Chapter 7

Operating System

Any system that must operate in a complex and changing environment must be compositional, that is it has to have elemental pieces that can be composed to create its more complex structures.[1] Figure 7.1 illustrates two broad compositional approaches that have been pursued in theories of cognition, as well as in robotics. The first decomposition works on the assumption that the agent has a central repository of symbolic knowledge. The purpose of perception is to translate sensory information into symbolic form. Actions are selected that result in symbolic transformations that bring the agent closer to goal states. This sense-plan-act approach is typified in the robotics community by early work on Shakey the robot,[2] and in the cognitive science community by the theories of David Marr.[3] In principle, the symbolic planning approach is very attractive, since it suggests that sensation, cognition and action can be studied independently, but in practice each step of the process turns out to be very difficult to characterize in isolation.

The difficulties with the compartmentalized symbolic planning approach have led to alternate proposals. In the robotics community Brooks[4] has attempted to overcome these difficulties by suggesting a radically different decomposition, illustrated in Figure 7.1B. Brooks' alternate approach is to attempt to describe whole visuo-motor behaviors that have very specific goals. Behavior-based control involves a different approach to composition than planning-based architectures: simple microbehaviors are sequenced and combined to solve arbitrarily complex problems. The best approach to attaining this sort of behavioral composition is an active area of research. Brooks' own *subsumption* architecture worked by organizing behaviors into fixed hierarchies, where higher level behaviors influenced lower level behaviors by

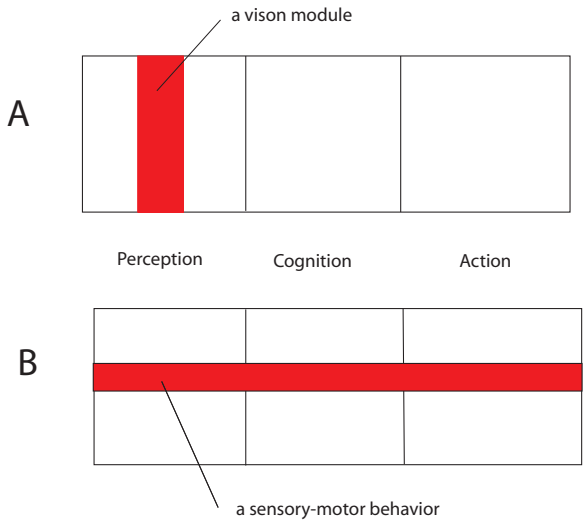


Figure 7.1: Two approaches to behavioral research contrasted. A) In the Marr paradigm individual components of vision are understood as units. B) In the Brooks paradigm the primitive unit is an entire behavior.

over-writing their inputs. Subsumption works spectacularly well for trophic, low-level tasks, but generally fails to scale to handle more complex problems.[5]

For this reason, two diverse communities - robotics and psychology - have been working on Brooks-like cognitive architectures for managing complex tasks that take a more integrated approach to vision and action, but both groups have moved toward model architectures that have a more hierarchical structure e. g. [Newell, 1990; Brooks, 1986; Firby et al., 1995; Bryson and Stein, 2001; Arkin, 1998]. In particular, robotics researchers have gravitated to a three-tiered structure that uses strategic, tactical, and detail levels to model complex behavior [Bonasso et al., 1997]. These versions of Brooks' behavior-based control centers around programs termed *microbehaviors*. A microbehavior is a complete sensory/motor routine that incorporates mechanisms for measuring the environment and acting on it to achieve specific goals. For example a collision avoidance microbehavior would have the goal of steering the agent to avoid collisions with objects in the environment. A microbehavior has the property that it cannot be usefully split into smaller subunits. Walter's microbehavior control architecture follows more recent work on behavior based control (e.g.[6,7]) that allows the agent to address

changing goals and environmental conditions by dynamically activating a small set of appropriate behaviors. Each microbehavior is triggered by a template that has a pattern of internal and environmental conditions. The pattern-directed activation of microbehaviors provides a flexibility not found in the fixed subsumption architecture.

At this point we have laid the groundwork to discuss the running of programs. You might think that this was done in Chapter five. That venue laid out the basic cycle of navigating between states and actions, but many additional and vital issues arise in using programs to direct the body in the course of behaviors. Taken together they comprise the kinds of tasks that a silicon computer's operating system does. At this point its obvious that the brain computes in a way that is very different form silicon computers, but particularly for computational abstractions at this level, the problems faced are similar. Let us introduce them first before characterizing them in detail.

Working memory In the 1950s, Miller famously describe the human ability to memorize '7 ± 2' items. This figure has since shrunk to 4, but the fact remains that human short term memory for new items is limited. Allen Baddley has made extensive study of this limit and coined the term *working memory*. He has also shown that visual working memory is separate from audio working memory under certain conditions. But our focus is on its purpose for behaviors. In particular its role in computation. It turns out to have a rather obvious one. The brain has a huge inventory of programs for doing things, but they can be much more useful if they can be indexed by variables. You may have a way of spreading jam on bread. It turns out this skill would work for mayonnaise as well as mustard. So naturally the best thing to do would be to structure the program in a way that it could have a variable 'the-thing-that-is to-be-spread.' Another example would be everyday conversation. You can describe an everyday situation, but would like it organized so that, if it is about a particular person, his or her name could simply be inserted in your more or less stock description. This is a putative role for working memory: to extend the value of standard programs by giving them variable arguments. Working memory is the place-holder for particular values of these arguments.

Program execution Another important issue is that of *program execution*. In the earlier description of programs as cortex-basal ganglia transitions, the

issue of whether or not actions were successful or not, was glossed over. However in the real world with fallible human bodies things can easily go wrong and there have to be ways of recognizing that things are going awry and compensating accordingly. Coming to grips with this requires extending the state-action paradigm to incorporate measurements taken by routines.

Multi-tasking Given that programs are being executed, how many of them can be simultaneously active? this is a complicated question as its answer impacts many levels of abstraction. At the very lowest level, a worry is that, since the neural architecture is shared, there might be cross-talk wherein the neural code for one program interacts destructively with that of another. At a higher level, there is the issue of variable binding. Given that working memory is needed for this job, is it a shared resource when multiple programs have variable binding needs? We will review some psychological data that speaks to this issue.

Indexing Given that a program's execution can be managed, how is it picked in the first place? You could drop what ever you are doing right now and brush your teeth, but for all but a small set of venues, it would be an inappropriate behavior. Given the vast set of possible behaviors that you can do, how does the brain pick an appropriate set for the situation of the moment? I term this the *indexing problem*. How to solve it is a huge open question upon which very little work has been done. However it is solve though, there is an important triage that can be done, and this is necessary to characterize the program library. Programs can be in three different activation states (not to be confused with the cortex's 'states' !): *asleep*, *active*, and *engaged*. A program is asleep is it has no trace in terms of neural spikes. It potentially could be activated given an internal bodily need or some external stimulus, but in the asleep state, it is extant only by virtue of its synaptic connections. A program is active if there are at least cortical spikes denoting its current state. It may be keeping track of the transit of the body or it may be used in some simulation of an abstract thought process, but the key element for the active state is that it is not interfacing with the body's large scale sensorimotor resources.

7.1 Working memory

The neural network coding of the driving example illustrates both sides of a cognitive coin. On the one hand a vast array of successful driving experiences means that you have seen a lot of data to program your table. Take a moment to estimate the number of visual fixations you have made while driving using the standard rate of three per second multiplied by the total time spent on the road. You'll be impressed with yourself! On the other hand, even given all this experience there are still situations that arise that are not covered in the table. What happens then?

The penultimate sentence is not completely true in the sense that although its overwhelmingly unlikely that there will be a table entry for a similar situation, the rub is that there will need to be some work done to represent this similarity, because the table entries are a compressed code that circumscribe the situations seen so far. To make this concrete and sticking with driving, imagine as a westerner driving for the first time in India and encountering a cow at a busy intersection in an urban setting. You know how to handle pedestrians and perhaps animals in pastoral settings but the particular cow-urban combination has never come up. The cow will have its albeit primitive strategy for avoiding you but what is it? There all all kinds of table entries for situations similar to this for various slow moving objects, so its just a matter of somehow getting the cow features interpreted as a table entry for the right table.

The cow coding problem is just another example of the binding problem. We need some special apparatus to interpret data that is to play a role in a table that we already have. This job is done by *working memory*, a somewhat loosely defined concept to address this situation. Working memory was unearthed by Miller who noticed that, for a variety of different circumstances where we had to remember things, the number of novel things that can be remembered is rather small. Miller bracketed the number in his landmark paper as "7 plus or minus 2," but subsequent experiments have revised this downwards to about four. Just what constitutes an item of working memory? For unrelated digits, the item is likely to be a single digit as in 7651993826547915. But how about the following digits?

1492177619391963

They are easily remembered in the US as the year of Columbus's landing, the year of the American independence declaration, the start of world war II, and the year of Kennedy's assassination. What has been done, in terms of

psychological jargon, is that the items have been "chuncked," or associated with existing tokens. Although the psychologists, particularly Baddley in this case, have been extraordinarily good at ferreting out the measurable properties of working memory, its the case that a computational account provides a crisper description of the underlying technical issues and how they might be addressed. So at this point we'll switch to that venue.

