An Introduction to Fuzzy Logic Applications for Robot Motion Planning

Mr. Paul Yanik[1], Dr. George Ford[2], Dr.Brian Howell[3]

Abstract - This paper considers a fuzzy logic application for navigation and obstacle avoidance of a robotic vehicle in a 2D environment. An algorithm for 2D navigation was implemented in a simulation, and its effectiveness in various obstacle fields is discussed. Fuzzy algorithms have proven effective in achieving performance objectives for attainment of goal configurations and real time autonomous operation as exemplified in an environment which was simulated using National Instruments LabVIEW version 8.2 software.

Keywords: fuzzy logic applications, robot motion planning, trajectory planning, fuzzy logic

INTRODUCTION

In the field of mobile robotics, the goal of autonomous robot operation is a topic of considerable research. Robots may frequently find themselves in environments for which a reliable map of obstacles and terrain is unavailable due to the dynamics of the environment, imprecise sensory data or simply a lack of prior knowledge of the environment [4]. In such situations, it may be necessary for a robot to recalculate its trajectory online. While a globally computed motion plan for the robot in terms of gross motions over longer distances may remain valid, a response to a moving or unforeseen obstacle may force the robot to alter its path amid local obstacles while en route from the initial configuration to the final goal. As a possible means to meet such requirements, fuzzy logic based algorithms have found application in the area of robot motion planning because of their inherent ability to deal with imprecise inputs and their low computational complexity.

Fuzzy algorithms execute in three major stages: fuzzification, inference, and defuzzification. In the fuzzification stage, real world sensory inputs in a given universe of discourse are characterized on the closed interval [0, 1] according to their levels of membership in fuzzy sets. These sets are given names which express qualities of the input variable using easily understood linguistic terms. A membership function maps the value of the input variable to a degree of membership in each of the fuzzy sets. The fuzzified value then, represents the level of truth of each of these linguistic terms for a given input. For example, the angular direction of a near obstacle to a mobile robot might have a universe of discourse of -90° to 90° where 0° denotes the current heading of the robot. Figure 1 below shows a possible definition of fuzzy sets for how a "crisp" (real world) input angle θ might linguistically reflect the level to which the obstacle is left, in front, or right of the vehicle. Here, an angle of 30° represents a direction having fuzzified degrees membership in the set of angles to the right ("rightness") of μ_R and frontness of μ_F. In this manner, fuzzy sets are capable of handling imprecise inputs. Should an input angle be sensed inaccurately, its levels of truth according to the linguistic terms may vary while its relative membership levels in the sets remain qualitatively the same. Sets are often defined to have the piecewise-linear shape shown so as to reduce the computational complexity of determining set membership [3]. Other shapes include triangular, square, singleton, Gaussian or asymmetric types. Other variables commonly of interest in robotic applications include distance to an obstacle and speed of the robot with respect to an obstacle.

[1] Western Carolina University, Belk 161, Cullowhee, NC 28723. pyanik@wcu.edu
[2] Western Carolina University, Belk 161, Cullowhee, NC 28723. gford@wcu.edu
[3] Western Carolina University, Belk 161, Cullowhee, NC 28723. bhowell@wcu.edu

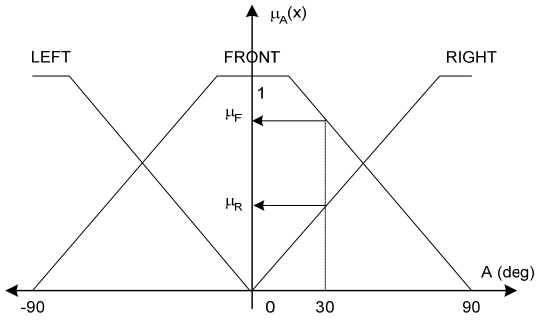

Figure 1 - Example fuzzy set definition with a fuzzified input

The inference stage applies the fuzzified input value to a rule base to determine a [still fuzzy] command output. The rule base contains the operational intelligence of the system. A generalized format for a rule is:

If <predicate conditions> Then <consequence> (1)

Where *predicate conditions* are combinations of each input variable and their relative levels of linguistic truth, and *consequence* is a fuzzified output command. An example rule which might apply to a mobile robot is given below.

