
Journal of Intelligent and Robotic Systems **30:** 311–329, 2001.

Intelligent Adaptive Mobile Robot Navigation

S. NEFTI
Manchester Metropolitan University, Department of Engineering and Technology, John Dalton building, Chester street, Manchester M1 5GD, UK

M. OUSSALAH
K.U. Leuven, PMA, Celestijnenlaan 300B, 3001 Heverlee, Belgium

K. DJOUANI and J. PONTNAU
LIIA (Laboratoire d'informatique industriel et de l'automatique), Université Paris XII, 122, Rue Paul Armangot, 94400 Vitry sur Seine, France

(Received: 23 September 1999; in final form: 18 July 2000)

Abstract. This paper deals with the application of a neuro-fuzzy inference system to a mobile robot navigation in an unknown, or partially unknown environment. The final aim of the robot is to reach some pre-defined goal. For this purpose, a sort of a co-operation between three main sub-modules is performed. These sub-modules consist in three elementary robot tasks: following a wall, avoiding an obstacle and running towards the goal. Each module acts as a Sugeno–Takagi fuzzy controller where the inputs are the different sensor information and the output corresponds to the orientation of the robot. The rule-base is generated by the controller after some learning process based on a neural architecture close to that used by Wang and Menger. This leads to adaptive neuro-fuzzy inference systems (ANFIS) (one for each module). The adaptive navigation system (ANFIS), based on integrated reactive-cognitive parts, learns and generates the required knowledge for achieving the desired task. However, the generated rule-base suffers from redundancy and abundance of data, most of which are less useful. This makes the assignment of a linguistic label to the associated variable difficult and sometimes counter-intuitive. Consequently, a simplification phase allowing elimination of redundancy is required. For this purpose, an algorithm based on the class of fuzzy c-means algorithm introduced by Bezdek and we have developed an inclusion structure. Experimental results confirm the meaningfulness of the elaborated methodology when dealing with navigation of a mobile robot in unknown, or partially unknown environment.

Key words: neuro-fuzzy, fuzzy c-means, navigation, mobile robotics.

1. Introduction

An ultimate goal of the research in a mobile robot systems is to develop control strategies and execution modules which support autonomous operations in an unknown or partially known environment. Several methods for controlling mobile robot system have been put forward. They can be subdivided into two main parts: global planning and local control. The former is based on the complete knowledge of the environment and the robot. It allows to generate collision-free paths, which are assumed to be executed correctly. The knowledge about the system and the environment is originated either from the modelling through a prior knowledge or

from the perception through a sensory system [1, 6, 13]. The second class consists of local control or behavioural strategies. It does not require a global map of the environment but only sensor information be used. The robot motion decision is made by considering the up to date status of the robot and the relationships with its environment. The main advantage consists in the ability to handle the changing aspect of the environment because the structural modelling of the environment is not necessary. Usually, the behavioural strategies are well suited for real time implementation, even if, it is known that they suffer from a "dead-lock" problem since the high level planification is no longer available [5]. This paper deals with the second approach where particular interest focuses in fuzzy control strategy. The latter provides a method of interpreting and implementing the linguistic rules pertaining to the process control strategy. Each rule defines a relationship between fuzzy input(s) and fuzzy output(s). A typical rule, for instance, is:

$$\text{If } x \text{ is } A \text{ and } y \text{ is } B, \text{ then } z \text{ is } C,$$

where x and y are linguistic variables representing the system conditions, z is a linguistic variable pertaining to a control action. A, B and C stand for linguistic values. The number of rules required for smooth operation depends on the system being controlled.

This kind of reasoning is quite close to a human based reasoning regarding his reaction in unknown situations and complex task execution.

The human behaviour in an unknown environment can be seen as a set of basic actions, named "Basic Virtual Guidance Activities (BVGA)", such as turn left, follow a wall, avoid an obstacle, etc. Human path planning and path tracking can then be modelled as symbolic knowledge of the universe using some abstract representations. This symbolic description defines a sequence of actions to be executed in order to move a nonholonomic vehicle such as mobile robot from its initial position to another pre-defined position [7]. Consequently, the on-line navigation system of a mobile robot adaptively determines the required action based on the integrated reactive-cognitive parts. The use of a suitable controller guarantees to achieve the desired behaviour. Different controllers are considered in the navigation system, which play the role of different BVGAs. This means that, they have to imitate the different human behaviours using some abstract representations. More specifically, three controllers are considered: wall following (in the left or right side), obstacle avoiding and running towards the goal. Usually, wall following serves as a tool for obstacle avoidance. This means that the latter looks like a decision-making procedure that sets off wall following mechanism. It seems then rational that when the task of the robot is to reach some target, the controller allowing to move towards the goal will be favoured while the two others permit to avoid unexpected circumstances.

Fuzzy systems as a natural framework for incorporating human knowledge are adopted. In a purely fuzzy system, the parameters do not appear in an analytical way, therefore, can not be applied for learning and tuning fuzzy rules. On the

other hand, neural networks can produce mapping rules from empirical training set through learning. But the mapping rules in the network are not visible and are difficult to understand. Combining both aspects may be useful. The fundamental concept of such hybrid system is to complement each other and thereby, create new approaches to solve problems. This gives rise to neuro-fuzzy systems, which have recently been investigated by many researchers, see, for instance, [4] and references therein.

The next section of this paper describes the general architecture of the controllers based on neuro-fuzzy system. The third section emphasizes the simplification of the rule base generated by previous controllers including inclusion based fuzzy c-means algorithm, estimation of consequent parts and handling conflictual situations. Finally some navigation examples, based on the neuro-fuzzy architecture ascribed to the three basic controllers, are pointed out.

2. Architecture of Each Controller

2.1. GENERAL SCHEME

As previously mentioned, the system consists in three main controllers, which permit to enable the subtasks of wall following, obstacle avoidance and running towards the goal. Basically, each controller has as inputs the ultrasonic measurements provided by robot sensors. Three measurements are considered d_1,d_2, and d_3 corresponding respectively to the frontal, left and right sensor with respect to robot configuration (see Figure 1). The output corresponds to the orientation θ of the controller with respect to the robot axis. Note that the velocity of the robot, when moving, is supposed to be constant.

