Journal of Intelligent and Robotic Systems **19:** 339–356, 1997.

Neurofuzzy Motion Planners for Intelligent Robots

L. H. TSOUKALAS and E. N. HOUSTIS
Purdue University, W. Lafayette, Indiana 47907-1290, U.S.A. e-mail: tsoukala@ecn,purdue.edu

G. V. JONES
The University of Tennessee, Knoxville, Tennessee 37996, U.S.A.

(Received: 19 January 1997; accepted 4 February 1997)

Abstract. A neurofuzzy methodology is presented for motion planning in semi-autonomous mobile robots. The robotic automata considered are devices whose main feature is incremental learning from a human instructor. Fuzzy descriptions are used for the robot to acquire a repertoire of behaviors from an instructor which it may subsequently refine and recall using neural adaptive techniques. The robot is endowed with sensors providing local environmental input and a neurofuzzy internal state processing predictable aspects of its environment. Although it has no prior knowledge of the presence or the position of any obstructing objects, its motion planner allows it to make decisions in an unknown terrain. The methodology is demonstrated through a robot learning to travel from some start point to some target point without colliding with obstacles present in its path. The skills acquired are similar to those possessed by an automobile driver. The methodology has been successfully tested with a simulated robot performing a variety of navigation tasks.

Key words: intelligent robots, instructible robots, anticipatory systems, motion planners, neurofuzzy control, collision avoidance.

1. Introduction

As robotic technology makes its way off the factory floor and into homes and work environments that are both unstructured and dynamic, the need for robots that can perform a variety of tasks in different settings is rapidly growing. The complexities of such environments present formidable computational and modeling challenges and invite a new human role in directing a robot's operation, i.e., instructing a robot what to do. As with a child who is taught a new skill, a robot may acquire new skills through detailed human guidance and repetition. Instructible robots learning new tasks chosen by the user in his own environment derive their intelligence, at least in part, from the interaction with the user [6]. Such robots may acquire new skills and improve existing ones through *learning by being told* and *learning by doing*.

In the approach presented here, *learning by being told* is accomplished primarily through fuzzy descriptions, while *learning by doing* through neural adaptation. The robot learns a sequence of (fuzzy) commands from an instructor that can subsequently be invoked by name alone, and adapts them to future specific situations through modifying an internal neural representation of the sequence. For example, a robot that has learned a sequence of detailed instructions for 'cleaning

the floor' (in a specific house by a specific user), when the command 'clean the floor' is issued in the future, it does not require detailed programming of the task. Having this behavior in its repertoire, it employs a neural-type adaptation to solve problems due to possible changes on the floor, and performs the task autonomously.

There are two major approaches taken to intelligent robot design. One emphasizes local sensor information and the other stored internal state [7]. The first (often referred to as the *classical approach*) assumes that an intelligent robot needs a significant number of world models, including models of its perceptual environment, models of other agents, and models of itself. The second approach (often referred to as the *reactive approach*) views robots as a collection of relatively independent (primitive) behaviors that are largely sensor driven and require minimal modeling. Our approach attempts to reconcile both classical and reactive approaches by integrating fuzzy and neural tools for the purpose of incrementally adding and maintaining modeling capabilities in a flexible and learn-as-needed manner.

Many tasks for which we would use a robot are well-suited to instruction. For example, a robot navigating with reference to objects in the environment may benefit from user directions and advice in motion planning. As far as the later is concerned, instructible mobile robots may be viewed as versatile mechanical devices equipped with actuators and sensors whose main goal is to travel from some start point $\mathbf{q}_{\text{init}}$ to some target point $\mathbf{q}_{\text{goal}}$ without colliding with obstacles present in their path. Such robots are of immense interest in a wide variety of applications including manufacturing, hazardous waste processing, space exploration, construction, telematic surgery and assistance for the disabled and considerable progress has been made in recent years towards their development [2, 3, 13, 16, 20]. Motion planning, a relatively simple task for humans, turns out to be quite difficult to achieve in robots and raises difficult computational questions especially in connection with problems such as real-time motion control, sensing and task planning. A variety of methodologies for motion planning is found in the literature. Most of them may be categorized as variants of three major approaches: *roadmaps, cell decomposition* and *potential field* methods.

The roadmap approach consists of capturing the connectivity of the robot's free space in a network of one dimensional curves, called the *roadmap* R; and motion planning is reduced to connecting the initial and goal configurations to points in R and searching R for a path between these points. Several methods are used to compute different types of roadmaps, e.g., *visibility graphs, Voronoi diagrams, freeway diagrams* and *silhouettes* [13]. Cell decomposition approaches, on the other hand, consist of decomposing the robot's free space into simple regions called *cells*, such that a path between any two configurations in a cell can be easily generated. A non-directed graph called the *connectivity graph* is used to represent the adjacency relation between the cells and is constructed and

searched to produce a sequence of cells called *channel* out of which a continuous free path can be computed.

Both roadmaps and cell decomposition methods capture the global connectivity of the robot's free space into a condensed graph which is subsequently searched for a path. *Potential field methods* treat the robot as a particle under the influence of an artificial potential field U representing the 'structure' of the free space. The potential function is typically (but not necessarily) defined over free space as the sum of an *attractive* potential pulling the robot toward the goal configuration and a *repulsive* potential pushing the robot away from the obstacles. Potential field methods were originally developed as on-line collision avoidance approaches, applicable when the robot does not have a prior model of obstacles but senses them during motion [9, 10, 11]. When obstacles in the workspace are moving, robots have to deal with a *dynamic motion planning problem*, which unlike its static counterpart is rather difficult to solve by roadmaps, cell decomposition, or potential field methods.