If ObstacleAngle is RIGHT and Distance is NEAR Then
SteeringDirection is LEFT_BIG (2)

A rule base must cover all permutations of input variables having degrees of truth in all possible linguistic terms. Hence the total number of rules N which must be represented in a rule base either by explicit statement or default action is given by in the relation below.

$$N = \prod_{i=1}^{m} p_i \tag{3}$$

where m is the number of input variables (angle, speed, distance, etc.), and p_i is the number of linguistic terms for the i^{th} variable [5]. Multiple rules in the rule base may have their predicate conditions satisfied to greater or lesser degrees by fuzzified input variables. Such rules are said to have *fired*. In fact, where fuzzy sets are defined to overlap on their universe of discourse, at least two rules are guaranteed to fire for any input value. Each fired rule, then, possesses an *adaptability* to the associated output command through the fuzzy operation (AND, OR, sum, bounded sum, product, etc.) stated by the rule [2]. Commonly, the AND (min) operation is used as shown in the example above. In such a case, the minimum fuzzified value of the predicate conditions becomes the adaptability of that rule to its consequence.

The defuzzification stage extracts a crisp command output from inferences drawn from fired rules. Techniques for defuzzification generally involve some analysis of the regions created by cutting the output fuzzy sets using adaptabilities from each of fired element in the rule base. An example of such a region is given by Figure 2. Common methods for this operation include taking the centroid of largest area (CLA), and mean of maximum value (MOM). Numerous other approaches exist which are not considered here.

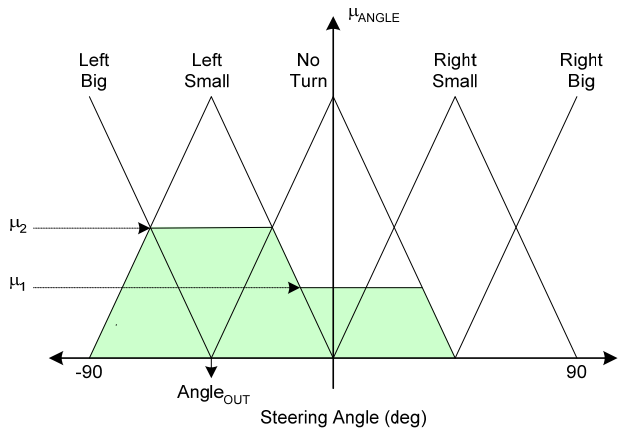

Figure 2 – Example of defuzzification (by MOM method).

This paper discusses an application of fuzzy logic based algorithms to robot navigation control in a 2D environment. The first parts of the paper discusses theory related to 2D robot motion control. The second part describes the simulation of a 2D navigation algorithm which was implemented for the purpose of examining the practical subtleties of fuzzy controller design.

MOBILE ROBOT NAVIGATION IN A 2D ENVIRONMENT

Yang, Maollem and Patel [3] proposed an augmentation to previous applications of fuzzy logic to 2D robot motion planning. All figures, equations and algorithm details in this section are attributed to this work unless otherwise stated. They cited previous research in which the use of fuzzy methods was chiefly focused on short range reactive control. That is, robots were navigated by simply reacting to near obstacles as they were detected while taking into account a global goal direction. While such algorithms have frequently proven effective, they often encounter situations in which the goal configuration becomes unreachable by the robot despite the availability of a traversable path. More commonly, reactive fuzzy navigation may suffer from "shortsighted" behavior wherein the angle to the final goal influences all steering decisions in unison with local sensor data. Situations then arise in which short range sensors may not detect obstacles between the current configuration of the robot and the goal. In these cases, a path may be selected that is less desirable than others available. Figure 3 compares the results of shortsighted behavior to an end-to-end path plan. Through purely reactive fuzzy control using short-range sensors, the robot attempts to move in the direction of the goal at point B when obstacle 2 is still out of its perceptive range. The undesirable path ABCDEG is the result. Clearly, through the benefit of long-range planning, path AJKG would be seen as more desirable.

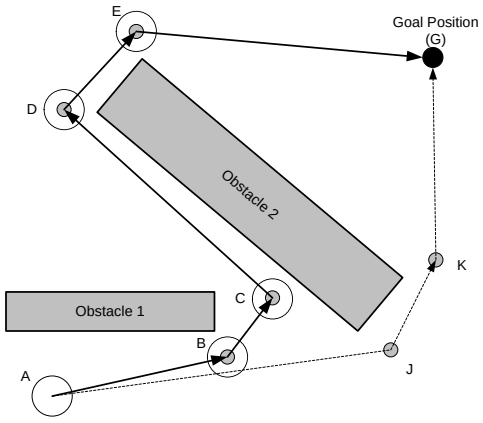

Figure 3 – Shortsighted goal-seeking versus long-range path planning.

