Robotica (2005) volume 23, pp. 681–688. © 2005 Cambridge University Press
doi:10.1017/S0263574705001682 Printed in the United Kingdom

Implementation of a hexapod mobile robot with a fuzzy controller

Makoto Kern and Peng-Yung Woo

Electrical Engineering Department, Northern Illinois University, Dekalb, IL 60115 (USA)

(Received in Final Form: January 22, 2005)

SUMMARY
Fuzzy logic has features that are particular attractive in light of the problems posed by autonomous robot navigation. Fuzzy logic allows us to model different types of uncertainty and imprecision. In this paper, the implementation of a hexapod mobile robot with a fuzzy controller navigating in unknown environments is presented. The robot, MKIII, interprets input sensor data through the comparison of values in its fuzzy rule base and moves accordingly to avoid obstacles. Results of trial run experiments are presented.

KEYWORDS: Mobile robot; Hexapod; Fuzzy controller; Sensor data.

I. INTRODUCTION

Intelligent and autonomous control is a new and rapidly growing field of research.[1,2] There has been rapid growth in the field of artificial intelligence for robotic control with techniques involving neural networks, fuzzy logic and genetic algorithms. These techniques offer a variety of ways in modeling nonlinear systems in a broader sense. Autonomous robot navigation in an unstructured environment is a challenging task, which requires dealing with a large amount of uncertainty. The technique presented in this paper is a fuzzy algorithm, with which a hexapod mobile robot can guide itself to move around avoiding obstacles successfully.

Fuzzy logic has several features that make it an adequate tool to address this task. These include the ability to represent different types of imperfect knowledge; express nonlinear control laws in the form of heuristic if-then rules; and integrate numeric and symbolic aspects of reasoning.

The mobile robots that have been developed can be basically broken down into two categories, wheeled or legged. These mobile robots are not only a collection of algorithms for sensing, reasoning and moving, but also the physical embodiment of the rules and ideas that must survive with all the dynamics of the real world. As such, the main objective of this paper is to implement a fuzzy controller for a low-cost mobile robot. Due to the advantages of being able to transverse over rugged terrain, having the ability to continue locomotion even if one leg became damaged and the ability to choose the contact points of the leg to the land, research on developing an autonomous legged robot becomes our goal.

Autonomous legged robotic systems are a relatively new research area[3] that has gained much ground in the past few years with the advent of artificial intelligence theories. For mobile systems to behave in large-scale environments, we must deal with the incremental acquisition of knowledge, the estimation of error, the ability to recognize important or familiar objects/places and exhibit real-time response, etc. Furthermore, it requires all these items be exhibited in concert. The tasks of sensing, reasoning and moving are fundamental problems in the study of mobile robots.

In the light of animal locomotion being superior to any artificial walking machine in existence, research on mimicking nature's most successful mobile creatures is being done. R.J. Full oversees a lab in UC Berkley that is regarded as a "gymnasium" for insects where their movements are analyzed and studied so that nature's secrets of locomotion can be extracted and applied to a wide range of robotic problems. Rhex, which is developed at the University of Michigan, is a highly mobile hexapod that uses legs similar to the self-stabilizing sprawled posture found in a cockroach. These legs are rotated in a windmill-like fashion to achieve speeds of up to 9 feet per second.[4] R. Brooks designed two other famous 6-legged robots in the early 1990's.[5] These robots Hannibal and Attila were constructed as experimental platforms for autonomous planetary exploration. Important research on rugged terrain locomotion, and fault-tolerant and real-time behaviors was done with these legged robots.[6]

Autonomous mobile robotic systems have often been constrained to specially designed environments.[7, 8] In real world conditions, the outdoor terrain of obstacles and other hazards are difficult to model, yet nature has developed animals with very little mathematical intelligence but strong ability to survive. How are simple creatures able to effectively survive in the real world? What methods can be used to "train" current mobile robots to logically move about and survive in their environment? They are always the questions of the researchers.

Section II details the hardware and software used. Section III describes the fuzzy system used to control the hexapod mobile robot. Sections IV and V shows experimentation results of the robotic system when implemented in real world situations. Section VI concludes the paper.

II. THE ARCHITECTURE OF THE HEXAPOD MOBILE ROBOT

The design of a mobile robot with insect-like locomotion is an ingenious integration of nature's intelligence with a man-made machine. Extensive research on four and six-legged vehicles has been done to determine the most efficient way that an autonomous robot could move. Robust, flexible

