FUZZY CONTROL FOR RACING GAMES

Coursework report for CE1184 – Artificial Intelligence for Games, University of Abertay, Dundee

Richa Sachdeva
M.Sc. Computer Games Technology
Student Id. – 1304869
17th January 2014

Introduction

Video games have come a long way from their humble beginning as mere pixels on the screen in the games like Pong, with characters and a modest AI in the Pac-Man, to present day games with real life like graphics and much improved AI. Artificial intelligence is required or is an indispensable part of games as it's the behavior of the non player characters (NPC's) that makes the game unpredictable, challenging and fun. Without AI, NPC's behavior would become predictable and soon the player would lose interest in the game. Thus, over the course of evolution of video games, AI have also evolved and made them indispensable for the games industry. However, there is little match in development of the graphics and the AI leaving room for further exploration and experimentation in the field.

Fuzzy logic was first proposed in 1965 by Lotfi Zadeh, a professor at University of California Berkeley in his paper on linguistic variables [wikipedia]. As in the words of the creator, "fuzzy logic is a means of presenting problems to computers in a way akin to the way humans solve them" [Woehr J, 1994, *Dr. Dobbs Journal*]. Thus, fuzzy logic enables one to pose and solve problems using linguistic terms similar to what one might use and it enables one to think in normal terms while using very precise tool such as computers [Bourg, D.M, *AI for Game Developers,* 2004: 188].

Fuzzy logic relies on linguistic variables expressing the action or state of the system under study and a bunch of if-then statements which are actions to be taken in case the state of the system changes. Thus, the steps involved in coding a fuzzy system are as follows -

- Define the system, inputs, and output.
- Define the possible states the input and output variable can be in. Like if temperature is one of the input state, then it can be hot, cold.
- Define the relationship between the input and the output and based on it, formulate the rules as series of if-then statement. Like is temperature is cold, heater will be turned on (considering turning on/off heater as an output).
- Define fuzzy logic membership functions in terms of input, output variables and the relationship between them based on the rules deduced above.
- Program the membership function using Matlab or some other fuzzy logic system and test the working of the system.

The coursework also expected to implement the same system using a different methodology. The alternative strategy that's been used is the Chase and Evade method, popularly used in predator-prey cases. In the chase and evade method, the predator chases the prey. The algorithm used to implement it is known as First Chase Algorithm, in which it is assumed that there are no obstacles present. While, in a real world it doesn't seem natural as obstacles are bound to be present, but in the given case it was so assumed and implemented.

The reason this algorithm was chosen was because of its inherent simplicity and effectiveness. The system under study is a simple structure built with the intent of understanding fuzzy systems and their working. Thus, many features like obstacles in the racing path, bending of the path and/or other factors were neglected. Also, all fuzzy systems require some amount of tuning which proved tedious to understand and work around in the graphical application accompanying the report. Moreover, the chase and evade algorithm proved to be analogous to the current situation, as in a way car is chasing the road and hence the car's behavior can be deduced from that of the road.

The rest of the report discusses in detail how the fuzzy system was implemented, the results. It also outlines the alternate method used, results and compares the two processes and their respective results.

Methodology

The steps involved in designing a Fuzzy Inference System were as follows:

Fuzzy System Design

The initial step in the design of the system was to define the system, input and output. The problem describes a closed system of controlling a racing car. The inputs and output were also defined as-

Inputs -

- Distance of the car from the line, or the relative distance between the car and the line
- Rate of change of line with respect to the car, i.e. the line's velocity with respect to the car

Output -

- Some value that relates to steering the car back towards the line, or is the rate of change of car's velocity with respect to change in the inputs

Using the inputs and the output, the next step is to define the possible states the input and the output variable are present in. Defining the states, and/or defining the membership functions are clubbed in a single step. Thus, the membership functions for the given Fuzzy system are as follows-

Membership functions

To ensure a smooth transition from one state to another, to reduce jiggling and to be able to cover as many cases as possible, a total of five membership functions were chosen for the input variables. For the output variable, a total of nine membership functions were chosen, which allowed greater control on the car's steering, which resulted in near natural or acceptable car's behavior.

The membership function used were -

For Input variables -

- Distance - Far Left, Left, Zero, Right, Far Right
- Velocity - High Left, Left, Zero, Right, High Right

For Output variable -

- Steer - Extreme Left, Large Left, Left, Slight Left, None, Slight Right, Right, Large Right, Extreme Right

Universe of Discourse

Universe of discourse represents range of all possible values for an input to a fuzzy system. In this case, the range was chosen to be -100 to 100 for inputs as well as the output variables. The fuzzy application was coded using Java, in which domain of distance used was {-300, 300} and hence suitable mapping (multiplying values by 3) was applied. Likewise, for velocity and steer, domain used was {-5, 5}. Velocity was mapped using a factor of 20 (dividing values by 20), but for steer some tuning was essential and it was mapped by a factor of 10 (dividing steer values by 10).

