11 The Whole Iguana

Rodney A. Brooks

11.1 Introduction

There has been a flurry of work all over the world in the last three years on autonomous mobile robots. In this chapter we present the philosophy behind the MIT AI Lab mobile robot project and report on the significant departures this work has made from the bulk of current mobile robot work. This is evident right from our starting premise; we set out to build mobile robots in order to study the general problem of how to build artificially intelligent beings; we do this by building robots that autonomously carry out a number of generic task-achieving behaviors without human control or commands. Together, through the dynamics of the world, these behaviors combine to produce useful high-level operation.

Moravec (1984) eloquently argued that mobility has many times been the main impetus that has led to the evolution of intelligence. Dealing in real time with a dynamic and perhaps hostile environment forces the issue of understanding the world and reacting to it appropriately.

Our goal is to understand intelligence and to build intelligent beings. It seems very easy to cheat accidentally in building a sedentary being or program. It is so easy, even unintentionally, for a human to abstract away the truly hard problems. By making sure our programs run a real mobile robot in a dynamic unstructured domain, we believe there is less chance for oversimplifying the problems. More important, however, we have come to realize that this approach makes explicit to us what the real problems are in intelligence. Many of the issues that have been studied over the last 25 years turn out to be irrelevant for a real artificial being in a real world. The true details of interacting with the world are not the same as abstract thinking has led many workers in Artificial Intelligence to believe. Likewise, many problems tackled traditionally by robotics researchers tend to be largely irrelevant if the world is dynamic and unstructured.

Other current mobile robot projects tend to join the worlds of traditional AI with that of traditional robotics. We believe progress suffers from two diseases in these projects. First, from the AI tradition, that a world model must be built and maintained. Second, from the robotics tradition, that this model must be precise.

While we find many useful ideas in these two traditions of research, we maintain a somewhat different mindset of what the purpose of each

algorithm or module is. The result is a radically different control structure, with radically different implementation possibilities.

The key thrust of our research is how to create a complete creature that can exist in a dynamic world, completely autonomously for long periods of time.

This chapter, then, is an answer to a challenge posed by Daniel Dennett (1978) in a commentary, titled *Why Not the Whole Iguana?*, on a paper by Pylyshyn (1978). Dennett challenged researchers to build a simulation of a complete creature. He suggested that simulating a person was too hard a problem but perhaps a starfish or a turtle was in reach. But that would require understanding everything about a turtle and its environment. Therefore he suggested *making up* a creature and its environment; perhaps a Martian iguana. His mistake was to suggest simulating the environment. It requires a great deal of work to simulate a rich enough environment to make the problems for the creature realistic; more work, I claim, than in simulating the creature itself. Thus in this chapter we use a real environment, and make up just the creatures—complete creatures, *whole iguanas.*

11.2 Approach

Over the past two years we have developed a new approach to exploring mobile robot control systems. In outline it is

- We decompose the problem based on parallel task-achieving behaviors, rather than on the traditional axis of information-processing modules. See figure 11.1.
- We build each individual task-achieving behavior from a few asynchronous simple finite state machines. Each has inputs and outputs; they are wired together with a fixed topology over low-bandwidth communications lines.
- We build one task-achieving behavior at a time. As we build each one we test and debug it extensively. The idea is that we then freeze that layer or behavior, running on its own hardware, and never have to develop it further. To add capabilities to the robot we simply add new layers of control.
- We are moving toward all onboard computation on our robots including vision. In particular, to achieve real-time vision we are concentrating on low resolution imaging systems.

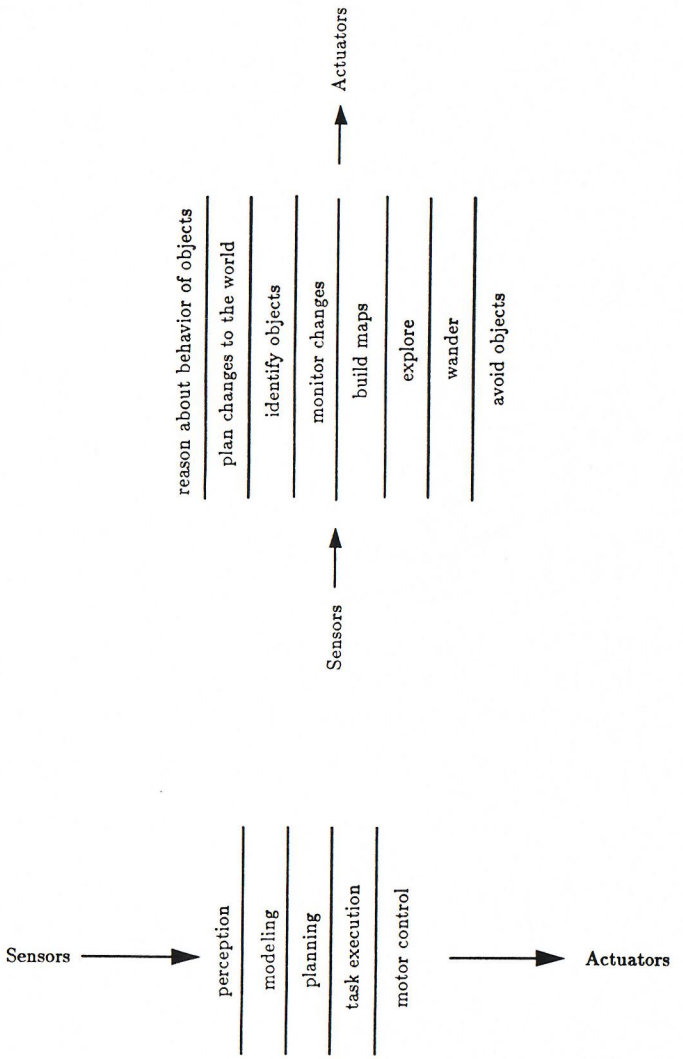

Figure 11.1
Traditionally mobile robot control systems have been decomposed for synthesis based on information-processing functions as on the left. By contrast we choose task-achieving behaviors as our primary direction for decomposing the control system into individually synthesizable components.