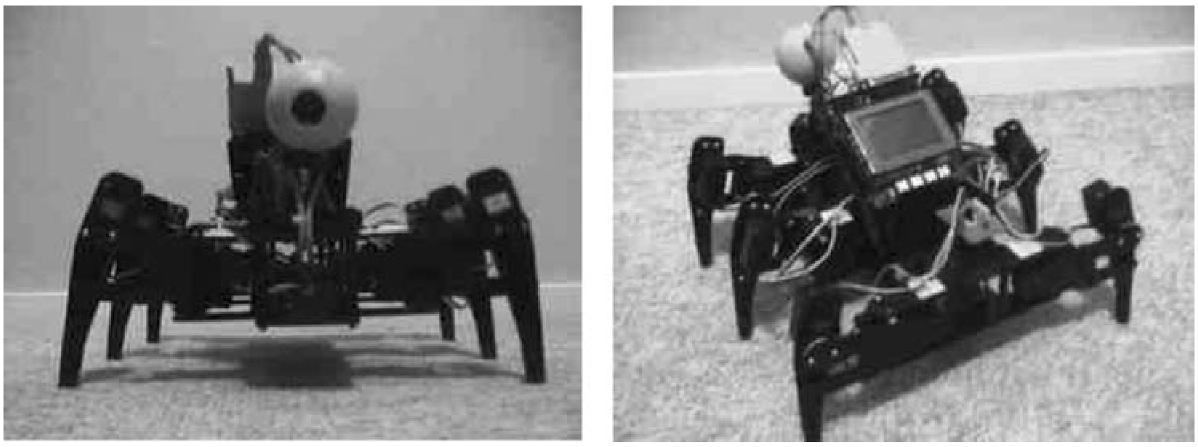
Fig. 1. MKIII hexapod mobile robot.

locomotion is implemented using ideas from insect gaits and strategies used to traverse natural terrain. The successful design of a legged robot depends on the leg design chosen. Since all aspects of walking are ultimately governed by the physical limitations of the legs, it is important to select a leg that allows for a maximum range of motion and a good performance. Figure 1 demonstrates the completed MKIII hexapod mobile robot with the camera, controller and sensors.

The controller that consists of a 35MHz, 32-Bit micro-controller board with a graphics display and a digital color camera allowing it to perform on-board image processing acts as the "brain" of the robot. The controller has 1MB of RAM and 512KB of ROM for system and user programs. This controller allows the robot to be fully autonomous and self-sufficient rather than being tethered to a desktop computer system.

**Camera*
A Logitech QuickCam Color V2 camera for Windows can be connected to the parallel port of the controller to be used for basic image processing. The resolution of the images when displayed onto the screen, however, makes it difficult to distinguish obstacles from noise. Therefore the camera is not used.

**LCD display*
The LCD Display connected to the Eyebot can display text (8×16 characters) and graphics (64×128 pixels).

**Power supply*
There are two pins for the power supply that are marked as "+" and "−". A supply voltage between 7V and 9V can be used to power the controller. If an overload occurs, a surface-mount fuse located right above the power switch will open, thus preventing the expensive circuitry from being damaged. This fuse, which is also called a Fast Recovery Diode 50V 3A, has blown several times due to the overloading of the motors during travel.

**Microphone*
A miniature microphone is built in the controller and can be used to detect sounds.

**Serial connector*
A standard 9-pin RS-232 extension cable can be used to connect the controller to a PC. Software can be downloaded or uploaded between the controller and the PC.

**DC motor and encoder connection*
There are 2 connections for DC motors and encoders. Distance traveled data can be achieved with the encoder connection. Two motor drivers are integrated into the controller, which are pin compatible for motors from Faulhaber, MiniMotor, and MicoMo.

**Servo connections*
There are 12 connectors in which servos can be directly plugged into. Two standard servo manufacturers are Futaba and Hitec.

**Infrared connections*
There are 6 connectors for infrared sensors. All control logic is included in the controller's embedded logic. The connector are pin compatible for Sharp GP2* family of infrared sensors.

**Speaker connections*
There are two connectors on the front side of the controller for either a piezo speaker or an external standard 8-Ohm speaker. Using an external speaker improves the sound quality significantly. Speaker volume can be adjusted by a potentiometer located next to the speaker connector.

**Extension connections*
There are three separate connectors for adding additional I/O, i.e., Digital input connector (2 inputs, 4 pins), Digital output connector (3 inputs, 3 pins) and Analog input connector (6 inputs, 10 pins). Additional location sensors such as a digital compass can be connected to robot to enhance the direction sensing capabilities of the robot.

**Wireless connection*
Wireless communication between other robots or a PC is possible with the controller if a wireless add-on is purchased. Multi-agent robot control is implemented through the University of Western Australia Eyebot Soccer team with this feature.

**Background debugger*
The Motorola 68332 32-bit processor is the main computing power behind the controller. This processor allows debugging from a PC under DOS or Linux via a background debugger program. A 10-pin connector is used to link the "BDM" to the parallel port of a PC. The BDM allows changes to the flash-ROM on the controller to take place for upgrading the RoBios operating system.

A walking gait is the way in which an animal moves its legs to propel its way around. Several different gaits have been researched from different legged animals to determine the simplest technique to maneuver the hexapod mobile robot. The basic concepts behind walking gaits and the description/integration of the robot's servomotors are described in the following paragraphs.

Human locomotion is typically more complex than insect locomotion.[9] Humans use what are known as dynamically stable gaits. When the center of mass is plotted as a human walks, it looks like an inverted pendulum motion. Potential energy at the peak is converted to kinetic energy in the next step; the kinetic energy is recovered and converted back to potential energy again. By constantly adjusting the muscles in the body, humans are not very stable even when standing. This is what is meant by dynamic stability.