Many researchers have investigated the problem of a robot moving from one place to another in unknown terrain while avoiding a set of obstacles [1, 13, 15]. Fuzzy methodologies for motion planning in unknown terrain using fuzzy rules for quantifying a robot's reactive behavior and for coordinating different types of reactive behaviors possible have also been developed [4, 17, 18]. A heading angle (the angle between the robot and a specified target) as well as distances between the robot and the obstacles to the left, front, and right locations, acquired by an array of ultrasonic sensors are inputs to the robot. The outputs are commands for the speed control unit of rear wheels; the robot performs reasonably well in a dynamic motion planning situation.

In order to study motion planning in an unknown terrain we developed an idealized robot called *MITOS*, equipped with a motor similar to that of a wheelchair, moving in a path with obstacles presented arbitrarily. *MITOS* can steer left, right, straight or in any direction in between and its speed can be controlled. The motor control variables, i.e., *steering* and *speed* depend on the contingencies of the environment, since the physical environment of the robot may be cluttered with objects. For a static environment where the position of objects in a terrain is known, motor control can be fully programmed *a priori*, with the motion planning process usually satisfying various optimality criteria. *MITOS* is capable of operating in a changing environment where the position of the objects are not known, motor control is dynamically evaluated, and optimization is a secondary concern. In the approach taken here, basic robotic behaviors, such as locomotion and object avoidance are built through the robot's interaction with both the instructor and the environment. The overall architecture of the approach is schematically illustrated in Figure 1.

The remainder of this paper is organized as follows. In Section 2 we introduce and review the necessary fuzzy, neural and neurofuzzy terminology. In Section 3, we examine specialized neurons built via fuzzy tools. They provide an adaptable

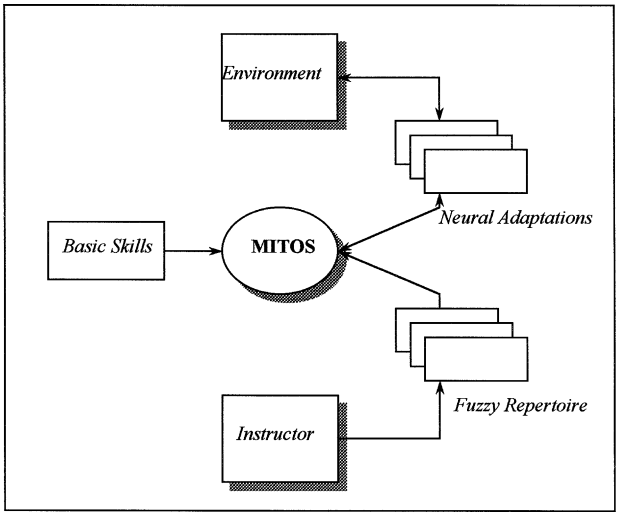

Figure 1. Architecture of an intelligent robot learning through instruction and neural recall.

representation of the fuzzy repertoire. In Section 4, we present an example of neurofuzzy motion planning and in Section 5 final comments and our plans for future work.

2. Review of Neural, Fuzzy and Neurofuzzy Concepts

A neural network consists of densely interconnected information processing units called *neurons*. Each neuron has *external inputs, synapses, dendrites*, a cell body or *soma*, and an *axon* through which individual neural output is transmitted to other neurons. Let us consider the jth neuron of the network. A vector of external inputs $[x_1, x_2, \ldots, x_n]^T$ enters the jth neuron and gets modified by weights $w_{1j}, w_{2j}, \ldots, w_{nj}$ representing the synaptic junctions of the neuron. In general, the synaptic weights may be functions of the external inputs, i.e., $w_{1j}(x_1), w_{2j}(x_2), \ldots, w_{nj}(x_n)$. Each synaptic output constitutes an input to the soma called the *dendritic input.* Thus, the input to the jth neuron's soma is the vector of dendritic inputs $[d_{1j}, d_{2j}, \ldots, d_{nj}]^T$, where each dendritic input is a transformed version of an external input x_i, i.e.,

$$d_{ij} = w_{ij}(x_i). \tag{1}$$

The weighting function $w(\cdot)$ that models the synaptic junction between the axon of the transmitting neuron and the dendrite of the receiving neuron, is thought of as a memory of the neuron's past experience, capable of adapting to new experiences through learning.

The ith neuron produces an output response when the aggregate activity of all dendritic inputs exceeds some threshold level T_j. Mathematically, this is expressed as

$$I_j = \sum_{i=1}^{n} d_{ij}, \tag{2}$$

where n is the number of dendritic inputs to the neuron. Besides summation, other aggregation operators, for example *min*, *max*, and more generally T-norms and S-norms may be used [19].

Finally, the output y_j of the jth neuron is produced by the other essential operation within a neuron's soma, which is that performed by the *activation function* Φ_j (really a *decision function*). The neural output y_j is mathematically expressed as

$$y_j = \Phi_j[I_j, T_j], \tag{3}$$

where Φ_j is the activation function that describes the degree to which the jth neuron is active, I_j is the total aggregate input activity incident on the *soma* of the neuron and T_j is the inherent threshold level for this neuron.