11.2.1 The Consequences

Our synthesis method has lead to a number of consequences very different from those found in other mobile robot projects. For instance:

- The very first test of our first mobile robot was in a dynamic environment where it had to avoid collisions with people moving about.
- There is no locus of central control, no central data structures, and no global plan that the robot is following at any time. In particular, there is no way to tell the robot to do something. Instead it senses the world and does what is in its nature to do!

11.2.2 The Approach

The key idea of levels of task-achieving behaviors is that we can build layers of a control system corresponding to each level of competence and simply add a new layer to an existing set to move to the next higher level of overall competence.

We start by building a complete robot control system that achieves a lowest-level task. It is debugged thoroughly. We never alter that system. We call it the zeroth-level control system. Next we build another control layer, which we call the first-level control system. It is able to examine data from the lower-level system and is also permitted to inject data into the internal interfaces of that level, *suppressing* the normal data flow. This layer, with the aid of the lower level, achieves some new task. The lower layer continues to run unaware of the layer above it that sometimes interferes with its data paths.

The same process is repeated to achieve higher levels of competence. See figure 11.2. We call this architecture a *subsumption architecture.*

In such a scheme we have a working control system for the robot very early in the piece—as soon as we have built the first layer. Additional layers can be added later, and the initial working system need never be changed.

But what about building each individual layer? Do we not need to decompose a single layer in the traditional manner? This is true to some extent, but the key difference is that we do not need to account for all desired perceptions, processing, and generated behaviors in a single decomposition. We are free to use different decompositions for different sensor-set task-set pairs.

We have chosen to build layers with a set of small processors that send messages to each other. A prototypical such machine is shown schematically in figure 11.3.

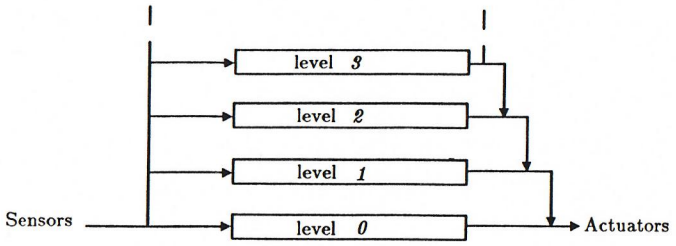

Figure 11.2
Control is layered with higher-level layers subsuming the roles of lower-level layers when they wish to take control. The system can be partitioned at any level, and the layers below form a complete operational control system.

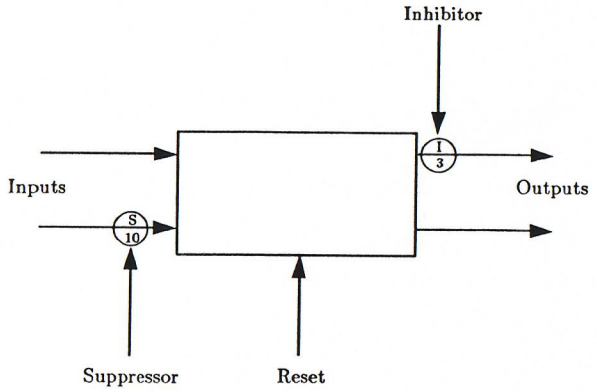

Figure 11.3
A module has input and output lines. Input signals can be suppressed and replaced with the suppressing signal. Output signals can be inhibited. A module can also be reset to state NIL.

Each processor is a finite state machine with the ability to hold some data structures. Processors send messages over connecting "wires." There is no handshaking or acknowledgment of messages. The processors run completely asynchronously, monitoring their input wires, and sending messages on their output wires. It is possible for messages to get lost—it actually happens quite often. There is no other form of communication between processors; in particular, there is no shared global memory.

All processors (which we refer to as modules) are created equal in the sense that within a layer there is no central control. Each module merely does its thing as best it can.

Inputs to modules can be suppressed and outputs can be inhibited by wires terminating from other modules. This is the mechanism by which higher-level layers subsume the role of lower levels. Figure 11.4 shows three layers of control that we have run on our mobile robots.

The lowest-level layer of control makes sure that the robot does not come into contact with other objects. If something approaches the robot, it will move away. If in the course of moving itself it is about to collide with an object, it will halt. Together these two tactics are sufficient for the robot to flee from moving obstacles, perhaps requiring many motions, without colliding with stationary obstacles. The combination of the tactics allows the robot to operate with very coarsely calibrated sonars and a wide range of repulsive force functions. Theoretically, the robot is not invincible, of course, and a sufficiently fast moving object, or a very cluttered environment might result in a collision. Over the course of a number of hours of autonomous operation, our physical robot (see below) has not collided with either a moving or fixed obstacle. The moving obstacles have, however, been careful to move slowly.

The next layer of control, when combined with the lowest, imbues the robot with the ability to wander around without hitting obstacles. This control level relies to a large degree on the zeroth level's aversion to hitting obstacles. In addition it uses a simple heuristic to plan ahead a little in order to avoid potential collisions that would need to be handled by the zeroth level.

The last level is meant to add an exploratory mode of behavior to the robot, using visual observations to select interesting places to visit. A vision module finds corridors of free space. Additional modules provide a means of position servoing the robot along the corridor despite the presence of local obstacles on its path (as detected with the sonar sensing system).