Unlike in dynamic stability, static stability is always stable and can be achieved with insects having six legs. This is because three of the legs are always kept on the ground in a configuration that keeps the insect from falling. The tripod gait is the fastest of the gaits. In it, the insect always has two legs on the ground on one side and one leg on the ground on the other side such that it forms a tripod. Thus, three of the legs are contacting the ground and moving backwards while the other three legs are raised and moving forward. As the feet on the ground move backward, the body of the insect moves forward. Then, when the raised legs are all the way forward, they are lowered to make contact and the legs that have been down are now raised. This whole procedure is repeated.[10]

Figure 2 shows the similarity of ground impact forces versus time for a range of animals. Each bump represents one step [half the total cycle]. One or more legs work together to distribute the forces over space and time during each step. For the roach, for instance, each bump represents one tripod step. The time phasing of the 3 tripod legs acting upon the ground – the front leg decelerating, the rear accelerating, and the middle doing both-results in the integrated curve shown. For the dog, each bump is the integrated force on a "diagonal"

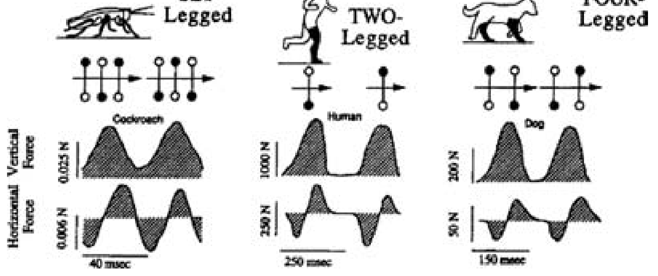

Fig. 2. Ground impact force for different legged animals.

[opposite corner legs]. Conceptually, this type of gait is both *statically* and *dynamically* stable. An arthropod resting on one tripod will not fall over. Compare this to a quadruped standing on one diagonal, or a biped standing on a single leg - both of these are dynamically, but not statically, stable. The tripod gait is the type of walking pattern that is implemented on the hexapod mobile robot in this paper.

Twelve Futaba servomotors are used to actuate each of the six robot legs during locomotion. These servos are normally used in radio controlled (RC) devices such as airplanes and cars. These small motors are ideal for low-cost robots and are very strong for their size (Torque = 44.4 oz-in at 4.8V and 56.9 oz-in at 6.0V). The servo is a small device with an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo maintains the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. The servomotor has three wires, power (+5V), ground and control that are connected to the controller, which is used to control an angular motion of 0 to 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built onto the main output gear. To communicate the angle at which the servo should turn is done by Pulse Coded Modulation. For every 20 ms, the controller sends a pulse (1-2 ms) through the control line that tells the motor to rotate by a certain amount.[11]

Numerous experiments lead to the determination of the amount of rotation needed for the robot to successfully transverse in a Forward, Backward, Left and Right direction so that "drifting" to one direction is minimized. Rotation of the motor to control the up/down movements of the legs is kept between 188° (up) and 68° (down). The forward/back movements are kept between 98° (forward) and 158° (back), with 128° representing the (neutral) position. These angles are determined to be the best solution for movements, because if the speed of the legs (speed being one of the fuzzy outputs) is either too slow or too fast, the robot demonstrates difficulties with walking.

Sharp's GP2D02 detectors are used because of the availability of not only object detection but also range information. The sensors operate by using an infrared LED emitter and a PSD detector. These detectors use triangulation and a small linear CCD array to detect the distance and/or presence of objects in the field of view. Because of the basic trigonometric relationship in the triangle formed by the emitter, the reflection spot and the receiver, the output of these detectors is nonlinear with respect to the distance being measured (Figure 3). Due to this non-linearity, the sensor response is modeled using the following equations:

$$\begin{aligned} y &= -0.25x + 95.00 \qquad x \in [60, 120] \\ y &= -1.14x + 148.57 \qquad x \in [25, 60] \\ y &= -5.33x + 253.33 \qquad x \in [10, 25] \end{aligned} \tag{2-1}$$

where y is the output of the sensor and x is the real distance to an object. The output of the sensor is an 8-bit value.[12] The graph in Figure 3 demonstrates that objects between 10 cm – 80 cm is somewhat logarithmic. It's important to note that at distances less than 10 cm, the output drops sharply and is

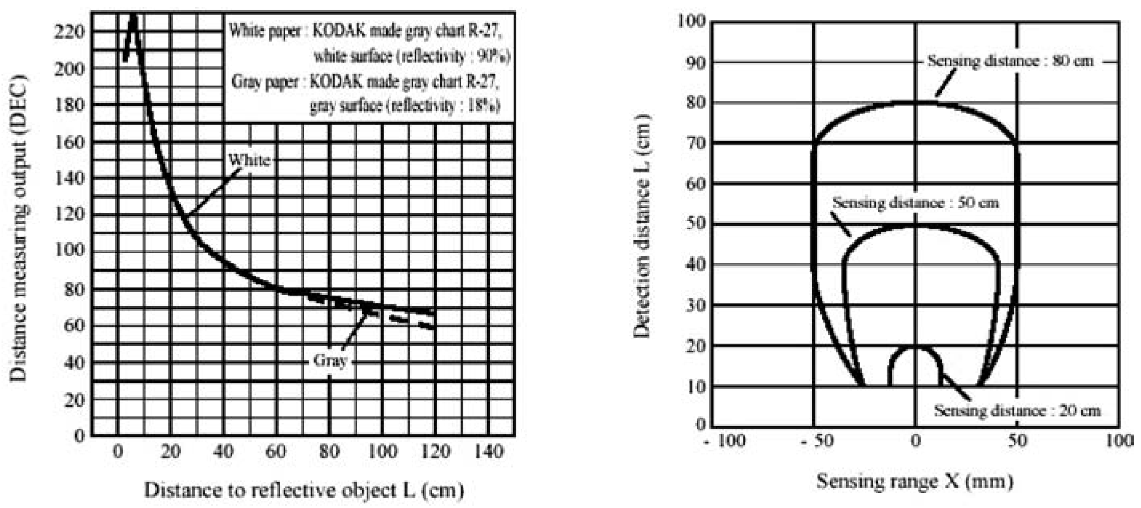

Fig. 3. Sharp GP2D02 distance output (left) and sensing range (right).