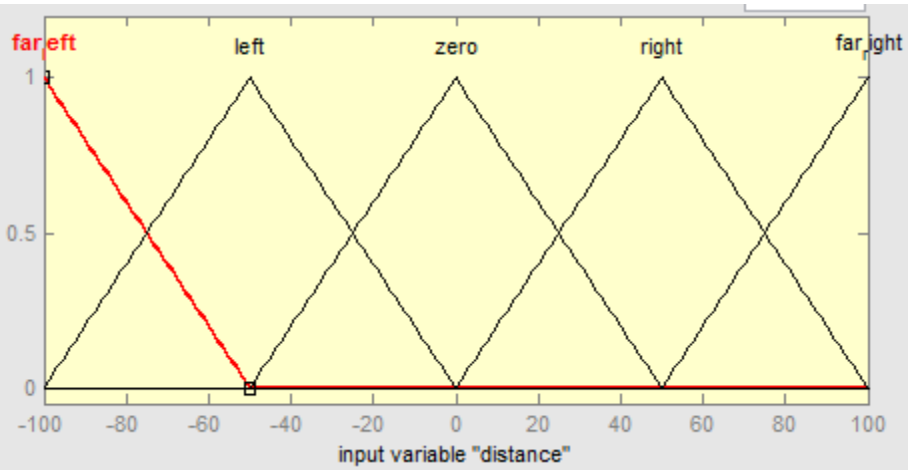
Shape of Membership function

As the intent of the exercise was to get familiar with the fuzzy logic and understand the fuzzy systems in detail, to be able to develop fuzzy systems in the future, the shape of the member functions was kept simple. A triangle style membership function was chosen for all sets. Though it can be easily applied to trapezoid or a Gaussian function, as was experimented in the early stage of the project, but due to the restrictions in the library used to test the application, triangular shaped membership functions were finally used.

Domain of Membership functions

As overlapping of the membership function states is essential for correct working of any fuzzy system, the membership functions were equally spaced from each other to ensure that at any given input of distance and velocity, the output is part of two (or more) membership functions, as depicted in the pictures below -

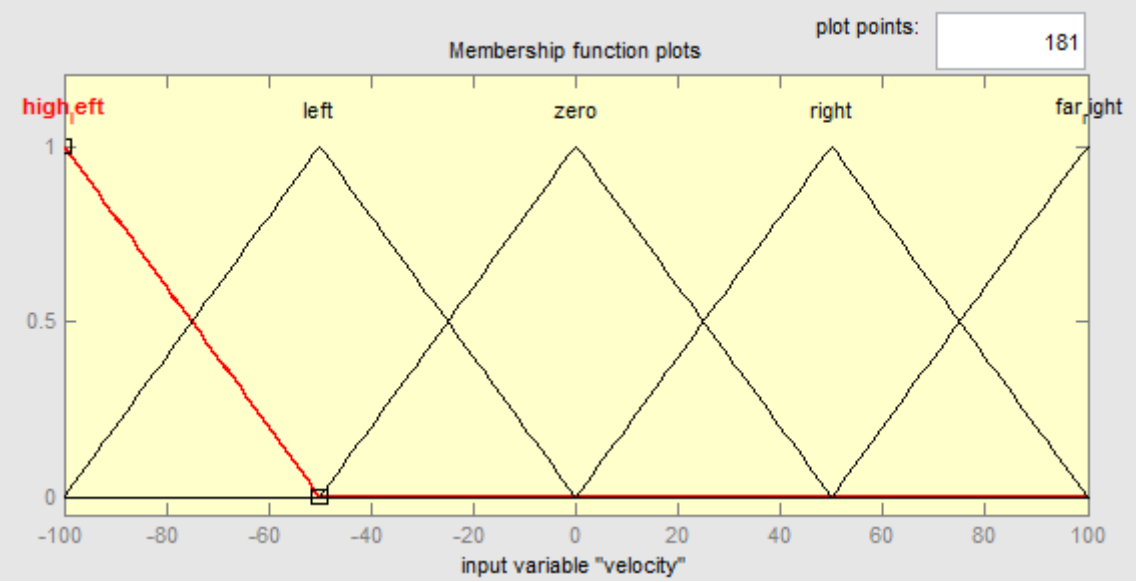
Membership functions for input variable, distance



The domain of membership function in the distance variable was:-

- Far Left - { -150, -100, -50 }
- Left - { -100, -50, 0 }
- Zero - { -50, 0, 50 }
- Right - { 0, 50, 100}
- Far Right - { 50, 100, 150}