Yang, et al [3] formulated an approach which attempts to overcome the past tendency of fuzzy algorithms to such behaviors using a "layered, goal-oriented" navigation strategy. Two layers were proposed. These can be qualitatively separated into long-range versus short-range information assessment. The first layer uses long-range sensor data and the global goal angle to determine a direction that is both traversable (free of obstacles) and desirable (in the direction of the goal). The qualities of directional traversability and desirability are represented as fuzzy sets and are fused to produce a way-point along the path to the goal. A high-level block diagram of the first layer of the planner is given by Figure 4.

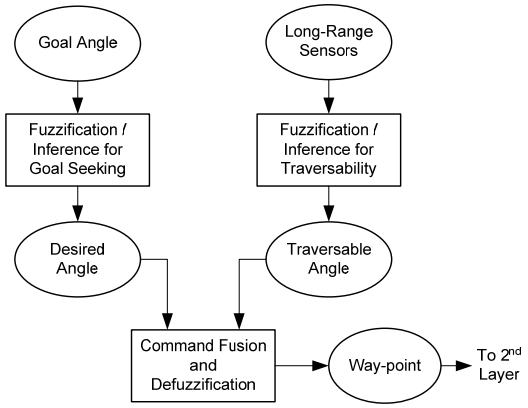

Figure 4 – Block diagram for long-range way-point calculation (layer 1).

Sensors are positioned at intervals around the perimeter the robot body and are activated upon detection of far away obstacles. The signal strength of each sensor indicates the relative nearness of an obstacle. This strength is fuzzified through a collection of trapezoidal fuzzy sets for angles of -180° (left) to +180° (right). Where adjacent sensors detect an obstacle with strengths of μ_1 and μ_2, an untraversable area τ_i is the composed fuzzy set found by the bounded sum of the two strengths as given by the equation below.

$$\tau_i = \mu_1 \oplus \mu_2 = \min\{1, \mu_1 + \mu_2\} \tag{4}$$

An example of a composed fuzzy set for untraversability is given by Figure 5.

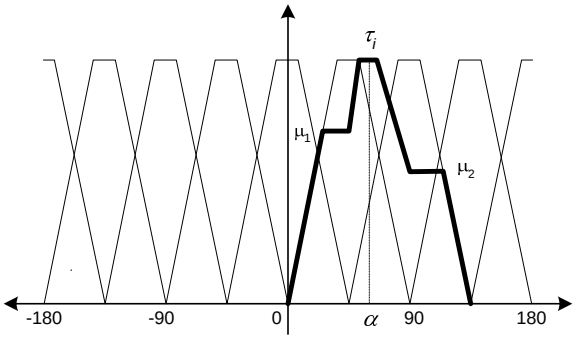

Figure 5 – Composed fuzzy set for untraversability in layer 1.

The *traversable* area Γ is given by the complements of all τ_i as defined by the equation below.

$$\Gamma = not \bigvee_{i=1}^{n} \{\tau_i\} = 1 - \max_{i=1}^{n} \{\tau_i\} \tag{5}$$

where n is the number of activated sensors [3]. The desirability □ of a potential steering direction is a fuzzified representation of the goal angle ϕ with respect to the current heading of the robot. It is determined by a composed set from the relative strengths (□$_1$ and □$_2$) of adjacent triangular fuzzy sets at 90° intervals on the same universe of discourse as traversability. It is defined by the equation below.

$$\Omega = \mu_1 \vee \mu_2 = sum\{\mu_1, \mu_2\} \tag{6}$$

A composed fuzzy set for desirability is depicted in Figure 6.

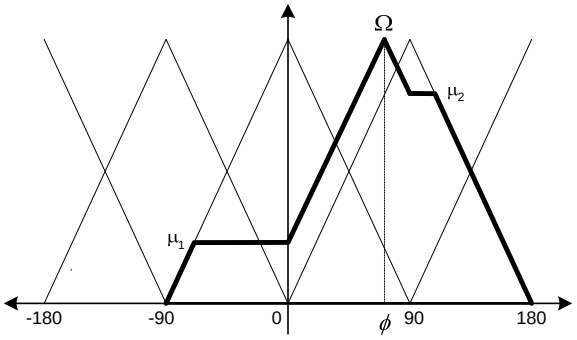

Figure 6 – Composed fuzzy set for desirability (goal-seeking) in layer 1.

