MULTI-AGENT ROBOT ARCHITECTURES: THE DECOMPOSITION ISSUE AND A CASE STUDY

ELPIDA S. TZAFESTAS, SPYROS N. RAPTIS, and SPYROS G. TZAFESTAS

Intelligent Robotics and Automation Laboratory
Electrical and Computer Engineering Department
National Technical University of Athens
Zographou 15773, Greece.

ABSTRACT

In this paper, some fundamental issues of modern multi-agent robot architectures are discussed. It is argued that the multi-agent approach provides the necessary flexibility and adaptivity for such architectures, and that the primary issue in designing a multi-agent robot architecture is the selection of the granularity level, i.e., the decision on decomposing the overall desired functionality physically or across tasks. It is explained why at the various system levels different decomposition grains are needed; physical components, tasks or hybrid. This granularity decision is made on the basis of specific criteria of control localization, knowledge decoupling and interaction minimization so as to identify the decision points of the overall functionality. The above criteria lead to a dual composition-decomposition relation, which provides a good basis for system scaling. The paper specializes the discussion to a proposed neuro-fuzzy multi-agent architecture, which is then applied to design the local path planning system of an indoor mobile robot.

Keywords: Multi-agent robot systems; distributed artificial intelligence; robotic modeling; decomposition; neuro-fuzzy systems; fuzzy agents; neural agents; robot path planning.

1. Introduction

In the past, robots have been used in industrial contexts to perform tasks that necessitated high precision and accuracy, such as assembly, welding and other mechanical operations. During the last decade, however, research trends in industrial robotic systems shifted toward system integration, production flexibility, process adaptivity to unforeseen events, uncertainty handling and fault-tolerance [3, 7, 18, 27, 29, 34]. The concept of cellular manufacturing has emerged as a unifying paradigm that captures the need for product-oriented rather than process-oriented layout of the production plant [29]. On the other hand, the relatively recent field of service robotics [42], demonstrates the same needs for enhanced autonomy, behavioral adaptivity, flexibility and fault-tolerance, to cope with tasks such as indoor cleaning, supervision and inspection.

With modern robots being assigned increasingly more difficult tasks in decreasingly structured and certain environments, the demand for autonomy has therefore become more and more imperative. The traditional sense-model-plan-act paradigm can no longer cope with such demands necessitating a different, more robust design methodology. The key properties that a modern, relatively autonomous, robotic system must demonstrate seem to be flexibility and adaptivity.

Flexibility involves the possibility to reinitialize or reconfigure the system, or a component of it, on a new set of functional/operational demands, while adaptivity supports the ability to prevent or recover from failures of various types. Flexibility therefore concerns mainly the connection of a component to the other components of the same system, while adaptivity refers to its local self-sustenance [44]. For the case of a single-robot system, those principles apply to the robot's architecture, so that flexibility is the result of architectural modularity, whereas adaptivity corresponds to individual component completeness and plasticity. Note that the same objectives of flexibility and adaptivity hold for higher-order robotic systems, as well, such as multi-robot production plants or service groups.

The above have led to a reconsideration of the two fundamental assumptions of conventional robotic architectures. The first is the assumption of almost perfect knowledge, according to which all or almost all of the parameters that affect the operation and the performance of the robot are known in advance. For example, transport scheduling presupposes that the quantities, types, etc. of objects are known, as well as the number, positions and I/O flows of conveyor belts, temporary stockage platforms, bins etc. Moreover, this assumption states that, the outcome of any action, at any moment, is correct and predictable. The second is the centralized-decision assumption, according to which, and since there is almost perfect knowledge about the task, the designer of the system can derive a method of controlling it to meet the required operational goals, or else, make decisions in advance about the operation of the system.

The first assumption has been relaxed over the last decade since no complete information about the overall task is available in advance, or, if it is, using it to derive a perfect decision (such as a schedule or a plan) is often intractable. The optimization principle has therefore lost ground in favor of more qualitative and heuristic techniques that may manipulate less formal knowledge about the process. Those techniques are basically artificial intelligence techniques and their integration into robotic systems [1, 17, 31] has paralleled the emergence of the intelligent control field that combined conventional control methodologies with artificial intelligence techniques [22].

On the other hand, doubts on the centralized decision have only recently begun to arise and find their way into the scientific arena [10, 13, 28] (a marginally relevant effort has also been presented by Wang & Beni [33]). Some robotic tasks that involve multiple goals and subtasks may therefore be modeled in terms of concurrently acting cooperative or competitive components, usually called agents. In such systems, decisions have to be made separately at almost every point during operation of the system, since the exact overall state cannot be thoroughly known in advance and long-term planning/scheduling soon becomes obsolete. Local rules or heuristics at various points become more important and complicated analytical pre-processing loses value.

Introducing autonomous agents at those decision/redecision points — i.e. points where some local, intelligent, or for that purpose autonomous, "processing" is necessary —

looks beneficial. The advantageous consequences of such decomposition of the overall task into a number of interacting agents are *ease of design* (through complexity multiplication during synthesis) and *reconfigurability/scalability*.

