Agent-Like Neurofuzzy Architectures for Mobile Robot Path Planning

S. N. RAPTIS AND S. G. TZAFESTAS

Intelligent Robotics and Automation Laboratory
National Technical University of Athens
Zografou 15773, Athens, GREECE

Abstract: *This paper provides a blueprint of a system architecture that attempts to decompose a complex behavior, that a control system is desired to exhibit, to combinations of simple independent sub-systems each one represented by an appropriate agent and having appropriate relevance under specific circumstances. The main issues of such an architecture are discussed and several possible choices available during its design and implementation are presented. As an illustration of this approach, the case of a system addressing the local path planning problem for an indoor mobile robot is presented.*

Keywords: *fuzzy systems, agents, basis functions, learning, mobile robots, path planning.*

Introduction

The proposed system is a problem-solving architecture based on the principle of *decomposition*. A complex problem is decomposed to simpler ones, an appropriate subsystem is created to meet each sub-problem's requirements, and then an overall system output is obtained by appropriately merging the partial outputs of these subsystems.

In a broader sense a similar idea stands behind all systems that explicitly or implicitly make use of *basis functions* and produce outputs as (linear) combinations of them. Examples of systems that explicitly define and use basis functions are the radial basis function neural networks (RBFNs) and the B-spline networks. Fuzzy systems implicitly define and use basis functions known as fuzzy basis functions (FBFs) [1, 2].

This analogy may be extended even to the cases of CMAC, Kawato's non-recurrent single-layer neural network, and hierarchical neural networks may be considered to be members loosely included in the above family, since they formulate some mappings at the first layers and use these as building blocks to locally approximate a desired overall function.

All these paradigms share many common properties and can be studied uniformly under the general class of *associative memory networks* (AMNs) [3]. Technically, the AMN class itself is considered to be a subclass of artificial neural networks.

Since there exist good arguments in favor of designing and building artificial systems in terms of human-like notions, we will take advantage of this throughout the following

discussion. So we may equivalently state the problem as follows: we wish to build a system that exhibits a certain *behavior*. To this end we try to decompose this behavior to a combination of simpler behaviors and design appropriate subsystems to exhibit them which we may call *behavioristic elements*. We provide the system with the means to efficiently mix such elements, i.e. with the ability of *behavior coordination*. Each element is independent and self-sufficient. These characteristics make it very similar to a concept widely used in both artificial intelligence and computer science: the *agent*.

So, up to now, we see that our subsystems may be regarded from various perspectives, e.g. as behavioristic elements, as basis functions, or as agents, all of which refer to the same characteristic: their ability to serve as building blocks for reducing a problem solution to appropriate combinations of partial solutions to sub-problems.

The analysis that follows is qualitative rather than strictly mathematical and aims at highlighting some key issues related to the properties of the decomposition as described above.

These concepts prove to be very efficient for the design of a mobile robot path planning system. Using linguistic fuzzy rules as the basic building block of the system a human-like navigation behavior may be directly hosted to the system to efficiently bootstrap it. Learning can then guarantee to fine-tune these rules to result in a path planning system of high quality with low design time and effort.

The Framework

Below follows a discussion on the various 'interpretations' of the basic system building blocks. Although these are equivalent in many of their details, they assign different meanings to these basic building blocks.

The Agent-Based Computing Perspective

Although the concept of an agent is widely used in both artificial intelligence and computer science, there exist no strict definition of what an agent really is [4]. Nevertheless, it is common that agents possess at least the following properties:

- *autonomy*: they are self-sufficient and do not need human or any other support;
- *reactivity*: agents have the means to perceive their environment;
- *pro-activeness*: agents exhibit a goal-directed behavior and not only feedback responses.

In [5], it is argued that a rational agent possess characteristics as *beliefs*, *situation likings* and *dislikings* relatively to its surrounding world as it perceives it. So the aim of a rational agent is to try to change the world to meet its likings. A detailed investigation of the concept of agents may be found in [4]. Although quite old, [6] still remains one of the important sources of information relatively to agents and agent architectures (societies).

Agent are often conceptualized by processes running concurrently under a UNIX-like environment. It is our belief that the self-sufficiency of agents is very efficiently captured by objects in an Object Oriented Programming context where inheritance, overloading, polymorphism, etc. may be beneficially used to provide the agents with all the required functionality [7].

This paper makes use of agent notions in the sense of independent intentional modules working in parallel and responding to their environment as they perceive it. The overall system architecture is then defined as a set of competing agents to each one of which an *activation level* is assigned which gives an indication of the *relevance* of the agent in a particular situation. The higher the activation level, the more the agent will influence the overall behavior of the system.

The Adaptive Fuzzy Controller Perspective

To fully define a fuzzy system, one needs to define four modules, namely the fuzzifier, the defuzzifier, the fuzzy rule base, and the fuzzy inference engine (Fig. 1). To extend the definition to the case of adaptive fuzzy systems, one should also include an adaptation algorithm. A description of fuzzy rule based systems may be found in any classical textbook (e.g. [8]). An investigation of adaptive fuzzy systems may be found at [2].

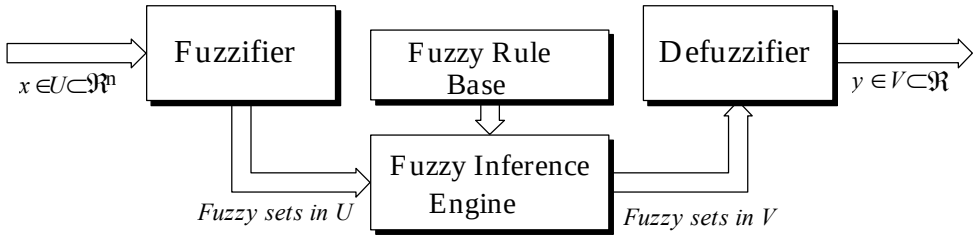

Figure 1. *The structure of a fuzzy system.*

Under the fuzzy system perspective, the role of the agent is conceptually undertaken by the fuzzy rules: a fuzzy rule is both autonomous and reactive. But from a mathematical point of view, the output of a fuzzy system is a combination of the so-called *fuzzy basis functions* (FBFs, [1, 2, 9]). The FBFs are indirectly defined during the design of the system and strongly depend on the specific choices made for the fuzzifier, defuzzifier, and inference rules, as well as the from of the membership functions of the fuzzy sets involved.

