An Assessment of Fuzzy Logic for Collision Avoidance in a Multiple Autonomous Mobile Robot System

Christopher G. Dodds and Albert Y. Zomaya
Parallel Computing Research Laboratory
Department of Electrical and Electronic Engineering
The University of Western Australia
Perth, Western Australia, 6907
dodds_c@ee.uwa.edu.au
zomaya@ee.uwa.edu.au

Abstract

A fuzzy logic approach for collision avoidance in a multiple autonomous mobile robot environment is proposed. Up to four robots are given starting points and destinations they must reach within a hypothetical indoor environment. The fuzzy logic method enables the robots to reach their intended destinations while avoiding stationary and mobile obstacles alike.

1. Introduction

With the increasing demand to explore regions that are either too hazardous or unattainable to humans, the need arises for the development of mobile robots that can navigate autonomously in unstructured and unexplored environments. Many of these situations require several robots in order to complete the task at hand. Unfortunately, sensors used for positional information and collision avoidance are often too imprecise and difficult to interpret due to the complexity of the environment. Therefore a system must be designed that can use this limited information to accomplish the goals set out by the robot, which are: 1) avoid collisions with other objects, and 2) to reach its targeted destination. This requires a control strategy with intelligent decision making capabilities that can judge various situations from multisensor data and act accordingly. It must be able to determine which incoming information are relevant and be able to formulate suitable procedures in order for the robot to safely reach its destination.

In classical motion control, a set of equations are derived to describe the vehicle and its surroundings. These are then implemented with feedback control laws in order to calculate the robot's positional data. Several attempts have been made using this type of approach. For instance, Shibata (1993) proposed a hierarchical intelligent control method and Brooks (1987) developed a bottom-up approach using state transition machines. Unfortunately both of these methods were unable to tolerate complex situations. Also classical motion control approaches are commonly unable to handle small perturbations in the system, such as changes in the environment or the vehicle's speed. This is an important setback since these are common occurrences in the real world. What is required is a control algorithm that is able to adapt to various situations and still reliably get the robot to is intended destination.

One control methodology that is recently gaining acceptance is fuzzy logic. Martinez (1993) was successfully able to navigate a mobile robot collision free through an unknown, semi-structured environment using fuzzy logic. However the real world often presents more complex situations that were not present in this environment.

The aim of this paper is to test fuzzy logic in a multi-agent environment that includes both stationary and moving obstacles. The user of the system is able to select the starting and destination points for up to four robots. The additional robots in the system provide for the moving obstacles. In addition, the user may place up to ten stationary obstacles in the room. Allowing the user such control enables the opportunity to examine the flexibility and robustness of fuzzy logic.

2. Fuzzy Logic

2.1 Introduction to Fuzzy Logic

If a person were to control the robots directly, it is not hard to imagine that the destinations would be achieved, collision free, without much of a problem. After all, many people do this everyday when they drive to work. Then one has to wonder why it is so difficult for classical control theory, with its complex algorithms, to perform the same task. One idea is that these type of control systems perform unnecessary precise calculations. Certainly a person does not have to be so precise when controlling their car. This point of view suggests two things: 1) while the system's input and output should be accurate, it does not have to be extremely accurate, and 2) the data from the system's input is not being used effectively. The first point says that the level of accuracy

in the system reaches a point in which any further increase in accuracy would be redundant and could adversely affect system performance. The second point suggests that a different control strategy should be used.

In 1965, Lotfi Zadeh published the paper "Fuzzy Sets" which challenged traditional probabilistic theory (McNeill and Freiberger 1993). Instead of trying to determine whether or not an event occurs, fuzzy theory determines to what degree an event occurs. Instead of categorizing events in 'black and white' fashion, fuzzy theory blurs these strict cut-off points and puts values into one or more highly descriptive groups. For example, take the case of a person's age. Probability theory may define a person as being 'old' for ages 65 and above and 'not old' otherwise. On the other hand fuzzy theory is not restricted to just two groups. It may have any number of groups, depending on how many the designer sees fit. For instance it might say that the age of 65 is 'somewhat old' and 'partly middle aged'. The use of these descriptive linguistics enables a system to have a more precise description and a better understanding of its inputs and outputs.

