were then updated with the added states.

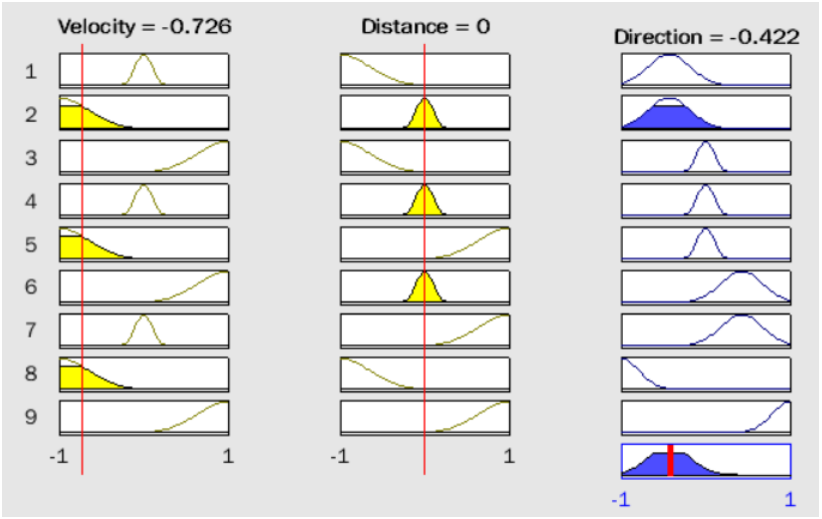
8. If (Velocity is Left) and (Distance is Left) then (Direction is HeavyLeft) (1)
9. If (Velocity is Right) and (Distance is Right) then (Direction is HeavyRight) (1)

As above, the graph is symmetrical meaning the car within the program will behave the same whether the direction it must go is left or right. Although the graph is quite rigid, this helps add realism when in the simulation. As a driver's movements aren't fluid whilst driving unless they are a professional.





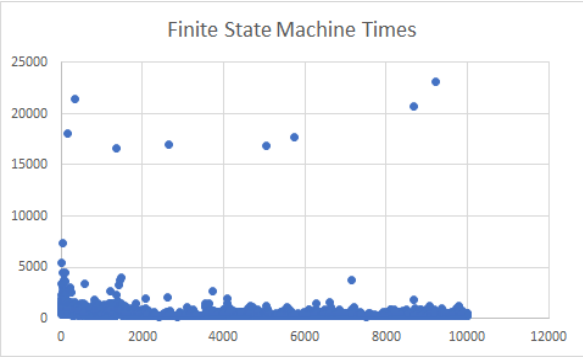
Whilst testing the system, the rules page below was utilised. Inputs were changed to monitor whether the expected results would be received. This made it easier to cross-reference the system with the original plan in place on how the car should behave.