Membership functions for input variable, velocity

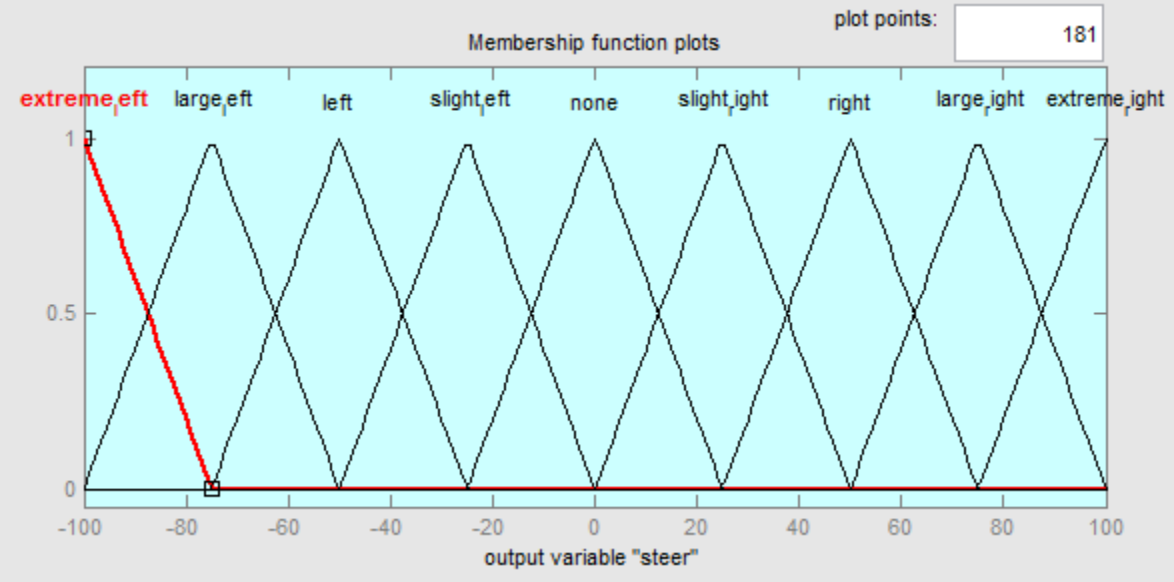


The domain of membership function in the velocity variable was:-

- High Left - { -150, -100, -50 }
- Left - { -100, -50, 0 }

- Zero - { -50, 0, 50 }
- Right - { 0, 50, 100}
- High Right - { 50, 100, 150}

Membership function for output variable, steer



The domain of membership function in the steer variable was:-

- Extreme Left - { -125, -100, -75 }
- Large Left - { -100, -75, 50 }
- Left - { -75, -50, -25 }
- Slight Left - { -50, -25, 0}
- None - { -25, 0, 25 }
- Slight Right - { 0, 25, 50 }
- Right - { 25, 50, 75}
- Large Right - { 50, 75, 100}
- Extreme Right - {75, 100, 125}

Rules

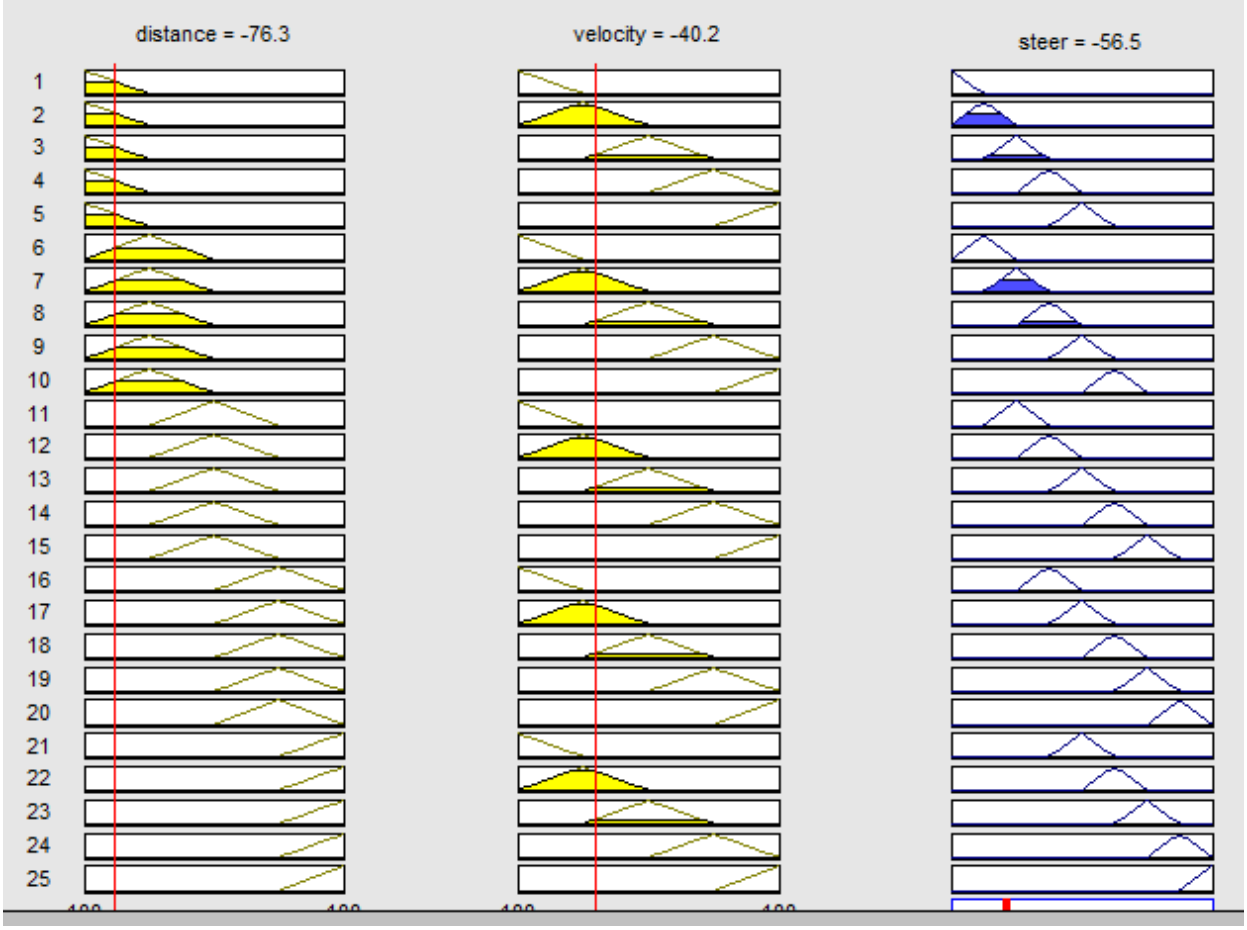
For a fuzzy system, rules are verbal statement formulated using ‘if’ and ‘and’ operator and constructed as statements. For example, If Distance is left, and velocity is left, then steer towards

left is a fuzzy system rule. Using the same approach, rules for the fuzzy controller were devised using the fuzzy associative map (FAM).