Working memory can be seen computationally as the state needed to direct behavioral programs. To harken back to the driving venue, when driving down a busy street with directions to turn right at the third light, its the number of lights seen so far that is the important state. All the other features of the dynamic scene are - for this task - unimportant. Furthermore you will have noticed that the amount of storage we need to describe the essential state is very modest. At any point in a lengthy complicated program, you just need to have enough state to keep track of where you are in the program.

Another issue related to state is the notorious "chunking," a term used somewhat out of desperation, I think. What is going on can be informed by the computer science concept of a pointer, wherein a small amount of information can herald or "point to" a more elaborate description.

Although the human brain is radically different in many ways from conventional silicon computers, they both have to address many of the same problems. Thus it is sometimes useful to look at how problems are handled by silicon computers. One major problem is that of variable binding. As recognized by Pylyshyn [1989] in his FINST studies, for symbolic computation it is often necessary to have a symbol denote a very large number of bits, and then modify this reference during the course of a computation. Let us examine how this is done using an artificial example.

Table 7.1 shows a hypothetical portion of memory for a computer video game where a penguin has to battle bees. The most important bee is the closest, so that bee is denoted, or pointed to, with a special symbol "the-bee-chasing-me." The properties of the lead bee are associated with the pointer. That is, conjoined with the symbol name is an address in the next word of memory that locates the properties of the lead bee. In the table this refers to the contents of location 0001, which is itself an address, pointing to the location of beeA's properties, the three contiguous entries starting at location 0011. Now suppose that beeB takes the lead. The use of pointers vastly simplifies the necessary bookkeeping in this case. To change the referent's properties, the contents of location 0001 are changed to 1000 instead of 0011. Changing just one memory location's contents accomplishes the change of

0000	the-bee-chasing-me	0000	the-bee-chasing-me
0001	**0011**	0001	**1000**
0010		0010	
0011	beeA's weight	0011	beeA's weight
0100	beeA's speed	0100	beeA's speed
0101	beeA's No. of stripes	0101	beeA's of stripes
0110		0110	
0111		0111	
1000	beeB's weight	1000	beeB's weight
1001	beeB's speed	1001	beeB's speed
1010	beeA's No. of stripes	1010	beeA's of stripes
1011		1011	

Table 7.1: A portion of computer memory illustrating the use of pointers. *Left:*Reference is to beeA. *Right:* Reference is to beeB. The change in reference can be accomplished by changing a single memory cell.

reference. Consider the alternative, which is to have all of the properties of the "the-bee-chasing-me" in immediately contiguous addresses. In that case, to switch to beeB, all of the latter's properties have to be copied into the locations currently occupied by beeA. Using pointers avoids the copying problem.

It should be apparent now how deictic reference, as exemplified by eye fixations, can act as a pointer system. Here the external world is analogous to computer memory. When fixating a location, the neurons that are linked to the fovea refer to information computed from that location. Changing gaze is analogous to changing the memory reference in a silicon computer. Physical pointing with fixation is a technique that works as long as the embodying physical system, the gaze control system, is maintaining fixation. In a similar way the attentional system can be thought of as a neural way of pointing. The center of gaze does not have to be moved, but the idea is the same: to create a momentary reference to a point in space, so that the properties of the referent can be used as a unit in computation. The properties of the pointer referent may not be, and almost always are not, all those available from the sensors. The reason is that the decision-making process is greatly simplified by limiting the basis of the decision to essential features of the current task.

Both the gaze control system and neural attentional mechanisms each

dedicate themselves to processing a single token. If behaviors require additional variables, these must be kept in a separate system termed working memory.[?,?,?,?] Although the brain works on very different principles than a computer, the problem faced is the same. In working memory the references to the items therein have to be changed with the requirements of the ongoing computation. The strategy of copying that was used as a straw man in the silicon example is even more implausible here, as most neurons in the cortex exhibit a form of place coding[?,?,?] that cannot be easily changed. Thus it seems that at the one-third second time scale, ways of temporarily binding huge numbers of neurons and changing those bindings must exist. That is, the brain must have some kind of pointer mechanism.

With the concepts of state and pointers we can summarize the issue related to working memory. The brain needs to keep track of where it is in a program, so the neurons must represent the essential state. This state has the special characteristic that it is using the state elements in a way that is not hard coded in the cortical table. What is desired is that items that are hard coded be used in a role that is also hard coded. This new relationship is captured by a temporary cabal of neurons that are specially suited for the job. The neurons themselves do not need to represent all the information, but just the parts that are not already hard coded. In that sense they can be thought of as pointers to the hard coded information.

7.2 Program execution

Given that the appropriate variables have been bound to the program the next step is to execute it. At an abstract level the program is going to be represented as a state-action table that can be sequenced through. To make this description more concrete let us consider the down to earth example of making a peanut butter sandwich. One level of description would have the major steps denoted as in Table 7.2. You might quibble with this level, as being too abstract as each of the steps could be broken down into finer grained steps. For example, spreading the peanut butter onto the bread may involve more than one trip to the jar depending on the size of the bread and the size of the glob of peanut butter on each transit of the knife. The exact way to handle such hierarchies is not completely settled, and of course the steps both visual and motor routines, but whatever the final result, it should be cast in terms of state-action table lookups.

Task Code	Description
BT	Put the bread onto the plate
PLF	Take the lid off the peanut butter jar
JLF	Take the lid off the jelly jar
KH	Pick up the knife
POB	Spread the peanut butter onto the bread
JOB	Spread the jelly onto the bread
PLO	Put the peanut butter lid on
JLO	Put the jelly lid on
KT	Put the knife on the table
FB	Flip the bread to complete the sandwich

Table 7.2: The organization of human visual computation from the perspective of the microbehavior model.

Table 7.3 summarizes the scheduling of 10 subtasks in making a peanut butter sandwich by 3 human subjects. We make some coding assumptions such as the knife is picked up only once and is not put down until spreading finishes. Despite that some chronological constraints, e.g. BT, PLF and KH must precede POB and JOB, have ruled out most of the 10! orders, the number of possible orders remaining is still a large number If we divide the 10 subtasks into 3 stages: {BT, PLF, JLF, KH}, {POB, JOB} and {PLO, JLO, KT, FB}, we have at least $4! \times 2! \times 4! = 1152$ different orders. However, experiments with additional subjects show that the orders picked by human subjects display common features.

If the sandwich construction is proceeding smoothly, then the maker's standard program can be followed but in the event of a mishap, some other program has to be invoked to correct the error. How that happens is unsettled, but a way station that is needed for its solution is some way of detecting that something has gone awry. This component of the problem has been modeled by Yi[?] and we can gain a tremendous amount of understanding by following its structure. The task is to recognize the stage of the sandwich construction process and in particular, to identify the subtask underway. Figure 7.2 shows two frames from the on-line recognition algorithm. In line with the routines assumption, the algorithm tests for color information in the foveal region and also classifies the momentary trajectories of the

SEQUENTIAL TIME INTERVALS

SUBTASK LIST	1	2	3	4	5	6	7	8	9	10
BT	ABC									
PLF		A	C		B					
JLF		BC				A				
KH			AB	C						
POB				A	C	B				
JOB				B		C	A			
PLO					A				B	C
JLO									C	AB
KT							C	AB		
FB							B	C	A	

Table 7.3: Scheduling of Subtasks. The task is decomposed into ten subtasks including BT (putting bread on table), PLF (taking peanut butter lid off), JLF (taking jelly lid off), KH (grabbing knife in hand), POB (spreading peanut butter on bread), JOB (spreading jelly on bread), PLO (putting peanut butter lid on), JLO (putting jelly lid on), KT (putting knife on table), and FB (flipping bread to make an sandwich). Letters A, B and C denote the orders of subtasks taken by 3 subjects, e.g. in the first 2 steps subject C put bread on the table and took jelly lid off.

A. Spreading peanut butter on the bread B. Spreading Jelly on the bread

Figure 7.2: Steps in sandwich making recognized by a computational model that uses Bayesian evidence pooling to pinpoint steps in the recipe by observing the sandwich constructor's actions. The algorithm has access to the central one degree of visual input cetered at the gaze point, which is delimited by the cross-hairs. Also the position and orientation of each wrists is measured. The label in the upper left is the algorithm's estimate of teh stage in the recipe.

two wrists. It aslo keeps track of the temporal stage of the construction. All of this information provides evidence, and the computational task is to turn this evidence into a task estimate.

It turns out that the Bayesian algorithm described in Chapter four for estimating velocities works nicely for this problem also. It just has to be scaled up to the more elaborate venue. In the velocity estimation algorithm, three sources of information, two velocity estimates and a prior, were combined in another node. This calculation can be expressed graphically as shown in Fig. 7.3

Armed with this notation, the larger graph for sandwich-making of Fig. 7.4A can be easily interpreted. The shaded nodes represent data that is directly measured by a routine. Once these measurements are taken their evidence can be quickly propagated throughout the graph. The graphical nodes are shorthand for the different numbers of state values as shown by the inset table. Fig. 7.4B shows how the overall construction is handled. Suppose the first task done is t_1. Then there is a probability p of doingt_2 next and a probability q of doing t_4 after that. Each time a task is posited, the evidence for it is evaluated.