So the fuzzy rules themselves as incorporated in the rule base of a fuzzy system do not seem to qualify for agents. On the other hand, if we use a different scheme to represent fuzzy systems we may find that the basic properties required for agents are satisfied as long as we exclude the fuzzifier and defuzzifier modules (Fig. 2). Such a fuzzy system may be considered as a *fuzzy agent-based architecture*.

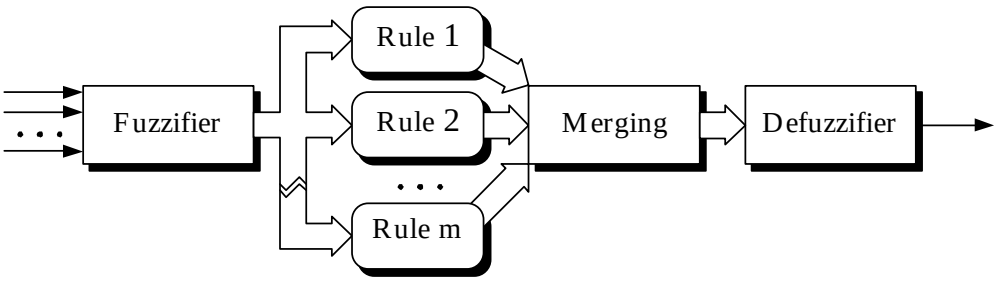

Figure 2. *A different representation of a fuzzy system.*

If one decides to include the fuzzifier and the defuzzifier in the analysis then the real agents are the FBFs rather than the fuzzy rules. For example, consider the case of a

fuzzy system with singleton fuzzifier, center average defuzzifier, product-inference rule and Gaussian membership functions. It is easy to shown [2] that the input-output relation of such a system is of the following form:

$$f(x) = \frac{\sum_{l=1}^{M} \bar{y}^l R_i^l}{\sum_{l=1}^{M} R_i^l} \text{ where } R_i^l = \prod_{i=1}^{n} a_i^l \exp\left[-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right]$$

So, the output is then expressed in terms of a linear combination of the fuzzy basis functions: $R_i^l / \sum_{l=1}^{M} R_i^l$. Note that equations very similar to these give input-output relations of radial basis function neural networks.

The Function Approximation Perspective

One of the most important uses of fuzzy logic systems is as models of nonlinear systems. Since fuzzy systems are capable of approximating a wide range of nonlinear systems they are qualified as models of general nonlinear systems.

Kolmogorov 's theorem roughly states that any nonlinear function may be approximated using a *linear* combination of 'simpler' nonlinear functions under certain conditions. So, assuming that the agents perform nonlinear mappings, agent-based architectures provide the flavor for building universal approximators.

However, it is crucial to obtain methods and techniques to efficiently select and train an adequate set of agents that can guarantee that the system's responses will be both complete for learned patterns (i.e. recalling may be of arbitrary precision), but also reasonable for novel inputs (i.e. the system should exhibit good generalization performance).

The Associative Memory Network Perspective

In [3] the associative memory networks (AMNs) which are basically a class of artificial neural networks are investigated. It is shown that fuzzy systems may also be viewed as members of the class of AMNs since the basic information processing principles are the same and under certain technical conditions the low level algorithms are also identical. Other members of the AMN class are the CMAC network, the B-spline network and the radial basis function networks.

One of the main characteristics that all members of the AMN class share is that their behavior strongly depends on a set of basis functions which are directly or indirectly defined during the design of the system. These basis functions may be considered as the mathematical equivalent to the concept of agents as described above.

The System

The System Architecture

Based on the discussion above, one may easily obtain an architecture similar to the one depicted in Fig. 3. This architecture includes all the issues discussed.

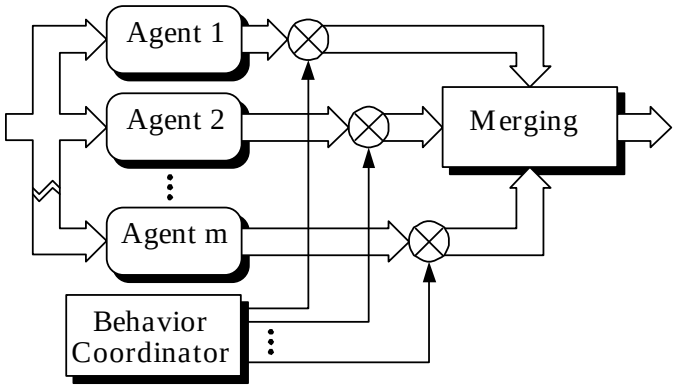

Figure 3. *The structure of the proposed agent-based system.*

The various modules of the system are described in the following.

The Agents

Agents are used as computational black boxes implementing nonlinear mappings from the input to the output domains. An agent may be implemented as algorithmic, neural, fuzzy or any kind of computational block as long as it performs a nonlinear mapping. The input and the outputs of the agent are denoted by thick lines to represent the fact that they may be real numbers as well as fuzzy sets, vectors, or others. Another important characteristic of the agents is that they are MISO mappings. There is no loss of generality since any MIMO mapping can be reduced to a number of independent MISO mappings.

It is important to note that the system may include agents that do not require any inputs. Their role is to capture fixed behaviors that the system needs exhibit. However, there is still dependence of the contribution of such agents to the final output through their degrees of relevance as assigned by the behavior coordinator.

The Behavior Coordinator

This module is responsible for calculating the relevance of each agent in a particular situation. Situations are distinguished on the basis of the inputs. All situations when the system accepts the same inputs are considered to be identical and are, in fact, indistinguishable. Clearly, each relevance is a scalar value that needs to act upon the output of the respective agent in a ‘multiplicative’ way so as to affect the agent's firing level. Usually such quantities are limited in the [0,1] interval. In many cases, the behavior coordinator may possess have some kind of memory in order to recognize more efficiently the current situation and make more ‘intelligent’ decisions relatively to the agents’ relevance degrees.