Fuzzy Associative Map	Velocity of Line				
Distance of Line	High Left	Left	Zero	Right	High Right
Far Left	Extreme Left	Large Left	Left	Slight Left	None
Left	Large Left	Left	Slight Left	None	Slight Right
Zero	Left	Slight Left	None	Slight Right	Right
Right	Slight Left	None	Slight Right	Right	Large Right
Far Right	None	Slight Right	Right	Large Right	Extreme Right

Once the rules were formulated, they were passed into the fuzzy system and using the Rules Editor in the Matlab, rules were verified and tested for several cases, including the boundary cases.

Rules Editor in Matlab

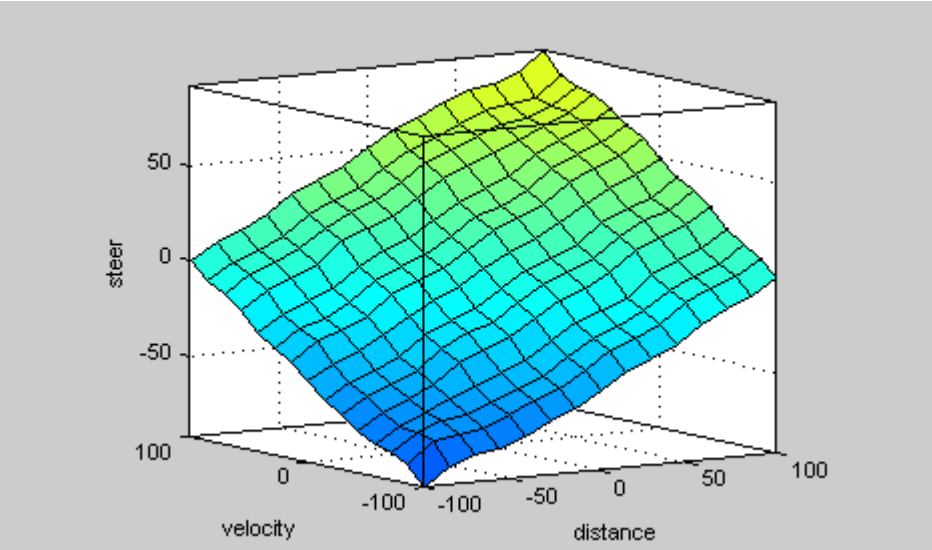


For a total of five input states for distance, and five for velocity, a total of twenty-five (5*5) rules were formulated and tested. The set of rules were exhaustive and covered nearly all the cases. They provided a smooth transition for car approaching towards the line from distance. Also, with the overlapping member functions, the controller reacts in a realistic way for given set of inputs.

Surface

Surface in a fuzzy system represents the transition of the controller from one input to the other and it gives an idea regarding whether the transition would be smooth or will it meet some local minima/maxima. The surface for given controller was smooth, representing smooth transitions without any jiggle effect or local/global maxima/minima for the car.

Surface for the given controller



Defuzzification

Defuzzification is the process by which fuzzy rules; variables are assigned numerical values, producing result in terms of degrees, instead of verbal statements. The defuzzification method used in the system is the Centroid method. This method returns the center of area, along which the shape balances and for triangle, centroid also represents the point where the three medians meet.

Software Design: Proof of Concept

The graphical application was developed with the goal of testing the fuzzy system so developed. Thus, the application was intentionally kept simple. For a car controlled by a fuzzy system, possible moves available for car are forward, left and right, of which forward motion is taken to be constant,

and the left, right motion is controlled by the controller. Neglecting the constant forward motion, the possible motion of car is restricted horizontally. For road, the distance between the road and the car can either be positive or negative, which essentially means that the possible distance is also restricted horizontally. Thus, the road represented by a line moves horizontally across the width of the screen from 0 to 600 {or -300 to 300} and the car represented by the circle follows the motion of the line and adjusts itself on top of the line.

Software used

The fuzzy system designed above was initially coded and tested using the Matlab. The Matlab generates a .fis (fuzzy inference system) file. The graphical application is coded using Java. It uses the jFuzzyLogic library, which is an open source fuzzy logic library implementing industry standards to simplify fuzzy systems developments. (http://jfuzzylogic.sourceforge.net/html/index.html). The jFuzzyLogic library helps in creating fuzzy system by reading in .fcl (fuzzy control language) file and using the library functions one can retrieve information from the fuzzy systems. But as Matlab didn't generate .fcl files, a different fuzzy logic library - qtfuzzylite, (which is a cross-platform, free and open-source Qt-based graphical user interface for fuzzylite. Its goal is to allow one to visually design fuzzy logic controllers and interact with them in real time. (www.fuzzylite.com))Was used.

The qtfuzzylite application allows to export the fuzzy interface system in .fis, .fcl, or .cpp format. Thus, using .fcl file and jFuzzyLogic library (which has an extension for Eclipse) the application for fuzzy controller was developed. The application consists of a panel, (which uses JPanel) and shows a start, stop button in the bottom part of the panel, to start and stop the animation. The center part is where the controller is tested. The top part of the panel consists of manually calculating the steer by providing the input values of distance and the velocity. As the automated part is kept separate from the manual one, the result so obtained in the manual part conforms with the values so obtained in the matlab, and is not mapped to suit the application.

Code organization and structure

The code organization is kept modular and makes use of inheritance and other object oriented concepts. The line and circle, both have their separate classes. The line class inherits from an abstract base shape class. The circle class inherits from the class Fillable, which extends from the base Shape class. Thus, the basic functions are all implemented in the base class and if required overridden in the sub class. Interaction between the Line and Circle was set up in a different class - 'FuzzyControllerPanel'. An instance of this class is created in the main method and then by initiating the thread, the animation starts, in which the line moves from one end to another and has different velocities based on the distance traversed.

