Introduction

The Application (henceforth referred to as “the App”) uses a Fuzzy Inference System (FIS) to steer a Car towards a moveable Racing Line.

Rapid advancements in Machine Learning techniques such as Artificial Neural Networks have brought Artificial Intelligence to the forefront of much academic, personal and political discussion. The abilities of computers to make complex and accurate decisions across various fields, and the far-reaching socioeconomic implications of this are years ahead of previous projections. When it comes to AI that is specific for games applications, there is still a wide variety of techniques available set apart from other popular techniques by two key factors.

Firstly, the current majority of game-related AI applications are expected to run in real time, ruling out many of the more computationally expensive techniques available.

Secondly, it is important to bear in mind that games AI is not necessarily required to perform a given task as quickly and as accurately as possible. The primary purpose of every element of a video game should be to provide the player with an enjoyable and engaging experience. As such it is often better to create AI that appears human and believable by providing players with a reasonable challenge and often by feeling human rather than performing a task “perfectly.”

With that second point in mind, the App aims to emulate a natural , “imperfect” human reaction to stimuli on a straight stack, such as a motorway. The car gently corrects for small changes in course, and responds to more drastic changes by steering sharply, oversteering, and then having to correct for that oversteer before settling back into its course.

The key principle of a Fuzzy Inference System is in its “fuzziness” mimicking human perception and analysis of the world. Where computers are innately rigid in their logic and reasoning as a result of clearly defined, “crisp” data, Fuzzy Logic allows for more vague, human-like perception and decision making by classifying data into less strictly defined sets and making decisions accordingly. There are definite limits to computerised Fuzzy Logic, as all data and computation is ultimately rigid and binary, but a FIS allows for an acceptable approximation to this with relatively low computational costs.