In the next two sections we give a brief overview of distributed artificial intelligence and multi-agent systems and we discuss the suitability of multi-agent systems techniques for robotic architectures and more specifically for systems with high demands of flexibility and adaptivity. In section 4, we present a few examples and we argue that for different purposes we need different decomposition grains (physical, task-based or hybrid). We also argue that this granularity decision should be driven by the criteria of control localization, knowledge decoupling and interaction minimization, so as to identify the decision points of the overall task. Those criteria define a dual composition and decomposition relation that appears as a good substrate for architecture scaling. In section 5, the multi-agent concept is used 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 to a particular situation. The higher the activation level, the more the agent will influence the overall behavior of the system. In section 6, the principal properties of the multi-agent architecture are illustrated in a local path planning system of an indoors mobile robot in unknown environments. This is a very representative case of a class of problems where both numerical and linguistic data are available and must be suitably blended and used under a common platform [36, 37, 41]. Finally, concluding remarks are given in section 7.

2. Multi-Agent Systems And Distributed Artificial Intelligence

Distributed artificial intelligence [4, 15, 17] is the sub-field of artificial intelligence that is concerned with distributed rather than mono-agent systems and has been traditionally subdivided into distributed problem-solving (DPS) and multi-agent systems (MAS) [7, 8, 12]. The first domain emphasizes in solving a particular problem by distributing it among a number of modules (or nodes) that cooperate by sharing knowledge/solutions etc., while the second deals with coordinating intelligent behavior among a collection of (possibly preexisting) autonomous intelligent agents. This partition of the field is no longer valid, however, since traditionally multi-agent systems like actors and open systems have evolved into reactive (and intentional) multi-agent systems, so that it is now generally believed that conventional distributed artificial intelligence is a sub-domain of multi-agent systems. Another neighboring domain usually included in DAI, although it is a historical predecessor rather than a typical exemplar of it, is the blackboard systems [11, 16]. Blackboard systems have evolved from production rule systems to systems of large knowledge sources that communicate and cooperate via a shared structure (called a blackboard) on which they post pieces of work, hypotheses etc. The themes of DAI and MAS research include task description, decomposition, distribution and allocation, interaction and organization, coherence and coordination, modeling of other agents, inter-agent disparities, such as uncertainty and conflict etc.

First, let us look closer into the notion of “agent”. According to Ferber [12], an agent is an autonomous entity that is allowed to coexist with other agents with whom it communicates directly or indirectly while in pursuit of its local goals. The global goal is

achieved as a by-product of the concurrent operation of individual agents. According to this view, an agent is more than simply a component in a DAI system. What makes the difference is its autonomy: an agent is self-contained and autonomous in the sense that it tries to achieve its own local goals which are not necessarily directly correlated with the global one, or else with the "purpose" of the system. As its name denotes, an agent acts for somebody else's sake and on his/her behalf and is only considered an agent because an external, third observer may attribute to it some intentionality — which is the very essence of autonomy as argued in [6, 32]. Using the term "agent" also implies *a common representation level*. Since the global goal is to be achieved as a side-effect of the operation of individual agents, the definition of agents should be such, that interactions between them are represented in a uniform way in the system, using *a shared interaction medium*, although agents themselves are allowed to be heterogeneous. For example, in the case of closed producer-consumer system, what is important is the interaction between producers and consumers — production flow and regulation, saturation, bottlenecks etc. — and not the degree of sophistication of the internal structure of agents (actually, this degree may vary widely).

Now, why should we need to agentify some problems? Or, when do we need to introduce multiple agents, of what types and how many? The key appears to be the achievement of the overall goal as a by-product of the individual agents' operation. Some phenomena — or, in design terms, some processes — can be thought of more naturally as collections of individual simple interacting processes, rather than a single complex one. Since each of these sub-processes has its own "state space", the state of the overall system is "distributed" over the agents and the overall state space may grow exponentially on the number of sub-processes. However, and provided that the right representation and management tools exist, modeling the individual processes and their interactions provides a methodological advantage over mono-agent modeling: *complexity is hidden during analysis and multiplied during synthesis*. This allows, for instance, for better scaling by adding new agents or new interactions or both, and also gives room to deeper insight into observable complex phenomena that may be of scientific rather than engineering/technological value, such as the "emergent" cooperation of locally competitive agents. Multi-agent modeling constitutes therefore the complexity management tool for processes that involve multiple identifiable entities loosely-coupled through a shared interaction medium.

3. Multi-Agent Systems In Robotic Modeling

Robotic tasks by definition involve multiple goals, local objectives and background tasks and so can at first glance be considered as good candidates for multi-agent modeling. However, and according to the principles of complexity management and autonomy as observable intentionality, multi-agent modeling will only be profitably applicable, if the individual entities (or agents) considered are more than mere, "blind" input-output passive devices (or else if they appear intentional) and their interactions less static. As pointed out in the introduction, this is precisely the case of modern robotic tasks: decisions have to be made separately at almost every point during operation of the system, since the exact overall state cannot be thoroughly known in advance and long-term planning/scheduling soon becomes obsolete. Local rules or heuristics at various points become more important and complicated analytical pre-processing loses value.

Introducing agents at those decision/re-decision points looks beneficial.

As explained before, the consequences of decomposing the task into a number of interacting agents are *ease of design* (through complexity multiplication during synthesis) and *reconfigurability/scalability*. Introducing autonomous agents at individual decision points solves the adaptivity problem, since all events are processed independently of whether they are expected and to what degree. On the other side, scalability responds to the flexibility objective — local decision rules or agents may be modified almost independently of other agents and new decision points may be added relatively easily. From the above, it is clear that ease of design and reconfigurability are *coupled objectives* and a methodology that responds to the first one is also valid for the second and vice versa.

