APPROVAL SHEET

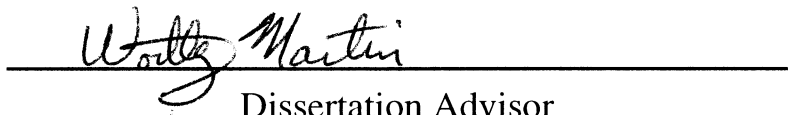
This dissertation is submitted in partial fulfillment of the

requirements for the degree of

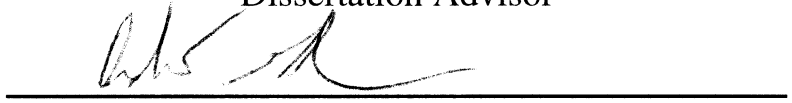
Doctor of Philosophy (Computer Science)

Author

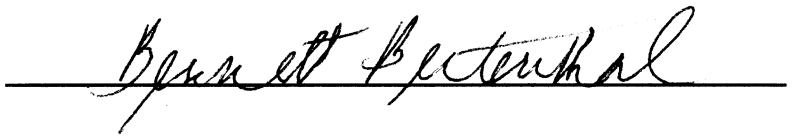
This dissertation has been read and approved by the Examining Committee:



Dissertation Advisor



Committee Chairman



Accepted for the School of Engineering and Applied Science:

Dean Richard W. Miksad
School of Engineering and Applied Science

January 1996

Representation of Local Space in Perception/Action Systems:

Behaving Appropriately in Difficult Situations

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

at the

University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy (Computer Science)

by

Frank Zachary Brill III

To my wife, Beth

and my daughters, Renee, Casey, and Megan

—FZB

Abstract

There have been two main approaches to determining action for autonomous agents: classical planning and reactive planning. Classical planners can construct plans which take into account complex interactions between the various actions the agent may take, but is computationally expensive, and requires complete knowledge of a static environment. In contrast, reactive planning systems simply map sensor inputs to actions. These mappings may be done in constant time, and the perception-based nature of reactive systems enables the agent to cope with dynamic and uncertain environments. However, reactive planning abandons the machinery needed to contend with complex situations.

This dissertation presents a new paradigm for interaction with complex, dynamic, three-dimensional environments which builds on the reactive approach. The centerpiece of this paradigm is the *effective field of view*, as implemented by marker-based representations of the local environment. The effective field of view is an extension of the standard field of view of a sensor, via representation of past sensor inputs. The effective field of view endows the agent with more information regarding the environment than the direct sensor inputs alone. By judiciously extracting and representing information for inclusion in the effective field of view based on the relevance of the information to a *task*, i.e., by *marking* the *useful* information, the competence of an autonomous agent is increased beyond that achievable by agents constructed using the pure classical or pure reactive approaches.

The effective field of view paradigm is demonstrated via an agent that interacts with a dynamic, three-dimensional, hostile virtual environment using visual perception alone. This agent is modelled after an herbivore which must collect food while avoiding obstacles and a predator. The addition of marker-based local-space representations to expand the effective field of view is shown to measurably increase the performance of such an agent.

The representations used to expand the field of view are amenable for use with advanced classical planners which relax the complete information assumptions required by older planners. This dissertation sets the groundwork for the construction of agents which capitalize on the strengths of both classical and reactive planning paradigms.

Acknowledgments

I'd like to thank some of the many people who helped me to complete this dissertation. First, thanks to my advisors, Worthy Martin and Tom Olson, who provided excellent advice and support, both technically and otherwise. This work would not exist without them, and I couldn't ask for better advisors. Thanks also to the other members of my committee, Andrew Grimshaw, Jim French, and Bennett Bertenthal, who made many helpful comments and suggestions.

The UVa vision group also provided invaluable support of my work, we are: Zhenqi Chen, Gabe Ferrer, Shalini Gupta, Glenn Wasson, and Bin Wu. Past vision group members I had the pleasure to work with are Sherrie Albrecht, Charles Bundy, Shawn Carnell, Ari Rapkin, John Taylor, Soumya Viswanathan, and Jennifer Wong. Special thanks to Glenn, who implemented the physical robot software, and helped me to hash out many of the implementation issues on the virtual robot.

The virtual reality software written by the UVa user-interface group made this project possible. I think there are several hundred people in this group, but in particular I'd like to thank Tommy Burnette, Rich Gossweiler, Shuichi Koga, and Randy Pausch. They put up with a lot of harassment from me as I continually asked for more of their software than it was designed to do. And thank you, Alice.

Finally, thanks to my family for putting up with me for all these years: my wife Beth; my daughters Renee, Casey and Megan; my parents, Frank Jr. and Penny; my in-laws, Len and Fran; and my uncle Jerry and aunt Judy.

P.S. Thank you Ann Robison, for FedEx'ing me the videotape I needed for my presentation after I had left it in Alaska.

Table of Contents

List of Figures and Tables

Chapter I: Introduction

We live in a complex world. It is impossible to know the complete state of the world at any given instant, and ridiculously impossible to predict the state of the world into the future. Yet, as human beings, we are often able to make and carry out plans to achieve our goals, given only limited information about the world. In order to do this, we rely on our senses to collect *relevant* information, based on our current goals.

Visual sensing is also complex. A large portion the human brain is devoted to visual processing, and yet the bulk of this machinery is used for concentrated processing in the fovea, a small region in the center of the human visual field of about four degrees of visual arc. The great effort required to extract information from such a small area motivates us to manage the visual processing resource to extract *only* the information relevant to the current goals.

One consequence of this concentration of visual processing is that it is often not possible to extract all the relevant information simultaneously, since the relevant information may spread across more than four degrees of visual arc. The natural solution to this problem is to remember what we saw in one area while looking somewhere else. In general, all sensors, regardless of modality, operate over a limited region of space. It is useful to remember previously sensed information, since this allows us to know more about the world than just the currently sensed information.

Successful human interaction with the world is made possible by focusing on the current goals, and such a focus must be applied to the design of artificial agents as well. When constructing autonomous agents which operate in the real world, we as system designers must confront the uncertain nature of the world and the limitations of sensing. This motivates us to adopt a *task-oriented* approach at every stage of the design. The task-oriented approach dictates that the agent's perception, action, and representation systems must be designed to explicitly support the tasks the agent is to perform.

This dissertation is concerned with the application of a task-oriented approach to the construction of autonomous agents, and the information which is sensed and represented by the agent in order to accomplish its tasks. This set of information is referred to as the agent's *effective field of view.* This work establishes the concept of the effective field of view, and discusses how the existing literature in autonomous agent design can be interpreted in terms of the effective field of view. I will describe how task-oriented design and the effective field of view can be applied to build autonomous agents which have multiple goals and use visual sensing of a dynamic three-dimensional environment. These concepts are demonstrated via an implemented agent which uses task-oriented spatial-memory

structures called markers to expand the effective field of view, and thereby measurably improve the agent's performance.

Perception, action, and representation are fundamentally intertwined. This fact must be reflected in the architecture of an autonomous agent. The agent model used in this dissertation is one in which the agent must determine and execute an appropriate action at each instant of time. Figure 1-1 shows a block diagram of such an agent. The agent receives input from the environment via its sensors, and takes action in the environment via its effectors. Within the agent, there are modules for perception, action, and memory. Information flow between the modules is bidirectional, so each module can effect the behavior of the others. Perception, action, and representation (i.e., memory) modules are tightly coupled within the agent, reflecting the necessity to address all facets of the agent design with respect to performing a given task. All modules must cooperate to perform the task.

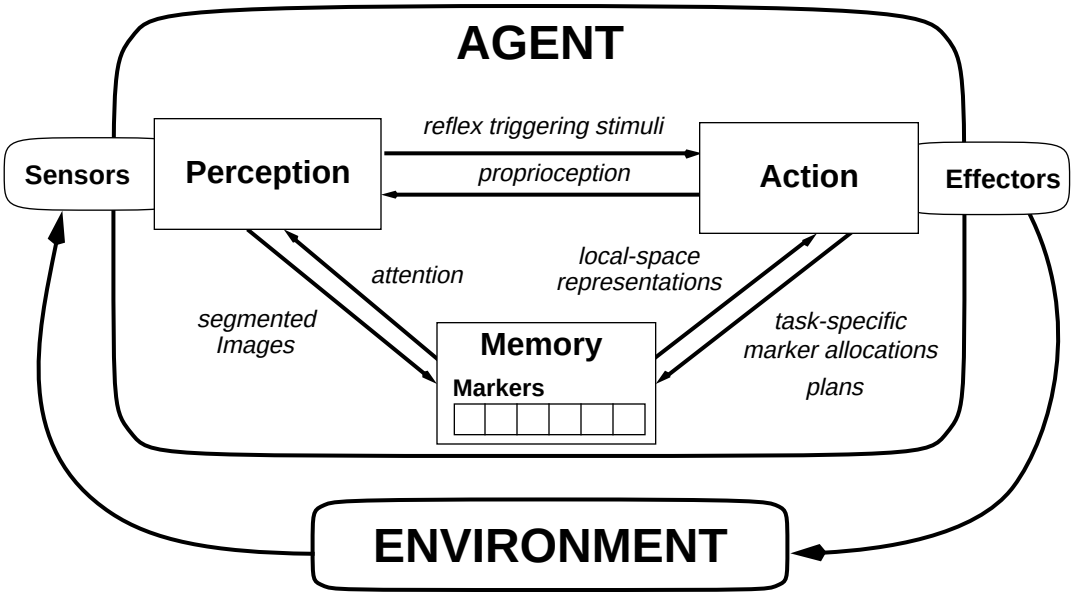


Figure 1-1: Autonomous Agent Architecture

To be successful, the agent must have available the information necessary to determine an appropriate action, whether that information is retained in memory or obtained directly from perception. This model of an agent is referred to as a *situated automaton*. A prominent model for controlling situated automata is called *reactive planning*. I will argue that the performance of reactive models can be improved by augmenting the reactive planning basic model with a sparse representation of the agent's *local space*, i.e., the configuration of objects in the agent's immediate physical environment. Further, these local space

representations may form an interface to *classical planning* models for controlling situated automata, which can further improve their performance on complex tasks.

The representation structure suggested for storing task-relevant information about the identity and location of objects in the agent's immediate environment (i.e., the *local space*) is called a *marker.* The idea of a marker was conceived and applied by others in two-dimensional environments; this dissertation develops techniques for applying markers in three-dimensional environments, and elaborates on the marker concept by identifying several types of markers, based on their relationship to the task and to each other. By recognizing the different roles of markers in the tasks, this work develops specialized strategies for acquiring, maintaining and using the markers to achieve the agent's goals. The marker is used as a means for expanding the agent's effective field of view.

In a complex world, an agent may have multiple tasks to accomplish, which may result in conflicting determinations of what action to take. I develop a system architecture which facilitates mediation among conflicting tasks, while allowing the tasks to be pursued independently when they are not in conflict. The architecture organizes the machinery for accomplishing a given task into *task-agencies*, and provides for communication (via markers) among the task-agencies in order to resolve conflicts.

A further potential difficultly with the task-oriented approach is that one potentially must design a new agent from scratch for each task one might want to accomplish. Fortunately, there is a great deal of commonality among tasks suitable for autonomous agents. I will analyze a spectrum of tasks and abstract a relatively small set of sub-tasks that can be parameterized and composed to perform a wide variety tasks.

As a demonstration of the concepts developed in this dissertation, I have implemented an agent that operates in a three-dimensional virtual environment. The agent's perception consists entirely of a sequence of images of the environment taken from the agent's perspective. The use of marker-based representations to increase the information available to the agent beyond that provided by the current sensory input (i.e., expand the effective field of view) is shown to measurably increase the performance of the agent.

This dissertation refines and extends the emerging situated automata model of agent design. The techniques for expanding the effective field of view by instantiating and manipulating marker-based representations of the environment provide the opportunity to capitalize on the strengths of both the situated automata model of agent design and the more traditional classical planning approach, thereby constructing mobile robots capable of performing tasks beyond the capabilities of those constructed using either approach alone.

Chapter II: Related Work

This dissertation addresses representations for perception/action systems operating in dynamic three-dimensional environments. The representations discussed are used to augment a reactive system and improve its performance. The representations used potentially form an interface between classical and reactive planning. The use of these representations may be used to capitalize on the strengths of classical and reactive approaches, especially in regards to sensing and representing the environment. I will concentrate on visual sensing, although the concepts to be developed Chapter III are applicable to sensing in other modalities.

This chapter briefly reviews the two approaches to determining action in the world, and visual sensing of that world. Reviews of classical planning can be found in [25, 60, 64]. A compilation of papers in classical planning, with a few papers in reactive planning as well, can be found in [3]. A collection of papers on reactive planning can be found in [41]. Rodney Brooks reviews reactive planning work at the MIT robotics lab in [12]. Papers from a recent workshop on the interaction of visual sensing with acting in the environment is in [43] Figure 2-1 depicts a time line of some of the major planning systems since the field's inception. The names on the lower portion of the diagram are classical systems, and the upper portion of the diagram names reactive systems.

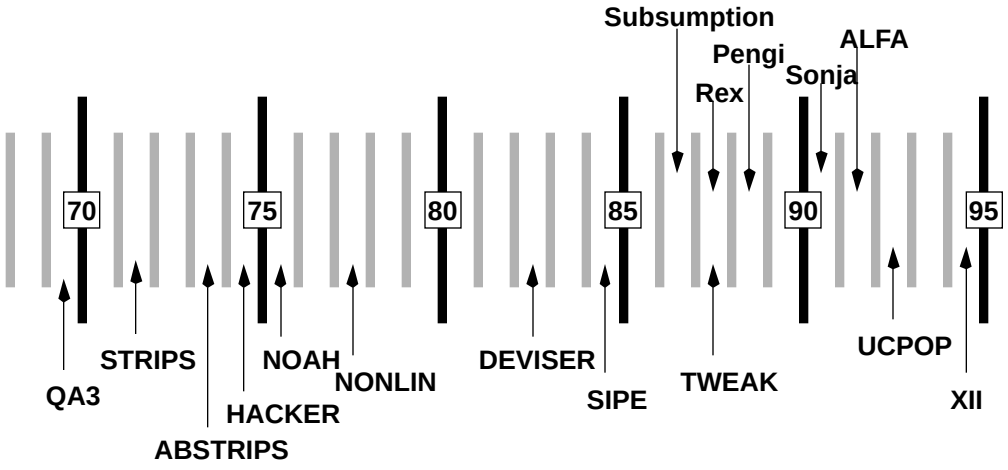


Figure 2-1: A time line of planning

2.1. Classical planning

The classical planning grew out of early work in resolution theorem proving systems such as Cordell Green's QA3 [27]. As such, it inherited many of the assumptions required by resolution theorem provers. For example:

- *omniscience*: the theorem prover has full access to all information
- *certainty*: all the information is true with complete certainty
- *consistency*: none of the information is contradictory

When resolution theorem provers are applied to planning problems, we must further assume:

- *sole cause of change*: the only changes in the world are caused by the agent
- *atomic time*: exactly one complete indivisible action occurs at a time

Given these assumptions, we can divide the world into discrete time slices that can be labelled, so that the resolution theorem prover can prove facts about the state of the world at time *t*, time *t+1*, *t+2*, etc. Under this formulation, one can modify a resolution theorem prover to construct plans via the usual resolution methodology, i.e., assume the negation of the theorem, and derive a contradiction. In planning, this means we assume there does *not* exist a state in which the desired goal is true, and start the theorem prover. The plan is produced as a side-effect of deriving the contradiction.

A major problem with this approach is the action representation required. We need to know the new state of the world after an action is executed, i.e., given the state of the world at time *t* and a description of the action taken at time *t*, we must to be able to derive all the relevant facts about the world at time *t+1*. Unfortunately, this means that in describing an action, we must not only specify all of the changes that occur in the world as a result of the action—we must also specify all of those things that *do not* change as a result of the action. This is an example of *the frame problem* in its purest form [46].

Fikes and Nilsson dealt with this incarnation of the frame problem in the STRIPS system [22] by decoupling resolution theorem proving from the search through world states. Resolution theorem proving was only used to determine facts *within* a world state. To move between world states they introduced what became knows as the STRIPS action representation, which consisted of an "add list" and a "delete list." Each action had an add and delete list associated with it. To generate the world state at time *t+1* after an action is executed, take the description of the world at time *t*, add all the statements in the action's add list, and delete all the statements in the delete list. All statements that are not explicitly mentioned in the add or delete lists are assumed to be unchanged.

The STRIPS planner was still left with a difficult combinatorial search problem; one still had to decide what action to apply at each world state. Heuristics were used with

limited success. Then in ABSTRIPS, the next version of the STRIPS system, Earl Sacerdoti introduced the idea of a hierarchy of abstraction spaces in which the planner first solved the problem at a high level, ignoring many details, and then used the solution to this easier problem to guide the searches at the more detailed levels [53]. The abstraction spaces were constructed by simply ignoring some of the preconditions of the actions. The choice of which preconditions to ignore was guided by *a priori* assigning *criticality* values to literals in the preconditions. This approach dramatically improved the performance of the planner.

ABSTRIPS laid the groundwork for Sacerdoti's next planner, NOAH [54]. By ignoring *all* the preconditions, one can start with a empty plan which is assumed to achieve the goal (i.e., it has the single "action" labelled "achieve goal"). Then one continually adds "refinements" to the plan which fill in the details of the "achieve goal" action. Actions are considered which achieve goals or portions of goals independently, and action ordering is only imposed as necessary. Sacerdoti used "critics" to impose such orderings as needed to resolve conflicts in the plans. This formulation fundamentally alters the search space from a search through a space of world states to a search through a space of plans of increasing detail. This type of planner is referred to as a "nonlinear" planner, since it does not consider a linear progression through a sequence of world states. NOAH is the foundation of all modern classical[1] planners.

By changing the search space, NOAH was able to solve a number of problems without backtracking (i.e., without "search"), notably the blocks-world problem known as the "Sussman Anomaly" which caused trouble for Sussman's HACKER system [58]. However, the critics in NOAH still made choices among alternative orderings—choices which may later turn out to have been wrong. This made NOAH incapable of solving a number of problems, since it didn't save the choice points for backtracking. Tate's NONLIN planner [59] reintroduced backtracking search to the planning problem.

Over the next few years, additional enhancements were made to the basic nonlinear planning algorithm. For example, Vere's DEVISER [63] considered planning in time, i.e., some actions must occur within a time window, and Wilkins' SIPE [66] had specialized mechanisms for dealing with limited resources. Eventually, Chapman examined the state of the art in planning and constructed TWEAK [15], a relatively simple planner without the extra mechanisms for time, resources, etc., which he proved correct and complete. At the heart of TWEAK was the *modal truth criterion*, which lists the necessary and sufficient conditions for ensuring that a statement is true at a given point in time. Chapman fur-

1. Please excuse the oxymoron. I think the meaning of "modern classical" in this context is clear.

ther proved that if a plan's actions contain conditional effects, evaluating the modal truth criterion is NP-hard. Since TWEAK evaluated the modal truth criterion in its innermost loop, Chapman concluded that planning with expressive action languages (such as those with conditional effects) was unlikely to be fruitful. Chapman rejected classical planning, and went on to make contributions in the realm of reactive planning.

Since publication of Chapman's landmark paper, it has been pointed out that it is not necessary to evaluate the modal truth criterion in a planner's innermost loop, since the planner need not evaluate whether something is true, but rather only to be sure to insert the actions needed to make it true. The UCPOP planner takes advantage of this fact to construct plans with actions having conditional effects and universal quantification in both preconditions and effects [64]. However, even simple planning (i.e., without conditional effects) is PSPACE complete if actions can have more than two conjuncts in their preconditions [14], and given sufficiently powerful action representations, planning is undecidable [20]. See [21] for extensive results on the computational complexity of planning.

Very recently, some work has been done on relaxing the basic underlying assumptions in planning. For example, the XII [26] planner begins to address the problem of planning with incomplete information. The XII planner constructs plans for "softbots," intelligent agents that perform tasks in the artificial world of a computer network.

2.2. Reactive planning

Classical planners were initially developed to be applied by mobile robots; in fact STRIPS, ABSTRIPS, and NOAH all built plans to be executed by SRI's robot named Shakey. However, about the time of the construction of NOAH, planners began to find more successful applications in other domains. NONLIN was applied to planning electricity turbine overhauls, DEVISER did mission sequencing for the Voyager spacecraft, and SIPE did advanced planning for aircraft carrier missions. Classical planners were found to of limited use in mobile robots, because none of the basic assumptions made by classical planners (omniscience, certainty, consistency, sole cause of change, atomic time) are true in mobile robotics domains.

Brooks developed an engineering methodology called *subsumption* [11], which rejected the use of planning, and instead used hard-wired "behaviors" in which sensor inputs were passed through simple combinational circuits to appropriate effectors. Increasingly complex behaviors were layered over simple low-level behaviors, with the high-level behaviors taking priority, or "subsuming" lower-level behaviors if necessary. The robot simply "reacts" to its current sensor inputs. This approach became known as *reactive planning*, which is somewhat of an oxymoron, since this reactive approach is

really the antithesis of “planning.” An alternate name for this approach is *situated automata*.

The reactive planning approach made diametrically opposed assumptions from classical planning:

- *ignorance*: the agent has no information but the current sensor readings
- *uncertainty*: even sensor readings are suspect, and actions may or may not work
- *inconsistency*: different behaviors have different ideas about what to do
- *dynamic environment*: the world changes all the time, regardless of the agent
- *continuous time*: actions may be aborted in the middle of executing

Given the state of the art in sensor and mobile robotics technology, these assumptions are much closer to being true for mobile robots than are the assumptions made by classical planners. Brooks was able to construct robots that performed simple navigation and obstacle avoidance tasks in dynamic, real-world environments, which is more than can be said for classical planning systems. Bolstered by this success, and by Chapman’s pessimistic classical planning results (Chapman was Brooks’ student) Brooks went on to reject classical planning, the use of representation, and the physical symbol system hypothesis [12, 13]. It is the role of representation that is of concern in this dissertation—and it is difficult to reconcile Brooks’ rejection of representation with the subsumption architecture, since subsumption allows for the use of instance variables, i.e., representation. A system that in fact, did *not* use internal state was Gat’s ALFA [24], which was built to really test the limits of the pure memoryless approach.

Leslie Kaelbling adopted a less radical stance in the construction of Rex [36], and Gapps [37], which allowed for internal state and moreover, were advertised as such. Kaelbling’s approach to situated action was that the agent performs a mapping from input to output *mediated by its internal state*. This is the approach I adopt in this dissertation, in which I will address the form, maintenance, and use of this internal state to perform the “mediation” of the action function in mobile robots.

2.3. Visual sensing

There is a large and growing literature describing various techniques for the recovery of information concerning a scene from an image. These techniques, such as shape-from-shading [34] and depth-from-stereo [50], can deliver rough estimates of measurable aspects of a scene, such as surface-tilt and 3D-depth. These computations are accomplished by exploiting constraints imposed on the possible configurations of the scene by the physical world [42]. These constraints must be used by the visual system, due the inherent complexity of the visual problem; otherwise the problem is underconstrained. Yet even with the use of these constraints, and despite considerable effort, current visual sys-

tems are not capable of delivering an accurate, three-dimensional model of a realistically complex scene from image data.

A number of advantages and computational simplifications can be obtained by considering the tasks to which vision is applied [6]. As a complement to searching for additional constraints which might further enable the construction of a world model, we can relax the demands on the visual system by reducing the requirements of the resulting world model. In the limit, one would consider the visual component of a reactive system, i.e., no world model at all. In a reactive system, the role of vision is reduced to recognition of the object at the current point of fixation. The requirements of the visual system would remain substantial in order to perform this recognition, but the three-dimensional position of objects would not be retained by the agent over time to build up a model of the environment.

The argument put forward here and by others elsewhere [4, 6, 16], is that the purpose of vision is to enable an agent to *do the right thing*, and so the labels assigned by the visual system must be relevant to some task. Perception and action are so intertwined that they must be studied in conjunction with one another. This means that in order to effectively investigate perception beyond the lowest level mechanisms, one must do so with respect to a task to be performed by some agent. Dana Ballard is generally recognized for introducing this task-oriented approach, known as *active* or *animate* vision [6]. A number of recent papers on this approach can be found in [43].

2.4. Psychophysical evidence

This dissertation is concerned with the acquisition and maintenance of the environment representations via sensing. The particular form of the representation used extensively in this dissertation is the *deictic* representation, in which the representation basically consist of “pointers” or “markers” on objects in the physical world. The concept of marking a limited set of objects originated in the psychological literature with Pylyshyn’s FINST (Finger of INSTantiation) model of visual tracking [52]. This model proposes that a set of markers, or FINSTs, can “point” to objects in the world, and the FINSTed objects form the basis of spatial perception. Pylyshyn and Storm present several psychophysical experiments which provide evidence supporting the theory [51]. Recently, additional experiments by Yantis suggest that the spatial relationships between the marked objects are critical for visual tracking [68].

Pylyshyn’s model considered these FINSTs only as they indexed currently visible objects, but this dissertation will go further to argue that the deictic representations are maintained in memory even if the indexed object is not currently visible. There is also

psychophysical evidence to support this position; some of the strongest is provided by Attneave and Farrar [5]. Subjects studied a row of seven objects laid out on a shelf, and then turned around with their backs to the objects. The subjects were then asked questions regarding the relative positions of the objects behind them (e.g., on which side of the duck is the shoe?). Subjects performed well on the task, even though the left/right distinctions are inverted from those made when facing the objects. Subjects reported that they answered the questions as though they were viewing the objects with "eyes in the back of the head." Attneave and Farrar remarked (my emphasis):

> Our internal representation of the world around us is based in part on current sensory input, but *in a much greater part on past sensory inputs*, i.e., upon memory.

Several others have investigated the way in which such internal representations of the environment are used to aid in navigation tasks. Mittelstaedt and Mittelstaedt [48] report that geese represent information regarding a "home" location, and update the relative position of the home location by integrating optic flow information, so that they always know which way is "home," even if home is not currently visible. Muller and Wehner [49] report a similar technique is used by desert ants, except that the ants use proprioception of their movements instead of optic flow to update the home location. Loomis et al [40] report similar abilities in humans; blindfolded people can use proprioception of their movements to maintain a representation of the location of an external point.