For the steering of the circle to follow the line, the distance of the line from the car and the velocity is taken, and mapped appropriately. The input values are then passed to the jFuzzylogic library,

which gives the output value. The output value is then mapped and applied on the circle, which follows the line. Some tuning was required to assure that the circle follows the line as closely as possible. In the application, the line initially is placed in the center of the panel. The circle is also positioned at the center, but due to the language feature, the position coordinates specified for circle were taken as its left side coordinate (as implementing a circle in Java is done using the drawoval method present in the Graphics class, and this method accepts left side coordinate of the oval). This essentially means that instead of positioning the line at center of circle, line is situated at left side of the circle and while the program is running, the circle is steered by the left side and the attempt is to align the left side of the circle with the line.

Application testing

The application so developed is tested using two methods

- Automatic
- Manual

In the Automatic way, a graphical simulation of the application is initiated and the results were recorded. In this, the distance is bounded between the range {-300, 300} and the velocity is in the range of {-5, 5}. The range of steer is {-5, 5}. The line traverses from end of the panel to another, where velocity changes as per the distance travelled across the panel and the circle tries to steer itself along the line.

In the Manual way, user simply provides the input value in the text boxes provided in the top part of the panel and the output is displayed. As it's a direct method, so no mapping for the input or output is done, and the result corresponds with the values obtained using the Matlab.

The manual way was one of the ways of testing the correctness of the system as these are the values which are then mapped and provided to the simulation.

Alternative Strategy

Chasing and Evading method

It consists of two parts. The first part involves the decision to initiate a chase or to evade. The second part involves affecting the chase of evasion - that is, getting the predator to the prey, or having the prey get as far from the predator as possible without getting caught [Bourg, D.M. AI for Game Developers, 2004: 6]. In the case of controlling a racing car, car is chasing the road. Thus, the algorithm implemented will be a chasing algorithm, where car acts like a predator, and is chasing the road (prey).

Software Design

The graphical application developed to test the working of Chase-Evade algorithm follows many of the design principles established earlier for the fuzzy system. In this application, like the previous one, forward motion of the car is taken as constant and hence the movement of car is observed in horizontal direction only. Also, the range of distance covered is taken to be {-300, 300} and that for velocity and steer is taken to be {-5, 5}. But as this is a standalone application, no mapping is required. Also, as previously mentioned the road is represented by a line moving horizontally across the width of the screen from 0 to 600 {or -300 to 300} having different values for the velocity based on the distance and the car is represented by a circle which follows the motion of the line and adjusts itself on top of the line.

Software's used and Code Organization

The alternate technique was coded using Java and makes use of the classes developed for the fuzzy system controller. The car was represented as a circle, and road was represented as a line. Line extends from the abstract base Shape class. Circle extends from the Fillable class, which extends from the base Shape class. The class ChasePanel takes care of the interaction between the line and the circle. It's in this class, where the chase algorithm is applied by taking in account the distance of the car from the road. If the distance of the road is more than the car, then the car speed (in a way it's distance) is increased and if the distance is less, than speed is decreased.

Application Testing

The application developed was tested using the automatic simulation. In this, the user can start, stop the simulation and as the simulation runs one can observe the circle following the line.

Comparison of the two strategies used

The data from both the techniques was collected, in order to compare the two methods and analyze the result. The two controllers are compared on their accuracy and effectiveness using the correlation coefficient. The correlation coefficient is a measure of relation between two quantities. It has its value between -1 to 1. A coefficient of 1 indicates that the two variables are perfectly related in a positive linear sense; a correlation coefficient of -1 indicates that two variables are perfectly related in a negative linear sense, and a correlation coefficient of 0 indicates that there is no linear relationship between the two variables. The standard deviation and the mean of the values are also calculated.

The correlation coefficient is calculated using the inbuilt function in the Microsoft Excel, which uses the Pearson correlation coefficient method to calculate the value. In Pearson's method, correlation coefficient between two variables is defined as the covariance of the two variables divided by the product of their standard deviations [wiki]. The formula is given as -

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$ [wikipedia.com]

