Fuzzy Logic in Mobile Robot Navigation

Carlos E. Torchia

17 November 2008

Table of contents

List of figures

Abstract

The human-like cognition of fuzzy logic makes the use of fuzzy logic a good candidate for motion-planning during robot navigation. Other methods of robot navigation often are difficult to implement or computationally expensive. The fuzzy reactive control method uses a fuzzy logic controller that allows the robot to react in real time to moving obstacles and to head towards its goal. The agoraphilic algorithm uses an artificial potential field coupled with a fuzzy logic controller that helps place more importance on force vectors in the direction of the goal. The data-driven learning algorithm approach uses a data-driven learning algorithm to learn what the best fuzzy rule base is in order to minimize error in navigation. The genetic-fuzzy approach is similar, but uses a genetic algorithm to maximize the efficiency of the path taken by the robot. This complements the reactive control method so that the robot can adjust to new environments. Robot navigation is a complex process that models the navigation of an intelligent organism.

Introduction

A human can navigate around a cluttered environment by using instinct and approximate reasoning rather easily, yet a mobile robot using a numerical model of its environment does not have nearly as much success. Indeed, humans have very desirable methods of cognition. We can process huge amounts of information from sensory receptors and perform actions such as walking and running without any computation of a numerical model of the environment. This fact is the biggest motivation for designing automated systems that use methods of cognition similar to those used in human reasoning. Much investigation has been done in the area of using human-like methods of cognition for the navigation of a mobile robot (Pradhan et al. 2008). In this paper, I discuss various successful methods of navigation using fuzzy logic and fuzzy inference[1] that have been described by several authors.

In controlling low-level processes (i.e. stabilizing internal temperature),

1 *Fuzzy logic* refers to a system of logic that is based on the idea that things can be somewhere between true and false. *Fuzzy inference* is the inference of facts using this logic.

mathematical models are often used (Bekey 2005). PID controllers are also often used successfully. However, in higher-level applications, such as observing and making decisions in a dynamic environment, numerical models may either be very difficult to compute or they may not even exist. Therefore, expert systems[2] that use human-like cognition are often used instead (Karray and De Silva 2004).

Expert systems used in practical applications often use a method called *fuzzy control*, which is based on the theory of fuzzy logic. Fuzzy control consists of making approximate inferences by consulting a fuzzy rule base, which is a set of IF-THEN rules (Karray and De Silva 2004).[3] The first fuzzy logic controller (FLC) was invented by Ebrahim Mamdani and Sedrak Assilian in 1973. It was successfully used to control the pressure and piston speed of an engine (McNeill and Freidberger 1993). Since then, the method of fuzzy inference used by Mamdani's and Assilian's controller, dubbed *Mamdani-type inference*, has been widely used in fuzzy logic controllers. As I discuss in this paper, these controllers can be used for robot navigation.

Indeed, fuzzy logic has been particularly successful in robot navigation. Robot navigation is defined as the task of finding the best path to take by a mobile robot in order to avoid obstacles and reach a goal (Bekey 2005). In idealized conditions, such as in a smooth terrain and static environment, this can be done by using computer vision or wheel odometry (Bekey 2005). However, in more complex environments, these are not enough, and more complicated methods are needed. Many such methods exist, including non-fuzzy ones. A few common non-fuzzy methods will be described here.

One method used for robot navigation is the artificial potential field (APF) method. The APF method essentially consists of creating an imaginary potential field or an imaginary force field[4] that acts on the robot so that the robot will be repelled by its

2 A computerized decision-maker that uses some form of reason based on knowledge.

3 *Fuzzy* refers to the fact that the logic is approximate, i.e. not precise. For an explanation of *fuzzy logic* and *fuzzy inference*, the reader is encouraged to read their respective entries in the glossary on page 16 of this paper. But basically, a *fuzzy controller* (or *fuzzy logic controller*) is something that takes values like temperature as input, and it outputs values like speed, but it does so using a model of approximate reasoning that consists of converting quantites (like “5 meters”) to and from hedges (like “tall”). All the controllers described in this report use a set of rules which are used to deduce the correct decision. The set of rules that a fuzzy controller uses is called its *fuzzy rule base*.

4 See *potential field* and *force field* in the glossary. The APF method uses a force field instead of a potential

obstacles and attracted by its goal, depending on where it is. However, according to Xu et al., this can cause the robot to get trapped in places where the potential has minimum value far from the goal (i.e. at *local minima*) or oscillate near multiple obstacles (1998). Also, according to Pratihar et al., a new force field has to be generated each time the robot moves, which makes the APF method computationally expensive to use (1999).

Another method commonly used in navigation uses some variant of the distance transform (DT) algorithm,[5] which is slightly different than the APF method. In the DT algorithm, a map is used of the environment, and distances are detected and placed in an array successively as the algorithm progresses away from where the robot is located; the path to the goal from the robot's position is given by the path that maximizes how quickly the distance changes from cell to cell (Gupta and Riordan 2004). However, according to Gupta and Riordan, the conventional DT algorithm does not control relative robot-obstacle speed (2004). Also, optimization of the robot's path is inhibited due to incomplete sensory information (Gupta and Riordan 2004). Gupta and Riordan offer a version of the DT algorithm that uses fuzzy inference (2004). However, a major inherent weakness of the DT algorithm is that it cannot handle a dynamic environment (Gupta and Riordan 2004), and therefore will not be considered in this paper.