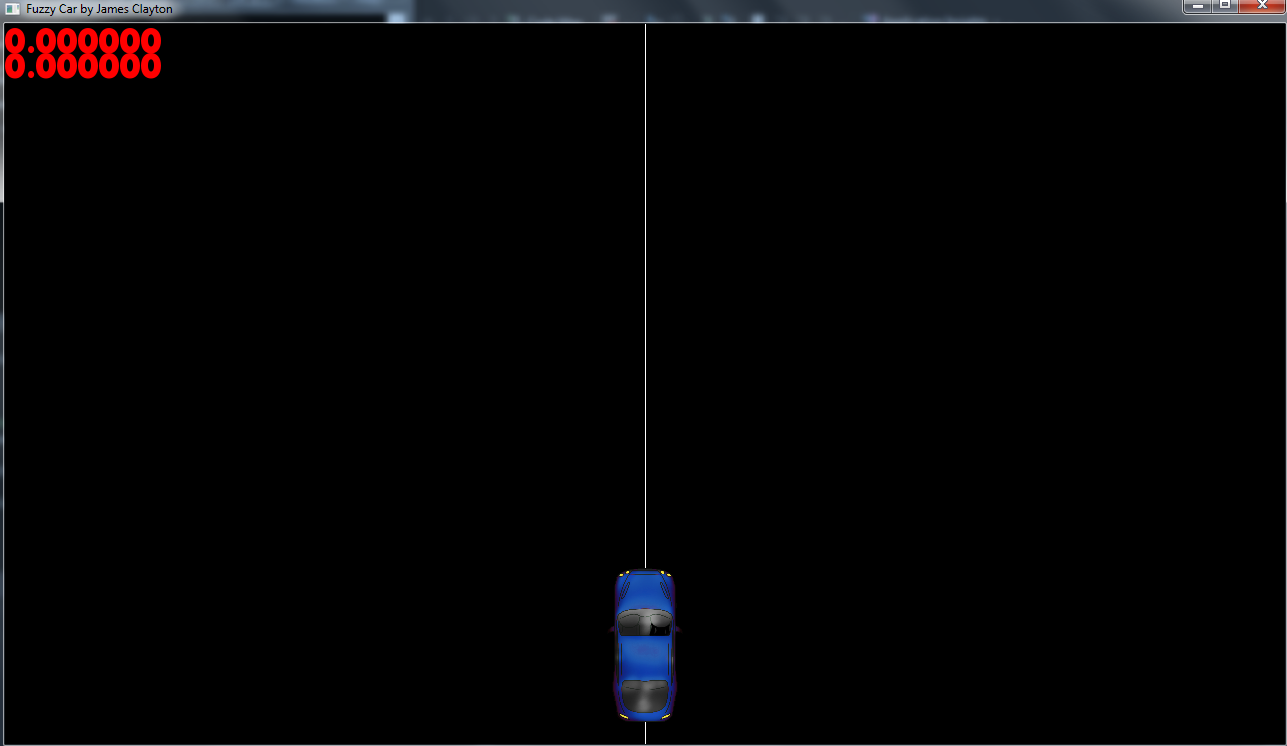
The Front End

When started, the application presents the user with the option of running the application in one of two states;

1. Test Mode, where the user can enter arbitrary values within a specified range and the application will display the corresponding output. A sample of these test values in included in the Results section.
2. Application Mode, where the graphical representation of the application is displayed.

The Graphical representation of the FIS is written in SFML, and assumes a window size of 1080 X 720. In the top right are two numbers displaying the current Velocity of the Car in terms of pixels travelled per tick and the current displacement of the car in pixels relative to the Racing Line in that order.

The Front End of the application was developed before the FIS in order to establish the FIS’s Universe of Discourse, as will be detailed in the following section.