The fundamental principle behind robotic architectures modeling as multi-agent systems is to keep things as simple as possible, so as to better control them and be able to get rid of them when they go out-of-date. It is interactions between such simple agents that will account for observable complexity. This fundamental principle underlies the behavior-based robotic architectures [2, 5, 30], whose functionality emerges out of the simultaneous operation and interaction of many independent modules or agents, called "behaviors". The issues of modularity, reconfigurability and behavior combination in this context have been discussed in [21, 23, 43].

Various exemplar architectures have been proposed to model the desired decomposition in the field of autonomous robotics, but two have attracted the most interest and research. On the one hand, the so-called hierarchical architectures map different modules to different functionalities based on the various tasks the robot is involved into (e.g. world modeling, path planning, motion control, etc.) which are then divided into high level layers (model & plan) and low level layers (sense & execute) and act asynchronously. On the other hand, behavior-based architectures employ simple but complete units capable of exhibiting "elementary behavior" as their building blocks (e.g. moving towards a target, avoiding obstacles, following a corridor wall, etc.) with the sophisticated overall behavior emerging from the interaction and competition of such units. Hybrid techniques resulting from the combination of a hierarchical overall organization with a behavior-based decomposition of the execution level are gaining increasing interest.

Multi-agent modeling is therefore possible and beneficial for all those robotic tasks that impose heavy flexibility and adaptivity constraints and for which precise analytical description is generally problematic. The example in sections 5 and 6 refers to navigational control, which belongs to the above category.

4. Decomposition In Multi-Agent Systems

The notion of decomposition into simpler parts is rather old. It has been used to express a "complex" whole on the grounds of "simpler" superimposed constituents. Multi-agent systems provide the framework to port the decomposition principle from the combination of passive and isolated constituents to a "population" of active and interacting ("intelligent") entities.

The central issue of the design of a multi-agent robotic architecture is the choice of the granularity level, or else, the identification of the appropriate law of task decomposition

into agents. To comply with the principles of individual agent autonomy and minimization of interaction complexity, as well as with the objectives of process adaptivity and flexibility, the notion of decision points has been introduced. A *decision point* is a point in the process where some local intelligent, or for that purpose autonomous, "processing" is necessary to ensure validity of the undertaken action. This processing takes as input the current perceived state of the process (actually, since this state is distributed across all agents, it needs only consider the states of a few "relevant" or "adjacent" agents) and outputs/executes the appropriate action according to its internal rules/heuristics. Those rules and heuristics are in turn allowed to change throughout execution, or else the agent is allowed to be adaptive. This operation generally passes directly or indirectly control to an adjacent decision point. Furthermore, agents corresponding to different decision points are allowed to execute concurrently on an event-driven basis. What is considered as a decision point during analysis, is implemented as a separate agent during synthesis.

In the example of sections 5 and 6, we assume a multi-agent system designed for the navigation of an autonomous mobile robot, where each agent is responsible for a primitive functionality, such as obstacle avoidance, target following etc., and the exact environment of each agent at any moment can not be known in advance. Although the agents are organized in a single layer and their interactions are minimal, the behavior of the system resulting from their synthesis proves to be "sophisticated" enough to address the undertaken task successfully.

4.1. *Physical Decomposition*

One family of multi-agent robotic architectures assume a physical decomposition, i.e., one where each agent corresponds to a physical component of the robot: force sensors, sonars, wheel motors etc. Examples of this approach may be found in the earlier behavior-based literature, where different behaviors are, implicitly or explicitly, managing disjoint sets of sensors and actuators [2, 5]. The characteristic of these architectures is that they encompass a large number of sensors that may be functionally grouped, and those sensor groups are disjoint. This issue is discussed in [14]. This approach has been found useful whenever assessing the performance of a variety of physical sensor settings. Substantial improvements in the performance of the robot have been found to translate to minimal and smooth changes in the robots' physical apparatus, so that the system can be said to scale easily on the number of sensors. This approach is therefore beneficial when we can include dedicated sensors for each "functional" specification and does not necessitate any classification of perceptions.

4.2. *Task-Based Decomposition*

A second more modern family of multi-agent robotic architectures assume a task-based decomposition, i.e., one where each agent corresponds to a task or functional component of the robot: explore, follow fellow robots, recharge batteries etc. Typical examples of this approach are the behavioral systems that aim to reproduce on a simulated or real robot the behavioral sequences of a real animal (for instance, [38, 39, 40]). Other examples of the same approach may be found in the explorer robots literature [23, 31]. The important feature of this approach is that all arbitration between simultaneously active behaviors (or tasks) may be performed at the "motivational" level, i.e. using a

uniform measure of preference for each task, usually called the motivation or the activation level of the task. Each task uses as resources some of the robot's physical components, i.e., sensors and actuators, and only a few of these components exist, that are shared among tasks.

This approach is suggested whenever two architectural conditions hold:

- Passing control and data from task to task is straightforward, if necessary, i.e., it does not necessitate too complex decisions, and
- Reconfiguring the overall architecture involves adding/removing tasks, while the sensors and actuators configuration does not change significantly.

Besides, this architectural arrangement allows for evaluation of architectures at the task or operational level, for instance, if we want to assess the performance of various exploration algorithms with respect to certain environmental conditions.

4.3. *Hybrid Decomposition*

Finally, a third family of multi-agent robotic architectures assume a hybrid decomposition, i.e., one where each agent may correspond to a functional task or a physical component group of the robot (sonars, recharge batteries etc.) and may be recursively composed by other agents of a lower level. Examples of this hybrid and hierarchical approach may be found in [25, 19]. Rather than reasoning on the nature of "structures" and "functions", this approach emphasizes on interactions between agents and attempts to decompose so that they are naturally "represented" in that granularity level. For example, a sonar sensing system may be included in a localization agent where it is used passively or considered an agent by itself if its role is to emit a critical situation warning, such as a dead-end or a collision. In [43], we called this principle the incrementality principle, and we argued that this is the vehicle that will allow a designer to pass forth and back from agent specifications to operational objectives at the architectural level.