Various methods from soft computing (i.e. approximate reasoning) have been used in robot navigation, as summarized by Pradhan et al. (2008). Of these, fuzzy logic has been shown to be successful by several studies. Of the methods described in all such studies, I will discuss fuzzy reactive control, the agoraphilic algorithm, the data-driven learning algorithm approach, and the genetic-fuzzy approach. These methods all use approximate reasoning. As such, they model the intelligence of animals and therefore have potential to solve the problem of robot navigation. However, I will not discuss how they are implemented in the hardware or software of the robot, and I will not discuss the mathematical theory behind them. I will only discuss how it is that they essentially function with respect to the fuzzy logic used in the fuzzy controller used by a robot.

---

field because the net force vector of the robot is directly related to the speed and orientation of the mobile robot's wheels, so force fields are easier to use for the physical movement of the robot.

5 The term *algorithm* will be used throughout this paper. It refers to a set of steps that a computer (which is basically the brain of a robot) executes in order to complete a task. There are infinitely many varieties of algorithms that a computer can execute. In this case, the DT algorithm is a set of steps executed by the robot's “brain” in order to plan a path to the goal.

1. Fuzzy reactive control

In a study by Xu et al., the fuzzy reactive control method for robot navigation was described (1998). It consists of real-time inference from rules in a fuzzy rule base that cause the robot to directly react to a dynamic environment.

A robot using fuzzy reactive control to navigate in an environment would do so as follows. When there are no obstacles in sight, the robot simply steers towards its target. If there are obstacles in sight, then the obstacles contribute a repulsive force on the robot and the target contributes an attractive force (Xu et al. 1998). This makes fuzzy reactive control a bit like the APF method. However, this is all done using a fuzzy logic controller that takes the distance from the obstacles (dr, dc, and dl) and target orientation (tr) as input, and as output the fuzzy controller determines the steering angle (sa) for the robot as well as the speed variation (acceleration) for the robot (dsp) (Xu et al. 1998). Figure 1 provides graphs of the fuzzy membership functions for the linguistic variables that describe the inputs and outputs to the controller, as well as the units in which they are measured.

In the fuzzy reactive control method as it was implemented in the study of Xu et al. (1998), a fuzzy rule base was used for the inferences done in determining the steering angle or speed variation given the obstacle distance information and goal direction. This rule base is provided in Table 1, and was generated by human driving experience. Essentially, to determine the outputs of the controller from the inputs, a fuzzification-defuzzification process is used.[6] Xu et al. used the conjunction (minimum) operation for the inferences of the individual rules, then the disjunction operation for the aggregation of these inferences is used, and then the centroid method is used for the defuzzification of the output variable (1998). Fuzzification was done by simply finding the membership values[7] of the controller inputs (Xu et al. 1998). Therefore, the process used was actually a

6 *Fuzzification-defuzzification* refers to the process that the fuzzy controller uses to convert a numerical value to and from a fuzzy value. First the controller fuzzifies some quantity, like distance in metres. Then the controller does some reasoning to reach a conclusion, such as “turn left, and step on it.” From this conclusion, the controller has to make precise what the actual decision is. Therefore, the conclusion will be defuzzified into something like, “turn left 40 degrees, accelerate at 1 m/s/s.” That's fuzzy control.

7 A membership value is the value of a membership function (see *membership function* in the glossary) for

variant of Mamdani-type fuzzy inference.

Using this method of fuzzy reasoning, the robot is able to avoid an obstacle of reasonable size and flatness. However, Xu et al. note that when an obstacle is quite wide and concave, the robot will get trapped by it or wander back and forth in front of it (1998). This is because the the robot's attraction to the target interferes with the fuzzy control for avoiding obstacles (Xu et al. 1998). To remedy this problem, the target angle is artificially and temporarily changed to allow the robot to escape the trap (this is called *target switching*) (Xu et al. 1998).

According to Xu et al., the fuzzy reactive control scheme was implemented in a Nomad 200, and experiments were conducted that confirmed the efficiency and robustness of this method (1998).

Another study that used a similar fuzzy reactive control scheme to guide a robot to its target and avoid obstacles was done by Pradhan et al. (2008). Again, an FLC was used with which the robot avoids obstacles and heads towards a target, but the inputs to the controller were the distance to the obstacle on the left of the robot (LD), the distance to the obstacle on the right of the robot