As shown above, the application includes a movable sprite representing the Car, a moveable Racing Line for the car to follow and a pair of floats displaying the velocity and displacement of the car respectively relative to the racing line. The window is 1280 \* 720 pixels. The Line is moved via the arrow keys, and the car is operated by the Fuzzy Inference System.

The Design

As per the coursework specifications, the FIS makes use of two factors.

As the front end of the application is simple and two dimensional, it follows that the universe of discourse be simply categorised into

Velocity

Displacement

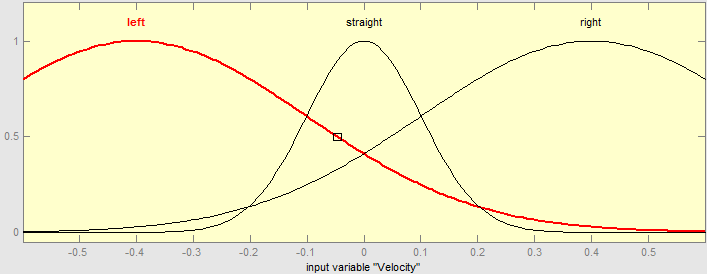
|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Moving Left | Straight | Moving Right |
| Left of Line | Hard Right | Soft Right | No Change |
| Centre | Soft Right | No Change | Soft Left |
| Right of Line | No Change | Soft Left | Hard Left |

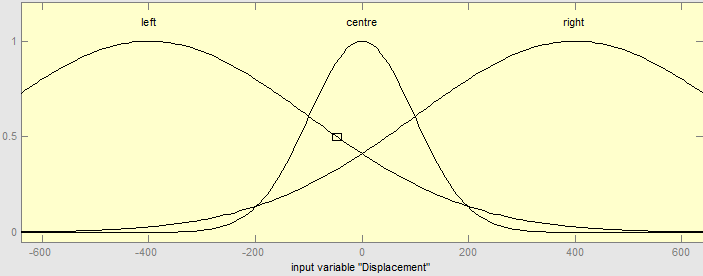
This system has a total of 5 outputs:

1. Hard Right Turn
2. Soft Right Turn
3. No Turn
4. Soft Left Turn
5. Hard Right Turn

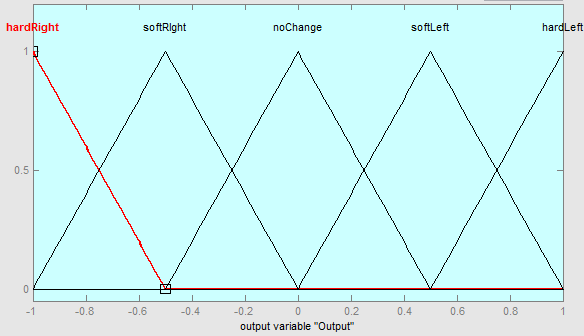
This system was then modelled in MATLAB.