The Merging Module

All the agents' outputs, after being ‘multiplied’ by their respective relevance, are merged in an ‘additive’ way to produce the overall system output. This is the task of the merging module.

The above elements constitute the kernel of the system. When required, as in the case of fuzzy agents whose inputs and output are fuzzy sets, additional modules like fuzzifiers and defuzzifiers may be added externally to the kernel as shown in Fig. 4.

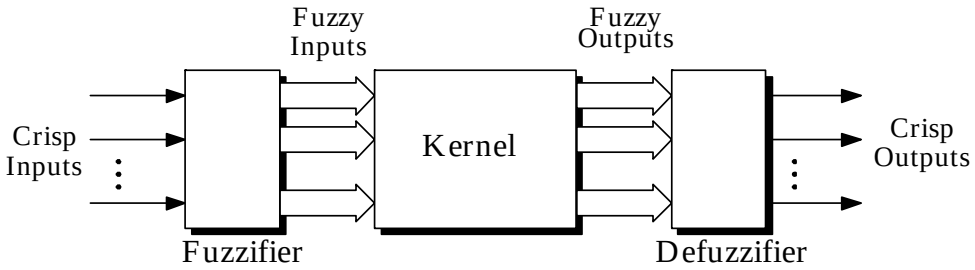

Figure 4. *Using additional modules.*

Training the System

The Back-propagation Algorithm

One of the most handy representation schemes of a system, an algorithm, or a process is that of a feedforward network. A very important reason to use such a representation is that there is already a training algorithm available for such networks: the *back-propagation algorithm* (BP). Although the BP algorithm has its roots in the field of neural networks, its basic concepts may be applied to any feedforward neural network.

What we need to have available in order to apply the BP algorithm is an expression for the error at the network's output layer, usually in the terms of the actual to desired output difference. This error is then propagated backwards, towards the input layer. Since the BP algorithm is a gradient descent algorithm, it is guaranteed to decrease the system's output error and to drive the system to a (local) minimum state.

Applying the BP algorithm to the network representation of the system is straightforward and provides the means to adjust all its model parameters. There are many choices for the selection of the system parameters to be adjusted. The first candidates for adjustment are, of course, the relevance of the agents.

Adjustable System Parameters

Up to now, we made no assumption concerning neither the way the agents (i.e. the rules of behavior) are implemented nor the behavior coordinator (i.e. the rules' relevance under the specific circumstances). With no loss of generality we may assume that the relevance degrees of the agents are implemented using neural networks whose generalization abilities prove to offer an important advantage. In most cases, multi layer perceptrons (MLPs) with a small number of hidden units are a fair choice. It should be pointed out that the problem of training the overall system is not just transferred to the one of training these MLPs. Remember that the system's inputs and output may be not only numbers but also fuzzy sets while relevance degrees and the MLPs to implement them are trained using solely numerals.

Although the implementation of the agents themselves may take place through various techniques, fuzzy logic seems to provide an excellent framework when a priori knowledge is to be inserted to the system or when knowledge is to be extracted by the system after the learning phase is completed. Fuzzy logic may easily incorporate human knowledge in linguistic form, filter the noise from the inputs, compensate for environmental uncertainties or sensor failures etc. So, *fuzzy agents* are usually a very efficient choice.

For the case of fuzzy agents, trainable system parameters are the parameters of the membership functions of the antecedent and decedent part of the rules. E.g. for Gaussian membership functions, these could be their center point, center value, spread, etc.

Training Equations

Assume that the error at the output layer of the system is calculated by an expression of the form:

$$e = \tfrac{1}{2}\left[f(\mathbf{x}) - d \right]^2$$

where: e is the error as measured at the output layer, $\mathbf{x}$ is the system's input vector, $f(\mathbf{x})$ denotes the actual system output, and d is the desired system output as provided by a supervising module.

A training rule for an adjustable system parameter, say p, will have the following form:

$$p(k+1) = p(k) - a\left.\frac{\partial e}{\partial p}\right|_k \text{, } k=0,1,2,...$$

where a is a constant stepsize representing the learning rate, and k=0,1,2,....

On-line learning

It is clear that the BP algorithm in the form considered above, is only applicable for off-line training where all the training samples are available and the derivatives are known. If this is not the case, *iterative learning algorithms* are in order to on-line adjust the parameters.

In the latter case, the actual derivative is replaced by an estimation while the error introduced by this approximation may be measured and treated as noise. The efficiency of such algorithms is quite satisfactory in most cases. Iterative training algorithms are discussed in [3].

Automatic Agent Formation

The performance of a system based on basis functions strongly depends on the choice of an adequate set of such functions. Similarly, in the design of an agent-based system, the selection of a set of agents is a crucial point.

There are many choices when training such agent-based systems of the form described above. One may choose to select a *constant* set of agents and perform training by adjusting the agents' relevance or one may wish to allow the agents themselves to change depending on the specific problem at hand. The former choice permits the user to insert an arbitrary set of agents, train the system, and then read back the relevance of each agent while the latter allows various conventional, statistical, and other training algorithms to be applied in order to refine the agents and to optimally fit the training samples. As examples of such training algorithms we may consider the probabilistic general regression, the orthogonal least squares, the nearest neighborhood clustering, etc.

So, basically, the following choices are available:

(i) constant set of agents;

(ii) set of agents initialized with linguistic knowledge and adjusted during training;

(iii) automatically created agents in order to optimally fit the numerical training data

In all the above cases, the agents' relevance are subject to adaptation. Moving from choice (i) to choice (iii) the approach from ‘completely intuitive’ becomes ‘completely mathematical’. It should be noted that agents may also be dynamically created and tested on-line by making use of genetic algorithms or other appropriate methods.

