A Modified Reactive Control Framework for Cooperative Mobile Robots

J. Salido[a], J.M. Dolan[a], J. Hampshire[b] and P. Khosla[b]

[a]The Robotics Institute, Carnegie Mellon Univ.
Pittsburgh, PA 15213-3891 USA

[b]Dept. of Electrical & Computer Engineering, Carnegie Mellon Univ.
Pittsburgh, PA 15213-3890 USA

ABSTRACT

An important class of robotic applications potentially involves multiple, cooperating robots: security or military surveillance, rescue, mining, etc. One of the main challenges in this area is effective cooperative control: how does one determine and orchestrate individual robot behaviors which result in a desired group behavior? Cognitive (planning) approaches allow for explicit coordination between robots, but suffer from high computational demands and a need for a priori, detailed world models. Purely reactive approaches such as that of Brooks are efficient, but lack a mechanism for global control and learning. Neither approach by itself provides a formalism capable of a sufficiently rapid and rich range of cooperative behaviors.

Although we accept the usefulness of the reactive paradigm in building up complex behaviors from simple ones, we seek to extend and modify it in several ways. First, rather than restricting primitive behaviors to fixed input-output relationships, we include memory and learning through feedback adaptation of behaviors. Second, rather than a fixed priority of behaviors, our priorities are implicit: they vary depending on environmental stimuli. Finally, we scale this modified reactive architecture to apply not only for an individual robot, but also at the level of multiple cooperating robots: at this level, individual robots are like individual behaviors which combine to achieve a desired aggregate behavior.

In this paper, we describe our proposed architecture and its current implementation. The application of particular interest to us is the control of a team of mobile robots cooperating to perform area surveillance and target acquisition and tracking.

Keywords: Cooperative mobile robots, Social behavior, Reinforcement learning, Intelligence

1. INTRODUCTION

Autonomous mobile robot control poses a challenge to researchers in intelligent systems due to the broad range of possible applications and the difficulty of dealing with dynamic, *partially unknown environments.* In this paper we present a general, scalable *'multi-agent* architecture'*. In the case of particular interest to us, the intelligent agents are mobile robots cooperating in a hostile environment to perform surveillance and reconnaissance missions.

This paper is organized as follows. In Section 1 we identify, motivate, and situate the problem with respect to previous work and give desired characteristics of the architecture. Section 2. gives a conceptual description of the architecture, and Section 3. describes that part of the architecture which has been implemented. Section 4. outlines contributions and future work.

Other author information: J.S.: Visiting Scholar from Castilla-La Mancha Univ. (Ciudad Real/Spain), supported by the Spanish MEC under grant PF96-0070984147; Telephone: 412-268-1413.
J.M.D.: Telephone: 412-268-7988.
J.H.: Telephone: 412-268-7961.
P.K.: Telephone: 412-268-5090.
Fax: 412-268-3890. Email: {jsalido, dolan, pkk}@cs.cmu.edu, hamps@ece.cmu.edu

*In this paper we use the term 'intelligent agent' and 'mobile robot' synonymously. For various definitions of 'intelligent agent', see Ref. 1.

1.1. The problem

We seek a multi-agent architecture with the following characteristics: taskability of the multi-agent aggregate as a single logical entity, scalability (i.e., the logical structure of the architecture should apply at each level in the hierarchy), learning capability, and simulation capability which allows the merging of real and virtual environments.

In our particular scenario, given a high-level mission description for a team of mobile robots, we seek to achieve mission goals through cooperation among robots and on-line interaction between them and human users. The details of inter-robot cooperation should be transparent to the user. Mission phases include *deployment* and target *detection* and *tracking*.

1.2. Related work

The first intelligent robots originated within the AI arena.[2] Systems like PLANEX, STRIPS,[3] PRODIGY[4] used symbolic world models to do exhaustive planning. This strategy is referred to as the *deliberative approach.* The main hypothesis in the deliberative approaches is that *principles of theorem demonstration can be applied to a symbolic representation of the world.*[2]

In spite of the fact that some sort of planning is needed for goal-oriented behavior, earlier work[5] demonstrates significant weaknesses in the deliberative approach:

1. *Representation problem:* It is a very complex problem to decide the best knowledge representation to use for reasoning. Popular paradigms for representing knowledge (production rules, semantic nets, frames, etc.) are often suitable for solving problems in artificial worlds abstracted from the real world. They are less suited, however, for handling aspects of the real world like uncertainty and the need for common-sense reasoning.