Version 1



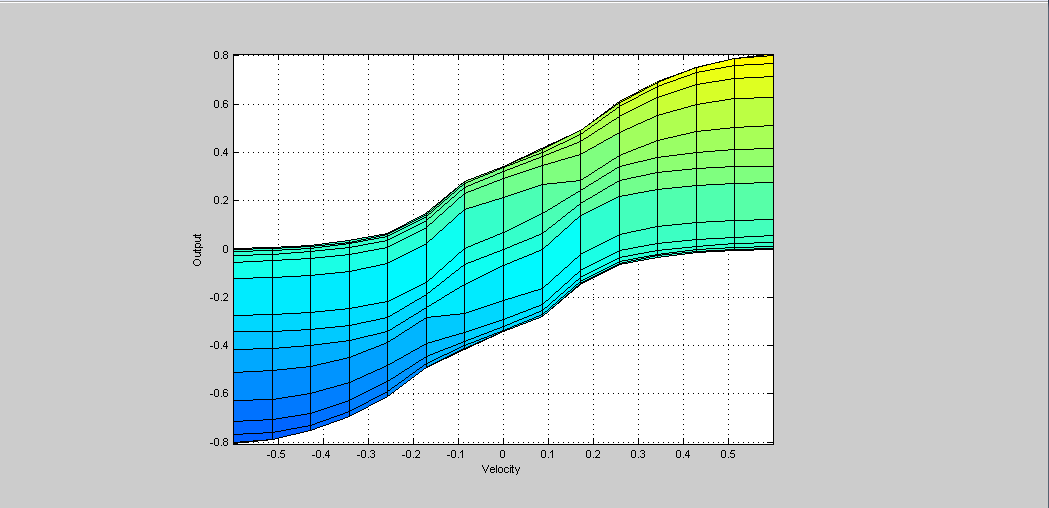
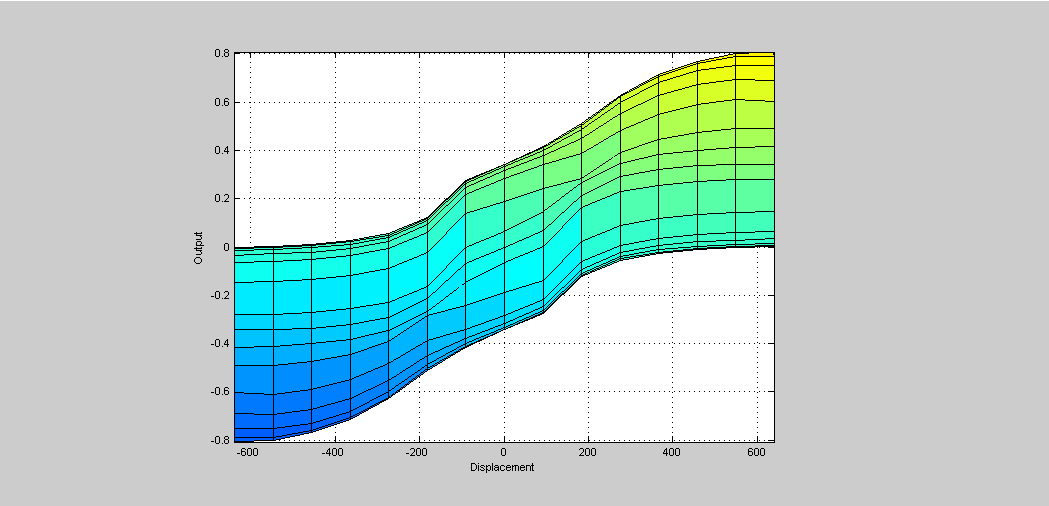
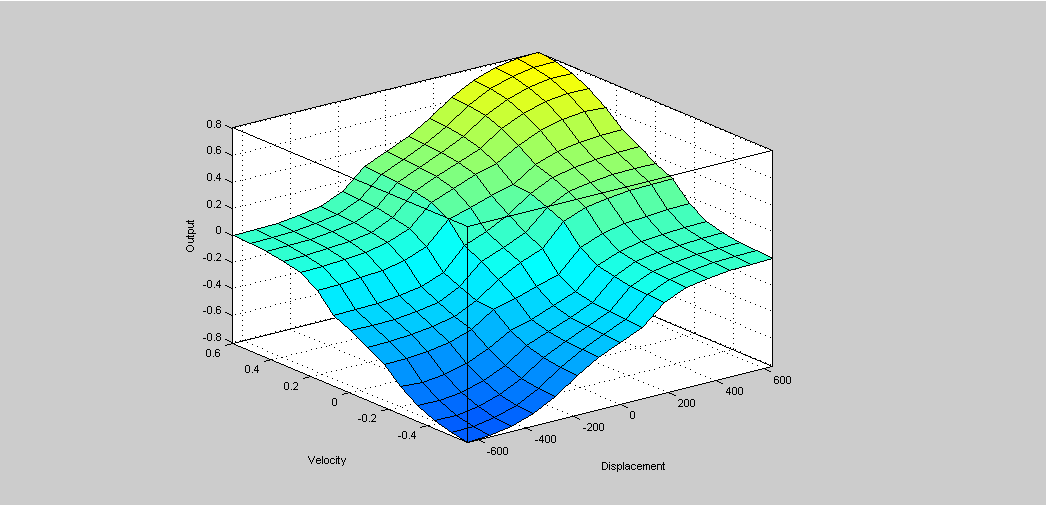