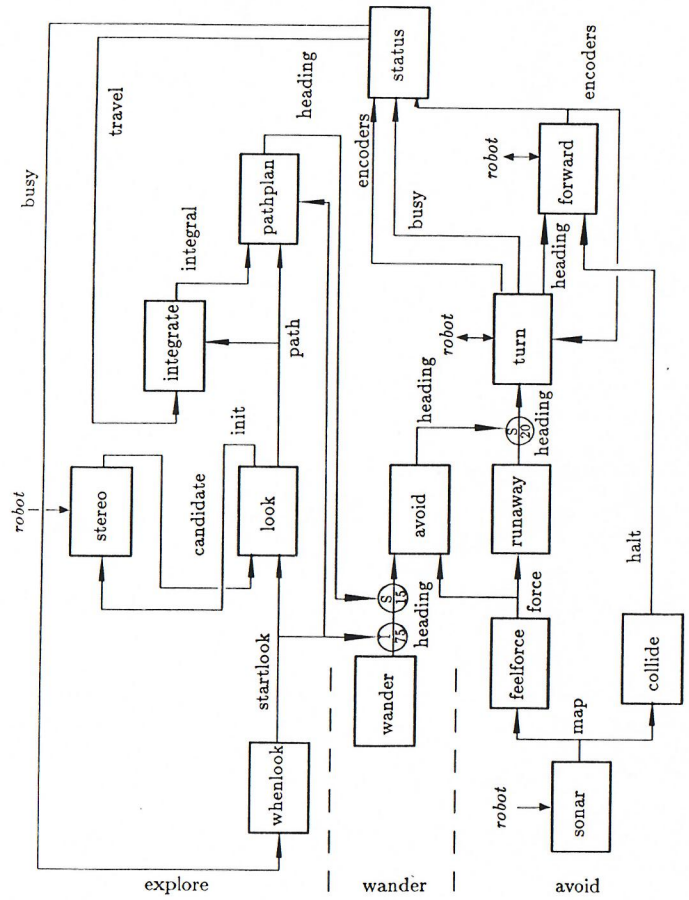

Figure 11.4
We wire finite state machines together into layers of control. Each layer is built on top of existing layers. Lower-level layers never rely on the existence of higher-level layers.

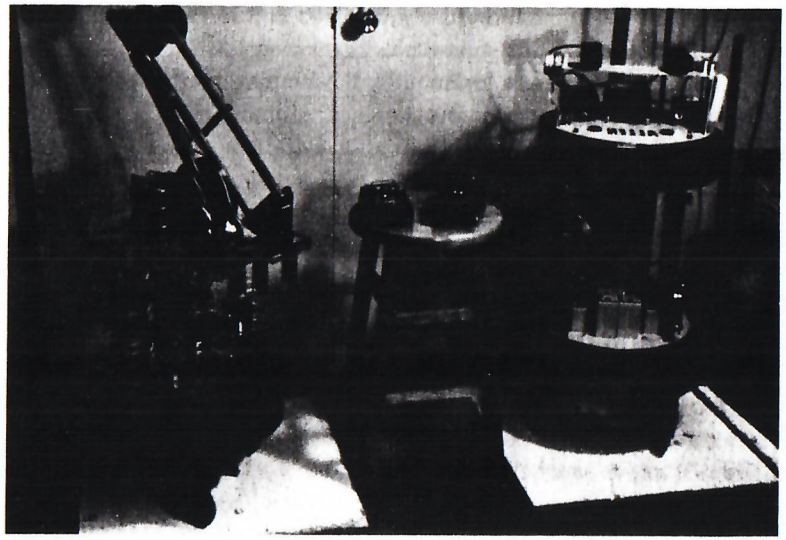

Figure 11.5
The four existing MIT AI laboratory Mobots. Rightmost is the first built Allen, which relies on an offboard Lisp machine for computation support. The leftmost one is Herbert, shown with a 24-node CMOS parallel processor surrounding its girth. In the middle are Tom and Jerry, based on a commercial toy chasis, each with a single PAL (programmable array of logic) as its controller.

The lower-level two layers still play an active role during normal operation of the second layer. (In practice, we have so far only reused the sonar data for the corridor finder, rather than use stereo vision.)

11.3 Some Prototypes

We have built four mobile robots, shown in figure 11.5, named Allen, Herbert, Tom, and Jerry. A fifth robot, named Seymour, is currently under construction. Allen, the first robot, has mostly offboard computers and has been used extensively in experiments for over a year. Herbert, is totally self-contained with an on-board parallel processor. It is just now starting to move around under subsumption control and will be our main experimental workhorse for at least the next year. Tom and Jerry were a diversion to demonstrate the idea of compiling down the control architecture to very

simple on-board computation hardware. Tom and Jerry each have only a single 256 gate PAL (programmable array logic) as their brains, and have been operational for a few months. Seymour is an all passive-sensor robot, again with onboard computation.

11.3.1 Allen the Robot

Our first mobile robot consists of a commercial three-wheeled base that can turn in place, a central cardcage for communications and on-board processors, and a sensor platform mounted above that. The cardcage and sensor platforms are coupled to the wheels so that they always point "forward."

The bulk of the processing is done off-board on a Lisp machine that supports the subsumption architecture by simulating individual finite state machines and wires connecting them. Onboard are two CMOS microprocessors: one to servo the drive motors and one to run the sensors and handle off-board communications via a 1,200-baud serial cable.

The sensor platform has a ring of 12 Polaroid sonars and two Sony CCD cameras. All of our real-time experiments to date have only used the sonars. They are arranged in a symmetric circular ring. We ping opposite sensors in parallel, and thus get a complete set of 12 readings in under half a second. The cameras have been used in some static experiments; they are mounted with a fixed parallel geometry on a tilt head (pan can be achieved by spinning the robot base in place). The cameras are fed through a switching box and then to a single TV transmitter. Images are captured off-board by a demodulator and frame grabber on a Lisp machine.

We have implemented precisely the three layers of task-achieving behaviors that are illustrated in figure 11.4; see Brooks (1986) for details. The robot has wandered around a laboratory and a machine room for many hours under control of these layers.

The very first experiments we did were with just the lowest-level layer operatonal. The very first experiments included dynamic obstacles (people), which caused the robot variously to flee and halt. After an initial shakedown period the robot has operated with demos every few days and only hit an obstacle once (a sponsor, as it happened, and we later tracked it down to a loose wire on one of the forward-looking sonars).

The higher levels were added and debugged later. The second layer lets the robot wander around with a drunk's walk. The third layer looks for distant points and then integrates the wheel shaft encoders in order to get

to the target in spite of perturbations caused when the lowest level avoids previously unseen obstacles. On occasion, problems with calibration of the shaft encoders have made the upper layer somewhat unreliable. But in all cases, the more primitive lowest-level layer, which does not rely on any but the coarsest of qualitative calibrations, is able to function flawlessly and in parallel. Thus the robot is at least kept out of danger, not colliding with any obstacles. We believe that these accidental experiments have demonstrated the robustness of our decomposition scheme.

Brooks and Connell (1986) report on an alternative set of higher-level layers built on top of exactly the same first level. The second layer let the robot follow walls, skimming past doorways. The third layer explicitly looked for doorways and directed the robot through them.