seen as a longer-range reading. This will be a problem if the robot approaches an object, slows down and then speeds up as it reaches the sensor's minimum threshold.

There are three fundamental kinds of error when using these types of sensors:

- Object color reflectivity – As the light reflectivity varies with color, the darker the object the further away it appears.
- Global illuminance – As regular illumination contains light in the infrared band, the brighter the light the further away the object appears.
- Quantization error – This sensor converts the measure to 8 bits. This conversion is not linear in the full range, and thus, the further away the object, the less accurate the reading.

Normalization of the output is completed with a lookup table in the robot's HDT software to eliminate this possibility. Values in the table are adjusted to coincide efficiently with the fuzzy algorithm membership functions that are used to control the robot's input response.

The detection area for the MKIII hexapod mobile robot can be seen in Figure 4. The green lines show the mid-line and direction of the three sensors, with the widest portion of the beam at ~16 cm.

The three IR sensors are tested to determine the calibration of the sensors. Three different objects are used, a white book (with reflective lamination), a human hand, and a black book. Each object is placed at specified distances: the minimum detection distance, 15 cm, 31 cm, 46 cm, 61 cm and 76 cm, from each of the sensors labeled LEFT, FRONT, and RIGHT. Each object is set in four positions: dead-on (MID), rotated 5° left (A), rotated 5° right (B) and leaned back 5°(C). This determines how much variation in the sensor output would be observed because of the reflectivity errors described earlier.

When a mobile vehicle has a faster travel velocity, it would be wise to use the sensors' full capability to detect objects at a farther range so as to prevent collisions. Due to the characteristics of the fuzzy program, values that are measured out of the membership function ranges are viewed as 0. This causes the robot to react to objects that are very far as if being very close. To prevent this mistake from occurring, coding is added to convert any value greater than 400 to be 400 – the maximum distance perceived.

One can easily determine that a "low-cost" robotic system is equipped with items that have a significant degree of variation. This would make the task of deriving a mathematical model extremely difficult. Variations in the manufacturing of the motors and legs as well as deviations in the sensor distance depending on the type of objects detected, lead to obvious problems in real experiments. Overcoming these dynamic problems is the main effort of our project.

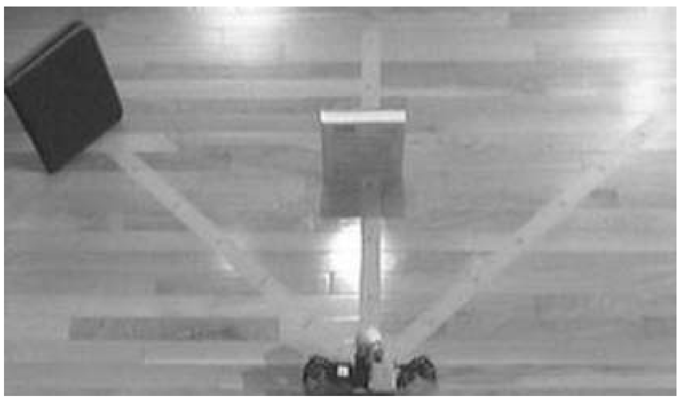

Fig. 4. Sensor detection area.

III. FUZZY SYSTEMS

With the advent of computer technology and the increasing research on new practicality of mathematics and applications, fuzzy logic has allowed researchers to sidestep the traditional approach of finding the perfect mathematical equation for dynamic control systems. For solutions to be relevant in the modern world, computer control systems must be able to work with imprecise terms.

L. Zadeh states in his paper[13] that most collections of objects that are encountered in the real world are not precisely defined. The concept of fuzzy sets is the rigorous precision of mathematics managed with the imprecision of human expressions and thoughts. Rather than a single

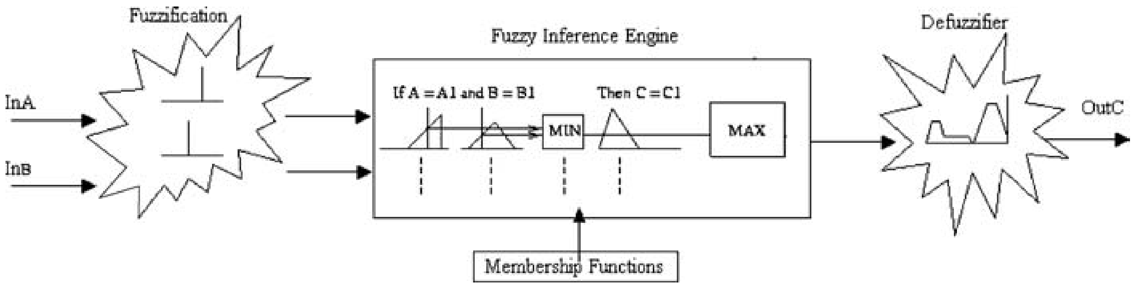

Fig. 5. Diagram of a fuzzy inference system

theory, the fuzzy theory should be regarded as a process of "fuzzification" or a methodology to generalize any specific theory from a crisp (discrete) to a fuzzy (continuous) form.[14]

From subway systems, to medical diagnosis, to engineering of automatic transmissions, Japanese programmers have embraced this form of control and have implemented it in a variety of technology used today.

