Applied Intelligence 16, 249–272, 2002

Soft Computing Based Pattern Classifiers for the Obstacle Avoidance Behavior of Intelligent Autonomous Vehicles (IAV)

OUAHIBA AZOUAOUI
CDTA—Centre de Développement des Technologies Avancées, Laboratoire de Robotique et d'Intelligence Artificielle, 128, Chemin Mohamed Gacem, BP 245 El-Madania, 16075, Algiers, Algeria
azouaoui@hotmail.com

AMINE CHOHRA
GMD—German National Research Center for Information Technology, Institute for Autonomous Intelligent Systems, Schloss Birlinghoven, D-53754 Sankt Augustin, Germany
chohra@gmd.de; chohra@hotmail.com

Abstract. To ensure more *autonomy* and *intelligence* with *real-time* processing capabilities for the obstacle avoidance behavior of Intelligent Autonomous Vehicles (IAV), the use of *soft computing* is necessary to bring this behavior near to that of humans in the recognition, learning, adaptation, generalization, reasoning and decision-making, and action. In this paper, pattern classifiers of spatial obstacle avoidance situations using Neural Networks (NN), Fuzzy Logic (FL), Genetic Algorithms (GA) and Adaptive Resonance Theory (ART) individually or in combination are suggested. These classifiers are based on supervised learning and adaptation paradigms as Gradient Back-Propagation (GBP), FL, GA and Simplified Fuzzy ArtMap (SFAM) resulting in NN/GBP and FL as Intelligent Systems (IS) and in NN/GA, NN/GA-GBP, NN-FL/GBP and NN-FL-ART/SFAM as Hybrid Intelligent Systems (HIS). Afterwards, a synthesis of the suggested pattern classifiers is presented where their results and performances are discussed as well as the Field Programmable Gate Array (FPGA) architectures, characterized by their high flexibility and compactness, for their implementation.

Keywords: intelligent autonomous vehicles (IAV), spatial obstacle avoidance situations, pattern classifiers, supervised learning and adaptation paradigms, neural networks (NN), fuzzy logic (FL), genetic algorithms (GA), adaptive resonance theory (ART), field programmable gate array (FPGA) architectures

1. Introduction

With increasing demands for high precision autonomous control over wide applications, conventional control approaches are unable to adequately deal with system complexity, nonlinearities, spatial and temporal parameter variations, and uncertainty. Intelligent control which is experiential based rather than model based is designed as a new emerging discipline to overcome these problems. This kind of discipline becomes more and more necessary for robot control in several development of real-time robotic applications.

In fact, today researchers have the requisite hardware, software and sensor technologies at their disposal for building intelligent dynamic systems particularly Intelligent Autonomous Vehicles (IAV). They have also in possession of computational tools which are far more effective in the design and development of intelligent systems than the predicate-logic-based methods of traditional Artificial Intelligence. These tools derive from a collection of methodologies known as *soft computing* which can deal with uncertain, imprecise and inexact data. Main components of *soft computing* are fuzzy-logic-based computing which contributes

the capability of approximate reasoning, neural computing which offers function approximation and learning capabilities and genetic algorithms which provide a methodology for systematic random search and optimization. These technologies have been experiencing extremely rapid growth in the spatial, underwater and terrestrial applications, where they have been shown to be very effective in solving real world problems [1–6]. In fact, the essence of *soft computing* is aimed at an accommodation with the imprecision of the real world. Thus, the guiding principle of *soft computing* is to exploit the tolerance for imprecision, uncertainty and partial truth in order to achieve tractability, robustness, low solution cost and better rapport with reality. These intelligent capabilities are required for robotic systems to adapt to dynamic environments and then to accomplish a wide variety of tasks under environmental constraints particularly the IAV obstacle avoidance behavior.

Indeed, the IAV endowed with such behavior have the ability to move and be self-sufficient in partially structured environments. They can carry out tasks in various environments by themselves like humans exploiting the recent developments in autonomy requirements, intelligent components, multi-robot systems, and massively parallel computers [1, 2, 7–14]. Thus, IAV designers search to create dynamic systems able to navigate and achieve intelligent behaviors like human in real environments. This is became a reality with the emergence of *soft computing* allowing the emulation of human performance and modeling of uncertain or ambiguous situations that are so often encountered in real life. To reach then their targets without collisions with possibly encountered obstacles, IAV must particularly have the capability to achieve the obstacle avoidance behavior with *autonomy*, *intelligence* and *real-time*. To acquire this behavior while answering these requirements, IAV must be endowed with recognition, learning, adaptation, reasoning and decision-making, and action capabilities. To achieve this goal, classical approaches have been replaced by current approaches based on *soft computing* implying particularly Neural Networks (NN), Fuzzy Logic (FL), Genetic Algorithms (GA) and Adaptive Resonance Theory (ART) individually as Intelligent Systems (IS) [4, 15–27] or in combination as Hybrid Intelligent Systems (HIS) [28–33]. Indeed, such systems are recognized to improve the learning, adaptation and generalization capabilities related to variations in environments where information is qualitative, imprecise, or incomplete [3, 34–42].

To increase the intelligence and autonomy of IAV navigation, several obstacle avoidance approaches based on *soft computing* are then developed. Indeed, for the most part of these approaches, the uncertain and imprecise knowledge of the environment is handled suitably by the FL theory increasing their performance [21–23, 26]. For instance, in [26] fuzzy obstacle avoidance control of a robot based on finding permissible passageways using the edges between the floor and the wall or obstacles is suggested. Other works based on how a human can instantly judge his rough condition of perceiving the danger degree of static and dynamic obstacles [21] and based on recognition of road shapes [22] are developed. They present a fuzzy obstacle avoidance approach which applied fuzzy reasoning and production rules [21] and a fuzzy expert system based on the human thought process using the idea of fuzziness [22].

Another current approach for the obstacle avoidance behavior is based upon adaptive capabilities of NN, exploiting learning and generalization to avoid collisions [16, 17, 19, 20, 24, 25]. Indeed, the implementation of a neural approach to the problem of planning a near-optimal collision-free path for a mobile robot is given in [20]. In this approach, the path planning problem is transformed to a pattern classification problem in which class labels are uniquely mapped onto the set of maneuverable actions of a robot while each time instance of a scenario is mapped onto a 2-dimensional binary or graded pattern. In another work, NN is combined with conventional techniques and used to control target localization and obstacle avoidance behaviors in an environment with dynamically changing obstacles [19] while Meng [24] uses only NN to develop the NEURO-NAV system for indoor vehicles. These NN are trained to interpret visual information and perform navigational behaviors such as hallway following and landmark detection.

A new promising tendency is the use of GA in solving the obstacle avoidance problem by exploiting their optimization capability often to find an optimal path for IAV [4, 15, 27]. Indeed, an intelligent path planning for mobile robots operating in cluttered environments using GA is developed in [15]. In this approach GA plan and optimize paths without placing restrictions on the shapes of the obstacles carrying out multiple constraints to find a collision free solution. In Xiao [27], the planner/navigator system combines the concept of evolutionary computation with problem-specific chromosome structures and operators. In fact,

this system combines off-line planning and real-time navigation in the same evolutionary algorithm using the same chromosome structure where the adaptability of the system is underlined particularly with respect to the operator probabilities and real-time navigation process.

One of the more recent trends in the intelligent obstacle avoidance research is to use a combination of such technologies i.e., a combination of NN, FL, GA, and ART to increase the autonomy and intelligence of IAV [28–33]. Recently many attempts have been made to combine FL and NN in order to achieve better performance in reasoning and decision-making processes. The systems that result from such a fusion called fuzzy neural networks (NN-FL) possess combined features. An example of such systems is the obstacle avoidance approach developed in [30] which uses NN-FL for the fuzzy reasoning and inference to decide static and dynamic obstacle danger degrees. One approach based this once on artmap neural networks (NN-ART) control system is developed in [28] which provides the mobile robot MAVIN with the capability to be adaptive to its environment and to learn from experience. This system includes NN-ART for early visual perception, eye motion control, pattern learning, emotional states and behavioral actions. Another approach, using the fuzzy artmap neural networks (NN-FL-ART), presented in [33] is applied to the motion planning controller for the mobile robot Nomad 200. This controller has a modified structure for vision input data acquisition and processing. In [31], an analogue approach is used for a reactive mobile robot navigation in an unknown, cluttered environment. The aim of this system is to provide a steering angle signal letting a robot reach a goal while avoiding collisions with obstacles. This system is based on a NN-FL-ART classifier performing a perceptual space partitioning and the neural associative memory storing system's experience and superposing influences of different behaviors.

Globally, these obstacle avoidance approaches are essentially based on the manner with which humans reason, establish their own decisions, and take the appropriate action to achieve the desired task. All these mimicked human aptitudes using *soft computing* technologies, based on learning and adaptation paradigms, make the obstacle avoidance approaches more robust and reliable than the classical ones.

This paper deals with the *soft computing* based *pattern classification* for the obstacle avoidance behavior of IAV in partially structured environments. The aim of this paper is to suggest different IS and HIS for the pattern classification of spatial obstacle avoidance situations and make a synthesis of these pattern classifiers. These classifiers are based on a single technology in IS or as it is more frequently used in combination in HIS. Indeed, in the main, FL, NN, GA and ART are combined in a complementary and synergetic fashion rather than in a competitive one [42] to handle the obstacle avoidance problem involving imprecise and noisy data from the sensors and environment. These technologies are used in the suggested classifiers under supervised learning and adaptation paradigms so that the required *autonomy*, *real-time* and *intelligence* capabilities for IAV are satisfied.

In this paper, the obstacle avoidance behavior problem of IAV in partially structured environments is stated in Section 2. This behavior is acquired using *soft computing* based pattern classifiers under supervised learning and adaptation paradigms such as Gradient Back-Propagation (GBP), FL, GA and Simplified Fuzzy ArtMap (SFAM). In Section 3, the suggested classifiers given as IS (NN/GBP and FL) or as HIS (NN/GA, NN/GA-GBP, NN-FL/GBP and NN-FL-ART/SFAM) are then developed. Finally, Section 4 presents a synthesis of these classifiers where their results and performances are discussed as well as the Field Programmable Gate Array (FPGA) architectures for their implementation.

2. Obstacle Avoidance Behavior of Intelligent Autonomous Vehicles (IAV)

Recent research on IAV has pointed out a promising direction for future research in mobile robotics where *real-time*, *autonomy* and *intelligence* have received considerably more attention than, for instance, optimality and completeness. Many navigation approaches have dropped the assumption that perfect environment knowledge is available. The representation of the environment knowledge is, in fact, based on acquisition of intelligent behaviors that enable the vehicle to interact effectively with its environment [43]. Consequently, IAV are facing with less predictable and more complex environments, they have to orient themselves, explore their environments autonomously, recover from failures, and perform whole families of tasks in real-time. More, if vehicles lack initial knowledge about themselves and their environments, *learning* and *adaptation* become then inevitable to replace missing or incorrect environment knowledge by experimentation,

observation, and generalization. Thus, in order to reach a goal, learning and adaptation of vehicles rely on the interaction with their environment to extract information [44].

Currently, most obstacle avoidance approaches are inspired from observations of human navigation behavior. Indeed, human navigators do not need to calculate the exact coordinates of their positions while navigating in environments (roads, hallways, etc.). The road-following or the hallway-following behavior exhibited by humans is a reactive behavior that is learned through experience. Given a goal, human navigators can focus attention on particular stimuli in their visual input and extract meaningful information very quickly. Extra information may be extracted from the scene during reactive behavior; this information (e.g., approaching an intersection) will usually be stored away and may be retrieved subsequently for higher level reasoning.

In general, these observations have dictated two kinds of obstacle avoidance approaches based on learning and adaptation with regard to the type of obstacles (static or dynamic). In this paper, only the partially structured environments such as factories, passenger stations, harbors and airports with static obstacles are considered. In fact, human perceives the spatial situations in such environments as topological situations: rooms, corridors, right turns, left turn, junctions, etc. Consequently, trying to capture the human obstacle avoidance behavior in such environments, several approaches based on a recognition of topological situations have been developed [17, 22–26, 45, 46].

Thus, IAV should have the capability of recognizing spatial obstacle avoidance situations of partially structured environments and maneuvering through these situations on the basis of their own judgement to enable themselves to navigate from one point of space to a destination without collision with static obstacles. Such obstacle avoidance behavior is acquired using *soft computing* based pattern classifiers under *supervised* learning and adaptation paradigms which allow to recognize topological situations from sensor data giving vehicle-obstacle distances.

2.1. Vehicles and Sensors

2.1.1. Vehicles. For the sake of simplicity, the vehicle movements are possible in three (03) directions and consequently three (03) actions A_i $(i = 1, \ldots, 3)$ are defined as action to turn to the Right, action to move Ahead and action to turn to the Left, as shown in Fig. 1. They are expressed by the action vector $\mathbf{A} = [A_R, A_A, A_L]$.

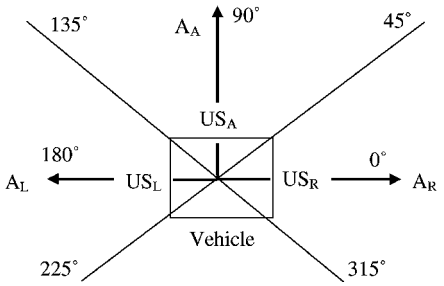

Figure 1. Vehicle and sensors.