Recently we have began experiments (Brooks, 1987) with a fourth and a fifth layer built on top of our original three. The fourth layer takes over when it notices that Allen is in a corridor and follows along it. The fifth looks for safe cul-de-sacs at the ends of corridors and sets the robot in a position with its cameras pointing back along the open path.

11.3.2 Herbert the Robot

Our second robot (Brooks, Connell and Flynn, 1986; Brooks, Connell and Ning, 1987) is still under construction, although it does now move under its own computer control, and runs two layers of subsumption architecture. The lowest layer is again *avoid objects*. Above that is a *wall follow* layer.

To demonstrate graphically that there really is no hidden central control or source of synchronization in the subsumption architecture, we decided to build a distributed parallel processor with absolutely no central resource other than power. In particular this means no backplane, no bus among all processors, no switching network for messages, no global clock, and no shared memory.

Additionally, the range of tasks we can demonstrate with Allen is limited since its only form of actuation is to move. A manipulator arm on-board our second robot seemed appropriate.

The parallel processor has 24 independent processor boards. We plan on using them in two modes. At first we shall use one processor to simulate each finite state machine. Later, as we add control layers, we shall partition the wiring diagram and simulate up to six finite state machines with each processor.

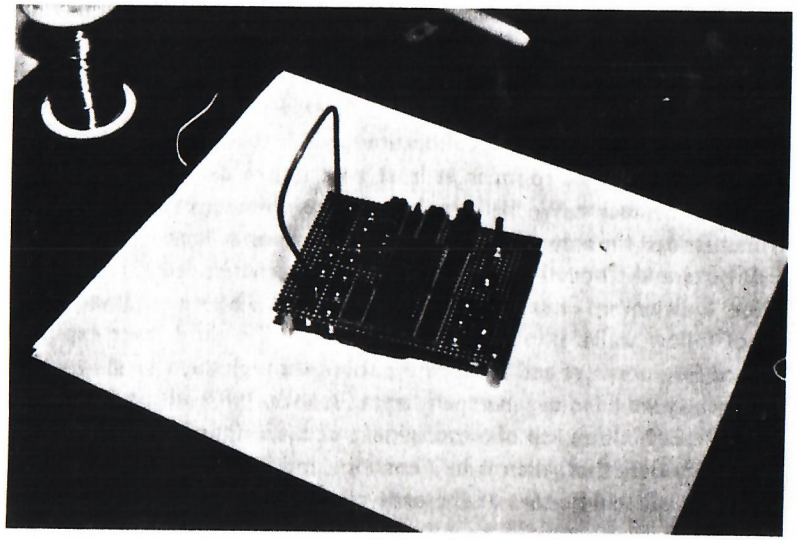

Figure 11.6
Each processor board has a single-chip microprocessor plus support chips for extra memory and an optional suppression node. The boards are coupled with absolutely no central resource other than power. There is no backplane, no bus between all processors, no switching network for messages, no global clock, and no shared memory.

The processors are arranged in three layers of 8 arranged in a circle around the body of the robot. Figure 11.6 shows one of the processor boards. It has a CMOS Hitachi 6301 (a version of the well known 6800 family) single-chip processor. The processor chip has 128 bytes of RAM onboard and accepts a piggyback 8K EPROM for programming. Additionally each processor has 2K bytes of off-chip memory. We arrange three serial inputs and three serial outputs for each processor by actively polling some of the parallel port lines available on the chip. The communications protocol over these asynchronous lines makes use of two conductors, one for control and one for data. In our initial protocol we use 24-bit messages. The serial lines are terminated on each board in minature telephone jacks. Wiring diagrams like those in figure 11.4 are implemented by patching the boards together with physical cables using this distributed patch panel.

Each processor board has a little extra room. Some use it to hold a parallel port so that processors can communicate with input/output devices. Others have a hardware suppression node, with two inputs and one output. It is completely independent of the processor with which it shares a board. The suppression node has its own inputs and outputs and can be wired into the network at any position. Still other boards include a serpentine memory for processing last light stripe images.

The manipulator is a lightweight two-degree-of-freedom chain-driven device. There is a parallel jaw gripper which always points directly downward for grasping objects. The gripper can be moved in a plane that is vertical and pointing in the direction the robot moves. There is a 40-inch high and 18-inch long rectangle in which the gripper can be moved. This means the robot can grasp objects at both table top and ground levels, and transfer objects weighing up to 2 pounds between those two levels.

The initial sensors on the robot are a set of up to 48 infrared proximity sensors for local obstacle and intruder detection. A laser light striper is under construction, to be mounted next to the manipulator.

11.3.3 Tom and Jerry the Robots

Our third and fourth robots, Tom and Jerry (Connell, 1987), are identical and are completely operational. Their absolute level of performance is not particularly advanced. They were built to demonstrate the idea of compiling our subsumption architecture down to a network of gates, without any conventional computing element.

Each physical robot is based on a commercial radio controlled toy. We removed the control electronics and replaced it with a self-contained larger board. We mounted four infrared proximity sensors, three forward looking and one rearward looking. The motors can be driven forward and back and the left and right front wheels have steering brakes. There are thus 4 bits of input to and 4 bits of output from the subsumption network.

The subsumption network itself is implemented in a single 256-gate PAL. There are three layers implemented. Some changes had to be made to accommodate 1-bit data paths. One of the many networks we have implemented is shown in figure 11.7. The lowest level is our familar *avoid objects* layer of control, which includes avoidance and halting behaviors. The next layer of control lets the robot explore large areas. The third level gives some extra heuristics for backing out of tight situations.

