Integration of Reactive Navigation with a Flexible Parallel Hardware Architecture

Thomas R. Collins, Ronald C. Arkin, Andrew M. Henshaw

School of Electrical Engineering and College of Computing
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

To demonstrate the flexibility and portability of both a schema-based software architecture and a message-passing hardware architecture, the two were integrated within a very short period to be used in a mobile robot competition. The experience confirmed the advantages of onboard computational capability in mobile systems.

1 Introduction

Autonomous machines with sensory, manipulative, and locomotive capabilities are a significant class of intelligent systems holding great promise for performing hazardous or mundane tasks. Although much work has been performed with isolated aspects of intelligent machines, including vision, sonar, manipulator control, and knowledge-based reasoning, the algorithms are often not considered within the context of a complete machine. In many prior efforts, both software architectures and hardware architectures have been developed to meet the requirements of specific projects, with little regard to reusability in other applications. Often, experimental systems are not robust, failing due to relatively minor environmental variations or task redefinitions. This paper describes the integration of two separate efforts to address these problems. One is a reactive software architecture which has been demonstrated to perform a variety of tasks well, and the other is a targeted, yet flexible, computer architecture that provides modularity and expandability.

2 Background

Most of the earliest work with intelligent machines relied on direct programming in declarative languages [22, 23, 24]. This resulted in software architectures that could only accommodate certain situations, and even then only in a "do this, then do this" fashion. There has been a gradual trend toward *reactive* software architectures, which combine relatively simple behaviors to implement complex tasks. Since these simple components are designed to respond to general stimuli in "common-sense" fashion, greater robustness is achieved, even when a higher level of *deliberative* behavior is added.

From a hardware architectural standpoint, many early autonomous machines relied on offboard control, since the necessary computers were so large. Then, as microprocessors became widely available to provide onboard intelligence in relatively small packages, many more projects began. Even today, however, many mobile robots depend on detached workstations, because of the increasing computational demands and the need for sophisticated user interfaces. It has become increasingly evident that autonomous machines must incorporate multiprocessor architectures, but it is not quite clear what form the architectures should take.

The majority of these specialized multiprocessor architectures have taken a hierarchical form, often a simple tree [14, 22, 24]. In the context of an intelligent machine, the processing tasks closest to the environment (i.e., the "low-level" tasks) tend to map most obviously into a tree. Higher levels of thought prefer to describe both perceptions and actions in concise symbolic terms, while environmental interactions tend to involve large amounts of data, often to or from loosely-coupled subsystems. It is thus advantageous to have multiple perceptual processes and multiple control processes running simultaneously, each having extensive interaction with the environment, but little interaction with each other. Communication with higher levels is less frequent, and the messages tend to contain condensed information in a symbolic format. Intermediate levels of processing act to integrate data (on the sensory side) or to coordinate simple tasks (on the control side). In this simple paradigm, only one highest-level, or reasoning, task is required. Unfortunately, this process usually forms a computational bottleneck, since it is an integral part of every sensor/control path.

More recently, hierarchical designs have emphasized connection across the hierarchy, as in the NASA/NBS standard reference model for telerobot architectures

(NASREM) developed by Albus [1] and the Multiresolutional Control Architecture of Meystel [21]. These architectures allow nested control loops via distinct task levels within the hierarchical structure, providing higher bandwidth for low-level tasks that require shorter response times. Each higher level in the hierarchy thus provides for increasing levels of abstraction in perception, reasoning, and control, and each maintains a model of the world appropriate for its purposes. No clear consensus exists, however, for the number and type of levels to use in such architectures.

The hierarchical viewpoint is increasingly being challenged by reactive approaches. The concept of schemas, as typified by Lyons [20] and Arkin [2, 3], implements sensor-effector connections in a more flexible fashion. A schema is a pattern of behavior exhibiting a stimulus-response characteristic. Typically, each schema monitors only a portion of the available input data and produces an output which may have to be combined, superimposed, or otherwise reconciled with other outputs. Schemas may communicate with other schemas or with sensors and effectors. Normally, an intelligent machine would create instances of predefined schemas as necessary to produce more complex behaviors. Such apparent complexity arises both from the ability of schemas to use other schemas and from the parallel actions of independent schemas.