2. *Interpretation problem:* Translating sensory perceptions from the real world into an accurate 'mental' representation, i.e., into useful and general conceptual constructs, remains a major challenge to AI.

3. *Resources requirements problem:* Even in simple environments (e.g. the 'blocks world') some problems turn out to NP-hard, making them impractical for time-constrained systems.

The major response to the deliberative approach is the so-called *reactive approach*[6,7]. The main features of the reactive approach are: *an enforced absence of any internal representation of the world; the belief that intelligent behavior will emerge from the interaction between creature's primitives behaviors and its environment; and embodiment of intelligent inside physical robots, rather than simply in software or simulation.*[6] Despite some impressive results, the reactive approach requires a high degree of fine-tuning in order to work well for a given task. Additionally, in its pure form, it does not allow for learning from experience.

An approach which attempts to combine the advantages and limit the disadvantages of the deliberative and reactive architectures is the *hybrid approach.* Some researchers postulate a fourth approach designated as *behavior-based approach.* The key distinction between reactive and behavior-based approaches appears to be that the latter adds internal state in order to permit learning. One could thus say that the 'behavior-based' approach is a hybrid approach attained from a reactive,[8,9] rather than a deliberative,[10] starting point. There are other hybrid approaches which lie at various points along the deliberative-reactive spectrum.[11–13]

Researchers have tried to incorporate learning into the various approaches (See Refs. 4,14,15,9,16,17). In deliberative and hybrid approaches, learning improves performance based on past experience. In reactive approaches modified to include learning, learning provides a method for tuning the interaction among primitive behaviors.

Encouraged by the simplicity and appealing performance exhibited by robots controlled following reactive and behavior-based approaches, a number of researchers have investigated the control of multiple cooperating robots. Much of the most recent work is devoted to controlling *'societies of artificial creatures'* (see Refs. 18,16).

Previous approaches to single-or multi-agent architectures have been fairly "brittle", i.e., only able to accommodate the particular application under consideration. We propose a general, hardware-independent methodology to design intelligent agents capable of accommodating a wide variety of the above-described approaches. A key component of our architecture is its scalability, i.e., its conceptual application at various hierarchical levels (e.g., to a single robot, a group of robots, or a group of groups of robots).

1.3. Desired architecture characteristics

This section describes the main features of our approach: the presence of both goal and event-driven behaviors, the major importance of perception in acquiring knowledge, and the ability to learn from experience.

• **Tasking as single logical entity:** *Social behavior* and *individual intelligent behaviors* are strongly interrelated. Nature and daily life teach us how to increase the performance of productive behaviors through individual specialization and cooperation with peers. Moreover, very often, *redundancy and reliability* are synonymous because single faults do not become so critical when redundant components are present. A team of relatively inexpensive robots cooperating to achieve a common goal is therefore an appealing idea. However, a human user should not have to spell out individual tasks for each robot; the control architecture should allow high-level tasking and perform task decomposition for the aggregate of robots.

• **Planning requirements:** Our robots need to be flexible enough to achieve global objectives in a continuously changing, partially unknown environment. This is carried out by planning the right sequence of operations before execution and modifying the plans 'on-the-fly' whenever it is needed. *Context changing* is provided by *perceptual triggers*, which allow one to make real-time decisions without having to take into account all possible behaviors. Analogously, almost all of our daily activities as humans are based on schedules and plans in which spatial-temporal references provide a framework to situate our perceptions. The use of such a scheme provides goal-oriented behavior without requiring exhaustive replanning whenever the situation changes.

• **Event-driven responses:** Mobile robots in a dynamic environment need to *react* quickly to unforeseen events, such as a mobile obstacle crossing one's path, low power-level status, etc. These events need to be checked during the robot's normal operation, perhaps as a background event-loop process generating interrupts to the normal control processes, stopping[†] them, and activating specific controllers to handle events. Although reactivity is need to deal with unforeseen events, a completely reactive robot, without any internal representation of its perceived world, is too simple for achieving complex tasks. Fully reactive robots have the following disadvantages:

1. *Cyclic-response problem:* Reactive creatures can get stuck in repetitive loops. Even if there is some kind of *reactive noisy response,* it can be difficult to avoid undesirable behaviors (the *"fly-at-the-window"* problem).

2. *Priorities set-up problem:* For event-driven behaviors it is necessary to establish a priority among events for conflict resolution. As the number of primitive behaviors increases with task complexity, the assignment of behavior priorities becomes more difficult.