The direction to the way point is the fuzzy set $\tilde{\gamma}$ represented by the intersection of composed sets for traversability and desirability.

$$\tilde{\gamma} = \Gamma \wedge \Omega = \min\{\Gamma, \Omega\} \tag{7}$$

Over the range of possible angles, the intersection set will have multiple peaks. Each peak will be separated from others by an interval of zero membership since the sets for untraversability have crossing point values greater than 0.5. This intersection of composed sets is defuzzified by taking the mean of maximum of the largest area (MOMLA) to generate a crisp angle γ which is the direction to the way-point. This defuzzification technique is a combination of the mean of maximum (MOM) and centroid of largest area (CLA) techniques. It was shown to offer an improved outcome when finding the crisp target angle over the CLA method that is often used elsewhere. The position of the way-point and its orientation angle can then be found by

$$x_w = x_i + \rho\cos(\theta_i - \gamma) \tag{8}$$

$$y_w = y_i + \rho\sin(\theta_i - \gamma) \tag{9}$$

$$\theta_w = \theta_i - \gamma \tag{10}$$

where (x_i, y_i, θ_i) is the initial configuration of the robot, (x_w, y_w, θ_w) is the configuration of the way-point, and ρ is the distance from the robot's current position to the way-point and

$$\rho = L + \delta \qquad \rho \le R \tag{11}$$

where L is the distance between the robot and the obstacle closest to the way-point, □ is the size of the robot and R is the effective radius of the sensors.

The second layer takes the way-point produced by the first layer as a subgoal. This layer produces a local trajectory which guides the robot toward the way-point while avoiding collisions with obstacles. Here, a ring of

short-range sensors is used to implement a reactive navigation algorithm wherein local direction and speed commands are generated in response to near obstacles as the robot seeks the way-point. A high-level block diagram of the second layer of the planner is given by Figure 7.

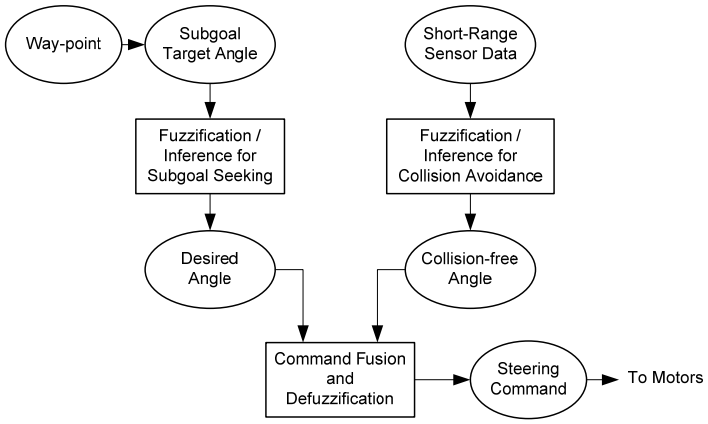

Figure 7 – Block diagram for short-range steering command generation (layer 2).

In a manner analogous to the first layer algorithm, the way-point is used as a desired direction for the robot while sensor inputs are used to infer a direction that allows the robot to avoid obstacles. The simultaneous objectives of subgoal seeking and obstacle avoidance are combined to produce a safe heading for the robot. The authors note that in seeking the subgoal, it is acceptable for the robot to nearly touch obstacles. Hence, only a single threshold is needed for determining closeness to them. This gives rise to a rule base for the second layer of the general form shown below.

If sensor S is fired, Then
the direction indicated by S becomes fully disallowed. (12)

Here, a fired sensor denotes one in which the sensor reading for a near obstacle is above some threshold (which may be set to a high level). By fully disallowing the direction associated with a fired sensor, the subgoal is pursued as aggressively as possible. The subgoal need not be attained precisely, however, since it is not the global objective. Further, the subgoal may become unreachable by dynamics of the environment or the robot may become lost or stuck while seeking the subgoal. In such situations, the second layer may need to abandon the subgoal and pursue a new one. This judgment can be made by assigning a time limit T for the robot to reach the way-point. The time limit can be found as

$$T = \frac{\rho}{v_b} \tag{13}$$

where v_b is the typical speed of the robot in an unobstructed environment. The second layer of the algorithm can accept a new way-point from the first layer whenever the time limit expires.