2.1.2. Sensors. To detect possibly encountered obstacles, the perception system is assumed to have three (03) UltraSonic sensors (US) necessary to get distances (vehicle-obstacle) covering the three areas: US_R to get distance d_R on the Right, US_A to get distance d_A in Ahead, and US_L to get distance d_L on the Left, as shown in Fig. 1.

2.2. Partially Structured Environments

The uncertainty of the real world knowledge is the main problem of IAV. This uncertainty is due to the fact that the environment representation is based essentially on the vehicle perception systems which must be taken into account in the data representation. For the suggested classifiers, the perception system uses US to detect static obstacles giving the vehicle-obstacle distances necessary for the obstacle avoidance behavior. Indeed, the problem of correctly evaluating noisy and incorrect data for the interpretation of US signals is often an encountered one [17]. This problem is taken into account, for these classifiers, by the use of parallel basis structures of NN, FL, NN-FL or NN-FL-ART with their inherent characteristics of adaptivity and high fault tolerance with respect to defective sensors or noisy sensor data [34, 38, 40, 41].

2.2.1. Possibly Encountered Static Obstacles. In reality, static obstacles in partially structured environments are of different shapes representing walls, pillars, machines, desks, tables, chairs, etc. as shown in Fig. 2(a).

2.2.2. Spatial Situations Structured in Topological Situations. The possible movements of vehicles lead us to structure the spatial situations in six (06)

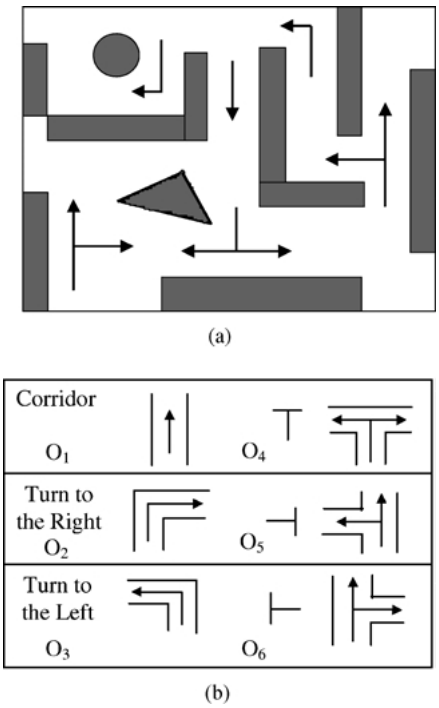

Figure 2. (a) A partially structured environment. (b) Obstacle avoidance situations.

topological situations called obstacle avoidance situations as shown in Fig. 2(b). These situations are defined with six (06) classes $O_1, \ldots, O_j, \ldots, O_6$.

2.3. Training Environment

The training environment must contain all obstacle avoidance situations O_j as shown in Fig. 3. This environment allows vehicle to be in different positions i.e., in different obstacle avoidance situations. Thus, training examples are defined by randomly selecting fifty four (54) vehicle positions (patterns), where each particular position corresponds to one training example for a particular obstacle avoidance situation. This environment has been used to train the suggested NN/GBP, NN/GA, NN/GA-GBP, NN-FL/GBP and NN-FL-ART/SFAM classifiers.

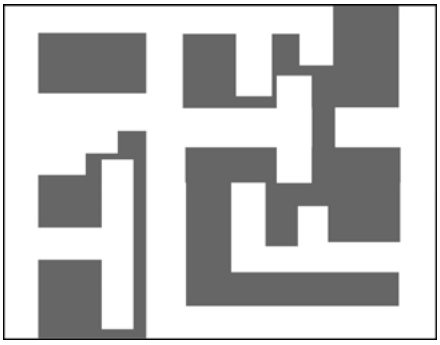

Figure 3. Training environment.

3. Soft Computing Based Pattern Classifiers Under Supervised Learning and Adaptation Paradigms

In this Section, the obstacle avoidance problem is solved using the suggested pattern classifiers in IS (NN/GBP and FL) and HIS (NN/GA, NN/GA-GBP, NN-FL/GBP and NN-FL-ART/SFAM) by recognizing numerous topological situations in order to avoid static obstacles in partially structured environments. Indeed, IS and HIS have been recently recognized to improve the learning and adaptation capabilities related to variations in environments where information is qualitative, inaccurate, uncertain or incomplete [39]. Particularly, the use of HIS combining NN, FL, GA and ART [38–41] is necessary to provide IAV with more *autonomy* and *intelligence*.

During the navigation, each IAV must build an implicit internal map (i.e., obstacles and free spaces) from sensor data, update it and use it for intelligently controlling their obstacle avoidance behavior. This behavior is acquired by learning and adaptation of the suggested pattern classifiers from ultrasonic sensor data of partially structured environments. For all these classifiers, the input vector is defined by $\mathbf{X} = [X_1, \ldots, X_i, \ldots, X_3]$ while the output vector is defined by $\mathbf{O} = [O_1, \ldots, O_j, \ldots, O_6]$. Indeed, in each step, the vehicle-obstacle distances d_R, d_A and d_L are updated from US_R, US_A and US_L, respectively and used, differently according to the given classifier, to define the input vector $\mathbf{X}$. Thus, for each input vector $\mathbf{X}$, each classifier must provide the vehicle with the capability to recognize in which situation O_j it finds itself, see Fig. 2(b), to avoid possibly encountered static obstacles.

3.1. Intelligent Systems (IS) Based Obstacle Avoidance Behavior

3.1.1. NN/GBP Classifier. NN have fared better than well-established conventional options when the characteristics of the system are poorly known. Fast parallel computation and use of generic function forms for

complex nonlinear mappings have motivated the proposal of a growing number of IAV control involving NN [47]. In fact, the interest in NN stems from the wish to understand principles leading in some manner to the comprehension of human brain functions and to build machines able to perform complex tasks requiring massively parallel computations [34, 35, 37, 38, 40, 41]. Essentially, NN deal with cognitive tasks such as learning, adaptation, generalization and optimization. The application of NN to solve problems of complex dynamic systems such as IAV has the potential to enhance the *safety*, *reliability*, and *operability* of these systems. Indeed, learning systems are attractive for solving highly nonlinear, uncertain, incomplete or non-stationary problems. NN pattern classifiers exploit NN features such as implicit knowledge representation, learning and generalization from experience (from examples), robustness related to the fault tolerance with respect to defective sensors or noisy sensor data, and massively parallel processing (real-time).

The NN/GBP classifier is a multilayer feedforward NN built of three (03) layers as shown in Fig. 4.

The distances d_R, d_A and d_L are pre-processed to constitute the input vector $\mathbf{X}$ of this classifier:

$$\begin{aligned} X_1 &= (1/\rho)\exp(-d_R/a), \\ X_2 &= (1/\rho)\exp(-d_A/a), \\ X_3 &= (1/\rho)\exp(-d_L/a), \end{aligned} \tag{1}$$

where ρ: norm of the input vector $\mathbf{X}$ and a: input pre-processing factor with $a > 1$.

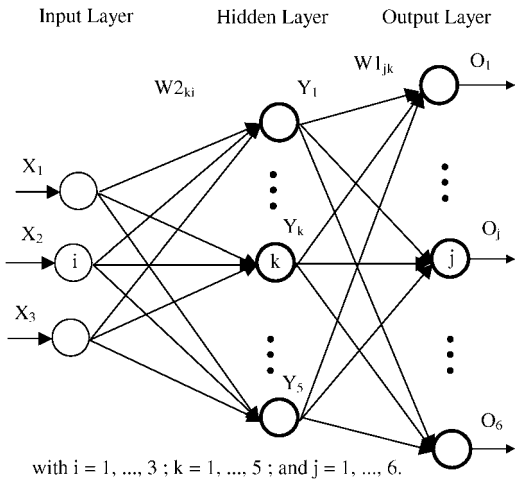

Figure 4. NN/GBP, NN/GA, NN/GA-GBP classifiers.

Input Layer: with three (03) input nodes receiving the components of the input vector $\mathbf{X}$. This layer transmits these inputs to all nodes of the next layer.

Hidden Layer: with five (05) hidden nodes, where their outputs are obtained using the output sigmoïd function f as follows:

$$\mathrm{net}_k = \sum_i X_i \mathrm{W2}_{ki}, \tag{2}$$

$$\mathrm{Y}_k = f(\mathrm{net}_k), \tag{3}$$

where

$$f(x) = \frac{1}{1+\exp(-x)}. \tag{4}$$

Output Layer: with j sigmoïdal output nodes which are obtained by:

$$\mathrm{net}_j = \sum_k Y_k \mathrm{W1}_{jk}, \tag{5}$$

$$O_j = f(\mathrm{net}_j). \tag{6}$$

To acquire the obstacle avoidance behavior, this classifier is trained, in the training environment shown in Fig. 3, using the *supervised* GBP paradigm detailed in [35] where the error is computed as follows:

$$e_{ex} = (1/2)\sum_j (\mathrm{Desired}\, O_j - O_j)^2, \tag{7}$$

where the index ex runs over all examples of the training set.

The weights are updated until the network convergence: a state permitting the coding, i.e., the classification of all the training examples or input space. This state is reached when the error e_{ex} is very close to the tolerance i.e., the error for all training examples is reduced to an acceptable value. Practically, the tolerance is determined to end training while preventing inappropriate memorization, also called over-training.

The NN/GBP classifier is trained from fifty four (54) examples of the training set with the input pre-processing factor $a = 3$. The weights $\mathrm{W1}_{jk}$ and $\mathrm{W2}_{ki}$ are adjusted from a random weight initialization between $[-1, +1]$ with the learning rate $\eta = 0.1$. This classifier yields convergence to the tolerance $\mathrm{E_T} = 0.01$ in well under the cycle number $\mathrm{CN(GBP)} = 927$ cycles.

3.1.2. FL Classifier. Acknowledging the environment structure and interacting with it involves abstract appreciation of natural concepts related to the

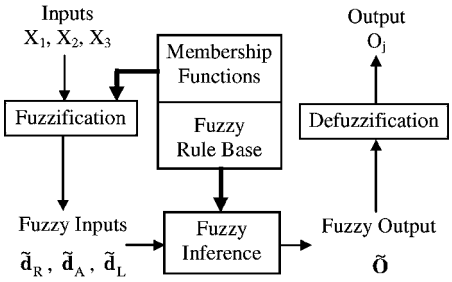

Figure 5. FL classifier.

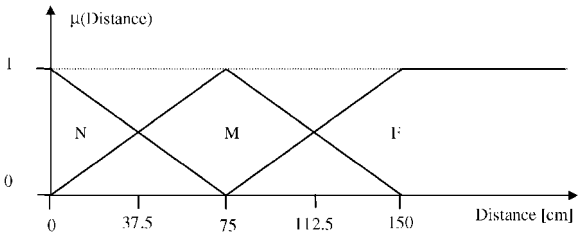

Figure 6. Distance membership functions.

proximity, danger degree, etc. The implied natural language must be represented through fuzzy sets of the FL which involve classes with gradually varying transition boundaries. Basically, FL provides an effective means to capture the approximate and inexact nature of the real world [38–41, 48, 49]. Also, FL can be viewed as an attempt to bring together conventional precise mathematics and human-like decision-making concepts [50]. FL pattern classifiers exploit FL features such as explicit knowledge representation, adaptation and generalization from fuzzy rules, the capability to capture the approximate and inaccurate nature of the real environment, and parallel processing (fuzzy associative memories).

The FL classifier is a fuzzy system as shown in Fig. 5 where $\tilde{\mathbf{d}}_{\mathrm{R}}$, $\tilde{\mathbf{d}}_{\mathrm{A}}$, $\tilde{\mathbf{d}}_{\mathrm{L}}$ and $\tilde{\mathbf{O}}$ represent the fuzzy d_{R}, fuzzy d_{A}, fuzzy d_{L} and fuzzy $\mathbf{O}$, respectively.

The distances d_{R}, d_{A} and d_{L} constitute the input vector $\mathbf{X}$ of this classifier. The input vector components (distances) have the same degree of qualitative values: Near (N), Medium (M) and Far (F) defined by membership functions shown in Fig. 6.

3.1.2.1. Fuzzification. The fuzzification process calculates the degree of the measured data belonging to the membership functions of d_{R}, d_{A} and d_{L}; e.g., for d_{R}, the membership degrees of fuzzy sets N, M and F are $\{\mu_{\mathrm{N}}(X_1), \mu_{\mathrm{M}}(X_1), \mu_{\mathrm{F}}(X_1)\}$, respectively.

3.1.2.2. Fuzzy Rule Base 1. This classifier uses a fuzzy rule base to mimic the input/output mapping of a human expert in recognition of obstacle avoidance situations. Indeed, this expert has formulated his knowledge in a linguistic form by establishing a set of fuzzy rules given in Table 1. Note that the particular cases (X_1 is N and X_2 is N and X_3 is N) and (X_1 is F and X_2 is F and X_3 is F) must be handled by the navigation system at a higher level.