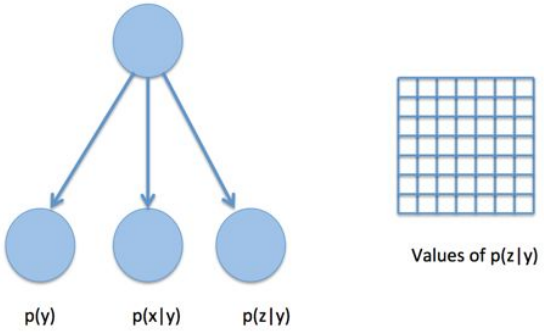
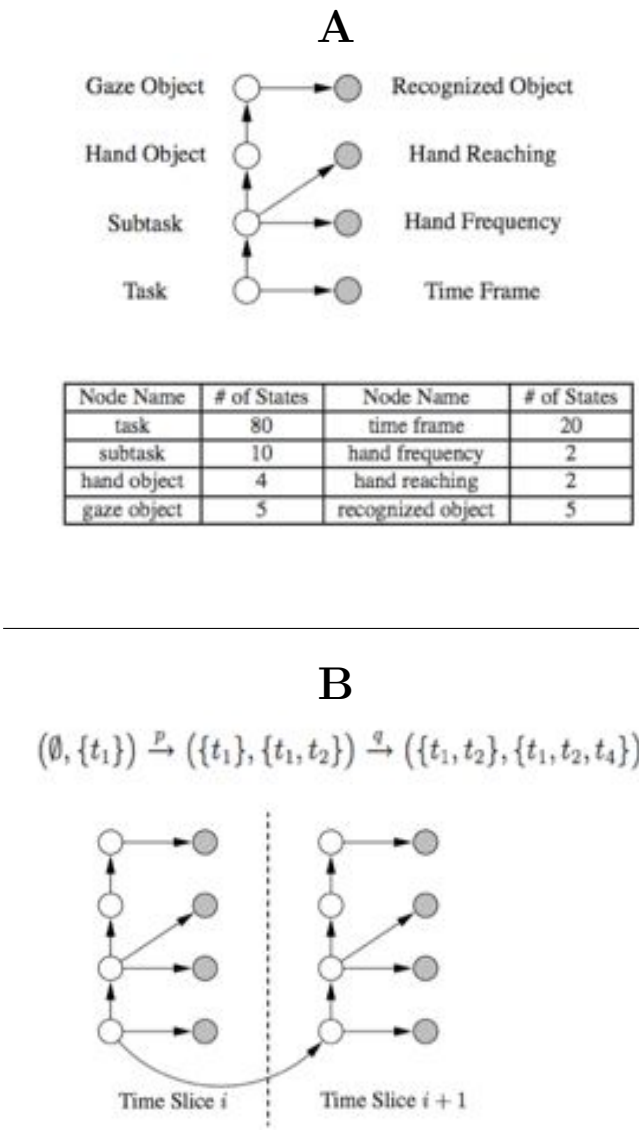


Figure 7.3: (*Left*)The model for velocity detection turns out to be generalizable to any graph. Evidence can be propagated very quickly throughout graphs that have no loops, such as the larger sandwich-making evidence graph. (*Right*) The nodes in the graph summarize many different states that have different numerical values. For example the node $p(z|y)$ is shorthand for the distribution over its many possible states.

At this point the whole problem of making a sandwich is far from being solved, however now there is some insight as to the steps in the process. As the actions are generated, an ancillary algirthm can monitor their outcomes. If the desired next stage, where a subtask has been achieved is communicated with a high probability, all is well, otherwise some fault handling must be invoked.

7.3 Multi-tasking

Before taking up the issue of how the brain might run multiple programs at once, it may be helpful to review how silicon computers do this. The simplest strategy would be to put them in a queue and run them in the queue order, but the problem with that strategy is that a very long unimportant program may hog the processor, making more important programs wait. Thus the standard solution is to break processor time into quanta of about 100 milliseconds or so and work or multiple jobs a little bit. A good analogy is a simultaneous chess exhibition where a grandmaster will play several opponents simultaneously. The grandmaster typically will have the opponents arranged in a rectangle around her so she can transit standing positions at



Node Name	# of States	Node Name	# of States
task	80	time frame	20
subtask	10	hand frequency	2
hand object	4	hand reaching	2
gaze object	5	recognized object	5

Figure 7.4: The computational model behind the sandwich recipe recognizer. A. A graph similar to the Bayes network described in Chapter two but more elaborate is used to weight measurements from sensors (grey nodes) and propagate it to a 'Task' node. The graph shown is a convention for one that has many nodes. The exact numbers used are shown in the inset table. B. The sandwich making is coded as a sequence of large-scale actions, such as spreading peanut butter. Such tasks are denoted by a specific index. The k^{th} task would be t_k. The task graph by itself codes all the different sequences of tasks that can be used in sandwich making.

each of the boards easily. What happens is that she generally will know who the best players on the other side of the board are, and so will stop opposite them and spend more time thinking before choosing a move. In the same way a processor transits the programs, executing the next set of the instructions in each program, and spends the most time on the important programs' instructions.

What sense does it make to think of the neural circuitry as doing more than one thing at a time? As mentioned before, this obviously happens for the neural circuitry that control the basic life support functions of the body. As for cognitive behaviors that require complex sensory-motor correspondences, the issue is very much open. Certainly if the brain was forced for some reason to do single programs sequentially, it would face all the disadvantages that silicon computers do. This would also be exacerbated by the slowness of the neural circuitry, for as we know, in processing a single program, the smallest step usually requires on the order of 200 milliseconds minimum, and usually much more, say seconds. These small differences are important, particularly for some motor behaviors. Land has shown that the difference between expert cricket batsman and duffers is that the experts are 100 milliseconds faster in anticipating the bounce point of the ball.

The facility with which the brain can do two things at the same time has received extensive study by psychologists. In particular, Ruthruff et al[?] had subjects do a dual task where they had subjects map one of three tones and one of three letters visually displayed onto two key presses. Subjects had to correctly press the appropriate two keys when the sound and letter were displayed simultaneously. The data are shown in Fig 7.5. The subjects fell into two groups. In one group, subjects were able to see the task as a combined tone and letter task and therefore could simultaneously press the two answer keys. In another one task was done either about 300 milliseconds before or after the other.

The authors relate this data to a well known test of rapid serial visual representation or RSVP. When searching for two images one after the other in a very rapidly presented set of images subjects have difficulty responding correctly when the second target is within 300 milliseconds of the first. An example query would be indicate when a chicken follows a telephone."

Given what we know about working memory, it is hard not to suggest that it is the reason for the delays. In terms of programs, if the program has no variable arguments then there is no issue, but if there is a variable argument, then apparently the neural circuitry needs time to establish the

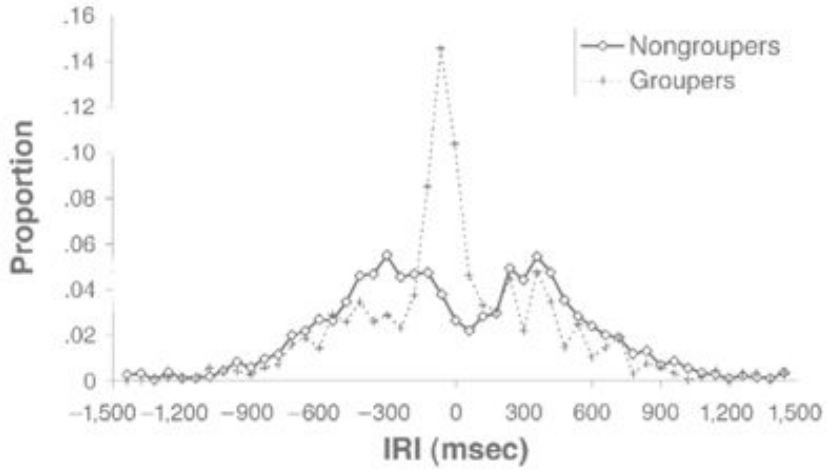
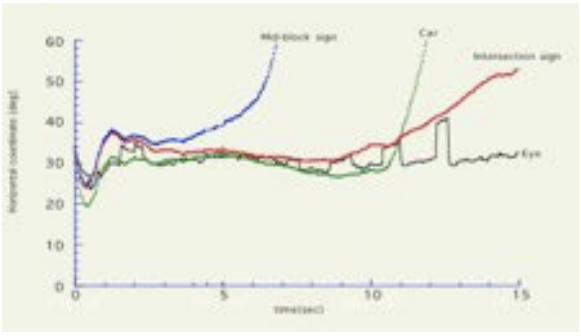


Figure 7.5: The Pashler laboratory study of multitasking. Subjects viewing a screen see one of three characters and have to respond by pressing a corresponding key. Simultaneously they hear one of three tones and have to press a corresponding key. The plot shows that some subjects (the 'Groupers') can treat the two tasks as a single complex task and press both keys simultaneously. Other subjects order the tasks so that one is handled before the other. Elaborate controls imply that the Nongroupers are dealing with a cognitive bottleneck.

temporary binding. In other words to move a pointer.