2.5. Representations for mobile robotics

The classical form of representation for mobile robotics is perhaps exemplified by the work of Kosaka and Kak [38], in which a complete CAD-model representation is provided to the robot *a priori*. A number of techniques are used to deal with clutter in the environment that is not contained in the CAD model, and in errors in the estimated position of the robot with respect to the CAD model. These techniques enable the robot to achieve impressive performance, but acquisition of the model is not addressed, and the robot navigation is completely dependent on the existence and accuracy of this CAD model.

Agre and Chapman introduced the deictic representations to the reactive planning literature [1, 2]. They implemented an agent that used the deictic representations to play a video game called Pengi. The agent played the role of a penguin that avoided a swarm of bees. Rather than identify and label all of the bees, as would be the case in the classical paradigm, markers were placed on only the nearest (or otherwise most task-relevant) bees. The markers served as input to reactive-style planning circuitry. The maze environment of

Pengi was well-structured and rectilinear, easing the planning task considerably. In subsequent work, Chapman developed another video game agent, Sonja [16], that operated in a less-structured, though still two-dimensional environment. Sonja also interacted with a human advice-giver, accepting deictic instructions with pronouns that were bound to objects depending on the current situation.

Chapman [17] investigated visual strategies for operating in a complex environment, but only addressed the problem at the level of *intermediate* vision, and did not attempt to solve the problems associated with using an *early* vision system, such as occlusion and the underconstrained nature of the early vision problem. Further, no memory was used in either of these video-game agents; markers were only placed on currently visible items.

Maja Mataric developed the use of maps in a reactive planning context [44]. Sonar sensors and a low-resolution digital compass were used to identify landmarks and construct a distributed topological map of the environment. The map enables the robot to navigate to locations in the environment as directed by a human. These maps are useful for navigating the large-scale space, but are fundamentally different from the local-space representations used in this dissertation. The local-space markers are metric, in that they identify the locations of objects relative to the agent in a low-resolution coordinate system, whereas the Mataric maps are primarily topological. The Mataric maps are useful for navigating the large-scale space (such as using a map to drive from New York to Los Angeles) while the local representations are used for coping with the immediate surroundings (such as driving on the highway in heavy traffic, avoiding accidents). In this way, the two representations are complimentary. An extension to the work of this dissertation would be to investigate the interface between the marker-based local-space representation with a Mataric-style representation of the large-scale space.

Ian Horswill constructed a visually-guided robot which was endowed with a map, i.e., an internal topological representation of the environment [33]. His system demonstrated that a visually guided robot can be implemented for a reasonable cost and operate on the basis of low-resolution images, but it had no memory for objects it had recently seen, and could not, for instance, know to brake for an object that had passed outside the current field of view.

At the University of Rochester, Dana Ballard and several students have an ongoing research program in deictic visual behaviors, and have conducted experiments that show humans use very little memory in accomplishing natural tasks [31]. When possible, humans employ a strategy in which memory is limited to a single item, acquired immediately before use of the information. The memory in question is not spatial memory of the

type addressed in this dissertation, but rather factual memory (e.g., the color of a given item). Rochester also has a virtual reality system, and has been conducting experiments in a virtual environment. They have constructed a “go-cart” simulator, for virtual driving, and are investigating the role of deictic visual behaviors in that task [45]. Their work is primarily concerned with the use of the fovea as a marker, and in learning of the visual behaviors associated with the use of the fovea, rather than the spatial memory that may be required to direct the visual system to place the fovea a given object that may be currently outside the field of view. Spatial memory is limited to simple left-right distinctions. In contrast to the Rochester go-cart work, we will be concerned with higher resolution maintenance of multiple markers.

Classical planning addresses the general problem solving issue in potentially complex, yet static and certain environments in which the agent has complete information. Reactive planning addresses issues in dynamic, uncertain environments and real sensors, but the reactive approach limits the capability of the agents to deal with moderately complex problems. Neither approach contains an adequate account of representation of the agent’s local environment. Dealing effectively with the local environment is necessary for intelligent interaction with a dynamic, uncertain, three-dimensional environment via realistic sensors and effectors.

The most promising account of local space representation is Agre and Chapman’s marker-based approach. Markers provide a means of mapping a set of percepts into an appropriate action, enabling the agent to take consider its spatial relationships to the percepts and the spatial relationships of the percepts to each other. Hence, markers provide a powerful means of reasoning about the local space. However, Agre and Chapman use markers only to index into the current perceptual input, and they in fact deny markers should be used as memory mechanisms, insisting instead that markers should *only* index current percepts. This stance is inadequate in a three-dimensional world in which sensors are subject to occlusion and a limited field of view. But by promoting markers to true memory mechanisms, this dissertation realizes an effective means of representing the local space.

Chapter III: Building Agents for the Real World

The history of artificial intelligence has seen the construction of many "agents," most of which operated in artificial environments. The study of such agents in artificial environments resulted in a great deal of theoretical progress, but researchers have found it difficult to apply these results to agents operating the real world. A fundamental difficulty for agents attempting to apply the traditional "classical planning" approach to the real world is that the input to a classical planner is a complete and flawless description of the environment and the actions the agent can take in the environment. Clearly, it is impossible to construct such a description of the real world. This fundamental difficulty is exacerbated if some or all of the world description is to be obtained by real sensors. Real sensors are error-prone, may require computationally intensive processing, and have a limited region of space over which they operate.

As a consequence, the field of reactive planning has quite appropriately focused attention on those differences between theoretical worlds and the real world, and has resulted in additional progress in the field of mobile robotics. The reactive planning approach abandons the idea of a world description, and in its extreme form, further purports to reject representation altogether. Rather than attempting to construct general-purpose reasoners, reactive planning researchers carefully analyze a specific simple task and construct an agent to perform that task. By focusing on a specific task, the designer can limit demands on the sensors so that they only extract the information needed to accomplish a specific task. This approach makes at least some tasks in the real world tractable for artificial agents. A major tenet of this dissertation is that the real world demands a task-orientation for perceiving and acting agents.

The rejection of representation by some members of the reactive planning community is prompted by an underlying assumption that sensors can be used to uniquely determine the correct action to take based purely on the current readings of those sensors—memory is not necessary. For instance, when executing a sequence of actions, it is not necessary to remember and explicitly represent an entire sequence of world states and the current position in the sequence, since execution of the first action in the sequence will result in a change in the world that will enable the next action. It may appear that the agent is stepping through actions in a sequence, but the actual system is merely mapping the currently sensed situation to an action, a paradigm which has many appealing features.

I assert that due to the limited nature of sensors, it is sometimes not possible to simultaneously sense all the information needed to determine an effective action. I further contend that by augmenting a reactive style agent with some carefully chosen and main-

tained representations, the combination of these representations and current sensor readings can be used to determine an effective action; a determination that may not have been possible based on the current sensor readings alone.

The reactive model requires constructing a mapping from the current sensor readings to an action. The model I propose extends the reactive model through the concept of an *effective field of view,* which is created by a agent acquiring and maintaining limited representations of task-specific world attributes. Using the effective field of view concept, we can justify maintenance and use of a limited representation within the agent and still retain the attractive features of the reactive paradigm. In order to expand the effective field of view in an agent operating in a three-dimensional environment, I co-opt the idea of a *marker*, a term which I define as I intend to use it in section 3.1.1 before going on to discuss the effective field of view.

One basic difficulty with the task-oriented paradigm of mapping situations to actions arises when an agent has multiple tasks. In order to accomplish one goal, the current situation may map to some action. In order to accomplish a different goal, that same situation may map to a different action. I will discuss an architecture and techniques for resolving such conflicts in section 3.2. Section 3.3 elaborates techniques for using markers to expand the effective field of view, distinguishes different types of markers and their uses, and discusses techniques for maintaining the markers in a dynamic environment.

3.1. Representation in an uncertain world

After becoming disenchanted with the classical planning paradigm [15], Agre and Chapman developed a theory of activity [1, 2] in which situations were mapped to actions, enabling an agent to perform well without planning in the classical sense. At the heart of this theory, and what differentiated Agre and Chapman's system from previous situated automata, was the use of deictic representations, or *markers* on relevant objects in the environment. The agent's current situation (which was mapped to an action) was identified by the current state of the markers. Agre and Chapman developed this system as an alternative to classical planning, but it has since become a goal of the research community to reconcile such a system with classical planning, so as to apply the advantages of both. After defining the term marker in this section, I develop a framework based on markers which capitalizes on the strengths of both the situated and classical paradigms.

3.1.1. Markers

A *marker* is a data structure that contains two primary pieces of information: *what* an object is in terms of its role in the current task, and *where* the object is in an egocentric

coordinate system. Markers provide *spatial memory* for an agent. However, the agent does not (and cannot) remember everything about the local space. In the definition, by *what* I do not mean the marker contains a complete, objective, geometrical description of the object. Rather, the marker identifies an object relative to the current task being performed by the agent. One of the most important aspects of markers is their task dependence. Accomplishing a task in the world requires certain items be used; markers provide the place holders for these items. For example, the task of pouring a bowl of cereal requires two items: a bowl and a box of cereal. There may be many bowls and boxes of cereal in the cupboard, but we need only one of each. By placing a marker on one bowl and one box, the pouring task need only refer to *the* bowl and *the* box. The marker need not contain descriptive information such as the color of the bowl or the brand of the cereal, since they are not relevant to the current task.

Similarly, by *where,* I do not mean the object is located in some external coordinate system. The purpose of marker-based representations is to enable an agent to interact with its immediate surroundings, so the appropriate coordinate system is egocentric. I will now flesh out this definition with analogies I have found to be useful in my understanding of markers and their function. These analogies will also provide motivation for my use of markers.

3.1.1.1. Marker as pronoun

As the above definition makes clear, the primary analogy is that markers are reference pronouns augmented with hand gestures. For example, if a person in a classroom is asked “who is the teacher?” the answer might be “Professor Jones,” or it might be “her,” (with an associated finger point). The latter response is in a marker mode.

3.1.1.2. Marker as register

Markers are analogous to registers that contain information regarding important objects. To extend this analogy further, we can think of the physical world as the “main memory” of an agent [12], and the agent copies information from main memory into registers in order to improve its efficiency. The determination of the objects to mark is therefore analogous to a register allocation problem. Moreover, when data is replicated, there is a possibility that one of the copies may be modified, and not the other. In a dynamic world, it is likely that the world changes, but our internal representation of it does not. We therefore have a “cache coherence” problem with which to contend. Just as the data in a cache can fall out-of-sync with main memory, data in the markers can fall out-of-sync with the state of the world. This problem of markers falling out-of-sync with a dynamic world is ultimately a form of the frame problem [46].

In multiprocessor systems, cache coherence is traditionally handled by having writes to main memory be accompanied by broadcast messages advising all processors to invalidate their cache. No analog for such a broadcast exists in the physical world, so an agent acting on the basis of internal state always runs the risk of acting on stale data, which is part of the original motivation for the reactive approach (i.e., no internal state reduces the risk). The Agre and Chapman environment simulation updated the state of the markers automatically once they were placed; thus their environment had the equivalent of the broadcast message that not only invalidated the old data, but updated them, thereby solving the cache coherence problem.

The real world does not have such a broadcast mechanism, so agents must rely on other strategies to synchronize their internal state with the environment. A realistic treatment of the marker maintenance problem is a significant contribution of this dissertation. Fortunately, stabilities and regularities in the physical world mitigate the problem somewhat, so by limiting the amount of internal state and by monitoring the environment periodically to update that state, we sharply decrease the risk of acting on stale information.

The locations of some markers can be monitored visually, making marker maintenance for the visible markers amount to visual tracking with multiple targets. Maintenance of markers outside the field of view is rather more difficult. Use of an egocentric coordinate system requires that the marker locations be updated as the agent moves. One of the unique features of my system, as compared to reactive agents, is use of spatial memory for objects not currently visible, either due to occlusion or limited field of view.

3.1.1.3. Marker as parameter

Another helpful analogy is to think of the relationship of markers to tasks as like the relationship of parameters to subroutines. We can think of the cereal pouring task mentioned previously as a subroutine that takes two parameters: one of type BOWL, and one of type CEREAL_BOX.

In the “computability” sense, markers do not offer additional power to the autonomous robot—there are alternative strategies for accomplishing tasks without them, just as subroutines and object-oriented facilities do not add computational power to the basic Turing machine model. Yet subroutines are remarkably useful constructs. Like subroutines, the advantages to using markers are matters of efficiency and ease of implementation of useful algorithms.

3.1.2. The effective field of view

If an agent can perceive where everything is all of the time, then there is no need for spatial memory, since all of the data needed and more is available from the current sen-

sor readings. However, real sensors, visual and otherwise, are not capable of providing this level of omniscience, due to (among other things) *occlusion* and a *limited field of view.* These are real problems which are not addressed by the planning literature, either classical or reactive. Classical planning incorrectly *assumes* omniscient sensing or its moral equivalent—the world state is encoded in a perfect internal representation to which the agent has complete access. The reactive planning approach contends that the all of the relevant information can be extracted from the current sensor readings, provided that there are enough sensors and they are engineered properly. As I will describe, this can result in unnecessary profusion of sensors in the case of dealing with a limited field of view, and is simply ineffective in dealing with occlusion.

All sensors have a limited region of space over which they operate. Specifically, a sensor extracts *predicates* concerning the environment from some limited region of space. For example, a visual sensor might extract the information that "there is a red ball at location X," which is a predicate in that it is an assertion that may either be true or false. This predicate can only be extracted in the region of space in which the camera is pointing; not the area behind the camera, and furthermore not in the area in front of the camera that is so far away that it is impossible to identify a ball. I will refer to the region in which a given sensor can extract predicates as its "field of view," and apply the term to sensors which are not usually referred to as having a "view," such as sonars and contact sensors. These sensors have a field of view in the sense that there is a limited region of space over which they can extract predicates.

Furthermore, predicates extracted from real sensors have "certainties" associated with them. Outside of the sensors' *absolute* field of view, these certainties are at a minimum (i.e., complete *un*certainty). The absolute field of view of a sensor at a given instant of time is defined as the region of space in which the sensor can extract predicates with some nonzero degree of certainty at that time. Inside the absolute field of view, certainty may vary over the region, for example decreasing near the edges of the absolute field of view. Consider the camera searching for red balls; the region behind the camera is outside it's *absolute* field of view, since the camera cannot produce any information regarding the existence of red balls (or anything else) in the area behind camera. Within the camera's absolute field of view, the camera may have increasing difficulty resolving balls beyond a certain distance, thus the certainty regarding the predicate "there is a red ball at location X" may decrease with the distance to location X.

Even within the region of high certainty, the actual predicate returned by a sensor may be of limited use—it is *inferences* made from these predicates that are useful. For example, a camera does not really return information about the locations of objects, but

rather arrays of pixel values, which are predicates concerning the light falling on the sensor. We *infer* that the camera "sees" something. This inference is yet another predicate, moreover, it may be a far more useful predicate than the raw sensor predicates with regards to a task such as picking up an object. However, the inference process can inject more uncertainty into the resulting predicates (although several predicates supporting a single inferred predicate may reinforce the inferred predicate and each other, i.e., increase their certainties). There are formalisms for propagation of certainty/uncertainty along chains of inference; e.g., see [55].

In a dynamic world, certainty generally decreases with time, since things (including the agent) are moving and changing in unpredictable ways. An object may no longer be where we sensed it a minute ago, or even where we think it might be after accounting for an estimate of the rate and direction of change. Therefore, certainty generally decreases with *time*, *space*, and *depth of inference*. Certainty generally *increases* with *breadth* of inference; by breadth of inference I mean that evidence from diverse sources supporting a single predicate can increase certainty.

A series of actions (a *plan*) executed based on the truth of the entire set of (direct sensor and inferred) predicates may enable an agent to accomplish a given task with some certainty. The actions in the plan have preconditions, and produce results (more predicates). The certainty that these results will be obtained by the action depend on both the action itself, and the degree of certainty that the action's preconditions are true. The action description itself may be inaccurate to some extent, or may only be capable probabilistically of producing the intended effect. An action therefore has *result certainties* associated with it, which encode how likely it is that the action will produce its results *if* the action's preconditions are true. Also, the action's preconditions may not, in fact, be true in the environment, since these preconditions were sensed by real (imperfect) sensors. Therefore, chains of actions (plans) lead to decreasing certainty, since the uncertainty of the individual actions in the plan is cumulative. Having additional plans to achieve a single goal can converge to increase certainty. Ideally, the certainty of the goal being achieved by the plan can be computed from the certainties of the sensor and inferred predicates that the plan is based upon, *and* the certainty associated with the actions in the plan itself.

To summarize: a given set of sensors produces a set of predicates with associated certainty values. Inferences based on these *sensor predicates* produce *inferred predicates*, again with associated certainties. Based on the information in the sensor and inferred predicates, we can select or construct a plan to produce some goal result in the world.

A plan is *effectual* if it will produce a goal result with a given level of certainty. Note that this model is a superset of classical planning, except that in classical planning all

the certainties are maximal. In the real world, we must allow that the plan may not achieve success with complete certainty, otherwise there would be no such thing as an effectual plan. Note further that sensors can re-sense a predicate, which has the effect of building certainty, i.e., a predicate that was very uncertain suddenly becomes quite certain if it is verified by a reliable sensor.

Sensor data is *useful* if it can be used to construct or select an effectual plan. The usefulness of sensor data must therefore be determined in the context of a goal and a set of actions and inferences, e.g., see [28, 65]. The implication of this is that the *effective* field of view of a sensor is not completely intrinsic to the sensor, rather, the effective field of view must be defined in terms of the *usefulness* of the data it provides, which is in turn defined in terms of the agent's goals and abilities. I therefore define the *effective field of view* of a sensor to be that region of space and time in which the sensor data is useful. Note that given this definition, occlusion is a special case of a limited field of view—occluded regions are regions of space about which we cannot draw any useful inferences. Occluded regions also have the additional complication that they cannot be reasoned about *a priori*. This is in contrast to a situation in which, for example, a lens has a 30 degree wide field of view, which can be stated without regard to the current environment.

Note also that the definition of usefulness requires the agent to know the information for it to be useful—information that the agent is not aware of cannot be used to construct a plan. However, it is helpful for us as system designers to consider the information that *would* be useful if the agent knew it; I will refer to this as *theoretically useful* information. Once the agent acquires the information, it may become *practically useful* (or just plain *useful*). Furthermore, once the agent knows some information, it does not become directly useful until the information is actually used in a plan. Information that the agent knows, but is not yet used in a plan is only *potentially useful*. Information that the agent has incorporated into a plan is *directly useful*.

It is primarily the function of the agent's perceptual system to convert theoretically useful information into potentially useful information, and it is primarily the function of the agent's planning system to convert potentially useful information into directly useful information; although, as we will see, this is not a strict division of labor. In the discussion to follow, unless specified otherwise, I will use the word *useful* to describe practically useful information, i.e., the union of the directly and potentially useful information.

3.1.2.1. Spatial and temporal fields of view

The absolute field of view of a sensor is a purely spatial object; it has no temporal dimension, since one can determine the current absolute field of view based purely on where the sensor is pointing *now*. The *effective* field of view however, has a temporal

extent. This is because the effective field of view is defined not on where the sensor is pointing now, but rather on the usefulness of the predicates extracted from the sensor data. When we say a predicate has *temporal extent*, we mean that the predicate remains useful or potentially useful for some interval of time beyond the point in time at which the predicate was extracted. Note that for the agent to be able to utilize the temporal extent of a predicate, the agent must represent the predicate in some persistent storage.

The idea of a temporal extent includes the idea of usefulness. A predicate extracted some time ago (milliseconds, seconds, minutes, hours, or more) may still be useful, even if the absolute field of view does not currently contain the relevant object. This is because although the world is dynamic, the degree of change is limited. Most things that were true a second (minute, hour, etc.) ago are still true now. Therefore, remembering the predicates extracted by a sensor expands the effective field of view of the sensor.

Consider an analogy in which a flashlight represents a sensor, and the cone of light the flashlight gives off represents the sensor's field of view. In a dark room, we can turn on the flashlight, and immediately see the absolute field of view of the flashlight, since it corresponds exactly to those things that are "lit up" right now. Those things that remain "in the dark" are outside of the absolute field of view. As the flashlight is moved around the room, the absolute field of view changes as the light falls on different locations. To determine the absolute field of view, we simply observe what is lit up at any given moment.

If we remember previous views, however, the *effective* field of view depends on both where the light falls now, *and* where the light has fallen recently. Imagine that after the flashlight has been in a given region, that region continues to "glow," even after the flashlight has moved somewhere else. The glow corresponds to our memory for recently seen items. The effective field of view is the union of the locations that are currently lit by the flashlight and those locations that continue to glow. Furthermore, we can encode brightness as certainty, such that the current absolute field of view is brighter (more certain) than the glowing regions outside of the absolute field of view. As the flashlight is moved around the room, it leaves a glowing trail that fades with time. The agent can then use the information inside the glowing trail, as well as that in the current absolute field of view, in order to make decisions about what to do next. In this way, the use of memory expands the effective field of view.

Now I need to complicate this flashlight analogy somewhat, because in the simple version given above, brightness was encoded as certainty, whereas the effective field of view is defined in terms of the related but distinctly different concept of "usefulness." When brightness encodes certainty, the flashlight sweeps out a broad swath of glowing points. However, not all of these points can be used to draw useful inferences. Since use-

fulness is the more important concept, we should instead have brightness encode usefulness, in which case the flashlight leaves a trail of glowing isolated points.

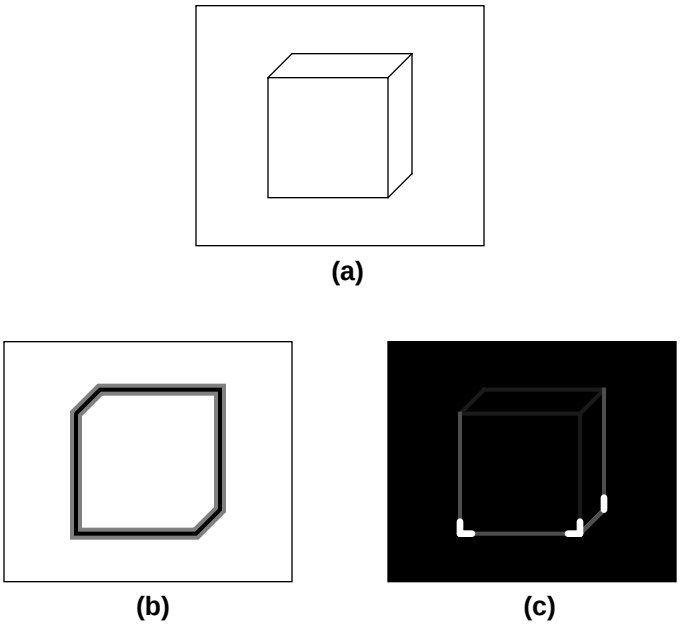


Figure 3-1: Images of a box (a) in which brightness encodes certainty (b) or usefulness to obstacle avoidance (c)

For example, consider the situation in which our goal is navigation with obstacle avoidance, and the flashlight (sensor) points at a box sitting on the floor in our path. Further assume that we can segment box pixels from non-box pixels. After the segmentation, we have a set of predicates, one for each pixel, concerning whether there is box at that pixel. If brightness encodes certainty as in Figure 3-1b, then the region in the interior of the image projection of box is quite bright, since we are very certain of those pixels being "box" pixels. The regions at the boundary between box and non-box pixels are dimmer, since we are less certain about the "boxness" of those pixels. However, in terms of usefulness, the border and corner points of the box are the most important for navigating around the box, so if brightness encodes usefulness, as in Figure 3-1c, the corners of the box are the brightest points. Instead of leaving broad swaths of glowing areas, the flashlight leaves a few glowing hot spots on important items, edges, and corners. Certainty contributes to the usefulness of these points, since if our estimate of the location of the corner of the box became sufficiently uncertain, the points would no longer be useful. However, usefulness is driven primarily by the task to be accomplished.

Having brightness encode usefulness instead of certainty affects the region *inside* the absolute field of view as well at the glowing trail it leaves behind. As any researcher in

computer vision knows, just because you are pointing the camera at something does not mean you can draw any useful inferences about the object, or can even determine what the object is. Our "effective field flashlight," therefore, does not light up everything in its absolute field of view, but rather only those things the agent can draw useful inferences about.

As a side note, given the current state of the art in perception technology, the absolute field of view of vision sensors is much larger than that of sonar, since cameras can see very far away with remarkable resolution—sonar by comparison is pathetic. However, the *effective* field of view of sonar is bigger than that of vision, since nobody yet knows how to extract much in the way of *useful* predicates from images, whereas sonar has been quite successful in navigation tasks—vision by comparison is pathetic. Sonar can extract useful predicates such a "there is an obstacle three feet to the left of me." Try *that* with a camera.

As I have noted, the usefulness of a percept depends on the end-to-end operation of the entire system, from the properties of the environment, to the sensor technology, to the sensor processing capabilities, to the actions the agent is capable of, to the planner that sequences these actions, to the ultimate goals of the agent that the planner seeks to achieve. All of these components must be considered when determining what the effective field flashlight illuminates. Improvements to any of these components of the autonomous agent ultimately expands the effective field of view. In fact, the entire field of autonomous agents can be defined in terms of expanding agents' effective field of view, since it embodies the idea that our goal, the goal of the research community, is to enable our agents to know more in order to do more things—to accomplish more goals in the world.

3.1.2.2. Planning and the effective field of view

Given the notion of the effective field of view, the subdisciplines of classical and reactive planning fit neatly at opposite ends of a spectrum with regards to their positions on the temporal extent of predicates. Classical planning assumes that the temporal extent of predicates is infinite—i.e., predicates that were true and useful a few minutes ago remain true and useful now, and by induction, remain true and useful until the agent changes them. This is evident in the "static world assumption" that is at the heart of classical planning, and determined the course of the field since its inception in the late 1950's, continuing well into the in the 90's (e.g., see [64]).

A classical planning system begins by assuming the existence of a complete description of the world. If one does not exist, go create one—we can wait. You can take as long as you want, since everything you find out during the first minute will still be true when you are done, since predicates are assumed to have infinite temporal extent. And we *will* have to wait a long time, since classical planning does not differentiate *useful* predi-

cates, so the perceptual system might generate *all* predicates regarding the environment—a very time consuming task indeed. It should be clear that not all predicates that are true are also useful; truth does not imply usefulness. Finding all true predicates (if you could) would potentially take far more work than finding the useful predicates.