Current reactive systems, often ingeniously implemented, lack a coherent methodology and need to be manually tuned during a lengthy experimentation period. Thus, building complex reactive systems is a difficult task, though some promising results are being achieved in the domain of *machine learning*[17] *(reinforcement learning, Markovian decision processes, evolutionary programming, etc.).*

• **Flexible behavior via perception:** Everything we know was 'recorded' in our minds during a perception process[‡] involving one or more of our senses (e.g. in a conference, a listener receives both visual images and sounds as sources of information). Therefore, perception determines the basic elements used in constructing knowledge. This does not mean knowledge is impossible when the amount of information received is reduced, but it does imply a change in 'internal' representation. For instance, blind people are able without the use of vision to construct internal representations used to 'navigate' in their daily 'world' with autonomy. The case of blind individuals illustrates how, in the real world, knowledge is gained even from restricted perceived information.

Conceptual interpretation of perception is possible when a system observes with reference to spatial-temporal cues. Perception without spatial or temporal cues does not provide useful information to interpret from a conceptual standpoint, making the observer's response less flexible. It is usually easier to focus on spatial changes of perception (e.g., gradient functions applied to instantaneous visual images); however, time-based observation is important to describe responses of a dynamic system (e.g., the out put of a PID controller output is based on integration and differentation of error signals). It is important to note that purely reactive controllers, for example, cannot produce integral-differential outputs because they do not store any internal status.

[†] We assume a control system that can be decomposed into several concurrent low-level controllers.
[‡] Throughout this paper we use the term *'observation process'* as a synonym for a generic perception process.

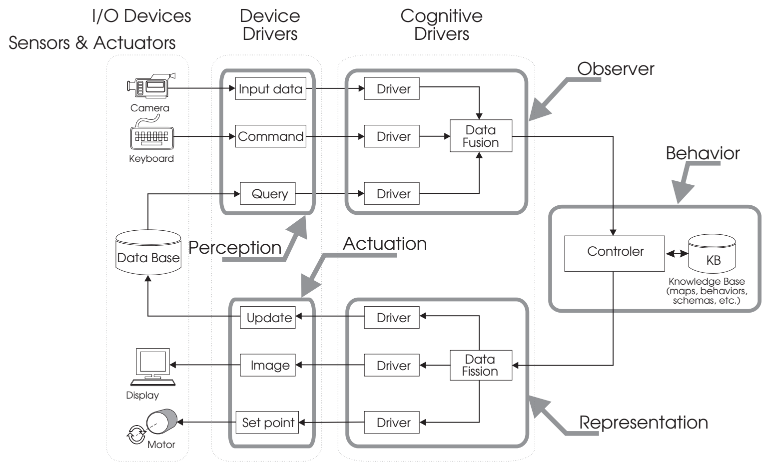

Figure 1. Functional modules in the proposed framework.

Our aim is to provide a general framework where each cognitive process can be regarded as a *observation-decision-action (ODA)* cycle where concepts extracted form perception are manipulated, in such a way that an intelligent agent's activity is a collection of embedded concurrent ODA-cycles.

• **Intelligence from experience:** One of the most valuable benefits of experience is a store of knowledge which allows improved performance when similar situations are faced. Our architecture provides a framework within which conventional control theory methods for locomotion can be conveniently combined with AI paradigms and behavioral approaches exploiting the newest methodologies on machine learning.

2. CONTROL FRAMEWORK OVERVIEW

In this section we describe the proposed control framework. Our approach is based on the issues discussed in the previous section.

2.1. Brief description of modules

Fig. 1 depicts the functional modules in the framework. At the highest level, there are three basic functions: input (consisting of the perception and observer modules), processing (consisting of the behavior module), and output (consisting of the actuation and representation modules). The input and output functions are each decomposed into two modules, one low-level and the other high-level. The low-level modules (perception for input and actuation for output) are hardware-dependent device drivers. Their use makes the high-level module (observer for input and representation for output) device-independent. The high-level "cognitive driver" modules are dependent on the sensor type (e.g. vision, sonar, etc.). Their use makes the behavior module independent of the sensor type.

2.1.1. Perception module

This module transforms a received sensory stimulus into internal information for processing (e.g. a visual stimulus is transformed into digital information using electric voltages). Perception can be not only of the physical world, but also of simulated virtual worlds or even abstract worlds*. In this context, the usual meaning of sensor is extended to designate any hardware or software artifact capable of transforming information. Some examples of such sensors are: sonars, cameras, queries to data bases and commands. The device drivers in this module provide device-independent information to the next module.