2.2. Fuzzy Theory

Figure 2.1 illustrates the basic idea behind fuzzy logic. Input value A is taken from domain X and is assigned a fuzzy value between zero and one. The fuzzy value is determined by a fuzzy association memory (FAM) matrix, which maps input data to fuzzy values. How the two sets of values correlate between each other is completely decided by the designer of the system. From the example of Figure 2.1, value A is assigned a fuzzy value of B = 3/4, which can be thought of as 'a particular action occurred to a degree of 3/4'. Since an action either occurs or it does not, it can also be said from the same data that 'a particular action did NOT occur to a degree of 1/4'. This is known as the compliment of fuzzy value B, or B^C.

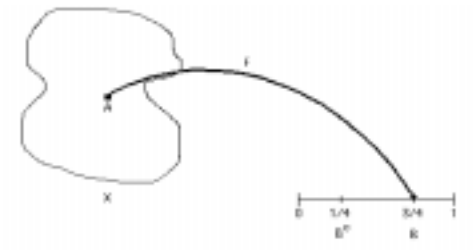

Figure 2.1. Function f Maps Value A From Domain X to Fuzzy Value B.

Probability works in much the same way, except for one major distinction. Instead of mapping to what degree an action occurred, probability maps 'the likelihood an action will occur'. They are, in essence, two different ways of observing the same action. This difference can be seen better through an example. Say that a person has been told that he has a fuzzy value and probability value of 0.7 that his dinner tonight will be good. As more information comes in (smelling the food being cooked, eating appetizers), the probability value is likely to change. This change continues until the action is complete (after dessert) at which time the probability will be restricted to either 0 or 1. But throughout the dinner, the fuzzy value remains at 0.7, even after dessert. This is because the fuzzy value is describing to what degree the meal is good, not predicting if it will be good or not.

But due to the virtue of its definition, the fuzzy values assigned are not known with certainty. Hence if B is not known with certainty, then neither is its compliment, B^C. This produces an overlap and underlay that do not exist in probability theory. In other words, Aristotle's laws of noncontradiction and excluded middle do not hold in fuzzy logic. This can be stated as follows:

$$B \cap B^C \neq \varnothing \qquad (2\text{-}1)$$

$$B \cup B^C \neq 1 \qquad (2\text{-}2)$$

These two equations violate the basic rules that probability theory is based on. Hence many of the definitions in probability theory do not hold in fuzzy theory. For example, the definition for set product and union in fuzzy logic are given by the following equations:

$$B \cap B^C = \min(B, B^C) \qquad (2\text{-}3)$$

$$B \cup B^C = \max(B, B^C) \qquad (2\text{-}4)$$

Using the example given in Figure 3.1, the following evaluations can be made:

$$B = 3/4$$

$$B^C = 1/4$$

$$B \cap B^C = 1/4$$

$$B \cup B^C = 3/4$$

3. Simulation Description

A program written in C was developed to test the reliability of fuzzy logic on collision avoidance for stationary and moving obstacles alike. The operating environment is a hypothetical indoor room with a hard, smooth floor surface. This type of surface ensures good maneuverability for the robot. The room is divided into X-Y coordinates 1 through 90. The heading of each robot ranges between 0 and 360 degrees with 0 being

referenced to the right side of the room. This layout is shown in Figure 3.1.

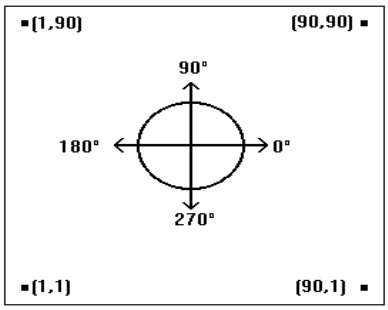

Figure 3.1. Specifics of Robot's Domain Including Co-ordinate and Heading Information.

This simulation supports up to four robots at any one time. The robots operate independently of one another in a decentralized system. This is in contrast to a centralized system in which a single master robot dictates the actions of several slave robots. A decentralized system has the advantage that it does not rely heavily on the functionality of the master robot, adding a level of fault tolerance to the entire system.