A fusion methodology between these three controllers is elaborated such that moving towards the goal will be favoured up, except if an obstacle is encountered. In the latter case, the obstacle avoidance controller and, thereby, the wall following controller are fired until the robot finds, according to its sensory information, the existence of enough free space to launch the running towards the goal controller. Nevertheless, the methodology is based on some fusion approach based on neuro-fuzzy architecture in such a way the switch between the different controllers is not hard as it sounds from previous description but it rather fits a smooth description. Figure 2 summarizes the general scheme.

An Adaptive Neuro-Fuzzy Inference System (ANFIS) paradigm is used as architecture of each controller. This permits to adaptively determine the required action based on the integrated reactive-cognitive parts. The controller design procedure is divided into two main phases. In the first phase, an off-line learning experiment determines the membership functions and the rule-base. In this step, a backpropagation through time (BPTT) algorithm is developed to train each controller. Then a fuzzy clustering based on inclusion structure, which will be explained later on (see [10] for more information), is used to simplify the generated rule base. This simplification occurs in the sense that similar membership functions

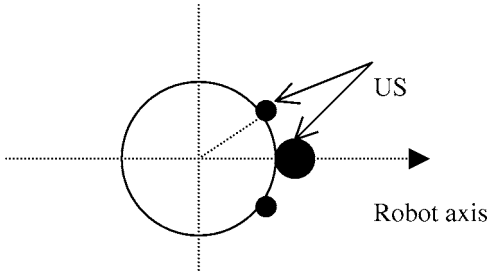


Figure 1. The employed Robot configuration.

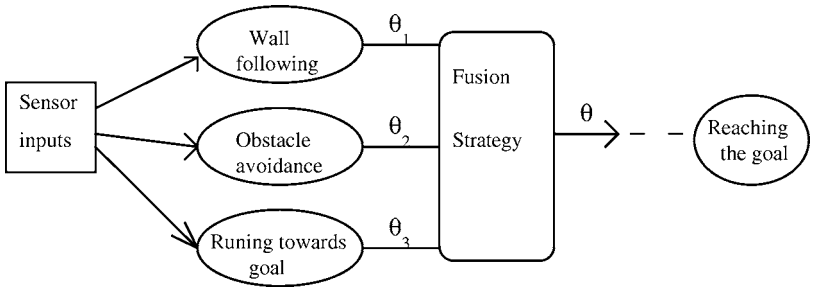


Figure 2. General scheme.

pertaining to the premises of the fuzzy rule-base are merged. This provides nonredundant membership functions to which meaningful linguistic labels are attached. Reduction of the total number of fuzzy sets is needed to constitute the model, which improves the semantic interpretation and reduces the demand in memory for implementation context. In the second phase, the controller parameters are optimized on-line. A Real Time Recurrent Learning (RTRL) algorithm is developed for on-line adapting and updating the membership functions and the rule-base of each controller [17].

2.2. ANFIS DESIGN AND CONFIGURATION METHODOLOGY

Takagi and Sugeno [11] have proposed a fuzzy system where consequent part(s) of the rules are a linear combination of the input variables. Identification methods are applied to estimate "THEN" part of the parameters. Based on the fuzzy logic controller paradigm, Wang and Mendel [14–16] have proposed a fuzzy logic system with Gaussian membership function:

$$\mu_{F_i^j}(x_i) = a_i^j \exp\left(\frac{-1}{2}\left(\frac{x_i - c_i^j}{\sigma_i^j}\right)\right), \tag{1}$$

where a_i^j, c_i^j and σ_i^j are respectively the maximum, the centre and the width values of the Gaussian membership function. The inference system uses the arithmetic product as t-norm and COG (center of area or center of gravity) method for de-

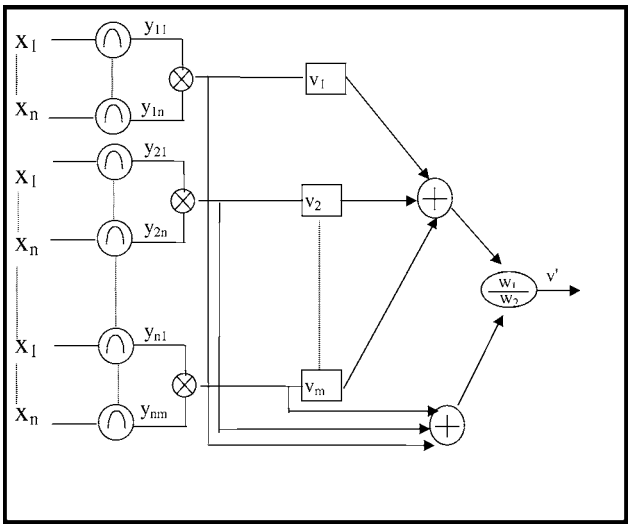


Figure 3. Adaptive Neuro Fuzzy Inference System (ANFIS) Block diagram.

fuzzification. This can be viewed in Figure 3 as a feedforward neural networks with three-layers: fuzzification layer, inference layer and defuzzification layer. The obtained fuzzy logic system is called Adaptive Neuro-Fuzzy Inference System (ANFIS) where the parameters appear in an analytical way. The output of this fuzzy system can be expressed as:

$$v' = \frac{\sum_j v_j \prod_i \mu_{F_i^j}(x_i)}{\sum_j \prod_i \mu_{F_i^j}(x_i)}, \tag{2}$$

where v_j is the point at which the linguistic value 'F^j' is maximum (i.e., $\mu_{F^j}(v_j) = 1$). The $\mu_{F_i^j}(x_i)$ is defined in Equation (1).