In most ecological settings there is usually sufficient time to move a pointer if need be. Ongoing behaviors need to continue for several seconds and need to manage multiple tasks at once as illustrated with an experiment by Shinoda and Hayhoe who had subjects drive a virtual car. This was done by having subjects use a car simulator that had a steering wheel and pedals (See Figure 7.6A) while wearing a head mounted binocular display that displayed a scene of a small town. Subjects were instructed to follow a lead car as well as obey traffic signs. This meant that when approaching an intersection, subjects had the dual tasks of following and looking for traffic signs at the intersection. What you can see from a typical subject's data shown in Figure 7.6B is that the two tasks are managed by switching the gaze successively from one to the other. At one point in time the subject is looking at the car and, at about 750 milliseconds later, the gaze is switched

A



B

Figure 7.6: A) Example of a Virtual Reality car simulator used to gather driving data. B) Eye traces from a similar environment where human drivers follow a lead car and obey traffic signs show gaze repeatedly sampling intersection sign and the lead car location.

to the sign, and then back to the car and so on. Remember that the visual cortical memory is retinotopic, so a major fraction of the neurons are sensitive to gaze position. Thus their firing patterns are very different when gaze is changed. How are we to interpret what is going on here with respect to running multiple programs? one possibility is that the follow program is switched on when gaze is following the car and the sign program is switched on when looking at the sign. But this seems very unlikely for two reasons. One is that, taking a cue from the dual-task experiments, there would be the overhead of turning the programs on and off. In addition, just from the Thorpe data we know that from a standing start it takes about 200 milliseconds to compute an answer, so stopping and starting would have a temporal overhead. The second reason for positing simultaneously active programs is that in order to change gaze the new gaze point for the next program must be computed while the current program is running - thus for at least this part of the computation, the two programs have to be running simultaneously.

7.4 Humanoid avatar models

In this discussion of program resource allocation, the focus has been on the internal architecture, but host of additional issues arise when considering

the external interactions of the body immersed in the world. To study these issues a new tool has become available. Research programs that focus on embodiment have been facilitated by the development of virtual reality (VR) graphics environments. These VR environments can now run in real time on standard computing platforms. The value of VR environments is that they allow the creation of virtual agents that implement complete visuo-motor control loops. Visual input can be captured from the rendered virtual scene, and motor commands can be used to direct the graphical representation of the virtual agent's body. Terzoupolous and Rabie[28] pioneered the use virtual reality as a platform for the study of visually guided control. Embodied control has been studied for many years in the robotics domain, but virtual agents have enormous advantages over physical robots in the areas of experimental reproducibility, hardware requirements, flexibility, and ease of programming.

During the course of normal behavior humans engage in a wide variety of tasks, each of which requires its own perceptual and motor resources. Thus the brain's 'operating system' must be mechanisms that allocate resources to tasks. Understanding this resource allocation requires an understanding of the ongoing demands of behavior, as well as the nature of the resources available to the human sensori-motor system. The interaction of these factors is complex, and that is where the virtual human platform can be of value. It allows us to imbue our artificial human with a particular set of resource constraints. We may then design a control architecture that allocates those resources in response to task demands. The result is a model of human behavior in temporally extended tasks that may be tested against human performance.

The aim is to deconstruct the mechanisms that manage resource allocation. By building a complete humanoid and giving it visuo-motor resources, we aim to show that the unity of attention is actually a hierarchy of interdependent resource allocation programs. For a demonstration we will use Sprague's virtual human model 'Walter.'? Walter has physical extent and programmable kinematic degrees of freedom that closely mimic those of real humans. His graphical representation and kinematics are provided by the DI-guy package developed by Boston Dynamics. This is augmented by the Vortex package developed by CMLabs for modeling the physics of collisions. The crux of the model is a control architecture for managing the extraction of information from visual input that is in turn mapped onto a library of motor commands. The model is illustrated on a simple sidewalk navigation task

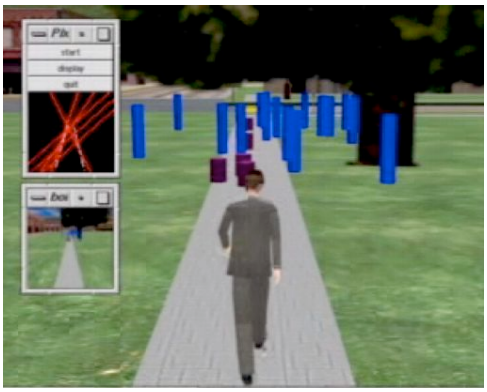


Figure 7.7: The Walter simulation. The insets show the use of vision to guide the humanoid through a complex environment. The upper inset shows the particular visual routine that is running at any instant. The lower insert shows the visual field in a head-centered frame.

that requires the virtual human to walk down a sidewalk and cross a street while avoiding obstacles and collecting litter. The movie frame in Figure 7.7 shows Walter in the act of negotiating the sidewalk which is strewn with obstacles (blue objects) and litter (purple objects) on the way to crossing a street.

The human operating system model The central tenet of Walter's control architecture is that, although a large library of microbehaviors is available to address the goals of the agent, at any one time, only a small subset of those are actively engaged as shown in Figure 7.8. The composition of this set is evaluated at every simulation interval, which is 300 milliseconds commensurate with the eyes' average fixation time.

Think of the control structure in terms of an operating system, as the basic functions are needed to implement it are similar, as shown in Figure 7.9. The behaviors themselves, when they are running, each have distinct jobs to do. Each one interrogates the sensorium with the objective of computing the current state of the process. Once the state of each process is computed then the action recommended by that process is available. Such actions typically involve the use of the body. Thus an intermediate task is the mapping of those action recommendations onto the body's resources. Finally the behavioral composition of the microbehavior set itself must be chosen. It is likely that,

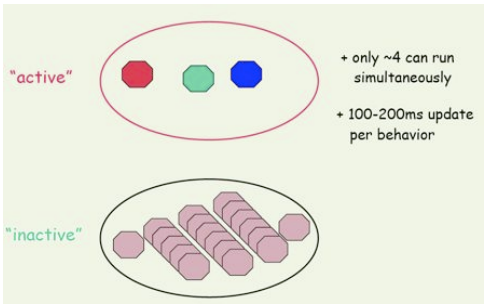


Figure 7.8: The model assumes that humans have an enormous library of behaviors that can be composed in small sets to meet behavioral demands. When an additional behavior is deemed necessary it is activated by the 'operating system.' When a running behavior is no longer necessary, it is deactivated.

similar to multiprocessing limitations on silicon computers, that the brain has a multiprocessing constraint that allows only a few microbehaviors to be simultaneously active.

Addressing the issues associated with this vantage point leads directly to an abstract computational hierarchy. The issues in modeling vision are different at each level of this hierarchy. Table 1 shows the basic elements of our hierarchy highlighting the different roles of vision at each level.

The behavior level of the hierarchy addresses the issues in running a microbehavior. These are each engaged in maintaining relevant state information and generating appropriate control signals. Microbehaviors are represented as state/action tables, so the main issue is that of computing state information needed to index the table. The arbitration level addresses the issue of managing competing behaviors. Since the set of active microbehaviors must share perceptual and motor resources, there must be some mechanism to arbitrate their needs when they make conflicting demands. The context level of the hierarchy maintains an appropriate set of active behaviors from a much larger library of possible behaviors, given the agents current goals and environmental conditions.

The issues that arise for vision are very different at the different levels of the hierarchy. Moving up the levels:

1. At the level of individual behaviors, vision provides its essential role of

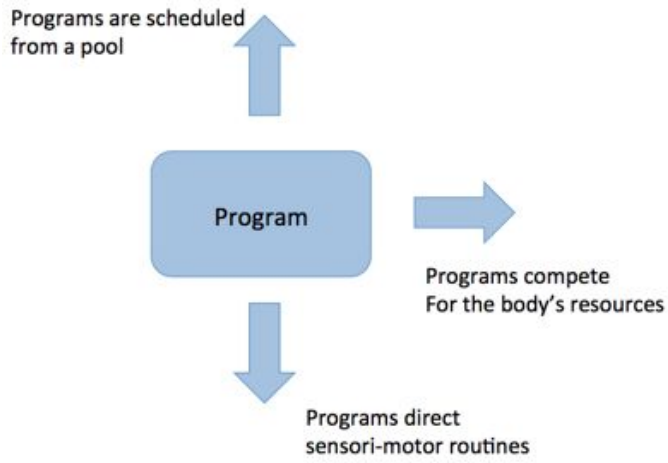


Figure 7.9: The operating system uses three distinct levels of abstraction. 1) At the most basic level sensory routines define the state of a behavior. 2) At an intermediate level, behaviors compete with each other for the body's resources.3) At the most abstract level the composition of behaviors must be continually adjusted.

Abstraction Level	**Problem Being Addressed**
Context	Current set of behaviors B is inadequate for the task. Have to find a new set
Arbitration	Active behaviors may have competing demands for body, legs, eyes. Conflicts have to be resolved
Routines	Need to get state information
	The current state needs to be updated to reflect the actions of the body

Table 7.4: The organization of human visual computation from the perspective of the microbehavior model.

computing state information. The issue at this level is understanding how vision can be used to compute state information necessary for meeting behavioral goals. Almost invariably, the visual computation needed in a task context is vastly simpler than that required general purpose vision and, as a consequence, can be done very quickly.

2. At the arbitration level, the principal issue for vision is that the center of gaze is not easily shared and instead generally must be allocated sequentially to different locations. Eye tracking research increasingly is showing that all gaze allocations are purposeful and directed toward computing a specific result.[8–10] Our own model[11] shows how gaze allocations may be selected to minimize the risk of losing reward in the set of running behaviors.

3. At the context level, the focus is to maintain an appropriate set of microbehaviors to deal with internally generated goals. One of these goals is that the set of running behaviors be response to rapid environmental changes. Thus the issue for vision at this level is understanding the interplay between agenda-driven and environmentally-driven visual processing demands.