A final feature of the Yang's, et al [3] algorithm was the implementation of a deadlock handling mechanism. Should the robot enter a situation in which obstacles block a direct path to the final goal and such obstacles are too large for long-range sensors to plan a path around, the fuzzy algorithm presented thus far could result in oscillatory motion that does not result in progress toward the goal. In such situations, the robot needs a strategy breaking the cycle of unproductive decisions. Typically, this is handled with a wall (or contour) following behavior until a safe trajectory is discovered. Instead, the authors temporarily replace the global goal angle with a new target angle. This allows the robot to make reasonable path alterations based solely on traversability when encountering convex

obstacles (even though it may move in a direction away from the global goal) in order to extricate itself from a deadlock scenario.

The authors implemented their algorithm experimentally on a Koala (6-wheeled) robot equipped a ring of sixteen infrared sensors. The sensors had a range of 5-20 cm and a field of perception of 10°. The robot was also equipped with wheel encoders. The infrared sensors were used as both long- and short-range detectors by setting their thresholds accordingly.

Experimental results were divided into those for static versus dynamic environments. In a static environment, the algorithm determined only reachable way-points. The arrangement of the way-points resembles planning by a visibility graph approach. Graphs of sensor readings show that the readings were generally low except when a way-point was close to the vertex of an obstacle. This behavior is interpreted as the algorithm's ability to avoid obstacles before coming close to them. Further, this behavior represents avoidance of the shortsighted behavior problem associated with a purely reactive fuzzy controller.

The algorithm's behavior in a dynamic environment involved movement of obstacles during the robot's passage so as to render some way-points unreachable. In these situations, the robot was observed to abandon such way-points upon expiration of its deadline and to pursue new way-points until the final goal was attained.

Finally, in order to demonstrate the effectiveness of the layered algorithm over earlier methods, the Koala robot was also programmed (separately) with only reactive direction-based and speed-based fuzzy algorithms. Compared to these algorithms, the layered approach produced improvements in navigation time of 27.6% and 16.3% respectively. Further, it is seen through the graphed data presented in the article that the layered approach produced a smoother trajectory and generally avoided obstacles before coming in close proximity to them. This behavior allowed wheel velocities to remain roughly constant with respect to one another which resulted in less directional oscillation.

METHODOLOGY

As a platform to facilitate understanding of the subtleties of fuzzy algorithm implementation, a simulated environment which emulates the 2D layered approach of Yang, et al. [3] was created. The environment was implemented using National Instruments LabVIEW 8.2. The LabVIEW PID Controller Toolkit was employed to implement the core fuzzy controller. The environment is a Cartesian plane which displays the square region bounded by opposite vertices (0, 0) and (100, 100). Up to three triangular obstacles can be added dynamically. A robot modeled as a point navigates autonomously under fuzzy control from a programmable starting point to a goal point during run time. Due to time constraints of the project and limitations associated with the fuzzy logic controller, the calculation of way-points in the first layer of the algorithm discussed in [3] was improvised to show probable results. The second layer of the algorithm which implements short-range obstacle avoidance and subgoal seeking was fully implemented. Due to the graphical nature of the LabVIEW source code, images were deemed to expansive to be discernable in this document and have been omitted.

Fuzzy sets were defined in typical trapezoidal and triangular shapes. Set definitions are shown in **Error! Reference source not found.**. Input variables were Obstacle Angle and Goal Angle. The output Navigation Angle represents a correction to the current heading angle. It was observed that finer granularity in the fuzzy set definition (i.e. more linguistic terms) offered less dramatic command changes to the robot. Further, trapezoidal sets appear to facilitate smoother directional changes since a greater span of the universe of discourse is devoted to full set membership for each term. Using triangular sets over trapezoidal sets for a given input variable effected the behavior of the robot adversely even to the point of collision with obstacles en route to achieving the goal. A further subtlety can be seen where both obstacles and goal are directly in front of the robot. In such situations, a bias must be placed in the rule base toward navigation either to the right or left. Hence, navigation will appear to have a preference in the direction chosen by the designer. The rule base developed for this simulation environment is given in **Error! Reference source not found.**.