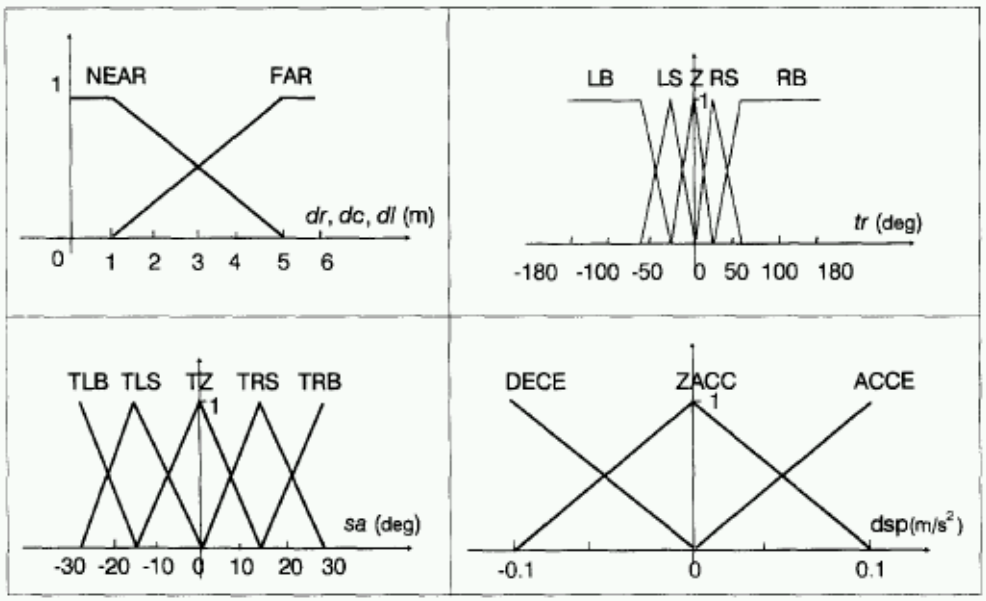


Fig. 1: Membership functions for dr, dc, dl; tr; sa; and dsp. (Xu et al 1998)

some inpu r a given input. For example, we could say that 5 metres is 0.4 in the set of very far things, which means that 5 metres is somewhere between very far and not very far. If it was 0 in the set of very far things, it would be not very far at all. If it was 1 in the set of very far things, it would be totally very far.

Rule	dr	dc	dl	tr	sa	dsp
1	Far	Far	Far	LB	TLB	ACCE
2	Far	Far	Far	LS	TLS	ACCE
3	Far	Far	Far	Z	TZ	ACCE
4	Far	Far	Far	RS	TRS	ACCE
5	Far	Far	Far	RB	TRB	ACCE
6	Far	Far	Near	LB	TZ	ZACC
7	Far	Far	Near	LS	TZ	ZACC
8	Far	Far	Near	Z	TZ	ZACC
9	Far	Far	Near	RS	TRS	ZACC
10	Far	Far	Near	RB	TRB	DECE
11	Far	Near	Near	LB	TRS	ZACC
12	Far	Near	Near	LS	TRS	ZACC
13	Far	Near	Near	Z	TRS	ZACC
14	Far	Near	Near	RS	TRB	DECE
15	Far	Near	Near	RB	TRB	DECE
16	Near	Near	Near	LB	TLB	DECE

Table 1: Fuzzy rule base for the study of Xu et al (1998)
(There were actually 40 rules in total, but I list only 16 for brevity.)
In this fuzzy rule base, for example, consider Rule 1. It states:
"If right-distance is far, center-distance is far, left-distance is far, and the target is left-back, then turn left-back and accelerate." ZACC means "zero acceleration."

(FD), and heading angle (HA), and the outputs to the controller were right wheel and left wheel velocity (RV and LV respectively) (Pradhan et al. 2008). This is essentially the same input/output combination as that of the FLC used in the study of Xu et al. (1998), except acceleration is not explicitly controlled here.[8]

The method of fuzzy inference used by the FLC in the study of Pradhan et al. (2008) was the same as that in the study of Xu et al. (1998) described above, except that the fuzzy rule base was slightly different, because the controller outputs were explicitly different (though functionally equivalent). Also, the fuzzy rule base here was derived from common sense instead of human driving experience (Pradhan et al. 2008).

The FLC in the study of Pradhan et al. (2008) also used different membership functions for the linguistic variables than those used by the FLC in the study of Xu et al. (1998). Here, the researchers experimented with different quantities and shapes of membership functions. For example, one set of membership functions used for the obstacle distance (for LD, RD, and FD) consisted of 5 bell-curve-shaped functions: very near, very far, near, far, and medium (Pradhan et al. 2008).

Pradhan et al. tested their fuzzy logic controller in a series of computer simulations (2008). In one of these, an environment with with 1000 robots was simulated and it was found that the robots were able to effectively avoid one another and their obstacles. In

8 Acceleration is not explicitly controlled in Pradhan et al.'s, but it is controlled indirectly, because changing the velocity is the same as changing the acceleration. Acceleration is the rate of change of velocity.

another simulation, 15 robots are simulated in a dead end; the result was that the robots were all able to figure their way out of the dead end and reach their target (Pradhan et al. 2008).

Despite these results, the fuzzy reactive method clearly has one drawback. According to Pratihar et al., the problem with the fuzzy reactive control method is that once a robot is using the method to navigate in the real world, the rule base is not flexible enough to "handle environments which the programmer [of the fuzzy rule base] did not foresee initially" (1999). Therefore, placing the programmed robot in a completely new environment may cause the robot to react to moving obstacles in untactful ways.

Next, we discuss a different approach to fuzzy navigation that is based on force calculation.