Fuzzy sets are mappings from a universe of discourse X to the interval $[0, 1]$ defined through membership functions, $\mu_A(x)$: $X \rightarrow [0, 1]$, where A is a given fuzzy set [21, 22, 23]. Although both fuzzy and neural approaches posses remarkable properties when employed individually, there are great advantages to using them synergistically resulting in what are generally referred to as *neuro-fuzzy* approaches [19]. Quite often, a strong point of fuzzy systems turns out to complement nicely a weakness – so to speak – of neural networks and conversely. This may not be so surprising after all, since both neural and fuzzy approaches have had strong biological and cognitive inspirations. In the realm of human intelligence there is certainly a synergism between the neuronal transduction and processing of sensory signals, on the one hand, and the cognitive, perceptual and linguistic higher functions of the brain, on the other.

3. Specialized Fuzzy Neurons and Networks

A fuzzy neuron has the same basic structure as a neuron, except that some or all of its components and parameters may be described through the mathematics of fuzzy sets. There are many possibilities for fuzzification of an artificial neuron and hence one encounters a variety of fuzzy neurons in the literature, all possessing interesting logic-oriented information processing properties. Figure 2 shows a fuzzy neuron, where the external input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in R^n$ is defined over the unit hypercube $[0, 1]^n$ and is comprised of fuzzy signals bounded by graded membership over the unit interval $[0, 1]$. The external inputs, after being modified by synaptic weights w_{ij} (also defined over the unit interval), become dendritic inputs d_{ij} to the soma. Input modification may be done through

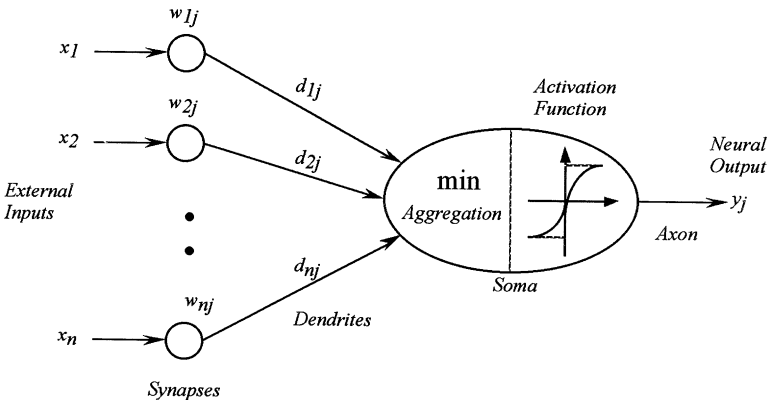

Figure 2. An example of a *min* (*AND*) fuzzy neuron.

straightforward multiplication $d_{ij} = x_i w_{ij}$ or taking the maximum between input and weight $d_{ij} = x_i \vee w_{ij}$ (i.e., like an OR-gate).

The dendritic inputs are processed by an aggregation operator I_j that selects the minimum $(\wedge)$ of the product (or max) modifications, for example

$$I_j = \bigwedge_{i=1}^{n} d_{ij} = \bigwedge_{i=1}^{n} x_i w_{ij}. \tag{4}$$

This type of fuzzy neuron may be thought of as the implementation of fuzzy conjunction (*AND-gate*). Generally, fuzzy neurons use aggregation operators such as *min* and *max* and more generally *T-norms* and *S-norms* instead of summation.

Each fuzzy neuron may be thought as the representation of a linguistic value such as *WIDE*, *FAR_AWAY*, etc. Hence the output of the neuron y_j in Figure 2 could be associated with membership to a linguistic value, i.e., y_j expresses the degree to which the input pattern $[x_1, x_2, \ldots, x_n]^T$ belongs to a given linguistic category. In other words, the output y_j is a real value in the interval $[0, 1]$ indicating the degree to which the applied external inputs are able to generate the given linguistic value. The jth neuron after receiving n inputs $[x_1, x_2, \ldots, x_n]^T$ and producing an output y_j, can subsequently convey this degree to the $m - 1$ other fuzzy neurons in a network consisting of m neurons.

The synaptic operations, but most importantly the aggregation operator, and the decision function determine the character of a fuzzy neuron. Using different aggregation operators and decision functions results in fuzzy neurons with different properties. Thus, many different types of fuzzy neurons can be defined. Consider for example the following neurons [12]:

MAX (OR) FUZZY NEURON

A *max fuzzy neuron* is a neuron that uses an aggregation function that selects the maximum ($\vee$) of the dendritic inputs to the soma, i.e.,

$$I_j = \bigvee_{i=1}^{n} x_i w_{ij}. \tag{5}$$

(A max fuzzy neuron is an implementation of a *logical OR*, hence we also call it an *OR fuzzy neuron*.)

MIN (AND) FUZZY NEURON

A *min fuzzy neuron* is a neuron that uses an aggregation function that selects the minimum ($\wedge$) of the dendritic inputs, i.e.,

$$I_j = \bigwedge_{i=1}^{n} x_i w_{ij}. \tag{6}$$

(A *min fuzzy neuron* is an implementation of a logical *AND*, hence it can also be called a *AND fuzzy neuron*.)

In general, the weights, the activating threshold, and the output functions which describe the interaction between fuzzy neurons could be adjusted via a learning procedure resulting in neurons that are adaptive. The aim is, of course, to synthesize a fuzzy neural network capable of learning from experience.

