FRASER BARKER

1600196

1600196@uad.ac.uk

ASSESSMENT REPORT

MAT301

Contents

[Introduction 2](#_Toc6476376)

[AI Techniques 2](#_Toc6476377)

[Finite State Machine 2](#_Toc6476378)

[Fuzzy Logic State Machine 2](#_Toc6476379)

[Reasoning 2](#_Toc6476380)

[Computational Efficiency 2](#_Toc6476381)

[Finite State Machine 2](#_Toc6476382)

[Fuzzy Logic State Machine 2](#_Toc6476383)

[Ease of Coding 2](#_Toc6476384)

[Method 3](#_Toc6476385)

[Variables 3](#_Toc6476386)

[Input 3](#_Toc6476387)

[Output 3](#_Toc6476388)

[Test Data 5](#_Toc6476389)

[Graphs 6](#_Toc6476390)

[Constant Velocity Graphs 6](#_Toc6476391)

[Constant Distance Graphs 7](#_Toc6476392)

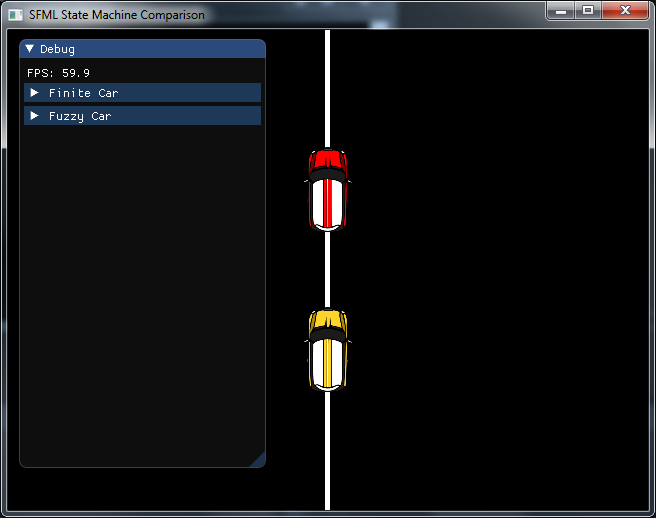
[Results & Conclusions 8](#_Toc6476393)

[References 9](#_Toc6476394)

# Introduction

## Application

The application uses a finite state machine to control the red car, and a fuzzy inference system to control the green car.



At start, the user is presented with a debug menu that they can expand to see the values being calculated for each car. The Fuzzy Car specifically can have its distance and velocity set by the user if they untick the “Calculate Variables” box. This will then allow the user to see values similar to what is documented in the Test Data section.

The user can move the line left/right using the A/Left arrow and D/Right arrow keys respectively.

## AI Techniques

### Finite State Machine

A finite state machine was chosen as the first AI technique as it was easy to implement when considering the application would be moving a car sprite left/right depending on its distance away from a driving line and the speed it was moving at.

It’s also the easiest to compare against a fuzzy logic state machine is there is some overlap between the two techniques.

### Fuzzy Logic State Machine

A fuzzy logic state machine was chosen to compare against a finite state machine due to there being overlap in how both techniques operate.

The inference system was generated in MATLAB. This was then imported into the project using the FuzzyLite toolset. The result moves the car left/right depending on its distance from the driving line and its speed.

## Reasoning

As both techniques are state machines, the only differing factor is how the input data is used to calculate what direction the car should move in.

This made them an ideal comparison as they both function similarly.

## Computational Efficiency

### Finite State Machine

Largely made up of floats and an enum to depict what state the car is in. Aside from SFML variables to render the sprite it’s pretty simplistic.

### Fuzzy Logic State Machine

Similar to the finite state machine except the direction of the fuzzy car is dependent on its direction being calculated via FuzzyLite.

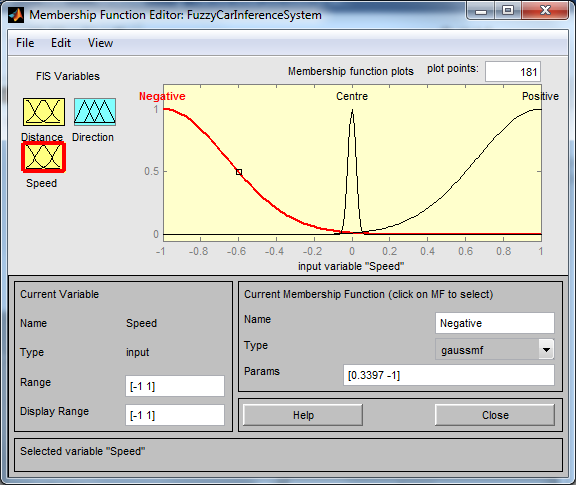
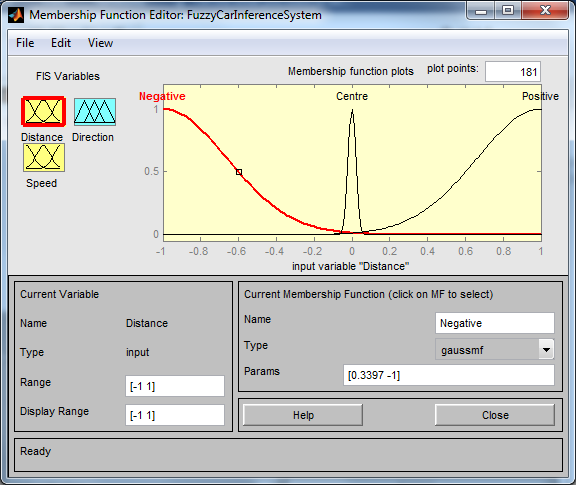
## Ease of Coding

Both systems were easy to set up. The trickiest part was getting the FuzzyLite library into the project which required building files via CMake.

