Robots in a Distributed Agent System

Didier Guzzoni, Kurt Konolige, Karen Myers, Adam Cheyer, Luc Julia

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

guzzoni, konolige, myers, cheyer, julia @ai.sri.com

Abstract

We describe how we manage cognitive information within our mobile robotics activities. Our approach can be divided into three parts: the local level through Saphira (the navigation system), the global state through OAA™ (the Open Agent Architecture), and the representation and human-user interaction through multimodal interfaces.

Introduction

In previous work (Konolige and Myers 1998) we discussed the requirements for autonomous mobile robot operation in *open-ended* environments. These environments were loosely characterized as dynamic and human-centric, that is, objects could come and go, and the robots would have to interact with humans to carry out their tasks. For an individual robot, we summarized the most important capabilities as the three C's: coordination, coherence, and communication. These constitute a cognitive basis for a stand alone, autonomous robot.

Coordination: A mobile agent must coordinate its activity. At the lowest level there are commands for moving wheels, camera heads, and so on. At the highest level there are goals to achieve: getting to a destination, keeping track of location. A complex mapping between these two levels changes, depending on the local environment. How is the mapping to be specified? We have found, as have others, that a layered abstraction approach makes the complexity manageable.

Coherence: A mobile agent must have a conception of its environment that is appropriate for its tasks. Our experience has been that the more open-ended the environment and the more complex the tasks, the more the agent will have to understand and represent its surroundings. We have found that appropriate, strong internal representations make the coordination problem easier, and are indispensable for natural communication. Our internal model, the Local Perceptual Space (LPS), uses connected layers of interpretation to support reactivity and deliberation.

Communication: A mobile agent will be of greater use if it can interact effectively with other agents. This includes the ability to understand task commands, as well as integrate advice about the environment or its behavior. Communication at this level is possible only if the agent and its respondent internalize similar concepts, for example, about the spatial directions "left" and "right". We have taken only a small step here, by starting to integrate natural language input and perceptual information. This is one of the most interesting and difficult research areas.

Although the above approach has proven useful for single robotics agents, in recent years our thinking has changed to a broader view of mobile robots, one in which they are considered to be a physical part of a larger, distributed system. Instead of having all the cognitive functions necessary for autonomy implemented on a single physical platform, the functions are distributed, both physically and conceptually, as a network of *agents*. An agent can be implemented in software and reside on some computer, or it can be a physical robot with some local sensing and computational abilities, and a wireless connection to the network. Each agent has its own capabilities, and together the network of agents constitutes the system.

There are many advantages to this agent-centered design. One is the ability to rapidly reconfigure the system to respond to a changing environment or changing task mix. Another is the ability to use agent components, with specialized expertise, that have been developed for other systems, for example a speech input agent or a map agent.

In this paper we will lay the broad outlines of this approach, by first looking at the local cognitive state of a robot, then the global agent architecture and how physical robots fit in, and finally some particular aspects of human interaction with the agent system.

Local cognitive state: Saphira

The Saphira architecture (Saffiotti 1995; Konolige and Myers 1998) is an integrated sensing and control system for robotics applications. At the center is the LPS (see Figure 1), a geometric representation of space around the robot. Because different tasks demand different representations, the LPS is designed to accommodate various levels of interpretation of sensor information, as well as *a priori* information from sources such as maps. For example, there is a grid-based representation similar to Moravec and Elfes' occupancy grids (Moravec and Elfes 1985) built from the fusion of sensor readings, as well as more analytic representations of surface features such as linear surfaces, which interpret sensor data relative to models of the environment.