Note that each level in the hierarchy can be seen as the subject of traditional explorations in 'attention,' as the relevant issue is about resources. Yet the processing at each level is very different. The hierarchy immediately presents us with a deconstructed description of attention and has an associated set of questions that are peculiar to the different levels: How do the microbehaviors get perceptual information? How is contention managed? How are sets of microbehaviors selected? Subsequent sections make use the hierarchical structure to address each of these in turn, emphasizing implications for vision.

State estimation using visual routines The first question that must be addressed is how individual programs map sensory information to internal state descriptions. This information is gathered by deploying visual routines. The arguments for visual routines have be made in the previous chapter.

Regardless of the specific methods of individual routines, each one outputs information in the same abstract form: the state needed to guide its encompassing microbehavior. The next section describes how an avatar can learn to use this information to guide its parent program.

Learning programs Once state information has been computed, the next step is to find an appropriate action. Each microbehavior stores actions in a state/action table. Such tables can be learned by reward maximization algorithms: Walter tries out different actions in the course of behaving and remembers the ones that worked best in the table. The reward-based approach is are motivated by studies of human behavior that show that the extent to which humans make such trade-offs is very refined[16] as well as studies using monkeys that reveal the use of reinforcement signals in a way that is consistent with reinforcement learning algorithms.[17]

Formally, the task of each microbehavior is to map from an estimate of the relevant environmental state s, to one of a discrete set of actions, $a \in A$, so as to maximize the amount of reward received. For example the the obstacle avoidance behavior maps the distance and heading to the nearest obstacle $s = (d, \theta)$ to one of three possible turn angles, that is, $A = \{-15^o, 0^o, 15^o\}$. The *policy* is the action so prescribed for each state. The coarse action space simplifies the learning problem.

Our approach to computing the optimal policy for a particular behavior is based on a standard reinforcement learning algorithm, termed Q-learning.[18] This algorithm learns a value function $Q(s, a)$ for all the state-action combinations in each microbehavior. The Q function denotes the expected discounted return if action a is taken in state s and the optimal policy is followed thereafter. If $Q(s, a)$ is known then the learning agent can behave optimally by always choosing $\arg\max_a Q(s, a)$(See Appendix for details). Figure 7.10 shows the table used by the litter collection microbehavior, as indexed by its state information.

Each of the three microbehaviors has a two-dimensional state space. The litter collection behavior uses the same parameterization as obstacle avoidance: $s = (d, \theta)$ where d is the distance to the nearest litter item, and θ is the angle. For the sidewalk following behavior the state space is $s = (\rho, \theta)$. Here θ is the angle of the center-line of the sidewalk relative to the agent, and ρ is the signed distance to the center of the sidewalk, where positive values indicate that the agent is to the left of the center, and negative values indicate that the agent is to the right. All microbehaviors use the logarithm of distance in order to devote more of the state representation to areas near the agent. All these microbehaviors use the same three-heading action space described above. Table 7.5 shows Walter's reward contingencies. These are used to generate the Q-tables that serve as a basis for encoding a policy. Figure 7.11 shows a representation of the Q-functions and policies for the

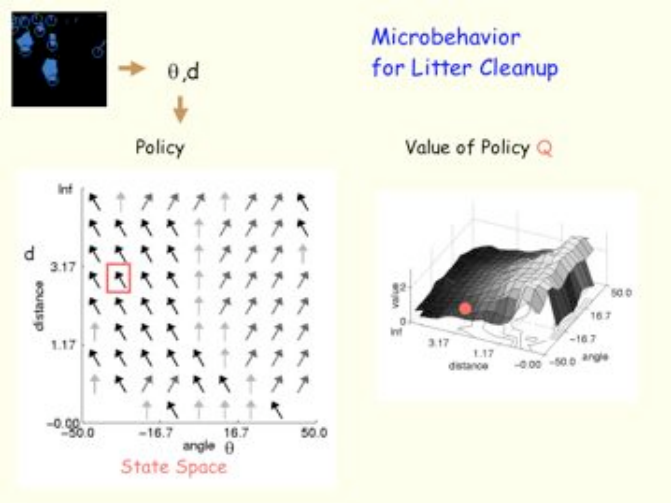


Figure 7.10: The central portion of the litter cleanup microbehavior after it has been learned. The color image is used to identify the heading to the nearest litter object as a heading angle θ and distance d. Using this state information to index the table allows the recovery of the policy, in this case $heading = -45^o$, and its associated value. The fact that the model is embodied means that there is neural circuitry to translate this abstract heading into complex walking movements. This is true for the graphics figure that has a 'walk' command that takes a heading parameter.

Outcome	Immediate Reward
Picked up a litter can	2
On sidewalk	1
Collision free	4

Table 7.5: Walter's reward schedule

three microbehaviors.

When running the Walter simulation, the Q-table associated with each behavior is indexed every 300 milliseconds. The action that is the policy is selected and submitted for arbitration. The action chosen by the arbitration process is executed by Walter. This in turn results in a new Q-table index for each microbehavior and the process is repeated. The path through a Q-table thus evolves in time and can the visualized as a thread of control analogous to the use of the term thread in computer science.

7.5 Program arbitration

A central complication with the microbehavior approach is that concurrently active microbehaviors may prefer incompatible actions. Therefore an arbitration mechanism is required to map from the demands of the individual microbehaviors to final action choices. The arbitration problem arises in directing the physical control of the agent, as well as in handling gaze control and each of these requires a different solution. This is because in Walter's environment, his heading can be a compromise between the demands of different microbehaviors but his gaze location is not readily shared by them. A benefit of knowing the value function for each behavior is that the Q-values can be used to handle the physical arbitration problem in each of these cases.

Heading Arbitration Since in the walking environment each behavior shares the same action space Walter's heading arbitration is handled by making the assumption that the Q-function for the composite task is approximately equal to the sum of the Q-functions for the component microbehaviors:

$$Q(s,a) \approx \sum_{i=1}^{n} Q_i(s_i,a), \tag{7.1}$$

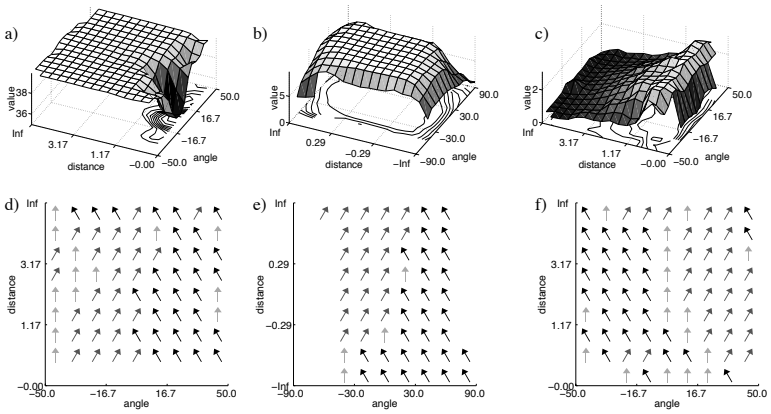


Figure 7.11: Q-values and policies for the three microbehaviors. Figures a)-c) show $\max_a Q(s, a)$ for the three microbehaviors: a) obstacle avoidance, b) sidewalk following and c) litter collection. Figures d)-f) show the corresponding policies for the three microbehaviors. The obstacle avoidance value function shows a penalty for nearby obstacles and a policy of avoiding them. The sidewalk policy shows a benefit for staying in the center of the sidewalk $\theta = 0, \rho = 0$. The litter policy shows a benefit for picking up cans that decreases as the cans become more distant. The policy is to head toward them.

where $Q_i(s_i, a)$ represents the Q-function for the ith active behavior. Thus the action that is chosen is a compromise that attempts to maximize reward across the set of active microbehaviors. The idea of using Q-values for multiple goal arbitration was independently introduced in[19] and.[20]

In order to simulate the fact that only one area of the visual field may be foveated at a time, only one microbehavior is allowed access to perceptual information during each 300ms simulation time step. That behavior is allowed to update its state information with a measurement, while the others propagate their estimates and suffer an increase in uncertainty. The mechanics of maintaining state estimates and tracking uncertainty are handled using Kalman filters - one for each microbehavior. In order to simulate noise in the estimators, the state estimates are corrupted with zero-mean normally distributed random noise at each time step. The noise has a standard deviation of .2m in both the x and y dimensions. When a behavior's state has just been updated by its visual routine's measurement, the variance of the state distribution will be small, but as simulations show, in the absence of such a measurement the variance can grow significantly.

Since Walter may not have perfectly up to date state information, he must select the best action given his current estimates of the state. A reasonable way of selecting an action under uncertainty is to select the action with the highest expected return. Building on Equation (7.1) we have the following: $a_E = \arg\max_a E[\sum_{i=1}^{n} Q_i(s_i, a)]$, where the expectation is computed over the state variables for the microbehaviors. By distributing the expectation, and making a slight change to the notation we can write this as:

$$a_E = \arg\max_a \sum_{i=1}^{n} Q_i^E(s_i, a), \tag{7.2}$$

where Q_i^E refers to the expected Q-value of the ith behavior. In practice one can estimate these expectations by sampling from the distributions provided by the Kalman filter.