2. Agoraphilic algorithm

Since it was obvious that the artificial potential field (APF) method for robot navigation had some major flaws, various attempts have been made to improve it (McFetridge and Ibrahim 2008). The agoraphilic algorithm, developed by McFetridge and Ibrahim, improves the APF method by utilizing fuzzy logic (2008). It uses a free space histogram to calculate the forces inferred from obstacle distances, and then, using a fuzzy controller, it essentially scales each of these forces in such a way as to make directions closer to the direction of the goal more important. Ultimately, the goal is to find the force, and the direction of this force, that should be propelling the robot towards its goal and away from obstacles.[9]

A free space histogram is essentially a structure of information that describes what the distance of the nearest obstacle is in each direction around the robot (McFetridge and

9 Calculating what the force vector that should be exerted on a robot is (i.e. of the propulsion by the robot's wheels) is equivalent to calculating what its speed and direction of travel should be, because, of course, pushing something in a particular direction will make it go faster in that direction. In light of this, all these navigation algorithms do essentially the same thing; they control the robot by making it react to its environment by moving in a particular direction. But that is more specific than the definition of robot navigation. Indeed, several navigation techniques actually consist of making the robot plan ahead and generate a map of its environment (Bekey 2005). I could not find a fuzzy navigation method that does this, so there are no such methods described in this report.

Ibrahim 2008). This is used to guide the robot to the direction with the farthest obstacle distance (2008). For each direction in the FSH, there is a force vector determined by the obstacle distance. The sum of all such forces influences the robot's decision of where it should move. According to McFetridge and Ibrahim, the use of an FSH reduces the robot's navigation to the line of sight so that the attractive pull of the goal does not interfere with the obstacles and cause local minima to occur (2008).

In order to attract the robot to the goal, the attractive forces inferred from the FSH are shaped in such a way as to cause forces that tend toward the goal to be stronger, thus allowing the robot to navigate in that direction (McFetridge and Ibrahim 2008). This is done using a fuzzy controller. For each direction, the controller takes as input the distance to the obstacle and the direction of the goal. The output of this controller is then used to weight the force in this direction in such a way as to cause there to be more force in the direction of the goal (2008). Figure 2 shows the difference in the force vectors acting on the robot when force shaping is used and is not used in the creation of forces. As you can see, without the shaping, the robot does not have a force that pulls it towards the goal.[10]

In the study done by McFetridge and Ibrahim (2008), McFetridge and Ibrahim tested the agoraphilic algorithm on a Pioneer 1 mobile robot using forward facing sonar sensors. For the fuzzy controller, the obstacle distance and goal direction were fuzzified using triangle-shaped membership functions (2008). The fuzzy controller used the fuzzy associative matrix (FAM) in Table 2.[11]

Table 2: Fuzzy rules used for mobile robots in study of McFetridge and Ibrahim (2008).

Angle	**Distance**				
	TooClose	Close	Near	Far	VFar
LLRear	Z	Z	Z	Z	Z
LRear	Z	Z	Z	Z	L
LSide	Z	Z	Z	M	M
LFront	L	L	L	H	VH
Front	Z	M	M	VH	VH

10 And that is because all those forces are the same magnitude, so they will cancel each other out.

11 An FAM is *just* a table that says what the output of a fuzzy controller is based on what its input is. In this case, for example, if the obstacle is to the left of the robot (LSide) and the distance to the obstacle is too close (TooClose), then the scaling of the force in that direction will be zero (Z), which means the robot will not want to go there. A more detailed explanation is in the glossary under *fuzzy associative matrix*.

RFront	L	L	L	H	VH
RSide	Z	Z	Z	M	M
RRear	Z	Z	Z	Z	L
RRRear	Z	Z	Z	Z	Z

McFetridge and Ibrahim tested the mobile robot in two different scenarios. In the first one, the robot was placed in a corridor with two successively narrower openings (McFetridge and Ibrahim 2008). The goal was located through the narrower opening. Three different potential field algorithms were tested in this scenario, and the algorithm that yielded the smoothest path from the start to the goal was the agoraphilic algorithm (2008).

In the second scenario, the robot is placed outside of a narrow hall, at the end of which there is the goal. From start to finish, the agoraphilic algorithm again yielded the smoothest path out of all the APF's used in the test (McFetridge and Ibrahim 2008).

Next, we discuss a method of fuzzy navigation that improves the fuzzy reactive control scheme by learning what the best fuzzy rules are to use in navigation.

3. Data-driven learning algorithm approach

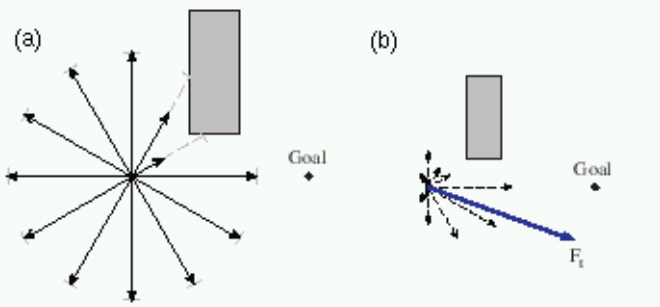


Fig.2: (a) Force vectors without fuzzy shaping. (b) Force vectors with fuzzy shaping. (McFetridge and Ibrahim 2008).

In their paper, Al-Khatib and Saade have presented a data-driven learning algorithm approach for fuzzy navigation (2003). Here, a fuzzy logic controller takes as input information on obstacle position, and using that information, it comes to a decision on where to move. However, the rule base for this controller is actually constructed off-line using a data-driven learning algorithm (Al-Khatib and Saade 2003).[12]

12 Learning algorithms are more ubiquitous than one would guess. For example, a marketing company can use

The overall methodology used by the robot in navigation was described by Al-Khatib and Saade (2003) as follows. Firstly, the robot is to find the position of the nearest obstacle forward (NOF) and to predict the NOF's future position. If the NOF is predicted to be blocking the way of the robot, the robot needs to deviate its direction of movement to a different direction. In light of this, the inputs to the fuzzy controller used to navigate are the distance from the NOF to the robot and the direction of the NOF with respect to the robot (given as an angle[13]). From this, the fuzzy controller determines an angle of deviation that will allow the robot to avoid the NOF.