Another significant reactive approach is the subsumption architecture developed by Brooks [9, 10, 11]. Multiple levels of competence are defined, connecting input and output in a layered system. Higher levels of competence inhibit or subsume all lower levels, and the hardware usually provides direct support for this subsumption characteristic by implementing each level within its own processing subsystem. This structure allows a machine to be developed in stages, building each level on top of a machine that already functions at some given degree of competence. Another advantage is that the lower levels still exhibit useful behaviors that are activated in the absence of any inhibition from above. For example, a low-level obstacle avoidance behavior is still useful even when a path planner exists to provide intermittent goals.

In spite of the emergence of these promising approaches, there is still a considerable amount of disarray in the overall architectural scene. The key limitations of much of the previous work, including both hardware and software considerations, have been:

- reliance on offboard computational facilities and radio communication,
- *ad hoc*, inflexible hardware architectures,
- lack of inherent support for parallelism,
- awkward development environments, and
- lack of portability.

These limitations hamper the development of reusable, modular hardware and software components, and they have thus slowed the development of a significant commercial market in sophisticated autonomous machines.

3 AuRA – the Autonomous Robot Architecture

AuRA is a hybrid architecture encompassing aspects of both deliberative and reactive control. It consists of 5 major subsystems:

- Perception – charged with collecting and filtering all preliminary sensory data.
- Cartographic – concerned with maintaining long-term memory (*a priori* models of the environment), short-term memory (dynamically acquired world knowledge), and models of spatial uncertainty.
- Planning – consists of a hierarchical planner (the deliberative component) and the motor schema manager (the reactive component).
- Motor – the interface software to the specific robot to be controlled.
- Homeostatic – a component concerned with dynamic replanning in light of available internal resources [7].

The overall architecture has been described in detail elsewhere. The reader is referred to [4, 6] for more information.

The hardware migration to ANIMA thus far has been concerned with the reactive and perceptual components of the system which run within the confines of the motor schema manager. Figure 1 presents the logical relationships between the varying schemas which constitute this portion of AuRA.

Action-oriented perception forms the underlying philosophy for channeling sensory information to the motor schemas (behaviors) [5]. Only the information that is essential to a particular motor behavior is transmitted to it,

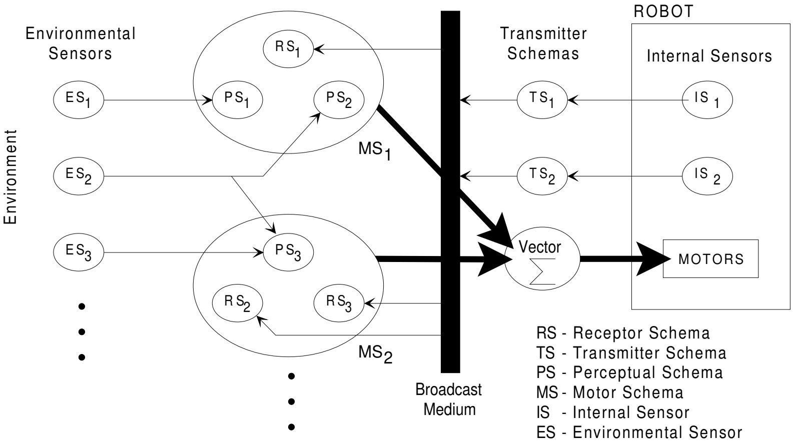

Figure 1. Inter-schema relationships.

essentially on a need-to-know basis. The message-passing paradigm found in ANIMA is well-suited for this type of information flow.

Each of the active motor schemas generates a velocity vector in a manner analogous to the potential fields method [18, 19] with the individual results being summed, normalized and transmitted to the robot for execution. The speed of operation is highly dependent on the rate of processing of incoming sensory data. The parallelism found in the transputer implementation described below is a natural match for this aspect of the AuRA architecture.

4 ANIMA hardware architecture

We have developed a flexible, real-time platform for the development of AuRA and other software architectures. The skeleton of our hardware architecture, ANIMA (Architecture for Natural Intelligence in Machine Applications), has been developed from basic principles. It incorporates a triad of basic systems, just as a conventional computing system includes input, output, and processing subsystems. This fundamental triad of subsystems carries over into the architecture of an intelligent machine, but a more general interconnection pattern is required. The addition of a communication channel between the input subsystem and the output subsystem allows the machine to exhibit *reflexive behaviors*. Such behaviors are analogous to reflexes in biological systems, where the communication channel is implemented by structures within the spinal cord and lower brain. While it would be possible to develop an autonomous machine without such a channel, it would not take advantage of the localized intelligence within the input and output subsystems. The resulting increase in computational load on the processing subsystem would result in slower response time.