Gaze Arbitration Arbitrating gaze requires a different approach than arbitrating control of the body. Reinforcement learning algorithms are best suited to handling actions that have direct consequences for a task. Actions such as eye movements are difficult to put in this framework because they have only indirect consequences: they do not change the physical state of the agent or the environment; they serve only to obtain information.

A much better strategy is to choose to use gaze to update the behavior that has *the most to lose* by not being updated. Thus, the approach taken

here is to try to estimate the value of that information. Simply put, as time evolves the uncertainty of the state of a behavior grows, introducing the possibility of low rewards. Deploying gaze to measure that state reduces this risk. Estimating the cost of uncertainty is equivalent to estimating the expected cost of incorrect action choices that result from uncertainty. Given that the Q functions are known, and that the Kalman filters provide the necessary distributions over the state variables, it is possible to estimate, this factor, $loss_b$, for each behavior b by sampling.[21] The maximum of these values is then used to select which behavior should be given control of gaze.

Figure 7.12 gives an example of seven consecutive steps of the sidewalk navigation task, the associated eye movements, and the corresponding state estimates. The eye movements are allocated to reduce the uncertainty where it has the greatest potential negative consequences for reward. For example, the agent fixates the obstacle as he draws close to it, and shifts perception to the other two microbehaviors when the obstacle has been safely passed. Note that the regions corresponding to state estimates are not ellipsoidal because they are being projected from world-space into the agents non-linear state space.

One possible objection to this model of eye movements is that it ignores the contribution of extra-foveal vision. One might assume that the pertinent question is not which microbehavior should direct the eye, but which location in the visual field should be targeted to best meet the perceptual needs of the whole ensemble of active microbehaviors. There are a number of reasons to emphasize foveal vision. First, eye tracking studies in natural tasks show little evidence of "compromise" fixations. That is, nearly all fixations are clearly directed to a particular item that is task relevant. Second, results in[22] suggest that simple visual operations such as local search and line tracing require a minimum of 100-150ms to complete. This time scale roughly corresponds to the time required to make a fixation. This suggests that there is little to be gained by sharing fixations among multiple visual operations.

7.6 Program indexing

The successful progress of Walter is based on having a running set of microbehaviors $B_i, i = 0, .., N$ that are appropriate for the current environmental and task context. The view that visual processing is mediated by a small set of microbehaviors immediately raises two questions: 1) What is the exact na-

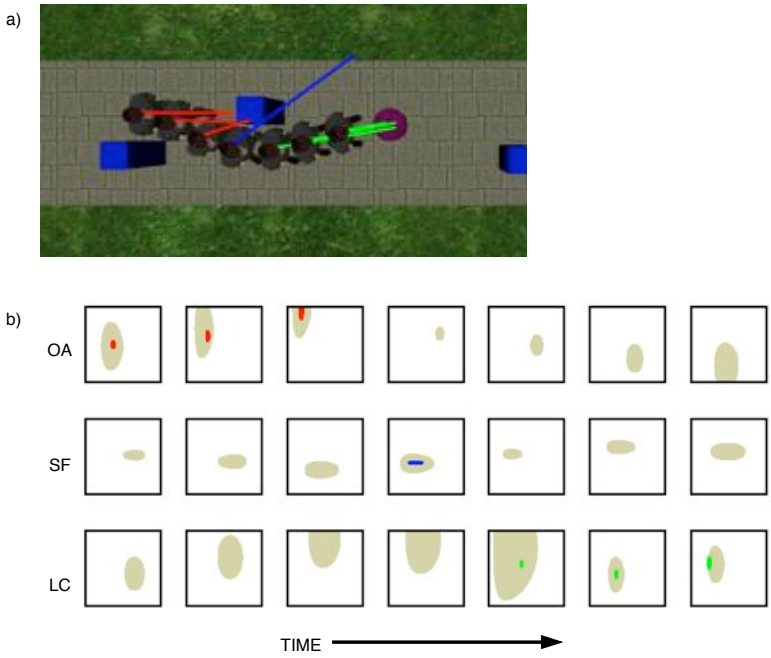


Figure 7.12: a) An overhead view of the virtual agent during seven time steps of the sidewalk navigation task. The blue cubes are obstacles, and the purple cylinder is litter. The rays projecting from the agent represent eye movements; red correspond to obstacle avoidance, blue correspond to sidewalk following, and green correspond to litter collection. b) Corresponding state estimates. The top row shows the agent's estimates of the obstacle location. The axes here are the same as those presented in Figure 7.11. The beige regions correspond to the 90% confidence bounds before any perception has taken place. The red regions show the 90% confidence bounds after an eye movement has been made. The second and third rows show the corresponding information for sidewalk following and litter collection.

ture of the context switching mechanism? and 2) What should the limit on N be to realistically model the limitations of human visual processing?

Answering the first question requires considering to what extent visual processing is driven in a top down fashion by internal goals, versus being driven by bottom up signals originating in the environment. Somewhat optimistically, some researchers have assumed that interrupts from dynamic scene cues can effortlessly and automatically attract the brain's "attentional system" in order to make the correct context switch e.g.[23] However, a strategy of predominantly bottom-up interrupts seems unlikely in light of the fact that what constitutes a relevant cue is highly dependent on the current situation. On the other hand, there is a strong argument for some bottom up component: humans are clearly capable of responding appropriately to cues that are off the current agenda.

Our model of the switching mechanism is that it works as a state machine as shown in Figure 7.13. For planned tasks, certain microbehaviors keep track of the progress through the task and trigger new sets of behaviors at predefined junctures. Thus the microbehavior "Look for Crosswalk" triggers the state NEAR-CROSSWALK which contains three microbehaviors: "FollowSidewalk", "Avoid Obstacles", and "Approach Crosswalk."

Figure 7.13B shows when the different states were triggered on three separate trials.

This model reflects our view that vision is predominantly a top-down process. The model is sufficient for handling simple planned tasks, but it does not provide a straightforward way of responding to off-plan contingencies. To be more realistic, the model requires some additions. First, microbehaviors should be designed to error-check their sensory input. In other words, if a microbehavior's inputs do not match expectations, it should be capable of passing control to a higher level procedure for resolution. Second, there should be a low latency mechanism for responding to certain unambiguously important signals such as rapid looming.

Regarding the second question of the number of active microbehaviors, there is reason to suspect that the maximum number that are simultaneously running might be modest. That is the ubiquitous observation of the limitations of spatial working memory (SWM). The original capacity estimate by Miller was seven items plus or minus two,[24] but current estimates favor the lower bound.[25] Lets hypothesize that this limitation is tied to the computer concept of 'threads' used to keep track of independent state information in independently running microbehaviors. The identification of the referents

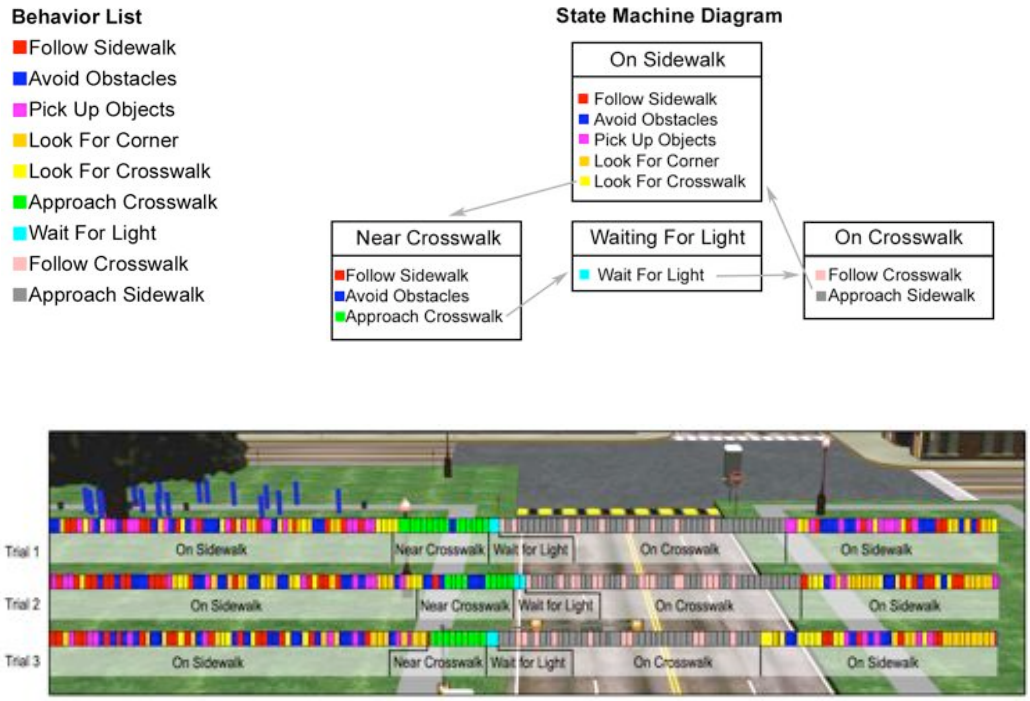


Figure 7.13: (Top left) A list of microbehaviors used in Walter's overall navigation task. (Top right) The diagram for the programmable context switcher showing different states. These states are indicated in the bands underneath the colored bars below. Bottom) Context switching behavior in the sidewalk navigation simulation for three separate instances of Walter's stroll. The different colored bars denote different microbehaviors that are in control of the gaze at any instant.

of SWM has always been problematic, since the size of the referent can be arbitrary. This has lead to the denotation of the referent as a 'chunk,' a jargon word that postpones dealing with the issue of not being able to quantify the referents. The thread concept is clearer and more specific as it denotes exactly the state necessary to maintain a microbehavior.