In a fuzzy neuron the aggregation operator I_j is generally a T*-norm* mathematically expressed as

$$I_j = T_{i=1}^{n} \delta_{ij}, \tag{7}$$

where

$$\delta_{ij} = \begin{cases} d_{ij}, & \text{for excitatory inputs,} \\ \bar{d}_{ij}, & \text{for inhibitory inputs,} \end{cases} \tag{8}$$

where the inhibitory inputs are fuzzy complements of the excitatory inputs

$$\bar{d}_{ij} = 1 - d_{ij}. \tag{9}$$

Often, but not always, fuzzy neurons do not explicitly use a threshold; thresholding may instead be contained within the choice of the activation or decision function. The activation function Φ_j is a mapping operator that transforms the membership of the aggregate fuzzy set I_j into the fuzzy set of the neuronal response y_j. In a sense, this mapping operation corresponds to a linguistic modifier such as *VERY*, and *MORE-OR-LESS*. The role of this modification is to enhance or diminish the degree to which the external inputs give rise to the fuzzy value represented by the jth fuzzy neuron before becoming an external

input to neighboring neurons. Thus, a general expression of the response of the jth fuzzy neuron may be written as

$$y_j = \Phi_j[I_j] = \Phi[T_{i=1}^{n}\delta_{ij}], \tag{10}$$

where each dendritic input δ_{ij} is given by Equation (8).

If the decision function is assumed to be a linear relationship with unit slope, i.e., $y_j = I_j$, we have a simplified fuzzy neuron whose response can be written as

$$y_j = T_{i=1}^{n}\delta_{ij}. \tag{11}$$

The concepts of T-*norm* and S-*norm* (or T-*conorm*), originally used in the field of probability theory, provide a means for generalizing and parameterizing fuzzy set operations such as *union* and *intersection* as well as implication operators, fuzzy inferencing and fuzzy neurons [8].

A T-*norm* can be thought of as a circuit (gate) with two inputs (x_1, x_2) and one output $T(x_1, x_2)$, also written as $x_1 T x_2$. The most widely used T-*norm* is the *min*, i.e., $T(x_1, x_2) = x_1 \wedge x_2$ (but also, *algebraic product*, *bounded product*, *drastic product* are all T-*norms*).

An S-*norm* can be thought of as a circuit with two inputs (x_1, x_2) and one output $S(x_1, x_2)$, also written as $x_1 S x_2$. A very common S-*norm* is the *logical sum* or *max*, i.e., $S(x_1, x_2) = x_1 \vee x_2$ (but also, *the algebraic sum*, *bounded sum*, and *drastic sum* are useful S-*norms*).

The relationship between T-*norms* and S-*norms* is given by *fuzzy De Morgan's laws* which may be written as

$$\begin{aligned} T(x_1, x_2) &= \bar{S}(\bar{x}_1, \bar{x}_2), \\ S(x_1, x_2) &= \bar{T}(\bar{x}_1, \bar{x}_2), \end{aligned} \tag{12}$$

where T is the T-*norm* and S is the S-*norm* and the bar over the symbols indicates negation.

Let us consider the simplified fuzzy neuron, shown in Figure 3(a), using a linear transfer function and output given by Equation (10). For simplicity we assume that the dendritic inputs are directly received from the external inputs ignoring any weight function modifications. This neuron can be thought of as the realization of a T-*norm* operation. With the aid of Equations (12) this simplified fuzzy neuron can be used to construct a network of neurons such as the one shown in Figure 3(b) that realizes an S-*norm*. As indicated by the small circles in the left neuron of Figure 3(b) the inputs are first negated in a fuzzy sense and then aggregated by a T-*norm* aggregation.

The output of this neuron is complemented again by the second neuron, in accordance with (12). Thus the network of neurons in Figure 3(b) is a realization of an S-*norm*, made out of cascaded neurons that individually use T-*norm* for the aggregation operation.

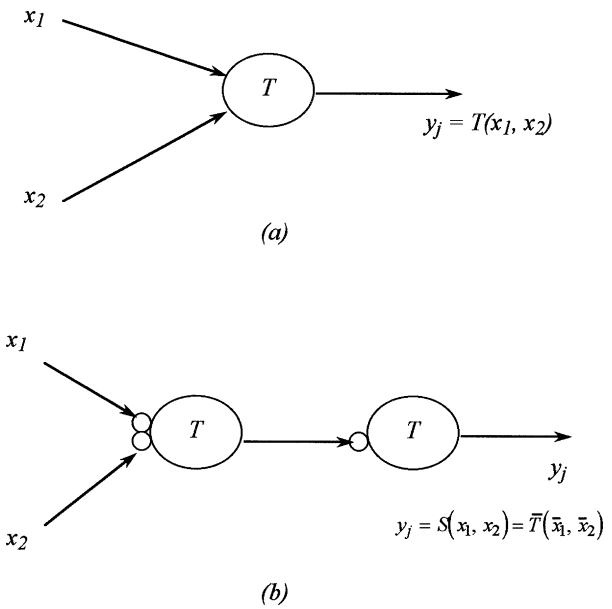

Figure 3. A simplified fuzzy neuron using T-*norm* aggregation and a linear transfer function can perform both (a) T-*norm* and (b) S-*norm* operations on the signals (x_1, x_2).

The process of learning in fuzzy neural networks consists of modifying their parameters by presenting them with examples of their past experience. In practice this is done by adjusting the weights of the network so that a certain performance index is optimized (maximized or minimized). This requires that a collection of input-output pairs be specified, in addition to a performance index that expresses how well the network maps inputs $\mathbf{x}_k$ into the corresponding target values of the output t_k.

In multilayer networks learning involves matching t_k (up to some error) with the output of the entire network y. For this purpose, a distance function, e.g., Euclidean distance between y and t_k is used. Then the performance index Q (actually a global error term to be minimized) reflects the state of the learning process as follows

$$Q = \sum_{k=1}^{N} [y(\mathbf{x}_k) - t_k]^2. \tag{13}$$

Optimization includes all the weights of the network between the input layer and the hidden layer as well as the hidden layer and the output layer. The simplest update scheme is that in which the modifications are driven by a gradient of the performance index taken with respect to the connections themselves. The learning formula can be expressed as

$$\Delta(\text{connections}) = -\eta \frac{\delta Q}{\delta(\text{connections})}, \tag{14}$$

where η denotes a *learning factor*, $\eta \in (0, 1)$. Detailed computations can be performed once the performance index and a parametric description of the network have been defined.