The robots themselves are square in shape with each side being one coordinate in length. Its current position in the room is referenced from the vehicle's centre, with each of its sides stretching out one half a coordinate. It is assumed the robots are able to determine its current position and heading at all times. This is easily realized with the use of wheel encoders and a direction determining device such as a gyroscope. The vehicles are able to move forwards or backwards and may reach a top speed of one coordinate per second.

During simulation runtime the current position, heading, and velocity for each robot are calculated and displayed. While the heading and velocity values are floating point values, the positions of the robots and the obstacles are rounded to integer values. Therefore two robots or a robot and an obstacle are able to get within two obstacle lengths of one another before colliding.

4. Obstacle Detection

To avoid a collision with any of the other robots, each robot must know the current status of the other vehicles. Therefore a global communication scheme needs to be implemented in order for the robots to share information. By exchanging current data with the other vehicles a robot can determine the next several positions of any of the vehicles. That way if it knows a collision with another robot is about to occur, it can act accordingly to prevent it. It accomplishes this by using the current position, velocity, and heading values of each robot.

Determining another robot's future positions, or traversing vector, is calculated with the following equations:

$$X(i)_n = \sum_{i=1}^{n} (V(i) \cdot \cos(\theta(i))) + X(i) \qquad (4\text{-}1)$$

$$Y(i)_n = \sum_{i=1}^{n} (V(i) \cdot \sin(\theta(i))) + Y(i) \qquad (4\text{-}2)$$

where $X(i)_n$ and $Y(i)_n$ are the n^{th} X and Y positions of robot i, V(i) is the velocity of robot i, $\theta(i)$ is robot i's heading, and X(i) and Y(i) are the current X and Y positions of robot i.

In addition to the other vehicles, each robot needs to be able to sense the stationary obstacles as well. To determine the location of any of these obstacles each robot is equipped with three ultrasonic sensors. One sensor is pointed directly ahead with the other two facing 90 degrees off on either side (Figure 4.1). These proximity sensors are able to determine the distance to an upcoming obstacle by sending out an ultrasonic pulse and measuring the time it takes for the reflection to be received. Each sensor has a range of four coordinates.

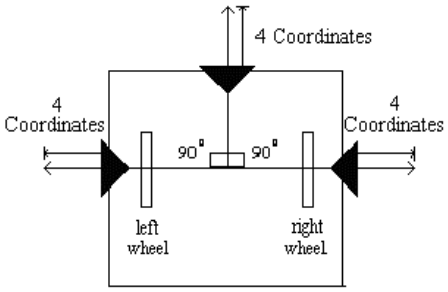

Figure 4.1. Configuration and Range of Ultrasonic Sensors on the Robot.

5. Design of a Fuzzy Controller

While the fuzzy controller is easy to control and understand, it can be quite difficult to design. There is no structured design theory to follow, therefore a good working knowledge of the system is required. When designing the controller, three basic tasks need to be accomplished.

The first of which is to determine the system's linguistic variables. These variables are the input and output signals used to control the system properly. The designer needs to decide what the important signals are and how to transform these signals into linguistic descriptions. In this simulation, the input signals are from the three ultrasonic sensors and the position, heading, and velocity data from the other robots. The output signals are the vehicle's speed and heading.

The second step in the design is the production of the fuzzy associative memories (FAM). A FAM rule maps the association of the system's inputs to the outputs. Each one is structured in an if...then manner and uses linguistic descriptions to describe the data. For instance in the case of stationary obstacles, a rule might be if the middle sonar reading is 'positive medium' and the left sonar reading is 'positive large', then the change in heading is positive small (to the left) and the change in velocity is negative small. This simulation uses the seven descriptions listed at the bottom of Table 5.1. Incorporating these with the three sonar measurements gives a total number of 64 if...then rules for stationary obstacles. A short-hand tabulation of the 64 rules is also listed in Table 5.1.