For some predicates, the assumption of infinite temporal extent is not a bad one. These predicates usually take the form of "general laws," and can be programmed into the robot initially. The law of gravity is one example, and it implies that everything must be on the ground or be supported by something that is on the ground. This is not always true, but it is true often enough to be useful.

As a side note, consider that, even as we have said that truth does not imply usefulness, it is also the case that usefulness does not imply truth. There may be many predicates that are not strictly true, but which can nevertheless form the basis for successful plans. As example, consider that the idea of "naive physics" is based on the fact that predicates concerning physics which may be false may also be useful [29].

The strong form of reactive planning takes a diametrically opposed position from classical planning with respect to the temporal extent of predicates, in that it assumes that there is *no* temporal extent—if you are not sensing it *now*, the certainty of the predicate being true is not high enough for it to be useful. This position leads to the "no representation" dogma, since representation is memory, and memory is an attempt to expand the temporal extent of predicates. Of course, it is difficult to build an agent that does anything useful which believes only those things that it is currently sensing, and as pointed out by Tsotsos [61], having no representation at all is, well, silly. However, the reactive planning community has constructed several working robots with impressive (given the current state of the art) capabilities. Thus, a question arises: how can we reconcile the apparent success of the agents built ostensibly under the patently inappropriate "no representation" doctrine. The answer of course, is that the doctrine was not strictly followed. However, given the temporal extent concept, we can describe the actual methodology, and why is was relatively successful.

Instead of having a continuum of temporal extent, one can divide predicates into two broad classes: those with infinite temporal extent, and those with little or no temporal extent. A reactive system "hard-wires" a set of sensor inputs to an action. This hard-wiring embodies knowledge concerning predicates of "infinite" temporal extent. For example, a robot was built to find and pick up soda cans, and then deposit them in the trash [18]. When the robot sensed an object of approximately the right size and shape in an appropriate location (using a laser range-finger), a "pick-up" action was triggered. Knowledge about the color, size, and shape of soda cans is a set of predicates of infinite temporal

extent: “facts” about the world. These infinite extent predicates are not explicitly represented in the agent, however, they are implicit in the agent’s reactive rules, e.g., “if you see a red blotch in the image of such-and-such a size and shape, move towards it.” The fact that this red blotch is assumed to be a soda can is an infinite extent predicate: red blotches of this size and shape are soda cans. This predicate can be “wired-in” and “hidden” precisely because it is of infinite extent. Since it is always assumed to be true, there is no need to represent it explicitly, but implicit “facts” such as these permeate the design of the agent. Moreover, these predicates *are* explicitly represented by the *designer* of the agent. Explicitly representing and reasoning about these predicates is what enabled the system designer to construct the robot. These sorts of predicates do not “count” as representation, since the agent does not represent them explicitly.

Other predicates in a reactive system, such as “I’m holding something” have no temporal extent. This requires the robot to have a sensor that signals at all times whether there is something in the gripper. Given the unreliability of the grippers, this is a very good idea, since it means that if the robot drops the object, there is an immediate signal of the change, which triggers a different set of actions (possibly to pick up the object again). If we were to give the “I’m holding something” predicate a longer temporal extent, the robot runs the risk of acting on false information, e.g., continuing to proceed to the trash can, even though the robot accidentally dropped the trash a while back. This kind of predicate really does not have much temporal extent. One reason the reactive robots are so successful is that they are unlikely to take inappropriate actions based on stale information.

Another reason reactive systems are successful relative to the traditional approach is that the current sensor technology is so bad that sensor predicates, and more importantly, inferences made from the sensor data, have very low associated certainties. By eliminating the temporal extent of these predicates, the agent is again likely to act on only very certain information. One of the major faults of non-reactive systems is a tendency to take actions based on inferences from the sensor data that are really quite tenuous. In practice, reactive systems eliminate this possibility basically by eliminating (or hard wiring) inference. Eliminating inference and inferred predicates leaves only the sensor predicates, which really do not have any temporal extent, so the division of predicates into those with infinite temporal extent and those with zero temporal extent is a pretty good characterization of the remaining predicates. The zero temporal-extent predicates are those determined by the robot’s rather poor sensors, whereas the infinite temporal-extent predicates are those determined by the human designer of the system, who brings a great deal of expertise to bear on the design of those predicates.

3.1.2.3. Markers and the effective field of view

A marker is a predicate with temporal extent concerning the location of an object in space. As we will see, markers are important because they efficiently expand an agent's effective field of view.

Recall our "effective field" flashlight: as it surveys a room, useful predicates "light up" and stay lit up, as if by magic, as the flashlight moves around the room. However, use of magic is not an option; we must extract and maintain these predicates. They only light up if we extract them, and they only stay lit up if we maintain them. The effective field of view only contains those predicates that are lit up by our efforts.

One overly simplistic way to maintain the information is to save all the raw sensor readings ever taken. However, saving these readings is highly inefficient; consider the case if the sensor is a camera. We would save every image the camera ever produced, which is an extremely memory intensive proposition. Moreover, the actual usefulness of these predicates (i.e., the pixel values), is rather limited; it is the information extracted from these pixels that is useful. We might imagine that the pixels themselves have only a faint "usefulness glow," whereas the information extracted from the pixels (say, corners) glows brighter. The extraction process concentrates the useful information; a marker is a storage mechanism that contains and maintains that concentrated information.

It is worth noting here that sometimes it may actually be useful to retain the raw image instead of just the percepts derived from it. The drawing of inferences is ultimately a form of data compression, and moreover, lossy data compression. There are an infinite number of inferences that we can draw from the sensor data, and we can only save some of them. Drawing only a few inferences and throwing away the original sensor data admits the possibility of throwing away potentially useful information. Saving the original sensor data is a "least-commitment" strategy, but it comes at the price of a high memory overhead. It is only useful if the agent is likely to gain additional information later that will determine what computations should be done to draw inferences from the sensor data.

This discussion of information that "might be useful later" hints at the subtle and complex structure of the concept of "usefulness." The overall usefulness of information contained in a predicate depends not only on the usefulness of the information in executing a currently active plan to achieve a currently active goal, but also upon the *future* usefulness of the information in plans and goals that are not currently active. Such information is, as defined previously, *potentially* useful. Since we are interested in agents that operate with incomplete information, it may be the case that information not known by the agent *now* (i.e., theoretically useful information) will become known *later* and change the value of other information. For example, consider an agent that has an action

which it may use to directly accomplish some goal, and that action has a precondition consisting of the conjunct of propositions A and B. The agent may have reliable information that A is true, yet it may not know about the truth of B. The possible falseness of proposition B limits the usefulness of the information that A is true, since A can only be used if B is also true. However, if B is later found to be true, then the information that A is true becomes much more valuable.

This sort of situation is one in which the previous generalizations about the roles of the perception and planning systems break down. Recall that I had stated that it is primarily the function of the agent's perceptual system to convert theoretically useful information into potentially useful information, and it is primarily the role of the agent's planning system to convert potentially useful information into useful information. However, at the point in time at which the agent knows predicate A above, but not predicate B, the agent may construct a plan to obtain information regarding B, thereby involving the planning system in converting a theoretically useful predicate into a potentially or practically useful predicate.

Given the discussion above, we can see that the overall usefulness of any proposition is dependent on not only its current value, but also on its potential future value. If we are considering discarding information that the agent may have gathered previously (and I contend that we must) then the decision on what information to discard must be based on the information's current and future potential values. Additional research is therefore needed to further elaborate the structure of the concept of usefulness.

There are any number of means of accomplishing the goal of expanding the effective field of view; markers are just one (particularly good) way. Another way to expand the effective field of view is to expand the *absolute* field of view, by simply getting more sensors, or sensors with a larger intrinsic field of view. For example, one could get a wide angle lens for a camera, or get more cameras (or sonars, contact switches, laser range finders, etc.). More, bigger, better sensors come at a cost and the increase the effective field of view's spatial dimension only, meaning that the agent can know more about the world *now* by increasing the absolute field of view, but does not know more about parts of the world it has seen *in the past*. Given a fixed set of sensors and an associated absolute field of view, the use of memory can potentially increase the agent's effective field of view beyond the absolute field of view provided by the sensors.

The use of markers increases the effective field of view's temporal dimension, and for movable sensors, marker use increases the sensor's spatial dimension as well. An alternative to marker use that also increases the temporal dimension (and by extension, the spatial dimension), is to save raw sensor data instead of the markers, i.e., do without the

data compression. The pros and cons of this approach are discussed above. Like everything else in engineering, it is a trade-off.

Another important argument in favor of the use of markers is the ability to deal with occlusion, which as mentioned previously is a special case of a limited field of view. The problem of occlusion is intrinsic to the sensor modality; there are simply some things that a sensor such as a camera or sonar cannot see through, and no lens or amplifier can change that fact. One solution is to change modalities (use x-rays or something). However, by working through the temporal dimension, the use of markers or other memory mechanisms can enable the agent to "see through" obstacles by remembering what was seen when the agent was on the other side of the obstacle.

An interesting possible alternative is the use of multiple cameras on independent bases which can broadcast information to each other, similar to [67]. This "cooperative" strategy expands the notion of an "agent" to encompass cooperating agents that communicate their perceptions. An agent could "see through" an obstacle by having one of its "friends" that is on the other side of the obstacle radio back what it sees. This is also a variation on changing modalities.

3.1.3. Conclusion

The notion of an effective field of view encapsulates in a single concept the definition, motivation, purposes, and goals of active perception for autonomous agents. The effective field of view is defined not in terms of a physical sensor, but rather in terms of *useful* predicates, predicates which contain information about the world that can help the agent accomplish its tasks. The effective field of view consists of those predicates that are useful in the context of the agent's goals and capabilities. The effective field of view is a focal point for research aimed at dealing comprehensively with the problem of incomplete and uncertain information that confronts autonomous agents operating in realistic environments.

All research in autonomous agents, from perception to planning to manipulation and action, can be cast in terms of how it relates to the fundamental operation of expanding the effective field of view. The use of markers is an efficient means of expanding the effective field of view, which works by using a memory and data compression to efficiently maintain highly useful information.

3.2. Task-oriented design

Given the centrality of the effective field of view to the design of agents that operate in real environments, and given further the centrality of the task to be accomplished to

the effective field of view, the first step in constructing any agent is to analyze the task or tasks the agent is to perform. The specification of an agent is therefore task-oriented, i.e., the central organizing principle is to support a set of tasks to be accomplished by the agent. Some amount of hardware and software mechanisms are needed to accomplish each task, with several component mechanisms required to accomplish a complex task. A supervisory apparatus is needed to coordinate the individual component mechanisms; we refer to this supervisor, together with the components it uses, as the task's *agency*, after Minsky [47].

One might like to think of task-agencies as subroutines, however, there are important differences between subroutines and task-agencies. Unlike subroutines, task-agencies are *active* and *opportunistic*. By "active," we mean that they can "look for" their markers/parameters, e.g., by pointing directional sensors appropriately, by launching the appropriate visual routines in the intermediate vision system, or by stimulating other agencies to perform actions likely to result in the appropriate world objects being marked. By "opportunistic," we mean that if a task-agency has its inputs available, whether or not they became available due to the direct actions of the task-agency, it can use those inputs to accomplish its task. A complete autonomous agent may have several tasks to accomplish, each with its respective agency for accomplishing it. Any number of these task-agencies may be active in parallel at any given time.

Any sufficiently complex agent will have multiple goals—some of these goals will be in conflict, while others will support one another. There is a vast variety of tasks one might wish to accomplish, and a task-oriented design dictates that agents with different tasks have different designs. However, all such agent designs have the common aspect that agent design must address the issue of task interactions. This section discusses general principles for dealing with several goals, and illustrates these principles via an example that will run throughout this dissertation. The example agent is a small herbivore that collects and eats food in the presence of obstacles and predators which it must avoid. It therefore has three tasks to accomplish: find food, avoid obstacles, and avoid predators. In later chapters, a simulation of this example is developed and analyzed.

The interactions among tasks come in two major varieties: conflicting and mutually supporting. The agent must have some mechanism for mediating among conflicting tasks; this issue is addressed in the next section. Mutually supporting tasks can be organized into a hierarchy reminiscent of the "hierarchy of abstraction spaces" in classical planning [53]. These task hierarchies are addressed later in this chapter.

3.2.1. Mediation among conflicting goals

An agent may perform several tasks, and is therefore made up of an interacting collection of agencies. Organizing the design via agencies assures that all the elements needed for a task are active when necessary, and conversely, that elements not needed for the current task are not active, and therefore do not consume resources unnecessarily.

This analysis will first consider the tasks to be accomplished independently, at the level of Marr's "computational theory," in which the goals of the computation are determined, and the strategies for attaining those goals are thought out [42]. The "representation and algorithms" (Marr's level 2) are developed by taking into consideration interactions among the tasks. Marr's third level (hardware implementation) is addressed at implementation time.

3.2.1.1. Resource conflicts among tasks

Each agency may have several components, and furthermore, these component sets are not disjoint, i.e., some components are used in more than one agency. The tasks of these agencies are not independent; they must be carried out continuously and simultaneously, in spite of the fact that the goals of the tasks are occasionally in conflict. In this analysis, the conflicts are cast as *resource conflicts*. The resources in question are components that are shared amongst agencies. Conflict arises when two or more agencies simultaneously require the same component in order to perform their task, and the alternative actions the agencies require the component to perform are mutually exclusive. Identifying and handling these sources of conflict is critical to the successful construction of a situated agent.

The block-diagram of the agent given in Figure 3-2 identifies three major subsystems: Action, Perception, and Memory. Resource conflicts can arise in all three. One of the most important resources involved in the construction of a situated agent is the set of effectors. Most effectors can only be doing one thing at any given time, so if two agencies simultaneously require the same effector for different purposes, a conflict arises.

The agent constructed for this research program has three primary tasks: obtain food, avoid obstacles, and avoid predators. The agencies responsible for each of these tasks are referred to as the EAT, BUMP, and RUN agencies, respectively. The agent analyzed here only has two effectors, its body and its neck. The body accepts two commands, a forward speed and a turn speed, while the neck accepts one command, a head angle relative to the body. This implies that the agent has a distinguished "forward" direction, which is the direction of motion. An example of effector resource conflict in this domain arises between the EAT and RUN agencies when a predator is seen next to a food item. The EAT

agency wants to issue effector commands that direct the agent to move towards the food, whereas the RUN agency prefers to run away from the predator (and thereby run away from the food item). Unfortunately, the agent can only move in one direction at any given moment.

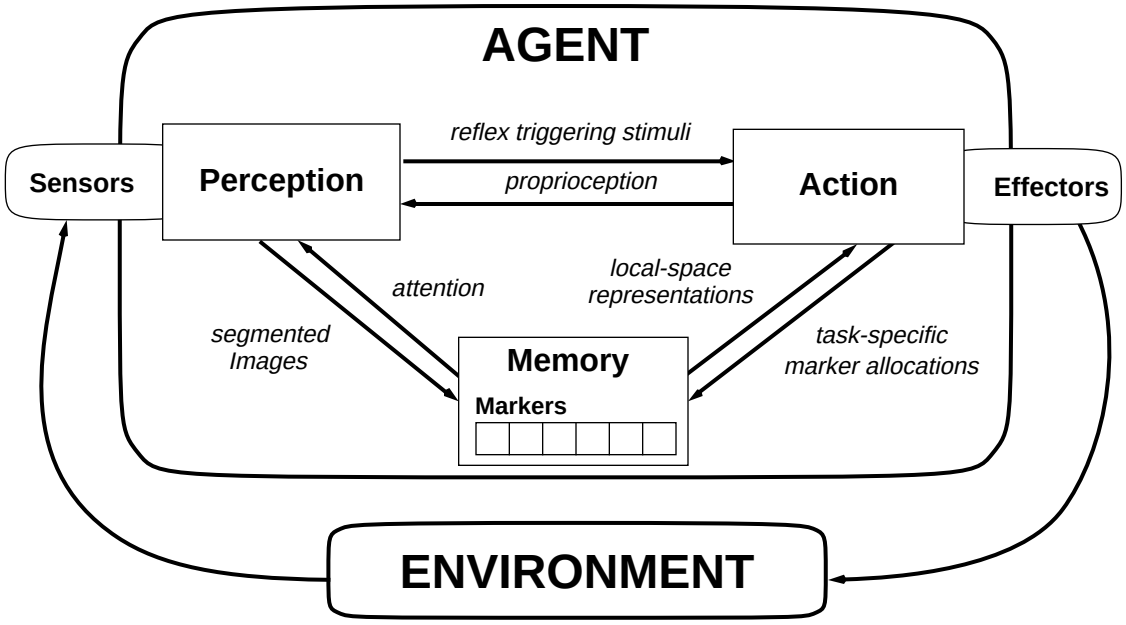


Figure 3-2: Autonomous Agent Architecture

Another important resource is the agent's set of sensors. In the case of the our agent, its only sensor is its "eye," as realized by the image stream generated by the simulation. While fleeing from a predator, a conflict might arise between the RUN and BUMP agencies over what commands to send the neck effector to orient the eye sensor. The RUN agency might prefer to keep looking back towards the predator, in order to keep track of the predator's position, and thereby ensure that it directs the agent to move directly away from it. Looking directly towards the predator keeps the predator in the agent's absolute field of view, which presumably keeps the predicate concerning the predator's location at a high degree of certainty, as opposed to remembering the last known location of the predator, which would be a predicate of lower certainty. On the other hand, the BUMP agency would much prefer to keep the eye pointing forward, so that it can see where it's going and avoid any obstacles. The predicate that the BUMP agency needs to establish with a reasonably high certainty is that "there is a clear path immediately in front" of the agent. This predicate may have a short temporal extent, especially if the agent is moving quickly in order to evade the predator. Keeping the eye pointed forward evaluates a "clear path" predicate with a high degree of certainty.

Another critical resource in the perceptual system is intermediate visual processing power. At the lowest-level, visual processing is bottom-up, with all early maps being computed in parallel regardless of the current task, but visual routines at the intermediate level are top-down and sequential [62]. This implies a limited amount of intermediate "visual processing cycles." Rather than compute all of the possible spatial relationships in the visual input, currently active agencies direct the visual routines processor to perform only those computations necessary for carrying out their tasks, i.e., extract the useful spatial relationship predicates. This conserves the visual processing resource, but also gives rise to resource conflicts when more than one agency is active concurrently. An example of this type of conflict arises between the EAT and RUN agencies when a predator is noticed. EAT wants the intermediate visual processor to continue to find food (extract predicates about food locations, which are useful to its task), whereas RUN wants to find a place to hide immediately (extract predicates useful to *its* task), without wasting time looking for food. Which agency controls the intermediate visual processor is dependent on the relative "strengths" of the agencies (see section 3.2.1.4), which in this case is dependent on the proximity of the predator.

Finally, a central point in this discussion is that there are resource conflicts in the short-term memory subsystem, since the extracted predicates must be stored and maintained in order to remain useful. As memory mechanisms, markers record useful predicates concerning the location (and possibly some visual attributes) of items important to a current task [1, 2]. There is overhead involved in maintaining markers. In a dynamic world the locations (and visual attributes) of important items may change; thus there is potential for the recorded information to no longer properly represent the current world state. Maintaining markers is not simply a matter of updating an internal representation, but also requires overt sensing activities in order to re-verify the information at appropriate intervals, in order to synchronize the world state represented by the markers to the actual world state. Clearly, acting on an incorrect world model is potentially hazardous, and as discussed in section 3.1.1.2, is really a form of the frame problem.

To minimize the maintenance overhead, the number of markers must be kept low. The agencies are in conflict over how to use the limited set of markers. The EAT agency would like to mark food, RUN wants to mark predators, and BUMP wants to mark obstacles. There are far more objects that might be marked than could be established and maintained using limited computational and sensing resources, so a choice must be made.

3.2.1.2. Independent task analysis (level 1)

Despite the acknowledged interactions between tasks, it will simplify the analysis to initially consider each task independently. This initial analysis will be confined to the

"computational theory" level of information processing, since the next level (representation and algorithms) must consider interactions among tasks. Each agency has components in each of the major subsystems of Perception, Memory, and Action. In some cases, the same component is shared among agencies, but the conflicts this implies are addressed at a subsequent level of analysis, or even at implementation time.

3.2.1.2.1. The EAT Agency. In our sample domain, the EAT agency must identify food and direct the agent to move toward it. We do not attempt to simulate the manipulation involved in a realistic eating behavior; rather, upon coming in contact with a food item, the agent automatically "eats" it. The agent should accomplish the eating task as efficiently as possible, i.e., the agent should travel as small a distance as possible between food items.

The EAT agency must have a component which identifies food in the incoming image stream. Once identified, the food's location relative to the creature must be determined; this location might be noted by placing a marker on it. Based on the current knowledge of food locations (both from the current perception and from memory of items seen previously, but now out of the absolute field of view), the agent must choose a target food item and send appropriate effector commands to direct the agent to move toward the target. If no food is currently visible or remembered (via a marker), a visual search strategy must be used, which could be a simple random walk, or a sophisticated search routine.

3.2.1.2.2. The RUN Agency. The RUN agency must perceive the existence of a predator, and determine the magnitude of the potential threat (based primarily on the perceived distance to the predator). If an escape behavior is deemed necessary, the appropriate commands should be sent to the effectors. A more subtle task of the RUN agency is to be wary of potential hiding places for predators. If the RUN agency only becomes active after the predator is seen, it may be too late to perform an effective escape maneuver.

3.2.1.2.3. The BUMP Agency. The task of this agency is to prevent the agent from running into obstacles. The simplest way to accomplish this is to prevent the agent from moving at all. However, this strategy will not allow the other agencies to perform their functions. The BUMP agency is interesting because its preferred action is to do nothing. Thus, most of its activity is necessitated to the extent that it interacts with the other agencies, which is the topic of the next section.

3.2.1.3. Task interactions (level 2)

Identifying and resolving conflicts among the agencies is a critical aspect of the design of an agent with multiple goals. This section discusses general principles for conflict resolution between agencies, along with specific solutions for the interaction between the agencies in our sample agent. We argue that satisfactory resolution of some conflicts

requires the agencies in question to communicate via a shared representation. The proposed representation for this purpose is marker-based, and examples of marker-based communication among the agencies are given in the context of our sample domain.

The first point to note when addressing the issue of conflict among the agencies is that it is not necessarily a purely adversarial relationship between the agencies. Compromise and cooperation between the agencies is possible, and the best solution may require such cooperation. To show the form that this cooperation may take, we will consider the interaction between the EAT and BUMP agencies. First, an adversarial (non-cooperative) strategy is developed and analyzed, followed by the development of a cooperative strategy.

3.2.1.4. Adversarial resource conflict resolution

As discussed above, if the BUMP agency were working in isolation, a simple strategy would be to not allow the agent to move at all. The problem with this is that it unnecessarily prevents the other agencies from pursuing their goals. This strategy is clearly not cooperative. Another possible strategy is to cancel or redirect any effector command that might result in a collision. This subsumption [11] style strategy allows the other agencies to pursue their goals, except when near an obstacle, when the BUMP agency takes precedence. However, this strategy is not cooperative either, since the winner-take-all control precedence still casts the agencies as adversaries.

In general, the adversarial method of resource conflict resolution proceeds as follows. At any given time, each of the agencies is assigned a numerically valued “strength” based on the current situation, where the current situation is determined based on the current sensor input and state of memory. When a resource conflict arises, the agency with the highest strength obtains the resource, and uses it as if it were working in isolation. In the first version of the BUMP agency (that never allowed the agent to move), the strength of the BUMP agency was always higher than that of the other agencies. In the second version, the BUMP agency’s strength only rises above that of the other agencies when the agent is sufficiently close to an obstacle. In cases where the alternative actions desired by different agencies are truly mutually exclusive, the adversarial form of conflict resolution is the only option.

However, adversarial resource conflict sometimes fails to solve the problem. As an example, consider the action taken by the BUMP agency when it wins control of the effector resource from the EAT agency. Presumably, this occurs when the EAT agency directs the agent towards a food item, with the selected path containing an obstacle. (It may seem inappropriate to suggest that the agency would select a path containing an obstacle, but for an agent with finite width, it is possible for a “clear” line-of-sight path to contain an effec-

tive obstacle. For example, the agent may see a food item through a gap between a rock and a tree, but the gap is too narrow to allow passage.) When the agent comes sufficiently close to the obstacle, the BUMP agency takes over, and determines a course of action.

One course the BUMP agency can take is to cancel the move command sent by the EAT agency, or alternatively command the agent to stop. If this is done, at the next time step the situation is the same, in that the path to the food will still intersect an obstacle, so the EAT agency's attempt to move towards the food is again canceled by BUMP. This can continue ad infinitum, and no further progress is made. The BUMP agency functionally locks out the other agencies.

In pure adversarial resource conflict, one of several agencies competing for the resource is the strongest, and the resource is used by the winner as if it were working in isolation. In the case of effector resources, this means that each agency has an action it wants to take, and exactly one will be chosen, based on the agencies' strengths. But in the case of the EAT-BUMP conflict, the correct action to take was not that desired by the EAT agency (go forward), nor that desired by the BUMP agency (stop). Nor is the correct action a weighted combination of the two (go forward slowly?).

Reactive systems built using a subsumption approach usually deal with this problem by executing a pre-planned (with possibly some random elements) behavior designed to change the situation sufficiently that progress is possible, e.g., [11, 18]. For example, in a subsumption based system, the BUMP agency, instead of instructing the agent to stop, might direct the agent to back up slightly, turn a little to the left, and then move forward a little. Upon completion of the behavior, the normal rules again apply, and the EAT agency directs the agent to move towards the food. The assumption here is that from the new location of the agent, a different path to the food will be selected. However, this may still result in a collision, in which case BUMP will take over again, and execute its semi-random behavior. Control alternates between EAT and BUMP until eventually, the "back-and-turn-left-then-go-forward" action will cause the agent to circumvent the obstacle on the left.

The problem with this approach is that it is often difficult to find a single behavior that is appropriate in all situations. It may be the case that the only effective route around the obstacle is to the right, rather than the left. Modifying the BUMP action from "stop" to "back-and-turn-left-then-go-forward" is a special purpose hack, which is exposed as a hack when a better action is to turn right, not left. To determine a more efficient action, the BUMP agency needs to have knowledge of the EAT agency's goals. In general, the preferred behavior is not uniquely determined by the current perceptual input, but is also dependent on the goals of the other agencies.