11.3.4 Seymour the Robot

We have started work on constructing Seymour, and all passive-sensor robots. Our concentration on this robot is to develop reliable real-time

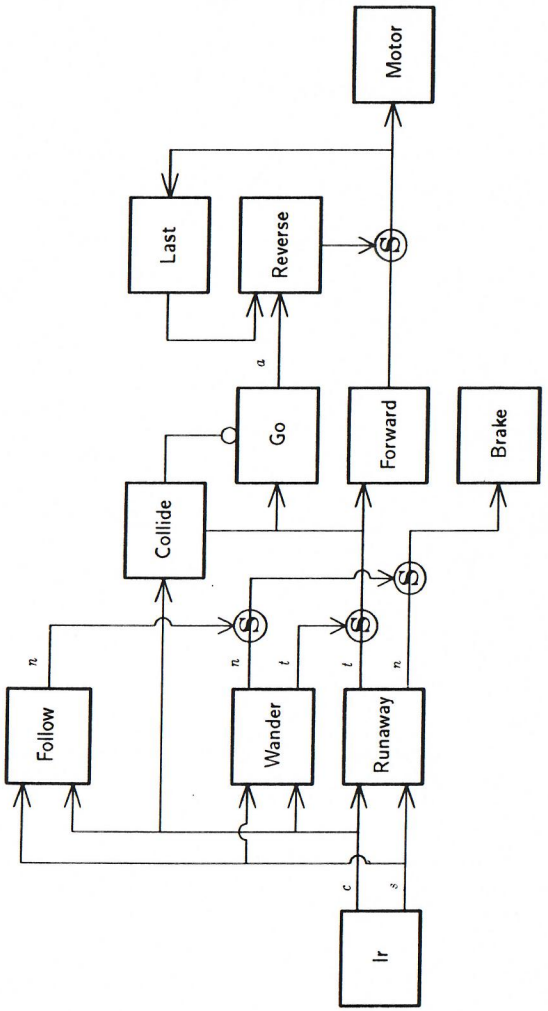

Figure 11.7
The layers of control for Tom and Jerry are all implemented with 1-bit data paths.

vision that is robust over a wide range of circumstances. Brooks, Flynn and Marill (1987) report on some vision algorithms we have developed for this robot.

All vision is to be done on-board. The algorithms are completely task dependent. So far we have concentrated on vision for obstacle avoidance. We feel we can achieve this goal with rather low-resolution cameras. We have constructed some prototypes. Our plan is to mount 8 to 10 cameras on the robot, each a low resolution CCD camera (some 32 × 32 pixels, and some 1 × 1,024). We are constructing a special purpose processor to handle the early processing of the visual data before it gets injected into the subsumption network. There is a Hitachi CMOS 68010 as host processor driving a CMOS Analog Devices DSP2100 slave. The later can do linear convolutions at 8 million pixels per second. The complete processor system draws only 5 watts.

11.4 The AI Problem

This and the next two sections present the reasons we have chosen to build our mobile robots in such a nontraditional way.

Autonomous mobile robots must be intelligent to some degree. Naturally, therefore, they must be *artificially intelligent*. Much of the work in mobile robots during the sixties and seventies was in fact done in Artificial Intelligence Laboratories (e.g., Nilsson, 1984; Moravec, 1983; Giralt et al., 1984). The first of these (Nilsson's robot Shakey) was a source of much inspiration for work in Artificial Intelligence, while the latter two were the first to apply successfully many AI techniques to real world situations.

The Shakey project is often quoted as a smashing success of applied AI. In contrast I feel its methodology was misguided, and caused a misdirection of the bulk of AI research during the seventies and eighties. AI is only just now beginning to recover from this disaster.

Throughout the sixties there had been an implicit assumption that *intelligence* could be separated from *perception* and *action*. See, for example, the collections of papers in Feigenbaum (1963) and Minsky (1968). All of these papers describe disembodied reasoning systems. They are given inputs in terms of a priori databases and strings of characters received from a teletype. They provide outputs in the form of characters sent to a teletype.

Shakey followed this tradition and seemed to validate it in that it demonstrated that a robot could actually exist and operate in the world with very separate perception, action, and intelligence subsystems.

But Shakey cheated.

Shakey's world was a very carefully engineered toy. Walls were specially constructed and painted with matte finishes. Besides walls and Shakey there was a set of large polyhedral blocks, each of whose faces was painted in a solid matte color.

One reason for such a simplified environment was the low perceptual rate achievable with computers (even mainframes) of the day. There were simply no known algorithms that could process visual information reliably in any but highly constrained circumstances. Such algorithms (still in their infancy even today) required the advent of both new theories and faster computers so that they could be tested out in reasonable time frames.

Another reason for the simplified environment was that it simplified the reduction of the total problem into relatively independent pieces with clean interfaces. A simple world meant that the interface language could be simple and descriptions of the world could be short enough to fit into the limited amount of memory available in mainframes of the time. In fact, the interface was not completely straightforward, as a little bit of the real world did manage to creep in, and collusion between intelligence and perception was necessary. As Shakey wandered around, its position estimate from its shaft encoders drifted. It was necessary to refer to its world model for a nearby vertex and then match that with its perceptions to triangulate and correct its position estimate.

Shakey's major contribution to AI thought at the time was the idea of having a complete model of the world. It seemed to show that with an *adequate* model it was sufficient to reason within the model and forget about the world itself. The model, after all, encoded everything of relevance that occurred in the world. I have not seen any papers of the time claiming that these were contributions of Shakey. I do not claim that these were recognized as results of the Shakey experience per se. Rather they were operational lessons that were absorbed into the culture of AI research.

There are two serious drawbacks with Shakey's approach to intelligent systems existing in the world:

1. There is an implicit assumption that the black box of perception will be able to deliver descriptions of the world that are such exact matches to the world that it is possible to reason with the world model and not be missing any critical facts. Shakey achieved this goal only partially in that it did require an a priori description of the world save the exact positions of the large painted blocks. In any case, even for that simple degree of world model updating, Shaky relied on the class of unknowns being very simple and completely understood a priori.
2. There is an implicit closed world assumption. Shakey knew everything about the world and everything that could influence it. The only exceptions were that a single large block could be magically and grossly displaced from its "known" position. When such a discrepancy was detected Shakey could take its time to update its world model. Things never changed enough that its position estimate update routines would be overloaded. In a real, dynamic, world a robot can not know everything about the world. It must rely on more high-frequency sensing to detect discrepancies and update its world model. When many things can change simultaneously, the simple approach taken by Shakey simply will not work. The changes in the world interact with the robot's use of its world model that is necessary to perceive accurately what needs to be updated.