*By abstract worlds we mean, for example, perception of the internal status of the agent.

2.1.2. Observer module

This module interprets the perceived information, translating it into *abstract concepts* used for making decisions. By abstract concepts is meant internal representation of perception, for example, 'wall' might correspond to a particular set of percepts from sonar sensors. These abstract concepts are referenced to both *perceptual states* and *transitions* between perceptual states (e.g. a wall always is a wall but when we refer to it we might locate it 'on the left side' and 'now', 'later'...'). Temporal references to perception can be done using *'perceptual sequencing'*, which describes the methodology by which multiple perceptual algorithms can be scheduled within the context of a single behavior.[19] During the observation process the observer module interprets into concepts the information coming from sensor devices. In general, the interpretation achieved by observers is a *'remove details'* process. It is also possible to see the concepts as a code that identifies a specific class created from perceived information. For example, a robot in a room could see with its camera different kinds of objects identifying every one with a code or label: walls, desks, doors, windows, etc.

We can distinguish two activities in the observer module:

1. *Cognitive drivers:* These provide a transparent link between the perceived world (from the agent's point of view) and the internal one. By transparency is meant that agents should not be concerned about the nature of the world that generates their perceptions, i.e., it might be a physical world, a virtual world, or even the abstract world constituted by their own knowledge. Moreover, cognitive drivers provide information which is independent of the sensing modality. If a cognitive driver detects movement, the output information associated with this movement will contain no specifics about its origin (sonar, vision, etc.). It is important to note that the same cognitive driver might be used in several different observers in the same agent. For example, the cognitive driver that provides notions of stationary objects could be used in a behavior module to avoid them and in other to update a map.

2. *Perception fusion:* This component performs *data fusion* by collecting and integrating information received from diverse sensor sources.

The observer module needs to deal with two main problems:

- *Representation problem:* As described earlier, this problem is concerned with the suitable way to represent the agent's knowledge and its acquisition. In our approach we need to find the right set of concepts to gather from perception. Because agents should be able to communicate with an human operator, a common communication protocol is needed. This protocol may be achieved from *'supervised learning'* (e.g. teaching the agent which concept to use to classify its perception in each perceptual state).

- *Interpretation problem:* The set of concepts representing knowledge should be as general as possible. Making decisions by matching perceptions with previously stored concepts may exhibit low performance, as in traditional AI rule-based systems[†] (RBS). In RBS, concept generalization is carried out using variables inside the patterns.[2] On the other hand, our approach uses spatial-temporal references where concepts are observed. For instance, obstacle presence might be detected observing spatial and temporal gradients on sonar readings. Using gradients to define perceptual concepts gives them great generality, because it is not necessary to deal with a specific reading to interpret a concept. Specific values are needed to represent the actual response to actuator devices. The interpretation problem can be solved by using an associative memory created by supervised learning when perceptual concepts are taught.

2.1.3. Behavior module

The behavior module contains the robot's *controllers*. It makes the decisions to achieve a desired overall output behavior. Its input is a set of abstract concepts instantiated with a vector of parameters. These parameters characterize useful information from a control point of view (e.g. accuracy of readings, spatial-temporal coordinates, observed values, etc.). Its output is also an abstract concept with a vector attached to it. In this case, the vector is used to build a proper response in the actuator module.

[†]It is notorious than in rule-based systems, the *'pattern- matching'* stage typically demands 90% or more of total execution time.

An intelligent agent might contain several controllers, each of them dealing with a different kind of perceived information. As we noted earlier, the observation process can be done at different levels of abstraction. Behavior modules can therefore be attached directly to perceived information from the external world or to an abstraction built from the external world.

2.1.4. Representation module

The major purpose of this module is to build suitable representations of inputs received from controllers. In this context a 'suitable' representation yields information usable by the actuators. The representation process can be understood as a *'details addition'* process applied to abstract information.

There are two different activities in the representation module:

1. *Actuation fission:* It performs a data fission process which disaggregates the decisions made by a behavior module into information directed to the proper cognitive drivers.

2. *Cognitive drivers:* Like cognitive drivers in the observer module, their mission is to provide a transparent link between decisions and actuation actions. Drivers deal with low-level issues to perform the actual actions, so it is transparent to behavior modules whether the actions are directed to a physical or virtual world, or even an abstract world. The same driver might receive requests from different representation modules.

2.1.5. Actuator module

This module produces the output to an 'external world' referenced by a representation module. It is possible to consider actuator modules at an abstract level, where the actuation produces a new kind of abstract information. Examples of actuators are: electrical motors, update operations on data bases, and received information through a communication link.