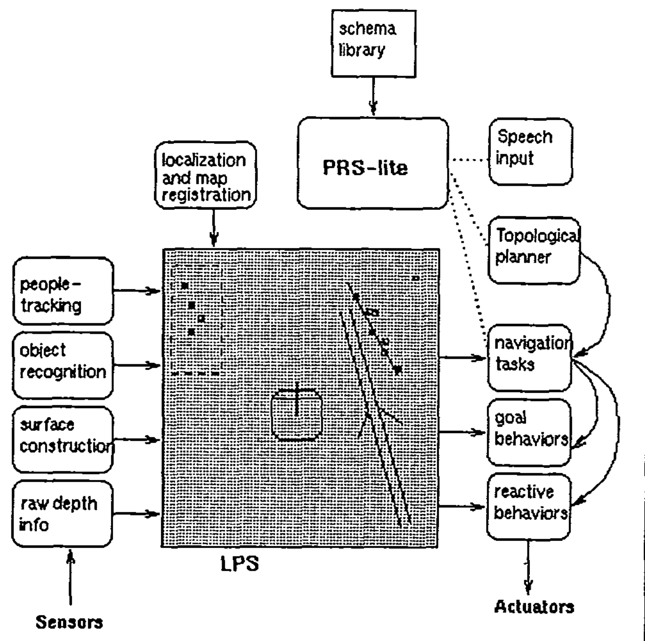

Figure 1 Saphira system architecture. Perceptual routines are on the left, action routines on the right. The vertical dimension gives an indication of the cognitive level of processing, with high-level behaviors and perceptual routines at the top. Control is coordinated by the Procedural Reasoning System (PRS-lite), which instantiates routines for task sequencing and monitoring, and perceptual coordination.

Semantic descriptions of the world use structures such as corridors or doorways (*artifacts*). Artifacts are the product of bottom-up interpretation of sensor readings, or top-down refinement of map information.

The LPS gives the robot an awareness of its immediate environment, and is critical in the tasks of fusing sensor information, planning local movement, and integrating map information. The perceptual and control architecture makes constant reference to the local perceptual space. One can think of the internal artifacts as Saphira's *beliefs* about the world, and most actions are planned and executed with respect to these beliefs.

In Brooks' terms (Brooks 1986) the organization is partly vertical and partly horizontal. The vertical organization occurs in both perception (left side) and action (right side). Various perceptual routines are responsible for both adding sensor information to the LPS and processing it to produce surface information that can be used by object recognition and navigation routines. On the action side, the lowest-level behaviors look mostly at occupancy information to do obstacle avoidance. The basic building blocks of behaviors are fuzzy rules, which give the robot the ability to react gracefully to the environment by grading the strength of the reaction (e.g., turn left) according to the strength of the stimulus (e.g., distance of an obstacle on the right). Navigation routines make use of map information to guide the robot toward goal locations, for example to a corridor junction. At the same time, registration routines keep track of sensed objects, constantly relating them to internal map objects to keep the robot accurately positioned with respect to the map. Thus, Saphira is able to accept a plan, a sequence of waypoints to a final goal, and execute it while keeping the robot localized within the global map.

Behaviors

At the control level, the Saphira architecture is behavior-based: the control problem is decomposed into small units of control called *basic behaviors*, like obstacle avoidance or corridor following. One of the distinctive features of Saphira is that behaviors are written and combined using techniques based on fuzzy logic. Each behavior consists of an *update function* and a set of fuzzy rules. The purpose of the update function is to extract information from the LPS and turn it into a set of fuzzy variables appropriate for the behavior. For example, an obstacle-avoidance behavior might have the following variables, indicating where the robot's path is blocked:

front-left-blocked
front-right-blocked
side-left-blocked
side-right-blocked

Each fuzzy variable takes a value from the interval [0..1], indicating the degree to which its condition holds.

Coherence

Reactive behaviors such as obstacle avoidance often can take their input directly from sensor readings, perhaps with some transformation and filtering. More goal-directed behaviors can often benefit from using *artifacts*, internal representations of objects or object configurations. This is especially true when sensors give only sporadic and uncertain information about the environment. For example, in following a corridor, a robot will not be able to sense the corridor with its side sonars when traversing open doorways or junctions. It would be foolish to suspend the behavior at this point, since over a small distance the robot's dead-reckoning is good enough to follow a "virtual corridor" until the opening is passed.

In other situations, an artifact may represent an artificial geometric entity that guides the behavior. Such situations occur frequently in human navigation, for example in crossing a street one tends to stay within a lane defined by the sidewalks on either side, even when there is no painted crosswalk. Similarly, in the follow-corridor behavior, the robot is guided by a lane artifact that is positioned a foot or so in from the corridor walls.

In accordance with these behavioral strategies, artifacts in Saphira come from three sources:

- From *a priori* information. Typically, the robot will start with a map of the corridors and offices in its environment.
- From perceptual features. When a perceptual process recognizes a new object, it may add that object to the list of artifacts.
- Indirectly, from other artifacts or goal information. For example, if the user gives the command, "Move 3 feet forward," a goal artifact is created at a position three feet in front of the robot.