In this fuzzy rule base the fuzzy obstacle avoidance situation vector $\tilde{\mathbf{O}}$ is expressed by:

$$\tilde{\mathbf{O}} = \big[\mu_{(X_{1m},X_{2m},X_{3m})}(O_1), \ldots, \mu_{(X_{1m},X_{2m},X_{3m})} \times (O_j), \ldots, \mu_{(X_{1m},X_{2m},X_{3m})}(O_6)\big], \tag{8}$$

where $\mu_{(X_{1m},X_{2m},X_{3m})}(O_j)$ represents the membership function degree of O_j with $m = 1$ or 2, see the simplification given below.

3.1.2.3. Fuzzy Inference. The fuzzy inference is achieved by the Min and Max operations as detailed in [50, 51]. The particularity of the distance parameter is that for each given distance only two (02) membership function degrees are different from zero, and consequently those equal to zero must be not considered [52]. From this simplification, for each specific IAV situation, the values of the distances are mapped to the discrete intervals to form the fuzzy sets $\tilde{\mathbf{d}}_{\mathrm{R}}$, $\tilde{\mathbf{d}}_{\mathrm{A}}$ and $\tilde{\mathbf{d}}_{\mathrm{L}}$ expressed by:

$$\begin{aligned} \tilde{\mathbf{d}}_{\mathrm{R}} &= \{\mu_1(X_1), \mu_2(X_1)\}, \\ \tilde{\mathbf{d}}_{\mathrm{A}} &= \{\mu_1(X_2), \mu_2(X_2)\}, \\ \tilde{\mathbf{d}}_{\mathrm{L}} &= \{\mu_1(X_3), \mu_2(X_3)\}, \end{aligned} \tag{9}$$

where $\mu_m(X_i)$, with $m = 1$ or 2, are the membership function degrees of the distances. With this description, one can have eight (08) possible conditions corresponding to eight (08) fuzzy rules. Then, the level of certainty of each condition triplet μ_1, or $\mu_2, \ldots,$ or μ_8 can be found using the Min operation:

$$\begin{aligned} \mu_{\mathrm{cond}}(X_{11}, X_{21}, X_{31}) &= \mathrm{Min}(\mu_1(X_1), \mu_1(X_2), \mu_1(X_3)) = \mu_1, \\ &\ldots \\ \mu_{\mathrm{cond}}(X_{12}, X_{22}, X_{32}) &= \mathrm{Min}(\mu_2(X_1), \mu_2(X_2), \mu_2(X_3)) = \mu_8, \end{aligned} \tag{10}$$

Table 1. Fuzzy rule base 1.

If (X_1 is N and X_2 is N and X_3 is M) Then $\tilde{\mathbf{O}} = [0, 0, 0.6, 0, 0, 0]$
If (X_1 is N and X_2 is N and X_3 is F) Then $\tilde{\mathbf{O}} = [0, 0, 1, 0, 0, 0]$
If (X_1 is N and X_2 is M and X_3 is N) Then $\tilde{\mathbf{O}} = [1, 0, 0, 0, 0, 0]$
If (X_1 is N and X_2 is M and X_3 is M) Then $\tilde{\mathbf{O}} = [0.2, 0, 0.2, 0, 0.9, 0]$
If (X_1 is N and X_2 is M and X_3 is F) Then $\tilde{\mathbf{O}} = [0.2, 0, 0, 0, 0.9, 0]$
If (X_1 is N and X_2 is F and X_3 is N) Then $\tilde{\mathbf{O}} = [1, 0, 0, 0, 0, 0]$
If (X_1 is N and X_2 is F and X_3 is M) Then $\tilde{\mathbf{O}} = [0.2, 0, 0, 0, 0.9, 0]$
If (X_1 is N and X_2 is F and X_3 is F) Then $\tilde{\mathbf{O}} = [0, 0, 0, 0, 1, 0]$
If (X_1 is M and X_2 is N and X_3 is N) Then $\tilde{\mathbf{O}} = [0, 0.6, 0, 0, 0, 0]$
If (X_1 is M and X_2 is N and X_3 is M) Then $\tilde{\mathbf{O}} = [0, 0.2, 0.2, 0.9, 0, 0]$
If (X_1 is M and X_2 is N and X_3 is F) Then $\tilde{\mathbf{O}} = [0, 0, 0.9, 0.2, 0, 0]$
If (X_1 is M and X_2 is M and X_3 is N) Then $\tilde{\mathbf{O}} = [0.2, 0.2, 0, 0, 0, 0.9]$
If (X_1 is M and X_2 is M and X_3 is M) Then $\tilde{\mathbf{O}} = [0.2, 0.2, 0.2, 0.9, 0.9, 0.9]$
If (X_1 is M and X_2 is M and X_3 is F) Then $\tilde{\mathbf{O}} = [0, 0, 0.2, 0.9, 0.9, 0]$
If (X_1 is M and X_2 is F and X_3 is N) Then $\tilde{\mathbf{O}} = [0.2, 0, 0, 0, 0, 0.9]$
If (X_1 is M and X_2 is F and X_3 is M) Then $\tilde{\mathbf{O}} = [0.2, 0, 0, 0, 0.9, 0.9]$
If (X_1 is M and X_2 is F and X_3 is F) Then $\tilde{\mathbf{O}} = [0, 0, 0, 0, 0.6, 0]$
If (X_1 is F and X_2 is N and X_3 is N) Then $\tilde{\mathbf{O}} = [0, 1, 0, 0, 0, 0]$
If (X_1 is F and X_2 is N and X_3 is M) Then $\tilde{\mathbf{O}} = [0, 0.2, 0, 0.9, 0, 0]$
If (X_1 is F and X_2 is N and X_3 is F) Then $\tilde{\mathbf{O}} = [0, 0, 0, 1, 0, 0]$
If (X_1 is F and X_2 is M and X_3 is N) Then $\tilde{\mathbf{O}} = [0.2, 0, 0, 0, 0, 0.9]$
If (X_1 is F and X_2 is M and X_3 is M) Then $\tilde{\mathbf{O}} = [0, 0.2, 0, 0.9, 0, 0.9]$
If (X_1 is F and X_2 is M and X_3 is F) Then $\tilde{\mathbf{O}} = [0, 0, 0, 0.6, 0, 0]$
If (X_1 is F and X_2 is F and X_3 is N) Then $\tilde{\mathbf{O}} = [0, 0, 0, 0, 0, 1]$
If (X_1 is F and X_2 is F and X_3 is M) Then $\tilde{\mathbf{O}} = [0, 0, 0, 0, 0, 0.6]$
with $X_1 = d_{\mathrm{R}}$, $X_2 = d_{\mathrm{A}}$ and $X_3 = d_{\mathrm{L}}$.

where cond represents the fuzzy set of conditions which is then written as follows:

$$\mathrm{cond} = \{\mu_1, \ldots, \mu_8\}. \tag{11}$$

Each possible ordered triplet of condition is associated with an obstacle avoidance situation O_j. Then, the certainty of each situation is obtained by the Max and Min operations as follows:

$$\begin{aligned} \mu_{O1} &= \mathrm{Max}\big\{\mathrm{Min}\big(\mu_1, \mu_{(X_{11},X_{21},X_{31})}(O_1)\big), \\ &\quad \ldots, \mathrm{Min}\big(\mu_8, \mu_{(X_{12},X_{22},X_{32})}(O_1)\big)\big\}, \\ &\quad \ldots \\ \mu_{O6} &= \mathrm{Max}\big\{\mathrm{Min}\big(\mu_1, \mu_{(X_{11},X_{21},X_{31})}(O_6)\big), \\ &\quad \ldots, \mathrm{Min}\big(\mu_8, \mu_{(X_{12},X_{22},X_{32})}(O_6)\big)\big\}, \end{aligned} \tag{12}$$

The collection of situations forms the final fuzzy obstacle avoidance situation vector $\tilde{\mathbf{O}}$:

$$\tilde{\mathbf{O}} = \{\mu_{O1}, \ldots, \mu_{Oj}, \ldots, \mu_{O6}\} \tag{13}$$

3.1.2.4. Defuzzification. The Max operation is used for the defuzzification process to give the final obstacle avoidance situation O_j:

$$O_j = \mathrm{Max}\{\mu_{O1}, \ldots, \mu_{Oj}, \ldots, \mu_{O6}\} \tag{14}$$

The FL classifier is based on a "*supervised* adaptation paradigm" traduced by the use of fuzzy obstacle avoidance rules given by a human expert (supervisor). This classifier is used to capture the behavior of a human expert while controlling the obstacle avoidance behavior i.e., recognizing the obstacle avoidance situations O_j.

3.2. *Hybrid Intelligent Systems (HIS) Based Obstacle Avoidance Behavior*

3.2.1. NN/GA Classifier. GA are one of the optimization methods based on the biological evolution process [36, 40, 41, 53]. They are search algorithms that are based on the mechanics of natural selection and natural genetics. They perform a global, random, parallel search for an optimal solution using simple computations. Starting with an initial population of genetic structures, genetic inheritance operations based on selection, crossover, and mutation are performed to generate "offspring" that compete for survival ("survival of the fittest") to make up the next generation of population structures [40]. One of the GA's characteristics is the multiple points' search which discriminates the GA from the other random search methods [54]. That is, each chromosome corresponds to a candidate of the optimal result in the search space and many such results exist in the GA's search. The GA typically starts by randomly generating initial population of chromosomes. Each chromosome is transformed into the fitness value to obtain a quantitative measure. On the basis of the fitness value, the chromosomes undergo genetic operations to find a set of parameters that search an optimal solution to the problem or to reach the limited generations [54]. Indeed, these operations are used as means of preserving beneficial information in the NN weights in each chromosome (each chromosome represents the weight values of the NN) with the overall aim of arriving at classifying the topological situations of the obstacle avoidance behavior (i.e., better solutions to the problem).

The suggested approach consists to apply the GA to train a multilayer feedforward NN in which the weights are seen to form a parameter space, in order to obtain a best weight configuration [55, 56]. The NN/GA classifier is a multilayer feedforward NN as shown in Fig. 4. The distances d_R, d_A and d_L are pre-processed as in Eq. (1) to constitute the input vector $\mathbf{X}$ of this classifier. This classifier is trained, in the training environment shown in Fig. 3, to acquire the obstacle avoidance behavior based on a *supervised* GA learning paradigm as shown in Fig. 7.

The population size which is a self-explanatory parameter is set to fifty (50) chromosomes as suggested in [55, 56] and the genetic structures represent weight values of the NN connections, see Fig. 4. Each chromosome (i.e., the weights in the NN) is encoded as a list of real numbers with the

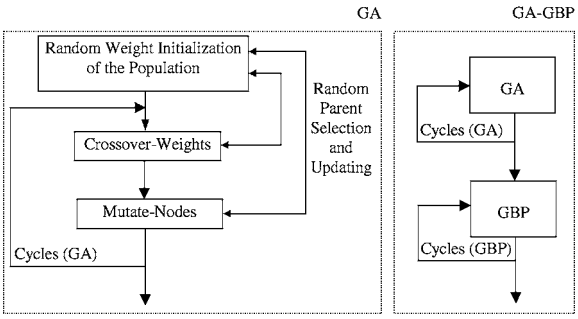

Figure 7. Synopsis of supervised GA and GA-GBP learning paradigms.

error:

$$E_{\text{pop}} = e_1 + \cdots + e_{\text{ex}} + \cdots + e_{54}, \qquad (15)$$

with the index pop which runs over the chromosome population: $\text{pop} = 1, \ldots, 50$ and e_{ex} as defined in Eq. (7).

3.2.1.1. Crossover. The used operation is the Crossover-Weights [56]. This operation puts a value into each position of the child chromosome by randomly selecting one of the two parents and using the value in the same position on that parent chromosome:

- from two (02) selected parent chromosomes $\text{chromosome}_{\text{PARENT1}}$ and $\text{chromosome}_{\text{PARENT2}}$, this operation creates two (02) children $\text{chromosome}_{\text{CHILD1}}$ and $\text{chromosome}_{\text{CHILD2}}$. The crossover positions of both positions pos_1 and pos_2 are determined randomly among $[1, \ldots, 30]$ and among $[31, \ldots, 45]$, respectively.
- then, the two (02) chromosomes, having the smaller fitness function (higher errors), of the population must be updated by the two (02) chromosomes among $\text{chromosome}_{\text{PARENT1}}$, $\text{chromosome}_{\text{PARENT2}}$, $\text{chromosome}_{\text{CHILD1}}$, and $\text{chromosome}_{\text{CHILD2}}$ which have the higher fitness function (smaller errors).

3.2.1.2. Mutation. The used operation is the Mutate-Nodes as detailed in [56]. This operation selects nbr non-input nodes of the network represented by the parent chromosome. For each of the ingoing links to these nbr nodes, the operation adds a random value v_{mut} to the link's weights from the initialization probability distribution. It then encodes this new network on the

child chromosome. In our experiments, nbr' is equal to 2:

- from one (01) selected parent chromosome $\text{chromosome}_{\text{PARENT}}$ this operation creates one (01) child $\text{chromosome}_{\text{CHILD}}$. The mutation positions of both positions nbr_1 and nbr_2 are determined randomly among the non-input nodes i.e., either nbr_1 and nbr_2 among $[Y_1, \ldots, Y_5]$ or nbr_1 and nbr_2 among $[O_1, \ldots, O_6]$ with $\text{nbr}_1 \neq \text{nbr}_2$.
- then, the parent chromosome of the population must be updated by the child one if the E_{CHILD} is smaller than E_{PARENT}.