3.2.1.5. Cooperative resource conflict resolution

There is often an option that allows cooperation between the agencies and is superior to the purely adversarial approach. In this case, the action to be taken is dependent on the goals of other agencies. If an agency must know the goals of another agency in order to determine a good course of action, those goals must be communicated amongst the agencies in some way. One way an agency might learn more about the goals of another agency is to intercept the other agency's outputs, and modify its own outputs based on them. For example, the BUMP agency might intercept effector commands sent by the EAT agency. If the EAT agency wanted to turn right, then we might assume the best strategy for BUMP is to circumvent the obstacle on the right. But it is easy to imagine situations where this assumption fails. Moreover, it may not be possible to determine the actual goal of the EAT agency based purely on the effector commands it sends.

As a solution to this cooperation problem, I propose the use of markers to communicate among the agencies. By communicating the what and where aspects of their intentions via a set of markers, agencies can cooperate to share a limited pool of resources, and determine an effective action to take. This use of markers can be likened to the "marker as register" analogy given in section 3.1.1.2. Processes and subroutines often communicate by placing information in a register to which both entities have access. To see how this works, we can re-examine the EAT-BUMP effector resource conflict.

The behavior of the EAT agency when acting in isolation is to notice nearby food items, and place markers on them. Then it chooses the nearest marked food item as its target, and sends effector commands to direct the agent to move towards the target item. The EAT agency communicates its intentions to other agencies by tagging the appropriate marker as its goal location. If the agent should then move too near an obstacle, the BUMP agency can use the extra information in the goal marker (established by the EAT agency) to determine an efficient course of action. The BUMP agency can circumvent the obstacle in the direction that is judged to be more efficient in the context of the goal of the EAT agency.

In fact, with the agencies communicating via the markers, we can streamline the process even further. As soon as the EAT agency tags a given location as its goal destination, the BUMP agency can look for potential obstacles in the path towards that destination. If such an obstacle is found, BUMP can place a marker on the object, and label it as an obstacle associated with EAT's particular destination. By only marking the obstacles associated with a destination, the number of things that the BUMP agency needs to mark is kept small. It is then possible for the EAT agency to notice when there is an obstacle associated with its current destination, i.e., the new markers established by the BUMP

agency, and modify its heading so that the agent never passes close enough to the obstacle to trigger any other special actions by the BUMP agency. To paraphrase and summarize this communication, the exchange takes place as follows:

> EAT:“I want to go *there*.”
> BUMP:“You can’t go straight there because of *that* obstacle.”
> EAT:“O.K., I’ll go around *that* on my way *there*.”

Each of the italicized deictic words (*there*, *that*) above refers to a marker. *There* was placed by EAT, and *that* was placed by BUMP. Here we have a clear example of cooperative conflict resolution. If we only look at the problem at the lowest level, there is a major conflict; the EAT agency wants to move towards the food, whereas the BUMP agency want to prohibit that action due to an imminent collision. But by having the agencies communicate via the shared representation, the conflict is effectively avoided altogether.

3.2.2. Task hierarchy

The preceding discussion regarding task agencies emphasizes the independence of the agencies, and the resolution of conflicts that may arise when otherwise independent goals conflict. In that context, it was appropriate to consider the operation of the obstacle avoidance process as communication between independent entities. However, if we step back and consider the operation of the agent as a whole, what was described as “cooperative resource conflict resolution” can instead be viewed as a distributed algorithm for plan generation. The goal to be achieved was to consume the berry, and communication between EAT and BUMP resulted in the insertion of a step in the plan to circumvent an obstacle in pursuit of the goal.

In general, several task agencies can cooperate to achieve a goal. Moreover, these cooperating agencies can be organized into a hierarchy, in which the agency responsible for achieving a primary goal is at the top of the hierarchy, and subsidiary tasks are beneath the primary task. In the example given above, the primary task was berry consumption, with the subsidiary task being obstacle avoidance. Furthermore, the obstacle avoidance task itself had the subsidiary tasks of finding the obstacle and finding a route around the obstacle. At the lowest level of the hierarchy are the actual actions that must be taken in order to achieve the goal. Traversing along the leaves of this hierarchy, one can find the individual steps in a plan to achieve the given goal.

3.3. Markers and task-agencies

The ties that bind the steps in a plan together are predicates regarding the state of the world. Predicates form the preconditions for each action, and the results each action produces. For agents interacting with the real world, these predicates can be effectively embodied by markers. In such agents, it is therefore the state of the markers that determines what actions are appropriate at any given time, what agencies are active, and how these agencies interact. The remainder of this chapter is a discussion of the roles that markers play in relation to the task agencies. An abridged version of this discussion appears in [9]. Note that while it may not always be explicitly mentioned below, the concepts of usefulness and the effective field of view are embodied in the idea of a marker, since, as discussed above, a marker is a useful predicate (with temporal extent) regarding the location of an object, used to expand an agent's effective field of view.

3.3.1. Categorization of markers

Although all markers have the "what" and "where" properties given in our definition at the beginning of this chapter, markers can be logically divided into several categories based on their general role in a task. Recall that the "what" aspect of markers is defined in terms of a task. Markers may therefore have different properties, depending on the task and how the marker relates to that task.

3.3.1.1. Image markers versus egocentric 3D markers

One distinction among markers can be made based on the coordinate system in which they are located. In previous work with markers this was not an issue, since there was only a single coordinate system: image coordinates [1, 16]. However, in a realistic visual environment, the natural coordinate system for visual perception (image coordinates) is not the natural coordinate system for acting in the environment, instead an egocentric 3D coordinate system is more effective. In a 2D video game environment, the perception and action coordinates can be confounded, whereas in a 3D environment, they cannot. This gives rise to the two coordinate systems, and hence, two types of markers. Actually, there are an infinite number of coordinate systems, e.g., object-centered coordinate systems are useful for many tasks, and external 3D coordinate systems are useful for tasks such as reading maps, but I will not consider these coordinate systems in the following discussion.

Image coordinate markers are placed on objects in the current image, and are used to facilitate visual routines. They mark objects such as edges, lines, terminations, etc. Image-coordinate markers which are instantiated based primarily on the incoming data

stream make up a set of items similar to Marr's raw primal sketch [42]. Image-coordinate markers which require a "top-down" processing component for their instantiation correspond to the markers in Ullman's visual routines [62]. These two types of markers are analogous to the "activating" and "active-only" markers described below.

Egocentric 3D markers are the focus of this dissertation; all markers discussed in the remainder of this dissertation will be in egocentric 3D coordinates. When the visual system has processed the input sufficiently to determine what and where a task-critical object is in the environment, a marker may be instantiated. The natural coordinate system for acting in the world is egocentric, and furthermore, is centered on the effector taking the action. This idea is well known in the active vision community [6]. For example, if an effector is a manipulator grasping an object, the correct direction to move the effector is given by the signs of the coordinates of the object in the *manipulator* coordinate system. In the sample agent constructed for this research program, markers are used for navigation by a robot that accepts velocity commands in the form of a forward speed and a turn speed. Clearly, the natural coordinate system for such markers is polar, centered on the robot, with the zero angle directly ahead. When this is the case, the direction to turn to get to a destination at coordinate (r, θ) is given by the sign of θ, the amount to turn is given by the magnitude of θ, and the distance to move is given by r.

3.3.1.2. Activating versus active-only

Another distinction that can be made among markers is based on how they are instantiated by the task-agency. Task-agencies are not always active, e.g., the "find food" agency is not active if the agent is not hungry. However, some agencies may have components in the perceptual system that perform computations continuously, even when the agency is not active. For example, there is some computation performed to determine whether the agent is hungry. If so, the "find food" agency is activated.

The result of this "continuous" computation may be a marker, more specifically, an *activating* marker, since the instantiation of the marker activates one or more agencies. Once an agency is active, it may require other inputs in order to carry out its task. The agency directs the perceptual system to carry out activities that will instantiate markers, this time *active-only* markers, which are markers which are instantiated by an *active* task-agency, that serve as the agency's other inputs.

As an example, recall the sample domain of a small creature that must avoid predators. This creature has a RUN agency to coordinate the predator avoidance task (see Figure 3-3). Whether this agency is active or not, a "predator-detector" scans for predators over the entire field of view. If one is found, an *activating* marker is instantiated and placed on the predator's location. The instantiation of the predator marker activates the

RUN agency, which directs the creature to start looking for a place to hide. This might be a rock or a tree, but the object's role in the current task is as a "hiding-place," so when a suitable object is found, an *active-only* marker is instantiated to serve as the "hiding-place" input to the RUN agency.

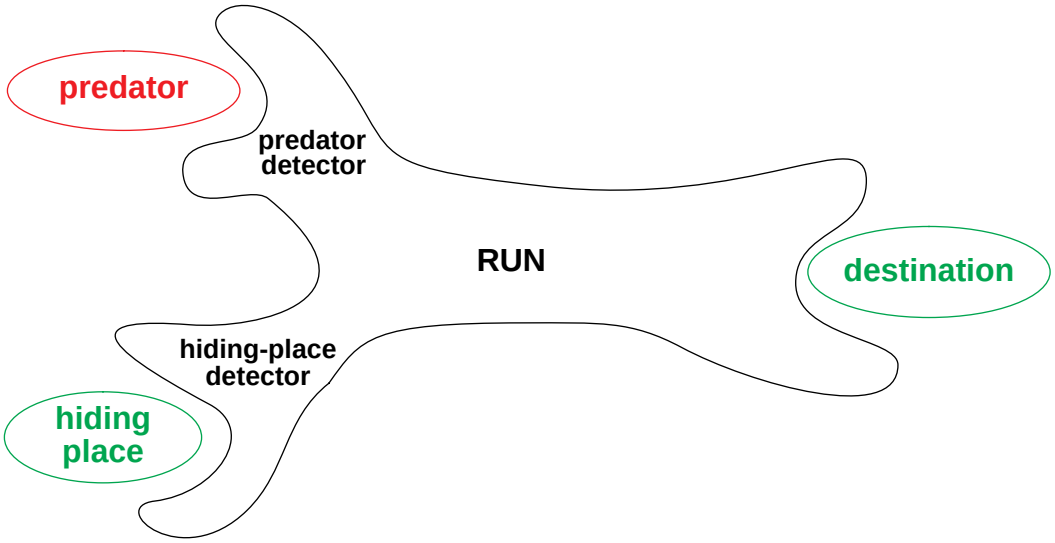


Figure 3-3: RUN and its markers

The predator detector and hiding-place detector are sub-tasks of the predator avoidance task. Note that the hiding-place marker is put on a rock or tree, objects that would not normally be marked if it were not for the active task-agency. Put another way, exactly the same visual stimulus may or may not be marked, depending on the internal state of the agent, i.e., the activation of its task-agencies. Finally, observe that whether a marker is "activating" depends on the perspective of the agency using it. For example, the destination marker above is an active-only marker from the perspective of the RUN agency, since it is only created when RUN is activated. However, the instantiation of the destination marker activates the navigation agency, so this same marker is an activating marker from the navigation agency's perspective. Finally, note that both types of marker have a direct relationship to a task-agency. Markers are instantiated only if the predicates they embody are potentially useful to some task-agency.

3.3.1.3. Primary goal markers versus dependent markers

As has been mentioned previously, at the most abstract level we can impose a hierarchy among tasks, i.e., there are primary tasks and secondary tasks. Secondary tasks can have further subsidiary tasks, etc. For example, we may have the primary task of "staying alive." Subsidiary to that is the task of "getting food," and subsidiary to that is the task of

"getting money" to buy food, and then "getting a job" to get money. Getting and keeping a job may have any number of sub-tasks.

Once tasks are accomplished, there is no need to continue with their sub-tasks. For example, getting money is no longer an issue if we win the lottery, and neither is having a job (although "keeping a job" may subserve goals other than just getting money, and may therefore still be an active task).

The task hierarchy imposes a similar hierarchy on the markers associated with particular tasks. As we have discussed, primary tasks have markers associated with them, e.g., a rabbit's goals are to find food and avoid predators, so berries and predators are marked. The BUMP task-agency is activated by instantiating an appropriate *destination* marker, e.g., by the berry-detector of the EAT agency. However, it may not be possible to proceed directly to the primary destination, due the presence of an obstacle.

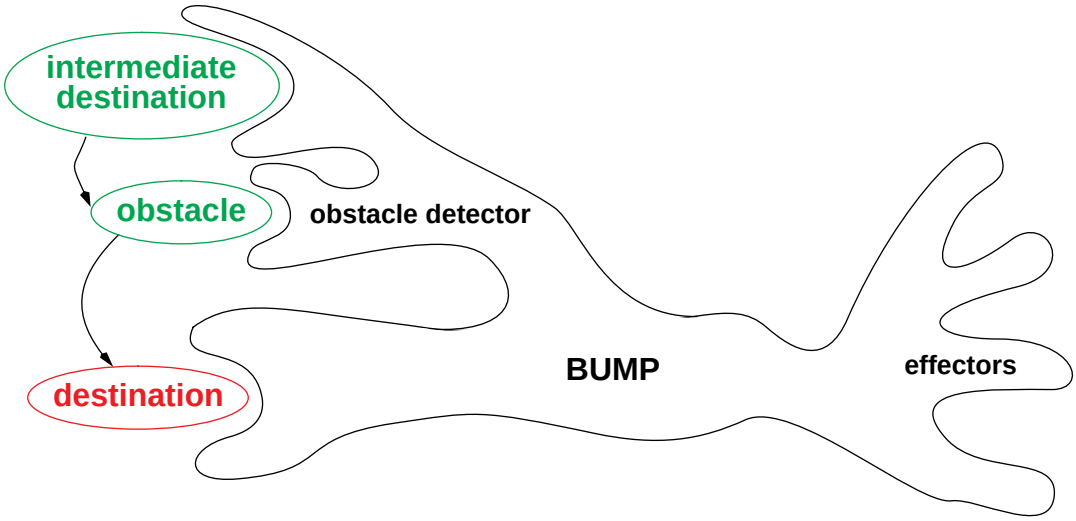


Figure 3-4: BUMP instantiates *obstacle* and *intermediate-destination* markers

The BUMP task-agency, rather than allowing the agent to blindly move towards the destination, launches perceptual machinery to find any obstacles to its given destination. If an obstacle is found, it is marked with an *obstacle* marker. Obstacle markers are always active-only markers, since an object is only an obstacle to the extent that it is in the path to a destination. We say that obstacle markers are *dependent* on the associated primary destination, so that if the destination marker is ever deleted or changed, the obstacle marker is deleted as well. This situation is illustrated in Figure 3-4.

If an obstacle marker is instantiated, BUMP looks for a path around the obstacle. When an appropriate location is found, BUMP instantiates an *intermediate-destination* marker which is dependent on the obstacle marker. The agent moves towards the intermediate destination just as it would any other destination. Reaching the intermediate destina-

tion triggers a reevaluation of the "obstacleness" of the associated object with respect to the primary destination.

As the domain becomes increasingly complex, this pattern can be extended to hierarchies of markers. These task and marker hierarchies bear resemblance to the classic concept of a partially ordered plan [25], with the task hierarchies equivalent to the increasing plan detail found in constraint-posting nonlinear planners. Further research is needed to determine the full relationship of marker hierarchies to partially-ordered plans. Marker hierarchies may potentially bridge the gap between reactive and classical planning, at least for navigation tasks.

The entire process—searching for an obstacle, creating intermediate-destination markers, and moving to the intermediate destination—repeats until there is no longer an obstacle in the path to the original destination marker, and the destination location is reached. The combination of destination, obstacle, and intermediate-destination markers is sufficient for most simple navigation tasks.

3.3.1.4. Tentative markers

A single image may contain evidence for the existence of an object; however, that "evidence" may simply be an artifact of the noise in the imaging process. If we immediately mark an "object" without retaining some measure of our confidence of the object's existence, the control system may act upon the marker as if there is no doubt as to the existence of the object, even if the evidence is scant. As discussed in previous sections, the predicate regarding the existence of the object has a high degree of uncertainty. While there may be a continuum of "certainty," we adopt the simplification of categorizing markers into two classes based on their certainty measure: those we intend to act upon, and those we do not. *Tentative* markers are placed on objects for which there is some evidence, but which we do not intend to act upon until more evidence for their existence is obtained. Such predicates are only potentially useful, in that they are not being used in a current plan. Further evidence may be obtained, and the predicate will become useful in the full sense. Note that the agent may construct a plan to obtain such additional evidence.

Our intent with the tentative markers is to filter out noise in the perceptual input, so one useful measure of certainty is whether the percept is stable over a sequence of images. One implementation of a certainty measure can therefore be a count of the number of times the object has been found in the image stream at the expected location. When first seen, an object is given a tentative marker, but when the number of frames in which it has been seen crosses some threshold, the tentative marker is upgraded to an activating marker, which activates the appropriate task-agency. This implementation is especially effective if coupled with an ego motion induced change in viewpoint. If the viewpoint

changes between consecutive frames, the location of the potential object's projection in the image will change. If the ego motion is known, the new location of the projection in the next image can be predicted. It is highly unlikely that noise will move in the image in a way that is consistent with the ego motion, so the tentative marker on noise will be dropped.

3.3.1.5. Hypothesized-object markers

In some cases, especially in a 3D environment with occlusion, it is necessary to reason about objects that *might* exist, even though they are not currently visible, or possibly have not even been seen yet. For example, consider an agent in the presence of rocks and predators. Since a predator might be behind any given rock, the agent could consider the possibility that such a predator exists. In fact, the agent might behave as if there were a predator behind every rock. We can induce this behavior by hypothesizing the existence and location of a predator, and placing a *hypothesized object* marker labelled "predator" at that location. In some cases, the desired action to take depends on whether the object is actual or hypothesized, whereas in others it is not necessary to make such a distinction. An agent, once it hypothesizes the existence of a predator behind a rock, may behave exactly as if there were an actual predator. The marker update procedure will treat real and hypothesized object markers identically.

3.3.2. Marker maintenance

In our control system, at every instant of time an action is selected for execution based on the information about the state of world currently available. In a purely reactive system, the available information about the world is limited to the current sensor values only. We expand on the reactive model by including the state of the markers as additional information, thereby expanding the agent's effective field of view. The advantage of this approach is that the agent is better informed when determining an action, since it can now remember a portion of the world around it. A disadvantage of this approach is that it admits the potential for the representation to be incorrect with respect to the actual state of the world (recall discussion in section 3.1.1.2). We have found that an incorrect representation is worse than no representation at all, since it encourages the agent to confidently take an action that is not at all appropriate in the current circumstances. Brooks reports that adding representation to an agent usually decreased the competence of the robots [12]; we believe his observation is a result of this phenomenon. Representation is not intrinsically bad—it's only bad if it's wrong. It is therefore critical that the representation be maintained accurately, or discarded. There is of course the additional consideration of the cost of computing the information, and also the cost of computing an action given a

large amount of information to examine, but correct information itself is not harmful. We must first strive to maintain the correctness, and hence the usefulness, of the information.

3.3.2.1. Compensating for ego-motion

When the agent moves, updating the markers requires estimating the coordinate transformation between the agent's previous and current locations. It is then a simple matter of applying the transformation to all the marker coordinates. We distinguish short term and long term approaches to estimating the transformation. Short term methods estimate the transformation continuously (or as near to continuous as possible), but have a tendency to drift from the actual transformation over time due to accumulated error. Long term methods are used periodically to correct the drift.

There are three main short-term methods: dead-reckoning, acceleration measurement, and optical-flow [32] based methods. Dead-reckoning assumes that all motion of the autonomous agent is due purely to the motor commands given by the agent itself. Since the agent knows the motor commands issued, the expected motion can be calculated directly. However, motor commands are often not executed perfectly, and forces external to the agent can influence the motion. If the motor commands are accurate and well characterized, this is the easiest option to implement. Another method is to measure the acceleration continuously, and then integrate twice to determine change in position. Measuring the acceleration enables the marker locations to be updated even if the motion of the agent is not purely the result of the agent's motor commands. Optical flow can also estimate the transformation under these conditions. An ideal system will use a combination of all three methods.

The primary long term method is to locate several points in the environment, and then solve a structure from motion problem. However, this option requires a correspondence problem be solved. (Another possibility might be to use a "global positioning system," if it is an option in the domain.)

3.3.2.2. Correspondence for visible markers

Marked objects are not featureless points; they have perceptual properties that allow them to be identified. Those same properties can be used to re-identify them later. Clearly, to do this we must retain some of the perceptual information associated with the marker from frame to frame, i.e., primary visual cues, e.g., the color and size of a berry. The obvious place to keep this information is in the marker.

Our definition of "marker" required only two elements: a role in a task (what), and a location (where). This does not mean that markers must be limited to only these two pieces of information. In order to maintain object/marker correspondence, markers must

also retain some information about the perceptual aspects of the objects they mark. We might say that markers must have the ability to “find themselves” in the new perceptual input, and can therefore be augmented to retain additional information to aid in the correspondence process.

3.3.2.3. Perception overrides memory

As we have discussed, memory for the location of objects runs the risk of being incorrect, and acting on incorrect information is potentially hazardous. Information derived from current perceptual input is more reliable than that stored in memory, and therefore overrides the information in memory. In a first cut at an implementation of this policy, whenever a marker cannot find itself in the current input, the marker is dropped. But this policy would drop any marker outside of the current absolute field of view, so we amend it to dropping any marker we *expect* to find in the input but do not. For visual input, the two main cases in which we do not expect to find the marked object are when the object is outside the absolute field of view, e.g., behind the camera, and when the object is occluded. The field-of-view case can be detected by knowing the field of view of the sensor and comparing it with the location stored in the marker. The second case requires additional visual processing; one must determine that there is some object along on the azimuth towards the marker, but is nearer than the marked object. The visual routine that performs this occlusion computation is only executed when a marker indicates that an object should be in the field of view, but no visual cue for the object is found [e.g., see section 5.2.2.1]. Note that this policy enables the maintenance and deletion of hypothesized object markers without any special machinery.

3.4. Conclusion

The nature of the real world and real sensors dictate that a task-oriented approach be adopted in constructing autonomous agents. Exactly what the task-oriented approach means in terms of perception, action, and representation is summarized in the concept of an effective field of view. The effective field of view contains exactly that information that the agent can use to determine the actions necessary to perform tasks in the world.

In a complex world, an agent may have multiple tasks to accomplish, and each task induces a different effective field of view, which may result in conflicting determinations of the actions to take. By organizing the separate tasks into task-agencies, we can identify and mediate conflicts among the task-agencies.

The use of markers is a means of expanding the effective field of view, and communicating information between task-agencies. There are several types of markers, depending on their relationship to the task agencies and to each other. By recognizing the

different roles of markers in the tasks, we can build strategies into our agents for acquiring, maintaining and using the markers to achieve the agent's goals.

Chapter IV: Applications and Primitives

The task-oriented approach to agent design requires that the use to which the agent will be applied is considered explicitly in the design. There are any number of tasks we might want an agent to accomplish, which presents the possibility that a complete redesign of an agent is required for each task. However, the tasks with which this dissertation is concerned all involve action in the real three-dimensional world, and there is a great deal of commonality in the tasks. Rather than begin from the ground up for each agent, we can identify a set of primitives which can be composed to construct arbitrary agents. This chapter presents a number of example tasks, and then abstracts from those tasks to identify a proposed set of primitive operations. The central thesis of this dissertation is that the use of a small amount of representation can be used to expand the effective field of view and thereby increase the performance of situated automata. The applications and primitives described in this chapter are therefore discussed primarily in terms of these representations and the specific implementation of these representations as markers.

4.1. Examples in the problem space

The marker-based representations developed in this dissertation are meant for use by agents in dynamic three-dimensional environments in which the agent has one or more tasks to accomplish. Clearly, one should not assert that marker-based representations are necessary for *all* tasks in *every* such task domain. I do claim, however, that these representations have a broad range of applicability in a large class of useful domains. This section delineates the applicability of the 3D marker-based representations through the use of a number of examples.

The sample domains discussed in this chapter have a number of aspects in common. Naturally, we confine the discussion to the *dynamic*, *three-dimensional* environments in which 3D markers are applicable. Furthermore, we intentionally do not consider task domains that require what is colloquially meant by “intelligence” or “creativity.” These tasks do not require generation of new knowledge—one might expect to be able to write an algorithm for them if requisite perceptual and effector technology existed. As our first example, consider the task of taking out the trash. Clearly, one need not be intelligent or creative to accomplish this task. One might write an algorithm for this task as follows:

1. Get a twist tie
2. Remove the trash can lid
3. Seal the bag with the twist tie
4. Carry the sealed bag to the dumpster
5. Open the dumpster lid

6. Put the bag in the dumpster
7. Close the dumpster
8. Get a new trash bag
9. Put the trash bag in the garbage can
10. Replace the garbage can lid

One might argue that "intelligence" is required to *construct* this algorithm. However, it should not require intelligence to *follow* this set of instructions, provided they are expressed in a language comprehensible to the agent expected to carry out the task. I do *not* intend for the agent itself to construct the "plan;" rather, a human designer is to construct the plan, and the agent is to follow it. By focusing on the role of a human designer, I do not mean to imply that it is impossible to construct a software system that designs the plans. On the contrary, the tasks discussed are expected to be amenable to automatic plan generation—however, that is a topic for future research.

Below is a list of sample applications in which I expect the techniques developed in this dissertation to be successfully applied. We want to be able to program a robot to accomplish these tasks in dynamic and sometimes hostile environments, e.g., the house should be cleaned while people are living and working in the house and messing things up. The robot should not destroy anything in the house (including the inhabitants, or any pets) in the course of cleaning.

1. Cleaning house
2. Getting from NY to LA
3. Driving in traffic
4. Playing basketball
5. Going down the basement stairs
6. Loading and unloading the dishwasher
7. Setting the table
8. Packing a suitcase
9. Running office errands
10. Building a house
11. Conducting experiments in a biology lab
12. Picking up trash
13. Mowing the lawn
14. Making breakfast
15. Baby-sitting
16. Waiting tables
17. Any minimum wage job
18. Exploring Mars
19. Rabbit World

Some of the tasks listed above are more "useful" than others (e.g., the usefulness of a basketball playing robot is dubious), but the dynamic and complex nature of all of these tasks are central to this research program. Note further that the marker-based representations that are the focus of this dissertation must be augmented for many of the tasks,

e.g., getting from NY to LA requires a map. Another goal of this chapter is to delineate those tasks that marker-based representations are *not* good for. In the remainder of this chapter, I will briefly describe the tasks in the list above and very high level strategies for accomplishing them.