Exhaustive Agent Generation

Case-specific algorithms for determining the system's agents are also possible. For example, consider the case of a system with 2 inputs ($x1$ and $x2$) and 1 output (y), defined on the universes U, V, and W respectively. Assume that we use fuzzy agents and that we define 3 fuzzy variables on U (USMALL, UMEDIUM, and ULARGE), 3 on V (VSMALL, VMEDIUM, and VLARGE), and 2 on W (WSMALL and WLARGE). Then there exist 18 different possible fuzzy rules:

IF $x1$ is USMALL AND $x2$ is VSMALL THEN y is WSMALL
IF $x1$ is USMALL AND $x2$ is VSMALL THEN y is WLARGE
. . .
IF $x1$ is ULARGE AND $x2$ is VLARGE THEN y is WLARGE

A possible training algorithm for such a system could invoke the following steps:

- exhaustively formulate all the possible fuzzy rules;
- adjust the system's parameters using the training samples and any of the algorithms mentioned above;
- purge the rules whose relevance degrees achieved the lowest values throughout their universes of discourse;
- re-train the system using only the remaining agents.

Case Study: A Path Planning System for Mobile Robots

To highlight some of the main points of the proposed agent-based system perspective, a local path planning system for the navigation of an indoor mobile robot in unknown environments will be addressed using such a system architecture. Parts of this system were presented in [10] and [11].

This problem is a very representative case of a category of problems where both linguistic and numerical data are available and must both be appropriately blended under the same system platform. Commonsense heuristic navigation rules which are easy to obtain may be used to efficiently bootstrap the system. These rules may then be fine-tuned using numerical training pairs from sensor data and desired robot motions so that the system’s behavior is refined.

Introduction

The design and implementation of truly autonomous mobile robots, i.e. robots that could act in abstractly defined unstructured environments exhibiting robust performance

and a certain degree of intelligence, have for long been the aim of much research in the fields of robotics and artificial intelligence. The research on this problem has, for many years, been divided in two major categories, namely global path planning and local path planning.

Global path planning makes use of some available a priori knowledge relative to the environment and the objects that consist it, in order to move the robot towards a target position. To this end, many methods have been proposed in the technical literature, which differ in the philosophy of the solving algorithm, the knowledge representation scheme, etc. Some of the most important methods are:

- the configuration space method [13], developed by Lozano-Perez [14] and other researchers [15],
- the generalized Voronoi diagrams [16]
- the methods of artificial intelligence [17], and
- lately a very interesting and promising approach: the artificial magnetic field methodology [18].

The common problem in all the above global path planning methods is the need of possessing full knowledge of the environment and the obstacles. In many cases this demand may not be satisfied. That is the reason why local path planning techniques, capable to deal with generally unknown environments, have been developed.

In *local path planning* the robot makes use of information obtained by various sensors in order to successfully move to the target position. Dividing the research work in the field of local path planning into categories is not a straightforward task. Considering the kind of sensors used, one can find algorithms that make use of cameras [21], simple distance measuring sensors [22], etc.

Quite popular in the field of obstacle avoidance are the hierarchical model [23], and lately Saridis' intelligent control scheme [24], often making use of fuzzy control methodology [25]. Although the hierarchical model aims at reducing the large complexity of path planning, problems arise due to the strict hierarchy and sequential nature of execution. The reason for this is that the complexity introduced by the identification tasks and the tasks that require intelligence, is not faced but only transferred to higher levels.

To the end of solving this problem, Brooks [26] combines asynchronous units together, to each one of which a different role is assigned. However, these units are not independent since they communicate to each other. In a recent work of Boem and Cho [27], a combination of two independent units is presented, the one of which has an obstacle-avoidance behavior and the other having a goal-seeking behavior. Combination of these two units (which do not communicate to each other), is achieved through a 'behavior-selector' which makes use of a bistable switching function to activate each unit.

The complex behavior required to lead a robot towards a target position can be reproduced by a combination of simpler independent 'behavioristic elements', e.g. heuristics of the form 'move towards the obstacles', 'move along the goal direction', 'avoid the obstacles that move to your direction', etc. Many such antagonistic behavioristic elements which are appropriate for different circumstances may be taken

into account and may be implemented and operate independently. Some of them make use of the sensor measurements while others do not. An appropriate combination of such elements may lead to a system that exhibits the desired overall behavior.

This 'behavior-based' design technique for both the control of dynamic systems [28] and for mobile-robot path planning [29, 30, 31], attracts increasingly more interest and an increasing number of related publications appear in the technical literature.

Problem Statement

The problem addressed here is *local path planning* for an indoor mobile robot. In local path planning, a robot equipped with sensors is requested to move from a starting position (source) to a target position (destination) avoiding any obstacles.

No assumptions are made relatively to the environment except that it is planar and it is considered to be completely unknown and uncertain. Since no knowledge of the environment is assumed path optimality cannot be guaranteed. Moreover, the system must also be capable of compensating for sensor imprecision and failures. The above characteristics, i.e. complexity, uncertainty, and imprecision, qualify fuzzy logic as good framework for the local path planning problem.

We assume that only the direction of the target is known at every step and not its exact coordinates. Furthermore, we assume that the robot has N distance sensors placed uniformly. This means that each sensor's beam is directed $360°/N$ degrees from its neighboring sensors. Assuming a body attached coordinate frame having its Ox axis coinciding with the direction of the target, the first sensor is placed on Ox. This technique was also successfully used in [11]. Figure 5 shows the directions of the beams in the case of 16 distance sensors (N=16).