As stated before, this method for robot navigation depends on a fuzzy rule base that is generated off-line using a data-driven learning algorithm. The algorithm itself is the result of the work of many researchers, and its internal mechanism is beyond the scope of this report. But the algorithm basically works as follows. It starts out with a simple rule base where the consequent of every rule is weak (Al-Khatib and Saade 2003). These rules span over all possible combinations of input variables (i.e. distance big, direction north; distance small, direction west; etc.) (2003). During training, the behaviour error is computed and the consequent of the fuzzy rule is changed in such a way as to minimize this error. During this process, the rule base will eventually converge to a set of rules that yields the minimum possible error. According to Al-Khatib and Saade, underlying this process is a form of hypothesis testing based on random data, which is why the algorithm is data-driven (2003). The fuzzy rule base that was generated by this learning algorithm is presented in Table 3. In real time, the controller is to infer its decisions during navigation using a similar fuzzification-defuzzification process as in the other fuzzy navigation methods described above.

Al-Khatib and Saade tested the rule base that was generated by the learning algorithm in a series of computer simulations where a robot using the rule base navigates in various environments (2003). These simulations covered “various scenarios containing different numbers of obstacles” (Al-Khatib and Saade 2003). According to Al-Khatib and

a learning algorithm to learn what your product preferences are while shopping online. In a robot, a learning algorithm works essentially by comparing the result a robot's action to the desired result. Using a variety of techniques, it tries to minimize the difference between the actual result and the desired result in future events. These algorithms often consist of trial and error.

13 Angles are equivalent to directions. For example, if I am standing to the right of you, the angle between the direction of me relative to you and the direction in which you are facing can easily represent the direction I'm in relative to you. For example, if the angle is 90^{O} left, then I am standing due left.

Saade, in each simulation that was run using the FLC, the robot managed to avoid all obstacles while maintaining an optimal path from start to goal (2003).

Recall that the problem with the fuzzy reactive control method was that it is unadaptable to new environments. As you can see, the problem with the fuzzy reactive control method has been solved in the data-driven learning algorithm method, because a robot can now learn which fuzzy control rules will work in a new environment. (Although the learning algorithm will have to be modified to work in real time instead of off-line.) Next, we discuss a similar method of navigation that uses a different learning algorithm.

4. Genetic-fuzzy approach

In their paper, Pratihar et al. describe a genetic-fuzzy approach to mobile robot navigation (1999). Their approach is similar to the data-driven learning algorithm approach, in which a learning algorithm is used to determine the optimal fuzzy rule base. Here, however, a genetic algorithm (GA) is used to learn what the optimal fuzzy rule base is (Pratihar et al. 1999). According to Karray and De Silva, GA's are algorithms that optimize some variable by using the survival of the fittest principle, where populations of control information are subjected to selection, crossover, and mutation, and therefore evolution occurs that will optimize the variables (2004).

The difference between the data-driven learning approach and the genetic-fuzzy approach lies in how the optimal fuzzy rule base is learned by the learning algorithm, which in this case is a genetic algorithm. In the study of Pratihar et al., the set of all possible rules that can be included in the fuzzy rule base consists of 20 rules drawn from intuition, and is presented as an FAM in Table 4 (Pratihar et al. 1999). From these rules, the genetic algorithm finds the subset of these rules that causes the robot to take the most efficient path to the goal (Pratihar et al. 1999). Pratihar et al. also present an approach where a GA is used to optimize the path taken by the robot by tuning how important the output values are for particular rules (1999).

Again, the input to the fuzzy controller is the distance to the NOF and the direction of the NOF with respect to the robot (given as an angle), and the output is the angle of

Final fuzzy system obtained by learning

Distance	Angle						
	A1	A2	A3	A4	A5	A6	A7
D1	V7	V9	V9	V9	V1	V1	V4
D2	V6	V9	V9	V9	V1	V1	V4
D3	V5	V8	V9	V9	V1	V2	V5
D4	V5	V6	V7	V8	V4	V4	V5
D5	V5	V5	V6	V7	V4	V5	V5
D6	V5	V5	V6	V7	V4	V5	V5
D7	V5	V5	V5	V6	V5	V5	V5
D8	V5	V5	V5	V5	V5	V5	V5

Table 3: fuzzy rule base that was generated by learning algorithm. (Al-Khatib and Saade 2003) (A1 is far left, A7 is far right, D1 is very close, D8 is very far, V1 is far left, V9 is far right).

deviation (Pratihar et al. 1999).

In the study done by Pratihar et al. (1999), various mobile robot scenarios were simulated on a computer in order to test the genetic-fuzzy approach. For example, one scenario consisted of three independent obstacles moving about such that the robot has to pass each of them to reach the goal (1999). Various fuzzy navigation approaches were tested, including using a GA to optimize the fuzzy rule base and not using a GA to optimize the fuzzy rule base, both of which were done in order to find what the difference between the two is (1999). From these simulations, it was found that using a GA improves the efficiency of the path taken by the robot (Pratihar et al. 1999).