4. Example: Motion Planning in Dynamic Environments

MITOS is a HERO-like robot used for testing intelligent neurofuzzy motion planners. Relying on rough instructions about avoiding obstacles, the robot navigates though a building it has never visited. It is equipped with a sound and light sensor, motion detectors, capabilities for ultrasonic distance measurements and is self-propelled. It moves by means of three wheels, one in the front and two in the back. The front wheel provides steering and is the drive wheel. It is assumed that the path the robot travels has walls on both sides, it is a level floor, and has a limited number of intersections (e.g., a university building with long corridors and intersections).

MITOS is programmed to travel from some start point $\mathbf{q}_{\text{init}}$ to some target point $\mathbf{q}_{\text{goal}}$ along a path that may have moving obstacles as shown in Figure 4. It is assumed that the size of the path is the space between the road's left and right boundary. The automaton's sensing equipment can determine the outline of any objects ahead. These objects are defined by their size and position relative to the road. The robot's motion planner allows it to move around the obstructing object (by steering and adjusting its speed). In Figure 4, *MITOS* goes around obstacle A and tries to go back towards the right hand side of the wall. As we shall see later, this behavior may be altered after suitable instruction and the robot may recognize that the simultaneous presence of a corner and obstacle B calls for a different path.

An instructor teaches the robot that if the object is on the left side of the road the robot should swear right on the road to go around the object. Conversely, the robot is instructed to go on the left side of the object if the relative position is on the right-side of the road.

The robot continuously processes information about its environment and changes the speed and steering based on the behaviors of the motion planner. The required information includes the *size of the road*, the *size of the obstructing object* and *the relative position* of the object on the road. On a straight road the robot moves straight i.e. at a 90° angle. It can, however, move in any direction left or right. The speed of the robot generally depends on the width of the road. *MITOS* has been instructed to reduce speed when there is an obstruction in its path.

Human path planning involves fuzzy rules such as:

> *If the width of the road is about 'medium' and if there is a 'medium size object' at the 'right-side' of the road then go around the object by making a 'left-medium' turn at 'medium' speed.*

The robot encodes (or learns) from the intelligence used by a human automobile driver while operating a car. Using fuzzy values such as *MEDIUM, LEFT-*

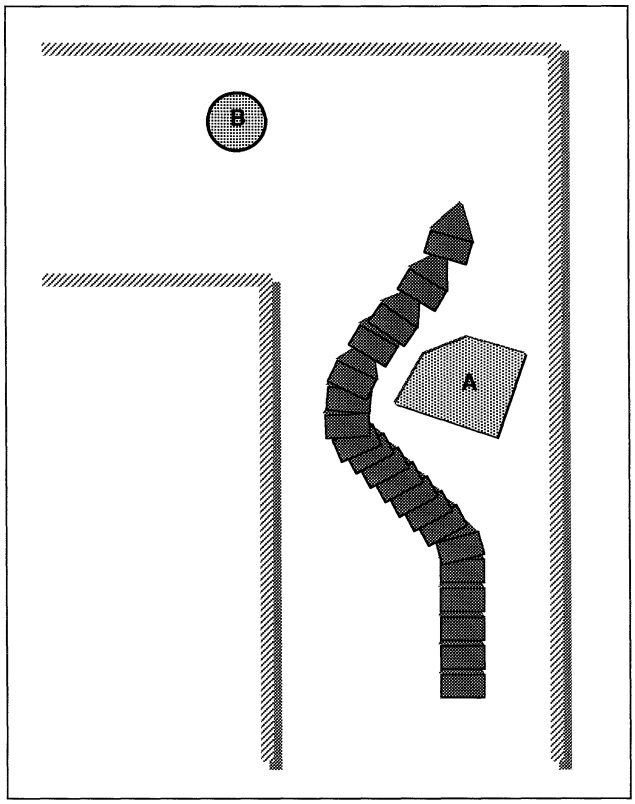

Figure 4. *MITOS*'s movement around an obstacle.

MEDIUM, etc., reduces the number of rules required to define the system and hence speeds up the process to a degree that real-time constraints may be met.

Through instruction the motion planner has acquired '27' rules representing a somewhat limited but typical repertoire for collision avoidance. The input to the robot is environmental data from the robot's sensors. The following left-hand-side variables were defined: the current *width* of the road (denoted as x_1), the size of the obstructing object (denoted as x_2), and the *position* of the obstructing object m (denoted as x_3). The output from the system is used to control the motor of the robot. This is defined by two action variables called *steering* (denoted as y_1) and speed (denoted as y_2). The value of *steering* defines the direction of *MITOS*'s movement. The robot can turn left or right or in any direction in between. The *speed* value defines the speed of the robot while it is making a turn. *Fuzzy* variables describe any object of interest to the system x, as a pair $\mu_A(x)/x$, where 'A' is a fuzzy category and $\mu_A(x)$ is the grade of membership of the object x in category A.