3.2.1.3. Fitness Function. The Fitness function, which evaluates each chromosome whether it is desirable or not, is defined by:

$$\text{Fitness} = 1/E_{\text{pop}}. \tag{16}$$

3.2.1.4. Selection. Based on the fitness value, each chromosome is selected or not to the next generation. Elite preservation strategy is adopted where the chromosomes having high fitness value remain to the next generation and sampling strategy that the chromosomes are selected randomly remain to the next generation. The more a chromosome that undergoes the genetic operation has high fitness value, the more it will survive in the next generation by selection. This selection which changes the population of chromosomes performs the *evolution* [54].

The NN/GA classifier is trained from fifty four (54) examples of the training set with the input pre-processing factor $a = 3$ and initial population $\text{pop} = 50$ chromosomes. The weights W1_{jk} and W2_{ki} are adjusted from a random weight initialization between $[-5, +5]$ with the random mutation value $v_{\text{mut}} \in [-0.25, +0.25]$. This classifier does not reach the tolerance $E_{\text{T}} = 0.01$, after the cycle number CN(GA) 10,000 cycles the error is equal to 0.244301.

3.2.2. NN/GA-GBP Classifier. Since GBP is a gradient descent approach, it has tendency to get stuck in local minima of the error surface and thus not find the global minimum. Moreover, GBP is sensitive to parameters such as learning rate and momentum [57] while the performance of GA does not depend on these parameters. The main advantage of using GA for the training of feedforward NN is that they can find global minima without getting stuck at local minima. However, the GA's problem is that they are inherently slow [36, 41, 53, 57]. In fact, when the search space is large, as is usually the case in training a multilayer feedforward NN, the GA approach takes a long time to converge. The length of search is due to the optimal generalization of the training process with no a priori knowledge about the parameter space [57]. Another problem of GA is their weakness in performing fine-tuned local search which is widely recognized. Although GA exhibit very fast convergence to a point of approximate solution in a search space, GA themselves do not entail a mechanism for local fine-tuning as seen in GBP.

The NN/GA-GBP classifier is suggested to remedy insufficiencies of slowness of GA and getting stuck in local minima of GBP. In this classifier, NN are trained in two (02) stages. First, GA train weights of nodes to find a location in the weight space which is closed to the optimal solution. Second, GBP starts from that point taking this weight configuration as initial weights and conducts an efficient local search. Thus, the NN/GA-GBP classifier is a multilayer feedforward NN as shown in Fig. 4. The distances d_{R}, d_{A} and d_{L} are pre-processed as in Eq. (1) to constitute the input vector $\mathbf{X}$. This classifier is trained, in the training environment shown in Fig. 3, to acquire the obstacle avoidance behavior using *supervised* GA and GBP learning paradigms, see Fig. 7.

The NN/GA-GBP classifier is trained by GA from fifty four (54) examples of the training set with the input pre-processing factor $a = 3$ and initial population $\text{pop} = 50$ chromosomes. The weights W1_{jk} and W2_{ki} are adjusted from a random weight initialization between $[-5, +5]$ with the random mutation value $v_{\text{mut}} \in [-0.25, +0.25]$ until the cycle number $\text{CN(GA)} = 70$ cycles. From the resulting weights this classifier is trained again by GBP with the learning rate $\eta = 0.3$ until the convergence to the tolerance $E_{\text{T}} = 0.01$, after the cycle number $\text{CN(GBP)} = 290$ cycles. Thus, this classifier yields convergence to the tolerance $E_{\text{T}} = 0.01$ in well under the total cycle number $\text{CN(GA-GBP)} = 70 + 290 = 360$ cycles.

3.2.3. NN-FL/GBP Classifier. The interest to establish relations between the FL and NN could be in part related to notions such as: cognition and generalization. More, L.A. Zadeh and P.J. Werbos [42] asserted that these relations are basically *complementary* rather than equivalent or competitive. This fusion technology is one of the rare concepts which uses *at once* an *explicit* and *implicit* knowledge. Recently, the tendency in the direction of using FL in the design of NN leading to fuzzy neural networks (NN-FL) is becoming perceptible [30, 32, 39, 40]. It combines the reasoning strength of FL and the learning and adaptation power of NN.

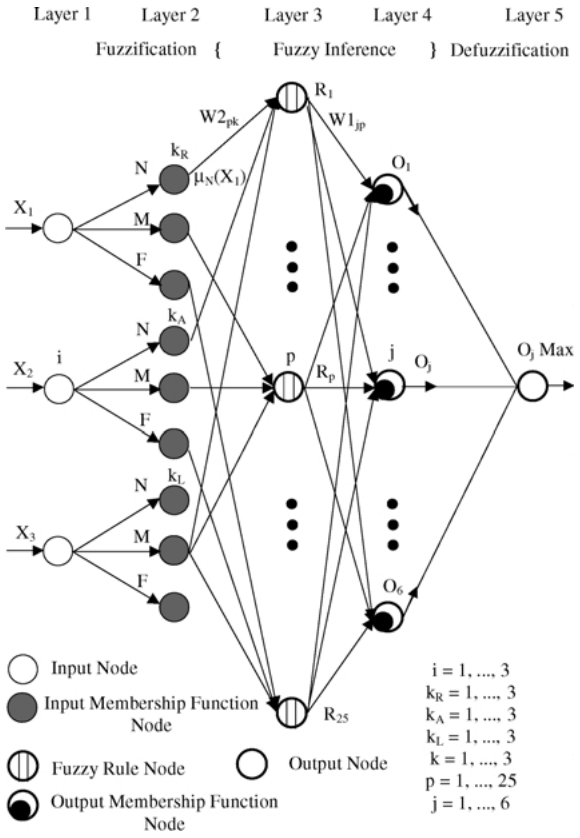

Figure 8. NN-FL/GBP classifier.

In particular, it appears that the capacities of NN can be significantly enhanced by endowing them with the ability to process fuzzy information. This provides a more powerful tool for fuzzy information processing and for exploring the functioning of human brains.

The NN-FL/GBP classifier is a multilayer feedforward fuzzy neural network built of five (05) layers as shown in Fig. 8.

The distances d_R, d_A and d_L are used to constitute the input vector **X** of this classifier. The input distance variables X_1, X_2 and X_3 have the same three degrees of qualitative values: Near (N), Medium (M) and Far (F) which are defined by the membership functions shown in Fig. 6.

3.2.3.1. Fuzzy Rule Base 2. This classifier uses a fuzzy rule base to mimic the input/output mapping of a human expert in recognition of obstacle avoidance situations. Indeed, this expert has formulated his knowledge in a linguistic form by establishing a set of fuzzy rules given in Table 2. Note that the particular cases (X_1 is N and X_2 is N and X_3 is N) and (X_1 is F and X_2 is F and X_3 is F) must be handled by the navigation system at a higher level.

Table 2. Fuzzy rule base 2.

If (X_1 is N and X_2 is N and X_3 is M) Then O_3
If (X_1 is N and X_2 is N and X_3 is F) Then O_3
If (X_1 is N and X_2 is M and X_3 is N) Then O_1
If (X_1 is N and X_2 is M and X_3 is M) Then O_5
If (X_1 is N and X_2 is M and X_3 is F) Then O_5
If (X_1 is N and X_2 is F and X_3 is N) Then O_1
If (X_1 is N and X_2 is F and X_3 is M) Then O_5
If (X_1 is N and X_2 is F and X_3 is F) Then O_5
If (X_1 is M and X_2 is N and X_3 is N) Then O_2
If (X_1 is M and X_2 is N and X_3 is M) Then O_4
If (X_1 is M and X_2 is N and X_3 is F) Then O_3
If (X_1 is M and X_2 is M and X_3 is N) Then O_6
If (X_1 is M and X_2 is M and X_3 is M) Then O_6
If (X_1 is M and X_2 is M and X_3 is F) Then O_3
If (X_1 is M and X_2 is F and X_3 is N) Then O_1
If (X_1 is M and X_2 is F and X_3 is M) Then O_5
If (X_1 is M and X_2 is F and X_3 is F) Then O_5
If (X_1 is F and X_2 is N and X_3 is N) Then O_2
If (X_1 is F and X_2 is N and X_3 is M) Then O_2
If (X_1 is F and X_2 is N and X_3 is F) Then O_4
If (X_1 is F and X_2 is M and X_3 is N) Then O_1
If (X_1 is F and X_2 is M and X_3 is M) Then O_6
If (X_1 is F and X_2 is M and X_3 is F) Then O_4
If (X_1 is F and X_2 is F and X_3 is N) Then O_6
If (X_1 is F and X_2 is F and X_3 is M) Then O_6
with $X_1 = d_R$, $X_2 = d_A$ and $X_3 = d_L$

3.2.3.2. Layer 1. Represents the input layer with three (03) input nodes receiving the components of the input vector **X**. This layer transmits these inputs to their corresponding membership functions in the next layer.

3.2.3.3. Layer 2. Performs the fuzzification process with nine (09) nodes representing the linguistic variables of the input membership functions. Each node calculates the degree of the measured data belonging to the kth membership function for the ith input variable e.g., for X_1, the membership degrees of fuzzy sets N, M and F are $\{\mu_N(X_1), \mu_M(X_1)$ and $\mu_F(X_1)\}$, respectively. This layer is not fully connected with layer 1. The connections exist only between the input nodes in the first layer and their corresponding membership function nodes in the second layer. The weights of these connections are set to 1.

3.2.3.4. Layer 3. Represents the fuzzy rule base with twenty five (25) nodes where each node corresponds to one fuzzy rule. The activation of nodes is achieved by the Min operation, then f the output sigmoïd function is applied in Eq. (17). The rule association is done among membership functions of different inputs. The output of each node represents the strength of firing the rule defined by that node. This layer is not fully connected with layer 2. Each node receives only three (03) connections, from the corresponding node (N, M or F) of the first input, second input and third input. The connection weights $W2_{pk}$ are used to perform precondition matching of fuzzy rules.

$$R_p = f\big(\mathrm{Min}\big(\mu_{k_\mathrm{R}}(X_1)\mathrm{W}2_{pk_\mathrm{R}}, \mu_{k_\mathrm{A}}(X_2)\mathrm{W}2_{pk_A}, \mu_{k_\mathrm{L}}(X_3)\mathrm{W}2_{pk_\mathrm{L}}\big)\big) \tag{17}$$

3.2.3.5. Layer 4. Performs the sum operation with six (06) nodes corresponding to output membership functions, then f the output sigmoïd function is applied in Eq. (18). This layer is fully connected to layer 3.

$$O_j = f\left(\sum_p R_p \mathrm{W}1_{jp}\right). \tag{18}$$

3.2.3.6. Layer 5. Represents the output layer with one (01) output node performing the defuzzification process using the Max operation to obtain O_j. This layer is fully connected to layer 4 and the weights of these connections are set to 1.

To acquire the obstacle avoidance behavior, this classifier is trained, in the training environment shown in Fig. 3, using the *supervised* GBP paradigm (see NN/GBP Classifier in Section 3.1).

The NN-FL/GBP classifier is trained from fifty four (54) examples of the training set. The weights $W1_{jp}$ and $W2_{pk}$ are adjusted from a random weight initialization between $[-1, +1]$ with the learning rate $\eta = 0.5$. This classifier yields convergence to the tolerance $E_\mathrm{T} = 0.01$ in well under the cycle number CN(GBP) = 3000 cycles.

3.2.4. NN-FL-ART/SFAM Classifier. One of the main features of human memory is its ability to learn many new things without necessarily forgetting things learned in the past. The ability of humans to remember many details of an exciting movie is a typical example of fast learning. Such learning can be achieved exploiting the ART in combination with NN and FL. Indeed, the ART can be used to design a hierarchical artmap neural networks (ART-NN) that can rapidly self-organize stable categorical mappings between i-dimensional input vectors and n-dimensional output vectors [35, 58]. To better reflect the human reasoning, fuzzy inference is incorporated to the basic architecture leading to the fuzzy artmap neural networks (NN-FL-ART) [39, 40, 58, 59]. NN-FL-ART are capable of *fast* and *stable* learning of recognition categories in response to arbitrary sequences of analog or binary input patterns. Therefore, they achieve a synthesis of FL and ART-NN by exploiting a close formal similarity between the computations of fuzzy subset hood and ART category choice, resonance, and learning.

The NN-FL-ART/SFAM classifier is a fuzzy artmap neural network built of raw input layer (complement coder), input layer, output category layer, and category layer (category input) as shown, after learning, in Fig. 9.

The distances d_R, d_A and d_L are pre-processed as in Eq. (1) to constitute the input vector $\mathbf{X}$ of this classifier.