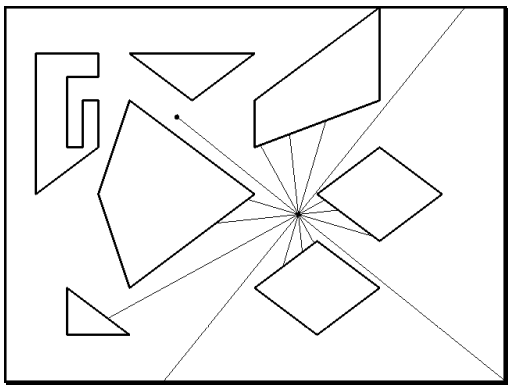

Figure 5. *The directions of the beams (N=16)*

Structure of the Proposed Model

Agents based on fuzzy logic are a fair choice for implementing the system's building blocks since:

- fuzzy logic provides the simplest way to translate heuristic rules to a computational algorithm,
- the system needs to deal with the uncertainty introduced by the sensor measurements,
- the domain of responsibility of each agent is by its nature fuzzy, and

- efficient algorithms exist to train fuzzy rules-based systems using numerical data [2, 19, 20].

The proposed model consists of such *n* agents connected to the sensors (i.e. their behavior depends on the specific circumstances) and *m* agents that do not depend on the inputs. Every agent produces an output independently from all the other agents. All these partial outputs are appropriately merged by the behavior coordinator. The sensor data is also fed to the behavior coordinator which may also have some kind of memory in order to recognize more efficiently the present situation. A threshold/selection module may be added after the merging module to ensure that a direction leading closer to an obstacle than a pre-specified value will certainly be rejected. This threshold value depends on the dimensions of the robot and the nature of the specific problem at hand. This unit works in a binary way: either a direction is safe or not, since collision is not a fuzzy concept!

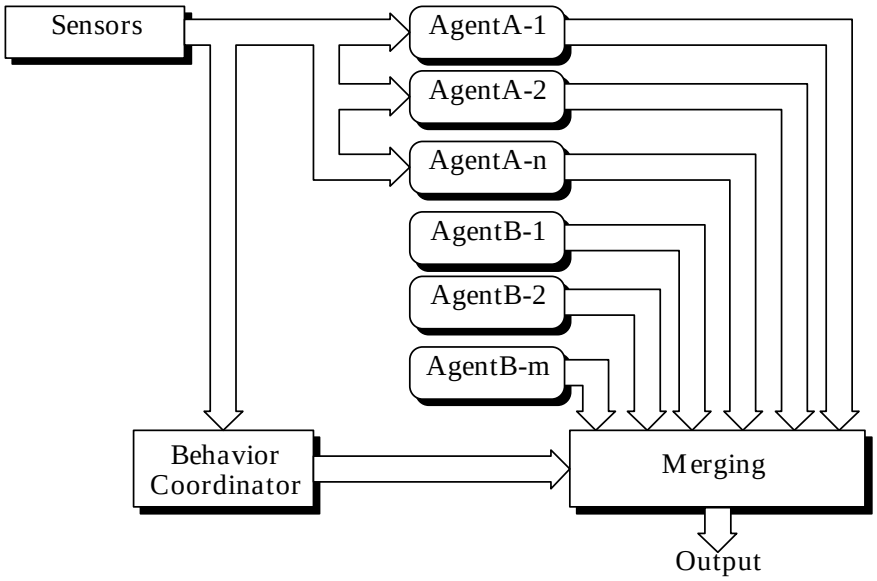

Figure 6. *General structure of the proposed model*

Assuming that the path planning system has *k* inputs (coming from the sensors) and *k* outputs only one of which is non-zero each time (indicating the direction to be followed), we can easily understand that this system has to implement a (very complex) function of *k* inputs and *k* outputs. Of course, if we additionally desire to control the velocity and/or the acceleration of the robot, more outputs would be required.

Under this agent-based perspective, it is attempted to divide the universe of discourse into subsets, and to implement the subsystems that approximate this function in every one of these subsets. Some difficulties arise during the determination of the subsets in which every subsystem is supposed to operate. This partially results from the fact that these subsets may be overlapping. As it will be shown in the next section, we use heuristic rules to determine the optimal domains of discourse of every agent. To this end, we will implement the behavior coordinator system based on fuzzy logic methods.

Our aim is to build a system capable of successfully driving the robot from the source to the destination and having two additional features:

- ability to host a priori knowledge in the form of human-like, heuristic, linguistic rules;
- ability to learn and refine its performance through adaptation.

The system of Fig. 6 possesses both these features and will be used as a base for solving the local path planning problem. The overall system architecture used for the path planning problem is given in Fig. 7.

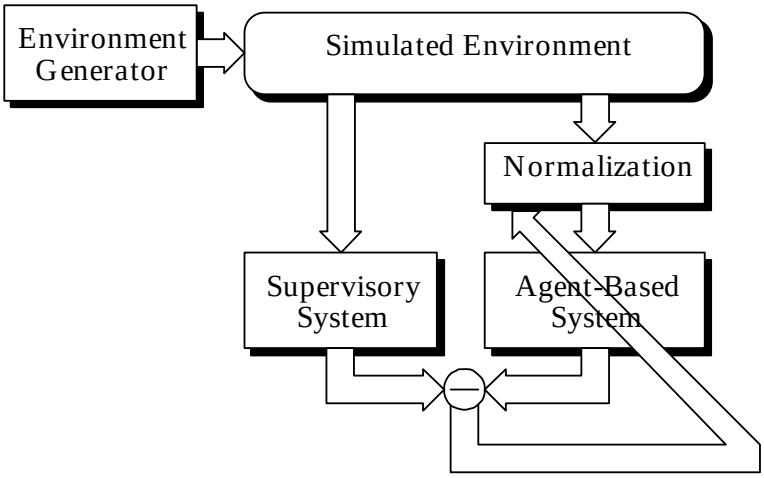

Figure 7. *The overall structure of the path planning system*

The overall system includes a supervisory system (SS). This system is assumed to have full knowledge of the environment during the learning phase and will be used as to supervise the learning procedure of the system in the sense of providing the necessary 'correct outputs' and error values needed from the back-propagation algorithm.

We will use the system described above in the following way: the system will accept as inputs the sensor measurements, denoted $s[i]$, and produce as outputs evaluations of the fitness of each direction of motion, denoted $d[j]$. The robot will eventually move along the direction receiving the highest fitness.

It is quite reasonable that the possible directions of motion should coincide with the directions of the sensor beams since we know nothing about the intermediate directions which could be occupied by obstacles.