Unfortunately most of the work in robot planning, for assembly especially, has adopted this idea of having a complete and closed world model. Winograd (1972) constructed a natural language interface (SHRDLU) to a program that simulated a perfect world of colored blocks and pyramids (like Shakey's world, only smaller) where a perfectly behaved robot stacked and unstacked them. Fahlman (1975) brought planning to a more realistic stage in his BUILD program, which generated plans for constructing assemblages in the blocks world by having a robot manipulator grasp blocks, stack them, build scaffolding, and produce complex creations. Again the world was simulated so the true problems of a real world did not creep in. Lozano-Pérez (1984) in a planning system (ATLAS) for assembly tried to make the world more realistic by dealing with bounded position and force uncertainties as would arise from parts tolerances and robot control error but still heavily relied on a closed world assumption. Their proposal remains untested in the real world.

These three examples illustrate how the AI side of robotics has relied on planning off-line in a closed world. Unfortunately the real world is not closed. Much of the planning work done in Artificial Intelligence is irrelevant to mobile robots. Those who argue that the planning work

available outstrips in capability the perception work that is available are being confused by a mismatch of expectations from the planning side versus the reality of what perception will ever be able to deliver.

11.5 The Role of Sensing and Perception

Sensors provide raw data bits for perceptual processes. But what should perceptual processes deliver as output? What is the purpose of perception?

Most work in computer perception has been based on visual sensing, although a little has been done on sound (but almost all in the realm of speech understanding), touch, and active sensing (e.g., radar). I believe that the majority of this work has suffered from two drawbacks that make many of its fruits irrelevant to mobile robots. First, there has been a concentration on object shape recovery for recognition, and second, there has been a concentration on extremely accurate shape recovery.

Marr (1982) pointed out that the *purpose of vision* depends on the task the perceiving organism is trying to achieve. His example in this section (p. 32) of his philosophical treatise is the housefly. He then talks about *advanced vision*, which he equates with human vision and uses as his examplar the task of object recognition **with the generality that humans can bring to bear on this task**. The rest of the book develops techniques or ideas for techniques for the transformation of representations from image to primal sketch to $2\frac{1}{2}$-D sketch to 3-D model representation in support of this task.

For object recognition with human level generality as determined by our own introspection, it seems that we need to recover the shape of objects. Indeed this has been a central focus of computer vision research. For instance in Brady (1981) a collection of 14 high-quality AI-flavored papers on vision, fully 8 are explicity concerned with shape recovery, and 3 more are explicity concerned with object recognition but bypass the shape recovery phase. Other monographs (e.g., Lowe, 1985) and textbooks (e.g., Ballard and Brown, 1982) explicitly state that object recognition is the major goal of computer vision and furthermore that explicit high-level world knowledge is an important ingredient in that recognition process.

I believe that such concentration on surface shape recovery and object recognition has been spurred by the belief that complete world models are needed by planning processes. But more insidiously there has also been a tacit belief that the descriptions of objects delivered by vision must be

extremely accurate. This may well be due to (1) the traditional use of position control in robotics, because of its computational simplicity and low sensory bandwidth (shaft encoders), and (2) the many simplistic noise-sensitive shape-matching attempts at computer vision. Thus we have seen over the last few years intense amounts of effort going into calibrating (e.g., Moravec, 1983; Faugeras, 1986) visual systems to a 3-D coordinate system.

Suppose now that we back off from this implicit model that computer vision should be like what we imagine human vision is, and reconsider Marr's (1982) point that the purpose of vision depends on the task to be achieved. Then we must examine what it is we want our robots to do, what information they will need to do it, and build vision algorithms that deliver precisely the necessary information. Trying to deliver more general information is a prescription for failure—it will be far too easy to slip into the traps we have seen computer vision fall into already (delivering accurate surface descriptions for their own sake, and worrying about second-order effects and accuracies that are irrelevant in the real world).

In particular, a robot that must avoid obstacles does not need to recognize the obstacles (as chairs, trees, etc.). At most it might need to generate some volumetric description of the obstacles, in order to avoid intersecting those volumes. Except in very cluttered situations the volumes can be generous bounding volumes. This certainly makes the vision problem simpler. But, in fact, there is another approach that simplifies it even more. Instead of using vision to concentrate on where obstacles are, perhaps a better strategy for this particular task is to concentrate on where free, navigable space is, and have the vision systems deliver that as their model of the world. The robot's task determines the most appropriate emphasis of the early vision algorithms being used in support of that task.

11.6 The Role of the World

Since globally consistent world models are hard to build and maintain, and are perhaps not useful, consider the possibility of not building them at all. Instead let us consider the possibility of using the world as its own model. It has the advantage of being a complete model of itself, of being totally accurate, and of always being up to date. The implementation impact of this idea is that rather than try to predict exactly what will happen in a model, we notice aspects of the world from our sensors, and act on those

aspects. We rely on the world to integrate all our independently controlled actions to give a cohesiveness to the overall behavior.

We give three examples; the first is another independent piece of work, the second is operational on our robots today, and the third is a scenario we are working toward for our robots.

11.6.1 Hopping Machines Rely on the Physics of the World

Raibert (1983) used this approach in essence in the control systems for his hopping machines. In the case of the one-legged hopper he independently controls three aspects of the robot in the world by controlling two actuators (*1. fluid flow in the leg* and *2. torquing of leg-body joint*) on the basis of five sensors: *a. foot-touching-ground sensor, b. leg extension, c. pressure in leg, d. leg to body angle*, and *e. global body attitude*. The controlled aspects of the robot in the world are

- *hopping height:* sensors *b* and *c* are integrated to determine control of actuator *1*;
- *body attitude:* during stance as determined by sensor *a*, sensors *d* and *e* are integrated to control actuator *2*; and
- *forward running speed:* during flight as determined by sensor *a*, the previous readings during stance of sensors *b*, *d*, and *e* are used to control actuator *2*.

The crucial ideas underlying this approach are

- There is no global model maintained of what the robot is doing. Three aspects of its world behavior are controlled independently and the result is that the world integrates these through physics to produce a running machine.
- Sensors are integrated in groups to provide partial models of what the robot is doing. Some sensors are used more than once to provide independent partial descriptions.

11.6.2 Finding Paths among Dynamic Obstacles

Consider the layers of control described in section 11.2. The robot does not attempt to segment the world into dynamic and static obstacles. Rather it takes instantaneous views of the world and reacts to those. If the world changes, the robot naturally changes its behavior—fleeing from an agressor, for instance. It is its interaction with the world that makes it avoid obstacles, not any internal representation of and reasoning about obstacles.