3.2.4.1. Raw Input Layer. With three (03) inputs corresponding to the components of the input vector $\mathbf{X}$.

3.2.4.2. Input Layer. With six (06) input nodes, constituting the input vector $\mathbf{I}$ in Eq. (19), obtained from the three (03) inputs implemented in a complementary coding form by a complement coder. The complement coder takes an input i-dimensional feature vector and produces $2i$ outputs where the additional i output

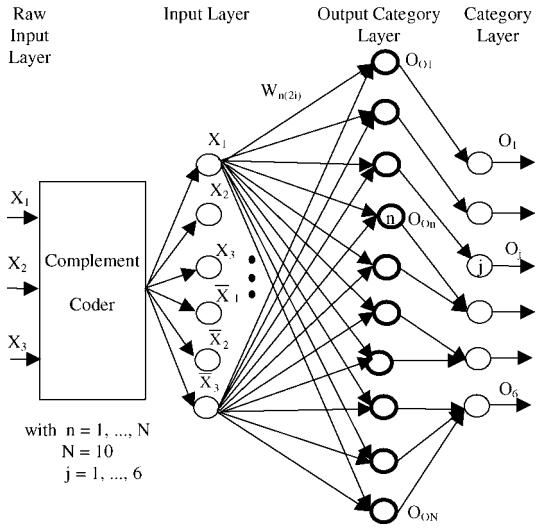

Figure 9. NN-FL-ART/SFAM classifier.

values are the ones complement of the original i inputs, respectively. The added input components are provided to make easier the formation of decision regions by the network. This layer transmits these inputs to all nodes of the next layer.

$$\mathbf{I} = [X_1, X_2, X_3, \bar{X}_1, \bar{X}_2, \bar{X}_3]. \tag{19}$$

3.2.4.3. Output Category Layer. Starts with no output category nodes (no weights) and grows in size either by incorporating an example into an existing output category node or creating a new output category node for it to reach finally ten (10) output category nodes.

For each input $\mathbf{I}$, the output activation function (choice function), T_n, is defined by:

$$T_n(\mathbf{I}) = \frac{\left|\mathbf{I}_{(2i)} \wedge \mathbf{W}_{n(2i)}\right|}{\lambda + \left|\mathbf{W}_{n(2i)}\right|}, \tag{20}$$

where λ is a small positive constant $0 < \lambda \ll 1$, the fuzzy AND [48] operator $\wedge$ is defined by:

$$(\mathbf{p} \wedge \mathbf{q})_m \equiv \text{Min}(p_m, q_m), \tag{21}$$

with $m = 1, \ldots, M$ and the norm $|\cdot|$ is defined by:

$$|\mathbf{p}| \equiv \sum_{m=1}^{M} |p_m|, \tag{22}$$

for any m-dimensional vectors $\mathbf{p}$ and $\mathbf{q}$ [40, 58–60].

The category choice is then indexed by N, where:

$$T_N = \text{Max}\{T_n : n = 1, \ldots, N\}. \tag{23}$$

3.2.4.4. Category Layer. With six (06) category nodes O_j, the maximum number of categories the network can learn, each to be labeled as a unique category or class.

To acquire the obstacle avoidance behavior, the NN-FL-ART/SFAM classifier is trained, in the training environment shown in Fig. 3, using the *supervised fast* and *stable* SFAM paradigm detailed in [60]. SFAM is specialized for pattern classification which can learn every single training pattern in only a handful of training iterations, starts with no connection weights but grows in size to suit the problem, and contains only one user-selectable parameter: the baseline of the vigilance. This incremental learning has proven to be stable because all adaptive weights can only decrease in time. This classifier decides if one or several output category nodes are required to represent a particular category. As the vigilance is increased, this classifier creates more output category nodes for the same input patterns, with finer category boundaries. Indeed, the network grows to represent the problem as it sees fit, instead of being told by the network designer to function within the confines of some static architecture.

The NN-FL-ART/SFAM classifier is trained from fifty four (54) examples of the training set with the input pre-processing factor $a = 3$ either by incorporating an example into an existing output category node or creating a new output category node for it. This classifier sprouted the output category node number $\text{OCNN} = N$ with $N = 10$ and yields convergence in well under the cycle number $\text{CN} = 1$ with the learning rate $\eta = 1$, small positive constant $\lambda = 0.000001$, baseline of the vigilance $\sigma = 0.4$.

Note that for the FL classifier there is no learning phase while for NN/GBP, NN/GA, NN/GA-GBP, NN-FL/GBP and NN-FL-ART/SFAM a learning phase is necessary where software programs, developed in Borland C++, train the networks until the convergence. Then, the obtained weights are used (stored) in ROMs of the corresponding FPGA architecture suggested in the following Section for each classifier.

4. Synthesis of the Suggested Pattern Classifiers and the FPGA Architectures for Their Implementation

In this Section, a synthesis on the suggested classifiers and the FPGA architectures for their implementation are presented. The synthesis is essentially based on the establishment of features, different relationships and comparisons among combined computational tools in each suggested pattern classifier as well as a discussion on the obtained results.

4.1. Features and Comparisons

4.1.1. Classifiers Based on Supervised Learning and Adaptation Paradigms (NN/GBP, NN/GA, NN/GA-GBP, NN-FL/GBP and NN-FL-ART/SFAM). The suggested classifiers based on NN present interesting characteristics such as learning, adaptation, generalization and real-time to solve the obstacle avoidance problem enhancing then the capacity of IAV to respond to the environment stimuli. Indeed, NN allow to learn and generalize from examples without knowledge of

neither classification criteria nor generalization rules. In applications where rules cannot be known, NN may be able to represent those rules implicitly as stored connection weights [39].

4.1.1.1. NN/GBP Classifier. GBP is a steepest gradient descent method which uses several parameters leading to a dependency of the algorithm on the choice of these parameters as well as on the selection of initial weights but learning by GBP has been quite successful when applied to specific problems. Nevertheless, problems can be occurred resulting in failure: this corresponds to getting trapped in a local minimum during gradient descent.

4.1.1.2. NN/GA Classifier. GA are characterized by robust performance using the capabilities of *adaptation* and survival as patterned after biological evolution. They have been theoretically and empirically proven to provide robust search in complex spaces, because they are not limited by restrictive assumptions of continuity, existence of derivatives, unimodality and other restrictions [40]. GA are global optimization methods but extremely costly in terms of computation and very slow [41].

4.1.1.3. Comparison of Supervised GA and GBP Learning Paradigms. The basic difference between GBP and GA based training mechanisms is that, unlike GBP, GA does not make use of local knowledge of the parameter space. This difference usually results in GBP outperforming GA in the training of feedforward NN. However, GBP often gets stuck in a local minimum and either does not converge to train the NN, or performs very poorly. It is apparent that in such cases, knowledge of the local parameter space is a disadvantage and hence the GA method is superior [57].

4.1.1.4. NN-FL/GBP Classifier. Neural and fuzzy systems have several characteristics in common. They are each model-free function estimators that can be adjusted or trained for improved performance. By their nature, they are each readily implemented with parallel processing techniques. A NN consists of connections among a distribution of nodes, fuzzy systems process rules that associate, in parallel, fuzzy output sets with fuzzy inputs [39]. Both NN and Fuzzy systems have been shown to have the capability of modeling complex nonlinear processes to arbitrary degrees of accuracy [41]. In fact, the integration of NN and FL brings the low-level learning and the computational power of NN into FL systems, and provides the high-level, human-like thinking and reasoning of FL systems into NN. Such synergism of integrating NN and FL systems into a functional system provides a new direction towards the realization of intelligent systems for various applications.

In general, the real world problems need to be represented, in particular for knowledge representation, of two (02) principal kinds. The first part need an *implicit* representation where NN are well appropriated for such task. The second part need an *explicit* representation where FL is well appropriated for such task. Indeed, the knowledge is expressed explicitly in the rules by the FL, while the NN express the knowledge implicitly in the weights. Thus, fuzzy systems could express the knowledge but are not able to learn (they only adapt themselves), while NN possess this ability. Thus, the benefits of neural navigation lie in the NN properties. In particular, the *adaptation* capacity presents a determinant interest for all evolutionary problems. It is largely related to the learning and generalization capabilities with which the network is able to take into account new constraints and data of external environment. By another way, the benefits of fuzzy navigation lie in its efficiency and intelligence since FL allows the achievement of state judgment like human being [39, 40].

4.1.1.5. NN-FL-ART/SFAM Classifier. NN-FL-ART/SFAM can learn every single training pattern in only a handful of training iterations, starts with no connection weights but grows in size to suit the problem, has learning equations that don't incorporate a single partial derivative and contains only one user selectable parameter. Normally when multilayer NN (with GBP) are used for pattern classification, a single output node is assigned to each class (category) that the network is expected to learn. In SFAM, the assignment of output nodes to classes is left up to the network [60]. This classifier has the capability to cope with changing non-stationary environments. Setting a certain association as 'wrong' is not definite, due to a nonmonotonical learning law it can become 'favorable' again. In such a way the system self-organizes its reactions to received stimuli. Learning after a disturbance results in an increase of perceptual categories number, which embodies system's capabilities to deal with non-stationarities of the environment, or of its own structure. In both cases, an importance of the FL-ART feature of preserving already stored knowledge while incorporating new data, can be easily seen [31].

4.1.1.6. Comparison of Supervised SFAM and GBP Learning Paradigms. Fuzzy artmap networks are self-stabilizing while in GBP networks new information gradually washes away old information. A consequence of this is that a GBP network has separate training and performance phases while artmap systems perform and learn at the same time. Besides, the absence of a self-scaling property in GBP prevents it from being able to alter the importance of each expected component when it is embedded in expected outputs of variable complexity. Therefore, instead of gradually discovering invariant characteristic properties of all the examples presented to it by the teacher and thus generating a prototype, each example is treated as a prototype. In contrast, the SFAM can discover critical feature patterns. Another difference is that while fuzzy artmap systems can learn both in a fast as well as in a slow match configuration, GBP networks can only learn in slow mismatch configuration. This means that an artmap system learns, i.e., adapts its weights, only when the input matches an established category, while GBP networks learn when the input does not match an established category. In GBP networks there is always the problem of the system getting trapped in a local minimum while this is impossible for fuzzy artmap systems. However, in fuzzy artmap systems learning may depend upon the ordering of the input patterns.

4.1.2. Classifier Based on Supervised Adaptation Paradigm (FL). FL Classifier: FL is built around the concept of reasoning in degrees rather than in absolute black-and-white terms [41]. It can express qualitative 'values' of human logic well and provide smooth actions through continuous membership functions [40]. In fact, FL looks at the world in imprecise terms by combining fuzzy sets with fuzzy rules to produce overall complex nonlinear behavior, in much the same way that our own brain takes an information [41]. The goal of the FL approach is then to mimic the aspect of human cognition called approximate reasoning [39]. In addition, the use of FL theory to solve navigation problems proves interesting and efficient if the classification criteria or generalization rules are known (given by an expert).

A major limitation of fuzzy systems is the complexity of designing the fuzzy rule base as the number of system inputs and outputs increases. Likewise, finding the exact membership functions to achieve the best system performance may be also difficult [39].

4.2. Discussions on Obtained Results

The different tests over the suggested classifiers, based on supervised learning and adaptation paradigms, SI (NN/GBP and FL) and SHI (NN/GA, NN/GA-GBP, NN-FL/GBP and NN-FL-ART/SFAM) have been achieved with regard to the same:

– fifty four (54) examples,
– tolerance ($E_T = 0.01$).

The suggested classifiers based on *soft computing* have efficiently solved the obstacle avoidance behavior problem underlying the fact that the used technologies can be used separately or in combination to learn the spatial situations allowing IAV to navigate successfully in partially structured environments. Table 3 summarized some characteristics of these pattern classifiers. As shown in this table, all of the classifiers converge to the tolerance under a fixed number of training cycles except NN/GA which must take more training cycles.

The combination of GA and GBP gives better result since NN/GA-GBP (curve c in Fig. 10) converges more rapidly than NN/GBP (curve a in Fig. 10) and NN/GA (curve b in Fig. 10) suggesting then that a hybrid learning algorithm (GA-GBP) is more efficient than the use of GBP or GA separately. In fact, learning by GA-GBP can not only remedy to the problem of local minima but also to the slowness of GA.

Table 3. Comparison of some characteristics of the suggested classifiers.

	NN/GBP	FL	NN/GA	NN/GA-GBP	NN-FL/GBP	NN-FL-ART/SFAM
Weight number	45	–	45	45	225	60
Rule number	–	25	–	–	25	–
Learning rate	0.1	–	–	0.3	0.5	1
Cycle number (GA)	–	–	10,000	70	–	–
Cycle number (GBP or SFAM)	927	–	–	290	3000	1

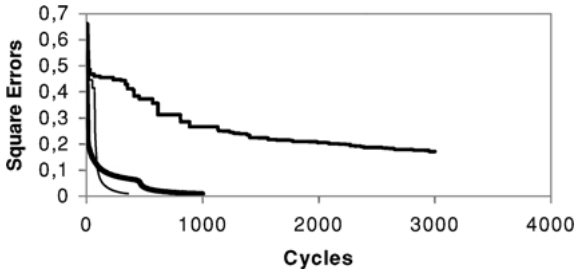

Figure 10. Error graphs during training for NN/GBP, NN/GA and NN/GA-GBP.