Humans do not calculate equations in their heads when performing an operation such as driving a car. They generalize by using linguistic terms such as "The object is CLOSE." or "The car is going VERY SLOW." Figure 5 demonstrates the block diagram of a fuzzy inference system.

Fuzzy logic application to a problem involves 5 steps[15]:

a. Fuzzify Inputs – Take the inputs and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions.
b. Apply Fuzzy Operators – Once inputs have been fuzzified, the degree to which each part of the antecedent has been satisfied is known.
c. Apply the Implication Method – This is the shaping of the output membership functions on the basis of the firing strengths of the rule. The input is a single number and the output is a fuzzy set.
d. Aggregate all Outputs – This is where all the outputs of each rule are unified. The fuzzy output from the implication process is inputted to this process and the output of aggregation is the combined output fuzzy sets.
e. Defuzzify – A crisp value is obtained from the fuzzy set through the use of a centroid, height or maximum method.

IV. IMPLEMENTATION OF FUZZY CONTROL ON THE HEXAPOD MOBILE ROBOT

When programming a fuzzy system with fuzzy rules, the system designer provides his own qualitative understanding of the problem. The overlap usually present between fuzzy sets means that more than one rule applies to a given set of inputs. This overlap provides fuzzy systems with the ability to generalize between rules. As a result, smooth transitions on the control surface occur between recommendations specified by the programmer. This feature allows the programmer to develop sophisticated nonlinear control functions without the use of mathematical expressions. The following form is used in the rule base:

If x_i is A_i^m, then y_i is B_i^m

where A_i^m and B_i^m are fuzzy sets. In this paper, the linguistic terms used to describe the membership function titles for the input sensor data are as follows:

Inputs:	NEAR (NC)		(4-1)
	CLOSE (CC)		
	FAR (FC)		
Outputs:	*Speed*	*Steering*	
	No Move (NM)	Forward (STR_FW)	
	Slow Move (SM)	Turn Right (STR_RT)	(4-2)
	Medium Move (MM)	Turn Left (STR_LT)	
	Fast Move (FM)		
	Very Fast Move (VM)		

These membership functions are Gaussian and are described as:

$$\mu_A(x_i) = e^{\left(\frac{x_i - a}{\sigma}\right)^2} \tag{4-3}$$

where A represents one of the fuzzy sets in (4-1) and (4-2) and x_i represents the input or the output. α is the center of A and σ is the width. The fuzzy membership functions can be seen graphically in Figure 6 for the inputs and Figure 7 for the outputs.

The Gaussian exponential function is used for these membership functions. Table I demonstrates the 27-fuzzy rules used to describe the If-Then reasoning of the algorithm.

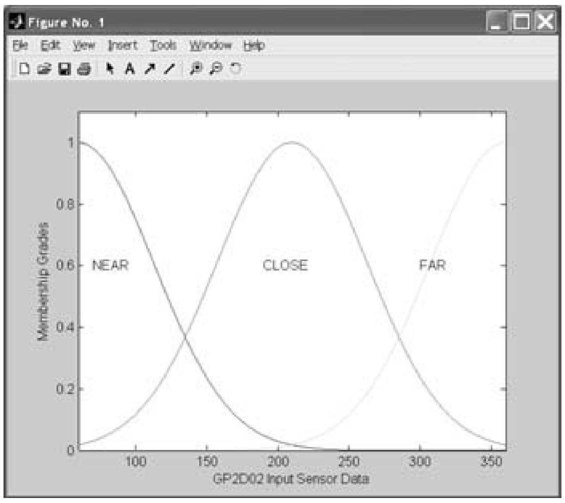

Fig. 6. Input membership functions.