Extracting features

To navigate through extended regions, Saphira uses a global map that contains imprecise spatial knowledge of objects in the domain, especially walls, doorways, and junctions of corridors. Using a map depends on reliable extraction of object information from perceptual clues, and we (as well as others) have spent many frustrating years trying to produce object interpretations from highly uncertain sonar and stereo signatures. (Drumheller 1985, Moravec and Elfes 1985). The best method we have found is to use extended-aperture sonar readings, perhaps augmented with depth information from the stereo system. As our robot Flakey moves along, readings from the side sonars are accumulated as a series of points representing possible surfaces on the side of the robot. This gives some of the resolution of a sensor with a large aperture along the direction of motion. By running a robust linear feature algorithm over the data, we can find wall segments and doorways with some degree of confidence.

Anchoring

Artifacts exist as internal representations of the environment. When the physical object that an artifact refers to is perceived by the sensors, we can use this information to update the position of the artifact with respect to the robot. This is necessary to guarantee that behavior using the artifact operates with respect to the actual object, rather than with respect to an *a priori* assumption. We call *anchoring* the process of (1) matching a feature or object hypothesis to an artifact, and (2) updating the artifact by using this perceptual information (see (Saffioti et al. 1993) for more on anchoring).

In Saphira, the structure of decision-making for the anchoring problem takes the following form: As features are perceived, Saphira attempts to convert them to object hypotheses, since these are more reliably matched than individual features. These hypotheses are matched against artifacts existing in the LPS. If they match against an artifact, the match produces information for updating (anchoring) the artifact's position. If not, they are candidates for inclusion as new artifacts in the map.

If an artifact that is in view of the perceptual apparatus cannot be matched against an object hypothesis, then Saphira tries to match it against individual perceptual features. This is useful, for example, when the robot is going down a hallway and trying to turn into a doorway. Only one end of the doorway is initially found because the other end is not in view of the side sonars. This information is enough to anchor the doorway artifact, and allow the robot to proceed with the door-traversing behavior.

Global cognitive state: OAA

To collect and deal with local cognitive pieces of information coming from robots, we decided to take advantage of our recent integration of Saphira as an agent within the Open Agent Architecture (OAA)™. It is a framework for constructing multiagent systems that has been used by SRI and clients to construct more than 20 applications in various domains.

The OAA uses a distributed architecture in which a Facilitator agent is responsible for scheduling and maintaining the flow of communication several of client agents. Agents interact with each other through an Interagent Communication Language (ICL), a logic-based declarative language based on an extension of Prolog. The primary job of the Facilitator is to decompose ICL expressions and route them to agents who have indicated a capability of resolving them. As communication occurs in an undirected fashion, with agents specifying what information they need, not how this information is to be obtained, agents can be replaced or added in a "plug and play" fashion.

Each agent in the OAA consists of a wrapper encapsulating a layer written in Prolog, C, Lisp, Java, Visual Basic, or Borland's Delphi. The knowledge layer, in turn, may lie on top of existing stand alone applications, and serves to map the underlying application into the ICL.

Features

Applying OAA to a multirobot system provides the following advantages:

- Distributed

Agents can run on different platforms and operating systems, and can cooperate in parallel to achieve a common task. For instance, some agents can be placed locally on each robot, while other services can be offered from more powerful workstations.

- Plug and play

Agent communities can be formed by dynamically adding new agents at runtime. It is as easy to have multiple robots executing tasks as it is to have just one.

- Agent services

Many services and technologies encapsulated by preexisting agents can easily be added, as resources, to our agents community. Useful agents for the robot domain would include database agents, map manager agents, agents for text to speech, speech recognition, and natural language, all directly reusable from other agent based applications.

- Mobile

The agent libraries are lightweight enough to allow multiple agents to run on small, wireless PDAs or laptops, and communications are fast enough to provide realtime response for the robot domain.

System design

The system we developed features a set of independent agents (including robots), able to communicate to perform cooperative tasks. A human operator can graphically monitor the whole scene and interactively control the robots. Figure 2 is a diagram of the complete system.

All involved agents are connected to the facilitator, registering their capabilities so that other members of the community can send them requests. This is the essential part of this architecture: agents are able to access each other's capabilities in a uniform manner. In the next paragraphs, we briefly describe the capabilities of the involved agents.