4.4. *Decomposition Criteria*

Looking closer at the above examples, it becomes obvious that what drives the decomposition (physical, task-based or hybrid) is the nature of the correspondence between physical and functional components. For a simple correspondence of the type {*many sensors* $\Rightarrow$ *one task*}, or {*one sensor* $\Rightarrow$ *many tasks*}, it is straightforward to use either a task-based or a physical-based decomposition. When this correspondence is complex {*many sensors* $\Rightarrow$ *many tasks*}, it is preferable to decompose in a hybrid way. Often enough, it is also beneficial to decompose at multiple levels, i.e., hierarchically, if that simplifies the description and the design of the various agents and the interactions between them.

In all cases, the decomposition depends primarily on a correct identification of the decision points of the overall robotic task(s). In this sense, the usual parallelisms between physical and structural decomposition and between task-based and functional

decomposition become irrelevant and displaced: actually, tasks are not less structural and more functional components than sensors or actuators. In either case, the task or the physical component is the right *structural* or else *representational grain*, i.e., the one that allow the multi-agent system to exhibit ease of design and reconfigurability, as explained in section 2. Note that behind the choice of such a structural grain lies an implicit operational rule: that of maximizing individual agents' activity by distributing the overall process over a set of agents capable and "willing" to execute concurrently. Another important observation is that speaking about decision points and structural grains allows for more natural integration of hybrid representations at different levels of the architecture. Decomposition is therefore driven by the following criteria:

- *Control localization*. Identification and isolation of decision points means identification of all those entities that need some sophisticated self-control, where sophistication is in comparison with the entities' connections (interactions) with other entities; an example is a critical situation detection module.

- *Knowledge decoupling*. A decision point is one where some sort of "knowledge" is concentrated. Identification of individual decision points implies that their local "knowledge bases" are orthogonal or disjoint or, at least, minimally overlapping; an example is the decoupling between a localization and an obstacle avoidance module.

- *Interaction minimization*. The above two criteria imply that decomposition should be such that interactions between agents be minimized: we want agents to work most of the time and interact/communicate only occasionally and not vice versa. From the above, it also follows that the knowledge directly or indirectly communicated from agent to agent is of limited type and extent — in some architectures, such as the one in [2], there is no direct communication between agents, but only indirect through the world.

Note finally that the choice of the right decomposition grain for a system is not independent of the various representations used. For instance, an environmental property that can be immediately sensed, may be fed directly to a number of interested agents, whereas if it needs to be *deduced* by other sensed properties, a special property-detection agent becomes necessary.

4.5. *Duality between Decomposition and Composition*

The previous decomposition criteria may be applicable at many directions. However, applicability alone does not ensure suitability to specific contexts. What is also necessary is the choice of the particular direction of application and this is driven by the specific reconfigurability needs: task-based decomposition generally scales on the number of tasks, while physical decomposition scales on the number of physical components, as already stated. Hybrid decomposition may scale on the number of any of the different agent types. In all cases, scaling is only possible if interactions are well understood and explicitly modeled. Given the flexibility and adaptivity needs of the robotic architectures, it becomes therefore obvious that decomposition should be such as to allow re-composition or reconfiguration later on and the exact direction of reconfiguration is specific-process-dependent. Moreover, such a principled decomposition or design allows a better understanding of the resulting system and a better control of its operation, so that

not only the reconfiguration will be possible, but it will be prompted if necessary by the performance of the system itself. As a consequence, decomposition based on the three above criteria and a set of reconfiguration directions becomes *an inversible relation*.

5. A Neuro-Fuzzy Multi-Agent System

Based on the conversation from the previous sections, we may summarize some indicative functional characteristics that qualify a module as an agent:

- concurrently acting self-contained and autonomous/intentionality entities
- pursuit local goals
- able of taking distributed decisions on the grounds of local rules or heuristics
- allowed to communicate directly or indirectly in a cooperative or competitive fashion

On the other hand, every multi-agent system should, at least, provide the means for:

- agent communication
- agent coordination
- conflict resolution

In this section we provide a blueprint for a multi-agent system consisting of neural and fuzzy agents. First we motivate the use of fuzzy- and neural-based agents as general purpose building blocks and then we present some details on the overall system architecture to conclude with some considerations relatively to its training and on-line adaptation. The next section will specialize the system to address the path planning problem.

5.1. *Fuzzy and Neural Agents*

Relaxing, for a moment, the strict AI framework respected up to now, we may consider a fuzzy rule itself to be a rough, primitive form of an agent; it encapsulates a fragment of (often heuristic) knowledge which allows it to take local decisions. Moreover, fuzzy reasoning provides a well established framework to control collections of rules for inference and decision support purposes. Even in the case that a single fuzzy rule cannot capture the complex behavior that an agent needs to exhibit for some applications, a set of such rules can provide the required expressiveness. Furthermore, there exist good arguments in adopting fuzzy logic for the underlying implementation of agents:

- 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,
- efficient algorithms exist to train and adapt fuzzy rule-based systems using numerical data.

An important characteristic of an agent based on fuzzy techniques, henceforth called fuzzy agent, is its ability to automatically provide a measure of its “applicability” at the specific system state as it perceives it by means of its so-called activation level. The activation level is a measure of the consistency of a fuzzy rule premise with the currently observed situation.