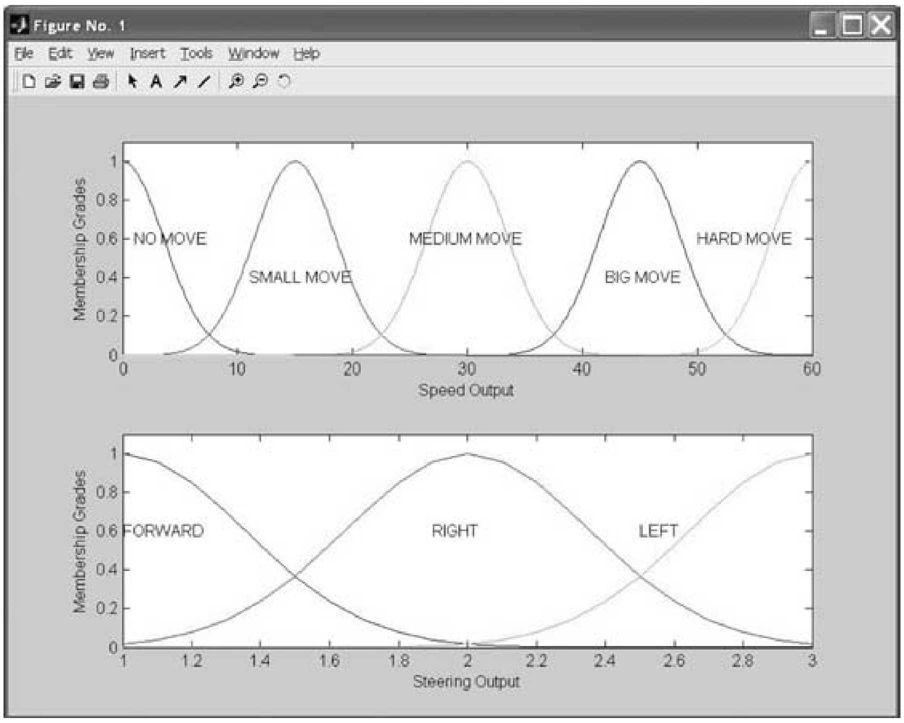

Fig. 7. Output membership functions.

Table I. Fuzzy rule base.

Rm:	INPUT *Right Sensor*	INPUT *Center Sensor*	INPUT *Left Sensor*	OUTPUT *Speed*	OUTPUT *Steering*
1	*If* X1 is ***NEAR***. . . *AND*	X2 is ***NEAR***. . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **NM**	y2 is **LT**
2	*If* X1 is ***NEAR***. . . *AND*	X2 is ***NEAR***. . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **NM**	y2 is **LT**
3	*If* X1 is ***NEAR***. . . *AND*	X2 is ***NEAR***. . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **NM**	y2 is **LT**
4	*If* X1 is ***NEAR***. . . *AND*	X2 is ***CLOSE***. . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **NM**	y2 is **FW**
5	*If* X1 is ***NEAR***. . . *AND*	X2 is ***CLOSE***. . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **MM**	y2 is **LT**
6	*If* X1 is ***NEAR***. . . *AND*	X2 is ***CLOSE***. . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **MM**	y2 is **LT**
7	*If* X1 is ***NEAR***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **MM**	y2 is **FW**
8	*If* X1 is ***NEAR***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **MM**	y2 is **FW**
9	*If* X1 is ***NEAR***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **FM**	y2 is **FW**
10	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***NEAR*** . . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **NM**	y2 is **RT**
11	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***NEAR*** . . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **SM**	y2 is **RT**
12	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***NEAR*** . . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **MM**	y2 is **LT**
13	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***CLOSE*** . . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **MM**	y2 is **FW**
14	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***CLOSE*** . . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **FM**	y2 is **FW**
15	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***CLOSE*** . . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **VM**	y2 is **LT**
16	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **VM**	y2 is **FW**
17	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **FM**	y2 is **FW**
18	*If* X1 is ***CLOSE***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **FM**	y2 is **FW**
19	*If* X1 is ***FAR***. . . *AND*	X2 is ***NEAR*** . . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **NM**	y2 is **RT**
20	*If* X1 is ***FAR***. . . *AND*	X2 is ***NEAR*** . . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **NM**	y2 is **RT**
21	*If* X1 is ***FAR***. . . *AND*	X2 is ***NEAR*** . . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **VM**	y2 is **RT**
22	*If* X1 is ***FAR***. . . *AND*	X2 is ***CLOSE*** . . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **SM**	y2 is **RT**
23	*If* X1 is ***FAR***. . . *AND*	X2 is ***CLOSE*** . . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **MM**	y2 is **FW**
24	*If* X1 is ***FAR***. . . *AND*	X2 is ***CLOSE*** . . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **MM**	y2 is **FW**
25	*If* X1 is ***FAR***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***NEAR***. . . *THEN*	y1 is **SM**	y2 is **FW**
26	*If* X1 is ***FAR***. . . *AND*	X2 is ***FAR***. . . *AND*	X3 is ***CLOSE***. . . *THEN*	y1 is **FM**	y2 is **FW**
27	*If* X1 is ***FAR***. . . *AND*	X2 is ***FAR*** . . . *AND*	X3 is ***FAR***. . . *THEN*	y1 is **VM**	y2 is **FW**

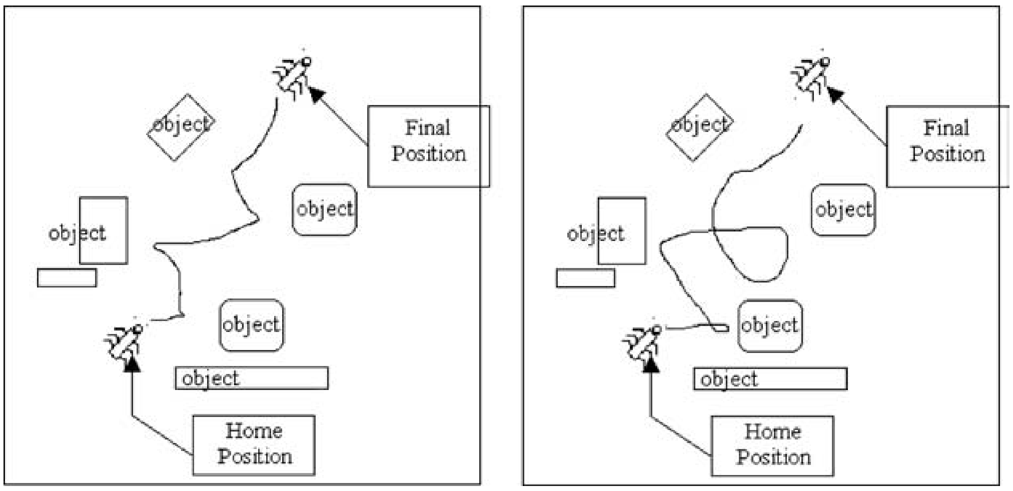

Fig. 8. Two paths taken by the robot during the experiment.