2.2. Main features

In this section we discuss the strengths of the proposed architecture.

2.2.1. Integrating control approaches

Our approach is flexible enough to accommodate a wide variety of control approaches which are often regarded as mutually exclusive. The boundaries for modules depend on the chosen observation process. That is, the framework can be considered *recursive*, so that it is possible to contemplate different replicas of the same framework at several abstraction levels[‡]. This has several advantages:

▷ **Low vs. High level control:** Discriminating between low- and high-level control only depends on the behavior module's point of view. The decisions made inside this module might be considered 'high-level control', while decisions at lower abstraction levels constitute the 'low-level control'. To decide among several control algorithms (linear PID, fuzzy logic, neural nets, etc.) is a question of designing optimum controllers.

▷ **Layered vs. Hierarchical control:** In layered control several primitive actions compete to produce the total response control *('bottom-up')* versus hierarchical schemas where there is an interrelation among different control actions to get the final response *('top-down')*. In the framework described, both of these schemas can operate simultaneously. For example, we can consider as hierarchical the relationship between controllers at different abstraction levels, whereas the net result of the activity of all controller at a given level is a layered phenomenon.

▷ **Reactive vs. Deliberative control:**

The actual degree of system reactivity is somewhat subjective. For instance, it is very easy to assume reactivity when we see a mobile robot avoiding an unforeseen obstacle but it is more difficult to assume when we do not see the cause that gives rise its behavior (e.g., low power level in batteries). In our approach, each behavior module reacts to its own perception, so that even planning could be considered a sequence of behavior module activations (set-point adjustments) derived from more abstract behavior modules, and therefore reactive in a certain sense.

[‡]These ideas have their main inspiration in some works done about *'cognitive autopoiesis*[20] *'*.

2.2.2. Learning to be smart

An intelligent agent dealing with dynamic, partially unknown environments should be flexible enough to learn from earlier experiences. In our approach this *'learning from experience'* can be looked at from several perspectives:

1. *Learning behaviors:* The controllers implemented inside a behavior module may be learned[§] off-line (supervised learning) or on-line (unsupervised learning). The learning method depends strongly on the controller characteristics (linear adaptive, non-linear neuro-fuzzy, etc.). This kind of learning could be seen also from the framework already described, because the controller tuning process may be achieved with the proper actuators over a behavior module.

2. *Building and using maps:* In partially unknown environments much past experience can be captured by building representations of perceptions[¶]. This knowledge can be used in the future to elaborate plans predicting the consequences of actions.[9,21]

3. *Reusing controllers:* A framework to develop intelligent agents should provide the adequate interface to reuse behavior modules developed for similar functions, in such a way that modules can be *'plugged and played'.*

In addition, it is possible to consider a learning process consisting of arranging the proper connections between modules to produce a composed behavior module. This issue is closely related with *'skills acquisition'* from primitives or more simple behaviors. We can extend this to higher levels of abstraction by, for example, learning how to do plans, or even *'learning how to learn'.*

Learning from experience requires training, and repeating experiments in a real world usually demands much time and high cost. In this context, having a framework that allows robots to be trained in a simulation world, where physical and virtual objects are merged, offers great flexibility with relatively small cost.

2.2.3. Resolving conflicts

Robustness is one of the most important issues is mobile robot control. The two main functions of robust control are: *diagnosis* and *error recovery.* In our approach these functions can be implemented by the various observer, behavior, representation and actuation modules. In order to do this, we need to provide specific supervisory behavior modules testing the correct responses of other behavior modules. At the same time, *'default behaviors'* must be provided to guarantee that an intelligent agent is able to show *'initiative'* in the absence of external perceptions. Then, default behaviors may be regarded as the *'observation of one self'* and the consequent reaction.

A difficult problem in planning is choosing between different conflicting decisions. Solving this problem by assuming explicit priorities only translates the problem into how to assign the right ones. In our approach we pursue *'implicit priorities'* in the same way that living creatures do (e.g. instinctive reflexes do not have explicit priorities attached). In practice, the use of implicit priorities means that when the first action that commanded to an actuator is the first action executed[‖]. Even in our approach priorities can be explicitly assigned by originating them from deeply abstract behavior modules, i.e., priorities could be set by a behavior module which decides priorities of actions in another behavior module. The architecture accommodates different levels of concurrence: inside a controller, a chain of controllers in distinct abstraction levels; among controllers in the same level of abstraction, controller actions lead to different actuation modules (see Fig. 2).