At the same time, and in parallel, the robot is trying to explore its environment. The exploration layer wants the robot to go in a particular direction. It tells the lower levels so, but then observes the behavior of the robot in the world to see what really happened. The higher-level layer would have to model the lower levels if it was to maintain a halfway accurate model without simply resensing what happens in the world.

11.6.3 Grabbing a Soda Can

Figure 11.8 shows a collection of behaviors that will enable Herbert to wander around office areas collecting empty soda cans from desks and depositing them in some home location. We do not attempt to build a detailed model of the world. In fact we do not even have much interbehavior communication on board the robot. Rather we let the world couple the behaviors.

For instance, one behavior looks for candidate soda cans on tabletops. When it sees one it tells another behavior the location, and that behavior,

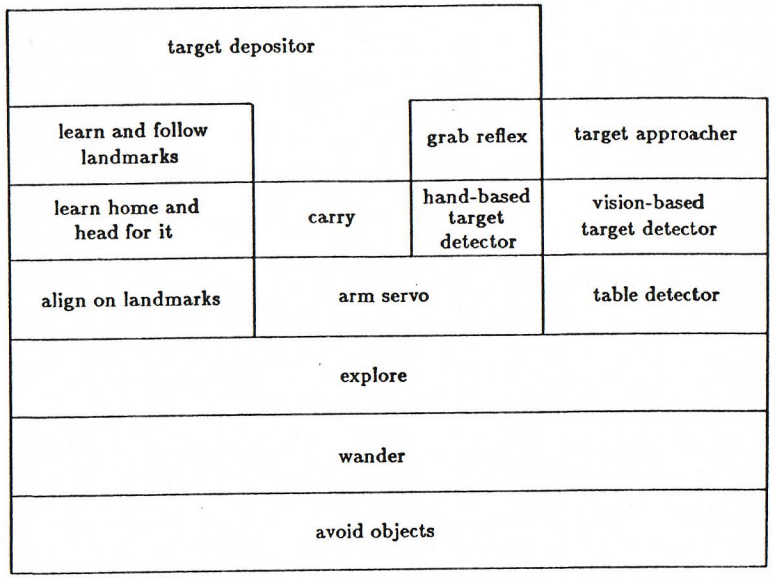

Figure 11.8
We let 14 independent behaviors run in parallel on Herbert. By indexing themselves on sensor values measured in the world, they combine to give the robot a cohesive externally observable behavior—collecting soda cans from offices.

if it is active and has control of the robot, moves the robot so that the gripper is above the target area. Notice that if the robot is busy fleeing or something else important is happening, the robot will not go toward the can. That is no concern of the original soda can detector. It was simply looking for cans and noticing them.

Another behavior is always monitoring sensors on the hand, and whenever it thinks the hand is above a soda-can-like object, it lets the grasp behavior know this fact. If other conditions are met for grasping, it will happen. If the object was incorrectly identified from afar, it will not be grasped. If in the meantime some other high-priority event happens, the soda can will be quietly forgotten. There is no need to program in elaborate abort strategies as each and every behavior is built to be active only when the world is in an appropriate state.

11.7 Dogma

There are a number of aspects of our work that are commonly questioned. In this section we outline these objections and (of course) refute them. Individually each of the questions or objections has merit. We claim, however, that the total picture makes each of our idiosynchratic foibles justifiable.

1. *Why not build regular world models?*
2. *Why not carefully calibrate sensors and actuators before running experiments?*
3. *Why restrict the model to simple networks of such simple machines?*
4. *Why restrict to low-bandwidth communications between processors?*
5. *Why not simulate all the finite state machines on a single large processor?*

The following subsections answer each of these challenges.

1. *No representation* Other projects can explore these topics. We have identified a number of methodological pitfalls in following this practice and have chosen as an object of study to see how far we can push the notion of not having explicit internal models. We want to see what levels of functionality we can reach without them.

2. *No calibration* The desire for calibration comes from the notion that there is an objective reality in the world and *the robot must know this objective reality*. Ignoring the first clause of the conjuction, we take issue with the second. We suspect that once this idea takes root, researchers search for accurate models of the world for accuracy's sake, not for the sake of the task the robot is to perform. Often they will argue that indeed the accuracy is needed because another algorithm within their robot relies on that accuracy. But we argue that it is that second algorithm that is incorrectly designed.

As an example, consider the problem of visually detecting obstacles in front of the moving robot in order to plan local collision-free paths around them. It is clearly not necessary to extract accurate surface models of the obstacles. One might argue that it is necessary to estimate their depth accurately so that the necessary path can be planned to avoid them. But notice that in carrying out the avoidance maneuver the robot must respond accurately to motion commands. In fact the accurate coordinate system used for recording obstacle position forces accurate electromechanical control of the robot. This internal accuracy merely serves to connect accurate vision and accurate motion, whose accuracy was only demanded by the existence of the internal requirement.

An alternative strategy is to connect vision to motion more directly and let them *calibrate each other*. For instance, if we can determine motion flow (often approximated by optical flow) in a series of forward-looking camera images, and if the robot is traveling with pure translation, then we can estimate the time to collision for a point ahead. It is simply the ratio of the distance of the point in the image from the center of expansion to the velocity of the point away from that center. If we plan to avoid obstacles in time-to-collision space, then we can easily control the robot. Its current velocity determines the scale of the planning space, not through any calibration, but through the physics of imaging in the world. We have removed the internal accurate model by choosing an appropriate coordinate system for a single task that involves both vision and motion. Brooks, Flynn, and Marill (1987) describe experiments using Allen the robot with such a self-calibrating vision algorithm, and in fact use it to calibrate another algorithm, stereo vision, for misalignment of the cameras.

3. *No complex computers* Originally it was not an explicit goal of this project to use simple computation elements. The original conception was that there would a network of 68000s linked by low-bandwidth communication channels. As we worked on the natural decomposition of the system, however, we found repeatedly that very little was demanded of each

parallel process, and that they could all be easily expressed as finite state machines running asynchronously.