If both cases, the wide domains for the Left and Right values paired with the narrow domains for the central values was intended to produce sizable overlap within the central membership function. This was intended to produce oversteer.



When choosing the Membership Functions for the output variable, I initially decided to go with simple triangles. Testing yielded unsatisfactory behaviour that was devoid of the desired human-like swerving, and was ultimately too bland. Suspecting fault lay with the output graph, I decided to first swap to Gaussian distribution to reflect the input variables and produce smoother, more natural results for Version 2.

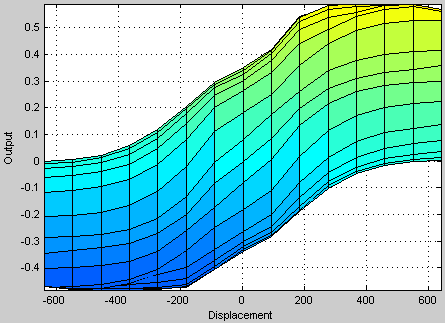
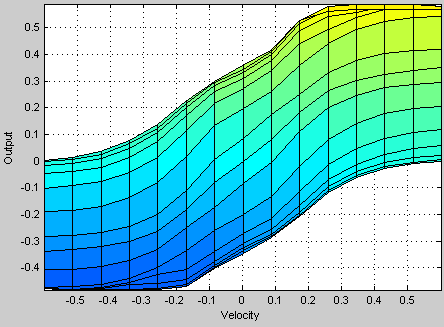
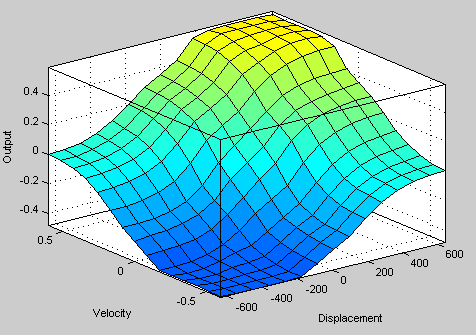
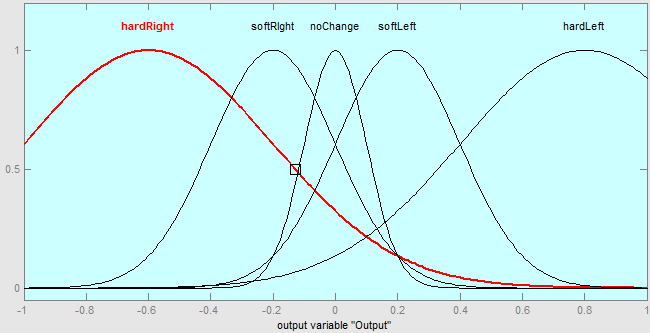
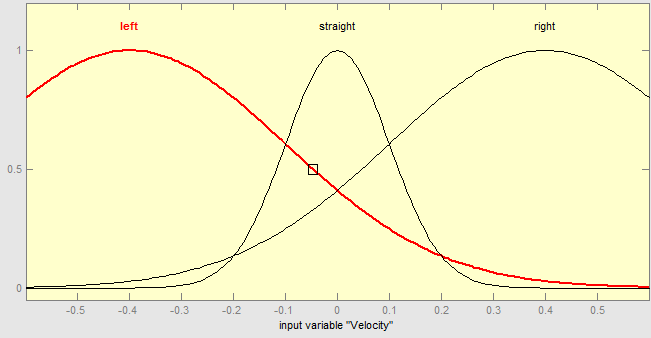
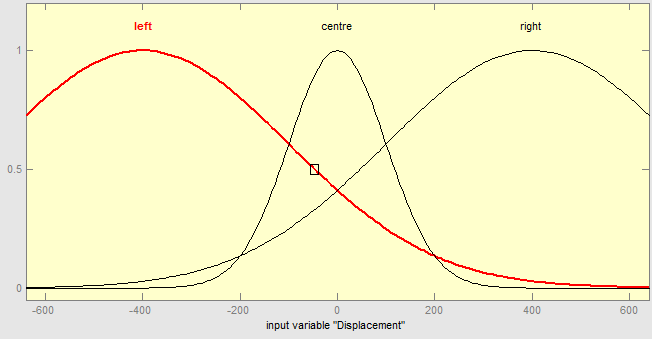
# Output Surface