if (MS = PL)	**then** (H = ZZ) and (V = PL)
if (MS = PM) and (MS >= RS) and (MS >= LS)	**then** (H = ZZ) and (V = NS)
if (MS = PS) and (MS >= RS) and (MS >= LS)	**then** (H = ZZ) and (V = NM)
if (MS = ZZ) and (MS >= RS) and (MS >= LS)	**then** (H = ZZ) and (V = NL)
if (MS = PM) and (RS > MS) and (RS >= LS)	**then** (H = NS) and (V = PM)
if (MS = PS) and (RS > MS) and (RS >= LS)	**then** (H = NM) and (V = PS)
if (MS = ZZ) and (RS > MS) and (RS >= LS)	**then** (H =NL) and (V = ZZ)
if (MS = PM) and (LS > MS) and (LS > RS)	**then** (H = PS) and (V = PM)
if (MS = PS) and (LS > MS) and (LS > RS)	**then** (H = PM) and (V = PS)
if (MS = ZZ) and (LS > MS) and (LS > RS)	**then** (H = PL) and (V = ZZ)
MS - middle sonar RS - right sonar LS - left sonar PS - positive small PM - positive medium PL - positive large NS - negative small NM - negative medium NL - negative large ZZ - zero	

Table 5.1. Short-hand List of the Fuzzy Associative Memories with Their Linguistic Adjectives.

A similar table is also made for the moving obstacles. It is nearly identical to Table 5.1 except that changes in the vehicle's heading and velocity are more abrupt. This is due to the simple fact that a collision with two robots will occur faster than a collision with a robot and a stationary obstacle.

The final step in the process is the development of the membership functions. In other words, assigning all of the input and output values to membership groups. Each value is placed in one or more linguistic groups and given a degree of membership. The degree of membership states how much a particular value belongs to a certain group. These values typically range from zero to one, with one being 'full membership'. The plot for this experiment's sonar membership function is shown in Figure 5.1. Note that all of the groups overlap one another, enabling some values to belong to more than one group. Research has shown that it is generally a good idea to have these groups overlap by 25% (Kosko 1993).

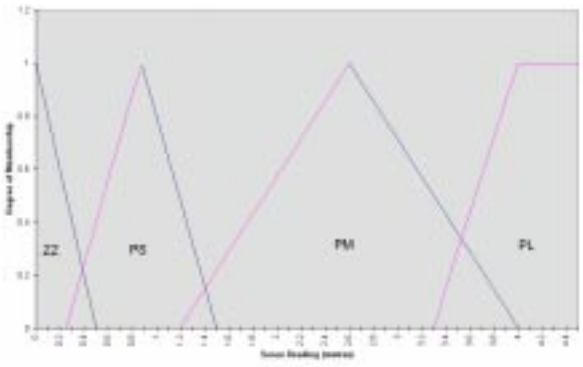

Figure 5.1. Membership Function for the Sonar Readings of Stationary Obstacles.

During normal operation of the system, sensor values are read and 'fuzzified' into its membership groups. The if...then rules are then evaluated to produce fuzzy output results. These values then must be 'defuzzified' to an analog value in order to adjust the output. One method of defuzzification is to use the FAM correlation-minimum inference procedure, or center of gravity method. The fuzzy centroid, $\overline{B}$, is calculated by the equation: (Kosko 1992)

$$\overline{B} = \frac{\sum_{j=1}^{P} y_j \cdot m_B(y_j)}{\sum_{j=1}^{P} m_B(y_j)} \qquad (5\text{-}1)$$

where y is the output value which corresponds to the membership function with a value of one, and m is the minimum degree of membership of all the inputs. An example of this method of defuzzification is shown graphically in Figure 5.2. Here the values MS = 1.45, RS = 0.8 , and LS = 1.15 result in a fuzzy centred value of 0.51 for velocity and 0.0 for heading. These values are then added to the current readings which results in the system's new output.

When the robot first starts, it calculates the angle it must turn in order to face the destination. It immediately makes this its current heading and tries to head straight towards its target point. Next it reads its sonars to determine if there are any stationary obstacles in the way. If so, it adjusts its heading and velocity accordingly. After doing this it then receives data from the other robots and determines if a collision is about to occur. Once again the heading and velocity are adjusted according to the data. Finally the robot moves for one second based on its adjusted heading and velocity. The entire procedure is then repeated.