GA converge faster than GBP in its early stages of training as illustrated by the curves b and a in Fig. 10, respectively. While the weakness of GA in local fine-tuning is obvious, although this problem was circumvented in the NN/GA-GBP method, the speed of convergence in the early stages of training is the crucial factor in evaluating their utility as a method of training NN. As shown in curve c in Fig. 10, GA successfully reduce the error at the beginning of the training while GBP conducts a powerful local search to reach the tolerance $E_T = 0.01$. Thus, the training of NN with GA-GBP shows effectively that it outperforms GA and GBP taken separately.

The introduction of FL in NN allows to take into account the imprecise and inaccurate data presented to the network. The capacity of reasoning is exploited via the fuzzy rules expressed as neurons to take various decisions which didn't exist in the NN structure enhancing then the reasoning capacity of NN. Thus, when a fuzzy rule base is available, the use of FL or NN-FL/GBP is advised. Moreover, although FL and NN-FL/GBP have difficulties in establishing the fuzzy rule base and the choice of membership functions, they present the benefit of stability.

The NN-FL-ART/SFAM converges in only one cycle suggesting that this classifier is faster than the others. This is foreseeable due to the fact that it captures many of the strong points of NN, FL and ART technologies leading to a hybrid method confirming that this kind of classifier is more efficient than the other classifiers and can be closer to the human brain model. The supervised learning SFAM does not present neither convergence difficulties nor slowness problem. With only one parameter to adjust, this type of learning is fast and stable. The system learns incrementally in order to let the IAV adapt to changing characteristics of non-stationary environment. Indeed, one of the most useful advantage of this classifier is the possibility of signaling the appearance of a new category (in this context: a new obstacle avoidance situation). If the NN is unsuited to a given situation (it does not know how to behave in a given situation), it can be taught on-line and can recognize the appearance of a new type of input creating a new output.

4.3. FPGA Architectures for the Suggested Pattern Classifiers

From the design with discrete components containing simple gates, the industry has moved from TTL to CMOS, then PLDs and now FPGAs and ASICs. Each of these moves has contributed to higher operating speeds, lower power consumption, and easier design as well as reduction in physical size. However, several factors affect the decision to implement a design using an ASIC or an FPGA. For instance, current ASICs have a low cost-per-gate advantages as well as an inherent speed advantage; in contrast, FPGAs have been winning with their time-to-application, low non-recurring engineering fees, and reprogrammable features [61]. Thus, continuing process advances suggest that FPGAs will and should hold an increasing share of the market in the near future.

In this Section, an FPGA hardware architecture, based on Xilinx technology, is suggested for each pattern classifier. The NN/GBP, NN/GA and NN/GA-GBP classifiers have the same FPGA architecture since having the same network topology. The FPGA architectures of NN-FL/GBP and NN-FL-ART/SFAM classifiers are slightly different in terms of number of layers, nature of the neuron while the one of FL classifier is completely different.

4.3.1. FPGA Architecture for NN/GBP, NN/GA and NN/GA-GBP Classifiers. The suggested architecture is based on a methodology optimizing the area [62–64]. The simplified model of a neuron illustrated in Fig. 11(a) can be represented it in its equivalent hardware model as shown in Fig. 11(b). This model is mainly based on a Multiply and ACcumulate (MAC) operator which computes the weighted sum $\sum_p \delta_p W_{pm}$. The MAC result points a Look-Up Table (LUT) which implements the sigmoid activation function

$$f(\alpha) = \frac{1}{1 + e^{-\alpha}} \quad \text{with } \alpha = \sum_p \delta_p W_{pm}.$$

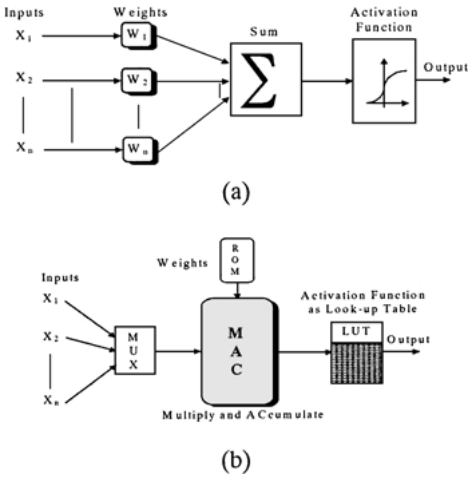

Figure 11. (a) Simplified model of a neuron. (b) Proposed hardware description for NN/GBP, NN/GA and NN/GA-GBP.

This network may be implemented using three (03) layers but taking into account the methodology presented in [63], this number is reduced to two (02) layers. Based upon the above hardware description, the algorithm describing the methodology is:

1. Determine the layer, other than the input one, which is constituted by the greatest number of neurons N_n (for our case it is the output layer $N_n = 6$).
2. The number of MAC needed in the architecture is then $N_{\mathrm{MAC}} = N_{\mathrm{n}}$.
3. Each MAC has its own ROM of weight. The depth of each ROM is equal to the number of node constituting the input and the hidden layer.
4. The inputs for each node will be controlled by a multiplexer (MUX). The size S of the multiplexer is determined by the number of neurons that constitute the most important layer other than the output one (for our case the size is $S = 5{:}1$).

Thus, the resulting architecture is illustrated in Fig. 12. It exhibits a high degree of regularity and can be easily modified (reduce or increase the number of MAC and change the size of the multiplexer) to achieve any other topology of feedforward NN.

The most important parts in the resulting architecture are MAC and LUT modules.

– Multiply and ACcumulate (MAC): The major problem when implementing NN stems from multiplication. In the literature, several algorithms and architectures to compute digital multiplication are presented [64–66]. Digital multiplication can be calculated using LUT [67, 68]. In this case the n-bit operand are applied as addresses to a preprogrammed memory. It is an attractive method and

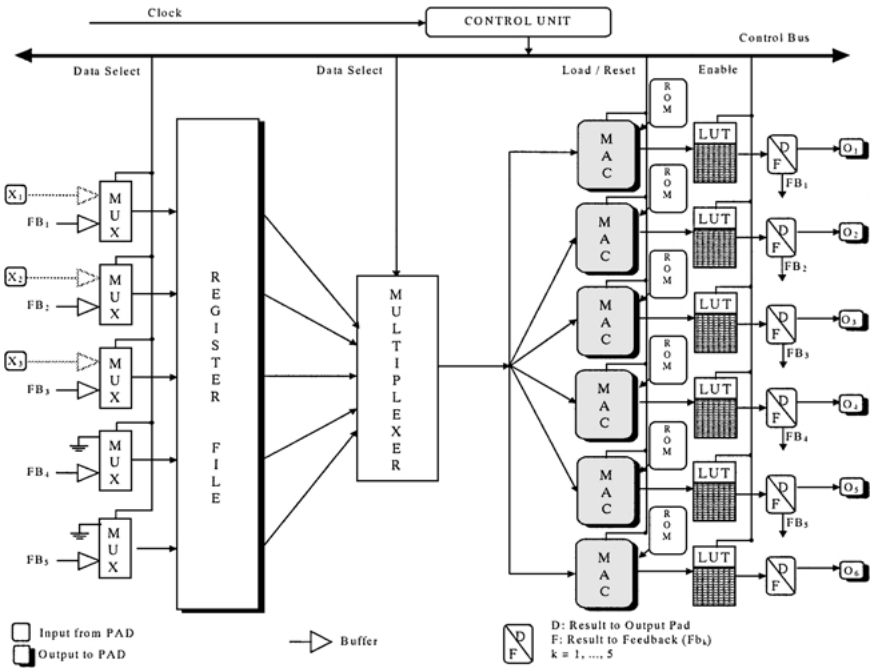

Figure 12. FPGA architecture for NN/GBP, NN/GA and NN/GA-GBP classifiers.

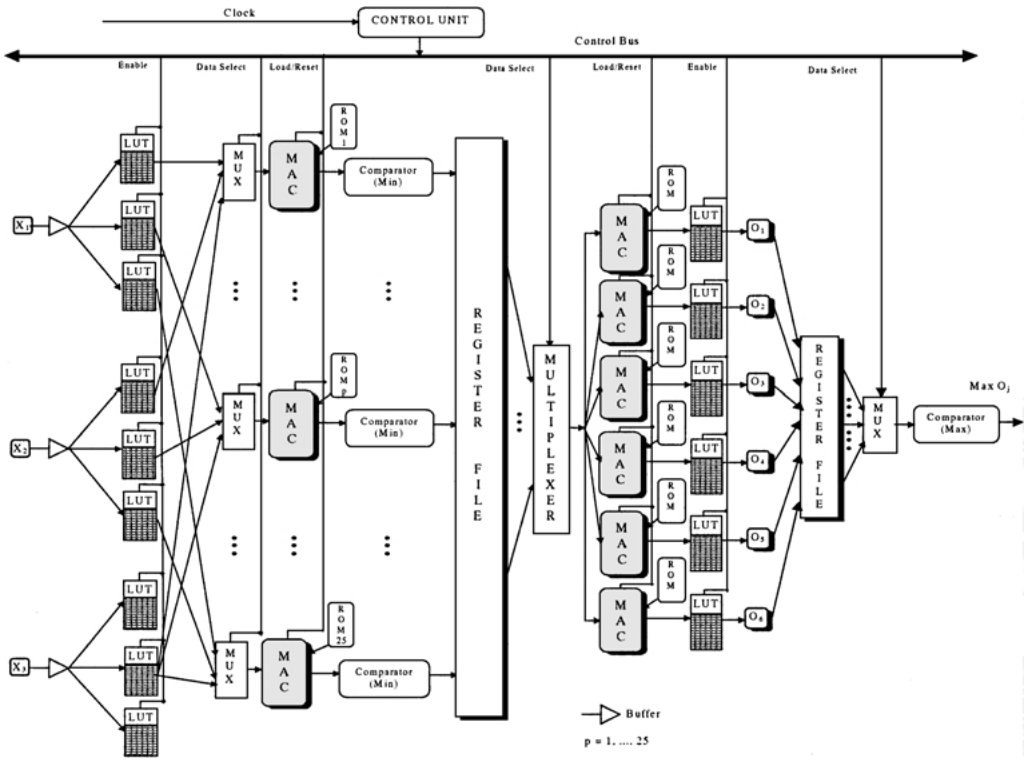

Figure 13. FPGA architecture for NN-FL/GBP classifier.

adapted to SRAM based FPGA. However, the use of LUT as MAC in the suggested NN involves some quandary: (i) The contains of the LUT must be different for each neuron, (ii) The LUT must be updated for computing each layer, introducing then, a supplementary circuitry and thus complicating the control part. Consequently, a high-speed and very compact two's complement modified booth parallel multiplier designed in [63, 64] is used in the suggested NN.

– Activation Function: An investigation of the behavior of the sigmoid function shows that it rapidly approaches its maximum value (1.0) as the input becomes greater than (7), and rapidly approaches its minimum value (0.0) as the input becomes lesser than (-8). These optima values are reached before the input leaves the 10-bit interval. Consequently, the 16-bits output of the MAC module are truncated to 10-bit. So, the activation function is implemented as a 1024 words of 8-bit width ROM.

The high regularity and compactness are the main features of the architecture resulting from the presented methodology. A fully integrated digital NN for the obstacle avoidance behavior has been entirely implemented using just a single XC4062EX Xilinx FPGA.

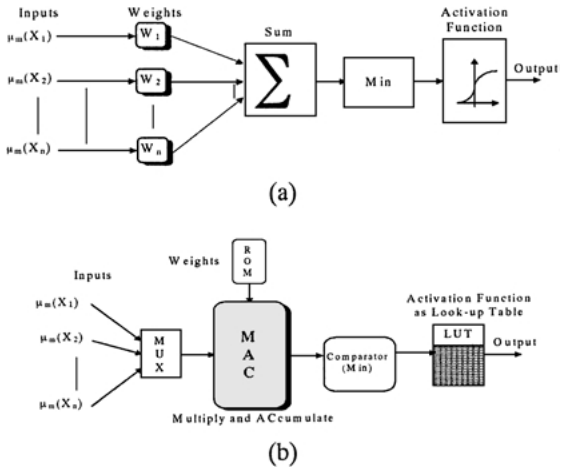

Figure 14. (a) Simplified model of a neuron. (b) Proposed hardware description for NN-FL/GBP.

4.3.2. FPGA Architecture for NN-FL/GBP Classifier. The layers in this classifier are not as regular as in the

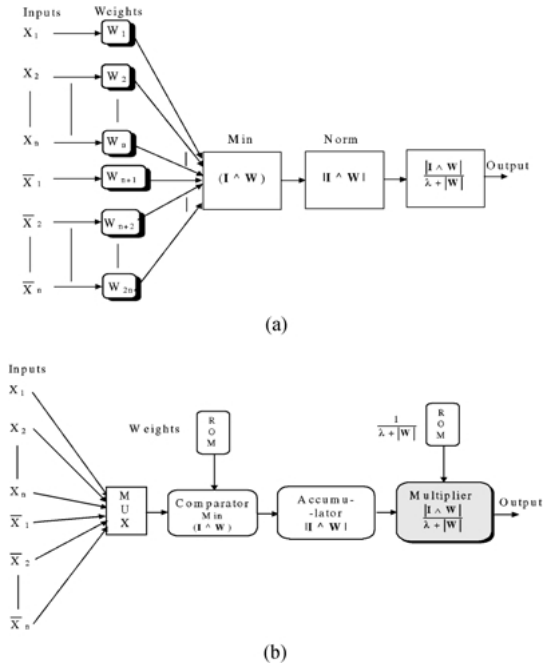

Figure 15. (a) Simplified model of a neuron. (b) Proposed hardware description for NN-FL-ART/SFAM.

precedent one. In this architecture, the network has four (04) layers, as shown in Fig. 13, where the second layer is essentially constituted of LUT which implement the fuzzification of inputs through the fuzzy linguistic variables N, M and F. The third one implements Eq. (17) using MAC components with ROM where the obtained weights (after learning) are stored to compute the products of the membership degrees by their corresponding weights, the results are then compared to give the minimum of these products using a comparator (Min) which points out a LUT implementing the sigmoid function, see Eq. (17). Thus, each neuron in Fig. 14(a) uses its equivalent hardware shown in Fig. 14(b) to implement each fuzzy rule listed in Table 2. Afterwards, to implement Eq. (18), the equivalent hardware shown in Fig. 11(b) for each neuron is used in this fourth layer. The ROM are used also to store the obtained weights (after learning) of this layer. Finally, the defuzzification of outputs is implemented using a comparator (Max) giving the final obstacle avoidance situation.