4.1.1. Cleaning house

There are many sub-tasks of cleaning house, but in this particular section, by "cleaning house," I mean that the robot is to find things that are "not in their place" and put them "where they belong," e.g., picking up children's toys and putting them away. There is, of course a fundamentally difficult recognition problem associated with this task, which I do not expect to address in this dissertation—I will assume an adequate recognition algorithm exists. Furthermore, the agent must know where everything belongs, i.e., there must be some kind of representation of the layout of the house, and there must be a list of items (or types of items) in the house, with pointers to where they belong, e.g.,

Board Games → **Top shelf in closet of bedroom 2**
Dirty Clothes → **Laundry hamper in hall**
Children's Books → **Bookcase in bedroom 3**

The actual representation of the house can be topological, metric, or some combination of the two. In any case, the agent must be able to navigate to the locations given in the list of items. Note that none of the representations discussed thus far is marker-based; a variety of representations must be used to perform complex tasks. These other representations are *useful*, in the sense that they are predicates about the world that the agent can use in a plan to achieve its goals. The concept of usefulness applies to all representations, not just markers.

Markers are used once an object is found to be out of place. The object itself is marked, i.e., its location relative to the robot is noted and updated as the robot moves. Marking an object in this way amounts to asserting that information concerning the object's location and identity is useful, and marking the object expands the effective field of view to include the object, even if the object is not currently in the absolute field of view.

Assuming there is a straight-line path to the object, the robot can simply servo on the marker location until is arrives near enough to the object to pick it up. This behavior demonstrates two fundamental primitives we expect our mobile robot to understand:

Mark object
Go to marker

Note here that maintenance of the marker is implicit in **mark object**, and obstacle avoidance is implicit in **go to marker**. Once the robot is near enough to the object to grasp it, the robot needs to pick the object up. While marker-based representations are useful for manipulation, (e.g., marking grasp-points on the objects), I leave a complete exposition of manipulation to future work. For the purposes of the current discussion, we will assume a **Pick up marked object** primitive.

The next part of the task is to put the object where it belongs. The robot must have a representation of the house, and know where the object belongs with respect to that representation. Let us assume that the robot has a topological map of the house, as in Figure 4-1, and that the robot has picked up a child's toy in the Living Room, and needs to put it in Bedroom 3.

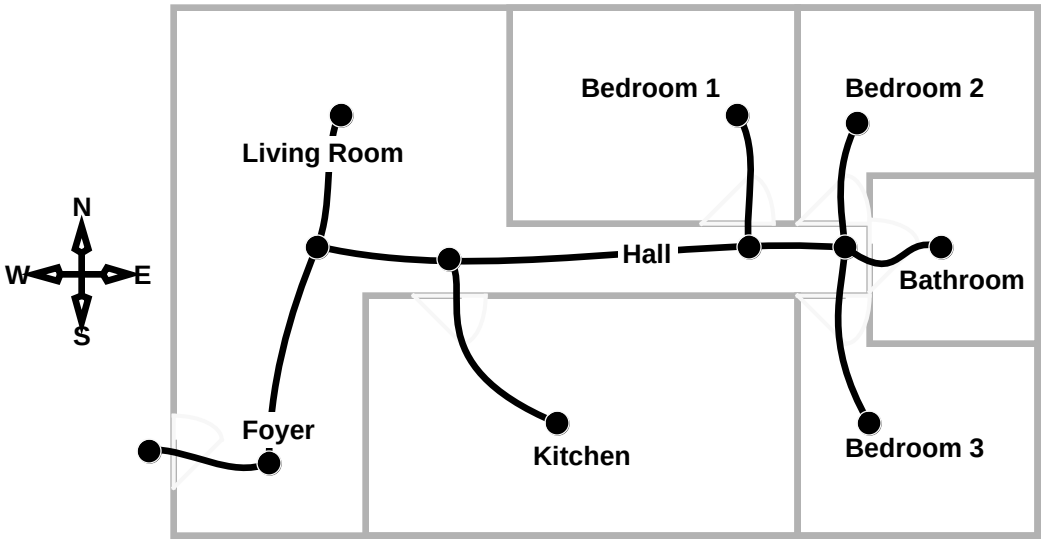


Figure 4-1: A map of the house to clean

The robot consults its topological map, and finds that it must go to the east end of the hall and turn right. This navigation is performed via a series of "mark location/go to marker" pairs: first by marking and going to the west end of the Hall, then the east end of the Hall, and finally into Bedroom 3. Once in the room, the robot can mark the location in the bedroom where the object belongs, and place the object there using a **Put object at marker** primitive. The algorithm for putting something away is summarized in the "*Put something away*" routine. The boldfaced items are primitives, while the others are "subroutines," as indicated by the parentheses. I will be rather loose with the syntax of the pseudocode in this section—the later sections will have a slightly more formal syntax.

For the purposes of exposition, the primitives discussed in this chapter are described as being composed in a linear sequence of actions. Writing such a *explicit* sequence of instructions may be the most intuitive way to construct the algorithm for put-

ting something away, however, the implementation in the architecture to be described later will use *implicit* sequencing (see section 4.2.6) when executing an algorithm. The situated automata model of agent design is not conducive to the execution of linear sequences in the standard way, rather, actions are taken when their preconditions are met, and the condition the action produces is a desired one. Linear sequences of actions, therefore, are implemented by having each action in the sequence produce the preconditions for the next action in the sequence. Sequencing is therefore *implicit* in the preconditions and effects of the actions. Implicit sequencing will be discussed more in depth later, but for now, I simply want to provide a warning that the explicit linear sequencing shown in the next few sections (because explicit sequencing easier to think about at this point) is not the model we are working towards.

```
Put something away() {
        Find out-of-place object ()
        Mark object location
        Pick up marked object
        Go to location in map ()
        Mark destination location
        Put object at marker
}
```

Cleaning the entire house consists of finding objects throughout the house and putting them where they belong. We can write a simplistic algorithm for this task as:

```
while (find something out of place) {
        Put something away()
}
```

However, this algorithm is extremely inefficient, since the robot would likely spend a lot of time moving from one end of the house to another, similar to the “Big Shell Game” example in [8]. A more efficient algorithm is to clean one room at a time, confining most activity to a single room. “*Clean house*” is a series of “*Clean room*” routines.

Using the primitives of marking objects, picking up marked objects, marking destinations, and putting objects at marked locations, one can easily write algorithms for *put_object_near_door() and put_object_in_proper_room().* This example illustrates the flexibility of the proposed primitives to construct alternative algorithms.

```
Clean house () {
        for (each room in the house) {
                Clean room()
        }
}

Clean room() {
        while (find something out of place in room)
                Mark out of place object
```

```
                    if (object belongs in room) {
                        put away object()
                    else
                        put object near door()

            for all (objects near door)
                    put object in proper room()
}
```

4.1.2. Getting from NY to LA

To get from NY to LA, the single most important item needed is a plane ticket. I usually get plane tickets by calling a travel agent. An often useful strategy for a robot working in natural environments is to ask for assistance from a human (or another robot), such as a travel agent. There may be any number of things that the robot needs help with, but I will confine myself here to discussion of getting help with obtaining physical objects (such as plane tickets). The robot may need an object in order to accomplish some goal, and may not have a plan for obtaining that object. The default plan for obtaining such an object is to ask the nearest human for it. The human may give the item to the robot directly (e.g., “here is your ticket”), or give the robot a plan for obtaining it (e.g., “call the travel agent and ask for it”). The issues involved in the communication are addressed in e.g., [16] and [56]. Getting assistance from humans or other robots/agents is useful enough that we should consider it a primitive operation:

Ask for object

where **object** is something the robot needs to accomplish its goals and can name. It is also important that the robot know the goal it expects to achieve using the object, since this may help in eliciting a better response from the human. For example the robot may attempt to ask for a screwdriver, and the response may be “I don’t have a screwdriver.” However, if the robot asks for a screwdriver to “pry open this box,” the response may be “use this letter opener instead.” When interacting with humans or other independent entities, it may often be necessary for the robot to communicate its goals. The robot must therefore always be prepared to answer the questions “what are you doing?” and “why?” Note that in order to do this, the robot architecture must represent its goals explicitly, and be able to communicate them, as in Shoham’s “agent-oriented programming [56].”

Once the plane ticket is obtained, one needs to get to the airport, perhaps via a cab, so we ask for a cab, similar to the way we obtained the plane ticket. In this example, we add a clause communicating the goal of getting the cab, e.g.,

Ask for cab to get to flight at 3:15 pm

The additional goal information communicated here enables the provider to schedule the cab appropriately. Note also here that there is a structure imposed on the goals, i.e., the goal of getting the cab is a sub-goal of getting on the flight, which in turn is a sub-goal of getting to LA. If for any reason the goal of getting to LA is retracted (e.g., the conference is cancelled), then all of the sub-goals are retracted as well. The agent must explicitly represent this hierarchy of goals in order behave appropriately. We can think of this as giving the agent the ability to answer repeated "why" questions, e.g.,

Robot: *I need a cab.*
Human: *Why?*
Robot: *To get to the airport.*
Human: *Why?*
Robot: *To go to LA.*
Human: *Why?*
Robot: *To go to a conference.*
Human: *You don't need to go to a conference—you're a robot! Go fix me a martini!*

Assuming the robot retains the goal of getting to LA, the remainder of the task is a matter of getting to the various transfer points at the appropriate time, e.g., to the cab when it arrives, and to the gate to catch the plane. The most interesting of these navigations is getting from the cab to the gate in a crowded airport. Performing this task requires domain knowledge about airports, the ability to use maps, and the use of marker-based representations in local space.[1]

Domain knowledge about airports is needed to determine the gate to catch the plane. Airports have displays that map from flight numbers to gate numbers, and the robot will need to know to look for such a display, and furthermore, where to look. Once the gate is determined, the robot needs to look for directional signs to the gate and follow them. Following a series of directional signs is identical to the skill of following a path in a topological map, except that the series of signs is in the robot's memory, rather than in the environment. In getting from one signpost or landmark to another, marker-based representations of the local space are needed. The robot places a marker on the next location in the map or the next signpost, and navigates toward it as was done when navigating the house in the previous section, and the entire journey is performed as a series of "mark location/go to marker" pairs.

1. Of course, the biggest problem for a robot would be getting through security, but we won't address that issue here.

4.1.3. Driving in traffic

Like navigating in an airport, driving in traffic requires elements of domain knowledge, large-scale maps, and local-space representation. Domain knowledge is needed concerning the operation of the vehicle and rules of the road—the things one learns in driver's education. Large scale topological map following is needed to navigate to the destination using signs and landmarks. And finally, one needs to deal with the dynamics of the local space, since there are other vehicles on the road. The unpredictability of other drivers makes it impossible to navigate open-loop on a real road, even if we had a perfect metric map of the path to the destination. Even if we could guarantee there are no other vehicles on the road, navigating open loop is a losing proposition, due to the imperfectly executed motor commands characteristic of any real system. In the remainder of this section, I will concentrate on the local-space problem.

While travelling along the highway to the next landmark, the area of interest is primarily in front of the vehicle. In fact, no matter what the form of locomotion, it is fundamentally more important to pay attention to where you are going than where you've been, since the obstacles are in front of you. While travelling along the highway, there might not currently be a vehicle in front of us, but if we come up on one, we'll need to mark it. Being prepared to mark an object that is noticed in some region is important enough to declare a fundamental operation for this task—a monitor operation.

Monitor area in front of the vehicle for traffic

The size of the area to monitor depends on the task. For driving, we need to monitor far enough ahead that we can brake effectively, and this is dependent on our speed and the road conditions. Note that in contrast to the previous operations, monitoring is a continuous process, rather than a one-time operation. Once a vehicle is noticed in front of us, it is marked, and if necessary, we brake. If we need to change lanes for any reason, then the area near us in the lane we want to move into is monitored. Another way of looking at this is that the "forward" direction changes when we want to move into the other lane.

Simultaneous with monitoring the road immediately in front of the vehicle, we are also monitoring the terrain for the next landmark or road sign. Since the vision sensors are limited, we must accomplish this "simultaneous" monitoring by time-sharing the vision sensor and processing resource. Changing the direction of gaze and remembering the important information over a series of gazes amounts to using memory to expand the effective field of view. An implementation of this idea is to use the concept of *duty cycles*. Depending on the importance and dynamics of the task, different monitors have different

duty-cycles, or rates at which they must use the vision processor. Monitoring for landmarks might have a duty-cycle of several seconds, since the landmarks are usually visible from rather far away, and missing a turn would be unpleasant, but not disastrous. Monitoring the road in front of the vehicle has a duty-cycle of a fraction of a second, since if we are moving quickly, another vehicle could "appear" in the road in front of the vehicle rather quickly, and the consequences of not noticing another vehicle stopped immediately in front of us are dire.

Once an object is marked, duty cycles can also be used to implement marker maintenance. The certainty of the location of the object when it is first detected is quite high, and may remain high for some period of time immediately afterward, i.e., the predicate concerning the object's location has temporal extent. In a dynamic world, the certainty deteriorates over time, so we may need to look at the object again to verify its location. How often we need to verify a marker is the marker's duty cycle.

4.1.4. Playing basketball

I'm not much of a basketball player, but what little I know about the game has a number of interesting aspects from the point of view of local space representation. In contrast to the previous tasks discussed, there is nothing in the way of a large-scale topological map of the space—all of the relevant aspects are local. There is a good deal of high-level knowledge involved for the rules of the game, and there are a lot of *skills*[1] needed, many of which are specific only to basketball, such a dribbling, shooting, passing, and shot blocking. In accordance with the basic model of situated automata, these skills are launched at the appropriate times based on the situation, and that situation is determined based on the spatial relationships of relevant (marked) objects such as the basket, the nearest opposing team member, and the basketball itself. Marker-based representations are at the core of the vast majority of basketball playing skills, and for this reason, basketball is a near-ideal domain to study the use of marker-based representations.

For example, when playing a "man-to-man[2]" defense, the primary objective is to stay between "your man" and the basket. It's also a good idea to know where the ball is, in case it is passed to your man. So three markers are required, one on each of the starred entities in the figure below. Given these markers, you can determine the direction to move to stay on the line between your man and the basket (shown as a dashed line in the figure).

1. These skills are similar to Firby's reactive action packages (RAPs) [23].
2. Please excuse the gender-specific term; in basketball, an opposing player is referred to as a "man" even if the "man" is a nine year old girl.

Maintaining these markers requires you to spend most of the time looking at your man, since that is your greatest responsibility, and he moves a lot, so the predicate concerning his position has a very short temporal extent and resulting duty cycle. Frequent glances toward the ball are needed also, since the ball also moves quickly. Less frequent glances are needed towards the basket, since it doesn't move, and its position can be maintained by dead-reckoning. There are also other visual cues that allow the position of the basket to be maintained, since the geometry of the court is constant and well known to the players. For example, if you can see the top of the key out of the corner of your eye, then you have a pretty good idea where the basket is, even without looking at it directly.

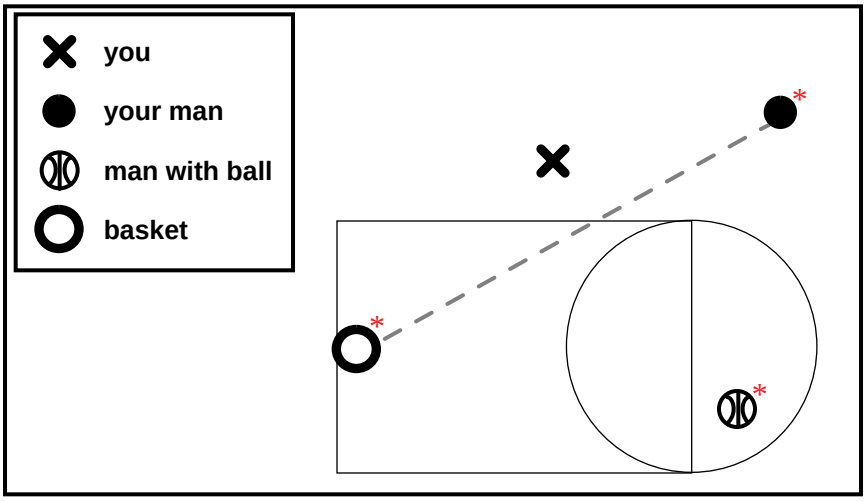


Figure 4-2: Playing defense in basketball

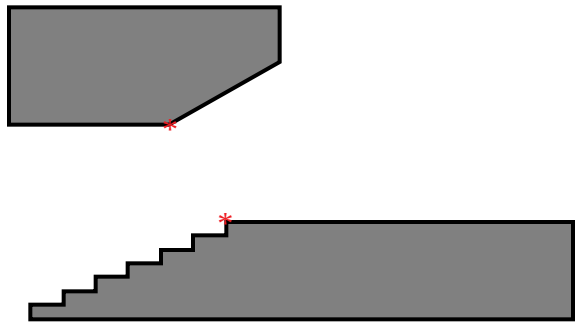
When playing a "zone" defense, the player is responsible for a region of the court; which corresponds to a monitor operation. The "monitor" region is somewhat larger than the actual zone, so that the player can mark opposing players that might enter the zone, so that there is a better response time when the opponent actually enters the zone. If an opponent enters the zone, the situation is similar to the man-to-man case, except that if the opponent leaves the zone, the defender drops the marker and doesn't follow.

Offensive players monitor the region between themselves and the basket. (Note that the monitor region is not a fixed location in any coordinate system, even an egocentric one; it is dependent on the spatial relationship of two objects that are moving relative to one another.) If a player has the ball, and the monitor region is empty, that player can drive to the basket, i.e., launch the "drive to basket and shoot lay-up" skill. Good players don't just wait for this opportunity to present itself; they have a number of ball-handling skills they use to try to create this situation. Well-coached teams also have a number of cooperative strategies (such as a "screen") to create the opportunity to drive to the basket. In addi-

tion, players that shoot well from the perimeter monitor the region very near to them in the direction of the basket, so that they might be able to take a shot from the outside without it being blocked.

4.1.5. Going down the basement stairs

Consider a situation in which a tall person walks down a flight of stairs over which is a low ceiling. A traditional model based approach would have the person examine the steps and ceiling, construct a model of it, and generate a plan for the correct posture needed to negotiate the stairs which is then executed. The scene need only be analyzed once. But this is not what people do; they continually shift their gaze back and forth between the stairs and the overhang until the overhang is passed. I suggest that the person is querying the world, in this case first asking questions such as “is there an object near my head?” This question might be asked continually over the entire visual field. When the query returns true, the person places a marker on the object, in this case the overhang. Another process might continually be looking for uneven terrain (such as stairs), and return a marker on the location of the stairs. Questions can then be asked with respect to the markers, such as “is my head getting too close to the overhang?” and “am I putting my foot in the right place?” These last two questions are the queries being made as the person’s gaze shifts between the ceiling and the stairs. Much more could be learned about gaze strategies for locomotion in humans from controlled psychophysical experiments that track a person’s gaze while walking. Implementing this marker-based strategy uses



the monitor operation over the area of space in the direction of travel, and then mark operations for the stairs and overhang (starred in the figure), with appropriate duty-cycles for the stair and overhang marker maintenance.

Loading and unloading the dishwasher, setting the table, running office errands, picking up trash. These tasks fit a general pattern that we saw in the “house cleaning” task, namely, picking up things and putting them in other places. The general mechanisms described in the house cleaning sections are directly applicable to these tasks, which illus-

trates the power of these few simple mechanisms. With some slight variations, one can also use the marker operations described thus far for the remaining tasks: *packing a suitcase, building a house, conducting experiments in a biology lab, mowing the lawn, baby-sitting, waiting tables, any minimum wage job, exploring mars, and making breakfast*. Constructing these variations is left as an exercise for the reader.

4.2. Behavior “primitives”

By surveying the list of tasks discussed in section 4.1, we can make a first-cut at the primitives that are needed to accomplish the tasks. When considering the form of the list of primitives, the question arises as to the level of abstraction at which one should refer to items as “primitives.” At the high end of spectrum, the task list itself might be considered a set of primitives, e.g., there are primitives for “Cleaning House,” “Getting from NY to LA,” etc. I think all would agree that this is far too high level, especially given the current state of the art in autonomous agents. At the low end, one might consider the set of primitives as the instructions directly realizable in the physical robot, e.g., “Drive Motor A at Speed X.” But this approach would tie us to a given machine architecture, and furthermore, it would require the robot programmer to work at the “machine language” level, whereas we are interested in the equivalent of a “high-level language” for behavior. Clearly we need “something in the middle,” but where?

By the end of this section we will have at least a high level idea of a language in which to express instructions for tasks such as taking out the trash; a “language of behavior” if you will. Marker-based representations play a fundamental role in this language. Marked objects are the data on which the primitives operate.

One way of thinking about the set of primitives is as the interface between reactive and classical planning. The classical planner (or human designer) constructs a routine using the set of primitives described below. The reactive system executes this plan, and therefore has mechanisms for detecting when a action is executable, and monitoring the execution of these actions.

4.2.1. monitor (region, object type, duty cycle)

The plans for agents which act in the physical world refer to physical objects that the plans manipulate. However, these plans may not have knowledge of specific objects that are to be used in the plan. The perceptual system will locate these objects as a part of the plan. This is in contrast to the classical planning approach, in which all the objects in the plan are known and available to the planner before the plan is generated. The planner

(or human designer) will insert the action **monitor (region, objecttype, dutycycle)** whenever a physical object is needed to be found to instantiate a parameter to an action.

The monitor action accepts two parameters, a region of space in which to operate the sensor, and the type of object which is of interest in that region. The object type has a perceptual action (e.g., a visual routine) associated with it, which is used to determine if an object of that type is in the current perceptual input. The effect of a monitor operation is to direct the sensor toward the given region and apply the perceptual action that corresponds to the object type. If the perceptual action finds an object, that object's location is made available to downstream processing.

The monitor operation is an ongoing process; the agent repeats the perceptual action until an object is found. This is because the world is dynamic, and the object of interest may come into the sensor's field of view. The rate at which the perceptual action is repeated depends on the duty cycle parameter. The monitor action repeats the perceptual action, once per duty cycle, until the monitor operation is explicitly cancelled.

One can implement the monitor operation using one or more hypothetical markers. By placing a hypothetical marker in the region to monitor, the mechanism for maintaining markers can be applied to perform the monitor operation. If the monitor region is larger than the absolute field of view, then a number of markers can be used, spread throughout the monitor region, so that the sensors are directed at each marker in turn, thereby covering the entire region. This approach was used in the implementation described in later chapters.

4.2.2. mark (object)

In the remainder of the primitives described below, the parameters are all marked objects, so a fundamental primitive must be to "mark" an object or location. To mark an object simply means to note the location of the object, and then track the object's location in an egocentric coordinate system. Note that maintenance of the object's location over time is implicit in the mark operation. The object marked has a "type," which assists in the maintenance task. The object type has an associated visual routine which is used to relocate the object in the visual input. Since the object at this point has already been found in the visual input, a more specific visual routine can be used than that which was used by the monitor operation to find the object initially. For instance, if the object to mark is a rubber ball, the monitor operation used to find the rubber ball initially must use the generic "rubber ball finding routine." Once a specific ball has been found more information is available to the maintenance procedure, e.g., if the specific rubber ball located during the

monitor operation was red, the marker maintenance procedure can use the “red rubber ball finding routine,” which may be more efficient than the generic rubber ball finding routine.

The type of the object to mark may also dictate a duty cycle for verifying the object location, e.g., objects that are known to move may have shorter duty cycles than stationary ones. The object type may even have a motion model that can be used for estimating the object’s location as it moves.

Not only are the marked locations parameters to the primitive actions, but together with the current sensor input, the markers form the model of the local space that the agent uses to select the actions to take (i.e., the markers are used to expand the effective field of view). All the object specific techniques for updating marker locations described here, together with the general marker maintenance techniques discussed in section 3.3.2 must be brought to bear to maintain the object locations as accurately as needed for effective operation of the agent, both in terms of determining the action to take, and facilitating an action once it is selected.

4.2.3. goto (marker)

Once an object is marked, we may want to move towards it using the **goto (marker)** operation. Obstacle avoidance is implicit in the goto operation. Also, we will not want the goto operation to try to put the exact center of the agent at the exact center of the marked object. The actual location to which the agent should go depends on the next action in the sequence. For example, if the next action is to pick the marked object up, the agent should move into such a position that the agent’s manipulators can grasp the object. We can take care of this problem to some extent through the use of implicit sequencing. As the agent moves towards the marked object, the agent will eventually be close enough that the preconditions of the action that picks the object up (**get**, below) will be satisfied. At this point, the get operation takes over, and it makes sure that the agent moves into the correct location to pick the object up.

4.2.4. get (marker) and put (object marker, destination marker)

In examining the long list of tasks described in section 4.1, I found it remarkable how much of these tasks consisted of variations on picking things up and putting them down somewhere else. I would want a robot that did nothing but sequences of reasonably robust get, goto, and put operations. These operations, used in the context of the framework developed in this dissertation, would provide most of what I personally expect from a mobile robot, and indeed, from artificial intelligence. Naturally, difficult perception and

manipulation problems must be solved before any of the operations can be “reasonably robust.”

In order to be robust, the get and put operations must be parameterized by the type of object. Getting an egg is quite different from getting a piano, the operations must reflect this fact. Clearly, agents that will be built in the near future will have a rather short list of types of things they can get or put.

4.2.5. askfor (marker) < because (goal) >

This primitive facilitates cooperative behavior between artificial agents, and between artificial agents and humans. It can also be used to compensate for the some of the limitations of the agent by allowing the agent to ask a human for things it cannot get for itself for some reason. This use should be applied sparingly, however; I would not want to have a robot that did nothing but ask me to do all of the work.