For initial testing, the Goal Angle represented a global goal. In such cases, the robot exhibits the shortsighted behavior discussed in Section 0. An example simulation depicting this behavior is given by Figure 8. The robot clears the lower obstacle and navigates directly toward the goal since the larger obstacle is still out of sensory range. Then, upon detection of the larger obstacle, the local trajectory prescribed by the rule base places the robot on a path that takes it around the upper obstacle and further from the goal before it can once again move in the desired direction.

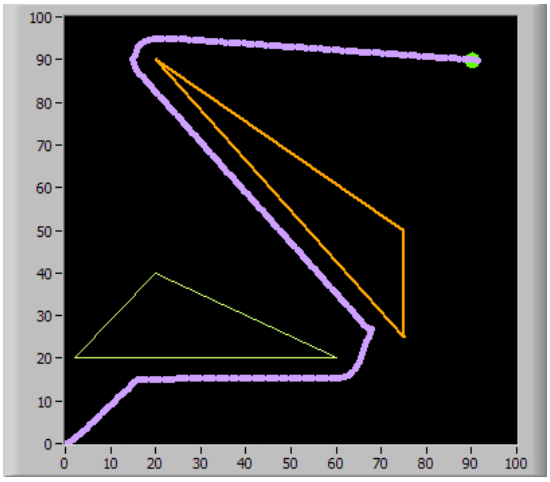

Figure 8 – Simulation results showing shortsighted navigation behavior.

With the addition of way-points, each way-point can be seen acting as a subgoal directing the robot toward the global goal. Each way-point is achieved in turn. The overall effect is similar to the vision graph approach in roadmap motion planning. The long-range sensors on the robot tend to seek the nearest vertices of obstacles since they represent the most traversable and desirable path where the vertex is approximately in line with the global goal. Figure 9 below shows the same obstacle field as in the previous figure with the path governed by generated way-points.

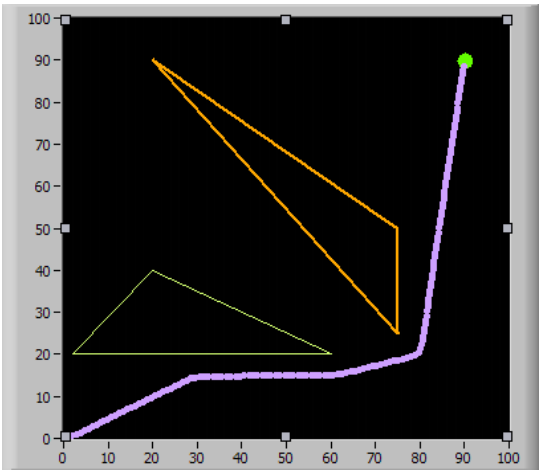

Figure 9 – Simulation results with way-points added for long-range navigation.

One of the stated benefits of fuzzy logic based control is real time trajectory alteration due to the low computational complexity of the underlying algorithm. Figure 10 below shows the ability of the algorithm to react to unforeseen obstacles. The small obstacle depicted in the figure is inserted dynamically during run time so as to provide an obstruction directly in line with the calculated way-point. The figure shows that the algorithm safely navigated around the obstacle and went on to the global goal.

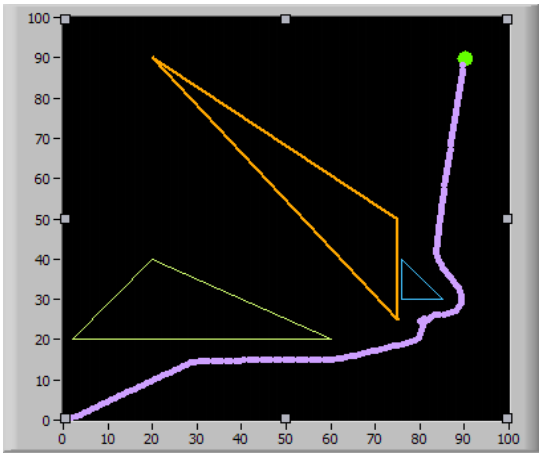

Figure 10 – Robot behavior with way-point obstructed by an unforeseen obstacle.