Clearly, reflexive behaviors are virtually “hard-wired” into the system, and their implementation is best reserved for behaviors that:

- must be performed reliably and quickly, usually to avoid danger to the machine or to humans,
- require little or no integration of information from multiple input systems, and
- although primitive, usually produce an effect more desirable than if no action at all had been taken.

The deliberative component controlling the input and output subsystems is called the *Reasoner*. A major aspect of this reasoning capability is the need to maintain some sort of world model based on sensory input, at least for anything more than basic reactive behavior.

The vast majority of intelligent machine research has assumed that the input/output devices, just as in a conventional computer, are largely independent in their low-level operation (at or below the level of the device driver). For input devices, the combination of these independent streams of data has often been referred to as *sensor integration*, and we include a process, called the *Integrator*, to perform this task. On the output side of the structure, the most appropriate term is *coordination*, although the specific definition varies considerably in the literature. Corresponding to the *Integrator*, we include a process called the *Coordinator*.

These additional parallel processes are illustrated in Figure 2. The independent sensor subsystems are called *logical sensors*, in much the same sense as those of Henderson [15] or Crowley [13]. At this point, a logical sensor is best thought of as a combination of a physical sensor, capable of estimating some property of the environment or the machine itself, and a generalized device driver. The extension of this concept to the *logical effector* is straightforward. Taken together, logical sensors and logical effectors are called *logical devices*. A single logical device can be composed of multiple physical devices, with appropriate drivers. This would be desirable in cases where the physical devices were virtually identical (except perhaps in physical location, scaling, or some other trivial factor), allowing the main driver to give the appearance of a single effective logical device.

These processes (*Reasoner, Integrator, Coordinator,* and representative logical devices) have been described using the notation of Communicating Sequential Processes (CSP) [16, 17]. By combining them into a system, it is possible to construct a proof showing that the system is free of deadlock [12]. By following basic design principles at the logical device layer and the *Integrator/Coordinator* layer, we provide a means of fault isolation to individual logical devices.

An implementation of this architecture is being developed based on the Inmos T800, a member of the transputer family

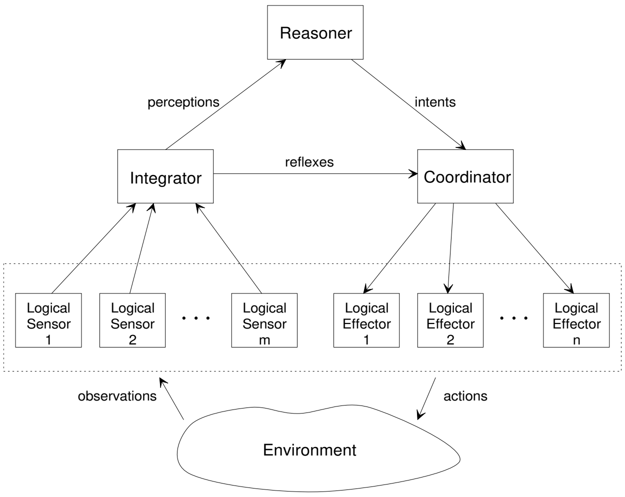

Figure 2. ANIMA structure.

of microprocessors developed for parallel processing. Each transputer provides four high-speed serial links for the required processor interconnection. While several languages are supported, Occam is the most effective for parallel processing, providing constructs that implement the important CSP operators.

The ANIMA architecture requires no parallel data busses or backplanes of any kind. Instead, it consists of modular components connected only by high-speed serial links. This allows the processors to be distributed to any convenient locations within the intelligent machine.

Relatively early in the architectural development, the entire structure was simulated on a single transputer and on multiple transputers in order to verify its operation. A fundamental premise of the simulation was that most of the processes would be directly portable to a real machine. Specifically, by making reasonable models of the environment, sensors, and effectors, one can use essentially the same *Integrator*, *Coordinator*, and *Reasoner* processes as would be used on a real machine [12]. The simulated machine wandered through a simple world with walls and obstacles, using simulated sonar and touch sensors. Sensors and effectors were deliberately modeled as being imperfect, and the machine (as part of its *Reasoner*) had to maintain its own model of the simulated world. The initial *Reasoner* was essentially a schema-based implementation, but hierarchical and subsumption versions have also been tested.