The following data objects are defined as fuzzy variables with a term set as indicated in the figures:

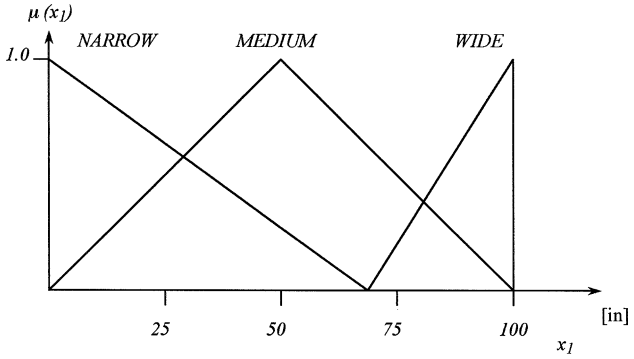

Figure 5. The values of the fuzzy variable x_1.

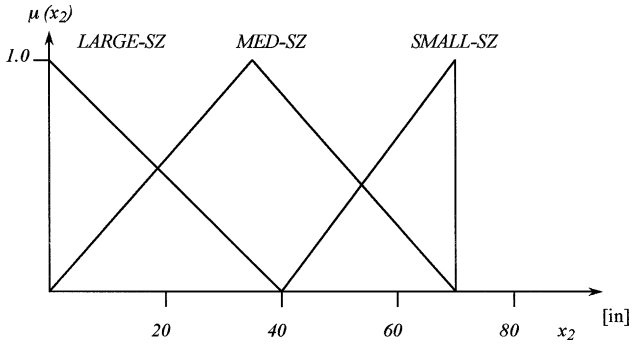

Figure 6. The fuzzy values of the variable x_2.

ROAD WIDTH (x_1)

The fuzzy values of x_1 are triangular fuzzy numbers $\{$*NARROW, MEDIUM, WIDE*$\}$ having membership functions as shown in Figure 5. The asymmetry observed with respect to the partition of the universe of discourse has to do with the fact that the robot ordinarily will stick to the right side of the road.

The *width* of a fuzzy category can be defined as the difference between the value whose membership equals '1' and the value whose membership equals '0'. The width of the fuzzy value *MEDIUM* as shown in Figure 5 is 30 inches. At any given time the crisp road width as determined from the robot's sensors has a fuzzy membership value in each of these categories. Fuzzy values for each category are calculated using Equation (15) bellow:

$$\mu(x) = 1 - \frac{|\text{peak\_value} - x|}{\text{width}}. \tag{15}$$

Based on Figure 5 a road width of 80 inches would belong to the category *MEDIUM* to a degree 0.7, to the category *WIDE* to a degree of 0.3 and would not belong at all to the category *NARROW*.

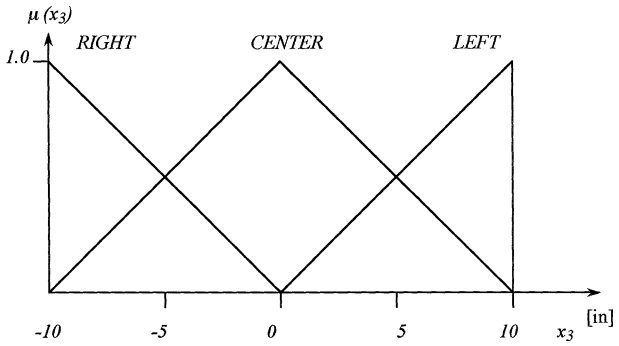

Figure 7. The fuzzy values of the variable x_3.

OBSTRUCTING OBJECT SIZE (x_2)

The robot's movement is controlled by the size of the obstructing object in its path. The size of the object is described as a fuzzy variable (x_1) whose values are shown in Figure 6.

OBSTRUCTING OBJECT POSITION (x_3)

The position of the obstructing object x_3 provides crucially important information for navigating *MITOS*. The robot can go around the object by either going left or right of the obstacle. If the object is on the left-side of the road the robot goes on the right-side of the road and if the object is on the right the robot goes on the left. The robot's steering control generally depends on the relative position of the obstacle in the road. Fuzzy variable x_3 has the values shown in Figure 7.

STEERING (y_1)

Steering control is an important piece of information for *MITOS*'s movement. Steering information is described in degrees. The steering-value can change from '30' i.e. a wide left-turn to '150' a wide right turn. In this system the fuzzy variable y_1 may take the seven values shown in Figure 8.

SPEED CONTROL (y_2)

This is a another data-object that drives the robot's motor control system. The speed value ranges from '0' to '60' in/sec. At '0' speed value the robot comes to a complete stop. This and the other values of y_2 are shown in Figure 9.

After the robot is instructed it constructs a fuzzy rule-base using the variables described above. An example of a rule is:

$$\begin{aligned}&\text{if } x_1 \text{ is } WIDE\ AND\ x_2 \text{ is } SMALL\_SZ\ AND\ x_3 \text{ is } LEFT\\ &\text{then } y_1 \text{ is } RIGHT\text{-}SMALL\ AND\ y_2 \text{ is } MEDIUM\text{-}SPEED,\end{aligned} \tag{16}$$

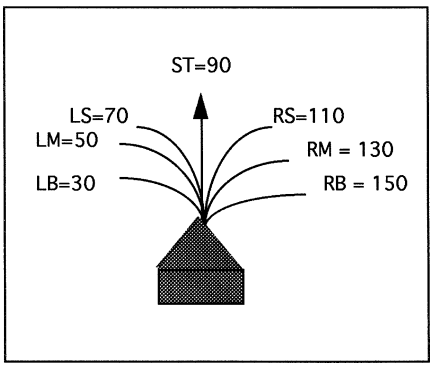

Figure 8. Steering control y_1 and its fuzzy values: ST (straight), LS (left small), LM (left medium), LB (left big), RS (right small), RM (right medium), RB (right big).

