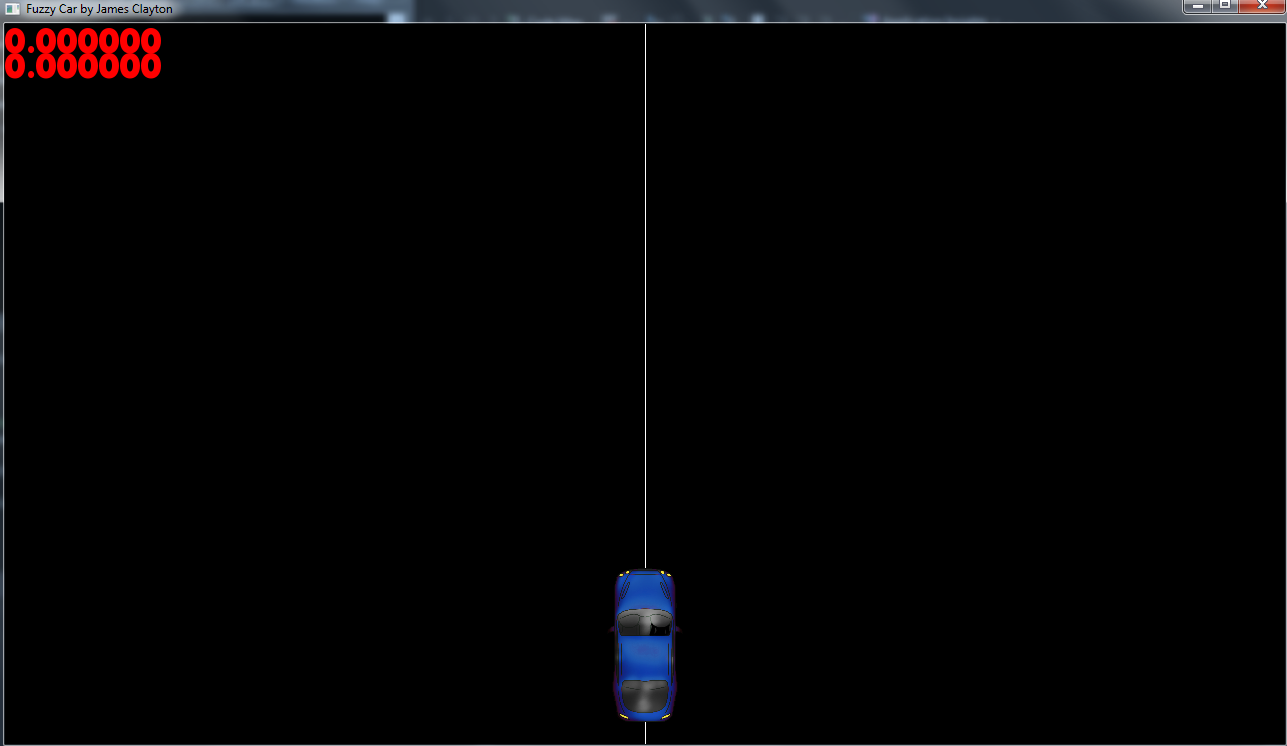
Introduction

The Application (henceforth referred to as “the App”) uses a Fuzzy Inference System (FISM) to steer a Car towards a Racing Line. In order to best understand and demonstrate the strengths of Fuzzy Logic within the brief, the App aims to emulate a natural human reaction to an abrupt change in course (i.e. swerving and potential oversteering before arriving back at the racing line.)

The Front End

To begin the App I created the front end of the application in SFML to get a better idea of the Universe of Discourse for the FIS and then develop a sensible architecture for ‘plugging in’ the FIS later.



As shown above, the application above 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 relative to the racing line respectively. The window is 1280 \* 720 pixels. The Line is moved via the arrow keys, and the car is operated by the Fuzzy Inference System.

Version 1

Next, I sketched out a rule diagram for a simple scenario with 3 Membership functions per input as follows:

Velocity

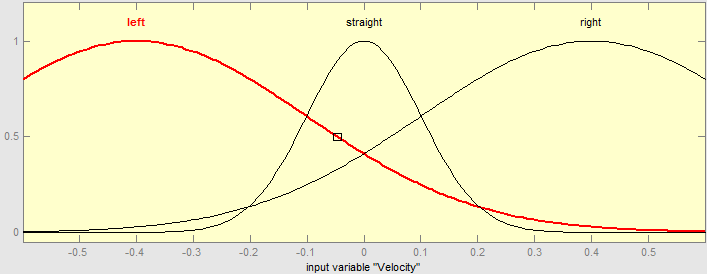
Displacement

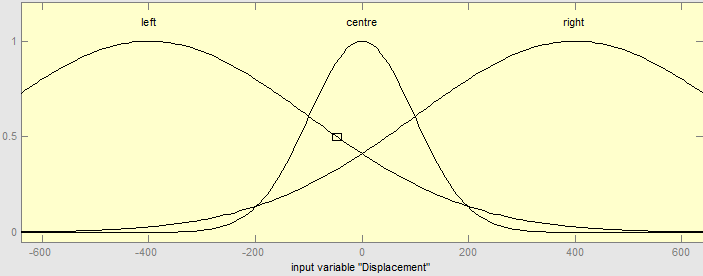
|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Moving Left | Straight | Moving Right |
| Left of Line | Hard Right | Soft Right | No Change |
| Centre | Soft Right | No Change | Soft Left |
| Right of Line | No Change | Soft Left | Hard Left |

This system has a total of 5 outputs:

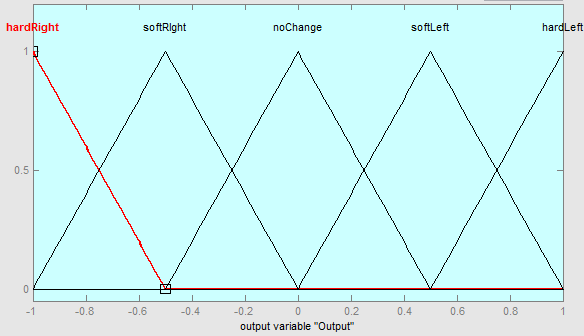
1. Hard Right Turn
2. Soft Right Turn
3. No Turn
4. Soft Left Turn
5. Hard Right Turn

This system was then modelled in MATLAB.





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.

/\*

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

Results

Conclusions

References