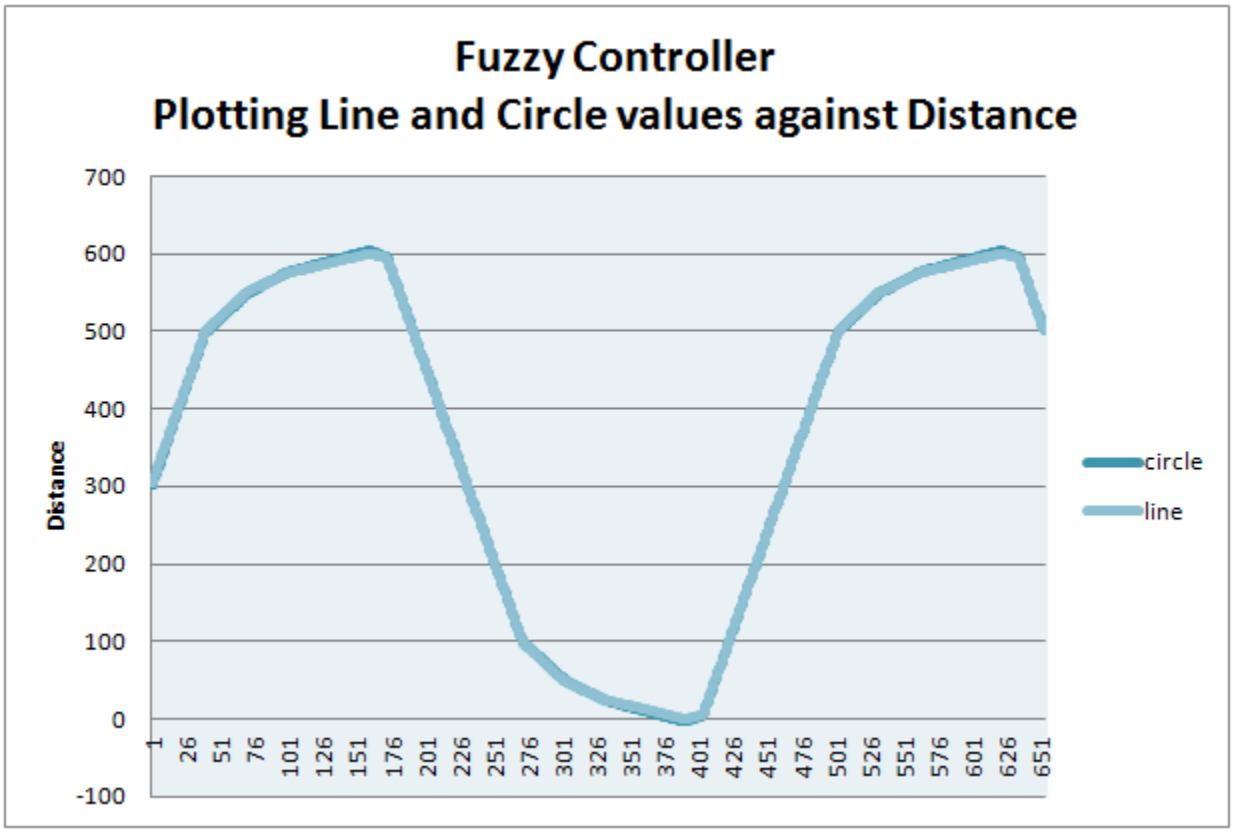
Where r - correlation coefficient

X, Y - two samples for which the coefficient is been calculated

The fuzzy system controller and the Java application were developed to test the accuracy of the system built. In it, the goal was to keep the car as close to the road as possible.

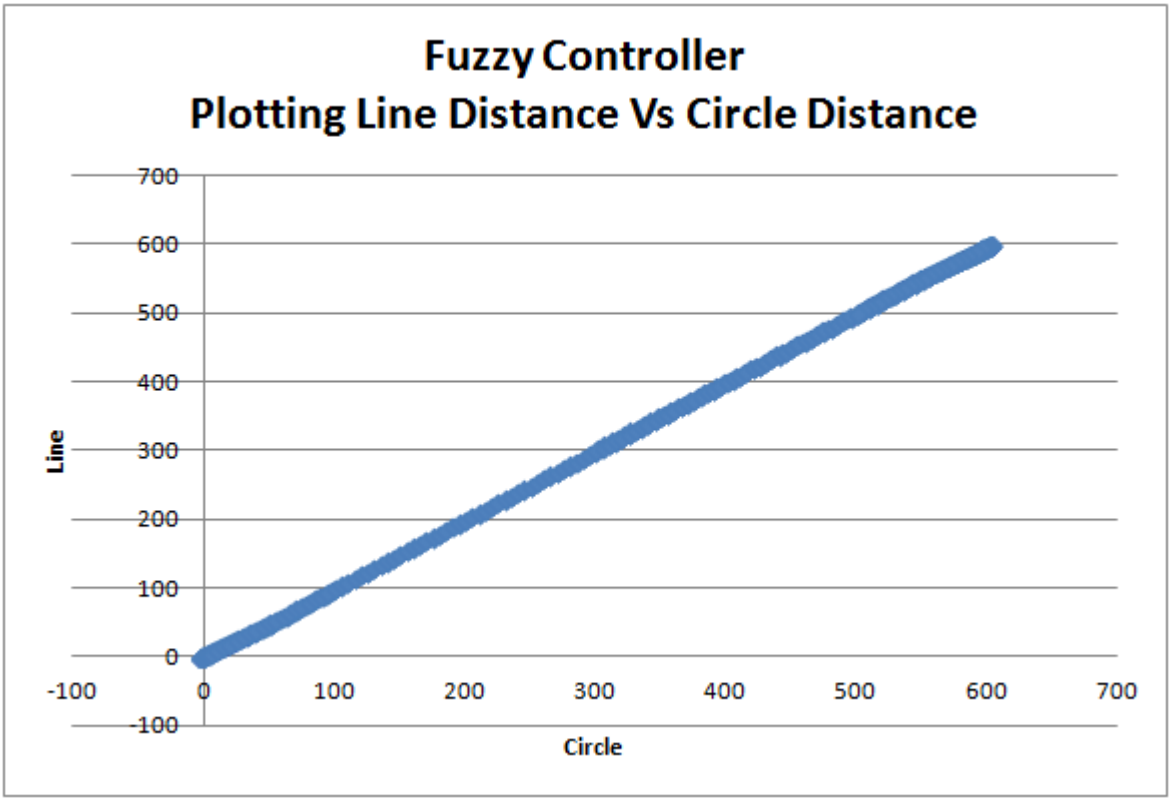
Results obtained using Fuzzy system

Plotting the values for line and circle against the distance traversed, the graph obtained -



It is evident from the above graph that the car follows the movement of the road closely.

Plotting the Line's distance values against Circle's distance, the graph obtained was-



A straight line in the above graph represents that the car and road distances match up to a good extent.