After these initial steps are completed, the fuzzy system can be determined. Applying the product inference engine, singleton fuzzification, center average defuzzification and Gaussian membership function, we have

$$y_i = \frac{\sum_{m=1}^{M} y_i' \mu_{A^m}(x_i)}{\sum_{i=1}^{M} \mu_{A^m}(x_i)} \qquad (4\text{-}4)$$

where y_i' is the center of the membership function B^m for y_i in the mth rule, $\mu_{A^m}(x_i)$ is the membership function of x_i in the mth rule and $M = 27$ is the number of rules. The resulting crisp output is calculated and MKIII then implements both outputs steering and speed. Results of trial run experiments are described in the next section.

V. EXPERIMENTAL RESULTS

Using the aforementioned fuzzy architecture, a series of real robot runs are executed to test the performance of the robot. MKIII is placed in a room containing a variety of objects of different material types and locations (Figure 8).

The distance between objects is relatively small, roughly 30 cm – 80 cm, which forces MKIII to test its obstacle avoidance ability. The robot starts at the bottom left and eventually works its way up to the top right corner of the room. In some instances the robot turns around in a circle when it approaches some of the objects before eventually reaching the same destination as before (in Figure 8 (right)). The robot does successfully travel around without colliding with any of the objects. The snap shots of these sequences can be seen in Figure 9. Although the basic behaviors of MKIII are primitive with the use of the fuzzy algorithm described, the performance results in a powerful activity. For example, when the robot is placed into a room with obstacles in random locations, the obstacle avoidance ability allows the robot to successfully navigate around the obstacles. Simple concave obstacle arrangements are easily escaped, and dynamic obstacles are dealt with in real time.

During the beginning trials of the fuzzy rule base, the robot got trapped in certain area patterns where it got stuck in the same movement loop for an indefinite amount of time. This is called the local minima, where the robot would jump between two fuzzy rule outputs such as MOVE FORWARD and TURN RIGHT. Adjusting a few of the rules so that the transitions between the different output movements become more gradual between sensor input data solves this problem.

VI. CONCLUSIONS

Fuzzy logic has features that are particular attractive in light of the problems posed by autonomous robot navigation. Fuzzy logic allows us to model different types of uncertainty and imprecision. It builds robust controllers starting from heuristic and qualitative models by integrating symbolic reasoning and numeric computation in a natural framework. In this paper, a fuzzy control system for a hexapod mobile robot navigating in unknown environments is presented. The robot, MKIII, interprets input sensor data through the comparison of values in its fuzzy rule base. After the fuzzy system aggregates the fuzzy membership functions from the sensor inputs, it then defuzzifies membership values and creates two crisp outputs: speed and steering. MKIII is then placed into a test area where it successfully navigates around without bumping into any of the objects.

Minor adjustments to the membership functions are made to decrease the local minima problem that occurs with certain input combinations. The speed and steering membership functions used have proved to be successful after a few adjustments.

The advantages of using a fuzzy control system in a dynamic, real environment are that the system is able to compensate for errors that occur with sensor input data, differences in mechanical design and component tolerances. MKIII proves through the use of a fuzzy algorithm architecture, that a mobile robot is "intelligent" enough to transverse through an unknown environment by adjusting its speed and direction to avoid obstacles. It accomplishes this with the data that it receives from just three input photosensors and completes the real-time adjustments to the motor speed and steering.

MKIII's fuzzy control structure can be modified easily by changing the rule base or the membership functions within