4.2.6. Implicit sequencing

The sequences of actions shown in the sections above when discussing tasks have all been explicit—one action follows the next in sequence. But this does not fit the standard model of situated automata, in which actions are taken based on the state of the world as determined by the sensors. A situated automaton should not take an action simply because the program counter is pointing at that action in the sequence, since such an action may be totally irrelevant to the current world state. The previous action may have failed, or some completely unforeseen change in circumstance requires a different action. For instance, if you are in the middle of loading the dishwasher, and you notice that the frying pan on the stove is on fire, just putting the next dish in the dishwasher is not an appropriate action; you should tend to the fire on the stove, i.e., be reactive.

Actions should be taken because (1) they achieve a desired goal, (2) their preconditions are met, and (3) the resources are available to take the action.[1] Moreover, for the tasks with which this dissertation is concerned (such as those described in section 4.1), the preconditions and goals are all conditions in the physical world. However, the model of a situated automaton from Figure 1-1 contains memory, and the determination of actions taken by the effectors is mediated by the internal state. If the memory contains an explicit sequence of instructions and a program counter, result of this “mediation” could be to

1. The “fire on the stove” example is a case of the application of rule 3. The need to tend to the stove made the effectors unavailable to continue with the dishes. Actually, rule 3 is a special case of rule 2, since if the resources aren’t available, the preconditions aren’t met.

actually be to cause the agent to take the action to which the program counter is pointing, regardless of the current sensor input. In my view, this would be a misuse of memory.

The action selection should be based on the current state of the world, and the memory structure used by a situated automaton should simply be a reflection of the current state of the world. The use of memory simply allows the agent to know more about the current state of the world than that which is currently available to the sensors; this approach is reflected in the use of the term "effective field of view." The memory simply allows the agent to "see more," and therefore make a more informed choice of an action.

The preconditions are all states of the world, as opposed to the case in explicit sequencing, in which the precondition is purely a state of the agent (its program counter). Implicit sequencing comes about by having actions produce effects in the world that are preconditions for other actions. This ensures that the actions taken are appropriate for the current world state.

The use of implicit sequencing has consequences in the construction of the agent. No special sequencing mechanisms are needed for the case when a sequence of actions must be aborted; other actions just start executing. No special mechanism is needed for returning to and repairing aborted instruction sequences—the effects produced by the actions are either still true, in which case the sequence can begin where it left off, or they are not, in which case the necessary actions are repeated. Implicit sequencing also allows for skipping actions if the effects they produce are serendipitously found to be true already. The agent can also take advantage of potential parallelism if more than one action can be taken simultaneously.

The agent's control structure can be implemented as a production system. On the other hand, it is sometimes the case that it is easier for us as human designers to describe the behavior as linear sequences of actions. It may be possible to automate the generation of a set of actions that can be executed via implicit sequencing from a linear sequence of actions produced by the human designer. This is a topic for future research.

Future research is also needed in integration of partial-order planning with the situated automate model. It may seem that situated automata and partial-order planning are incompatible, but creating and following plans gets at reason (1) for a situated automaton to execute an action, i.e., the action achieves a desired goal. One must have a plan in order to know whether an action achieves a goal, even if it is a trivial one-step plan. But rather than treating a plan as a sequence of actions to be rigidly followed to achieve a goal via explicit sequencing, the plan can be treated as a resource that *helps* to determine the actions at each step, but does not *dictate* the action to take. This approach to plans was first discussed by Agre and Chapman [1, 2], and is further developed by Hayes-Roth [30].

Chapter V: Rabbit World™—A Case Study

5.1. The simulation

I have implemented a marker based control system that illustrates many of the concepts developed in the preceding chapters. Using the *Alice* virtual reality rapid prototyping software developed at UVa [19], I have constructed an environment with trees, rocks, berries, and walls. An agent survives in this environment by eating berries and avoiding obstacles. Furthermore, the environment contains a predator which actively seeks out the agent. The agent has the concurrent goals of finding food, avoiding obstacles, and avoiding the predator. An example of a rendered image that forms the input to the vision processor is shown in Figure 5-1. Rocks, trees, and berries are located on a flat field surrounded by a fence. A predator is also shown.

Figure 5-1: The virtual agent's environment

The textures are not reproduced well by our printer, but are evident in the rock surfaces. The textures are not complex; they are treated as noise by our vision processes. Each image is rendered from the agent's perspective as in Figure 5-1—new images are rendered as the environment changes, either due to movement in the environment (such as predator movement), or due to motion of the agent.

The environment simulation is a separate process, running on separate hardware from the agent control software. The a block diagram of the complete simulation system is depicted in Figure 5-2. On the left is the environment simulation; the agent control system

is on the right. Each of these two major subsystems is further subdivided into two parts. The environment simulation consists of a *simulation engine* and a *rendering engine*. The simulation engine maintains a complete representation of the environment, including the locations, speeds and other relevant state information of all objects in the environment. Collisions between objects are detected by the simulation engine as well. The rendering engine accepts graphics database updates from the simulation engine and renders the scene from the viewpoint specified by the simulation engine. The rendering engine runs on a Silicon Graphics ONYX Reality Engine, producing high-quality, full-color, shaded and texture-mapped images in real time. The simulation engine may also run as a separate process on this same machine, or alternately on a separate machine, such as a Sun SPARC Workstation. They communicate via a stream-type internet-protocol socket connection.

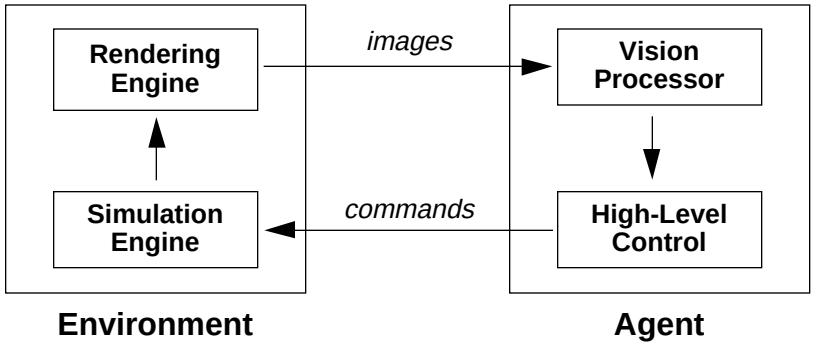


Figure 5-2: The complete simulation system

The agent control software does not have any access to the underlying representations used by the environment simulation, and must instead construct its own representation based on its perception of the environment. This perception consists entirely of the image sequence drawn by the rendering engine, from the viewpoint of the agent. After drawing each frame, the rendering engine quantizes the image to 64 colors, run-length encodes it, and sends it to the vision processor via a socket connection.

The vision processor identifies relevant items in the visual input: in this case, berries, the predator, and the ground line (described below). The locations of these objects in the image are forwarded (again via a socket) to the high-level control module. The control module is a multi-threaded process running on a Sun Workstation running the Solaris operating system. The agent's control module decides upon an action based on these images, and sends velocity commands to the simulation system, which updates the display based on the new location of the viewpoint. Even with the communication overhead of this design, the simulation runs in real time at 10-15 frames per second. The examples of

the 160x120 pixel, 64 color images that form the input of the vision processor are shown in Figure 5-3.

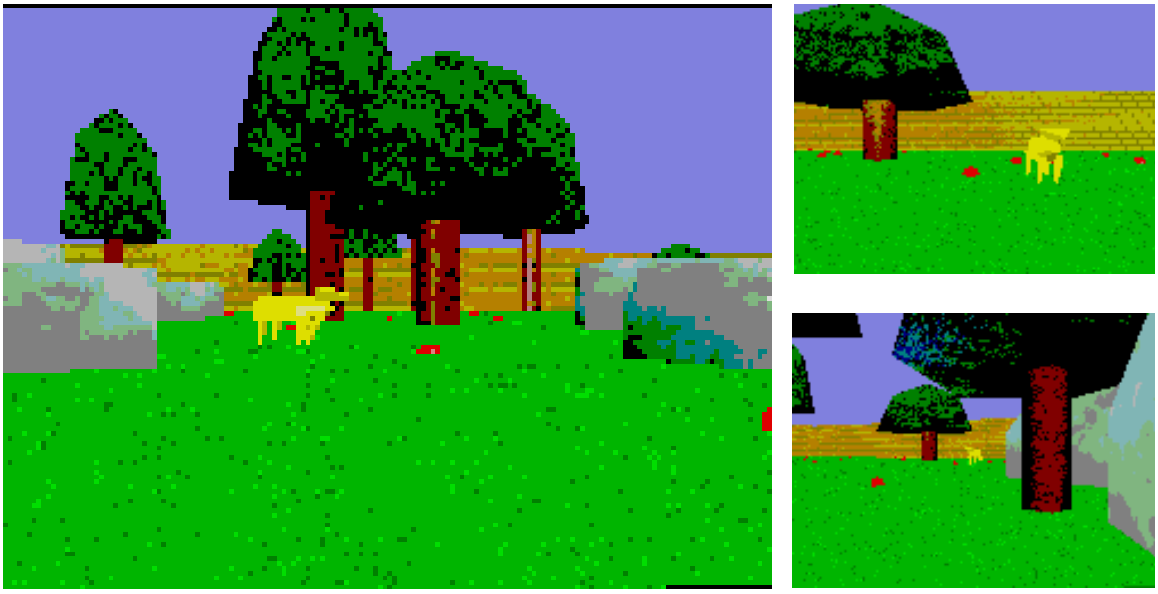


Figure 5-3: Images after quantization

In later versions, one additional feature was added—the ability to turn the viewing direction independently of the direction of travel, i.e., a “neck.” This feature was motivated primarily by the predator avoidance behavior needing to run away from the predator while periodically looking backward to verify the predator location. This necessitated additional machinery in the simulation and communication between the simulation and control system. A “head angle” command was added to the set of instructions the control system could send to the simulation. The complete set of instructions was therefore:

setForwardSpeed(s)
setTurnSpeed(s)
setHeadAngle(a)

The addition of the neck, and the distributed asynchronous nature of the simulation, made it necessary to add one additional “sensor” to the simulation, a proprioceptive “head angle” sensor. This is needed because after the control system issues a command, there is a nondeterministic delay before the command takes effect. The head angle sensor enabled the agent to know when head angle commands took effect. This is critical, since the images are useless without knowing the direction the eye was pointing when the image was generated. For example, if the agent is looking directly forward, issues a command to look behind it, and then sees the predator, the agent would not know whether the predator is in front of it or behind it without the head angle sensor. Each image was therefore paired

with the head angle at which it was generated, to disambiguate such cases. On a physical robot, such a sensor could easily be implemented via, e.g., a potentiometer.

Finally, another "sense" that the simulation provides is a sense of time that can be inferred from the regular production of the simulation frames. Each frame is generated a fixed amount of simulation time after the previous frame, so the production of frames provided a clock input to the agent. This was needed in order to update the markers properly when they are outside the field of view, since this update is done by integrating the velocity of the agent over time. On a physical robot, equivalent information can be obtained by equipping the robot's wheels with shaft encoders, and using the shaft encoder values to update the markers.

By strictly adhering to a principled design in which the agent and environment are completely decoupled, I have constructed a simulation system which retains the essential qualities of the real world which I am investigating, while abstracting away many of the difficulties associated with using a physical robot (battery maintenance, hardware failures, etc.). Each "sensor" in the simulation has a direct analog in a physical robot, and therefore many of the lessons learned in this simulation can be expected to apply to real-world robots. Recent work with a physical robot, to be described later in this chapter, has shown this to be true; the techniques developed for use in the simulation are directly applicable to physical robots in the real world.

5.2. The agent

This section provides a detailed description of the functioning of the rabbit agent, and its constituent task-agencies. At this point, identification of which computations are performed by which task-agency is left as an exercise to the reader. The main point to notice is the way in which marker-based representations are used to enable the agent to achieve its goals more effectively and efficiently than either a pure reactive system or classical planner.

5.2.1. Task-oriented design

There are any number of aspects of the environment that the perceptual system might compute, however, only a relatively small number of this computations produce results that are useful in accomplishing a given task. The agent constructed here is to accomplish the tasks of gathering berries and avoiding predators, so it is only natural that the perceptual system perform computations that identify predators and berries. The perceptual system will also notice higher level concepts and situations, such as the occlusion of a berry, or the "dangerousness" of the predator. This section details an end-to-end

specification of the sample agent, its agencies, and its markers at the second of Marr's levels: the representations and algorithms. The way in which the agencies interact in each of the major subsystems (Perception, Memory, and Action), are described.

The agencies are those identified above: EAT, BUMP, and RUN. In addition to their various components, the primary aspect of the agencies with which we shall be concerned is their "strength." Exact values for the strengths are left to the implementation; we will refer to their values as being "high" or "low" relative to the other agencies in the system.

5.2.1.1. The perception subsystem

Each of the agencies needs a component in the perceptual system to detect the items with which it is most concerned. The input to these components is a set of early maps for the hue, saturation, and intensity of the incoming image. The EAT agency has a *food-detector*, which in this case looks for small red blobs in the image of about the right size and shape (the food items in our simulation are like "berries"). The BUMP agency has an *obstacle-detector* that works similarly to Horswill's Polly robot [33] in that it looks for obstacles by starting at the bottom of the image and scanning "upward" for the first non-ground item. The agent will notice a change in the hue from that of the green ground. The output of the obstacle-detector is a "ground line" that represents the edge of free space in the current image. Another component of the obstacle-detector determines the extent of an object by looking for vertical "jumps" in the ground line, which usually correspond to the left or right edge of an object. Finally, the RUN agency needs a *predator-detector*, which looks for yellow blobs that are just above "jagged" portions of the ground line (the predator in the simulation is yellow). The use of the ground line here is mainly to differentiate between the predator and the walls (which are also yellow), and basically amounts to looking for the predator's feet.

For the sake of the agent's survival, the predator-detector must be active at all times, even when the strength of the RUN agency is relatively low. This implies that the ground line must be computed on every incoming image. This information can be also be used by the BUMP agency for obstacle detection, which must be done whenever the agent moves (which is also nearly all the time). The food-detector, however, need only be active when the strength of the EAT agency is high, i.e., when the agent is hungry. When the agent is not hungry, those visual processing cycles can be used by other agencies. Even when the agent is hungry, if a predator is perceived to be nearby, those processing cycles may be needed to find a place to hide. This case is a processing resource conflict between the EAT and RUN agencies, as discussed above. An adversarial scheme will be used to

resolve this conflict. Finally, the obstacle extent computation need only be done when a given obstacle must be circumvented.

5.2.1.2. The memory subsystem

The limited set of markers is a resource that must be allocated carefully. Each agency has need of markers in order to carry out its task. The EAT agency marks food items, so that when they are outside the field of view, an indication of their last known location is retained. A dramatic improvement in the efficiency of the eating behavior is realized by remembering the location of only a few food items. In some situations, however, many food items may have been seen recently, so not all of them can be marked. The EAT agency therefore marks only the four nearest food items (assuming enough markers are available). EAT also tags the single nearest food item as its current target. This tag is helpful to the BUMP agency.

The BUMP agency marks obstacles, but it is not feasible to mark every object that might be considered an obstacle. However, from the task-oriented point of view, an object is an obstacle only if it is in the path from the current location to some destination point. The BUMP agency need only mark obstacles that are between the current location and locations tagged as target destinations, e.g., tagged by the EAT agency as described above. In fact, it only need mark the nearest such obstacle, since in most simple navigation problems, only the nearest obstacle is immediately relevant. The navigation algorithm to be described in the Action section below will only consider two (marked) locations, the destination point and the first obstacle on the path to that destination. This navigation algorithm is employed by the other agencies (EAT and RUN), to guide the agent to target locations.

In order to navigate effectively, non-point attributes must be associated with obstacle markers, namely, the object's width. An obstacle is thus represented by two markers, the left and right edges of the object. Note that this representation is task and viewpoint dependent, and changes as the agent moves. The specific locations of these markers change, both relative to the agent, and relative to a hypothetical world coordinate system. This further exemplifies the idea that "obstacleness" is a subjective concept, and depends on the agent's goals.

The determination that an object is an obstacle in the path to a target location is made by a BUMP agency visual routine, which is triggered by the act of tagging a particular marker as a goal location. When facing the target location, the direct path towards it appears in the image as a cone with the target location at its apex. If any of the ground line falls within this cone, this is an indication of an obstacle. If an obstacle is indicated, the ground line is traced to the left and right in search of "jumps" in the ground line, which are

indications of the “edges” of the obstacle. These edges are taken to be the marker locations that define the obstacle.

The RUN agent marks the location of a predator when one is noticed. It instructs the agent to turn around (see Action below) and run away from the predator. Once the agent is running roughly away from the predator, the RUN agency looks for potential hiding places by using a variant of the obstacle finding visual routine described above. Once a hiding place is found, the RUN agency places a marker at a location estimated to be behind it, and the marker is tagged as a target location. Once this is done, the navigation algorithm guides the agent to the hiding place.

Since the number of markers is limited, there are also conflicts over which agencies control them. The conflict resolution in these cases is primarily adversarial in nature, with the strongest agencies gaining control of the markers. The strengths of all the agencies is initially rather low, with certain events causing the strengths to increase. (Perhaps an as yet to be constructed WANDER agency could be in control when nothing else is happening.) The strength of the EAT agency increases when the agent is hungry, allowing it to mark food locations. The RUN agency’s strength increases when a predator is noticed. Whether RUN is strong enough to take markers from EAT depends on the proximity of the predator and how hungry the agent is. The BUMP agency becomes active when the an obstacle is in the path to a target destination, and its strength increases with proximity to the obstacle.

5.2.1.3. The action subsystem

Finally, we have the Action subsystem, and the components of the various agencies within it. The Action subsystem consists primarily of the navigation algorithm, which guides the agent toward a given target destination, avoiding obstacles en-route. The conflict issue here is which agency determines the current target location. First, we will briefly describe the navigation algorithm, and then discuss the resource conflicts.

The navigation algorithm takes a marker as input, which identifies the target location. This marker may have an obstacle associated with it, which is in turn another marker. The obstacle marker identifies the first obstacle en route to the target location. If there is no such obstacle, then the navigation algorithm sets the forward and turn speeds such that the agent moves directly towards the target. Obstacle markers are also tagged as being associated with the particular destination, so if the destination is ever deleted or changed, the obstacle marker is deleted as well.

If an obstacle is found, the navigation agency looks for a path around the obstacle. When an appropriate location is found, the navigation agency instantiates an *intermediate-destination* marker, which is guaranteed not to have an obstacle obstructing the path to it.

The intermediate destination is to the left or right of the obstacle, depending on which direction is estimated to be the shorter path to the destination. The navigation agency guides the agent towards the intermediate destination just as it would any other destination. Once the intermediate destination is reached, the marker on it is deleted, as is the obstacle marker. If there is more than one obstacle en-route to this destination, then subsequent obstacles will be marked by the BUMP agent after the first has been circumvented.

Conflicts between BUMP and the other agencies are avoided for the most part due to communication via the markers. The BUMP agency only interacts with the navigation algorithm by placing obstacle markers. It does not select target locations, although in some extreme situations, when a collision is imminent, the strength of the BUMP agency can be high enough to override other effector commands and direct the agent to move away from the nearby obstacle.

The conflict between EAT and RUN are resolved by the agents competing for control. The strongest agent wins, where the strengths are determined by the amount of hunger and the proximity of a predator. Some of this conflict is also alleviated by a least-commitment strategy, especially on the part of the RUN agent. It is not necessary that the agent run *directly* away from the predator, just *generally* away. Therefore, RUN can allow the EAT agent to choose any target destination that is generally away from the predator.

5.2.2. The agent implementation

As described above, the agent is implemented by two separate processes: a vision processor and a high-level control module. On each frame, the vision processor extracts the following information from the input and forwards it to the high level control module: potential berry locations, potential predator locations, the current head angle, and the *ground line*. The ground line is the boundary between ground and non-ground in the image, and represents the edge of free space. A similar technique was used by Horswill [33] in his robot, Polly. This technique was used extensively by the agent in this simulation. Figure 5-4 shows a view of the environment with the ground line highlighted

In order to extract this information, the image is converted from raw RGB values to HSV to compensate for the shading and texturing to some extent, and segmented based primarily on the hue. Potential berries are taken to be regions of red hue, which results in some false positives, since the tree bark is often of a red hue, depending on the lighting. The ground line is found by scanning from the bottom of the image for the light green hue that makes up most of the ground coloring. Darker green pixels are not taken to be ground pixels, since they are often found in the rocks. Instead, the dark green flecks in the ground area are eliminated by median filtering. Potential predator locations are found by scanning

just above the ground line for yellow pixels. This results in a number of false positives, since the walls are also yellow. Most of these false positives, as well as most of the berry false positives, are eliminated by the first stage of processing in the high-level control module. The bounding rectangles of red connected components (berries), the ground line, and the predator warning locations are forwarded to the high-level control.

Figure 5-4: The ground line

In the first stage of processing in the control module, the potential berry locations are filtered based on their size and shape to eliminate false positives. Basically, unlike tree trunks, berries are small, short, and close to the ground. The potential predator locations are also filtered to eliminate the wall areas. This is done by examining the ground line in the region of the potential predator location for “jumps” in the ground line corresponding to the predator’s feet. There are usually no such jumps in the wall areas (see Figure 5-4). The output of this first stage of processing is a set of berry and predator locations in image coordinates, along with the ground line.

The next stage converts these image based representations to egocentric world coordinates. This stage also maintains the marker based representations across multiple frames. Object locations are determined based on azimuth and elevation in the visual field, by assuming that the ground plane is flat, and all objects are on the ground. The system attempts to keep the four closest berries and any nearby predator marked at all times. The markers are maintained by using dead-reckoning based on the most recent velocity commands to estimate the expected new position of the object. If the expected position is in the image, then the correspondent is found in the image if possible. The output of this

stage is a set of markers on berries and the predator. This set of markers forms the "world model" of the agent, and represents an estimate of the current state of the world.

5.2.2.1. Berry gathering behavior

The next stage of processing is to implement a strategy for survival by determining an action to take based on the current situation. The strategy for the agent is to head for the nearest berry, which may not be currently visible, due to occlusion or the limited field of view. The agent must carefully follow the "perception overrides memory" doctrine, since due to noise in the image caused by severe color quantization some erroneous markers are created. Markers also occasionally drift due to imperfections in the dead-reckoning procedure. The problems with erroneous markers are overcome by having the agent drop any markers that are supposed to be in the field of view, but don't have any perception corresponding to them. So for example, if a berry marker is maintained to the left of the agent, the agent turns to the left to look at it. If no berry is seen to the left, the marker is dropped. This behavior serves to allow the agent to use the markers that are accurate, without being led too far astray by incorrect markers.

The strategy described above must be augmented to notice the occlusion of marked objects and behave appropriately. The technique we used is similar to Horswill's obstacle avoidance routine [33] in that it relies only on being able to segment the ground from non-ground (i.e., obstacles). To determine if a marked object is occluded, points on the ground line are converted to egocentric 3D coordinates and compared with coordinates of the marker. If the ground boundary points in the direction of the marked object are strictly nearer than the marked object, then the object is assumed to be occluded. Occluded markers are not dropped, but now we need a strategy for navigating around the obstacle. The original behavior, which was to go directly towards the nearest marker, obviously will not work for occluded markers. We have implemented the navigation algorithm described in section 3.3.1.3 using the target berry's location as the destination marker.

To determine the location of the obstacle marker, the ground/non-ground boundary in the region of the goal marker is analyzed for sharp jumps that indicate the edges of the obstacle. Simple geometric reasoning is used to determine the shortest distance around the obstacle, and the obstacle's edge is marked. An intermediate-destination marker is then instantiated, and the agent is directed to move towards the intermediate destination, thereby circumventing the obstacle.

5.2.2.2. Predator avoidance behavior

Running concurrently with the berry gathering behavior is the predator avoidance behavior. When a predator is noticed, and deemed sufficiently "dangerous" the agent exe-

cutes a “run away and hide” behavior. This behavior has the following stages: look for a place to run, run away, look for a place to hide, and hide. The current stage is determined by the state of the world as encoded in the agent’s markers, rather than by a traditional program, with a program counter stepping through the stages. This way the agent can revert to previous stages or abandon the procedure at any time without using any special mechanisms.

The first step is to actually notice the predator. In order to accomplish this, the agent must “look around” much more often than is necessary for simply gathering berries, since the predator is quite fast, and may approach at any time, from any direction. The look-around behavior is accomplished by establishing “pseudo-markers,” at points around the agent’s body, with duty cycle timers set to request the agent to look at them at appropriate intervals. These are termed “pseudo-markers” since they are not actually on any object, and they are not updated in the usual way—they are at a fixed location with respect to the egocentric coordinate system.

An alternative would be to provide the agent with a “compass,” which would enable these markers to be place at fixed compass points. Intuitively, it seems this alternative may work better, but the implemented option works adequately, and it didn’t seem necessary to confound the results by adding an additional sensor. Another alternative would be to update these markers in the usual way, but this would take time, be subject to drift, and be unlikely to significantly improve performance.

The agent’s field of view is about 75 degrees, so these pseudo-markers were placed at 60 degree increments around the agent’s body, which covered the entire 360 range, and allowed for some overlap (see Figure 5-5). No pseudo-marker was placed directly ahead, since the agent usually looked directly forward anyway.

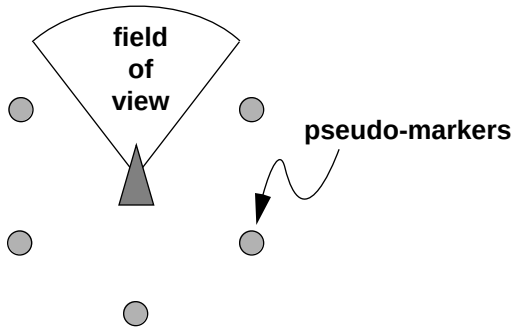


Figure 5-5: Pseudo-marker configuration

The duty-cycle timers on these pseudo-markers were set (empirically) such that the agent struck a balance between looking where it was going (forward), and looking around enough that the predator was usually noticed when it approached.

Once the predator is noticed, a determination of the "dangerousness" of the predator needs to be made. The predator detection visual routine uses color and the ground line shape to basically look for the predator's feet. Unfortunately, this requires that the predator be rather close in order to detect it, so if the agent detects the predator at all, it is probably dangerous. However, the predator detector also produced a number of false positives, when rocks against the wall caused a section of the wall to look like "feet." Such a situation is rather transient, however, so if the agent keeps moving, this false predator will disappear. So rather than running away immediately upon seeing a predator, the agent looks for the predator in a sequence of frames, in order to filter out some of these false positives, and keep the agent from wasting energy by running away from something that is not there.