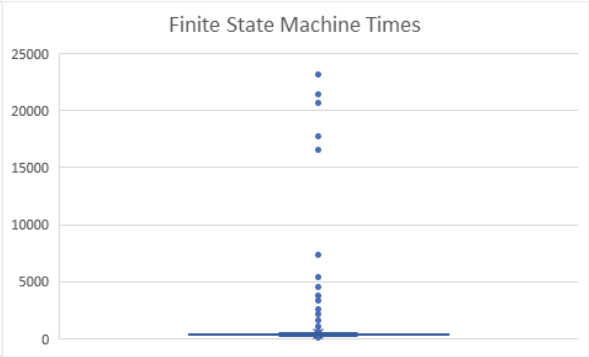


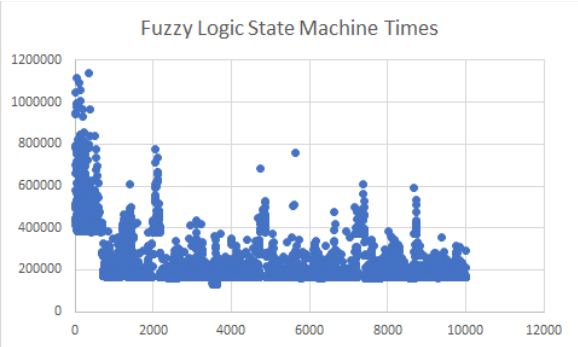
Results

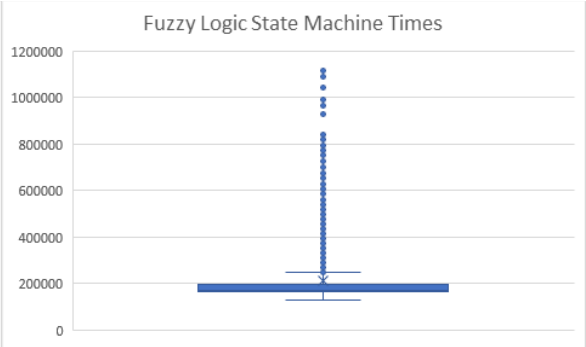
A clock was created using the chrono library built into C++ in order to time the systems, and how long it took for both system's update function to run. The recorded times were placed within an Excel file for the data to be analysed and turn into graphs.

The graph below displays the timings from the system's update functions using a collation of 10,000 iterations. The data from the graph shows that the FLSM is much slower than the FSM. An explanation for this would be that the program must utilize the fuzzy-lite libraries when the FSM only uses C++ statements. This reduced reaction speed of the Fuzzy operated car compared to the FSM car is seen within the game.









FSM

Mean $\bar{x}$	**400.55**
Median $\tilde{x}$	**400**
Mode	**400**
Range	23000
Minimum	100
Maximum	23100
Count *n*	10000
Sum	4005500
Quartiles	Quartiles: Q_1 --> 300 Q_2 --> 400 Q_3 --> 400
Interquartile Range IQR	100

FLSM

Mean $\bar{x}$	**212792.77**
Median $\tilde{x}$	**168400**
Mode	**166700**
Range	1008700
Minimum	130100
Maximum	1138800
Count *n*	10000
Sum	2127927700
Quartiles	Quartiles: Q_1 --> 166300 Q_2 --> 168400 Q_3 --> 198700
Interquartile Range IQR	32400

The finite state machine averaged at around 400 nanoseconds per update however there were some anomalies in the 20,000 range which were expected to be because of the computer the system was running on. The quartiles of the box show that the range between the majority of the tests was only a 100-nanosecond difference.

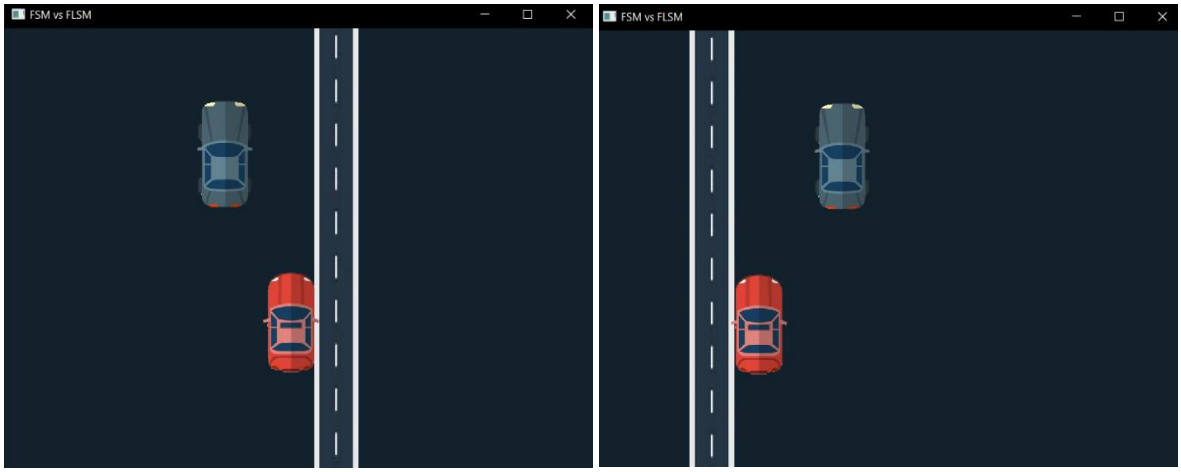
The fuzzy logic state machine averaged at around 168400 nanoseconds per update however there were a substantial number of anomalies which were upwards of 40,000 nanoseconds.

The quartiles of the box plot show that the range between the majority of the tests was a quite big 3,000-nanosecond difference. The scatter graph shows that the FLSM was much slower to start this may be due to setting up the engine with the external libraries. The time instantly decreased after the libraries were set up.