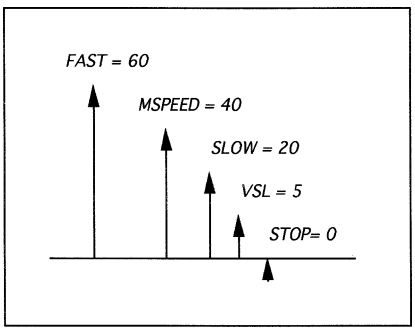

Figure 9. The values of the fuzzy variable y_2.

where the meaning is fairly clear. All rules have 3 fuzzy values in their *left hand side* and 2 values, *steering-control* and *speed*, in the right hand side.

MITOS uses *max-min composition* to evaluate its fuzzy rules. The membership value of a rule is the minimum membership value of all the fuzzy data-objects in the rule. The current road-width, the size of the obstructing object and the position of the obstructing object are the input to the system. These are precise values calculated from the robot's sensors. They are changed to fuzzy categories and the membership value in each category is calculated; e.g., the input values

$$\text{road} = 100, \qquad \text{object-size} = 65, \qquad \text{object-position} = 18$$

are converted to fuzzy values

$$\begin{aligned} \text{road} &= (WIDE/1.0, MEDIUM/0.0, NARROW/0.0), \\ \text{object-size} &= (SMALL\text{-}SZ/0.0, MED\text{-}SZ/0.2, \\ &\quad LARGE\text{-}SZ/0.8), \\ \text{object-position} &= (RIGHT/0.0, CENTER/0.2, LEFT/0.8). \end{aligned}$$

The resultant membership value is given by:

$$\mu(r_i) = \min\{\mu(x_1), \mu(x_2), \mu(x_3)\}. \tag{17}$$

The action-part is derived by taking an average of all matched rules (the average of all degrees of fulfillment (*DOF*'s))

$$\text{steering-angle} = \frac{\sum_i \mu(r_i) * \text{steering-angle}(r_i)}{\sum_i \mu(r_i)} \tag{18}$$

and

$$\text{speed} = \frac{\sum_i \mu(r_i) * \text{speed}(r_i)}{\sum_i \mu(r_i)}. \tag{19}$$

Simulations are performed on a Unix machine using the 'C' programming language. A sample run of the system is shown below. Assuming, for example, that the current-road-size is 78 in, *current-object-size* 45 in, and current-object-position is 4 in (from the center to the right). This input is converted to fuzzy sets:

$$\begin{aligned} \text{current-road-size} &= (MED/0.7, WIDE/0.3), \\ \text{current-object-size} &= (MED\text{-}SZ/0.8, LARGE\text{-}SZ/0.2), \\ \text{current-object-position} &= (CENTER/0.4, RIGHT/0.6). \end{aligned}$$

All combinations of the fuzzy values are matched against the 27 rules in the rule-base of this instructed behavior. A membership value is calculated for each of the matched rules:

r4: if x_1 is *WIDE AND* x_2 is *CENTER AND* x_3 is *MED-SZ*,
then y_1 is 20 AND y_2 is 40, *DOF* = 0.3

r5: if x_1 is *WIDE AND* x_2 is *CENTER AND* x_3 is *LARGE-SZ*,
then y_1 is 0 *AND* y_2 is 30, *DOF* = 0.2

r7: if x_1 is *WIDE AND* x_2 is *RIGHT AND* x_3 is *MED-SZ*,
then y_1 is 40 *AND* y_2 is 40, *DOF* = 0.3

r8: if x_1 is *WIDE AND* x_2 is *RIGHT*, x_3 is *LARGE-SZ*,
then y_1 is 20 *AND* y_2 is 30, *DOF* = 0.3

r22: if x_1 is *MED AND* x_2 is *CENTER AND* x_3 is *MED-SZ*,
then y_1 is 20 *AND* y_2 is 30, *DOF* = 0.4

r23: if x_1 is *MED AND* x_2 is CENTER AND x_3 is *LARGE-SZ*,
then y_1 is 0 *AND* y_2 is 30, *DOF* = 0.2

r25: if x_1 is *MED AND* x_2 is *RIGHT AND* x_3 is *MED-SZ*,
then y_1 is 40 *AND* y_2 is 30, *DOF* = 0.6

r26: if x_1 is *MED AND* x_2 is *RIGHT AND* x_3 is *LARGE-SZ*,
then y_1 is 20 *AND* y_2 is 30, *DOF* = 0.2.

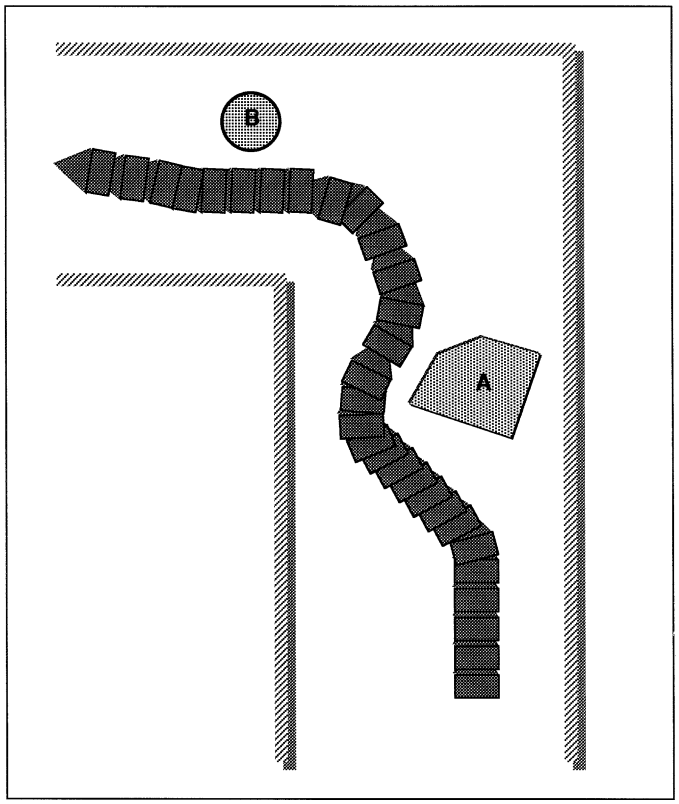

Figure 10. MITOS after being instructed to navigate a corner section with a second obstacle.