After making this leap it was noticed that a new benefit had accrued. Now each of the processors is so simple that it could actually be implemented on a very small piece of silicon. Perhaps it will be possible to fabricate complete mobile robot control systems on a single chip. To explore this, we have started work on a silicon compiler to transform a subsumption network into a single VLSI chip layout.

4. *No high-bandwidth communication* By restricting ourselves to low-bandwidth communication we can ensure that all the data paths in a VLSI implementation of the subsumption architecture can be quite narrow, and thus a large number of them can be implementable. Furthermore, in a more macroscopic scale implementation we eliminate the temptation to link processes with centralized, and ultimately capacity limited (see next subsection), resources, such as a switching communication network.

5. *No nonincremental resources* On robot Allen we simulated the finite state machines on an off-board Lisp machine. Eventually, as we added finite state machines, we overloaded the capacity of that single machine to respond in real time while carrying that simulation load. But well before we reached that saturation point we found we had to spend considerable effort prioritizing the running of the supposedly parallel finite state machines so that nothing critical was starved out.

Once saturation was reached the question of which single processor we should use to simulate all these finite state machines arose. Clearly no matter which machine we chose we would eventually overflow its capacity. Furthermore, we would again have to spend time prioritizing supposedly parallel processes to run on a single processor. Eventually the new processor would become saturated too.

To simplify experiments we therefore decided actually to build an indefinitely extensible parallel processor. The extensibility is ensured by the only global resource being electrical power. Besides being extensible it graphically makes the point of there being no locus of central control in the subsumption architecture.

11.8 Whole Iguanas

Whole iguanas are complex creatures. So are whole houseflies. Even a whole snail is beyond what we can achieve today. It is a completely

autonomous system that operates for a long period of time in a dynamic environment, achieving certain built-in goals necessary for its survival.

In this chapter we have argued that for a robot to achieve such levels of behavior as routinely achieved by billions of spieces of animals we need to control the vast complexity of many interacting parts of intelligence (where we extend the definition of intelligence to include the competence demonstrated by a snail, for instance).

The techniques we see as appropriate to controlling this complexity are

- Build systems incrementally.
- Build loosely coupled systems.
- Instead of trying to represent the world explicitly, find mappings from aspects of the world to appropriate actions.

Acknowledgments

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the research is provided in part by an IBM Faculty Development Award, in part by a grant from the Systems Development Foundation, and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

References

Ballard and Brown (1982) Dana H. Ballard and Christopher M. Brown, *Computer Vision*, Prentice-Hall, Edgewood Cliffs, NJ, 1982.

Brady (1981) J. Michael Brady (ed.), *Computer Vision*, special issue *Artificial Intelligence* 17, 1981.

Brooks (1986) Rodney A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, RA-2, No. 1, 1986, 14–23.

Brooks, Connell, and Flynn (1986) Rodney A. Brooks, Jonathon H. Connell, and Anita M. Flynn, "A Mobile Robot with Onboard Parallel Processor and Large Workspace Arm," *Proceedings AAAI Conference, Philadelphia, PA*, 1986, 1096–1100.

Brooks and Connell (1986) Rodney A. Brooks and Jonathon H. Connell, "Asynchronous Distributed Control System for A Mobile Robot," *Proceedings SPIE, Cambridge, MA*, 1986.

Brooks (1987) Rodney A. Brooks, in preparation.

Brooks, Connell and Ning (1987) Rodney A. Brooks, Jonathon H. Connell, and Peter Ning, in preparation.

Brooks, Flynn, and Marill (1987) Rodney A. Brooks, Anita M. Flynn, and Thomas Marill, "Self Calibration of Motion and Stereo Vision for Mobile Robot Navigation," MIT AI Lab Memo 984, August 1987.

Dennett (1978) Daniel C. Dennett, "Why Not the Whole Iguana?" *The Behavioral and Brain Sciences*, 1978, 103–104.

Fahlman (1975) Scott E. Fahlman, "A Planning System for Robot Construction Tasks," *Artificial Intelligence* 6, 1975.

Faugeras (1986) O. D. Faugeras and G. Toscani, "The Calibration Problem for Stereo," *Proceedings IEEE Computer Vision and Pattern Recognition, Miami, FL*, 1986, 15–20.

Feigenbaum (1963) Edward A. Feigenbaum and Julian Feldman (eds.), *Computers and Thought*, McGraw-Hill, San Francisco, 1963.

Giralt (1984) Georges Giralt, Raja Chatila, and Marc Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots," *Proceedings of the First International Symposium on Robotics Research, Bretton Woods, NH*, edited by Michael Brady and Richard Paul, MIT Press, Cambridge, MA, 1984, 191–214.

Lowe (1985) David G. Lowe, "Perceptual Organization and Visual Recognition," Kluwer Academic Publishers, Boston, 1985.

Lozano-Pérez (1984) Tomás Lozano-Pérez and Rodney A. Brooks, "An Approach to Automatic Robot Programming," *Solid Modeling by Computers*, edited by Mary S. Pickett and John W. Boyse, Plenum Press, New York, 1984, 293–327.

Marr (1982) David Marr, *Vision*, W. H. Freeman, San Francisco, 1982.

Minsky (1968) Marvin L. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, 1968.

Moravec (1983) Hans P. Moravec, "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE*, 71, July 1983, 872–884.

Moravec (1984) Hans P. Moravec, "Locomotion, Vision, and Intelligence," *Proceedings of the First International Symposium on Robotics Research, Bretton Woods, NH*, edited by Michael Brady and Richard Paul, MIT Press, Cambridge, MA, 1984, 215–224.

Nilsson (1984) Nils Nilsson, "Shakey the Robot," SRI AI Center Technical Note 323, April 1984.

Pylyshyn (1978) Zenon W. Pylyshyn, "Computational Models and Empirical Constraints," *The Behavioral and Brain Sciences*, 1978, 93–99.

Raibert (1984) Marc H. Raibert, H. Benjamin Brown, Jr., and Seshashayee S. Murthy, "3-D Balance Using 2-D Algorithms?" *Proceedings of the First International Symposium on Robotics Research, Bretton Woods, NH*, edited by Michael Brady and Richard Paul, MIT Press, Cambridge, MA, 1984, 279–301.

Winograd (1972) Terry Winograd, *Understanding Natural Language*, Academic Press, New York, 1972.

IV DESIGN AND ACTUATION