Once a predator has been seen for more than some threshold number of consecutive frames, the predator is deemed to be "dangerous," and the agent begins looking for a direction in which to run. Depending on the relative location of the predator, the agent may prefer to run in certain directions. Naturally, running "away from" the predator is a good choice. However, since the predator will run directly towards the agent until it catches the agent or runs into an obstacle, running laterally to the predator is also a good strategy, since this is more likely to cause an obstacle to be positioned between the agent and predator. Also, if the agent decides it wants to run in the direction it is currently facing, it can do so immediately, whereas if the agent decides to run in the opposite direction, if must take the time to turn around first, and it may be too late by the time it gets turned around.

The agent considers each of six possible directions to search for a place to run. These directions are, like the pseudo-markers discussed above, in 60 degree increments around the agent, again taking into account the agent's field of view to achieve complete coverage of the range of directions. Each of these potential directions to run is scored for its desirability, taking into account the relative predator location and direction the agent is facing as discussed above. Each of the six directions gets points for being away from the predator, being lateral to the predator, and being in the direction the agent is facing. Once each of the directions is scored, the agent picks the direction with the highest score and looks in that direction for a place to run. If it fails to find a place to run, it looks in the next most desirable direction, and so on, until it finds a place to run. If all the directions are tried, the agent starts the process over again.

A "place to run" is some open space in which the agent can move in a straight line for a sufficiently large distance. It can be found by analyzing the ground line, which is at the boundary of free space. Consider the scene shown in Figure 5-6a, in which the ground

line is highlighted. Some of the points on this line are too close, so there isn't room to run in that direction (see Figure 5-6b).

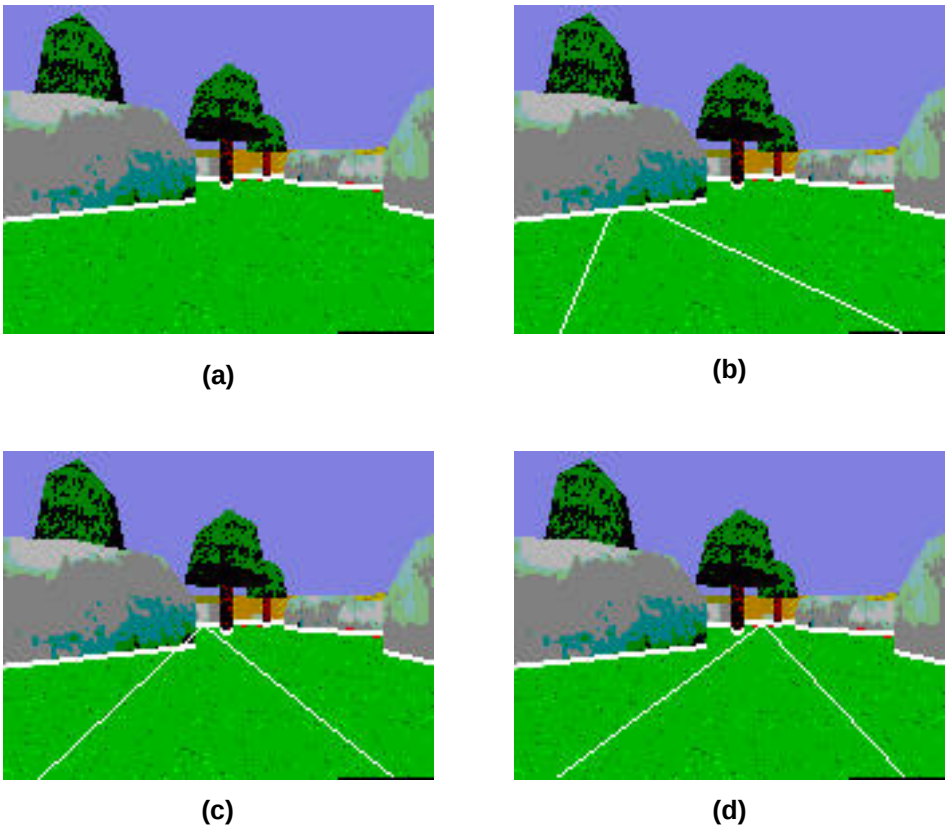


Figure 5-6: Finding a place to run

Other points may be far enough away, but there is not a clear path towards them. This situation is determined by projecting the path towards the given point back into the image, and seeing whether this projection intersects the ground line (see Figure 5-6c) If the ground line point is both far enough away and has a clear path, the point is marked as a place to run (as in Figure 5-6d). This procedure is identical to that for determining if there is an obstacle to any potential destination point. If no such point is found in the given viewing direction, then the next most desirable direction to run is tried.

Once a place to run is found, it is marked, and the agent moves towards the marker. The agent also accelerates to a speed that is slightly higher than that of the predator (otherwise it doesn't have a chance). The drawbacks of this acceleration are that the agent uses more energy to go faster, and any collisions with an obstacle will "hurt" more, since the penalty for collisions is proportional to the speed of the agent.

After running for some (empirically determined) length of time to put some distance between itself and the predator, the agent begins to look for a place to hide. By

"hide," I mean that the agent wants to run in such a way that it positions an obstacle between itself and the predator. It identifies such an obstacle by looking for its left and right edges in the ground line while looking in the forward direction. Once such a obstacle is identified, its edges are marked, and the agent computes where it needs to go in order to get on the opposite side of the obstacle from the predator, and this location is also marked. This goal location is then tested to see if the agent can proceed directly towards it, or if an intermediate destination needs to be established, as in the usual obstacle avoidance algorithm.

It is important to emphasize that the use of markers is central to the functioning of this hiding behavior. The current action to take is completely determined by the current state of the markers (their types and locations). The marker maintenance system is responsible for assuring that the marker state is a sufficiently useful and accurate estimate of the world state. The progression of the agent through the steps of the run-away-and-hide behavior is caused by a progression of marker configurations, which corresponds to a set of world states (where a world state includes both the observable state of the physical world and the internal state of the agent). Transitions between states are caused by the allocation, update, and deallocation of markers. Basing the actions to take on the marker states, rather than a fixed sequence of plan steps insures that the action taken is appropriate for the current world state, rather than taking a (possibly inappropriate) action simply because it is the "next step" in the plan. This is implicit sequencing is critical in a dynamic environment. The fact that the markers are continually updated and verified with the world state provides reactivity to unexpected events in the environment, yet these markers are also labeled with symbols that are meaningful steps in a plan. Here we see directly the potential for merging the reactive and classical planning paradigms through the use of marker-based representation. The plan above was hard-coded, but I believe the concepts are amenable to use with an automated planner as well.

To further illustrate the points in the previous paragraph, I will step though an actual example of the working predator avoidance behavior, pointing out the marker states and labels, and how they drive the progression of the system though the succession of plan steps. In Figure 5-7, we see a situation in which the predator is approaching the agent. (Recall that the agent does not have access to this overhead view.)

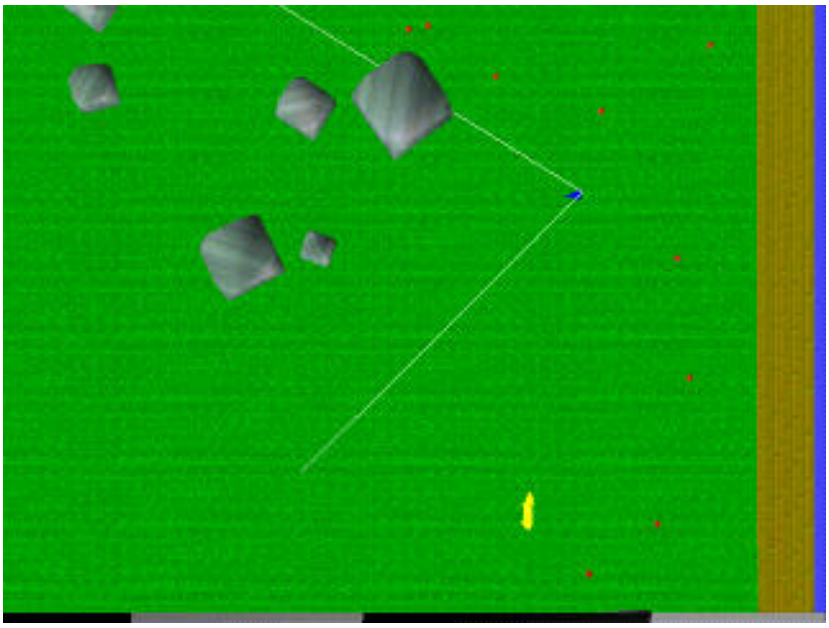
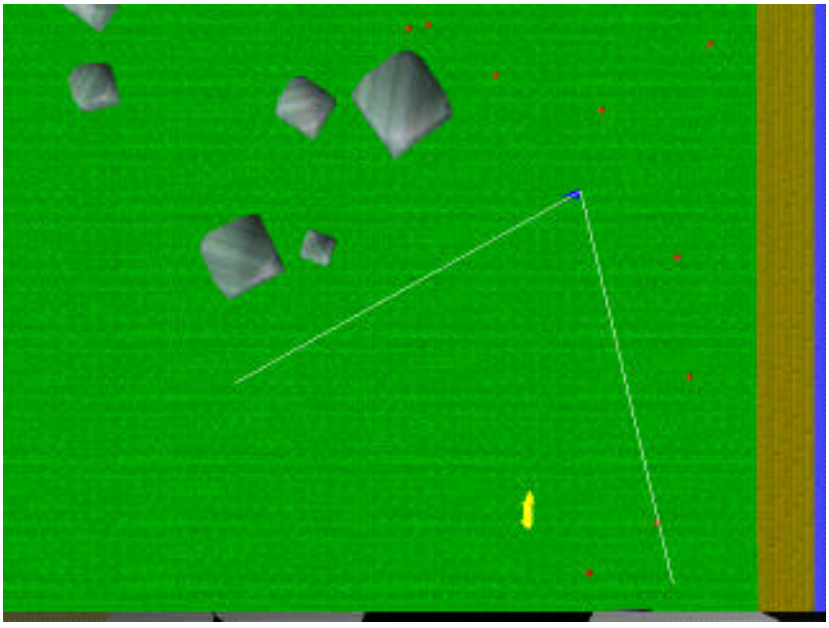


Figure 5-7: An impending predator encounter

The agent (the small blue triangle) is looking off to the left, with the yellow predator approaching from below. The marker maintenance system attaches a duty-cycle countdown timer to each of the markers, and requests that the head be turned in the direction of the marker with the lowest timer value. Whenever an object is seen in the current field of view, the timer is reset. The pseudo-markers described previously which make the agent "look around" have timers as well, and they may also cause a request that the agent look in their direction. In the situation above, the timer associated with the pseudo-marker is about to "go off" and cause the agent to look towards the predator. When the agent looking towards the predator, the predator is noticed, and a predator marker is placed on the estimated location of the predator, depicted as a yellow circle in Figure 5-8b. The predator marker is given an initial timer value of zero, causing the agent to look directly at the predator's location immediately (note the change in the direction of the agent's gaze between Figures 5-8a and 5-8b). If the predator is seen again when looking directly at it,

the predator marker is augmented with the label “dangerous,” which causes the agent to move to the next step in the sequence.

(a)



(b)

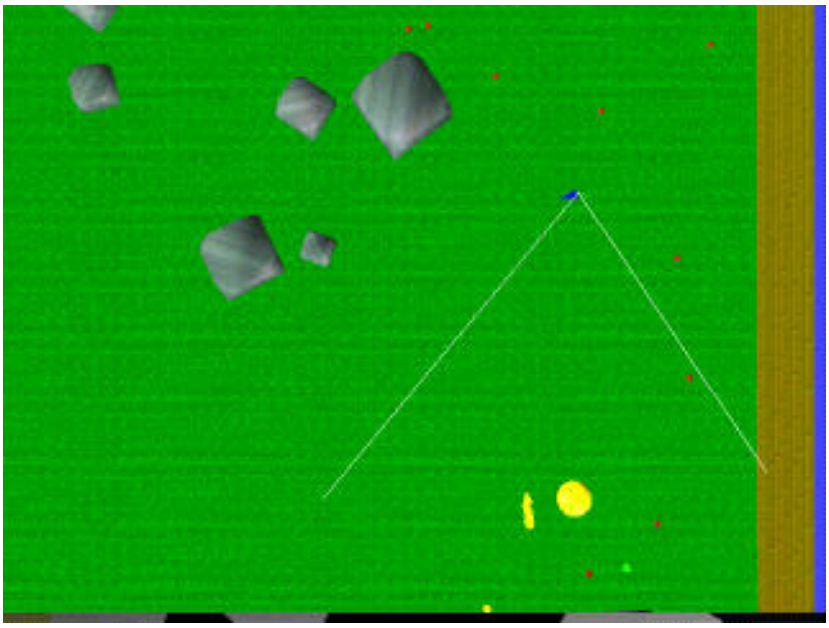


Figure 5-8: Noticing the predator’s approach

Once the predator marker is upgraded to “dangerous” status (as indicated by the red circle in Figure 5-9) the agent begins to look for a place to run. The six possible directions to search are evaluated as described, and the preferred direction to run is marked

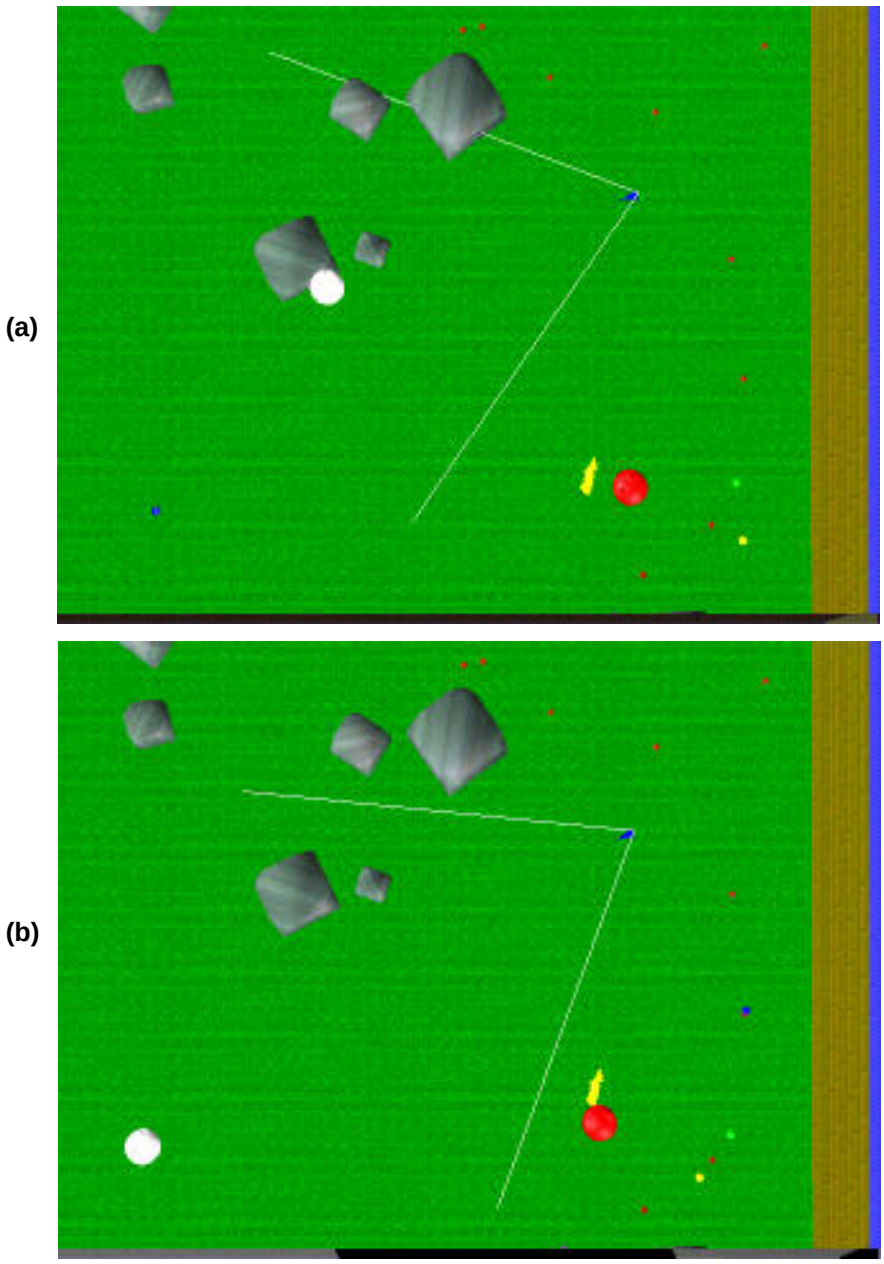


Figure 5-9: Looking for (a) and finding (b) a place to run

with a “search” marker (white circle in Figure 5-9a), again with a duty-cycle timer of zero, causing the agent to look directly at it immediately. Recall that any command given by the agent may take some nondeterministic amount of time before it is executed. After the agent issues the command to look at the search marker, it must therefore wait until that command is actually executed before actually searching in the ground line for a place to

run. The execution of the command is indicated by the head angle sensor, which will show the agent to be looking at the marker after the command has taken effect. The agent can then search the ground line and establish a “run” marker in an open area (white marker in Figure 5-9b).

While the agent is running it needs to occasionally look back at the predator to verify its location (especially since it is moving), and also to see if the predator has by chance become occluded, in which case the agent is hidden already. This is accomplished by the same timer mechanisms as described above. The timer associated with the predator is relatively short, causing the agent to make frequent glances back towards the predator (see Figure 5-10). An even shorter timer is placed on the run location (white marker in Figure 5-10) so that the agent is usually looking where it is going, in order to avoid collisions.

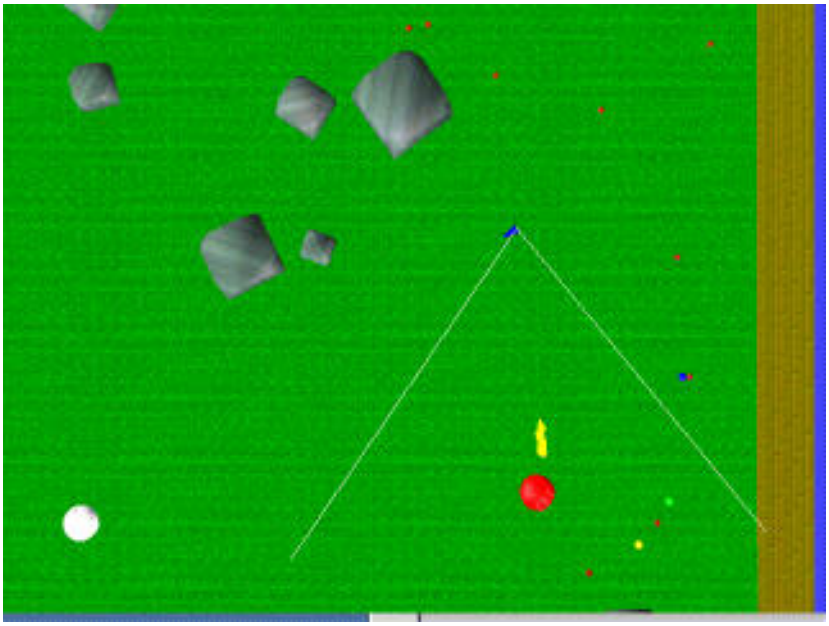


Figure 5-10: Verifying the predator location

Once the agent is moving (i.e., running away from the predator) it has the opportunity to search for a hiding place, since the agent is less vulnerable when it’s moving. While the agent is looking at the run marker, it searches the ground line for a hiding place. When one is found, the left and right edges are marked (black markers in Figure 5-11) and a goal location is chosen on the opposite side of the obstacle from the predator (blue marker in Figure 5-11). If necessary, the agent also establishes an intermediate goal marker (not shown in the figures), and proceeds towards this marker instead. The agent verifies the

locations of these markers as it approaches the hiding place and the predator advances, and the marker locations are readjusted as necessary (see Figure 5-11b).

(a)

(b)

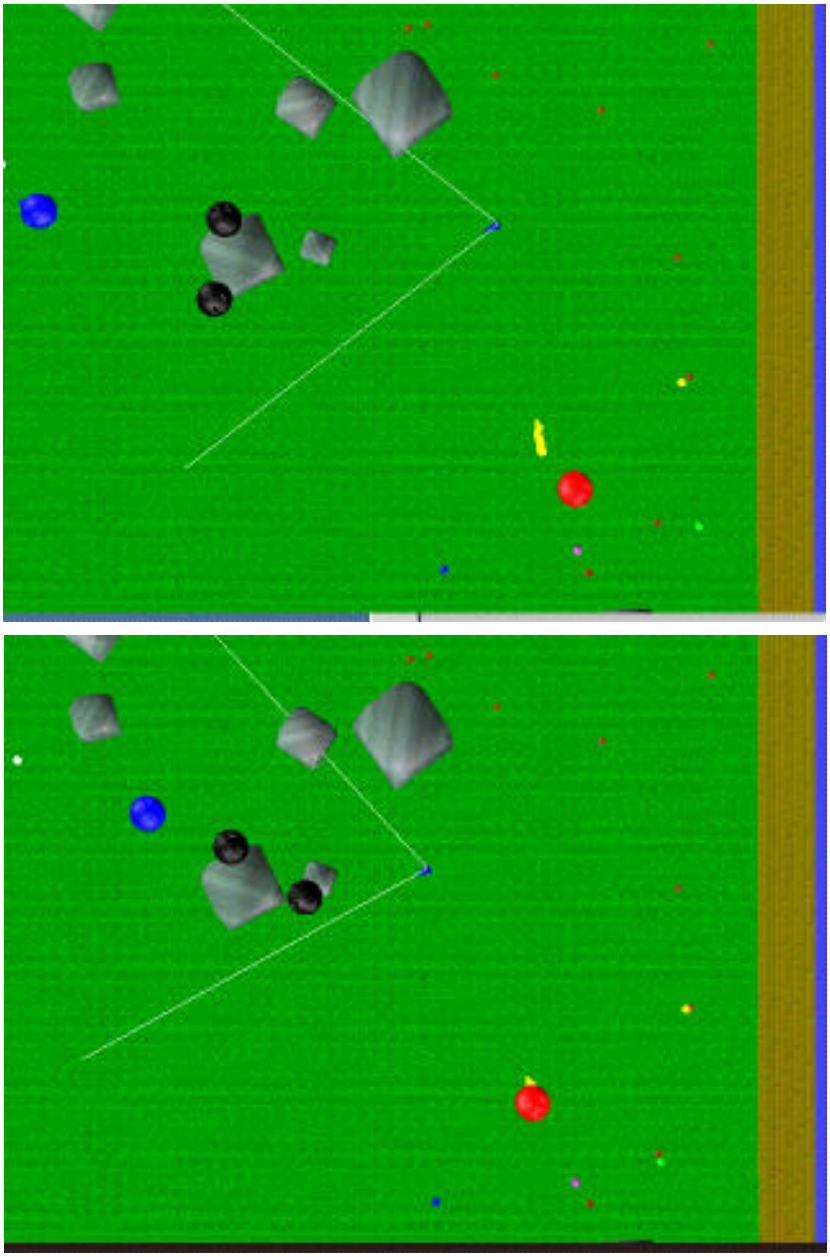


Figure 5-11: Finding and tracking a hiding place

The predator marker is updated as well by making occasional backward glances in the direction of the predator marker (Figure 5-12).

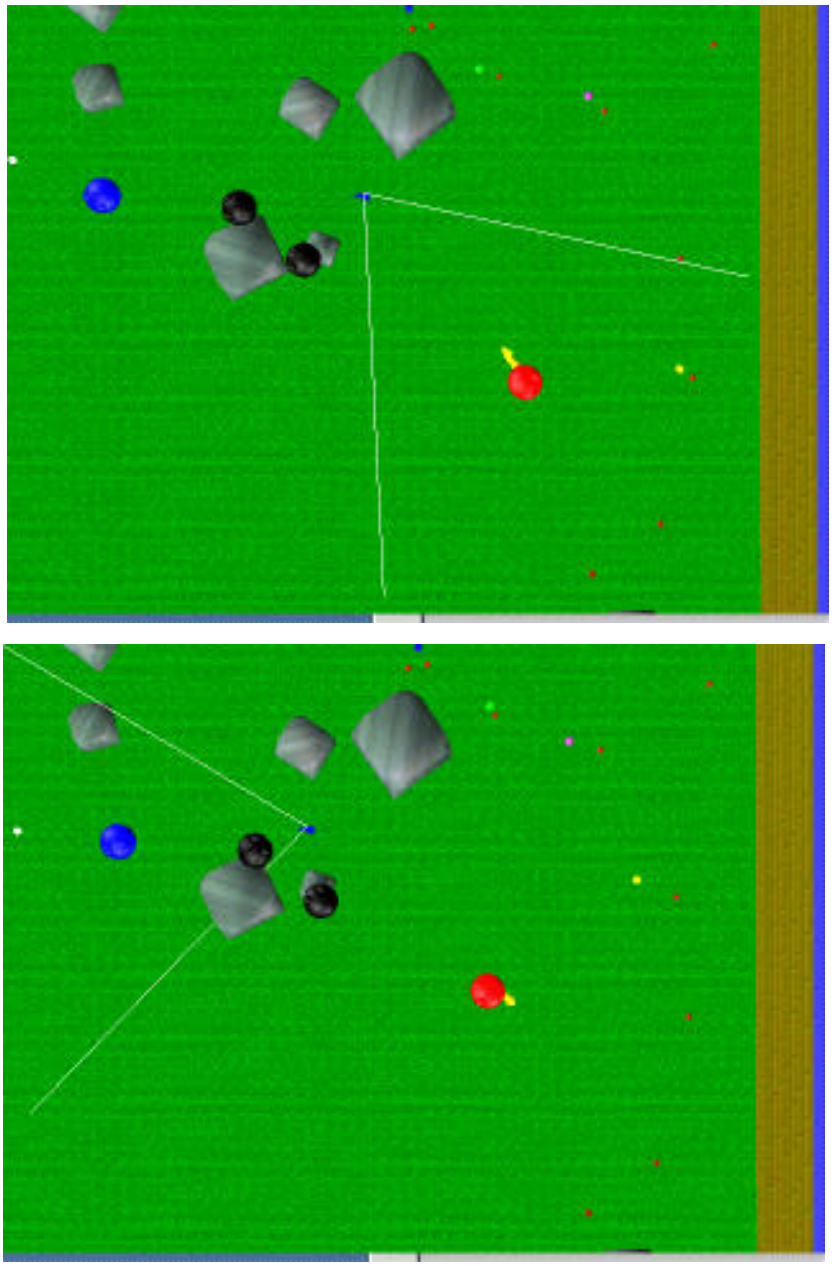
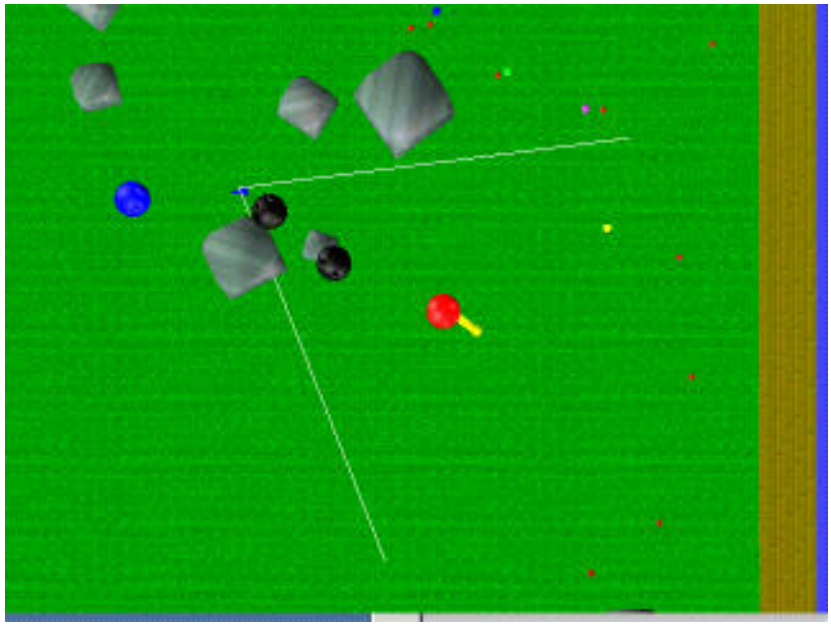


Figure 5-12: Monitoring the predator location

Eventually, if all goes well, the agent will look back towards the predator and notice that it has become occluded by the hiding place obstacle, as shown in Figure 5-13a. Once this occurs, the hiding place and obstacle markers can be deleted, since the goal they

subserve has been achieved. The predator marker is retained, however, as an estimate of the predator's last known location (see Figure 5-13b).

