|  |
| --- |
| **CMP304 AI**  **Car Racing With Fuzzy Logic** |
|  |
| **by: Daryl Grant 1702875** |

|  |
| --- |
| **1. Introduction** |
| 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 be created with Unity and 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** |
| 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.  25 rules were created to cover every combination of the 5 membership functions of the 2 variables with an appropriate output given in each case.  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** |
| For testing, values were chosen for both input variables between -3 and 3 (-3, -2, -1.5, -1, -0.5 0, 0.5, 1, 1.5, 2, 3) and tested each of these distances against each velocity for both the rule based and the fuzzy car, with the outputted steering results stored in a table for comparison. A few more “in between” values were chosen (-0.5, 0.5…) as this seemed to be where the fuzzy logic would show most of the difference to the rule based.  Before looking at the results of testing it was clear that the rule based car had much larger jumps in speed when changing from rule to another, where the fuzzy logic was a lot smoother in its changes. This is confirmed by the test results as it can be seen that the fuzzy logic shows a more gradual change (and would show an even more gradual change with more test values), especially towards the 0 values, while the rule based logic gives more concrete values until the specific conditions for the rules are met. This also causes the rule based car in the application to never reach the line, as every time it reaches the line it has the same positive or negative velocity every time it tries to correct itself. This issue, as well as the quality of the rule based system overall, could be improved by adding more rules causing the changes in each rule to be less of a jump. With enough rules the rule based method could even perform similar to the fuzzy results achieved here, however fuzzy can achieve these results with only 5 rules per variable showing that it is a much more effective method of controlling the car. |
| **4. Conclusion** |
| Overall it can be seen that the fuzzy logic solution to this problem is the more effective choice as it gives much smoother transitions between each rule condition that is met in comparison to the rule based solution.  The rule based system is much easier to implement(although this project uses an easy to implement fuzzy logic library), however the fuzzy method it is worth the extra coding effort if more precise results are required as adding extra membership functions/variable values to either method causes the rule count to become exponentially larger. Adding more rules could have also fixed the issue of the rule based car never settling in the centre, but seemed like an inferior solution to just changing to the fuzzy method.  Using a larger number of test values was also considered to compare the methods in detail but even this small amount shows the fuzzy logic starting to change gradually between values, while the rule based logic would still make sharp changes between specific numbers.  This project makes it clear that unless the most basic of AI method is acceptable, then the fuzzy logic is the right choice for an AI racecar following a racing line. |
| **5. References (5%)** |
| Fuzzy Logic Sharp - <https://github.com/davidgrupp/Fuzzy-Logic-Sharp> (Accessed 02/2019)  Unity3D - <https://unity.com/> (Accessed 02/2019) |