Although the number of active microbehaviors is limited there is reason to believe that it is greater than one. Consider the task of walking on a crowded sidewalk. Two fast walkers approaching each other close at the rate of 6 meters/second. Given that the main source of advanced warning for collisions is visual and that eye fixations typically need 0.3 seconds and that cortical processing typically needs 0.2-0.4 seconds, during the time needed to recognize an impending collision, the colliders have traveled about 3 meters, or about one and a half body lengths. In a crowded situation, this is insufficient advance warning for successful avoidance. What this means is that for successful evasions, the collision detection calculation has to be ongoing. But that in turn means that it has to share processing with the other tasks that an agent has to do. Remember that sharing means that the microbehavior has to be simultaneously active over a considerable period, perhaps minutes. Several elegant experiments have shown that there can be severe interference when multiple tasks have to be done simultaneously, but these either restrict the input presentation time or the output response time.[26] The crucial issue is what happens to the internal state when it has to be maintained for an extended period.

7.7 Credit assignment

One final issue at the operating system level is that of keeping the reward system calibrated. As noted in chapter two, when the brain is in charge of coming up with its own estimates of the value of doing things, there is lots of chances to loose one's bearings. In some helpful cases, like food intake, the body provides helpful feedback, but the more abstract programs represent a challenge. There are however some things that can be done, and furthermore these lend them selves to a computational account. One way follows directly from the multi-tasking venue. When multiple programs are simultaneously active, they can compare running estimates. Another obvious way for humans and many other animals is by observing another's behavior. If the state-action description of the demonstrator can be mapped onto the

observer's own internal representations, this allows for the observer's reward estimates to be modified accordingly.

Calibrating reward by comparing active program estimates Each active program represents some portion of the entire state space and executes some part of the composite action, but without some additional constraint they only have access to a global performance measure, defined as the sum of the individual rewards collected by all of the $\mathcal{M}$ active modules at each time step:

$$G_t = \sum_{i \in \mathcal{M}} r_t^{(i)}. \tag{7.3}$$

The central problem that we tackle is how to learn the composite Q values $Q^{(i)}(s^{(i)}, a)$ when only global rewards G_t are directly observed, but not the individual values $\{r_t^i\}$ (See Fig. 7.15).

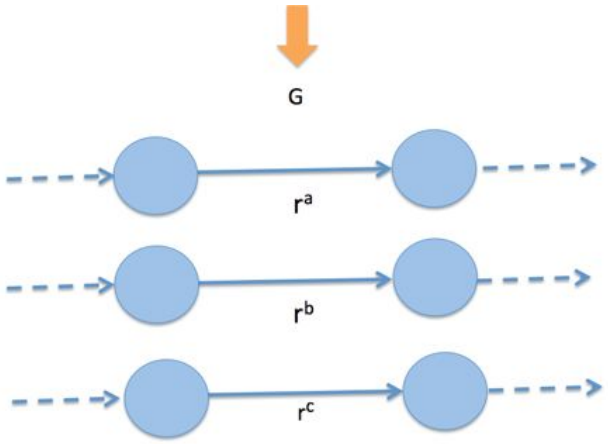


Figure 7.14: A fundamental problem for a biological agent using a modular architecture. At any given instant, when multiple modules $(i) = \{a, b, c\}$ are active and only a global reward signal G is available, the modules each have to be able to calculate how much of the rewards is their due to. This is known as the credit assignment problem. This setting simplifies the problem by assuming that individual reinforcement learning modules are independent and communicate only their estimates of their reward values. The modules can be activated and deactivated asynchronously, and may each need different numbers of steps to complete, as suggested by the diagram.

The key additional constraint that we introduce is an assumption that the system can use the sum of rewards from the modules that are co-active at any instant . This knowledge leads to the idea to use the different sets to

estimate the difference between the total observed reward G_t and the sum of the current estimates of the individual rewards of the concurrently running behaviors. Credit assignment is achieved by bootstrapping these estimates over multiple task combinations, during which different subsets of behaviors are active. Dropping the temporal subscript for convenience, this reasoning can be formalized as requiring the individual behaviors to learn independent reward models $r^{(i)}(s^{(i)}, a)$. The current reward estimate for one particular behavior i, is obtained as

$$\hat{r}^{(i)} \leftarrow \hat{r}^{(i)} + \beta\delta_{r^{(i)}} \tag{7.4}$$

where the error on the reward estimates δ_r is calculated as the difference between the global reward and the sum of the component estimates:

$$\delta_{r^{(i)}} = G - \sum_{j \in \mathcal{M}} \hat{r}^{(j)} \tag{7.5}$$

so that equation 7.4 becomes:

$$\hat{r}^{(i)} \leftarrow \hat{r}^{(i)} + \beta\Big(G - \sum_{j \in \mathcal{M}} \hat{r}^{(j)}\Big)$$

which can be informatively rewritten as:

$$\hat{r}^{(i)} \leftarrow (1-\beta)\hat{r}^{(i)} + \beta\Big(G - \sum_{j \in \mathcal{M}, j \neq i} \hat{r}^{(j)}\Big) \tag{7.6}$$

To interpret this equation: Each module should adjust its reward estimate by a weighted sum of its own reward estimate and the estimate of its reward inferred from that of the other active modules. When one particular subset of tasks is pursued, each active behavior adjusts the current reward estimates $\hat{r}_i$ in the individual reward functions according to equation 7.6 at each time step. Over time, the set of tasks that have to be solved will change, resulting in a different set of behaviors being active, so that a new adjustment is applied to the reward functions according to equation 7.6. This bootstrapping process therefore relies on the assertion that the subsets of active behaviors visits all component behaviors.

Calibrating reward by observing behavior Another important way the brain can calibrate the value of its programs is to observe the execution

of another person. The important and tricky step that is needed is that the learner must be able to take the observations of the demonstrator and translate them in to his or her own internal representation. This is not easy but monkeys can do it, as seen by famous experiments by the Rizzolatti laboratory. A neuron that fires when the monkey is reaching for a food item will also fire when the experimenter reaches for it **??**. But at the level of abstraction of the operating system, the action has to be mapped into a states-and-actions formalism programmed by reinforcement. Once that can be done the subsequent steps are very straightforward.

The data observed is going to be sequences of state-action pairs $O = \{(s_j, a_j), j = 1, \ldots, N\}$, so it is easiest to work withe the $Q-$value or action-value function $Q(s_j, a_j)$. So the observer sees a behavior and abstracts the sequence. Next, since he or she has learned a Q table, the Q values for that sequence can be easily accessed. Now what the brain would like to do is estimate the rewards R given O. This is a job for Bayes rule:

$$P(R|O) = \frac{P(O|R)P(R)}{P(O)}. \tag{7.7}$$

Now for a big assumption. Estimate $P(O|R)$ using

$$P(O|R) = \frac{1}{Z} e^{\alpha E(O,R)}$$

where Z is just a normalizing factor to make sure the probabilities sum to unity. But the expected reward $E(O, R)$ of the observations and a given reward set is just $\sum_j Q(s_j, a_j)$. But when using sets of programs the reward is just their sum so we can write the sum as:

$$\sum_i c_i \sum_j Q(s_j, a_j)$$

The important thing to take note of here is that given some observations, the only unknowns are the c_i, which reflect the relative values of the rewards. The intuition is that the form of the $Q(s_j, a_j)$ will not change. The only thing at issue is their values *relative* to each other. Since the equations turn out to be linear, they are easily solved.

7.8 Summary

The focus of this chapter was to introduce the issues associated with using a graphical agent as a proto-theory of human visuo-motor behavior. One

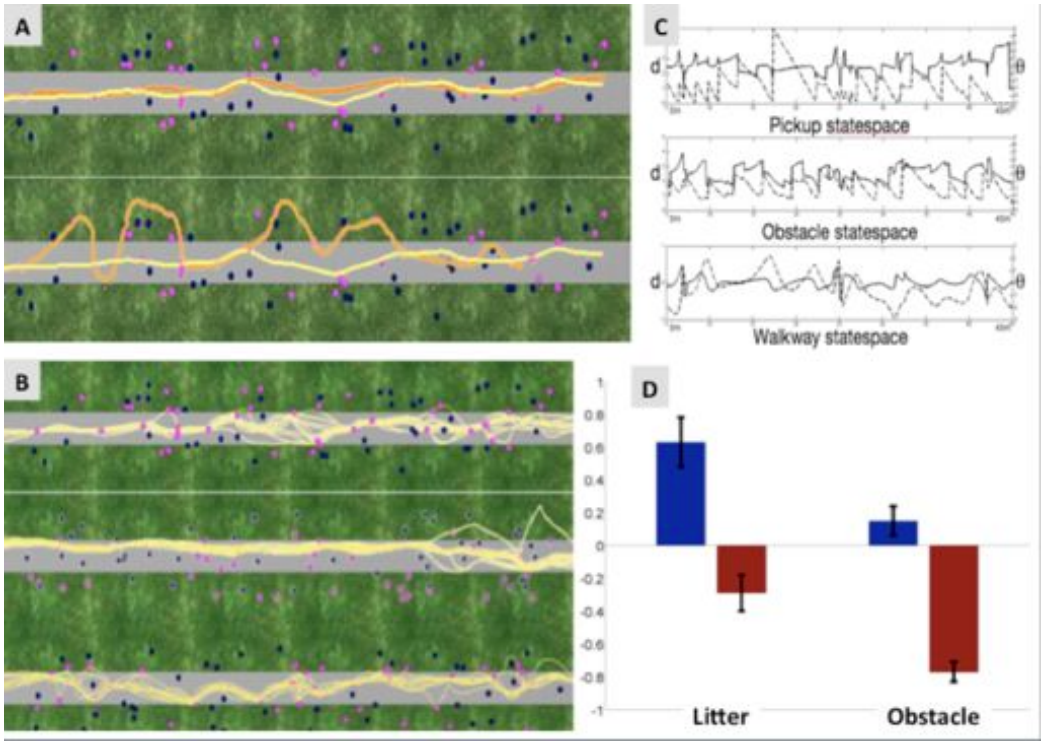


Figure 7.15: A fundamental problem for a biological agent using a modular architecture. At any given instant, when multiple modules $(i) = \{a, b, c\}$ are active and only a global reward signal G is available, the modules each have to be able to calculate how much of the rewards is their due to. This is known as the credit assignment problem. This setting simplifies the problem by assuming that individual reinforcement learning modules are independent and communicate only their estimates of their reward values. The modules can be activated and deactivated asynchronously, and may each need different numbers of steps to complete, as suggested by the diagram.