2.3. Controlling a society of robots

The inherent recursivity of our framework provides enough scalability to complex and powerful designs for an intelligent agent. We noted earlier the drawbacks of working with a single, expensive, sophisticated mobile robot to accomplish a mission in an hostile environment. Such risks are avoided using cooperative teams of cheaper robots. However, the issues that arise working with cooperative teams of autonomous agents are full of complexities.

Using the control framework described in the previous sections it is possible to an *agent observing other agents in its environment.* Moreover, communications amongst agents provide significant information to this observer process.

[§] Here learning may be understood as a *tuning* process in which the parameters for the controller are conveniently tuned.
[¶] If we think about perception as a kind of observation, the attached relations to maps might be spatial and/or temporal.
[‖] Here, asynchronous or sincronous execution of action is not assumed, since implicit priorities could be implemented in both models.

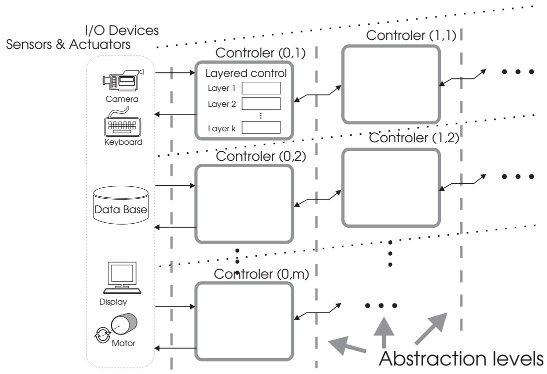

Figure 2. Levels of concurrence.

In our case we consider all the interactions among agents as different ways of observation. The difference between interaction via sensing vs. explicit communication should be transparent from a design point of view, the differences being handled by specific drivers.

Assuming a *decentralized architecture*, the decision between a distributed schema or a hierarchical** one should be dependent on the application and the number of robots required to complete the task. In problems such as we deal with in our current research (surveillance and reconnaissance missions for scout robots), a hierarchical organization appears the most convenient. However, the choice is greatly influenced by the number of agents on the team. The advantages for a distributed scheme are obvious: all the robots behave the same way, and the system exhibits fault tolerance and scalability. Unlike distributed schemas, hierarchical schemas demand dedicated mechanisms to recover from errors (in this schema the control is locally centralized) and scalability is not so easy. The advantages of hierarchical schemas arise in applications that require a large number of agents; as the this number increases, cooperation becomes more difficult due to limitations on the communications bandwidth. In such situations a hierarchical organization becomes more suitable. The adopted structure might be similar to the *'rank structure'* present in human (*'castes'* in other biological creatures) military organizations. Within a given rank, the interaction among agents arises hierarchically through *'leadership relations'*. Moreover, in this organization a higher control level could be implemented assuming that agents can supervise the tasks done by other agents.

Although in our application we have a very small number of physical robots, the designed framework permits us to populate the 'world' with a large number of virtual agents. In this way, we are able to experiment with hierarchical schemas, where it is possible to investigate the following questions: *How to determine the 'right' number of robots to use? How many ranks are needed for the hierarchy to be efficient? How to determine the assigned task to each rank? Which properties are desirable, or needed for the communication protocol? Which protocols are more suitable for error detection and recovery?*

3. IMPLEMENTATION

3.1. Software layers

From the implementation point of view we identify several desirable layers in the software application.

- *Native operating system (OS):* The primary OS for the chosen hardware platform (e.g./ Linux, Solaris, SunOS, etc.). It provides the basic languages, compilers, etc. used to develop applications.
- *Native programming languages and tools:* The natural paradigm to implement intelligent agents on software is *'object oriented programming' – OOP* (e.g. C++, Java, etc.).
- *Task control 'operating system' (TcOS):* This layer is composed of routines to manage interactions among low-level tasks. These routines deal with communication among tasks, execution monitoring, error recovery,