Conclusion

In this paper, I have outlined some typical methods that use fuzzy logic for robot navigation. Authors such as Pratihar et al. (1999), Pradhan et al. (2008), and Al-Khatib and Saade (2003) may be consulted for a more brief, yet more complete summary of the multitudes of fuzzy methods used in robot navigation. Generally, fuzzy navigation uses some kind of fuzzy logic controller to infer what the best direction and the speed are for a robot, given the path along which the robot should travel to avoid obstacles and reach its goal.

The fuzzy reactive control method uses a set of reactive rules in a fuzzy logic controller in order to react in real time to obstacles and reach a goal. This method has the

distance \ angle	L	AL	A	AR	R
VN	A	AR	AL	AL	A
N	AL	A	AL	A	AR
F	AL	A	AL	A	AR
VF	A	A	AL	A	A

Table 4: FAM that consists of all possible rules for the fuzzy rule base in the genetic-fuzzy approach. (Pratihar et al 1999)
(Here, VN means "very near," N means "near," etc.; L means "left," AL means "ahead left," etc.; and AR means "ahead right," etc.)

benefit of allowing a robot to navigate in a dynamic environment, although the robot may not react properly to moving obstacles in a foreign environment if it doesn't have a learning algorithm.

The agoraphilic algorithm uses an artificial potential field, but it also uses a fuzzy controller to place greater importance on forces in the direction of the goal. It was demonstrated that the agoraphilic algorithm allows the robot to take a smoother path to its goal than other APF variants.

The data-driven learning algorithm approach and the genetic-fuzzy approach add to the fuzzy reactive approach the ability to optimize the fuzzy rule base in such a way as to maximize the efficiency of the path and minimize the error made by learning from the environment. Since tests indicate that the use of a genetic algorithm to optimize the fuzzy rule base for the FLC makes the path taken by the robot more efficient, we can conclude that the use of a learning algorithm in conjunction with a fuzzy reactive rule base is superior to using reactive control without a learning algorithm.

In this report, I have summarized a few of the many ways robot navigation can be implemented using fuzzy logic. Many of the methods described here can be combined with other methods, and these methods can also be made more complex to factor in more real-world forces. In the last 20 years, there has been extensive research in the field of robot navigation (Pradhan et al. 2008). Modeling this complex process will continue to

challenge researchers.

Navigating towards a goal while avoiding obstacles is a task naturally performed by animals. Using intelligent thinking, machines can smoothly and efficiently reach a target while minimizing their collision with obstacles. It is evident that the various techniques of robot navigation in some way or another model the cognitive process of navigation that humans use. Using fuzzy logic and other methods from soft computing, intelligent behaviour such as navigation can be automated.

References

Al-Khatib and Saade. 2003. An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot. Fuzzy Sets and Systems 134 (2003): p 65-82.

Bekey, George A. 2005. Autonomous Robots: From Biological Inspiration to Implementation and Control. Cambridge, MA: MIT Press. p 473-507.

Gupta and Riordan. Path Planning for Mobile Robots Using Fuzzy Logic [Internet]. Dalhousie University; 2004 [cited 2008 September 22]. Available from: http://www.unbsj.ca/conferences/apics/2004/documents/GuptaRiordanNEW.doc.

Karray and De Silva. 2004. Soft Computing and Intelligent Systems Design: Theory, Tools and Applications. Edinburgh Gate, Essex: Pearson Ed. p 83-85,126-127,137-202,365-401.

McFetridge and Ibrahim. 2008. A new methodology of mobile robot navigation: the agoraphilic algorithm. Robotics and Computer-Integrated Manufacturing (2008), doi:10.1016/j.rcim.2008.01.008.

McNeill and Freidberger. 1993. Fuzzy Logic. New York: Simon & Schuster. p 107,108.

Pradhan, Parhi, and Panda. 2008. Fuzzy logic techniques for navigation of several mobile robots. Applied Soft Computing 9 (2009): p 290–304.

Pratihar, Deb, and Ghosh. 1999. A genetic-fuzzy approach for mobile robot navigation among moving obstacles. lnternational Journal of Approximate Reasoning 20 (1999): p 145-172.

Xu, Tso, and Fung. 1998. Fuzzy reactive control of a mobile robot incorporating a real/virtual target switching strategy. Robotics and Autonomous Systems 23: p 171–186.

Glossary

acceleration – rate of change of velocity. For example, 1 m/s/s refers to increasing your speed by 1 metre per second each second. So if I'm driving at 20m/s and my acceleration is 1m/s/s, then in one second, I will be driving 21m/s. *Deceleration* is the opposite of acceleration, where the speed is decreasing.

algorithm – simply a set of steps that a system takes in order to complete a task. e.g. to wash clothes: put clothes in washer, insert soap, and then push “start.”

array – very similar to a *matrix*. An array is basically a structure of data (i.e. information) that is stored the memory of a computer. The structure of an array is very simple, and can be thought of as a table. In an array, “elements” (i.e. cells) can be accessed given their rows and columns.

centroid – (or *center of gravity*) the centroid of an object is basically where the centre is at which the most “stuff” is located. In the *centroid method* for fuzzy inference, certain membership values have been activated for each fuzzy set of the output variable. Using the geometric object that results from considering the area under the function that is lower than these membership values, you get something that basically looks like a rock. Finding the centroid of this object in effect “defuzzifies” the inference made by the controller, so that the centroid is actually the output value.