4.3.3. FPGA Architecture for NN-FL-ART/SFAM Classifier. In this classifier, the output category layer

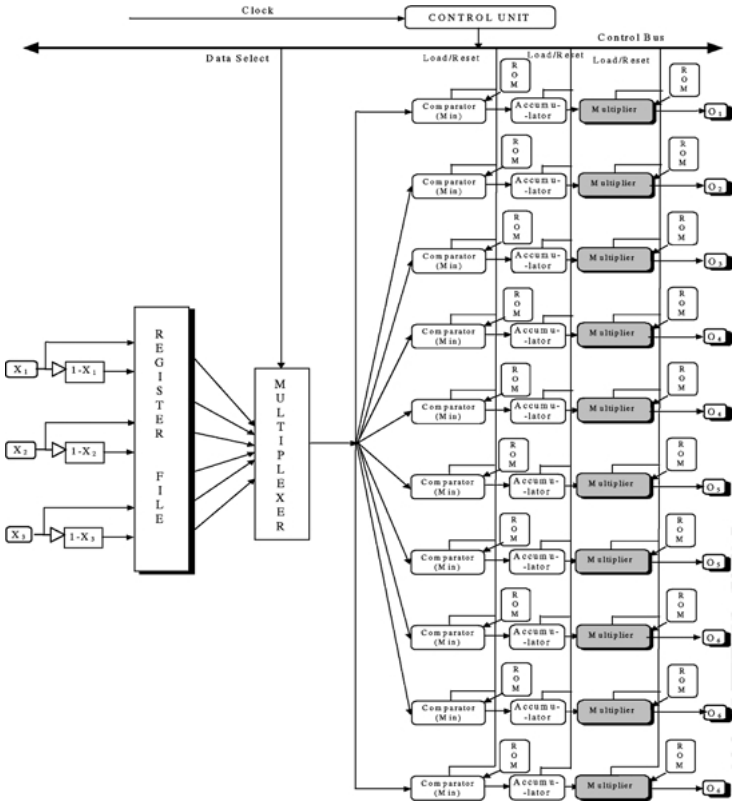

Figure 16. FPGA architecture for NN-FL-ART/SFAM classifier.

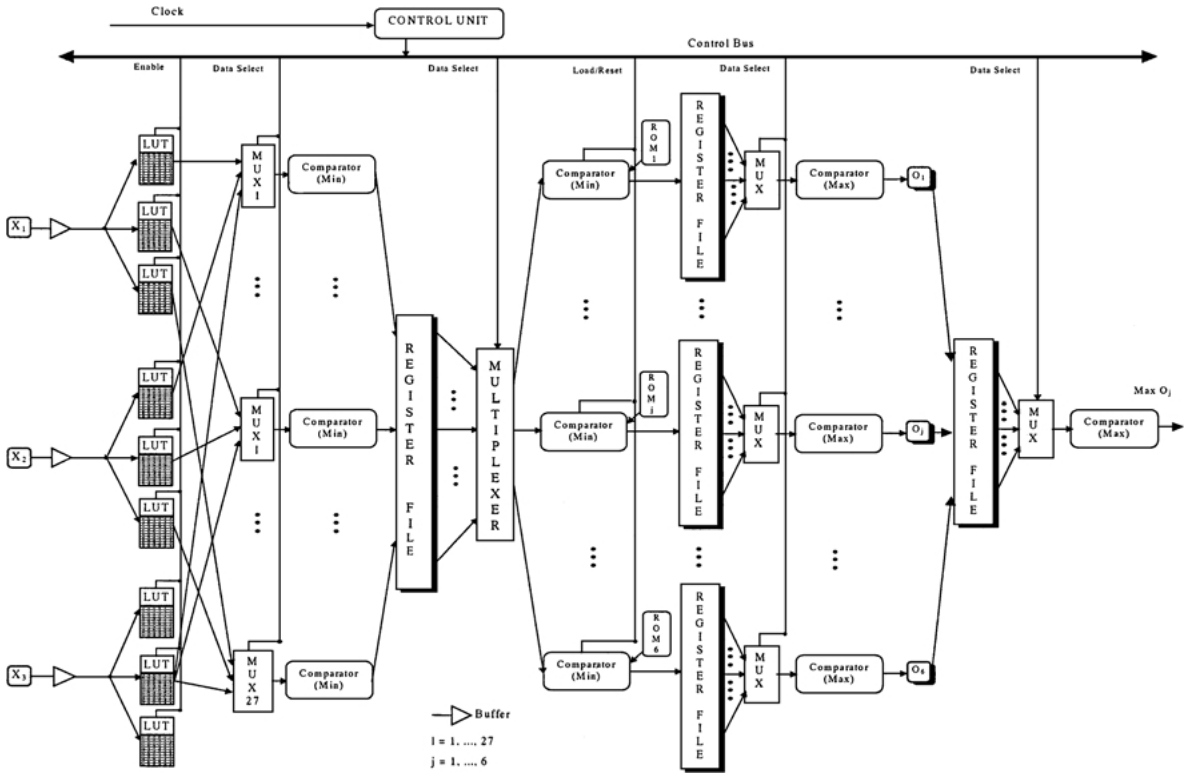

Figure 17. FPGA architecture for FL classifier.

is implemented such as each neuron in Fig. 15(a) is modeled in its hardware equivalent shown in Fig. 15(b) by a comparator (Min), an accumulator, a multiplier and ROM memories where the obtained weights (after learning) are stored as well as the corresponding term $1/(\lambda + |\mathbf{W}|)$. The global architecture, illustrated in Fig. 16, is simpler and uses less hardware than the others.

4.3.4. FPGA Architecture for FL Classifier. In this classifier, the fuzzification of inputs is implemented as a LUT, where the fuzzy linguistic variables are N, M and F, which points out the LUT containing the membership degrees as shown in Fig. 17. The fuzzy inference is implemented firstly by a comparator (Min) achieving the Eq. (10). Secondly, a comparator (Min) is used to give the minimum of the obtained result from the first step and the fuzzy obstacle avoidance situation component from the corresponding fuzzy rule (Table 1). Note the particularity in this implementation where the ROM are used to implement the fuzzy If/Then rules listed in Table 1. Afterwards, the elements of Eq. (12) are computed using a comparator (Max) giving the final fuzzy obstacle avoidance situation vector Eq. (13). Finally, the defuzzification of outputs is implemented using a comparator (Max) giving the final obstacle avoidance situation Eq. (14).

5. Conclusion and Discussion

To solve the obstacle avoidance behavior problem of IAV in partially structured environments, pattern classifiers in IS and HIS based on *soft computing* and exploiting the capabilities of NN, GA, FL and ART are suggested in this paper. These classifiers are based on *supervised learning* and *adaptation* paradigms to classify spatial obstacle avoidance situations (topological situations).

The suggested pattern classifiers are characterized by their learning, adaptation and generalization capabilities, inaccurate data processing, classification, decision-making, robustness, massively parallel computations and distributed memory. In fact, generalization is an essential part of learning as it permits to remember facts that apply to whole classes rather than remembering many specific facts that apply only to

individual members of the class. It serves as an efficient mode of memorization and storage. In short, generalization is an essential trait of intelligent behavior. By another way, the signals of sensors are often noisy or they are defective giving incorrect data. The suggested classifiers with their inherent features of *adaptivity* and *high fault and noise tolerance* handle this problem making the *soft computing* based approaches *robust*. Indeed, malfunctioning of one of the sensors or if a part of a classifier is disabled (e.g., a defective neuron for NN) do not strongly impair the obstacle avoidance behavior. For instance, this is possible because the knowledge stored in a NN is *distributed* over many neurons and interconnections, not just a single or a few units. Consequently, concepts or mappings stored in a NN have some degree of redundancy built in through this distribution of knowledge.

Afterwards, a synthesis of the suggested pattern classifiers is presented where their results and performances are discussed. The presented synthesis reveals the fundamental features distinguishing these classifiers and underlies divers aspects of human intelligence in *soft computing* which enhance the IAV behaviors. It shows also that although they are different, the used technologies are complementary in several aspects giving then interesting properties when used in combination.

The obtained results, from training and making in use these classifiers, show an interesting GA-GBP combination to train NN. This combination is an efficient approach to train NN because it takes advantage of the strengths of GA and GBP (the fast initial convergence of GA and the powerful local search of GBP), and overcomes the weaknesses of both paradigms (the weak fine-tuning capability of GA and a flat spot in GBP). The obtained results allow to conclude also that NN-FL-ART/SFAM is more efficient than the others since it combines the complementary strengths of NN, FL and ART to learn more rapidly (in one epoch) the spatial obstacle avoidance situations and therefore performs more quickly than the other classifiers with less parameters to adjust under the assumption that it can learn new situations. In other words, for a new class all the presented classifiers must be trained again with all the training set except for the NN-FL-ART/SFAM which must be trained only with the new training examples.

Finally, an FPGA architecture, characterized by their high flexibility and compactness, is suggested for the implementation of each pattern classifier. Indeed, FPGA present several advantages as high operating speeds, low power consumption, reprogrammable features, easier design and fast time-to-application, low nonrecurring engineering fees as well as a small physical size.

Once implemented on FPGA, the suggested *soft computing* based classifiers provide IAV with *more autonomy*, *intelligence* and *real-time* processing capabilities making them *more robust* and *reliable*. Thus, they bring their obstacle avoidance behavior near to that of humans in the recognition, learning, adaptation, generalization, reasoning and decision-making, and action.

An interesting alternative for future research is the integration of NN, FL and GA which will involve further progress on intelligent behaviors of IAV, control architectures, learning paradigms and mechanisms for combining the three techniques. Continued development of applications will reveal further areas for useful HIS, and guidelines for choosing practical applications will emerge.

References

1. O. Azouaoui and A. Chohra, "Evolution, behavior, and intelligence of autonomous robotic systems (ARS)," in *Proc. 3rd Int. IFAC Conf. Intelligent Autonomous Vehicles*, Madrid, Spain, March 25–27, 1998, pp. 139–145.
2. B. Bosacchi and I. Masaki, "Fuzzy logic technology & the intelligent highway system (IHS)," in *Proc. 2nd Int. IEEE Conf. Fuzzy Systems*, vol. I, San Francisco, CA, 1993, pp. 65–70.
3. T. Fukuda, F. Arai, and K. Shimojima, "Intelligent robotic system," in *Proc. Int. Multiconf. Computational Engineering in Systems Applications*, France, 1996, pp. 1–10.
4. T. Shibata and T. Fukuda, "Coordinative behavior in evolutionary multi-agent system by genetic algorithm," in *Proc. Int. IEEE Conf. Neural Networks*, vol. I, San Francisco, CA, 1993, pp. 209–214.
5. T. Tanaka, Y. Kojima, J. Ohwi, K. Yamafuji, and S.V. Ulyanov, "Intelligent control of technology operations for robot of service use with manipulator," in *Proc. Int. IMACS IEEE-SMC Multiconf. Computational Engineering in Systems Applications*, France, 1996, pp. 788–793.
6. T. Tanaka, J. Ohwi, L.V. Litvintseva, K. Yamafuji, and S.V. Ulyanov, "Soft computing algorithms for intelligent control of a mobile robot for service use Part I and Part II," *Soft Computing*, vol. 1, no. 2, pp. 88–106, June 1997.
7. T. Fujii and T. Ura, "Development of an autonomous underwater robot 'Twin-Burger' for testing intelligent behaviors in realistic environments," *Autonomous Robots*, vol. 3, pp. 285–296, 1996.
8. C. Kujawski, "Deciding the behaviour of an autonomous mobile road vehicle," in *Proc. 2nd Int. IFAC Conf. Intelligent Autonomous Vehicles*, Helsinki, Finland, 1995, pp. 404–409.
9. D.B. Marco, A.J. Healey, and R.B. McGhee, "Autonomous underwater vehicles: Hybrid control of mission and motion," *Autonomous Robots*, vol. 3, no. 2/3, pp. 169–186, 1996.