On the other hand, neural networks are well established as associative memory modules

capable of formulating nonlinear input-output mappings. Through their inherent generalization capabilities they may perform inference on the basis of exemplar training patterns. If a required behavior cannot be described in a structured manner —e.g. via causal production rules, or an algorithmic formalization, etc.— but only on the basis of stimuli-response pairs, neural networks are an excellent candidate for the underlying agent implementation.

5.2. *The Proposed Architecture*

A single layer multi-agent system that can host fuzzy, neural, algorithmic, or any other kind of agents, has the form shown in Fig. 1. Thick arrows were used to stress the fact that the data circulating into the system may vary significantly from scalar values to fuzzy sets, vectors, etc.

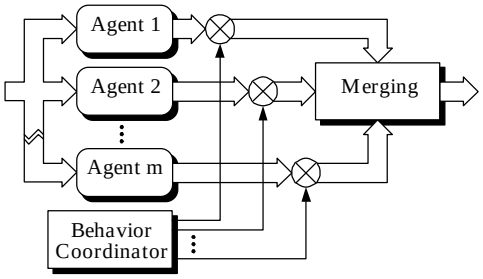


Fig. 1. The structure of the agent-based system

The main task of the behavior coordinator and the merging module is to resolve possible conflicts among the different agent outputs stemming from inconsistency in their design, graceful competition among the agents, or any other reason. To this end, the behavior coordinator can assign degrees of relevance to the output of each rule according to the current system state and the state of the world as perceived.

One thing that is not obvious from the figure, is that not all agents are required to make use of sensory data; some may use only information stored on an internal shared structured (e.g. a memory) and some may use no information at all capturing some fixed, constant behavior that the system needs to exhibit. However, the contribution of all agents to the output is indirectly affected by the current system state via the degrees of relevance assigned by the behavior coordinator. Finally the behavior coordinator itself is permitted to make use of the internal shared structured.

5.3. *System Training*

0The 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.

1The 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.

2Training Equations

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

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

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:

$$p(k+1) = p(k) - a\frac{\partial e}{\partial p}\bigg|_k$$

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

3On-line Adaptation

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.

4Automatic 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:

- constant set of agents;
- set of agents initialized with linguistic knowledge and adjusted during training;
- 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.

5Exhaustive 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.

6. A Case Study: Mobile Robot Path Planning

To highlight some of the main points of the proposed agent-based architecture, a local path planning system for the navigation of an indoor mobile robot in unknown environments will be addressed using such a system. Parts of this system were presented in a different context in [45, 46, 47].

6.1. *General Issues*

The research on path planning has been traditionally divided in two major categories, namely global path planning and local path planning.

Global path planning makes use of some a priori knowledge relating to the environment and the objects that consist it, in order to move the robot towards a target position. Possessing prior knowledge, the task actually translates into finding an optimal path, where "optimal" translates into minimization of a predefined criterion such as travel time, path length, path smoothness, etc. To this end, many methods have been proposed in the technical literature, which differ in the approach, the knowledge representation scheme etc., some of the most important being:

- the configuration space method [48], developed by Lozano-Perez [49] and other researchers [50],
- the generalized Voronoi diagrams [51]
- the methods of artificial intelligence [52], and
- the artificial magnetic field methodology [53].

The demand of optimality, necessitates full and precise a priori knowledge of the environment; a condition that cannot always be true or even realistic. That is the reason why local path planning techniques, capable to deal with generally unknown environments, have emerged.

In *local path planning* the system minimally relies on a priori knowledge: its main source of data is a set of sensors. 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 [54], simple distance measuring sensors [55], etc. Quite popular in the field of obstacle avoidance are the hierarchical model [56], and lately Saridis' intelligent control scheme [57], often making use of fuzzy control methodology [58]. Brooks [59] 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 work of Boem and Cho [60], 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 [61] and for mobile-robot path planning [62, 63], attracts increasingly more interest and an increasing number of related publications appear in the technical literature.

6.2. *Problem Formulation*

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; 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. Fig. 2 shows the directions of the beams in the case of 16 distance sensors (N=16). Common problems arising from the "blindness" of the robot between the sensors' beams, are faced by restricting its motion, at each point, only along the direction of any of the beams.

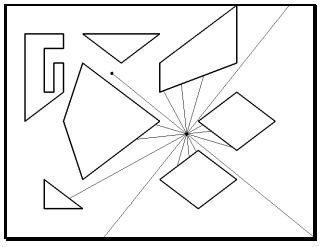


Fig. 2. The directions of the beams (N=16)

6.3. *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 [64, 65, 66].

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!

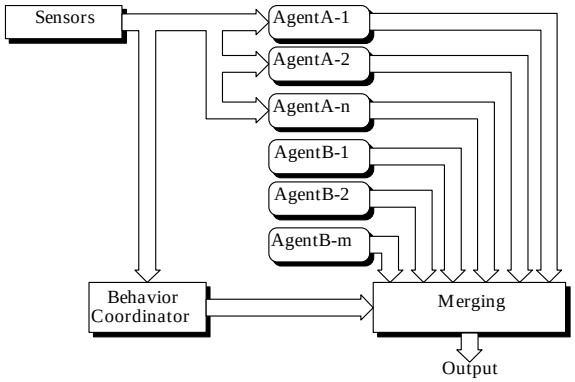


Fig. 3. General structure of the proposed model

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 navigational knowledge in the form of human-like, heuristic, linguistic rules;

- ability to learn and refine its performance through adaptation.