For the Fuzzy Controller -

Standard Deviation:

Line: 221.9585 (average: 368>1735)
Circle: 222.789 (average: 368.1727)

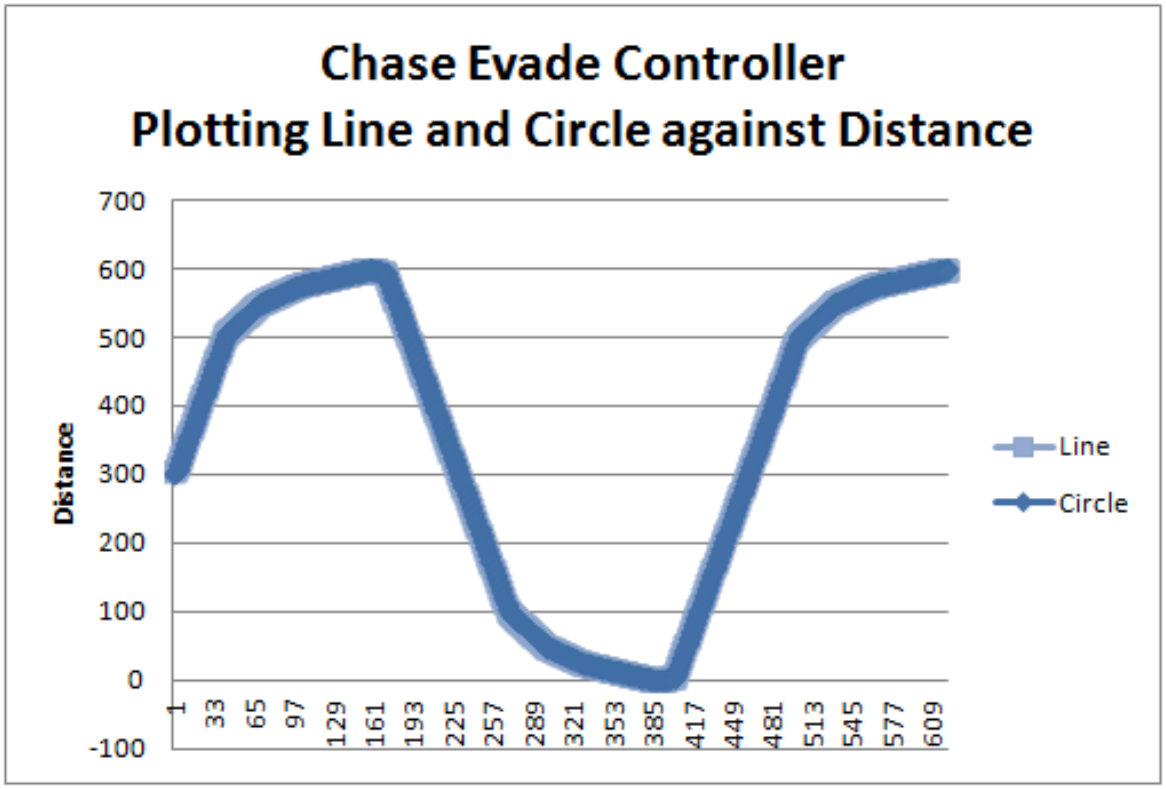
Correlation coefficient calculated using Microsoft Excel: 0.999977

As the correlation coefficient shows the degree of linear dependence between the two variables and in this case it is approximately 1, which means that the controller is linearly dependent on the motion of the road. As the relationship is positive, it just shows that the two variables move into the same direction. A higher value of line corresponds to higher values of circle, and vice versa (stackoverflow.com). Thus, it can be ascertained that the implementation of the controller was successful.

Results obtained using Chase and Evade system

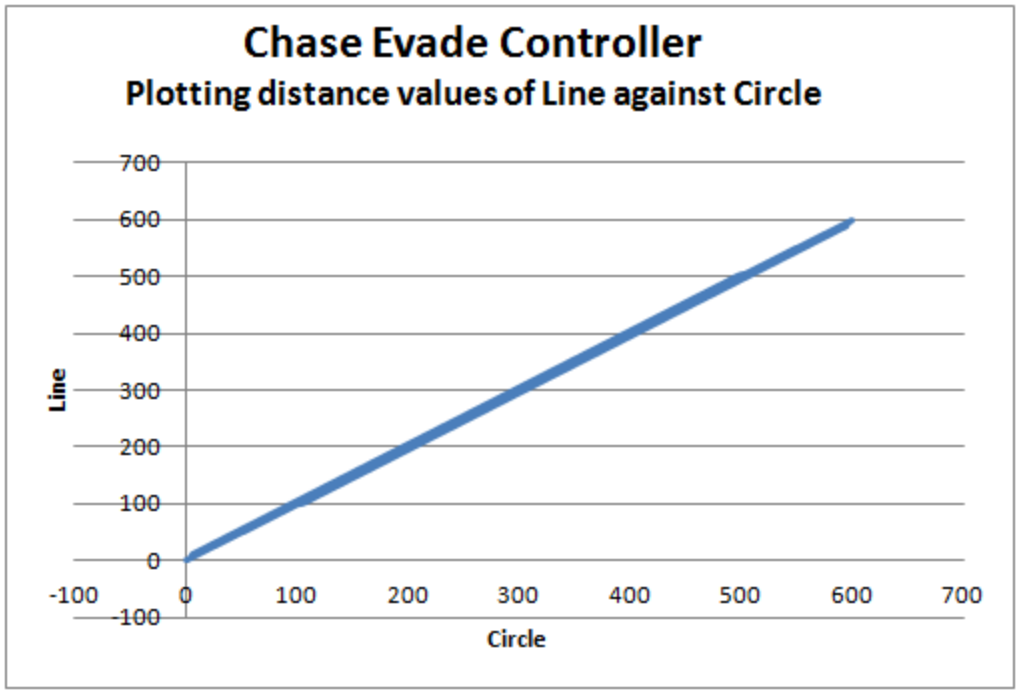
A different approach was used essentially to test the performance of the two strategies and to deduce which strategy is more effective.