2.3. OFF-LINE TRAINING

For an unknown environment, the definition and the design of the desired motion trajectories may not be feasible and so training by using absolute position values may not be useful. The sensor signals provide only relative distance values and they correspond to evaluative information rather than instructive information. This can lead to the definition of a utility or an objective function that evaluates the performance of each controller. So, by using relative distance values, an objective function may be written as

$$J = \frac{1}{2} \sum_{k=1}^{N} \sum_{n=1}^{ns} \| S_{n,k} - S_{n,k}^d \|^2, \tag{3}$$

where the data are gathered during the off-line experiments. $S_{n.k}$, corresponding to the signal issued from the sensor number n at the sample time k, provides the

distance values from the robot location to an object in the environment. Notice that the relative orientation of the robot with respect to an object can also be obtained using like encoder sensors. Depending on the situation (which BVGA has to be learned), the $S^d_{n,k}$ is defined as the minimum or the desired distance of each sensor from the object. ns is the number of the sensors used in the sequence and N stands for the total number of time samples.

To obtain optimum parameters for each controller, the above objective function J has to be minimized after the controller produced a sequence of control actions. Notice that, the closed loop control of navigation system (Figure 3) is a Recurrent Adaptive Network (RAN). For a sequence of N control actions it is possible to duplicate all units N times and re-arrange the resulting network into a feedforward network. It is assumed that the explicit formulation of robot dynamics and the sensor models are available. The use of a sequence of control actions virtually converts the system to a discrete-time representation.

Now, the backpropagation learning algorithm can be applied to the transformed feedforward network. The backpropagation is a well-known learning algorithm and therefore is not discussed here in details. It tries to optimize (minimize) the desired objective function by calculating the derivative of utility summed across over time with respect to current actions. These derivatives are then used to adapt the network parameters according to steepest descent method:

$$\Delta P^{i,m} = -\eta \frac{\partial J}{\partial p^{i,m}}, \tag{4}$$

where $\Delta P = [\Delta p^1, \ldots, \Delta p^m, \ldots, \Delta p^M]$ represents a change in a parameter (matrix) $P = [p^1, \ldots, p^m, \ldots, p^M]$ at the end of a sequence of N time steps, $p^m = [v^m, c^m, \sigma^m]^T$, $i = 1, 2, 3$ and η is the learning rate. The backpropagation for transformed network is called the Back Propagation Through the Time (BPTT) [15]. In contrast, in Nguyen's and Widrow's approach [8], the objective function contains only a final error value

$$J = \frac{1}{2} \| S_N - S^d_N \|^2. \tag{5}$$

Here, the objective function uses all error values obtained from each action of the network, therefore, the use of plant emulator to calculate error at each time step is not necessary.

3. Simplification of the Base-Rule

3.1. MOTIVATION AND METHODOLOGY

The fuzzy rules obtained from BBTT tend to result in a control model with some redundancy. This redundancy often occurs in the form of similar membership functions in the premise of the resulting rule-base. When there are many similar fuzzy sets defined for a variable, it becomes difficult to attach qualitatively meaningful

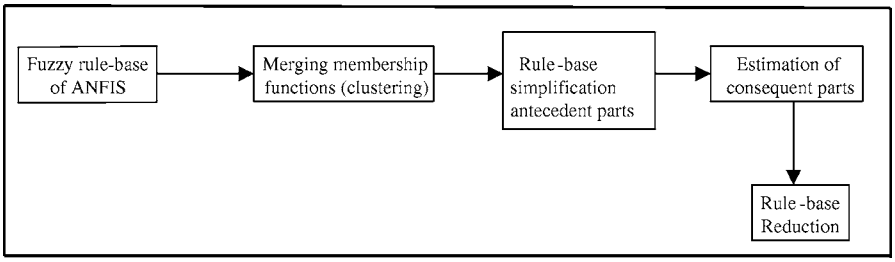


Figure 4. Block diagram of the fuzzy rule base reduction.

linguistic labels to the different membership functions. The high number of the membership functions makes difficult to grasp the meaning of the model and to handle the system. A semantically unclear model is not easily verified by an expert when verification is desired as a part of the model validation.

Indeed, after an off-line training phase, the obtained parameters can be interpreted as a fuzzy rule like:

$$\text{If } S_1 \text{is } A_{11} \text{ and } S_2 \text{ is } A_{12} \text{ and } S_3 \text{ is } A_{13} \text{ Then } v \text{ is } y_1,$$

where each sensory information is associated with a given universe of discourse, and each label A_{ij}, used in the fuzzy rule base, is characterised by two parameters m and σ (since the underlying fuzzy sets are Gaussian).

As it is clear from Figures 5a and 5b, there is a considerable overlapping among membership functions. Many of these functions are approximately similar and have a high degree of overlapping. This induces a redundancy in the rule-base and makes a linguistic description of the system difficult. Thus, it is desirable to merge similar membership functions in order to reduce the redundancy and improve the semantic interpretation of the rules. For this purpose, an approach based on fuzzy clustering method is used here which is described in the next section. Particularly, the inclusion-based structure seems to be in full agreement with expert based approach, which seeks for the meaningful inputs that match some linguistic representations. This permits, before real time implementation, to merge "almost" similar antecedent parts of the different rules and replaced by one common membership function. However, in order to achieve the simplification of the number of rules, as it is pointed out in Figure 4, an identification of the relevant and consistent consequent parts is required. This will be detailed later on.

3.2. FUZZY CLUSTERING ALGORITHM BASED ON INCLUSION CONCEPT

This part summarizes the ideas and results pointed out in [8].