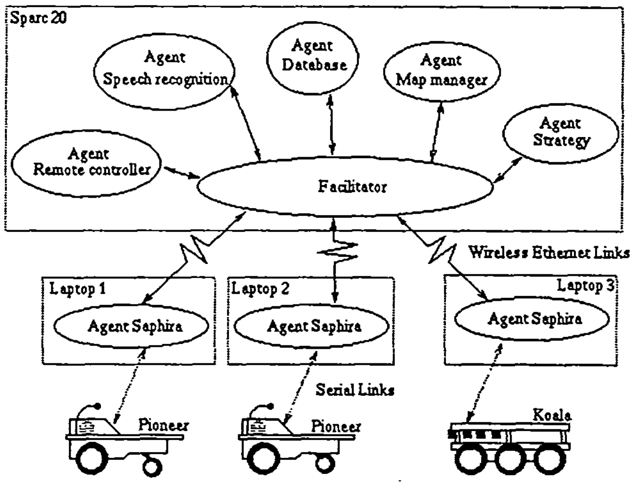

Figure 2 Organization of physical and software agents for the AAAI contest.

Database

Each robot agent provides information about its cognitive and physical states. The information includes

- •Position with respect to the robot's internal coordinate system
- •Robot movement status: stopped, moving forward, turning
- •Currently executing behaviors on the robot

An interesting problem is how two agents maintain a consistent coordinate system. Commands that are robot-relative, such as, "Move forward", are interpreted with respect to the robot's internal coordinate system. Other commands, such as "Go to office EK288," must be interpreted with respect to a common global framework. The *database agent* is responsible for maintaining a global map, and distributing this information to other agents when appropriate. Each physical robot has its own copy of the global map, but these copies need not be exactly alike. For example, an individual map may be missing information about an area the robot has no need to visit.

During movement, each robot keeps track of its global position through a combination of dead-reckoning (how far its wheels have moved) and registration with respect to objects that it senses. It communicates with the database agent to update its position about once a second, and to report any new objects that it finds, so they can be incorporated into the global database and made available to other agents. In this way, the database agent has available information about all of the robot agents that are currently operating.

Basic planner

The technology described in this paper was used in the "Hold a Meeting" event for the AAAI robotic contest organized in 1996. In this event, a robot starts from the Director's office, determines which of two conference rooms is empty, notifies two professors where and when the meeting will be held, and then returns to tell the Director. Points are awarded for accomplishing the different parts of the task, for communicating effectively about its goals, and for finishing the task quickly. Our strategy was simple: use as many robots as we could to cut down on the time to find the rooms and notify the professors. We decided that three robots was an optimal choice: enough to search for the rooms efficiently, but not too many to get in each other's way or strain our resources. We would have two robots searching for the rooms and professors, and one remaining behind in the Director's office to tell her when the meeting would be.

For this occasion, we designed a basic planner, the *strategy agent,* to control the coordinated movements of the robots, by keeping track of the total world stated and deciding what tasks each robot should perform at any given moment. While it would be nice to automatically derive multiagent strategies from a description of the task, environment, and robots, we have not yet built an adequate theory for generating efficient plans. Instead, we built a strategy for the event by hand, taking into account the various contingencies that could arise. The strategy was written as a set of coupled finite-state (FS) machines, one for each robot agent. Because the two exploring robots had similar tasks, their FS machines were equivalent. Figure 3 shows the strategies for these agents.

Note that the FS strategies are executed by the strategy agent, not the robots. Each node in the FS graph represents a task that the strategy agent dispatches to a robot, e.g., navigating to a particular location.

Cognitive state representation : multimodal user interface

An interesting problem is to combine human cognitive knowledge and its cognitive representation within the system presented. This step is realized by taking advantage of multimodal interfaces designed as agents, members of the OAA. For instance, if a robot becomes lost, it can query the facilitator to help relocalize. Currently, this means human intervention: the facilitator signals that a particular robot is lost, and asks for a new position for the robot. The state of each robot is displayed by the *map manager agent,* or *mapper.* All currently known objects in the database, as well as the positions of all robots, are constantly updated in a 2-dimensional (2D) window managed by this agent. Figure 4 shows the mapper's view of the database contents. Corridors, doors, junctions, and rooms are objects known to the mapper. A robot's position is marked as a circle with an arrow in it, showing the robot's orientation.