A final simulation scenario was conducted which shows the behavior of the robot in a deadlock situation. When the layer 1 goals of traversability and desirability denote opposite behaviors, an oscillatory motion can result. This is a common weakness for fuzzy logic based algorithms. As previously discussed, Yang, et al. [3] propose a deadline mechanism which forces the robot to choose a new subgoal upon expiration of a specified time period. Figure 11 shows a dynamically inserted obstacle (the small triangle) which renders a way-point unreachable. With the addition of this obstacle, the direction to the subgoal lies in direct opposition to the traversable directions available to the robot. A period of oscillatory motion is induced. However, after expiration of the deadline (70 system clock periods in this case), the robot abandons the subgoal and pursues a new subgoal based solely on traversability.

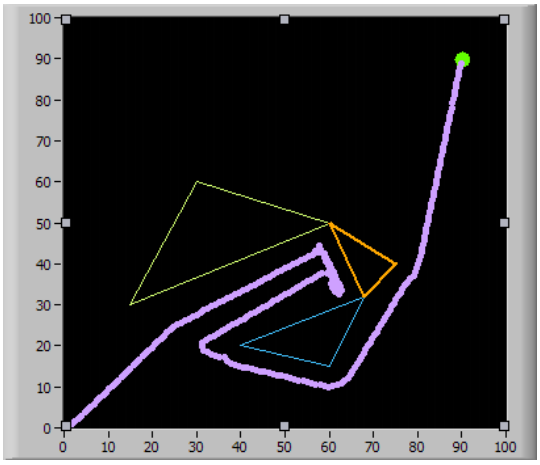

Figure 11 – Oscillatory (deadlock) behavior followed by expiration of deadline and redirection.

Development of this simplified fuzzy control environment represented a significant learning experience. Many hurdles were encountered during the design and programming phases. Even with the use of an established controller toolkit, it quickly became clear that creating an effective fuzzy set and rule base combination having logical consistency across all behaviors is challenging. The difficulty of this task would be amplified when one considers the increased complexity of adding more variables, each with its own terms and fuzzy set definitions.

CONCLUSION

The use of fuzzy logic for local navigation of robots is a much discussed topic in literature. Various implementations have been shown to be effective and efficient. The ability of fuzzy techniques to deal with

imprecise data allows for smooth trajectory execution while their low computational complexity allows them to react quickly to dynamic environments without the need to alter the robot's end-to-end path.

Because algorithms based on fuzzy logic depend on sensory data to make navigational corrections, they are essentially reactive in nature. This quality can elicit suboptimal behaviors including shortsightedness and the local minima problem. A knowledge of longer range obstacles as from long-range sensory instruments or conventional model-based planning algorithms could allow the reactive behavior of a fuzzy controller to make quick trajectory adjustments while still approximating the most preferred path.

The implementation of a fuzzy controller as discussed here required an iterative approach. The shapes of fuzzy sets and consequences of rules can have a dramatic effect on the overall effectiveness of the controller. Further, when the number of fuzzy variables and/or their linguistic terms increases, the number of rules which must be covered by the controller may become large and unwieldy. Logical inconsistencies across the range of robot behaviors may be accidentally implemented by the designer. Hence, the combination of rule construction and rule count presents a challenge to the formulation of uniform and desired outcomes. Extensive tweaking of both fuzzy sets and rules may be necessary. Future work would include the study of methods to automate the layout of fuzzy sets and the rule base. There is also a substantial body of literature which discusses the use of genetic algorithms to tailor the shapes of fuzzy sets. Neural networks have also been used to provide learned weights to fuzzified data. Both of these approaches make use of training data to produce a more effective fuzzy control platform.

REFERENCES

[1] D. Shi, E.G. Collins, and D. Dunlap, "Robot Navigation in Cluttered 3D Environments Using Preference-Based Fuzzy Behaviors," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 37, no. 6, pp. 1486-1499, Dec. 2007.

[2] K. Tanaka, *An Introduction to Fuzzy Logic for Practical Applications*, New York, NY, Springer-Verlag, 1997.

[3] X. Yang, M. Maollem and R.V. Patel, "A Layered Goal-Oriented Planning Strategy for Mobile Robot Navigation," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 35, no. 6, pp. 1214-1224, Dec. 2005.

[4] P.G. Zavlangas and S.G. Tzafestas, "Industrial Robot Navigation and Obstacle Avoidance Employing Fuzzy Logic," *Journal of Intelligent and Robotic Systems*, vol. 27, pp. 85-97, 2000.

[5] *LabVIEW PID Controller Toolkit User Manual*, National Instruments Corporation, Austin, TX, 2006.