Let us denote the Gaussian distribution by G and degree of inclusion of the Gaussian G_1 in G_2 by $\mathrm{Id}(G_1, G_2)$. One may, for instance, consider the idea that for the Gaussian distribution with parameters (m, σ), the 97% of the data are concentrated into the interval $[m - 3\sigma, m + 3\sigma]$. It is easy to see that if $m_2 - m_1$ is

less than $3(\sigma_1 + \sigma_2)$ then G_1 is more included in G_2 or equivalently, $\mathrm{Id}(G_1, G_2)$ is bigger. One can write:

$$\mathrm{Id}(G_i, G_j) = \begin{cases} |m_i - m_j| \exp(-3(\sigma_i + \sigma_j)) & \text{if } \sigma_i \leqslant \sigma_j, \\ \exp(1) & \text{if } \sigma_i \geqslant \sigma_j. \end{cases} \tag{6}$$

The fuzzy clustering problem for the rule-base obtained from off-line training phase can be written as a minimisation of:

$$\sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{\alpha} (x_i - v_j)^t A_j (x_i - v_j) \tag{7}$$

subject to:

$$\sum_{j=1}^{c} u_{ij} = 1 \quad \forall i = 1, 2, \ldots, n, \tag{8}$$

$$\forall j, \quad \sum_{i=1}^{n} (x_i - v_j)^t A_j (x_i - v_j) = \sum_{i=1}^{n} (|x_i^1 - v_j^1|^2) \exp[-9(x_i^2 + v_j^2)^2], \tag{9}$$

where j takes its values from 1 to c (number of clusters). x_i and v_j, each of which is defined by its pair (m, σ), stand respectively for the ith datum and the jth unknown prototype (center of the class). The constraint (9) means that, for each cluster, the amount of the inclusion value of the prototype v_j in each datum x_i could be supported by some distance structure considering the amount of distances of v_j to each x_i with respect to some positive definite matrix A_j. In other words, the constraint (9) makes a bridge between the inclusion structure and the distance structure (which is necessary for the fuzzy clustering formulation). Also, the resulting optimal cluster is in agreement with optimisation concept only in global sense, since it is concerned with the sum over all the available data.

The use of square in the right-hand side of (9) is justified by the square distance considered in the left right-hand side. Moreover, it provides a better reading in terms of matrix formulation. Indeed, (9) can be rewritten as

$$\forall j, \quad \sum_{i=1}^{n} (x_i - v_j)^t A_j (x_i - v_j) - \sum_{i=1}^{n} (x_i - v_j)^t B_1 (x_i - v_j) \times \\ \times \exp[-9(x_i + v_j)^t B_2 (x_i + v_j)] = 0, \tag{10}$$

where

$$B_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}; \qquad B_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

As a result of the optimization algorithm (see Appendix for more detail), the centers of the resulting classes are provided by the following

$$u_{ij} = \frac{1}{\sum_{s=1}^{c} \left[\frac{(x_i - v_j)^t A_j (x_i - v_j)}{(x_i - v_s)^t A_s (x_i - v_s)} \right]^{1/(\alpha - 1)}} \tag{11}$$

and by solving for v_j and β_j Equation (10), which, in the case of two-dimensional datum x_i, comes up with three non-linear equations, then the determination of the distance matrix A_j is given by the following

$$\sum_{i=1}^{n}(x_i - v_j)(x_i - v_j)^t(u_{ij}^{\alpha} + \beta_j) = 0, \tag{12}$$

$$A_j = \frac{\sum_{i=1}^{n} \exp(-9(x_i + v_j)^t B_2(x_i + v_j))}{\sum_{i=1}^{n}[\beta_j(x_i - v_j) + u_{ij}^{\alpha}(x_i - v_j)]} \times \times \frac{[B_1(x_i - v_j) - 9(x_i - v_j)^t B_1(x_i - v_j) B_2(x_i + v_j)]}{\sum_{i=1}^{n}[\beta_j(x_i - v_j) + u_{ij}^{\alpha}(x_i - v_j)]}. \tag{13}$$

The following algorithm summarizes the different steps leading to the identification of the prototypes (center of the classes) as well as the matrix U.

Step 1. Fix the number of clusters c, the defuziffier α, and initialize the matrix u using fuzzy c-means algorithm.

Step 2. Determine the prototype v and the parameter β by solving (12) using some numerical optimization method.

Step 3. Determining the matrix A using (13).

Step 4. Generate a new partition using (11).

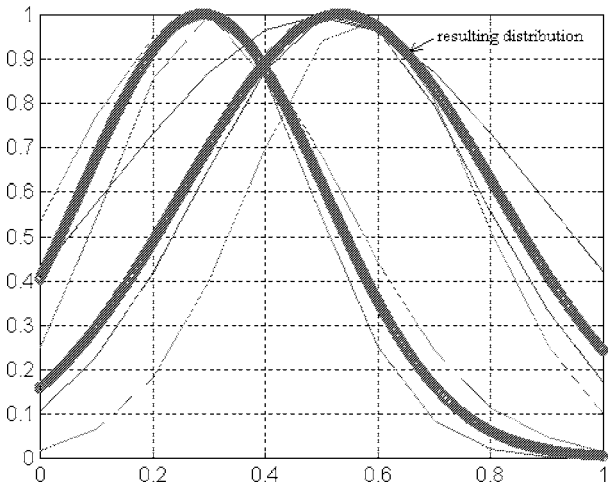
Step 5. If the cluster partition is stable, stop; else return to step 2.

3.3. RULE-BASE SIMPLIFICATION

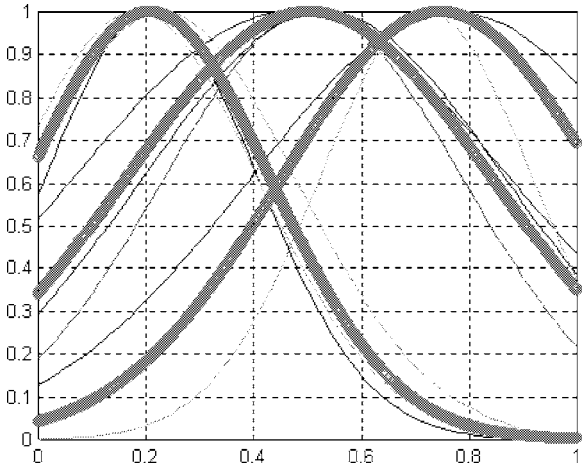
After generating the clusters using the above method, we attribute to each cluster a label and then put it in the rule-base. An example of such cluster is shown in Figure 5(a) and 5(b) with a thick line corresponding respectively to the case of two and three classes (clusters). Therefore, only these clusters are kept in the rule-base while the rest of membership functions are removed. Notice that the number of the clusters has to be specified beforehand.