The system of Fig. 3 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. 4.

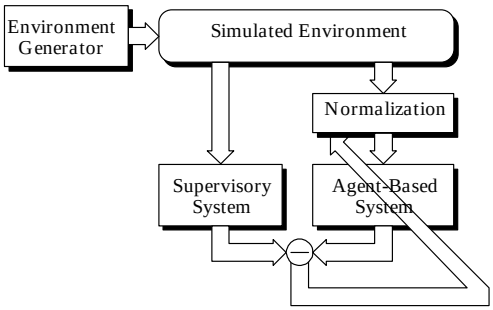


Fig. 4. The overall structure of the path planning system

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 to supervise the learning procedure of the system in the sense of providing the 'correct outputs' and error values required by 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 $s[i-1]$ is SMALL AND $s[i+1]$ 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 [47, 64].

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 [47] 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.

6.4. *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 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. 5.

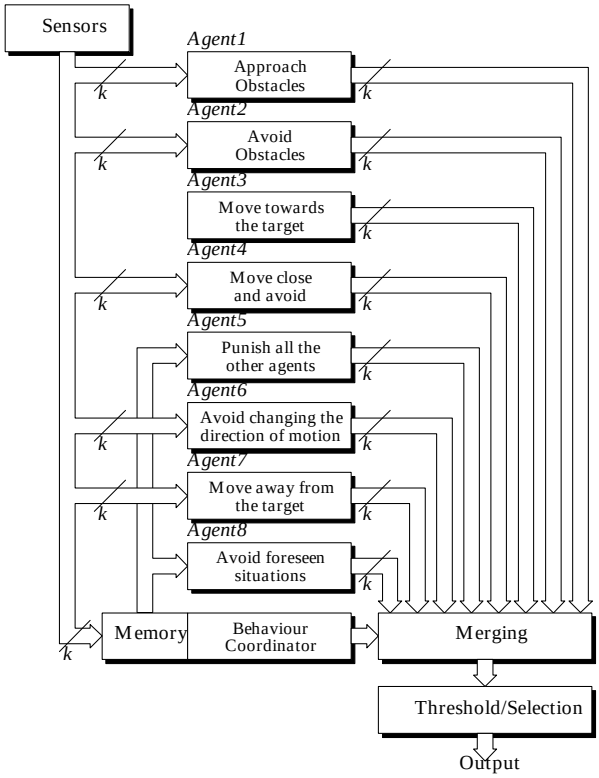


Fig. 5. The proposed path-planning system

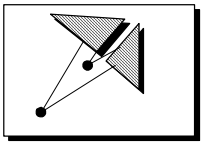
The output of the *k* sensors is fed to the appropriate agents (the input to the fifth and the eighth agent is the previous states from the memory), and every agent assigns *k* priorities, one for every direction. The output of every agent is then multiplied by a weight that characterizes its degree 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.

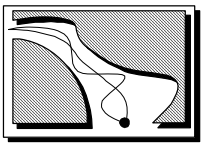
In the following operation of every agent will be analyzed along with the heuristic rules used to determine its domain of responsibility.

Agent 1: Approach Obstacles



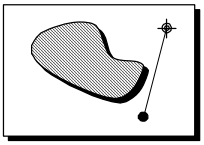
This agent forces the robot to move close to the obstacles. This is necessary for two reasons. The first reason is that by getting close to the obstacles, the robot increases its resolution since it is easier to recognize small corridors or passes among the obstacles which may sometimes be the only way 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).

Agent 2: Avoid Obstacles



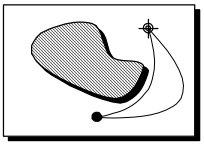
The second agent forces the robot to move away from the obstacles, i.e. it favors the directions along which long distances are measured. Such a behavior is useful since the further the robot moves from the obstacles, the safer the produced trajectory is. Moreover, this provides a way to optimize our trajectories in the sense of following smoother, less perturbed paths. This agent is relevant for situations where the obstacles are moving fast or when the distances measured by the sensors differ a lot.

Agent 3: Move Towards the Target



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.

Agent 4: Move Close and Avoid



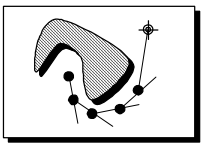
The role of this agent is to improve the trajectories with regard to the distance covered. Its logic is to avoid the obstacles while moving as close as possible to them. These directions can be identified by large differences between the distances measured along two successive directions. This agent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

Agent 5: Punish All the Other Agents

The fifth agent is actually 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

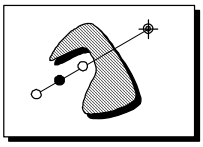
agents that lead to this situation should be ‘punished’. Punishment has the meaning of ignoring, for some time, what these agents suggest. The unknown positions are identifiable by the help of an associative memory implemented using neural network concepts.

Agent 6: Avoid Changing the Direction of Motion



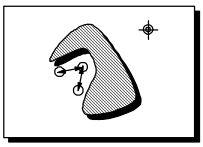
This agent plays the role of a ‘momentum’ term by helping the robot to avoid continuously changing its direction of motion. If a direction is chosen, the robot shouldn't easily change it except, of course, if another direction appears to be much more promising. This agent is relevant for handling situations where the direction leading to the target (and its close neighbors), are forbidden due to the presence of obstacles at small distances. When the robot initiates a maneuver to overcome an obstacle standing along its way to the target (which might temporarily drive it away from the target), it should keep up with it for a sufficient number of steps. In such a situation, combined actions of this agent and agent 7 are required.