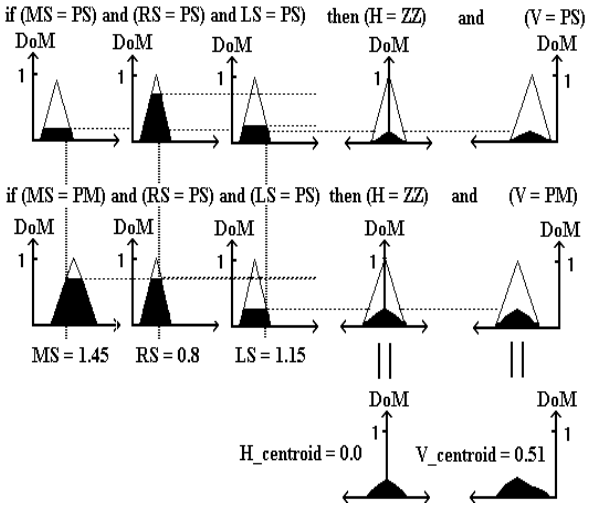

Figure 5.2. Calculation of Output Adjustment by the Max-prod Method and Defuzzification.

6. Results

Several test runs of the simulation revealed that the robot exhibited intelligent behavior in avoiding the obstacles and robots and was able to reach the destination every time using nearly the shortest path possible. This is quite impressive considering the low amount of data bandwidth that is used.

Several types of configurations were used when testing system. The ability of the controller to perform successfully on a variety of problems exhibits the adaptability that fuzzy logic can provide. This is a highly desirable aspect since the unpredictability of real world situations will require such adaptive characteristics.

Figure 6.1 shows a synaptic-vector histogram of which if...then rules were used for several test runs of the simulation. It can be seen that H = ZZ, V = PL is by far the most frequently used output. The fact that the equilibrium rule was the most widely used shows the quick adaptive nature of fuzzy logic. In other words the controller was able to find an efficient solution and it was able to find it rather quickly. This is very important for situations in which time and efficiency are important. It can also be seen that some of the if…then rules are not even used. This point highlights the robustness of the system. Hence the design of the controller does not have to be perfect, or even complete, for it to work properly. But on the other hand if…then rules that are rarely used could be quite difficult to debug.

Figure 6.2 shows the affect the number of obstacles, moving and stationary, has on the number of iterations required to complete the task. As expected, the number of rules increased with the number of obstacles, but this increase is surprisingly small. Again this displays the adaptive nature of fuzzy logic in that the controller is able to quickly find a solution, no matter how many obstacles are used.

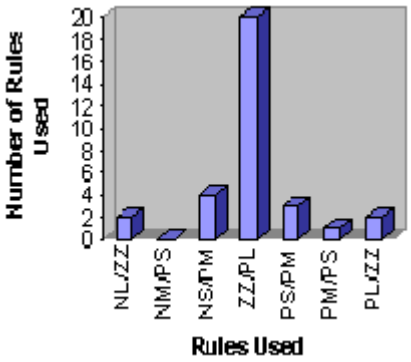

Figure 6.1. Synaptic-vector Diagram of Fuzzy Rule Base.

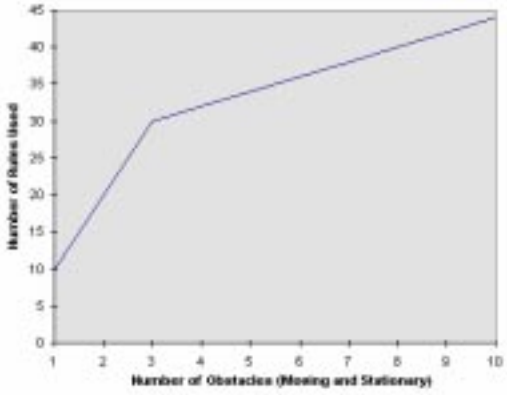

Figure 6.2. Effect the Number of Obstacles has on the Number of Rules Used.

While this particular system was successful for four robots, increasing this number any further tended to cause it to run into problems. With five or more robots, strange and complicated configurations can be made. But as long as the initial setups were relatively 'normal', the simulation was able to successfully handle up to ten robots. This fact suggests the problem does not lie in the fuzzy logic itself but rather in the collision avoidance algorithm used. Again this exemplifies the importance of having good previous experience when designing a fuzzy system. Perhaps a more complicated algorithm, such as the one suggested in by Nijhuis et al. (1992) would provide for better results.