3.3.1. *Estimation of Consequent Parts*

The mentioned clustering method permits only to identify the relevant inputs (clusters) in the sense of inclusion based criterion. In others words, this corresponds to the antecedent part(s) of the relevant rules. However, the consequent parts should be determined too. For this purpose, we compute the error between the output of each rule and the output of its simplified version (the class that best fits the datum in the sense that it ensures the largest value in the U matrix) i.e. $e = |y - y'|$, where $|\cdot|$ stands for the absolute value. This error is used to adapt the consequent part in a least squares sense, see Figure 6. After least squares estimation of the consequent part, if the error is important, which means that the number of cluster



(a)



(b)

Figure 5. Examples of clustering results.

used to represent the desired behaviour is not enough, then the number of clusters "*c*" must be increased and the whole procedure must be repeated again to obtain an acceptable error from the least squares estimation. This methodology can also be understood as an unsupervised learning approach, which permits to adjust the number of clusters (classes) that should be used in the described inclusion based fuzzy c-means algorithm. This contracts, with other approaches followed in the fuzzy literature where, usually, global criteria, ensuring sufficient separability of classes, are employed.

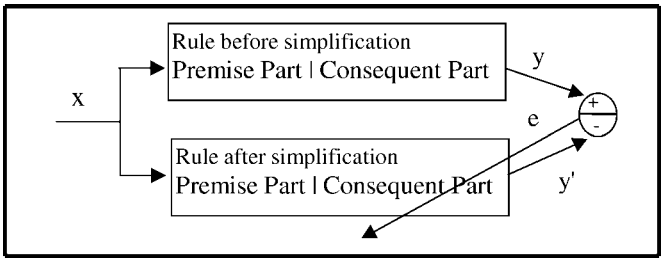


Figure 6. Estimation of consequent parts of the rules.

3.3.2. *Handling conflictual situations*

After simplification process, some inconsistent cases may arise. The inconsistency consists of the rules with the same "if parts" but with different consequences. This means that contradiction occurs in the rule base. In other words, this exhibits a nonoptimal identification of system parameters. Consequently, consequences have to be re-estimated according to the whole history of the results. The information gathered from the learning process will be re-examined according to the preceding. For this purpose, we propose the following heuristic approach based on the faithfulness index assigned to each rule. Namely, the larger index value the more consistent the underlying rule. Such methodology has also been employed by some authors, see, for instance, [3, 4]. Let us consider, for instance, two rules: rule 1 and rule 2, with n if parts and one output. For instance, we have for the rule 1:

If X_1 is A_{11} and X_{12} is A_{12}, and ... and X_{1n} is A_{1n} then Y is w^1, where A_{1i} stands for a Gaussian membership function with $(m_i^1 \sigma_i^1)$ as respectively the mean and the standard deviation. More formally, one may consider the vector representation (p^1 and p^2) for both rules defined as follows:

$$\begin{aligned} p^1 &= [(m_1^1, \sigma_1^1)(m_2^1, \sigma_2^1) \ldots (m_n^1, \sigma_n^1) w^1], \\ p^2 &= [(m_1^2, \sigma_1^2)(m_2^2, \sigma_2^2) \ldots (m_n^2, \sigma_n^2) w^2]. \end{aligned}$$

The parameter w^i is the output of the rule (consequent part).

Considering the T iterations done by the learning process, one may establish the following cost function J_k (for the kth rule):

$$J_k = \sum_{t=T-n}^{T} \prod_{i=1}^{n} \left[(m_{ij}^t - m_{ij}^{t-1})^2 + (\sigma_{ij}^t - \sigma_{ij}^{t-1})^2 + (w_{ij}^t - w_{ij}^{t-1})^2 \right]. \tag{14}$$

Here m_{ij}^t stands for the m (mean of the Gaussian) value pertaining to the jth rule associated to ith component of vector p^j obtained at the tth iteration of the learning process, while σ_{ij}^t stands similarly for the the standard deviation.

Finally the faithfulness index value is driven as

$$\alpha_1 = 1 - \frac{J_1}{\| J^T \|}, \qquad \alpha_2 = 1 - \frac{J_2}{\| J^T \|}, \tag{15}$$

where $\|J^T\|$ stands for a normalized factor over T iterations defined as $\|J^T\| = \max_k J_k$.

Thus, if $\alpha_1 > \alpha_2$, then the rule 1 will be selected since it is considered more faithful than the rule 2, while the rule 2 will be eliminated.

3.4. ON-LINE ADAPTATION

In a configuration phase, BPTT requires extensive computing resources for a sequence of control actions and can be only implemented off-line. However after the controllers are initialized and suitably configured in the first phase, a real-time updating of the parameters in each sample interval is required to guarantee a successful navigation task. Therefore, in the second phase, i.e., real time implementation phase, an on-line adaptation algorithm is used to optimize the performance of the mobile robot in an unknown environment.

The on-line training uses the Real Time Recurrent Learning (RTRL) algorithm to update parameters of the controllers [10, 17]. The RTRL is an approximation of the BPTT and instead of minimizing the objective function J at the end of a sequence, it minimizes the K ($k = 1, 2, \ldots, K$ and $K = 1, 2, \ldots, N$) step objective function J_K at each time step K, where:

$$J = \sum_{K}^{N} J_K \tag{16}$$

and

$$J_K = 1/2 \sum_{k=1}^{K} \sum_{n=1}^{ns} \|S_{n,k} - S_{n,k}^d\|^2. \tag{17}$$

This saves computation and memory requirements. At the beginning of the real time implementation phase, the network is initialized by the set of parameters obtained from BPTT.