5 Case study – "Buzz"

AuRA and ANIMA were first brought together in the development of a machine called "Buzz," to compete in the first robot exhibition and competition sponsored by the American Association for Artificial Intelligence (AAAI). The competition stressed the ability of mobile robots to explore an arena, avoid static or moving obstacles, locate goals, and visit goals in specified order. Many of the basic reactive behaviors that were needed had already been developed (using AuRA) within the Mobile Robotics Laboratory of the College of Computing at Georgia Tech. The previous work had been done on an older robot, and a new one was made available for the competition by Denning Mobile Robotics (Wilmington, Massachusetts). Much of the required programming effort would be to modify communication routines for the new robot, to develop new perceptual schemas for previously-uninteresting phenomena, and to combine the available schemas in ways appropriate for the competition tasks.

Realizing that the use of radio communication was both a reliability issue and a "showmanship" limitation, the team also added another development task: to use onboard computation for at least one phase of the competition. The ANIMA structure, although never before used on an actual robot, had been prototyped and used in simulations of both hierarchical and reactive robot systems. Most of the required work was to package it for the Denning robot, add an appropriate interface to the robot, and port all of the schemas *as they were developed*. With only about four months to complete all of these tasks, including the basic schema and communication development, it was clear that this would be a test of the flexibility and portability of both the hardware and software architectures.

The competition included three phases. In the first phase, each robot was to navigate the arena cluttered with obstacles without hitting anything, including the human judges. In the second phase, ten poles (labeled according to the needs of each robot) had to be recognized and visited, if possible, within a designated period. In the final phase, three of the previously-visited poles were designated to be visited in order. Additional information about the competition may be found in [8].

5.1 Robot description

The Denning MRV-3 is a three-wheeled cylindrical robot intended for general-purpose use, mainly in research. All three wheels turn simultaneously, providing (approximately) the ability to turn in place. The body itself does not rotate, except for gradual precession resulting from non-uniform slippage of the wheels against the floor. Twenty-four sonar sensors are equally-spaced around the body, as are six contact-switch bumpers. A single CCD camera was added to the standard configuration for use in the second and third phases of the competition. This camera was mounted on the top plate, which rotates to point in the direction of travel (along with the wheels). An infrared beacon detector was also available, but was not used during the competition.

The transputer architecture used for this implementation of ANIMA utilized five processors and an RS-232 interface spread over six TRAMs (integrated transputer daughterboards). The TRAMs were mounted on an PC-bus host board within a specially-packaged IBM-PC compatible system, complete with an electroluminescent display, a floppy disk drive, and a ruggedized hard drive. We have designed some other implementations which provide more flexibility with regard to usage of the processor links, but this system was more than adequate for the required tasks.

Although the performance of ANIMA benefits from separate high-speed channels to each physical sensor and effector, the Denning MRV-3 platform (like most commercial mobile robots) provides a single standard interface, in this case an RS-232 port. All communication with the sonar, infrared detectors, bumper switches, and motor controllers had to be multiplexed through this port.

The ANIMA hardware, of course, was restricted to using the onboard power sources. Since the robot may only function reliably for several hours even without the added burden of multiple transputers and a PC host, it was important that ANIMA not consume any more power than necessary. Even with the disk drives and electroluminescent display active, the ANIMA system and host required only about 100 watts and did not significantly affect the battery life of the system.

5.2 Parallel structure

The utilization of the five processors is shown in Figure 3. AuRA's motor schemas and much of its perceptual schemas were included in the *Reasoner* process (which can easily be split among additional processors as necessary). Some aspects of the perceptual schemas (sensor data processing, mostly) were included within the appropriate logical sensors.

Because of the relatively low processing demands placed on the *Integrator* and *Coordinator*, these were combined onto a single processor, and messages to all logical devices were multiplexed on a single channel. These logical devices also ran as parallel tasks on a single processor, since no especially sophisticated processing was done at this level. Provision was made for inclusion of a separate processor (actually, a group of processors) to perform vision, using the remaining link from the logical devices processor. Although speech output was not used in Buzz, we have the appropriate logical effector to add it at any time, as indicated in the figure.

The *Environment* process actually serves a dual role. In normal operation, it passes messages along to the RS-232 handler process. In simulation mode, it intercepts commands to the robot and emulates the behavior of the robot in a grid-based environment, passing back sonar and bumper data when requested. An additional processor (not shown) is used in simulation mode just to provide a graphical display of the simulation status. The impact of this is that simulation capability is built into the real code – no porting is required to keep the simulation current relative to the actual robot software. Of course, the usefulness of any simulation depends on its fidelity. This organization allows the simulator, as a separate parallel process, to be enhanced at any time. We found that the simulation provided good qualitative results with regard to new robot behaviors which were subsequently tested on the actual robot.