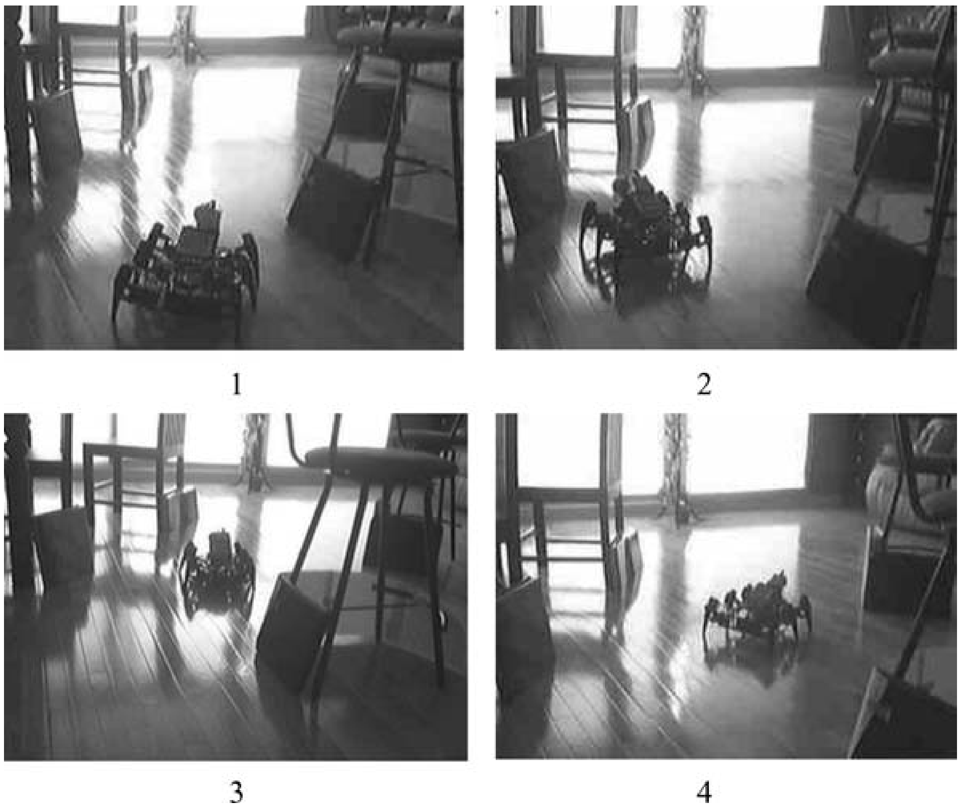

Fig. 9. Image sequence of MKIII maneuvering through test area.

the algorithm. The flexibility of the system is an important factor for future research and development.

Some ideas that could be used for future works include the following:

- Different mobile designs for faster travel velocities
- Different fuzzy control algorithms with different types of membership functions to self-tuning behaviors, where the fuzzy sets and rules can be adjusted automatically by the system itself and not by the programmer.[16]
- Neural-fuzzy systems, where the synergistic benefits of merging these two technologies can be a powerful tool in artificial intelligence and control.[17]

Through the successful implementation of an intelligent fuzzy system into a hexapod mobile robot, this paper demonstrates how the unification of mathematics, computers and nature is changing the field of robotics as well as the world around us.

References

1. L. Tsoukalas and R. Uhrig, *Fuzzy Neural Approaches in Engineering* (John Wiley and Sons, Inc. 1997).
2. J. Principe, N. Euliano and C. Lefebvre, *Neural And Adaptive Systems – Fundamentals through Simulation* (John Wiley and Sons, Inc., 2000).
3. B. Eriksson, *A Survey on Dynamic Locomotion for Legged Vehicles* (Royal Institute of Technology, Sweden, 1998).
4. R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, H. B. Brown Jr., D. McMordie, U. Saranli, R. Full and D. E. Koditschek, “RHex: A Biologically Inspired Hexapod Runner,” *Autonomous Robots* **11**, 207–213 (2001).
5. R. Brooks, “A Robot that Walks: Emergent Behaviors from a Carefully Evolved Network,” *Proceedings of the IEEE International Conference on Robotics and Automation* (1989) **Vol. 2**, pp. 692–694.
6. C. Ferrell, “A Comparison of Three Insect Inspired Locomotion Controllers,” *MIT AI Technical Report 1443* (May 1993).
7. J. Vandorpe and H. Van Brussel, “A Reflexive Navigation Algorithm for an Autonomous Mobile Robot,” *IEEE Conference on MultiSensor Fusion and Integration for Intelligent Systems* (1994) pp. 251–258.
8. T. Bräunl, *Eyebot MK3 Controller* (The Univ. of Western Australia. Department of Electrical and Electronic Engineering, Crawley, Perth, WA. Australia, 1999).
9. G. S. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto and M. Fujita. “Evolving Robust Gaits with Aibo,” *Proceedings of the IEEE International Conference on Robotics and Automation* (2000) pp. 3040–3045.
10. R. J. Full and C. T. Farley, “Musculoskeletal Dynamics in Rhythmic Systems: a Comparative Approach to Legged Locomotion,” *Biomechanics and Neurocontrol of Posture* (Springer, 2000).
11. J. Jones and A. Flynn, *Mobile Robots – Inspiration to Implementation* (AK Peters, Ltd., 1993).
12. Sharp Datasheet, GP2D02 Compact, High Sensitive Distance Measuring Sensor (1997).
13. L. Zadeh, “Fuzzy Sets,” *Information and Control* **8**, 338–353 (1965).
14. L. Zadeh, “Outline of a New Approach to the Analysis of Complex Systems,” *IEEE Trans. on Systems, Man and Cybernetics* **3**, (1973) pp. 22–44.
15. A. Kulkarni, *Computer Vision and Fuzzy Neural Systems* (Prentice Hall, 2001).
16. R. Bicker, Z. Hu and K. Burn, *A Self-tuning Fuzzy Robotic Force Controller* (Dept. of Mech. Engineering, University of Newcastle, Dept. of Computer Science & Technology, 2002).
17. S. Huang and W. Ren, “Use of Neural Fuzzy Networks with Mixed Genetic/Gradient Algorithm in Automated Vehicle Control”, *IEEE Transactions on Industrial Electronics* **46**, No. 6, 1090–1102 (December, 1999).