# Method

## Input

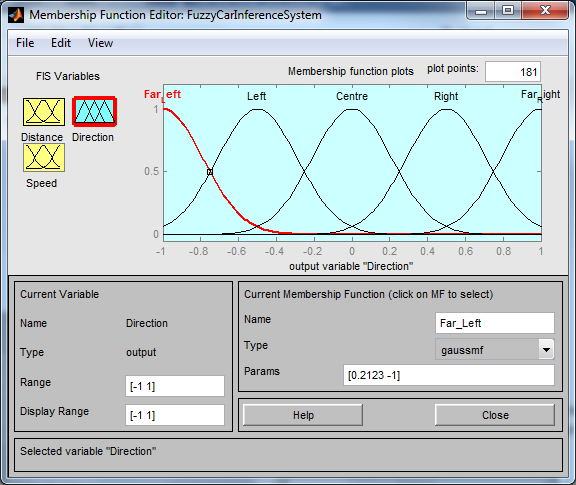
Both systems take the distance from the driving line and their current velocity as inputs to their systems.



## Output

The output for the finite state machine is acceleration which is determined via what state the car is in.

The output for the fuzzy logic state machine is direction which is calculated via centroid of the two input graphs distance and speed at their respective points of distance and velocity input.



# Test Data

|  |  |  |  |
| --- | --- | --- | --- |
| **MATLAB** | | | |
| **Velocity = 0** | | **Distance = 0** | |
| **Distance** | **Output** | **Velocity** | **Output** |
| -1 | -0.423 | -1 | -0.423 |
| -0.75 | -0.422 | -0.75 | -0.422 |
| -0.5 | -0.417 | -0.5 | -0.417 |
| -0.25 | -0.385 | -0.25 | -0.385 |
| -0.1 | -0.143 | -0.1 | -0.143 |
| 0 | 8.01e-18 | 0 | 8.01e-18 |
| 0.1 | 0.143 | 0.1 | 0.143 |
| 0.25 | 0.385 | 0.25 | 0.385 |
| 0.5 | 0.417 | 0.5 | 0.417 |
| 0.75 | 0.422 | 0.75 | 0.422 |
| 1 | 0.423 | 1 | 0.423 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Application** | | | |
| **Velocity = 0** | | **Distance = 0** | |
| **Distance** | **Output** | **Velocity** | **Output** |
| -1 | -0.422 | -1 | -0.422 |
| -0.75 | -0.421 | -0.75 | -0.421 |
| -0.5 | -0.415 | -0.5 | -0.415 |
| -0.25 | -0.380 | -0.25 | -0.380 |
| -0.1 | -0.140 | -0.1 | -0.140 |
| 0 | -3.231e-18 | 0 | -3.231e-18 |
| 0.1 | 0.140 | 0.1 | 0.186 |
| 0.25 | 0.380 | 0.25 | 0.380 |
| 0.5 | 0.415 | 0.5 | 0.415 |
| 0.75 | 0.421 | 0.75 | 0.421 |
| 1 | 0.422 | 1 | 0.422 |

Talk about discrepancy.

# Graphs

## Constant Velocity

## Constant Distance

Talk about discrepancy again.

# Results & Conclusions

# References

Unknown. 2010. A Short Fuzzy Logic Tutorial. [ONLINE] Available at:

<http://cs.bilkent.edu.tr/~zeynep/files/short_fuzzy_logic_tutorial.pdf> [Accessed 17/04/2019]

Juan Rada-Vilela. 2017. FuzzyLite. [ONLINE] Available at:

<https://fuzzylite.com/> [Accessed 17/04/2019]

Ocornut. 2018. ImGui-SFML. [ONLINE] Available at:

<https://github.com/eliasdaler/imgui-sfml> [Accessed 17/04/2019]

Ocornut. 2019. ImGui. [ONLINE] Available at:

<https://github.com/ocornut/imgui> [Accessed 17/04/2019]

Bahi. 2011. Cartoon Car- Mini. [ONLINE] Available at:

<https://opengameart.org/content/cartoon-car-mini> [Accessed 17/04/2019]

Laurent Gomila. 2019. SFML. [ONLINE] Available at:

<https://www.sfml-dev.org/learn.php> [Accessed 17/04/2019]

MathWorks. 2019. MATLAB. [ONLINE] Available at:

<https://uk.mathworks.com/products/matlab.html> [Accessed 17/04/2019]

KitWare. 2019. CMake. [ONLINE] Available at:

<https://cmake.org/> [Accessed 17/04/2019]