Agent 7: Move Away from the Target



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.

Agent 8: Avoid Foreseen Situations



The *eighth agent* is responsible for avoiding getting involved into foreseen situations. Obviously, these should be avoided since they have been 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). Path planning into a drastically changing environment (where a large number of continuously moving obstacles exist, e.g. doors opening and closing, corridors being blocked by people, etc.) this agent should be suppressed since past situations should be reexamined.

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.

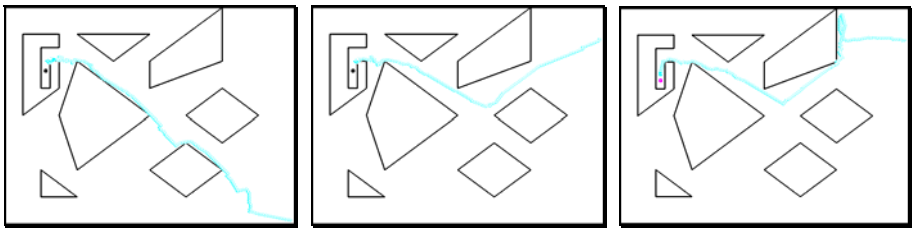


Fig. 6. Some representative paths

7. Concluding Remarks

It has been argued that multi-agent systems can respond to the flexibility and adaptivity objectives of modern robotic tasks by relying on individual agent autonomy and multiplication of complexity during synthesis/design. Designing a multi-agent robotic architecture corresponds therefore to identifying the appropriate decision points of the process in question and choosing the right structural decomposition grain. Several types of process decomposition into agents (task-based, physical or hybrid) have been examined and it has been found that generally decomposition should be driven by the criteria of control localization, knowledge decoupling and interaction minimization, applied across a set of reconfiguration and scaling directions. This way, decomposition becomes a reversible relation and is based on *the golden law of decomposition*: we decompose in a way that makes small changes have big effects or else makes the system scaleable. The decomposition principle was employed to address the problem of local path planning in unknown and uncertain environments: a general neurofuzzy multi-agent architecture was introduced and then specialized for the task at hand. A combination of a neural agent and a small number of fuzzy agents, has been shown adequate to produce collision-free paths for the mobile robot. The emergent collective abilities of the system were proven to greatly surpass a simple enumeration of each agent's separate abilities; a fact which is the essence of the decomposition principle.

References

[1] R. Almgren, On Knowledge-Based Planning and Programming Systems for Flexible Automatic Assembly, University of Linköping, Studies in Science and Technology, Thesis no. 176, LiU-Tek-Lic-1989/16.

[2] R. C. Arkin, Motor-Schema Based Mobile Robot Navigation, *International Journal of Robotics Research*, 8(4):92-112, 1989.

[3] K. E. Baldwin (1989), "*Autonomous manufacturing systems*", Proceedings of the 1989 IEEE International Symposium on Intelligent Control, Albany, NY, September 1989, pp. 214-220.

[4] A. H. Bond and L. Gasser (Eds.), *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, 1988.

[5] R. A. Brooks, A Robust Layered Control System for a Mobile Robot, *IEEE Journal of Robotics and Automation*, Volume RA-2, Number 1, 1986, pp. 14-23.

[6] A. A. Covrigaru and R. K. Lindsay, "Deterministic Autonomous Systems", *AI Magazine*, 12(3):110-117, Fall 1991.

[7] Y. Demazeau and J. -P. Müller (Eds.), *Decentralized A. I., Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '89)*, Elsevier/North-Holland, 1990.

[8] Y. Demazeau and J. -P. Müller (Eds.), *Decentralized A.I. 2, Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '90)*, North-Holland, 1991.

[9] A. A. Desrochers and M. Silva (Eds.), *IEEE Transactions on Robotics and Automation, Special Issue on Computer Integrated Manufacturing*, 10(2), April 1994.

[10] K. L. Doty and R. E. Van Aken, "Swarm Robot Materials Handling Paradigm for a Manufacturing Workcell", *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993, pp. 778-782.

[11] R. Engelmore and T. Morgan (Eds.), *Blackboard Systems*, Addison-Wesley, Reading, MA, 1988.

[12] J. Ferber, *Des Objets aux Agents: Une Étude des Structures de Représentation et des Communications en Intelligence Artificielle*, Thèse d'État, Université Pierre et Marie Curie, Paris, June 1989.

[13] K. Fischer, "Modelling Autonomous Systems in a Flexible Manufacturing System", *Proceedings of the IJCAI '93 Workshop on Dynamically Interacting Robots*, Chambéry, France, August 1993, pp. 109-116.

[14] A. M. Flynn and R. A. Brooks (1989), Battling Reality, *MIT AIM-1148*, October.

[15] L. Gasser and M. N. Huhns (Eds.), *Distributed Artificial Intelligence, Vol. 2*, Pitman Publishing/Morgan Kaufmann, London/San Mateo, CA, 1989.

[16] B. Hayes-Roth, "A Blackboard Architecture for Control", *Artificial Intelligence*, 26(1985):251-321.

[17] M. N. Huhns (Ed.), *Distributed Artificial Intelligence*, Pitman Publishing/Morgan Kaufmann, London/San Mateo, CA, 1987.

[18] P. F. Jones, *CAD/CAM: Features, Applications and Management*, Macmillan Press Ltd., London, 1992.

[19] C. R. Kube and H. Zhang, Collective Robotics: From Social Insects to Robots, *Adaptive Behavior*, 2(2):189-218, 1994.