5.3 Performance

Throughout the porting process, we were pleased with ANIMA's ability to perform sensor processing in the background. It was also possible to continuously keep track of the time between robot responses, providing the basis for a dead-man switch if the robot ceased communicating for any reason. The ANIMA-controlled system was able to negotiate obstacle-strewn areas about 50% faster than the Sun-controlled system, mainly because of the decreased latency of sonar data and the managed use of the RS-232 channel. We did not have sufficient development time to fully utilize the motor capabilities of the MRV-3, but we felt that additional performance improvements were easily possible. Much more benefit can be derived from the parallel structure as complex sensors like vision, more-sophisticated motor control algorithms, and additional schemas are added.

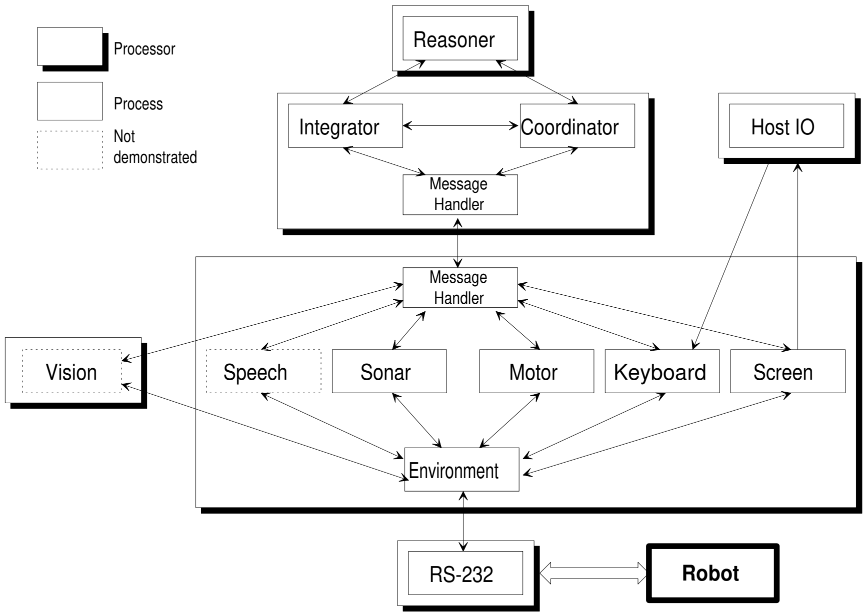

Figure 3. Partitioning of tasks on Buzz.

Phase 1 of the competition was intended primarily as a means of weeding out any robots which could not safely navigate within the arena in the presence of human beings. The judges deliberately stepped in front of the robots and corralled them into tight spaces, and Buzz performed satisfactorily. At about this time, some of the other robot teams were experiencing communication problems, since most were using the same frequencies for digital commands and/or video data. As one of the relatively few entries with all processing performed onboard (the eventual winner, the University of Michigan's CARMEL, was another), Buzz was immune from these problems in Phase 1. For the same reason, Buzz was also able to perform in combined demonstrations for the news media and the public, along with CARMEL and SRI's "Flakey."

5.4 Conclusions

Based on the relative ease in which the AuRA software was ported to the ANIMA architecture, it was clear that both components were sufficiently flexible and portable. Some of the specific features which aided this process were:

- the integrated development environment
- the inherent support for parallelism
- the use of generic proven CSP models
- the inclusion of simulation as a removable process

In phase 2 of the AAAI competition, the offboard Sun computer and radio link performed well, and Buzz ended up in second place, but radio problems in phase 3 limited us to a fifth-place finish overall. This seemed to indicate that a full port of the vision schemas would have improved our overall standing, since the machine would have been immune to radio interference. Since visual data could have been processed at much higher frame rates, it would have been possible to perform nearly continuous tracking, also improving Buzz's performance.

6 Future work

We have begun to adapt this system to a practical application for the Savannah River Site of the Department of Energy. Using another Denning robot, and adding onboard vision, we are building a prototype survey vehicle to monitor the condition of stored radioactive waste. As part of this effort, we intend to investigate the performance of ANIMA with a robot which has dedicated channels to the sonar and motor systems. This would eliminate the need to multiplex the data on a single RS-232 line, and overall system performance would improve considerably. ANIMA will also be used in all 3 phases of the 1993 AAAI competition.

