|  |
| --- |
| **CMP304 AI Part**  **Project Report (50%)**  **Title** |
|  |
| **by: Daryl Grant 1702875** |

|  |
| --- |
| **1. Introduction (5%)** |
| Relevant overview properly setting the context of the project.  For my project I choose to make both a Racing Car application as well as a Fuzzy Interface System to use with the application. The application will show a car following a controllable “racing line” using fuzzy logic and then be compared to another AI method controlling the car, which in this case will be a rule based system.  The fuzzy inference system will be designed in MATLAB’s Fuzzy Logic Toolbox and then implemented into the application using the Fuzzy-Logic-Sharp library from <https://github.com/davidgrupp/Fuzzy-Logic-Sharp>.  The application will show 2 separate race cars, one controlled by fuzzy logic and one controlled by the rule based system, on the same track following one line. |
| **2. Methodology (15%)** |
| Description of the steps followed and methods used including a complete explanation and rationale for the techniques and features chosen. You should also acknowledge the data and tools you used.  M:\Game\fuzzy1d.pngMATLAB’s Fuzzy Logic Toolbox was chosen to design the fuzzy inference system to be used by the racecar, as it is a relatively quick and easy to use tool for setting up such a system. The system has 2 input variables, distance from the racing line and linear velocity towards to racing line, and output a variable to control the acceleration of the car towards the line based on those 2 inputs.  The distance variable was given 5 membership functions; Far Left, Left, Centre, Right and Far Right. These were initially set to values between -1 and 1 but were later changed in scale as the distances in the actual application were different.  M:\Game\fuzzy1s.png  The velocity variable was set up in a similar way with the membership functions being Fast Left, Left, Still, Right, Fast Right and there values initially set between -1 and 1. The steering output variable was also given 5 membership functions to match this namely Hard Left, Left, No Steering, Right and Hard Right, with values gain between -1 and 1.  Unity, with the Fuzzy Sharp Library included, was the software chosen to create the racing car application as this would give a simple and effective way to create a game simulation where the fuzzy AI could be tested and allow for more focus on tweaking and testing the values and rules in both the fuzzy and rule based logic.  The scene was set up so that 2 racecars on a track would appear to continually drive forward (they have no real forward velocity) and a race line would sit in the centre of the track, with its left and right position controllable by the user. Data for both the fuzzy and rule based cars show on the left and right of the screen respectively giving information on each cars distance from the racing line, current velocity and calculated steering output.  Each car was given the same basic car script with functions for getting the cars current velocity and position on the track as well as a function to set the x axis acceleration of the car. The velocity of the car cannot be set directly and is only updated based on the acceleration each frame. Each car was then given an additional fuzzy or rule based script which would take in the appropriate cars velocity and position, as well as the current position of the racing line, and then use the relevant AI technique to calculate the required steering and then apply this back to the racecar.  For the fuzzy logic calculations the Fuzzy-Logic-Sharp library was imported into the project which made it simple to implement the variables, membership functions, and rules designed earlier in MATLAB. Some tweaks were needed however as the -1 to 1 ranges used previously didn’t fit well with the size of the car and road, so these were changed to go from -3 to 3 for each of the variables. The Centroid was chosen as the defuzzification method as this is simple yet useful as it takes into account all the effective rules and doesn’t lose information like other simpler method would. Since there were 2 input variables each with 5 membership functions, 25 rules were created to cover every possible scenario of the fuzzy racecar.  For the rule based car the values of the fuzzy membership functions were taken for each input variable and used to create a similar set of 25 rules based on the car reaching set positions on velocities. These would be checked 1 by 1 until 1 of the rule checks pass and a suitable steering value, similar to the similar membership function in the fuzzy output, is returned and applied to the rule based car as linear acceleration.  The application also includes the ability to pause each of these cars separately and enter values for their input variables manually, while still outputting the appropriate steering value to the screen. |
| **3. Results (10%)** |
| Comment on the performance of your application, including test cases. Tabulate and discuss your results. A quantitative measure of performance must be presented.  //Talk about how it was tested after you test it. Say initially you could see the jumps in rule based then back that up with the numbers. Say something about why this is that way and suggest improvements. Add pictures |
| **4. Conclusion (10%)** |
| Full analysis and summary of the project. |
| **5. References (5%)** |
| A number of references properly cited in Cite Them Right Harvard style. |

Structure, style, formatting, spelling, grammar, coherence (5%)