10. S. McMillan, D.E. Orin, and R.B. McGhee, "Efficient dynamic simulation of an unmanned underwater vehicle with a manipulator," in *Proc. Int. Conf. On Robotics and Automation*, vol. 2, CA, May 8–13, 1994, pp. 1133–1140.
11. W. Niegel, "Methodical structuring of knowledge used in an intelligent driving system," in *Proc. 2nd Int. Conf. Intelligent Autonomous Vehicles*, Finland, 1995, pp. 398–403.
12. K. Schilling and C. Jungius, "Mobile robots for planetary exploration," in *Proc. 2nd Int. IFAC Conf. Intelligent Autonomous Vehicles*, Helsinki, Finland, 1995, pp. 110–120.
13. M. Vainio, P. Appelqvist, T. Schönberg, and A. Halme, "Group behavior of a mobile underwater robot society destroying distributed targets in a closed process environment," in *Proc. 3rd Int. IFAC Conf. Intelligent Autonomous Vehicles*, Madrid, Spain, March 25–27, 1998, pp. 112–117.
14. H.H. Wang, S.M. Rock, and M.J. Lee, "OTTER: The design and development of an intelligent underwater robot," *Autonomous Robots*, vol. 3, pp. 297–320, 1996.
15. I. Ashiru, C. Czarnecki, and T. Routen, "Characteristics of a genetic based approach to path planning for mobile robots," *J. of Network and Computer Applications*, vol. 19, pp. 149–169, 1996.
16. O. Azouaoui and A. Chohra. "Neural group navigation approach for autonomous robotic systems (ARS)," in *Proc. Second International ICSC Symposium on Engineering of Intelligent Systems*, University of Paisley, Scotland, June 27–30, 2000.
17. A. Chohra, A. Farah, and C. Benmehrez, "Neural navigation approach for intelligent autonomous vehicles (IAV) in partially structured environments," *Int. J. of Applied Intelligence*, vol. 8, no. 3, pp. 219–233, 1998.
18. A. Chohra, R. Tiar, and O. Azouaoui. "Fuzzy motion controller (FMC) for intelligent autonomous vehicles (IAV)." in *Proc. Second International ICSC Symposium on Engineering of Intelligent Systems*, University of Paisley, Scotland, June 27–30, 2000.
19. H. Herbstreith, L. Gmeiner, and P. Preuß, "A target-directed neurally controlled vehicle," in *Proc. Int. IFAC Conf. Artificial Intelligence in Real-Time Control*, Delft, The Netherlands, 1992, pp. 67–71.
20. A.W. Ho and G.C. Fox, "Neural network near-optimal motion planning for a mobile robot on binary and varied terrains," *IEEE Int. Work. on Int. Rob. and Sys., IROS'90*, 1990, pp. 593–600.
21. Y. Maeda, "Collision avoidance control among moving obstacles for a mobile robot on the fuzzy reasoning," in *Proc. Eight Symp. Theo. and Prac. of Rob. & Man.*, Cracow, Poland, 1990,
22. M. Maeda, Y. Maeda, and S. Murakami, "Fuzzy drive control of an autonomous mobile robot," *Fuz. Sets and Sys.*, vol. 39, pp. 195–204, 1991.
23. M. Maeda, M. Shimakawa, and S. Murakami, "Predictive fuzzy control of an autonomous mobile robot with forecast learning function,"*Fuzzy Sets and Systems*, vol. 72, no. 1, pp. 51–60, 1995.
24. M. Meng and A.C. Kak, "Mobile robot navigation using neural networks and nonmetrical environment models," *IEEE Control Systems*, pp. 30–39, October 1993.
25. E. Sorouchyari, "Mobile robot navigation: A neural network approach," in *Proc. Art Coll. Neuro. Eco. Poly.*, Lausanne, 1989, pp. 159–175.
26. T. Takeuchi, "An autonomous fuzzy mobile robot," *Advanced Robotics*, vol. 5, no. 2, pp. 215–230, 1991.
27. J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 1, pp. 18–28, 1997.
28. A.A. Baloch and A.M. Waxman, "Visual learning, adaptive expectations, and behavioral conditioning of the mobile robot MAVIN," *Neural Networks*, vol. 4, pp. 271–302, 1991.
29. A. Chohra, "Fuzzy ArtMap Neural Networks (FAMNN) based navigation for intelligent autonomous vehicles (IAV) in partially structured environments," in *Proc. ICSC Int. IFAC-IEEE Conf. on Neural Computing*, Vienna, Austria, Sept. 23–25, 1998, pp. 747–754.
30. A. Chohra, A. Farah, and M. Belloucif, "Neuro-fuzzy expert system *E_S_CO_V* for the obstacle avoidance behavior of intelligent autonomous vehicles (IAV)," *Int. J. of Advanced Robotics*, vol. 12, no. 6, pp. 629–650, 1999.
31. A. Dubrawski and J.L. Crowley, "Self-supervised neural system for reactive navigation," in *Proc. Int. IEEE Conf. on Robotics and Automation*, vol. 3, San Diego, CA, May 8–13, 1994, pp. 2076–2081.
32. I. Hiraga, T. Furuhashi, Y. Uchikawa, and S. Nakayama, "An acquisition of operator's rules for collision avoidance using fuzzy neural networks," *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 3, pp. 280–287, 1995.
33. P. Szynkarczyk and A. Masiowski, "The fuzzy artmap neural networks as a controller for the mobile robot," in *Proc. 3rd Int. Symp. on Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, Sept. 10–13, 1996, pp. 1201–1206.
34. J.A. Anderson, *An Introduction to Neural Networks*, The MIT Press: Cambridge, MA, London, England, 1995.
35. J.A. Freeman and D.M. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley: New York, 1992.
36. D.E. Goldberg, *Algorithmes Génétiques: Exploration, Optimisation et Apprentissage Automatique*, Addison-Wesley: France, 1994.
37. T. Khanna, *Foundations of Neural Networks*, Addison-Wesley: New York, 1990.
38. B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall: Englewood Cliffs, NJ, 1992.
39. L.R. Medsker, *Hybrid Intelligent Systems*, Kluwer Academic Publishers: Dordrecht, 1995.
40. D.W. Patterson, *Artificial Neural Networks: Theory and Applications*, Prentice Hall, Simon & Schuster (Asia) Pte Ltd: Englewood Cliffs, NJ/Singapore, 1996.
41. S.T. Welstead, *Neural Network and Fuzzy Logic Applications in C/C++*, John Wiley & Sons: Toronto, 1994.
42. P.J. Werbos, "Neurocontrol and fuzzy logic: Connections and designs," *Int. J. of Approximate Reasoning*, vol. 6, pp. 185–219, 1992.
43. S. Cherian and W. Troxell, "Intelligent behavior in machines emerging from a collection of interactive control structures," *Computational Intelligence*, vol. 11, no. 4, pp. 565–592, 1995.
44. S. Thrun and T.M. Mitchell, "Lifelong robot learning," *Robotics and Autonomous Systems*, vol. 15, pp. 25–46, 1995.
45. O. Aycard, F. Charpillet, and D. Fohr, "Place learning and recognition using hidden Markov models," in *Proc. Int. IEEE/RSJ Conf. on Intelligent Robots and Systems*, Grenoble, France, 1997, pp. 1741–1746.

46. Y.S. Kim, I.H. Hwang, J.G. Lee, and H. Chung, "Spatial learning of an autonomous mobile robot using model-based approach," in *Proc. 2nd Int. IFAC Conf. Intelligent Autonomous Vehicles*, Helsinki, Finland, 1995, pp. 250–255.
47. M. Agarwal, "A systematic classification of neural-network-based control," *IEEE Control Systems*, vol. 17, no. 2, pp. 75–93, 1997.
48. L.A. Zadeh, "Fuzzy sets," *Information & Control*, vol. 8, pp. 338–353, 1965.
49. L.A. Zadeh, "The calculus of fuzzy if/then rules," *AI Expert*, pp. 23–27, 1992.
50. C.C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller, Part I and Part II," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 404–435, 1990.
51. M.K. Ciliz and C. Isik, "Fuzzy rule-based motion controller for an autonomous mobile robot," *Robotica*, vol. 7, pp. 37–42, 1989.
52. H. Farreny and H. Prade, "Tackling uncertainty and imprecision in robotics," in *Proc. 3rd Int. Symp. on Robotics Research*, Gonvieux, 1985, pp. 85–91.
53. J. Holland, "Les algorithmes génétiques," *Pour La Science*, no. 179, pp. 44–51, Sept. 1992.
54. H. Ishigami, T. Fukuda, T. Shibata, and F. Arai, "Structure optimization of fuzzy neural network by genetic algorithm," *Fuzzy Sets and Systems*, vol. 71, pp. 257–264, 1995.
55. H. Kitano, "Empirical studies on the speed of convergence of neural network training using genetic algorithms," in *Proc. 8th JMIT National Conf. in Artificial Intelligence*, Boston, MA, vol. 2, 1990, pp. 789–795.
56. D.J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proc. 11th Int. Joint Conf. on Artificial Intelligence*, Detroit, MI, Morgan Kaufman: San Mateo, CA, 1989, pp. 762–767.
57. M. McInerney and A.P. Dhawan, "Use of genetic algorithms with back propagation in training of feed-forward neural networks," in *Proc. Int. IEEE Conf. Neural Networks*, vol. I, San Francisco, CA, March 28–April 1 1993, pp. 203–208.
58. G.A. Carpenter, S. Grossberg, and D.B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.
59. G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 698–713, 1992.
60. T. Kasuba, "Simplified fuzzy artmap," *AI Expert*, pp. 18–25, November 1993.
61. R.K. Awalt, "Making the ASIC/FPGA design," *Integrated System Design*, pp. 22–28, July 1999.
62. O. Azouaoui, "Neural networks based approach for the manipulators inverse Jacobian problem," in *Proc. ICSC Int. IFAC-IEEE Conf. on Neural Computing*, Vienna, Austria, Sept. 23–25, 1998, pp. 966–972.
63. Y.L. El-Haffaf, A.K. Oudjida, and A. Chohra, "Digital artificial neural network architecture for obstacle avoidance suitable for FPGA," in *Proc. Int. Conf. on Engineering Applications of Neural Networks*, Sweden, 1997, pp. 313–316.
64. A.K. Oudjida, "High speed and very compact two's complement serial/parallel multipliers using FPGA," *Int. Conf. on Signal Processing Application and Technology ICSPAT'96*, Boston, USA, Oct. 7–10, 1996, pp. 924–928.
65. K. Hwang, *Computer Arithmetic, Principles, Architecture and Design,* John Wiley & Sons: New York, 1979.
66. H. Ossoinig, E. Reisinger, C. Steger, and R. Weiss, "Design and FPGA-implementation of a neural network," *Int. Conf. on Signal Processing Application and Technology ICSPAT'96*, Boston, USA, Oct. 7–10, 1996, pp. 939–943.
67. G.R. Goslin, "16-Tap, 8-bit FIR filter application guide," *Xcell Journal*, Xilinx, 1994.
68. G.R. Goslin, "Using FPGAs in digital signal processing application," *Int. Conf. on Signal Processing Application and Technology*, 1995, pp. 145–149.

Ouahiba Azouaoui received the Electronics Engineer degree from the Ecole Nationale Polytechnique (ENP) at Algiers, Algeria in 1988. She received her Magister's degree (Degree of Advanced Studies) in Cybernetics option Robotics from the LRIA (Laboratoire de Robotique et d'Intelligence Artificielle) of the CDTA (Centre de Développement des Technologies Avancées) at Algiers, Algeria in 1992. Currently, she is researcher at the LRIA/CDTA and her Doctorate es-science degree is in progress at the ENP since September 1999. Her research interests are mobile manipulators, control, navigation, autonomous robotic systems, group behaviors, soft computing, hybrid intelligent systems, reinforcement learning, adaptive resonance theory, neural networks, and evolutionary algorithms.

Amine Chohra received the Electronics Engineer degree option Control in 1988 from the Electronics Department of USTO (Université des Sciences et de la Technologie d'Oran) at Oran, Algeria. He received his Magister's degree (Degree of Advanced Studies) in 1991 in Cybernetics option Robotics from CDTA (Centre de Développement des Technologies Avancées) at Algiers, Algeria. He received his Doctorate es-sciences in 1999 from the Electronics Department of ENP (Ecole Nationale Polytechnique) at Algiers, Algeria. He was researcher at the LRIA (Laboratoire de Robotique et d'Intelligence Artificielle) of CDTA until November 1999 as member of the Structures and Dynamics of Robots team and leader of the Autonomous Robotic Systems team, respectively. He has a post

doctoral position of two periods from December 1999 to August 2000 with Behavior Engineering team at AiS-GMD, Sankt Augustin, Germany ; and from October 2000 to July 2001 with Dependable Computing Group team at IEI-CNR, Pisa, Italy. Currently, he is teacher/researcher at LVR (Laboratoire de Robotique et Vision) of ENSI (Ecole Nationale Supérieure d'Ingénieurs), Bourges, France. His research interests are mobile robotics, intelligent autonomous vehicles, navigation, behavior-based robotics, intelligent behaviors, machine learning, reinforcement learning, soft computing, hybrid intelligent systems, field programmable gate array, robotic soccer, remote robot control via internet, wireless communication systems, fault-tolerant systems, intelligent software maintenance, dependability and performability evaluation, real-time, autonomy, and intelligence.