The fitness of the i-th direction is expected to be depend the sensor measurements along that direction and, in some cases, on the measurements along some neighboring directions. For example an obstacle avoiding rule could be expressed as:

IF $s[i]$ is SMALL THEN $d[i]$ is SMALL

On the other hand, a rule that forces the robot to enter corridors or rooms while searching for the target, could be expressed as:

IF $s[i]$ is LARGE AND $s[i\text{-}1]$ is SMALL AND $s[i\text{+}1]$ is SMALL
THEN $d[i]$ is SMALL

Such rules are used to determine the fitness of each of the possible directions of motion and then the direction with highest fitness will be followed by the robot.

One question of main importance during the design of a fuzzy inference engine is the interpretation of the fuzzy IF-THEN rules. Such a rule is usually interpreted as a fuzzy implication, i.e. as a special kind of fuzzy relation defined on the Cartesian product of the input and output universes of discourse. But, there are various kinds of formulas used for fuzzy implication, e.g. fuzzy conjunction, fuzzy disjunction, generalized modus ponens (GMP), etc.

The choice of the appropriate interpretation of the fuzzy implication strongly affects the generalization behavior of the fuzzy rules, i.e. their response to unknown inputs. Different interpretations lead to fuzzy systems of different generalization properties, each one being appropriate for different problems. On the basis of these properties, fuzzy implication interpretations may be compared over certain intuitive criteria [12].

In our case, all rules should generalize in such a way that a rule of the type:

IF *x* is SMALL THEN *y* is SMALL

automatically implies:

IF *x* is MEDIUM THEN *y* is MEDIUM
IF *x* is LARGE THEN *y* is LARGE
IF *x* is VERY LARGE THEN *y* is VERY LARGE
etc.

Similarly, the rule:

IF *x* is SMALL THEN *y* is LARGE

should imply:

IF *x* is MEDIUM THEN *y* is MEDIUM
IF *x* is LARGE THEN *y* is SMALL
IF *x* is VERY LARGE THEN *y* is VERY SMALL
etc.

In [12] neural networks are used to guarantee that the desired behavior (expressed in the form of appropriate criteria to be satisfied) is exhibited by a fuzzy rule.

Using a Fixed Set of Agents

Any of the methods for the rule generation described above may be used for the path planning system. Thus, a constant set of agents may be created using linguistic knowledge in the form of fuzzy IF-THEN rules or rules may automatically be created based on sample input-output data and clustering or other techniques. In all cases, these rules may then be refined by a learning algorithm.

A fixed set of seven fuzzy and one ‘neural’ agent was first used do evaluate the performance of the resulting system through simulation.

The detailed form of the system for the case of a fixed set of agents is pictorially represented in Fig. 8.

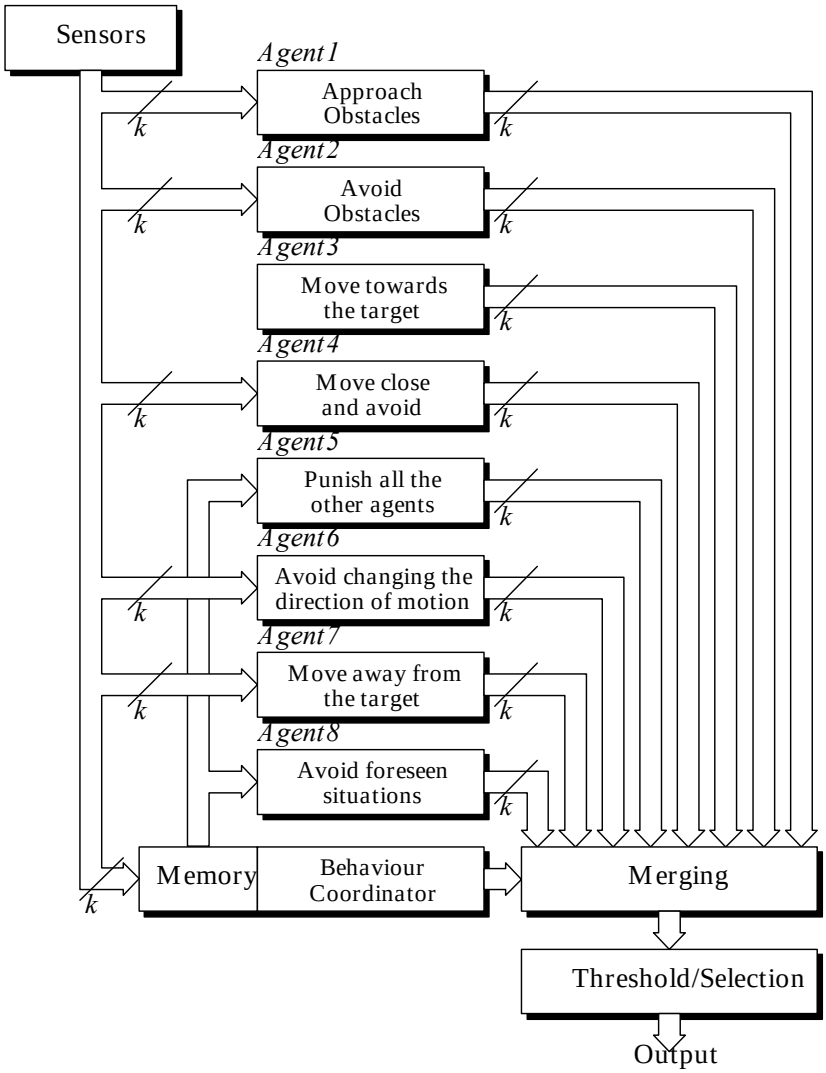

Figure 8. *The proposed path-planning system*

The output of the *k* sensors is fed to the appropriate agents (the input to the fifth and the eighth agent is the previous states from the memory), and every agent assigns *k* priorities, one for every direction. The output of every agent is then multiplied by a weight that characterizes its degree of relevance and is provided by the behavior coordinator. The result is finally brought to the merging subsystem, where the priorities supplied by all the agents for a specific direction are multiplied.