Plotting the Distance values for line and circle against the distance traversed, the following graph was obtained –



It is evident from this graph that the car chases the road in a close manner.

Plotting the Line's distance values against Circle's distance, the graph obtained was-



In this case also a straight line was obtained which represents that the car and road distances match up to a good extent.

For Chase - Evade Controller

Standard Deviation -

Line: 223.1456 (average: 356.8744)
Circle: 222.9441 (average: 356.3943)

Correlation coefficient calculated using Microsoft Excel: 0.999901 ~ 1

As the correlation coefficient is near 1, which means that there exist a linear relation between the car and the road, and that the car's distance increases/decreases based on the corresponding movement of the road.

Comparison of the Test Results

Correlation coefficient for fuzzy system: 0.999977

Correlation coefficient for chase system: 0.999901

Test results for both the controllers were almost same, which depicts that both the controller follows the road to a decent accuracy. The correlation coefficient of approx 1, also suggests a linear relation between the car and the road, where the car's steering is based on road's distance and

velocity of the car. Thus, it shows that the implementation of the fuzzy and chase controller was successful.

Conclusion

As the results of the calculations performed on the data sets for two cases were almost similar, it was difficult to judge solely on that criterion. While both strategies performed satisfactorily, it was crucial to look at other details like time taken in implementation, flexibility of the system to accommodate new situations, and adaptability of the given system to suit itself to the newer scenario.

Implementation

Implementing the fuzzy logic controller was undoubtedly tricky as first the controller was coded using Matlab and then it has to be ported to the language in which it was coded. In hindsight, coding the graphical application wasn't much of the challenge, rather formulating sound fuzzy rules, deciding on the membership functions, their shape and range for each of them was. As deciding the rules was an essential process of understanding the system, so spending enough time was essential. But once past that, coding the graphical application wasn't difficult, but finding the right fuzzy library was. Different fuzzy libraries provide different solutions, so finding the one that fits the given requirement was necessary, but time taking as there are numerous fuzzy logic libraries present.

In comparison, implementing the Chase controller was much easier as first, learning's from developing the fuzzy system helped. Second, most of the code for graphics was reused from the fuzzy system's graphical application. Thus, more time was given to understand and implement the chase algorithm.

Flexibility of the System

Flexibility of the software system means the ease with which the given system can be extended in the future to cover new set of requirements like modeling the given system to handle obstacles present on the road.

The fuzzy system is modeled around the distance and the velocity of the road from the car. A third parameter of the obstacle (with two membership function: obstacle is present or not) can be added to the present system. The rules need to be modified, or rather the present case can be then taken to be the one without obstacle and the new rules, for the case when the obstacle is present can be added. As the system right now, has low coupling it is a flexible system and can be extended to accommodate for the changes in the future.

In the chase controller, the car's behavior is tied with that of the road. As car and road are well encapsulated in their separate classes, adding a new parameter would mean either adding a new class or decreasing the cohesion of the system. If a new class is added, then based on the new

interactions, the current algorithm needs to be changed, rather than extending it. Thus, the flexibility of the chase controller is limited and is less than that of the fuzzy system.

Adaptability

Adaptability of the software system means the extent to which the given system can adapt to the changed scenario like what'll be the performance of the given system, or how much it'll degrade if the present controller is applied to the cases where the obstacles are present.

The fuzzy controller is developed to cover wide range of road movements where when the car is closer to the road, it slows down the speed, so as to align itself at the center of the road. For chase controller, no such regulation of speed is present. If nearby an obstacle, chances are more that the chase controller without any speed regulation will collide with it and thus, it is less adaptable to different scenarios in comparison with the fuzzy controller.

The fuzzy controller owing to its fuzzy nature and overlapping membership function performed better in steering the car and can easily be extended and/or adapted. The applicability of such a controller are limitless in game development and hence it was a fruitful exercise as it did helped in not only understanding and implementing fuzzy logic but also allowed to explore and learn about Chasing algorithm

References

- Bourg, D.M., Seeman G., 2004, *AI for Game Developers,* O'Reilly, USA
- Cingolani, Pablo, and Jesús Alcalá-Fdez, 2013 *"jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming"*, [Accessed: 15 January 2014] http://jfuzzylogic.sourceforge.net/html/index.html
- *Fuzzy Control Systems, 2014* [online] http://en.wikipedia.org/wiki/Fuzzy_control_system [Accessed: 15 January 2014]
- Juan Rada-Vilela, *Fuzzylite: a fuzzy logic control library*, 2014, http://www.fuzzylite.com.
- King, D., 2013, *CE1184A - Fuzzy Logic and Fuzzy State Machines,* Exeter: University of Abertay, Dundee, UK
- *Regression and Correlation analysis,* 2014 [online] - http://abyss.uoregon.edu/~js/glossary/correlation.html [Accessed: 15 January 2014]
- Okoh, S., 2002, *Fuzzy Logic,* Calvin College, USA http://www.calvin.edu/~pribeiro/courses/engr315/samples/Final%20Presentation.ppt [Accessed: 15 January 2014]
- *Stackoverflow,* 2014 [online] - http://stackoverflow.com/questions/7631799/what-does-correlation-coefficient-actually-represent [Accessed: 15 January 2014]
- *Pearson Correlaltion Coefficient* [online] - http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient [Accessed: 15 January 2014]