conjunction operation – this is the fuzzy AND operation. It takes two fuzzy propositions P and Q (see *fuzzy proposition*), and determines the truth value of the statement P AND Q. For example, the statement true AND false is equal to false, as anyone who's taken a course in logic knows. When we have different grades of truth, we *usually* take the minimum of the two truth values. For example 0.4 AND 0.9 is 0.4.

controller – a controller is essentially a device used to make a decision based on one or more input values. It does this by deducing a value, which can represent anything

depending on what we're trying to do. The value obtained from the controller is called the *output*, and some sort of device is used to actualize this output. For example, I am, in part, a controller trying to control my GPA right now. As input, I take my current GPA and how badly I want to do an honors degree. From that information, I determine using some kind of reasoning how hard I should study. How hard I study is the output.

Controllers may be implemented in a machine in several ways. The two most common are hardware and software. A hardware controller is a controller that reasons using electrical circuits. A software controller is one that reasons using a software program, which runs on electrical circuits, as do all software programs nowadays.

defuzzification – see *fuzzification/defuzzification*.

disjunction operation – basically is the fuzzy OR operation (see *conjunction operation*, which is analogous). The value of P OR Q is the maximum of the two truth values. For example, 0.4 OR 0.9 is 0.9.

force – a force is a the thing that causes something to change how fast it is moving. It can be measured in newtons (N) or pounds (lbs). Force is a vectored quantity, which means that it has a direction (like north-west) and a magnitude (like 5 lbs).

force field – a *function* that maps every point in space to a particular amount of force. For example, a force field might map 5 pounds of force in the north direction to the point at which I'm sitting typing this report. If the robot were here on my lap, and this force field were acting on it, then it would have 5 pounds of force that would push it towards the north.

function – an assignment of values to other values (which could be the same ones). For example, m(x)=5x would assign 4 to 20. We say, “m maps x to 5x.” Functions are often illustrated on axes with the input values on the horizontal axis and the output values on the vertical axis.

fuzzification/defuzzification – fuzzification is the process of taking "crisp" values (such as distance or temperature), and converting them into "fuzzy" values (such as near or very hot). Defuzzification is the process of taking fuzzy values (such as "very fast") and converting them into crisp, exact values (such as "5 meters per second").

fuzzy – this adjective refers to anything that uses or relates in some way to fuzzy logic or fuzzy inference. For example, a fuzzy drying machine is a drying machine that has a fuzzy controller in it.

fuzzy associative matrix – (see *matrix*, *fuzzy rule base*) a matrix that constitutes a fuzzy rule base by associating each row with a fuzzy set and each column with a fuzzy set. Example:

	Ball is near	Ball is far
Ball is high	Swing bat kind of soon	Swing bat after a while
Ball is low	Swing bat really soon	Swing bat kind of soon

For example, one rule here would be IF ball is high AND ball is near THEN bat kind of soon. The fuzzy sets I am referring to are the set of ball-positions that are high (i.e. how much in this set are they?), the set of ball-positions that are near (i.e. how much in this set are they?), etc.

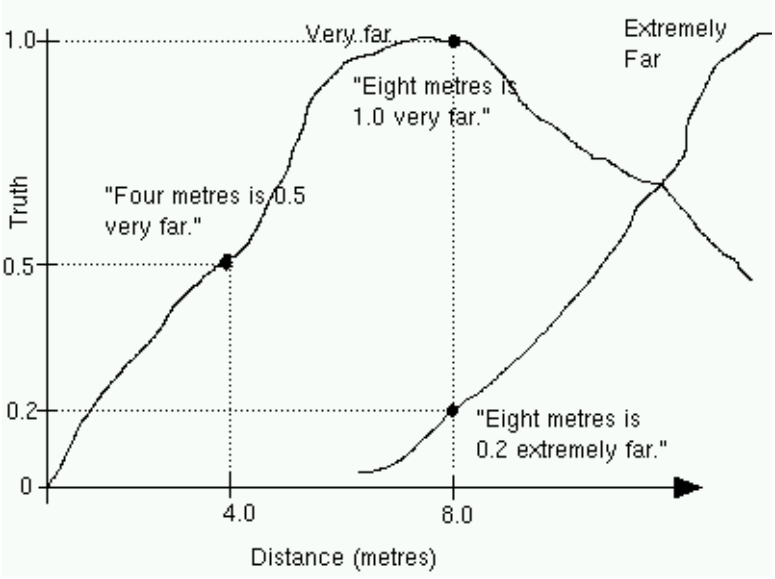
fuzzy inference – the process of taking fuzzy values for certain variables (such as very hot) and drawing a conclusion. For example, IF very hot THEN decrease temperature. Note that these hedges are described numerically by a certain truth value. For example, "temperature is 0.4 in VHOT." Various methods are used in different applications to numerically reach the conclusion of, in the above example, to decrease the temperature by, say, 4 degrees Celsius. It turns out that the most reasonable way to do this is actually to take the minimum of the fuzzy truth value of the consequent and the antecedent.

fuzzy logic controller – a controller that uses fuzzy inference to control a variable by

inferring from some inputs. Typically, it uses a fuzzy rule base to draw conclusions about what the output should be.

fuzzy proposition – propositions in fuzzy logic always have truth values between 0 and 1. 0 is for no truth, 1 is for total truth, and, for example, 0.6 would be “kinda true, but not quite.”

fuzzy membership function – (see *function*) the function that maps a value to how much it is a member of the fuzzy set with which it is associated. e.g. 4 meters is 0.5 in VFAR (or half very far, half not very far). The proposition “4 meters is in VFAR” is 0.5 true.