To conclude the algorithm, a step along the direction of maximum priority takes place, and the process is continuously repeated until the robot reaches the target position.

It is important to highlight that all the agents are functioning independently from each other and the only duty of the behavior selector is to assign a degree of relevance to each one of them. Thus, problem decomposition is taking place, since the original problem is divided into eight much simpler sub-problems, with the avoided complexity not being transferred to the task of recomposing the overall output from the partial outputs.

In the following operation of every agent will be analyzed along with the heuristic rules used to determine its domain of responsibility.

The *first agent* is a system that forces the robot to move close to the obstacles. This is necessary for two reasons. The first reason is that by getting close to the obstacles, the robot increases its resolution since it is easier to recognize small corridors or passes among the obstacles which may sometimes be the only way to reach the target position. The second reason is that this agent enables the robot to actually enter inside an identified corridor. This would have been avoided if no such rule existed since other directions would look more appealing. This agent is relevant for handling situations where the distances measured by the sensors are similar (i.e. small differences between them appear).

The *second agent* is a system that forces the robot to move away from the obstacles, i.e. it favors the directions along which long distances are measured. This is useful firstly because the further the robot moves from the obstacles, the safer the produced trajectory is. Moreover, this provides a way to optimize our trajectories. This agent is relevant for situations where the obstacles are moving fast or when the distances measured by the sensors differ a lot.

The *third agent* is a system that forces the robot to move to the direction of the target. Its usefulness is obvious since this direction, along with its neighboring ones, should be the most favorable. However, this does not always lead to desirable results, since when we are close to obstacles it is more important to avoid them than to move towards the target. So this agent is relevant for handling cases where there is enough space to move along the target direction and/or its neighbor directions.

The role of the *fourth agent* is to improve the trajectories with regard to the distance covered. Its logic is to avoid the obstacles while moving as close as possible to them. These directions can be identified by large differences between the distances measured along two successive directions. This agent is almost always relevant except of some situations that will be considered later.

The *fifth agent* is a subsystem that is activated periodically and checks if any progress has been made, i.e. if the current state relatively to the one evaluated during its last activation is ‘improved’. If no significant progress is observed, a ‘reaction’ process is initiated, the role of which is to suppress the action of all the other agents for a specific number of steps, and to lead the robot to the most unknown directions (i.e. directions leading to positions it has the least visited before). This agent is very important and has a global relevance since moving around with no progress is always undesirable and the agents that lead to this situation should be ‘punished’. Punishment has the meaning of ignoring, for some time, what these agents suggest. The unknown positions are identifiable by the help of an associative memory implemented using neural network concepts.

The *sixth agent* is a subsystem that helps the robot avoid to continuously change its direction of motion. If a direction is chosen, the robot shouldn't easily change it except, of course, if another direction appears to be much more promising. This agent is relevant for handling situations where the direction leading to the target (and its close neighbors,) are forbidden due to the presence of obstacles at small distances.

The *seventh agent* is implemented in such a way that it drives the robot away from the target! Its presence in the system is necessary since this may be the best thing to do under certain circumstances. This is the system's defense against getting trapped into a ‘local minimum’. Without the presence of such an agent, it would be difficult for the

robot to achieve this. The circumstances under which this agent is relevant, are the ones where the third agent is the least relevant, i.e. the complement of the relevance domain of the third agent.

The *eighth agent* is responsible for avoiding getting involved into foreseen situations. Obviously, these should be avoided since they are examined in the past and a path, if existed, would have already been found. This agent also helps the robot to avoid dead-ends caused by continuously looping around the same positions. To this end, the neural associative memory mentioned earlier is used. By adopting neural methods the system inherits the very important feature of generalization, i.e. not only the already visited situations but also their neighbors will be avoided (to a lesser, of course, degree).

This set of agents is not, of course, the only possible set one can think of. According to the specific problem at hand and the specific demands, we may add or remove agents from the system. The system is flexible enough to allow us to directly insert or delete agents in a straightforward manner. The way each agent is implemented, can be determined according to the role this agent is supposed to play in the system. Fuzzy logic provides a good solution when problems related to the accuracy of the sensor measurements occur. Neural networks, on the other hand, may enrich the system with learning/adaptation capabilities. Finally, simple algorithmic procedures may prove to be efficient enough under certain circumstances.

The behavior coordinator consists of eight different subparts, one for each agent, and provides to each one of them the heuristics needed to determine the domain of relevance of that agent.

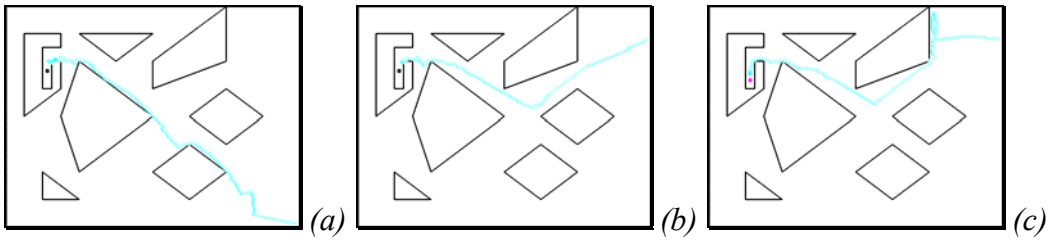

Figure 9. *Some representative paths*

Conclusions

A general design methodology was presented here which is based on the principle of decomposition, i.e. reduction of the problem's solution to the superposition of partial solutions. Partial solutions are obtained by local experts called agents which are abstractly defined basis elements sharing many common characteristics with other widely used basis functions which the agents eventually try to extend. Appropriate tools have been presented for both the design and adaptation of agent-based systems. The efficiency of such systems was demonstrated through the design of a path planner for indoor mobile robots. This system used a few fuzzy and a neural agent and proved efficient enough to produce collision-free paths in unknown and uncertain environments even for difficult mass-like cases.