criticism of such a project is that, even though the system is vastly reduced from that needed to capture a substantial fraction of human behavior, the model as it stands is complicated and has enough free parameters so that any data from real human performance would be easy to fit. Although the system is complex, most of the constraints follow from the top-level assumption of composable microbehaviors. Once one decides to have a set of running microbehaviors, the questions of how many and when are they running are immediate. Furthermore they have ready answers in observations of human behavior in the classic observations of working memory and eye movements: Working memory suggests the number of simultaneous microbehaviors is

small; eye movements suggest when a behavior is running as each fixation is an indication of the brain's instantaneous problem being updated. Table **??** summarizes the relationships between the hierarchy used by the model and the notions of attention and working memory.

The restricted number of active microbehaviors means that there must be a mechanism for making sure that a good behavioral subset has been chosen. Such a mechanism must interrogate the environment and 1) add needed microbehaviors as well as 2) drop microbehaviors if needed to meet the capacity constraint.

The essential description of microbehaviors is captured by reinforcement learning's Q-tables that relate the states determined by vision to actions for the motor system. Indeed the commands are in coded form, taking advantage of known structure in the body that carries them out. Assuming the existence of a table as is done at the reinforcement learning level finesses important details. Thus a more detailed model is necessary to account for how the table index is created.

The reinforcement learning venue provides a different perspective on gaze allocation. One of the original ideas was a bottom-up view that gaze should be drawn to the most salient locations in the scene as represented in the image, where salience was defined in terms of the spatial conjunction of many feature points. However recent measurements have shown that eye movements are much more agenda driven than that predicted by bottom-up saliency models. For example Henderson has shown that subjects examining urban scenes for people examine place where people might be even though these can have very low feature saliency.[15] Walter's use of Q-tables suggest that to interpret gaze allocation, an additional level of indirection may be required. For example, the controller for sidewalk navigation uses gaze to update the estimate of the location of the sidewalk. In order to predict when gaze might be allocated to do this, in our model, requires knowing the uncertainty in the current estimate of the sidewalk location.

The most important benefit of the kind of model presented in this paper is that it encourages the modeler to frame experimental questions in the context of integrated natural behavior. There are dramatic differences between this perspective and traditional approaches to studying vision:

1. The desired schedule of interrupts under normal behavior has a temporal distribution that is very different than worst-case laboratory situations. In the lab, subjects are typically in extremis with respect to

reaction times, whereas natural behaviors typically allow flexibility in responding.

2. In a multiple task situation, the most important task facing the deployment of gaze is to choose the behavior being serviced. This problem is hardly considered in the search literature which concentrates on within-task saliency of individual targets.

3. The natural timescale for studying microbehavior components is on the order of 100 to 200 milliseconds, the time to estimate state information. Below that one is studying the process of state formation, a level of detail is interesting in its own right but is below the central issues in human behavioral modeling.

4. The context for the deployment of visual routines is reversed from a laboratory situation. In that situation the typical structure of a task forces a bottom-up description. The image is most often presented on a previously blank CRT screen. In a natural task, the particular test needed in a gaze deployment is known. Furthermore this test is known before the saccade is made. Thus in the natural case the situation is reversed, the test can be in place before the data is available. This has the result of making the test go as fast as possible. The speed of tests may account for the fact that fixation times in natural situations can be very short. Dwell times of 100 milliseconds are normal, less than half those observed in many laboratory studies.

All of these observations underline the importance of graphic simulation as a new tool in the study of human vision. While the model has extensive structure, each component of the structure serves a specific purpose and the whole combine to direct the performance of human behaviors. A competing performance model might look very different but would have to address these issues.

Perhaps the most important theme in recent vision research, is that no component of the visual system can be properly understood in isolation from the behavioral goals of the organism.[27,28] Therefore, properly understanding vision will ultimately require modeling complete sensori-motor systems in behaving agents. The model presented in this paper is certainly not true in all of its particulars, and it leaves many details unspecified. However, it does provide a framework for thinking about action-oriented human vision. The

fact that developing complete and correct models of human vision is such a difficult task should not stop us from trying to put as many of the pieces together as possible.

Bibliography

[1] A. Newell, *Unified Theories of Cognition*. Harvard University Press, 1990.

[2] N. Nilsson, "Shakey the robot," Tech. Rep. 223, SRI International,, 1984.

[3] D. Marr, *Vision*. Oxford: W.H. Freeman and Co., 1982.

[4] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 14–23, Apr. 1986.

[5] R. Hartley and F. Pipitone, "Experiments with the subsumption architecture," in *Proceedings of the International Converence on Robotics and Automation*, 1991.

[6] R. J. Firby, R. E. Kahn, P. N. Prokopowicz, and M. J. Swain, "An architecture for vision and action," pp. 72–79, 1995.

[7] J. J. Bryson and L. A. Stein, "Modularity and design in reactive intelligence," in *International Joint Conference on Artificial Intelligence*, (Seattle, Washington), 2001.

[8] M. Land, N. Mennie, and J. Rusted, "The roles of vision and eye movements in the activities of daily living," *Perception*, vol. 28, pp. 1311–1328, 1999.

[9] M. M. Hayhoe, D. Bensinger, and D. H. Ballard, "Task constraints in visual working memory," *Vision Research*, vol. 38, pp. 125–137, 1998.

[10] R. Johansson, G. Westling, A. Backstrom, and J. R. Flanagan, "Eye-hand coordination in object manipulation," *Perception*, vol. 28, pp. 1311–1328, 1999.

[11] N. Sprague and D. Ballard, "Eye movements for reward maximization," in *Advances in Neural Information Processing Systems 15*, December 2003.

[12] S. Ullman, "Visual routines," *Cognition*, vol. 18, pp. 97–159, 1985.

[13] P. Roelfsema, V. Lamme, and H. Spekreijse, "The implementation of visual routines," *Vision Research*, vol. 40, pp. 1385–1411, 2000.

[14] D. Ballard, M. Hayhoe, and P. Pook, "Deictic codes for the embodiment of cognition," *Behavioral and Brain Sciences*, vol. 20, pp. 723–767, 1997.

[15] D. Ballard and N. Sprague, "Attentional resource allocation in extended natural tasks [abstract]," *Journal of Vision*, vol. 2, no. 7, p. 568a, 2002.

[16] L. Maloney and M. Landy, "When uncertainty matters: the selection of rapid goal-directed movements [abstract]," *Journal of Vision*, (to appear).

[17] R. E. Suri and W. Schultz, "Temporal difference model reproduces anticipatory neural activity," *Neural Computation*, vol. 13, pp. 841–862, 2001.

[18] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning Journal*, vol. 8, May 1992.

[19] M. Humphrys, "Action selection methods using reinforcement learning," in *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 1996.

[20] J. Karlsson, *Learning to Solve Multiple Goals*. PhD thesis, University of Rochester, 1997.

[21] N. Sprague and D. H. Ballard, "Modeling embodied visual behaviors," *International Journal of Pattern Recognition and Artificial Intelligence*, p. accepted, 2006.

[22] P. R. Roelfsema, K. P.S., and H. Spekreijse, "Subtask sequencing in the primary visual cortex," *Proceedings of the National Academy of Sciences USA*, vol. 100, pp. 5467–5472, 2003.

[23] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, pp. 1489–1506, May 2000.

[24] G. Miller, "The magic number seven plus or minus two: Some limits on your capacity for processing information," *Psychological Review*, vol. 63, pp. 81–96, 1956.

[25] S. J. Luck and E. K. Vogel, "The capacity of visual working memory for features and conjunctions," *Nature*, vol. 390, pp. 279–281, 1997.

[26] H. Pashler, *The Psychology of Attention*. Cambridge, MA: MIT Press, 1998.

[27] P. Stone, *Layered learning in mutiagent systems*. MIT Press, 2000.

[28] D. Terzopoulos and T. F. Rabie, "Animat vision: Active vision in artificial animals," *Videre: Journal of Computer Vision Research*, vol. 1, no. 1, pp. 2–19, 1997.