The final output is computed by taking the average of the results (*DOF*'s) of all the matched rules: speed = 13.0, steering-angle = 37.9. After the robot has been instructed to make a wider turn so that it may avoid obstacle B in Figure 4 its rule base has modified membership functions. *MITOS* proceeds by taking into account both things as shown in Figure 10, i.e., make a left turn and also avoid B.

5. Conclusions and Future Research

This paper presents a neurofuzzy methodology used for a robot to navigate in dynamic (time-varying) environments. The robot has to be equipped with only rudimentary sensing capabilities. One of the problems of handling a changing environment as opposed to a pre-specified environment is that it is computationally difficult for a robot to dynamically process the environment. Simple sensing requirements and fuzzy logic processing techniques reduce the processing time and make the approach presented very attractive; it may also be feasible to extrapolate the presented methodology to multiple mobile robotic systems. The robotic automata considered are devices whose main feature is incremental

learning from a human instructor. Fuzzy descriptions are used for the robot to acquire a repertoire of behaviors from the instructor which it may subsequently refine and recall using neural adaptive techniques. The skills acquired are similar to those possessed by an automobile driver. Future research will focus on neural-adaptive mechanisms and a comparative study of alternative *T*- and *S-norms* in the fuzzy neurons of the neuronal representation of the motion planner. In addition, it should be worthwhile to explore the possible amalgamation of the neurofuzzy motion planner with roadmaps, cell decomposition and potential field approaches.

References

1. Ahrikencheikh, C. and Seireg, A.: *Optimized-Motion Planning*, John Wiley and Sons, New York, 1994.
2. Bourbakis, N. G.: *Artificial Intelligence Methods and Applications*, World Scientific, Singapore, 1992.
3. Bourbakis, N. G.: Design of an autonomous navigation system, in: *IEEE Control Systems Magazine*, 1988, pp. 25–28.
4. Bourbakis, N. G.: Knowledge extraction and acquisition during real-time navigation in unknown environments, *Int. J. Pattern Recogn. Art. Intel.* **9**(1) (1995), 83–99.
5. Cameron, S. and Probert, P.: *Advanced Guided Vehicles, Aspects of the Oxford AGV Project*, World Scientific, Singapore, 1994.
6. Crangle, C. and Suppes, P.: *Language and Learning for Robots*, CLSI Publications, Stanford, CA, 1994.
7. Gat, E.: On the role of stored internal state in the control of autonomous mobile robots, *AI Magazine* (1993), 64–73.
8. Gupta, M. M. and Qi, J.: Theory of *T-norms* and fuzzy inference methods, *Fuzzy Sets and Systems* **40** (1991), 431–450.
9. Hwang, Y. K. and Ahuja, N.: A potential field approach to path planning, *IEEE Trans. Robotics Automat.* **8**(1) (1992), 23–32.
10. Hwang Y. K. and Ahuja, N.: Gross motion planning – a survey, *ACM Computing Surveys* **24**(3) (1992).
11. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Robotics Res.* **5**(1) (1986), 90–98.
12. Kwan, H. K. and Cai, Y.: Fuzzy neural network and its application to pattern recognition, *IEEE Trans. Fuzzy Systems* **2**(3) (1994), 185–193.
13. Latombe, J-C.: *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.
14. Mikio Maeda, Yasushi Maeda and Shuta Murakami: Fuzzy drive control of an autonomous mobile robot, *Fuzzy Sets and Systems* (1991), 195–204.
15. Rao, N. S. V.: Algorithmic framework for learned robot navigation in unknown terrain's, *IEEE Computer* (1989), 37–43.
16. Sheu, P. C-Y. and Xue, Q.: *Intelligent Robotic Planning Systems*, World Scientific, Singapore, 1993.
17. Tuscillo, A. and Bourbakis, N. G.: A neural and fuzzy control of a robotic hand, *IEEE Transactions on SMC* (1996).
18. Terano, T., Asai, K., and Sugeno, M.: *Fuzzy Systems Theory and its Applications*, Academic Press, Boston, 1992.
19. Tsoukalas, L. H. and Uhrig, R. E.: *Fuzzy and Neural Approaches in Engineering*, John Wiley and Sons, New York, 1997.
20. Tzafestas, S.: Introduction to intelligent robotic systems, in: S. Tzafestas (ed.), *Intelligent Robotic Systems*, Marcel Dekker, New York, 1991.
21. Zadeh, L. A.: A computational approach to fuzzy quantifiers in natural languages, *Comp. Math.* **9** (1983), 149–184.

22. Zadeh, L. A.: Fuzzy sets as a basis for theory of possibility, *Fuzzy Sets and Systems* **1** (1978), 3–28.
23. Zadeh, L. A.: Fuzzy sets, *Inform. and Control* **8** (1965), 338–353.