7. CONCLUSIONS

This paper presented a fuzzy logic control scheme in an environment that contained both stationary and moving obstacles alike. Overall, the fuzzy logic controller proved to be a satisfactory control strategy for a sensor-based method of navigation. It provides an easy to understand controller without having to define an

analytical control model. This is due to the fact that approximate solutions to the problem are acceptable. It is also easy to debug since specific if…then rules can simply be manipulated to correct a part that is working unsatisfactorily or improve performance.

One major disadvantage is the fact there is no straightforward way to design a fuzzy control system. This demands that the designer has previous experience and a good working knowledge of the situation. Another disadvantage is the large number of if…then rules needed for the rule base. The amount of rules required for a given system is based on the expression:

$$N = A^I \qquad (7\text{-}1)$$

where N is the number of if…then rules, A is the number of linguistic adjectives in the system and I is the number of inputs in the system. For instance, to avoid stationary obstacles three input sonars and four linguistic adjectives are used. This results in a total of $4^3 = 64$ if…then rules. However if a fourth sensor were added to this simulation, the number of possible rules would jump from 64 to 256. Likewise, if a fifth linguistic adjective were used, a system with three sensors would have 125 rules. But it is usually quite easy to summarize these rules by writing them in shorthand form just as in Table 5.1.

Due to the nature of fuzzy logic the solution presented in this paper is just one out of an unlimited number. As more experience is gained, improvements to the controller can be made. It is hoped the design presented will eventually be implemented in a physical system.

8. References

1. Brooks, R.A. (1987). "A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control", in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 106-110.
2. Brown, M., Fraser, R., Harris, C.J., and Moore, C.J. (1991). "Intelligent Self-Organizing Controllers for Autonomous Guided Vehicles: Comparative Aspects of Fuzzy Logic and Neural Nets", in *International Conference on Control '91*, Vol. 1, Edinburgh, UK, March 25 - 28, pp. 134-139.
3. Dimirouski, G.M., Iliev, O.L., and Burzeuski, V.N. (1994). "Fuzzy-Logic Control Algorithms Navigation of a Factory Vehicle-Robot", in *International Conference on Control '94*, Vol. 1, Coventry, UK, March 21-24, pp. 282–287.
4. Güracak, H.B., de Sam Lazaro, A. (1995). "Fuzzy Logic and Position Sensing for Precision Assembly", in *Journal of Robotic Systems*, Vol. 12, No. 2, pp. 135-146.
5. Ishikawa, S. (1991). "A Method of Indoor Mobile Robot Navigation by Using Fuzzy Control", in I*ntelligence for Mechanical Systems, Proceedings IROS '91*, Vol.2, Osaka, Japan, November 3-5, pp. 1013-1018.
6. Kosko, B. (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, Englewood Cliffs, NJ.
7. McNeill, D., Freiberger, R. (1993). F*uzzy Logic*, Simon & Schuster, New York, 1993.
8. Martinez, A., Tunstel, E., and Jumshidi, M. (1993). "Fuzzy Logic Based Collision Avoidance For a Mobile Robot", in *The Third International Conference on Industrial Fuzzy Control and Intelligent Systems*, December 1-3, Houston, Texas, pp. 66-69.
9. Nijhuis, J., Neuber, S., Heller, J., and Sponnemann, J. (1992), "Evaluation of Fuzzy and Neural Vehicle Control", in *Computer Systems and Software Engineering, CompEuro 1992 Proceedings*., May 4-8, The Hague, Netherlands, pp. 447-452.
10. Shibata, T., Fukuda, T. (1993). "Coordinative Behavior by Genetic Algorithm and Fuzzy in Evolutionary Multi-Agent System", in *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 1, May 2-6, Atlanta, Georgia, pp. 760-765.
11. Vestli, S., Tschichold, N., Adams, M., and Sulzberger, S. (1993). "Integration of Path Planning, Sensing, and Control in Mobile Robotics", in *IEEE International Conference on Robotics and Automation*. Vol. 3, May 2-6, Atlanta, Georgia, pp. 243-248.