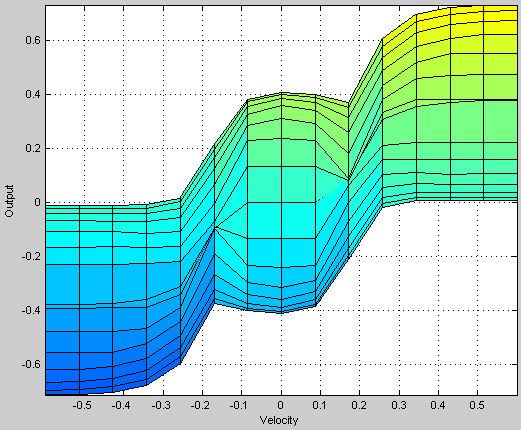
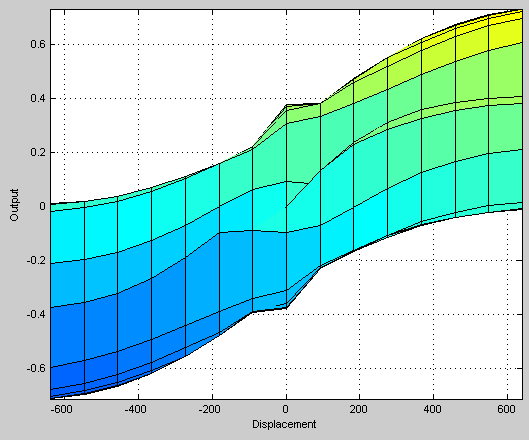
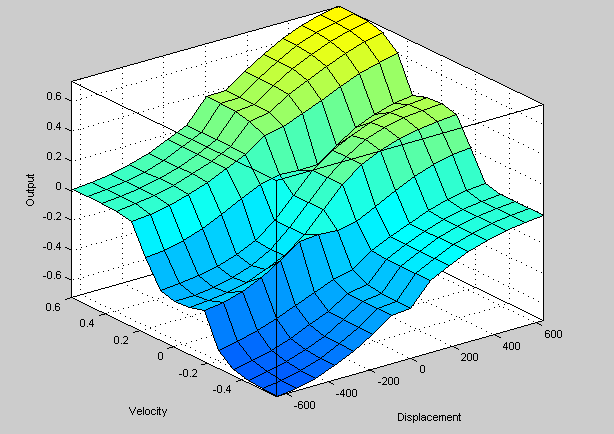
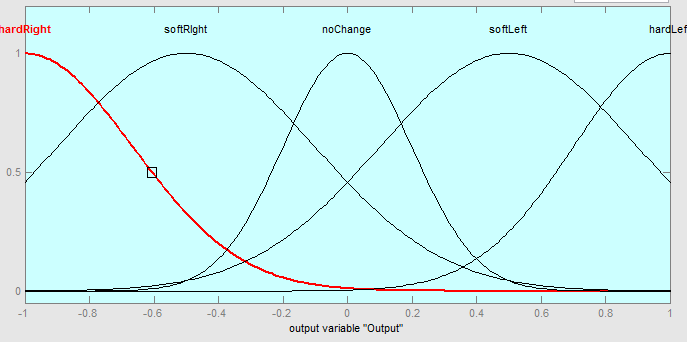
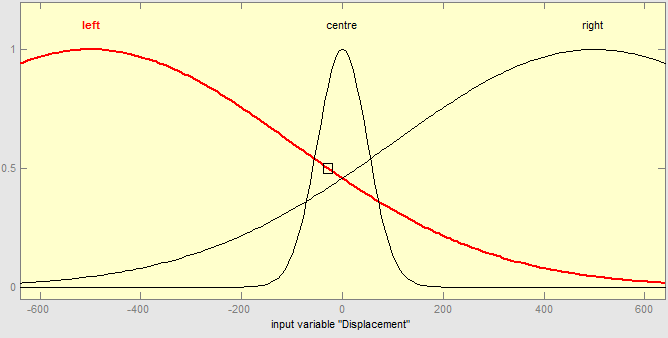
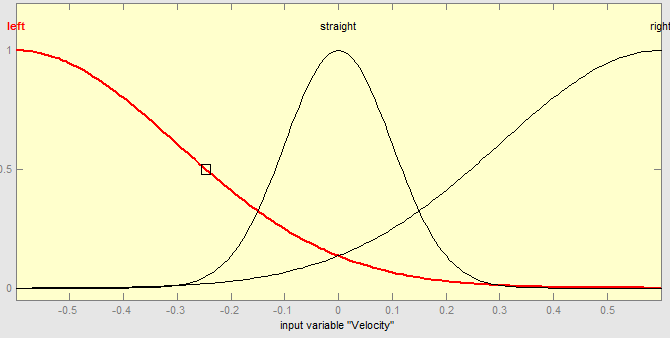
It was also noted that the output graph was smoother than anticipated. The area circled above was expected to be more flat to induce the desired oversteer. This was taken into account in subsequent iterations.

Version 2

Following the results of Version 1, the Output Functions were modified to use a Gaussian curve and repositioned



Version 3



I believed that this would be too simplistic for the behaviour that I had in mind, so I increased to five membership functions as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Inputs | Fast Left | Slow Left | Dead Ahead | Slow Right | Fast Right |
| Far Left | Hard Right | Hard Right | Hard Right | Soft Right | No Change |
| Left | Hard Right | Soft Right | Soft Right | No Change | Soft Left |
| Centre | Hard Right | Soft Right | No Change | Soft Left | Hard Left |
| Right | Soft Right | No Change | Soft Left | Soft Left | Hard Left |
| Far Right | No Change | Soft Left | Hard Left | Hard Left | Hard Left |

Following this, the Five Membership functions were implemented in MATLAB and evaluated.

\*/

Test Data

Conclusions

References