References

[1] L. -X. Wang and J. M. Mendel, “Fuzzy Basis Function, Universal Approximation, and Orthogonal Least Squares Learning,” *IEEE Trans. Neural Networks*, Vol. 3, No. 5, pp. 807-814 (1992)

[2] L. -X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, Prentice Hall, Englewood Cliffs, N. J. (1994)

[3] M. Brown and C. Harris, *Neurofuzzy Adaptive Modeling and Control*, Prentice Hall, UK (1994)

[4] M. Wooldridge and N. R. Jennings, “Intelligent Agents: Theory and Practice,” *The Knowledge Engineering Review*, Vol. 10:2, pp. 115-152 (1995)

[5] J. L. Pollock, *Cognitive Carpentry: A Blueprint for How to Build a Person*, The MIT Press, London (1995)

[6] M. L. Minsky, *Society of Mind*, Simon & Schuster, New York (1986)

[7] G. Agha, P. Wegner, and A. Yonezawa (eds.), *Research Directions in Concurrent Object-Oriented Programming*, The MIT Press (1993)

[8] M. Jamshidi, N. Vadiee, and T. Ross, *Fuzzy Logic and Control: Software and Hardware Applications*, Prentice Hall, Englewood Cliffs, N. J. (1993)

[9] H. M. Kim and J. M. Mendel, “Fuzzy Basis Functions: Comparison with Other Basis Functions,” *IEEE Trans. Fuzzy Systems*, Vol. 3, No. 2 (1995)

[10] S. Raptis and S. Tzafestas, “A Fuzzy Controller with Self-Formatting Rules Supervised by an Expert System for Robot Path Planning in Unknown Environments,” *Proc. WAC’96: The First Intl. Symp. on Intelligent Automation and Control*, Oct. 6-8, Milan, Italy (1996)

[11] G. Stamou, S. Raptis, and S. Tzafestas, “An Agent-Like Architecture for Autonomous Robot Motion in Unknown Environment,” *The First ECPD Intl. Conf. on Advanced Robotics and Industrial Automation*, Sept. 6-8, Athens, Greece (1995)

[12] S. Tzafestas, S. Raptis, and G. Stamou, “A Flexible Neurofuzzy Cell Structure for General Fuzzy Inference,” *Mathematics and Computers in Simulation*, Vol. 41, Nos. 3-4, pp. 219-233 (1994)

[13] T. Lozano-Perez and M. A. Wesley, “An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles,” *Comm. ACM*, Vol. 22, No. 10 (1979)

[14] T. Lozano-Perez, “Spatial Planning: A Spatial Configuration Space Approach,” *IEEE Trans. Computers*, Vol. 32, No. 2 (1983)

[15] R. A. Brooks and T. Lozano-Perez, “A Subdivision Algorithm in Configuration Space for Findpath with Rotation,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 15, No. 2 (1985)

[16] O. Takahashi and R. J. Schilling, “Motion Planning in a Plane Using Generalized Voronoi Diagrams,” *IEEE Trans. Robotics and Automation*, Vol. 5, No. 2 (1989)

[17] S. Kambhampati and L. S. Davis, “Multiresolution Path Planning for Mobile Robots,” *IEEE Journal of Robotics and Automation*, Vol. 2, No. 3 (1986)

[18] C. T. Lin and C. S. G. Lee, “A Multi-Valued Boltzmann Machine,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 25, No. 4 (1995)

[19] H. R. Berenji and P. Khedhar, “Learning and Tuning Fuzzy Logic Controllers Through Reinforcements,” *IEEE Trans. Neural Networks*, Vol. 3, No. 5 (1992)

[20] C. T. Lin and C. S. G. Lee, “Reinforcement Structure/Parameter Learning for Neural-Network-Based Fuzzy Logic Control Systems,” *IEEE Trans. Fuzzy Systems*, Vol. 2, No. 1 (1994)

[21] W. -H. Tsai and Y. -C. Chen, “Adaptive Navigation of Automated Vehicles by Image Analysis Techniques,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 16, No. 5, pp. 730-740 (1986)

[22] R. Jarvis, “Distance Transform Based Path Planning for Robot Navigation,” in *Recent Trends in Mobile Robots* (Zheng, Y. F., ed.), World Scientific.

[23] K. Fujimura and H. Samet, “A Hierarchical Strategy for Path Planning Among Moving Obstacles,” *IEEE Trans. Robotics*, Vol. 5, No. 1 (1989)

[24] G. N. Saridis, “Intelligent Robotic Control,” *IEEE Trans. Automatic Control*, Vol. 28, No. 5 (1983)

[25] T. Sawaragi, K. Itoh, O. Katai, and S. Iwai, “Integration of Symbolic Path-Planning and Fuzzy Control for Intelligent Mobile Robot,” in *Fuzzy Logic* (R. Lowen and M. Roubens, eds.), Kluwer (1993)

[26] R. A. Brooks, “A Robust Layered Control System For A Mobile Robot,” *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1 (1986)

[27] H. R. Beom and H. S. Cho, “A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 25, No. 3 (1995)

[28] H. Berenji, Y. -Y. Chen, C. -C. Lee, J. -S. Jang, and S. Murugesan, “A Hierarchical Approach to Designing Approximate Reasoning-Based Controllers for Dynamical Physical Systems,” in Proc. 6th Conf. on Uncertainty in Artificial Intelligence, Cambridge, MA (1990)

[29] A. Saffiotti, E. H. Ruspini, and K. Konolige, “Using Fuzzy Logic for Mobile Robot Control,” in *International Handbook of Fuzzy Sets and Possibility Theory*, D. Dubois, H. Prade, and H. J. Ziemmerman (eds.), Kluwer, forthcoming in 1997.

[30] M. Colombetti, M. Dorigo, and G. Borghi, “Behavior Analysis and Training — A Methodology for Behavior Engineering,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 26, No. 3 (1996)

[31] E. Tunstel, “Mobile Robot Autonomy via Hierarchical Fuzzy Behavior Control,” in *Proc. 6th Intl. Symp. on Robotics and Manuf.*, 2nd World Automation Congress, Montpellier, France (1996)