4. Experimental Results

In order to train each controller a particular BVGA, a specific off-line experiment for configuration phase has been designed. At the beginning of the configuration phase, each controller (ANFIS) is initialized randomly with 36 basic rules and 36 membership functions. During this test, the ultrasonic sensors scan the environment with sampling time T and deliver data to the central processing unit. At each time step k ($k = 1, 2, \ldots, N$), the ANFIS generates a control action v_k and the robot moves to a new position. A sequence of $N = 17$ control actions (time intervals) is considered here to produce evaluative data. After this length of time (NT) is elapsed, or if a sensor provides an alarm signal that the robot will collide an object,

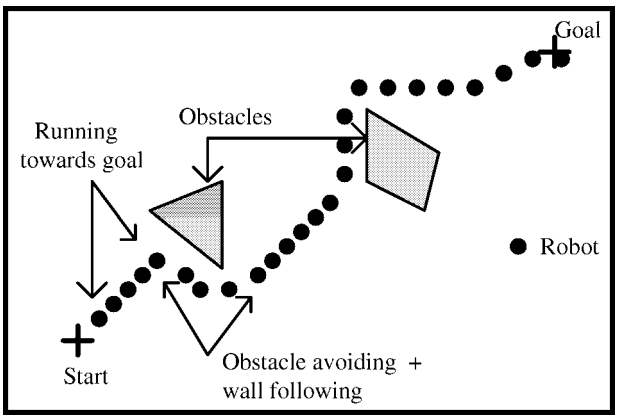


Figure 7. Illustration of the Cupertino the three sub-tasks: wall following, avoiding an obstacle, running towards the goal.

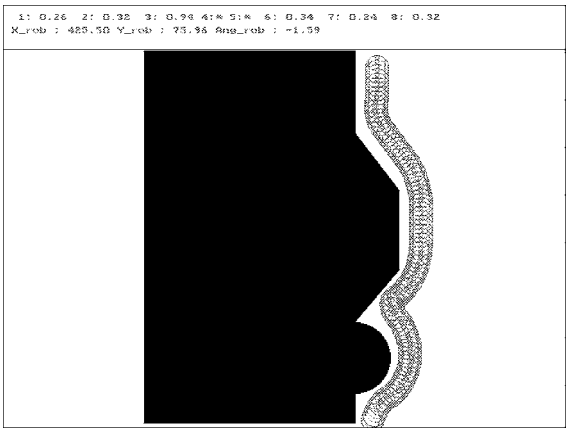
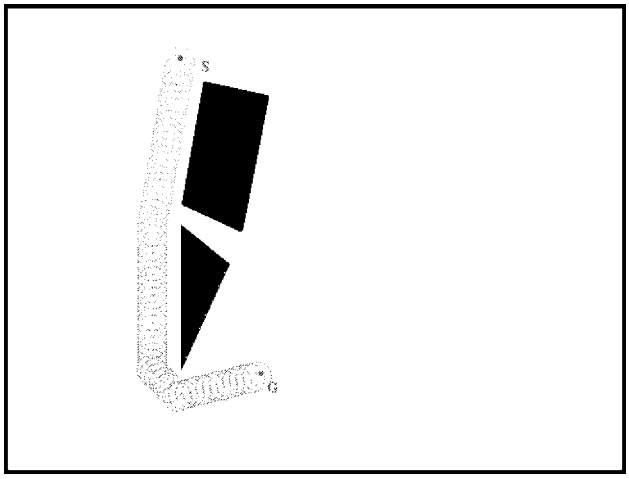


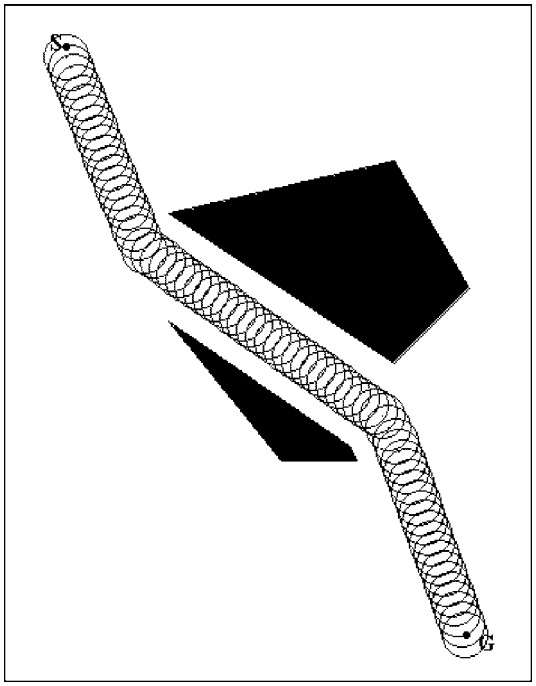
Figure 8. Typical example of wall following and obstacle avoiding.

the robot immediately stops and an off-line training of the controller then starts to update the ANFIS parameters.

Figure 7 shows a typical example of the co-operation between the three sub-tasks, that is, wall following, avoiding an obstacle and running towards the goal, in order to achieve the target. Typically, as soon as there is enough free space in the direction of the goal, the running towards the goal controller is fired. However, the notion of the free space is very restricted due to the limitations of the sensory information since only three ultrasonic sensors were used. Besides, the notion of free space is also more or less related to a relatively accurate position of the robot such that the location of the goal agrees with robot's knowledge. This induces sometimes, in practice, some conflictual situations. Basically, since the collision is always avoided by one of the robot sensors, the real impact of such conflictual situations, even in complex environments, are restricted to some oscillations or