To correct the position of a lost robot, the user can point to a position on the map where the robot is currently located, or simply describe the robot's position through speech input. This integration of multimodal capabilities is one of the most useful features of the OAA architecture.

Currently, the system accepts either voice input or pen gestures. The interpretation of the gestures depends on context. For instance, when the robot is lost, the user can tell it where it is by drawing a cross (for the location) and an arrow (to tell the robot where it faces) on the map. Using 2D gestures in the human-computer interaction holds promise for recreating the paper-pen situation where the user is able to quickly express visual ideas while using another modality such as speech. However, to successfully attain a high level of human-computer cooperation, the interpretation of online data must be accurate and fast enough to give rapid and correct feedback to the user. The gesture recognition engine used in our application is fully described in (Julia and Faure 1995). There is no constraint on the number of strokes. The latest evaluations gave better than 96% accuracy, and the recognition was performed in less than half a second on a PC 486/50, satisfying what we judge is required in terms of quality and speed (Moran et al. 1996)

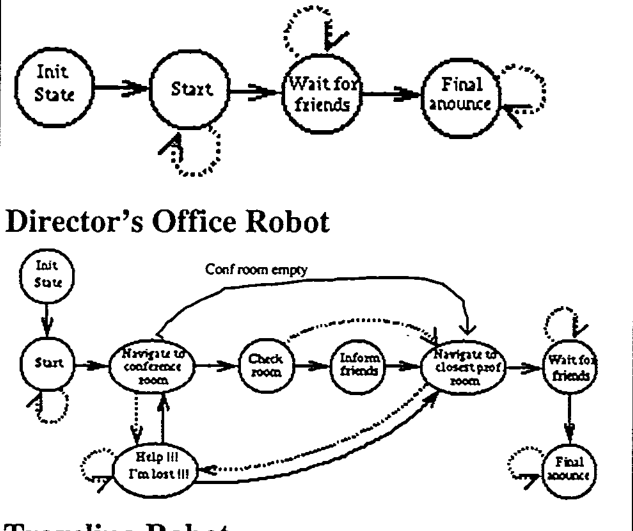

Figure 3 Finite State Strategy Machines for the two types of robots.

Given that our map manager program is an agent, the speech recognition agent can also be used in the system. Therefore, the user can talk to the system to control the robots or the display. For instance, it is possible to say ``Show me the director's room" to put the focus on this specific room, or ``robot one, stop", ``robot one, start", to control a given robot.

Using the global knowledge stored in the database, this application can also generate plans for the robots to execute. The program can be asked (by either a user or a distant agent) to compute the shortest path between two locations, build the corresponding plan, and send it to the robot agent. Plans are locally executed through Saphira in the robots themselves. Saphira returns a success or failure message when it finishes executing the plan, so the database agent can keep track of the state of all robots. In the figure, the plan is indicated by a line drawn from the robot to the goal point, marked by an "X".

Extracting wall and doorway features makes it easy to build a global map automatically, by having a Saphira-driven robot explore an area. The map is imprecise because of errors in the dead-reckoning system, and because the models for spatial objects are linear, for example, corridors are represented as two parallel, straight lines. As features are constructed they can be combined into object hypotheses, matched against current artifacts, and promoted to new artifacts when they are not matched. In practice, we have been able to reliably construct a map of the corridors in SRI's Artificial Intelligence Center, along with most of the doorways and junctions. Some hand editing of the map is necessary to add in doorways that were not found (because they were closed, or the robot was turning and missed them), and also to delete some doorway artifacts that were recognized because of odd combinations of obstacles.

A powerful way of entering spatial knowledge into the system consists in directly drawing a rough map of the robot's surroundings, letting the gesture recognition agent build a structured map of it, and finally storing it in the global database. The robot's navigation system (Saphira)

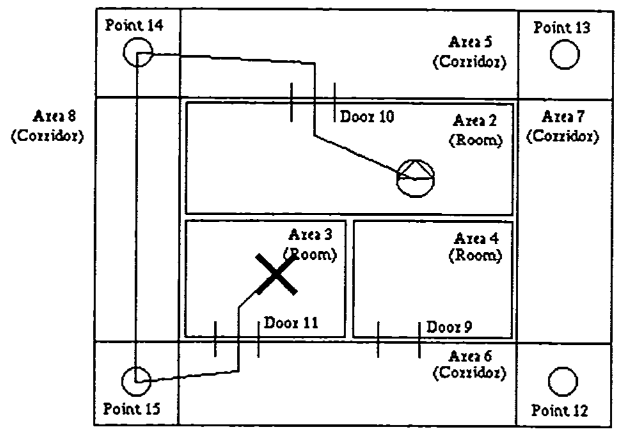

Figure 4 The mapping agent's view of the database.