**Unlike hierarchical schemas, in a distributed one, all the agents are equal with respect to control.

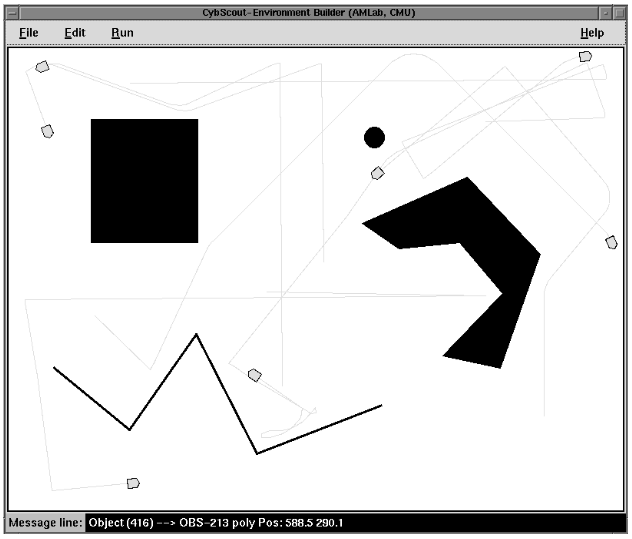

Figure 3. A simple scenario running primitive behaviors.

management of tasks in a distributed architecture, etc. We call this an 'operating system', because in the robots' task level this layer deals with the issues of proper to an OS.

- *Perceptual-action language (PAL):* In a higher level over TcOS, PAL provides a machine-independent natural language to describe the missions, plans and behaviors for robots. Thus, we can program robots without to worry about the low-level issues involved.

- *Automatic programming tools:* One of the big challenges of software engineering is the development of automatic programming tools. In robotics we talk in a similar way of *'planning'*. The proposed problem is the automatic generation of a program to achieve a high-level description of goals where details are sparse. When the generated program is given in a high-level language (in our case PAL), it is possible to get a clear idea of how the planner solved a problem, and even how to modify the generated plan in a desired way.

- *Developing and monitoring environment:* At the highest level, the developing and monitoring environment provides graphical user interfaces (GUIs) to increase the user friendliness of applications, hiding lower-level details. We are developing GUIs which provide toolbars for scenario editing, object inspectors to configure properties, monitors for inspecting the internal agents' status, remote connections via Internet, hypertext helps, and other features.

At present, we are developing a prototype from which, following a top-down methodology, we will isolate the described layers.

3.2. Simulator

Simulation is an inexpensive way to train intelligent agents (in the same way that simulators are used to train humans). Rather than building a specific application for simulation, we are including this feature as a component of the main control architecture so that virtual and real objects are indistinguishable from the GUI's point of view.

We have developed an initial prototype to start simulations and experiments with real robots. In Fig. 3 we see a typical plot obtained by the simple behaviors *'wander'* and *'avoid-obstacle'*. This early version provides tools for the editing scenarios populated by various kinds of objects (obstacles and mobile robots*).

*Though we consider obstacles and robots as main objects, we design our application to incorporate other kinds of objects like *surfaces* for exploration and *meta-objects* with special meanings (targets, marks, etc.).

4. CONCLUSIONS – FUTURE WORK

In this paper we present our approach to a reactive control framework for cooperative mobile robots with the following characteristics:

- *Integration of both reactive and deliberative* approaches through recursive observation-action processes at different abstraction levels.
- Implicit and explicit assignment of behavior *priorities*. Implicit priorities are assumed by the intrinsic connections between behavior modules to observer and representation modules. Explicit priorities may be handled in a behavior module internally or may be generated from a behavior module at a deeper abstraction level.
- *Scalability* of a team or society of cooperative robots in which either hierarchical or distributed schemas may be applicable according to the desired performance.
- Different *levels of concurrence* with potential exploitation of distributed processing.
- Transparent *integration of virtual-object perception,* making combined virtual-real operation feasible and thereby accommodating behavior learning without the need for large number of physical robots.

In order to realize the framework described in this paper, we will do further work in the following areas:

- Special interest needs to be devoted to the development of the TcOS and PAL software layers. *Visual programming tools* are very desirable in this regard, since they should provide an intuitive way to link modules. We rely on the important background acquired developing the CHIMERA[22] OS and the ONIKA[23] visual interface.
- *Strategies for learning.* We will explore alternatives coming from evolutionary programming, reinforcement learning, neural programming, and other methods. We will consider several applications of learning:
  - Adjust the parameters of behavior modules to gain maximum efficiency.
  - Explore the right interaction between behavior modules to achieve the global behavior desired.
  - Discover which perceptual patterns are most significant in describing the agent's environment.
  - Decide the right set of observers to capture the most useful perceptual patterns. Reconfiguration of sensors will be very fruitful in exploring this issue.
  - The previous point could be extended to higher levels in the hierarchy in order to deal with issues of population, configuration, and interaction of agents within an society of agents to achieve global goals.
- To validate the benefits from *simulation merging real and virtual worlds* is an interesting issue due to the important time and cost advantages in performing complex experiments partly in simulation. We will use a simulate/implement cycle which puts simulation results as quickly as possible into practice for further testing and refinement.
- Optimizing human group behavior for maximum productivity within industrial or other settings is an old problem. Despite the differences between human and machine societies, we will attempt to apply some of the insights resulting from this sort of productivity optimization to analogous societies of intelligent robots.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the support of the Robotics Institute (RI) and the Institute for Complex Engineered Systems (ICES) at Carnegie Mellon University (Pittsburgh, PA, USA). One of the authors (J.S.) is most appreciative for the hospitality shown to him while a visitor at the Advanced Mechatronics Laboratory (RI) during the completion of his work and the financial support from the Spanish Ministry of Education and Science (MEC) under grant PF96-0070984147.