As additional motor schemas are included, they will be placed in a parallel configuration, utilizing additional processors as necessary for the *Reasoner*. Eventually, we would like to make performance comparisons with other software architectures on the same hardware platform. Such comparisons would provide insight into the type of applications best suited for differing software architectures.

References

1. Albus, J. S., McCain, H. G., and Lumia, R. "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," NIST Tech. Rep. 1235, 1989.
2. Arkin, R. C. "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior," *IEEE 1987 Intl. Conf. on Rob. and Aut*. 1987, pp. 264-271.
3. Arkin, R. C., and Murphy, R. R. "Autonomous Navigation in a manufacturing environment," *IEEE Trans. Rob. Aut.,* vol. 6, no. 4 (Aug. 1990), 445-454.
4. Arkin, R.C., "Towards Cosmopolitan Robots: Intelligent Navigation of a Mobile Robot in Extended Man-made Environments," Ph.D. Diss., COINS Tech. Rep. 87-80, U. Mass., 1987.
5. Arkin, R.C., "The Impact of Cybernetics on the Design of a Mobile Robot System: A Case Study," *IEEE Trans. Sys. Man Cyber.*, vol. 20, no. 6, Nov/Dec 1990, pp. 1245-1257.
6. Arkin, R.C., Riseman, E. and Hanson, A., "AuRA: An Architecture for Vision-based Robot Navigation," *Proc. of the 1987 DARPA Image Understanding Workshop*, Los Angeles, CA, pp. 417-431.
7. Arkin, R.C., "Homeostatic Control for a Mobile Robot: Dynamic Replanning in Hazardous Environments," to appear in *Journal of Robotic Systems*, vol. 9, no. 2, (1992b).
8. Arkin, R. C. et al., "Buzz: An Instantiation of a Schema-Based Reactive Robotic System," to appear in *Proc. International Conference on Intelligent Autonomous Systems: IAS-3* (Pittsburgh, PA), Feb. 1993.
9. Brooks, R. A. "A Layered Intelligent Control System for a Mobile Robot," *Rob. Research: Third Intl. Symp*. 1985, pp. 365-372.
10. Brooks, R. A., and Connell, J. H. "Asynchronous Distributed Control System for a Mobile Robot," *Proc. SPIE*, vol. 727, 1986, pp. 77-84.
11. Brooks, R. A. "A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control," *Proc. IEEE Intl. Conf. Rob. Aut*. 1987, pp. 106-110.
12. Collins, T. R. "A Computer Architecture for Implementation within Autonomous Machines," Ph.D. Diss., Ga. Inst. of Technology, 1992.
13. Crowley, J. L. "Dynamic World Modeling for an Intelligent Mobile Robot" *Proc. 1984 Intl. Conf. Patt. Rec*. 1984, pp. 207-210.
14. Everett, H. R., and Gilbreath, G. A. *ROBART II: A Robotic Security Testbed*, Naval Ocean Sys. Center Tech. Doc. 1450, San Diego, CA, January 1989.
15. Henderson, T., and Shilcrat, E. "Logical Sensor Systems" *J. Robotic Systems,* vol. 1, no. 2 (1984), 169-193.
16. Hoare, C. A. R. "Communicating Sequential Processes" *Communications of the ACM*, vol. 21, no. 8 (Aug. 1978), 666-677.
17. Hoare, C. A. R. *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
18. Khatib, O., "Real-time Obstacle Avoidance for Manipulators and Mobile Robots", *Proc. IEEE Int. Conf. Rob. Aut.*, p. 500-505, St. Louis, 1985.
19. Krogh, B., "A Generalized Potential Field Approach to Obstacle Avoidance Control", *SME - RI Technical Paper* MS84-484, 1984.
20. Lyons D. M., and Arbib, M. A. "A Formal Model of Computation for Sensory-Based Robotics" *IEEE Trans. Rob. Aut.,* vol. 5, no. 3 (June 1989), 280-293.
21. Meystel, A. "Multiresolutional Feedforward/Feedback Loops" *Proc. 1991 IEEE Intl. Symp. Intelligent Control*. Aug. 1991, pp. 85-90.

22. Moravec, H. P. "The Stanford Cart and the CMU Rover," *Proc. of the IEEE*, vol. 71, no. 7 (July 1983), 872-884.
23. Raphael, B. *The Thinking Computer: Mind Inside Matter*, W. H. Freeman, San Francisco, 1976.
24. Thompson, A. M. "The Navigation System of the JPL Robot," *Proc. Fifth Intl. Joint Conf. on Artificial Intelligence*. 1977, pp. 749-757.