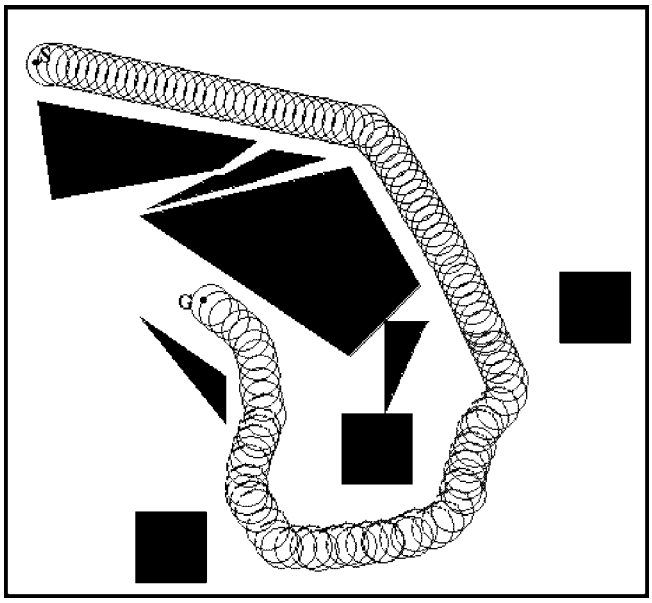


(a)



(b)

Figure 9. Examples of navigation process involving wall following and running towards the goal.



(c)

Figure 9. Continued.

instability in the robot trajectory, while the goal is always reached as it will be seen in the navigation examples.

In Figure 8 a particular example of wall following and avoiding obstacles is presented. At the beginning of the learning, the robot was very far from the wall but it tries to reduce its distance from the wall and finds the right orientation to follow the wall without any prior knowledge. Training continues and the robot achieves the desired distance after nearly 20 sequences and follows the wall successfully.

In the real time implementation phase, the robot has to navigate in an unknown environment, adapt itself without colliding with any obstacles and to reach the desired position. To achieve this, the robot inherits a simplified parameter set (using the above clustering method) from the previous phase as an initialization. It has to be improved further during the on-line adaptation using the RTRL algorithm. Notice that the wall chosen in this experiment is not like a straight line, however the robot can easily follow it without any collision. In Figure 9, we show some typical examples involving wall following and running towards the goal.

Particularly, Figure 9(c) illustrates a configuration where the robot is within a complex environment and reaches its goal successfully. Note that the path followed by the robot in this case is more or less very equivalent to that which will be possibly accomplished by the human in restricted circumstances. This restriction is understood in the sense that only three basic BVGAs modules are employed and, more importantly, the sensory information is quite poor. In other words, this can be compared to a navigation of a blind person using only his cane to get exterior

knowledge. Indeed, basically, the difficulty, encountered in handling complex situations, which may entail some conflictual cases in choosing a particular controller, is due to the poverty of the ultrasonic sensors. Because of the small number (only three sensors are used) and the multiple reflection phenomenon, which cheat the measurements.

Conclusion

Adaptive navigation of a mobile robot in an unknown environment by using many Adaptive Neuro-Fuzzy Inference Systems (ANFIS) is discussed in this paper. The adaptive navigation system (ANFIS) learns and generates the required knowledge for achieving desired goals using two different phases, i.e. the configuration phase and the real time implementation phase.

In the former phase, a self-learning procedure based on the BPTT is extended for an ANFIS. The BPTT generates and optimizes the rule-base without any prior knowledge about the behaviour that robot may exhibit. The robot started with 36 fuzzy rules and non-adjusted membership functions. The adaptation is performed for 17 control actions (time steps). In the ANFIS, linguistic terms related to each sensory information represent both the fuzziness in decision and sensory information. Unlike to approaches using purely neural networks, this methodology has an ability to accept, enhance and express the acquired knowledge from the human or training algorithm in the form of fuzzy inference rules. After learning, all obtained rules were studied and it was observed that some membership functions disappeared from the universe of discourse since they overlap to a large extent with other membership functions. A fuzzy clustering based on inclusion concept is used to simplify the rule-base and to remove useless membership functions and rules before real time implementation. Also, a procedure for dealing with conflictual situations as a matter of the consequent parts of the rule(s) is pointed out.

The RTRL has demonstrated the real-time adoption capability of the ANFIS to control the steering of a mobile robot to follow a wall and avoid obstacles after it inherits the set of parameters, learned in the configuration phase. Experience has shown that at end of first phase, the robot can not perfectly perform the desired behaviour in an unknown environment, when it was randomly initialized, due mainly to the poverty and sometimes conflictual information issued from the ultrasonic sensors. Finally, the experimental results confirm that the employed methodology works well for mobile robot navigation in an unknown environment. The navigation procedure seems very similar to human behaviour armed with only the three basic BVGAs, i.e., avoiding obstacle, following a wall and runing towards a goal, and with poor perception capability (like a blind person). Future works will deal with adaptive determination of the number of cluster needed to achieve the clustering process as an improvement of the procedure of the consequent parts of the rules.

Acknowledgement

This work is supported by LIIA Lab., Université Paris, Vitry sur Seine, France, which is gratefully acknowledged.

Appendix

By forming the augmented Lagrangian when gathering objective function and the related constraints, we have

$$\begin{aligned} J(\mu, v, \lambda, \beta) \;=\; & \sum_{i=1}^{n}\sum_{j=1}^{c} \mu_{ij}^{\alpha}(x_i - v_j)^t A_j (x_i - v_j) + \\ & + \sum_{i=1}^{n} \lambda_i \left(\sum_{j=1}^{c} \mu_{ij} - 1 \right) + \sum_{j=1}^{c} \beta_j \sum_{i=1}^{n} (x_i - v_j)^t A_j (x_i - v_j) - \\ & - \sum_{j=1}^{c} \beta_j \sum_{i=1}^{n} (x_i - v_j)^t B_1 (x_i - v_j) \times \\ & \times \exp\left[-9(x_i + v_j)^t B_2 (x_i + v_j)\right]. \end{aligned} \tag{A.1}$$