REFERENCES

1. M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," Oct. 1994. Revised January 1995.
2. N. J. Nilsson, *Principles of artificial intelligence*, Tioga Pub. Co, 1980. Springer-Verlag, 1982.
3. R. E. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *AI Magazine* **5**(2), pp. 189–208, 1971.
4. J. G. Carbonell, C. A. Knoblock, and S. Minton, "PRODIGY: An integrated architecture for planning and learning," Tech. Rep. CMU-CS-89-189, Carnegie Mellon University, Oct. 1989.
5. D. Chapman, "Planning for conjunctive goals," *AI Magazine* **32**, pp. 333–337, 1987.
6. R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation* **RA-2**(1), pp. 14–23, 1986.
7. P. Maes, "A spreading activation network for action selection," in *Intelligent Autonomous Systems*, vol. 2, pp. 875–885, 1989.
8. R. C. Arkin, "Motor schema-based mobile robot navigation," *International Journal of Robotics Research* **8**, pp. 92–112, Aug. 1989.
9. M. J. Matarić, "A distributed model for mobile robot environment-learning and navigation," Tech. Rep. AI-TR-1228, Massachusetts Institute of Technology, May 1990.
10. J. Blythe and W. S. Reilly, "Integrating reactive and deliberative planning for agents," Tech. Rep. CMU-CS-93-155, Carnegie Mellon University, May 1993.
11. R. Simmons, L. J. Lin, and C. Fedor, "Autonomous task control for mobile robots," in *International Symposium on Intelligent Control*, *Proc. IEEE* , 1990.
12. C. Thorpe, *Vision and navigation: The CMU Navlab*, Kluwer Academic Publishers, 1990.
13. F. R. Noreils and R. G. Chatila, "Plan execution monitoring and control architecture for mobile robots," *IEEE Transactions on Robotics and Automation* **11**, pp. 255–266, Apr. 1995.
14. J. Grefenstette, "Evolutionary algorithms in robotics," in *International Symposium on Robotics and Manufacturing (ISRAM)*, 1994.
15. D. Golberg, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley Publishing Company, 1989.
16. M. J. Matarić, "Reinforcement learning in the multi-robot domain," *Autonomous Robots* **4**, pp. 73–83, Jan. 1997.
17. T. Mitchell, *Machine Learning*, McGraw Hill, 1997.
18. Y. Cao, A. Fukunaga, and F. Meng, "Cooperative mobile robotics: Antecedents and directions," in *Intelligent Robotics and Systems*, *Proc. IEEE* **1**, pp. 226–234, Aug. 1995. Pittsburgh PA, USA.
19. R. C. Arkin and D. MacKenzie, "Temporal coordination of perceptual algorithms for mobile robot navigation," *IEEE Transactions on Robotics and Automation* **10**, pp. 276–286, June 1994.
20. H. Maturana and F. Varela, *The tree of knowledge: The biological roots of human understanding*, Shambhala / New Science Press, 1987. Revised edition released in 1992.
21. J. Borenstein, H. R. Everett, and L. Feng, *Navigating Mobile Robots: Sensors and Techniques*, A. K. Peters, Ltd., first ed., 1996.
22. D. Stewart, D. Shmitz, and P. Khosla, "The Chimera II real-time operating system for advanced sensor-based robotic applications," *IEEE Transactions on Systems, Man, and Cybernetics* **22**, pp. 1282–1295, November/December 1992.
23. M. Gertz, R. A. Maxion, and P. Khosla, "Visual programming language and hypermedia implementation within a distributed laboratory environment," *Journal of Intelligent Automation and Soft Computing* **1**, pp. 43–62, Apr. 1995.