will then use this information, confront it with real data coming from sensor inputs, and eventually correct it. This procedure could also be performed in the fashion of a feedback loop. The user draws a wall and artifacts, the robot starts looking for them in the real world and lets the human know about the real positions of these features, the user adds new objects to be seen by the robot, and so on.

Future work

Monitoring of agent activities

Adaptive behavior of agents and agent communities begins with effective strategies for detecting relevant changes in the operating environment. As such, *monitoring* will be an essential part of a multirobot framework. Monitoring will encompass a range of information and event types. Monitoring of resource usage will enable redirection of community activities if a critical resource becomes overloaded. Monitoring for the availability of new agents will enable off-loading of critical-path tasks that will improve overall productivity. Monitoring of interagent message traffic will provide insight into problem-solving strategies, which can be useful for evaluating strategies and for communicating to users the `state' of distributed problem solving. Finally, monitoring to evaluate progress through problem solving is critical for ensuring effectiveness of the overall agent community. Such monitoring will involve examination of success and failure in completing assigned tasks, and possibly consideration of partial solutions and measures of expected success/utility of agent activities. (For example, does it make sense for a robot to continue a given task if another agent has already produced an adequate solution for that task?)

User guidance for agent communities

We are interested in using agent technology to service human requests for information gathering and problem solving. For this reason, our framework will include a significant *user guidance* component that will enable humans to direct the overall process by which an agent community operates and to influence task delegation and individual robot behaviors.

Organizational structures for agents

The generality and flexibility of our framework should enable robot communities to dynamically reorganize themselves in response to critical events, to maximize robustness, resource usage, and efficiency. We will define and experimentally evaluate a range of organizational structures for agent communities to address issues such as the following.

Distributed facilitation: Facilitator agents in current-generation architectures often present a single point of failure, as well as a bottleneck for system communication. These problems can be addressed in two ways. First, the task delegation and management capabilities of a conceptually centralized facilitator can be transparently distributed among multiple agents, to increase the reliability and efficiency of the facilitation services. Second, conventions can be established for cooperation between facilitators in multifacilitator topologies (hierarchical or otherwise).

Communication links: It may be desirable to establish fixed communication links (such as peer-to-peer) links among agents that must frequently communicate.

References

Brooks, R. 1986. *A layered intelligent control system for a mobile robot.* Proceedings of the IEED Conference on Robotics and Automation.

Cohen, P. R., and Cheyer, A. J. 1994. *An Open Agent Architecture.* AAAI Spring Symposium.

Drumheller, M. 1985. *Mobile robot localization using sonar,* A.I. Memo 826, Massachusetts Institute of Technology.

Julia, L., and Faure, C. 1995. *Pattern recognition and beautification for a pen based interface.* In ICDAR'95, pages 58-63, Montreal, Canada.

Konolige, K., and Myers, K. 1998. *The SAPHIRA architecture: A design for autonomy.* In *Artificial Intelligence Based Mobile Robots: Case Studies of Successful Robot Systems,* D. Kortenkamp, R. P. Bonasso, and R. Murphy, eds., MIT Press.

Moran, D. B., Cheyer, J. C., Julia, L., Martin, D. L., and Park, S. 1996. *The Open Agent Architecture and Its Multimodal User Interface.* SRI Tech Note.

Moravec, H. P., and Elfes, A. E. 1985. *High resolution maps from wide angle sonar.* Proceedings of the 1985 IEEE International Conference on Robotics and Automation.

Saffiotti, A., Konolige, K., and Ruspini, E. 1995 *A multivalued logic approach to integrating planning and control.* Artificial Intelligence 76 (1-2).

Saffiotti, A., Ruspini, E. H., and Konolige, K. 1993. *Integrating reactivity and goal-directedness in a fuzzy controller. Proceedings* of the 2nd Fuzzy-IEEE Conference, San Francisco, CA.