[20] A. Kusiak, "Manufacturing Systems: A knowledge- and Optimization-Based Approach", *Journal of Intelligent and Robotic Systems*, 3(1990):27-50.

[21] Maes, Modeling Adaptive Autonomous Agents, *Artificial Life*, 1(1&2):135-162, Fall 1993/Winter 1994.

[22] A. Meystel, "Intelligent Control: A Sketch of the Theory", *Journal of Intelligent and Robotic Systems*, 2:97-107, 1989.

[23] J. Mataric, "Designing and Understanding Adaptive Group Behavior", Adaptive Behavior, 4:1, December 1995, 51-80.

[24] G. K. H. Pang, "A Blackboard System for the Off-Line Programming of Robots", *Journal of Intelligent and Robotic Systems*, 2(1989):425-444.

[25] W. Payton, D. Keirsey, J. Krozel, K. Rosenblatt, Do Whatever Works: A Robust Approach to Fault-Tolerant Autonomous Control, *Journal of Applied Intelligence*, 2(1992):225-250.

[26] S. Premvuti and S. Yuta, "An Experiment in Realizing Autonomous and Cooperative Behaviours Between Multiple Mobile Robots", *Proceedings of the IEEE International Workshop on Emerging Technologies and Factory Automation (Technology for the Intelligent Factory)*, pp. 301-304, R. Zurawski and T. S. Dillon (Eds.), Melbourne, Australia, 1993.

[27] P. G. Ranky, *Computer Integrated Manufacturing — An Introduction with Case Studies*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[28] J. Rasmussen, B. Brehmer and J. Leplat (Eds.), *Distributed Decision Making — Cognitive Models for Cooperative Work*, John Wiley and Sons, New York, NY, 1991.

[29] A. Rolstadås (Ed.), *Computer-Aided Production Management*, Springer Verlag, Berlin/Heidelberg, 1988.

[30] L. Steels, The Artificial Life Roots of Artificial Intelligence, *Artificial Life*, 1(1&2):75-110, Fall 1993/Winter 1994.

[31] S. G. Tzafestas (1990), "Artificial Intelligence Techniques in Computer-Aided Manufacturing Systems", pp. 161-212, *Knowledge Engineering*, H. Adeli (Ed.), McGraw-Hill, New York, NY, 1990.

[32] E. S. Tzafestas (1995). Vers une Systémique des Agents Autonomes: Des Cellules, des Motivations et des Perturbations, *Thèse de Doctorat de l'Université Pierre et Marie Curie*, Paris, December, 260 pages.

[33] J. Wang and G. Beni, "Cellular Robotic System with Stationary Robots and its Application to Manufacturing Lattices", *Proceedings 1989 IEEE International Symposium on Intelligent Control*, Albany, NY, September 1989, pp. 132-137.

[34] P. K. Wright and D. A. Bourne (Eds.), *Manufacturing Intelligence*, Addison Wesley, Reading, MA, 1988.

[35] T. Yakoh and Y. Anzai, "Physical Ownership and Task Reallocation for Multiple Robots with Heterogeneous Goals", *Proceedings of the International Conference on Intelligent and Cooperative Information Systems*, pp. 80-87, Rotterdam 1993.

[36] S. G. Tzafestas, D. G. Koutsouris and N. I. Katevas (1997). *Proceedings of the 1st Mobinet Symposium (Mobile Robotics technology for Health Care Services)*, Athens, May 1997.

[37] *U. Rembold, R. Dillmann, L.O. Hertzberger, and T. Kanade, Eds. (1995),* Proceedings of the 4th International Conference on Intelligent Autonomous Systems (IAS-4), *Karlsruhe, March 1995.*

[38] T. Tyrrell, *Computational Mechanisms for Action Selection*, Ph.D. Thesis, Centre for Cognitive Science, University of Edinburgh, 1993.

[39] T. Tyrrell, The Use of Hierarchies for Action Selection, *Adaptive Behavior*, 1(4):387-420, 1994.

[40] T. Tyrrell, An Evaluation of Maes's Bottom-Up Mechanism for Behavior Selection, *Adaptive Behavior*, 2(4):307-348, 1994.

[41] S. G. Tzafestas and H. Verbruggen, Eds. (1994), *Artificial Intelligence in Industrial Decision Making, Control and Automation*, Kluwer, Dordrecht/Boston.

[42] R. D. Schraft, E. Degenhart, and M. Hägele (1993), Service Robots: The Appropriate Level of Automation and the Role of Operators in the Task Execution, *Proceedings of the 1993 IEEE Systems, Man and Cybernetics Conference*, pp. 163-169.

[43] E. S. Tzafestas (1994*a*), "Implementing Reactive Algorithms on a Cellular Control Architecture", pp. 191-201, *Proceedings of the 10th International Workshop on Autonomous Mobile Systems (AMS '94)*, Stuttgart, October 1994, Springer-Verlag, Informatik Aktuell Series.

[44] E. S. Tzafestas (1994*b*), "Agentifying the Process: Task-Based or Robot-Based Decomposition?", pp. 582-587, *Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, October 1994.

[45] 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)

[46] S. N. Raptis and S. G. Tzafestas, "Agent-Like Neurofuzzy Architectures for Mobile Robot Path Planning," *Studies in Informatics and Control*, Vol. 6, No. 4 (1997)

[47] 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)

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

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

[50] 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)

[51] 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)

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

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

[54] 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)

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

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

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

[58] 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)

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

[60] 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)

[61] 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)

[62] 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.

[63] 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)

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

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

[66] 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)