The greatest difference between median times of FSM and FLSM was 168000 nanoseconds, FSM is 421 times faster. This would be noticeable during game play. To lessen the times between the two machines, an in-built fuzzy engine would have to be made, although this is not necessary as the library functions well.

Application

The screenshot below depicts the appearance of the game when run. The user can control the road with 'A' and 'D' to make it move left and right. The top blue car uses fuzzy logic, and the red car uses the finite state machine. When playing, is noticeable that the FSM car will always be closer to the road when compared to the FLSM car.



Conclusion

This assignment was a great opportunity to learn multiple non learning AI techniques which can be brought upon for personal projects and in my future career. I became quite intrigued by the usefulness of fuzzy logic when programming and want to work with it more.

Whilst playing the game, you can definitely notice that the FSM controlled car gets to the road faster than the FLSM controlled one. Although the fuzzy car looks more realistic because of this as it behaves closer to how I imagine humans would drive, giving time for looking at the road line, processing it and then steering. Due to this, if I was to implement this functionality into a big racing game, I would choose the fuzzy logic state machine. However, I would want to build my own system instead of using external libraries to hopefully speed up the processing time.

If I was to take this project further, I would make it so that the cars start at the same y level and add collision. With this, I would then make it, so the cars must also avoid each other while following the road line as to not cause a crash. This would mean my system could be better implemented into a racing game's AI system.

During this project I used Hierarchical programming to separate the different state machines into classes. I found this quite useful when testing and developing the application and will allow me to easily implement it into other projects

Demonstration Video

Here is a link to the video on YouTube:

https://youtu.be/AlSK1Jx982s

References

Videos

www.youtube.com. (n.d.). How to Move Columns in Excel (The Easiest Way). [online] Available at: https://www.youtube.com/watch?v=QghAw--Ul0o [Accessed 28 Mar. 2022].

www.youtube.com. (n.d.). How to Create a Box Plot Chart in Excel 2016. [online] Available at: https://www.youtube.com/watch?v=EiS-q6euyf8 [Accessed 28 Mar. 2022].

www.youtube.com. (n.d.). How to import Fis file in fuzzy logic controller in MATLAB Simulink | open FIS file in Simulink. [online] Available at: https://www.youtube.com/watch?v=-DqDVCUTPWs [Accessed 28 Mar. 2022].

www.youtube.com. (n.d.). BENCHMARKING in C++ (how to measure performance). [online] Available at: https://www.youtube.com/watch?v=YG4jexlSAjc&ab_channel=TheCherno [Accessed 21 May 2021].

www.youtube.com. (n.d.). SFML C++ Tutorial 01 | Setting up SFML. [online] Available at: https://www.youtube.com/watch?v=yEiZalvDOj4&t=6s&ab_channel=SurajSharma [Accessed 28 Mar. 2022].

www.youtube.com. (n.d.). C++ Tutorials- CSV output file. [online] Available at: https://www.youtube.com/watch?v=x2niMA5tzGo&ab_channel=YunusKulyyev [Accessed 28 Mar. 2022].

Websites

Game Development Stack Exchange. (n.d.). Fuzzy State Logic or Finite State Machine for AI. [online] Available at: https://gamedev.stackexchange.com/questions/56338/fuzzy-state-logic-or-finite-state-machine-for-ai [Accessed 28 Mar. 2022].

flylib.com. (n.d.). Fuzzy State Machines | AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors. [online] Available at: https://flylib.com/books/en/2.71.1.296/1/.

Media

VectorStock. (n.d.). Traffic roads top view set vector image on VectorStock. [online] Available at: https://www.vectorstock.com/royalty-free-vector/traffic-roads-top-view-set-vector-18917428 [Accessed 28 Mar. 2022].

Pinterest. (n.d.). 15 Best Car top view ideas | car top view, top view, top down game. [online] Available at: https://www.pinterest.co.uk/sajitharathnaya/car-top-view/ [Accessed 29 Mar. 2022].

Library

FuzzyLite. (n.d.). FuzzyLite. [online] Available at: http://fuzzylite.com/cpp/ [Accessed 28 Mar. 2022].

Sfml-dev.org. (2019). SFML. [online] Available at: https://www.sfml-dev.org/.

uk.mathworks.com. (n.d.). MATLAB - MathWorks. [online] Available at: https://uk.mathworks.com/products/matlab.html.