In the above example, we see that 4 metres is half in the set of very far distances, while 8 metres is completely in the set of very far distances. However, 8 metres is only 0.2 in the set of extremely far distances, meaning it is not very extremely far.

fuzzy rule base – (or *fuzzy knowledge base*) a set of IF-THEN rules which is used by a fuzzy controller to infer its output.

fuzzy set – a set of objects associated with a membership function (see *fuzzy membership function*). For example, some fuzzy set might consist of the set of all possible distances together with m(d)=0.5d when d is from 0 to 2, m(d)=2-0.5d when d is from 2 to 4, and

m(d)=0 otherwise. This is a candidate for the fuzzy set describing how near things are (possibly measured in meters). Note that 1 is half in this set, and would probably be more in the set of very near things. So when we say, for example, that the temperature is “very cold,” we're really saying, the temperature is, maybe, 0.9 in the set of very cold things.

input – a value that is given to a processor (e.g. a controller) that is used in some kind of computation. It's an input because you “put it in.” Note: *input* can be used as a verb. For example, “I inputted the information into a file on my computer using my keyboard.”

learning algorithm – an algorithm that causes a system to learn what its behaviour should be. For example, a human can teach a computerized car how to drive itself by using a learning algorithm (which the car's computer executes) that essentially tries to minimize the car's error at each turn (the human notifies the car's computer of the error).

local minima – see *minimum*.

Mamdani-type inference – a *controller* uses Mamdani-type inference if it uses the following type of *fuzzification-defuzzification* process. First, the controller fuzzifies the inputs to the controller. Then the controller uses a fuzzy operator to find the consequent of the antecedent in each of the rules of the fuzzy rule base. Then all of the consequents are also operated on by some fuzzy operator, and the result is defuzzified, e.g. using the *centroid* method.

matrix – essentially a table with rows and columns of values (or possibly other objects). For example:

1	2
3	4

In this example, the cell in the first row and second column has the value 2.

maximum – 1) The maximum of a set is simply the highest number in that set. e.g.

max(1,5)=5.

minimum – (pl. minima) 1) A minimum of a function is a point at which the function has the lowest value, at least locally. A local minimum is a point where the function is locally lowest, but it could be lower elsewhere. This is the problem with the APF algorithm, because a local minimum in the potential field would cause the robot to stay put there (because it is trying to minimize the amount of potential, and a local minimum is a point where there is a least amount of potential). However, what it really should to do is go to where the goal is. A *global minimum* is the point at which the function is lowest throughout its domain. 2) A minimum of a set of numbers is simply the smallest number in the set. For example, min(1,3,-4)=-4.

mobile robot – see *robot*.

operator – an operator is just an entity that does some operation on one or more objects. For example, + is an operator; it operates by adding the two numbers that are on its sides.

output – a value given *by* a processor (e.g. a controller) that is the result of some computation. For example, when we say that steering angle is one of the outputs of a fuzzy controller, we are saying that the fuzzy controller does some reasoning, and then it gives us this steering angle that is appropriate for the situation. This steering angle is given to the steering system of the robot as information, which the electronic devices on the robot use to physically perform the task of steering in that angle.

It is called the output because the processor or whatever "puts it out." For example, all my output is really what I say.

Note: *output* can be used as a verb. E.g. "the mailman outputted his response to the babysitter."

potential field – a *function* that maps every point in space to a quantity called "potential." Basically, in the universe, things like electrons do not like having lots of potential, and so

they do what they can to decrease it. (It's basically because they have a force exerted on them that makes them do so, like gravity or electromagnetism. That's how force and potential are related.) In the APF method of mobile robot navigation, a robot is given higher potential around areas where there are lots of obstacles, and it is given lower potential around the goal. The potential is what the robot tries to minimize. It's best thought of as this value that nobody wants, like bad credit.

proportional-integral-derivative (PID) *controller* – a controller that basically takes into consideration the current value of the thing it is trying to control (e.g. temperature), the past values of the thing it is trying to control, and how quickly this value is changing. It may or may not use fuzzy logic. Some variants (i.e. PD or PI controllers) only consider either the past values or how fast the value is changing.

robot – a machine that functions in an environment using reason and that can interact physically with its environment. A *mobile robot* can move around in its environment. An *autonomous robot* can function in its environment without very much external input from a human being. Robots are given *sensors* to detect its environment, and they are given *actuators* that they can use to manipulate their environments.

simulation – a simulation is basically when a computer determines what would happen in real life in a certain situation. For example, I could simulate tossing a coin using a computer by writing a program that picks “heads” or “tails” randomly.

variable – a quantity that changes. For example, temperature changes. My GPA changes.

vector – a quantity associated with a direction: e.g. 4 pounds of force in the left direction. A *force vector* is a vector representing force. In this paper, all forces consist of their magnitudes and directions. A vector is typically illustrated as an arrow pointing in the direction of the vector. The length of the arrow is set to the scale with respect to the magnitude of the vector.