The necessary conditions for optimality are found by setting the gradient of J with respect to its parameters to zero, i.e.:

$$\frac{\partial J}{\partial \mu_{ij}} \;=\; \alpha \mu_{ij}^{\alpha-1} (x_i - v_j)^t A_j (x_i - v_j) + \lambda_i = 0, \tag{A.2}$$

$$\frac{\partial J}{\partial A_j} \;=\; \sum_{i=1}^{n} (x_i - v_j)^t (x_i - v_j) \mu_{ij}^{\alpha} + \beta_j \sum_{i=1}^{n} (x_i - v_j)(x_i - v_j)^t = 0, \tag{A.3}$$

$$\frac{\partial J}{\partial \lambda_i} \;=\; \sum_{j=1}^{c} \mu_{ij} - 1 = 0, \tag{A.4}$$

$$\begin{aligned} \frac{\partial J}{\partial v_j} \;=\; & \sum_{i=1}^{n} \mu_{ij}^{\alpha} A_j (x_i - v_j) + \beta_j \sum_{i=1}^{n} (-2A_j)(x_i - v_j) - \\ & - \beta_j \sum_{i=1}^{n} \exp\left(-9(x_i + v_j)^t B_2 (x_i + v_j)\right)\left[(-2) B_1 (x_i + v_j) + \right. \\ & \left. + 18 (x_i - v_j)^t B_1 (x_i - v_j) B_2 (x_i + v_j)\right] = 0. \end{aligned} \tag{A.5}$$

The Equations (A.2) and (A.4) are similar to those used in classical problem of fuzzy c-means clustering when the distance d is replaced by the matrix formulation [2]. Assuming $\alpha > 1$, leads to:

$$u_{ij} = \frac{1}{\sum_{s=1}^{c} \left[\frac{(x_i - v_j)^t A_j (x_i - v_j)}{(x_i - v_s)^t A_s (x_i - v_s)} \right]^{1/(\alpha-1)}} \tag{A.6}$$

and Equation (A.3) could be rewritten as

$$\sum_{i=1}^{n}(x_i - v_j)(x_i - v_j)^t(u_{ij}^{\alpha} + \beta_j) = 0. \tag{A.7}$$

Thus, if v's (and x's) are vectors with two components, Equation (A.7) is transformed into a system with three non-linear equations (four equations where two are identical) and then provides the values of v_j (v_j^1 and v_j^2) and β_j. The Equation (A.5) can be rewritten for determining the matrix A_j such that

$$\begin{aligned} A_j \;=\; & \frac{\sum_{i=1}^{n}\exp(-9(x_i + v_j)^t B_2(x_i + v_j))}{\sum_{i=1}^{n}[\beta_j(x_i - v_j) + u_{ij}^{\alpha}(x_i - v_j)]} \times \\ & \times \frac{[B_1(x_i - v_j) - 9(x_i - v_j)^t B_1(x_i - v_j)B_2(x_i + v_j)]}{\sum_{i=1}^{n}[\beta_j(x_i - v_j) + u_{ij}^{\alpha}(x_i - v_j)]}. \end{aligned} \tag{A.8}$$

References

1. Acosta, C. and Moras, R. G.: Path planning simulation for a mobile robot, *Computer and Industrial Engineering* **19** (1990), 346–350.
2. Bezdek, J.: *Pattern Recognition With Fuzzy Objective Function: Algorithms*, Plenum Press, 1981.
3. Chiu, S. L.: Fuzzy model identification based on cluster estimation, *J. Intelligent and Fuzzy Systems* **2** (1994), 267–278.
4. Jang, J. S. R., Sun, C. T., and Mizutani, E.: *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, 1997.
5. Latombe, J. C.: *Robot Motion Planning*, Kluwer Academic Publishers, 1991.
6. Laumond, J. P., Jacobs, P. E., Tiax, M., and Murray, R. M.: A motion planner for non-holonomic mobile robot, *IEEE Trans. Robotics Automat.* **10**(5) (1994), 577–593.
7. Merklinger, A.: Performance data of dead recording procedures for non-guided vehicles, in: *Proc. of the Int. Workshop on Information Processing in Autonomous Mobile Robots*, Munchen, Germany, 1991.
8. Nefti, S.: Approche neuro-floue pour la modelisation et la commande des systèmes non-linéaires multi-variables 'application à la robotique', PhD thesis, University of Paris XII, France, July 1999.
9. Nguyen, D. H. and Widrow, B.: Neural networks for self-learning control systems, *IEEE Control System Magazine* **10**(3) (1990), 18–23.
10. Rohwer, R.: The moving targets training algorithm, in: Touretzkey (ed), *Advances in Neural Information Processing Systems*, Vol. 2, Morgan Kaufmann, 1990.
11. Takagi, T. and Sugeno, M.: Derivation of fuzzy logic control rules from human operated control actions, in: *IFAC Symposium on fuzzy Information, Knowledge Representation and Decision Analysi*, France, 1990, pp. 55–60.
12. Thuilot, B. B., Andrea-Novel, and Micaelli, A.: Modelling and Feedback control of mobile robots equipped with several steering wheels, *IEEE Trans. Robotics Automat.* **12**(3) (1996), 375–390.
13. Wang, L. X. and Mendel, J. M.: Generating fuzzy rules by learning from examples, in: *Proc. of 6th IEEE International Symposium on Intelligent Control*, Washington D.C. 1991, pp. 263–268.
14. Wang, L. X. and Mendel, J. M.: Backpropagation fuzzy systems as non-linear dynamic system identifiers, in: *Proc. of IEEE International Conference on Fuzzy Systems*, 1992, pp. 1409–1418.

15. Wang, L. X. and Mendel, J. M.: Fuzzy basis functions, universal approximation, and orthogonal least squares learning, *IEEE Transaction Neural Networks* **3**(5) (1992), 807–814.
16. Wang, L. X. and Mendel, J. M.: Design and analysis of fuzzy identifiers of non-linear dynamic systems, *IEEE Transaction Automatic Control* **40**(1) (1995), 11–23.
17. Williams, R. J. and Zipser, D.: A learning algorithm for continually running fully recurrent neural network, *Neural Comput.* **1** (1989), 270–280.