(a)



(b)

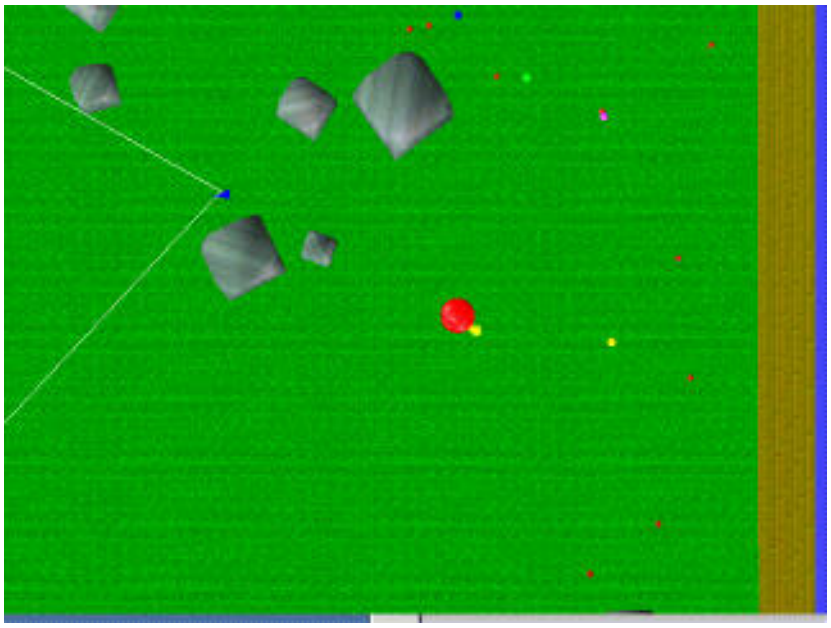


Figure 5-13: Hiding successfully

The agent will delete the predator marker when it believes (as indicated by its markers) that the predator is both occluded, and sufficiently far away. Having achieved the occlusion goal, the agent tries to further distance itself from the predator, by actually establishing a search marker, and thereby reverting to the beginning of the run away and hide behavior. This time, however, the occlusion of the predator by the obstacle will be

noted, and this, combined with the increasing distance from the predator, will cause the predator marker to be deleted. However, if the agent was unsuccessful in its attempt to hide, the cycle will be repeated, with the agent looking for a new place to hide.

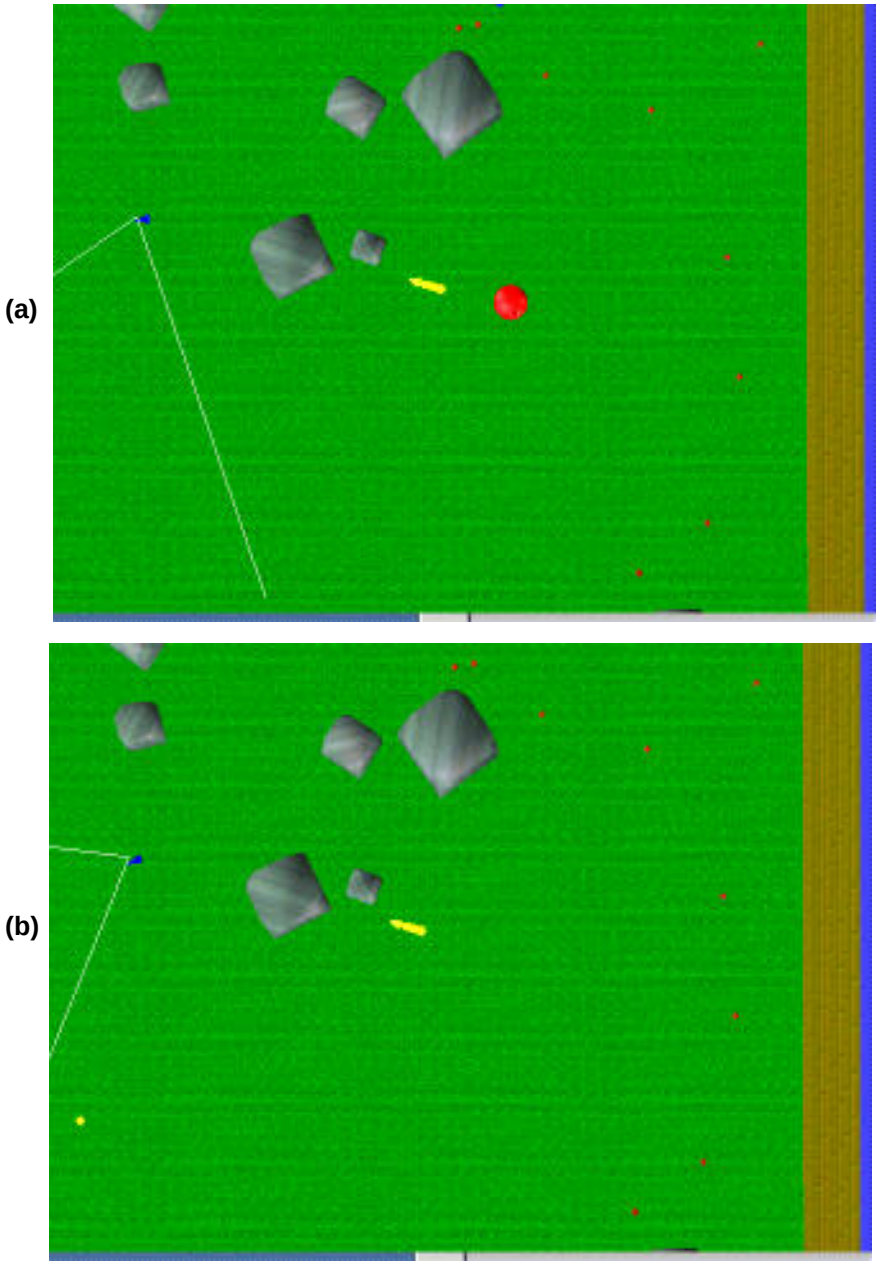


Figure 5-14: Increasing the gap and getting away

Note how the use of markers was central to the operation of each step in the preceding "routine." While is one cannot claim that markers are the only means for implementing this routine, nor even necessarily the best or most efficient, it contrasts well with the alternatives of classical and reactive planning. A classical planner would have trouble

with the dynamic world and incomplete information available to the agent. A reactive planning system, being memoryless, would not be able to deal with the limited field of view and occlusion, since it can't keep sensor contact with all of the information it needs simultaneously (although Chapman's system would work quite nicely if it were given the overhead viewpoint shown in the figures of this section).

Moreover, I do not claim that this routine is the best way to use markers to accomplish this task, only that it is a good way, i.e., it is more successful than a memoryless approach, as will be demonstrated quantitatively in the following section. One can think of many ways to enhance the performance of the system by using additional markers or making better use of the markers used here. However, these enhancements would still be using markers, which only further supports my main thesis.

5.3. Experimental results

The simulation system can easily be instrumented to measure the performance of the agent. In describing the "big shell game," I argued that the use of memory will enable the agent to more efficiently gather items. However, that is a mathematical argument in an idealized situation, in which the agent's memory is perfect and complete. The simulation provides the opportunity to make measurements of a more realistic situation, in which memory is limited and potentially error prone, and occlusion and a limited field of view may be factors.

In order to measure the efficiency of the various agents, I have instrumented the simulation to compute the average "inter-berry distance." As the agent is gathering berries, it must travel some distance to its intended target berry. We can say that one agent is more efficient than another if over the long run, the more efficient agent travels less distance in gathering berries, and that the average distance between consecutive pairs of berries would be smaller for the more efficient agent.

Below are comparisons of three different agents in two different conditions. In the first condition, the environment is an open field with no obstacles, and in the second condition obstacles are added. The three agents compared are a completely reactive agent, an agent with four[1] markers to place on berries, and an agent with four markers and a "neck." The strategy for reactive agent is to always move towards the nearest berry that it "knows about," and the reactive agent only knows about berries in the *absolute* field of view. The strategy for the "four marker" agent is identical—to move to the nearest berry it knows about. However, the agent with markers may potentially know about more berries than the reactive agent, since it may have a larger effective field of view. The agent with a "neck" is identical to the four marker agent, except that rather than always looking directly forward in the direction of travel as does the four marker agent, the "neck" agent may look in any direction, independent of the direction of travel. In the berry gathering task, the neck agent uses this ability to look directly at the target berry, and also to glance around from time to time to get a more complete picture of its surroundings. Redirecting the absolute field of view in this way, in conjunction with memory, may dramatically increase the effective field of view. The "looking around" behavior is implemented by using the "pseudo-markers," see section 5.2.2.2, Figure 5-5. Timers are placed on these pseudo-markers such that the agent glances around at regular intervals. A much shorter timer is placed on the goal

1. A discussion of the choice of *four* markers, as opposed to some other number is postponed until later in this chapter.

berry, so that the agent is usually looking at the goal location. Figure 5-15 shows the comparison of the performance of these agents in an open field.

Each agent is allowed to collect 100 berries in the field. The field initially contained 50 berries, and as each berry is "eaten" a new one is added at a random location in the field. Over a given run of 100 berries, the mean and standard deviation of the inter-berry distance for that run is computed. The mean and standard deviation are used to construct 95% and 99% confidence intervals for the actual mean inter-berry distance for the agent, assuming that the inter-berry distance is roughly normally distributed. The blue error bars in the graph indicate the 95% confidence interval, and the green bars indicate the 99% confidence interval. Three different runs are made for each agent.

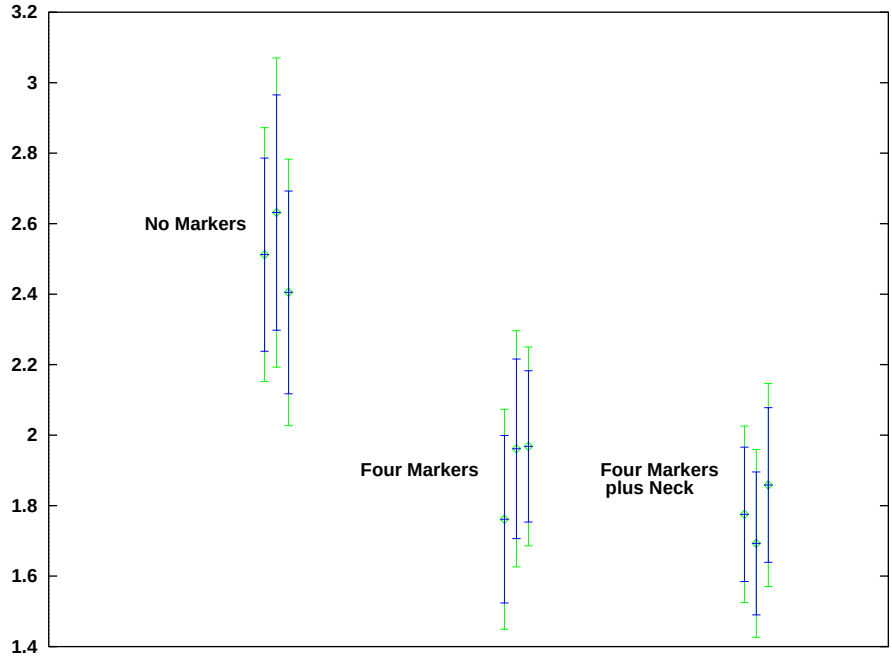


Figure 5-15: Average inter-berry distance in an open field

We can see in Figure 5-15 that the inter-berry distance of the "no marker" agent is somewhat higher than that of the other two agents, indicating poorer performance for the memoryless agent. The other two agents performed roughly the same in this open field condition.

Figure 5-16 shows the results of another experiment, which is identical to the experiment shown in Figure 5-15, except that the field contained a number of obstacles. Performance for all of the agents degraded slightly in this condition as opposed to the open field condition (note the change in scale on the vertical axis). Performance is more variable in this condition as well, as indicated by the wider error bars. In addition, the per-

formance of the four marker agent (without the neck) has degraded relative to the neck agent in the presence of obstacles.

The poorer performance of the memoryless agent in both experiments is a result of the agent not taking full advantage of areas in with there is a cluster of several berries. As the agent approaches a cluster, many of the other nearby berries go out the absolute field of view. This is illustrated in Figure 5-17, in which the gray triangle indicates the agent with its absolute field of view indicated by the conical shape. The memoryless agent may miss these nearby berries completely, and instead move towards a berry directly in front of it, but relatively much further away. The agent with markers will remember the approximate locations of these nearby berries, and can turn towards them appropriately. From watching the agents perform, I can say subjectively that this occurs frequently—this observation is borne out in the statistics illustrated in Figures 5-15 and 5-16.

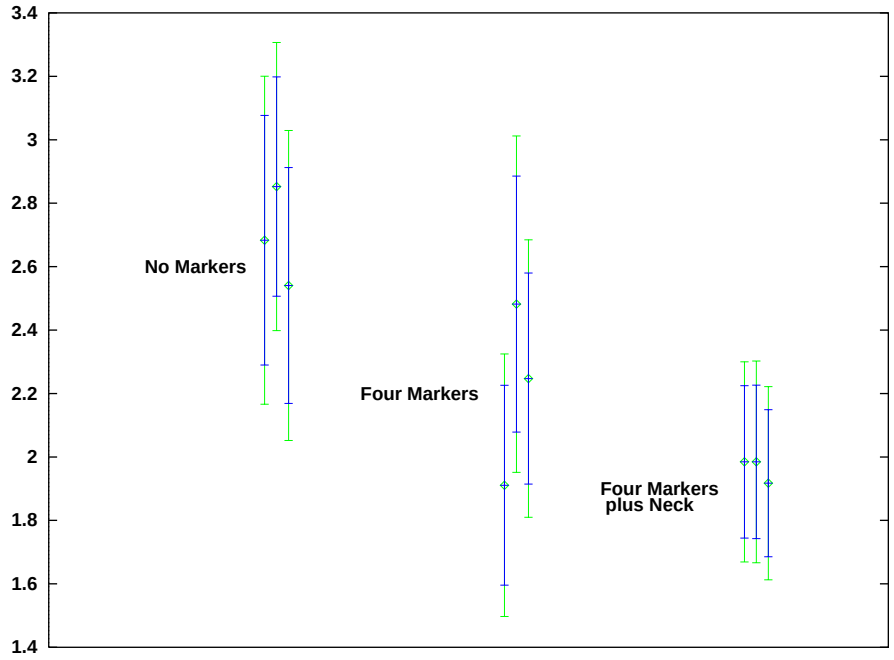


Figure 5-16: Average inter-berry distance in a field with obstacles

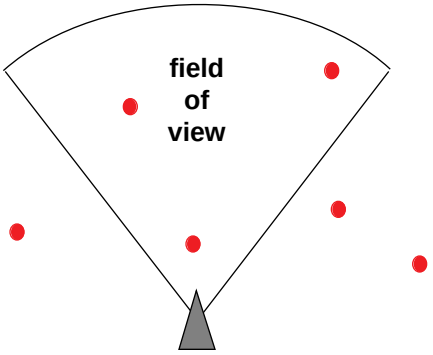


Figure 5-17: Nearby berries passing out of the field of view

The difference in performance between the “neck” and “no-neck” versions of the marker using agents can also be explained by observing that the agent which “looks around” more may potentially know more about its surroundings, and in this specific case, know about more nearby berry locations. In the open field case, this potential advantage didn’t significantly improve the measurable performance of the agent with the neck, since in the absence of occlusions, the agent need not look around much to get a good idea of its surroundings.

However, in the case with obstacles, occlusions could hide information that can later be obtained by looking around. Consider the scenario in Figure 5-18, in which at time 1 (in blue) the agent can only see the single berry on the far right, since the nearer berry is occluded by a rock. Later, however, in the course of pursuing the further berry, the nearby berry becomes unoccluded at time 2 (in green). The agent which always looks forward will not see the nearby berry until after it goes all the way to the further berry, if at all. The agent that simply glances around occasionally can notice it, however, and can take a shorter path to obtain the two berries. This kind of scenario is responsible for the improved performance of the agent with a neck, which is evident in Figure 5-16.

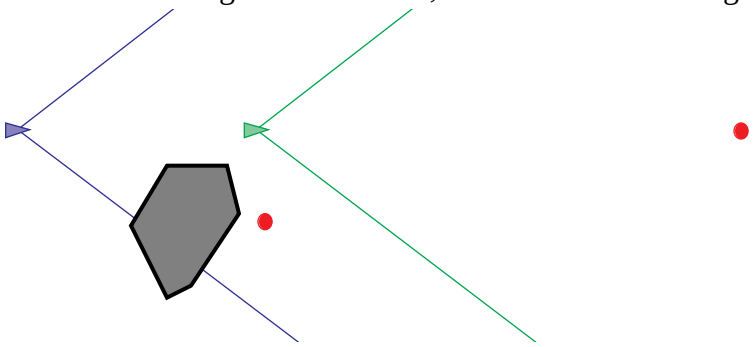


Figure 5-18: Occlusion thwarts the no-neck

A real rabbit agent might consider a shorter path to be more efficient under an energy-expended metric. To model this efficiency aspect, we can give the agent an “energy” budget, in which eating berries increases the agent’s energy, and moving around decreases it. Under this model an agent is considered successful if its behavior patterns enable it to maintain a positive energy budget (i.e., an energy level of zero constitutes death). We can also add a “resting” energy consumption rate, so that the uninteresting strategy of sitting in one place is unsuccessful.

This is done with each of the three agents discussed above, plus a “null” agent which simply sat in one place. The field had obstacles as in the second experiment described above. The agents are given an initial energy value of 200 units, and berries are worth 20 units each. Energy decreased at a rate of 5 energy units per unit of distance trav-

elled, and there is a resting energy consumption rate of 0.2 units per simulation "tic." The energy level of the agents were recorded over time (in simulation tics), as shown in Figure 5-19.

The null agent just sat in one place, and as expected, its energy depleted in 1000 tics (1000 tics x 0.2 energy units per tic = 200 energy units). The memoryless rabbit fared slightly better: as it moves, it depletes energy faster than the null agent, but with each berry eaten, the energy level jumps up by 20. This agent managed to survive for more than 1200 simulation tics. However, the agents with memory managed to survive indefinitely (the simulation was stopped after 3000 tics). The agent without the neck held its energy value around 200, and the energy level of agent with the neck steadily increased.

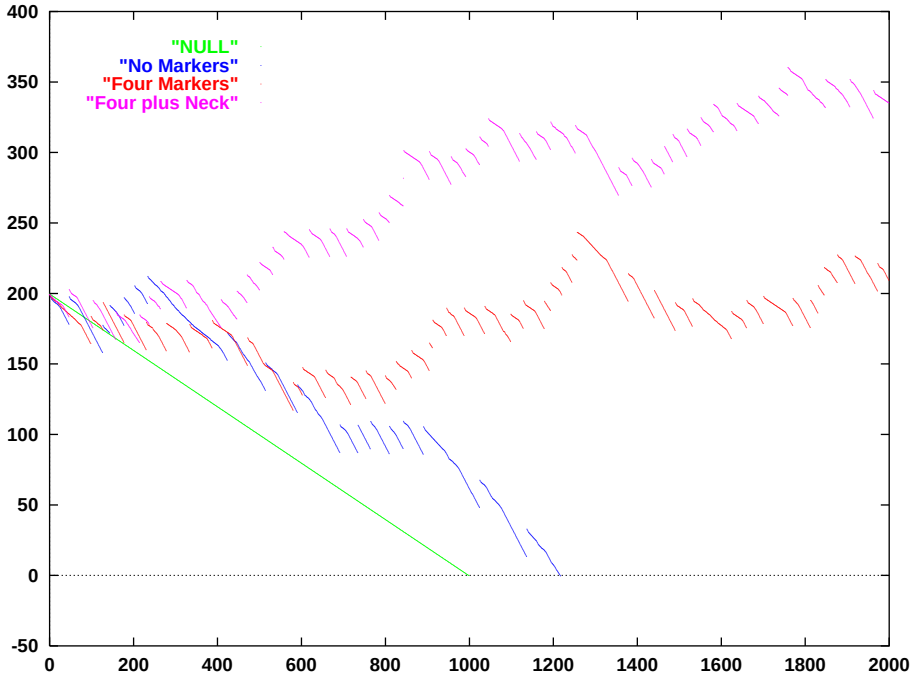


Figure 5-19: Energy time series for 4 different agent types

The reader may wonder if there are other parameter values (i.e., energy depletion rates) at which the memoryless rabbit performs better than the marker-maintaining rabbits. However, for this particular task, and this set of parameters, I have not found any such (reasonable) parameter values, and do not believe any exist. Of course, we could enable the memoryless rabbit to outperform the other agents with a different set of parameters, in which we might, say, penalize for the use of memory (e.g., each marker costs one energy unit per simulation tic). But by manipulating the energy parameters used in this experiment (initial energy, resting energy, moving energy, and berry energy), the marker based rabbits will always perform better. The problem space in this scenario is well characterized—some things make the energy go up, and others make it go down, and there are no

complicated interactions among the various parameters. (Note that we could actually make the null agent perform better by setting the resting energy rate to zero, and the moving energy rate quite high, but this case is not very interesting. We could also make the berry energy negative, but that would just be unreasonable.) By say, doubling the berry energy, all of the agents would perform better, but the relative ordering would stay the same. It is at the point where some agents succeed and others fail that the experiment is interesting.

In more complicated scenarios, there may be some overhead associated with the additional marker based behavior that decreases the performance of the more complex agent for some parameter settings. I will discuss this further when dealing with obstacle and predator avoidance.

We can see that the simulation developed for this research enables us to measure and compare the performance of various agents on the berry gathering task. In the remainder of this chapter, I will use this methodology to compare the performance of agents on a variety of tasks. I will demonstrate that an agent can acquire and maintain useful representations which expand the effective field of view—even in a dynamic and uncertain 3D environment—and that the use of these representations leads to a measurable increase in performance.

5.3.1. The usefulness of multiple markers

The presentation of the previous experiments begs the question “how did you decide on four markers?” and furthermore “how many markers are necessary?” Of course the answer to the latter question is that it depends on the problem. Markers are meant to represent useful predicates, and whether a predicate is useful can only be evaluated in the context of a task. A predicate is only useful if the agent can construct a plan in which the predicate is a precondition for some action which accomplishes a desired task. The answer to the former question is that I observed the agent and chose to use the number of markers at which I subjectively decided the agent had “enough,” in that the agent did not appear to be missing berries due to a lack of markers. Naturally, it is worthwhile to actually analyze the situation more completely, and to measure the performance of agents with different numbers of markers, which is the purpose of this section.

In our rabbit agent, the planner is not very sophisticated, in that the “plans” it constructs usually consist of a single “step” which is to get the nearest berry (note that multiple steps are needed if there is an obstacle in the direct path to the berry). Also, since all the agents have a limited absolute field of view, the “go to berry” step must be preceded by a “find nearest berry” step. One way to do this would be to mark all the berries visible

from the current location, and then scan through the marker-based representation to find the nearest one. This "find out everything first" approach is the classical decomposition of perception and action, and would require a lot of markers, and a lot of searching to find all of the berries. Fortunately, it's not necessary.

In order to find the closest berry, it's only necessary to represent the nearest berry seen *so far* in a scan of the surroundings. If we are willing to stop and do a scan of the environment before moving, then we can choose the nearest berry using only one marker. We cannot take this approach with zero markers, since when looking to the left, it is necessary to remember what was seen on the right, so that a comparison can be made between the nearest berry on the left and the nearest berry on the right. The problem with this "scan before moving" approach is that in the presence of obstacles, one may not see the nearest berry on a given scan because it is occluded, yet a little later, the berry becomes "unoccluded," as is shown in Figure 5-18. To deal with this case, the agent would either need a sophisticated understanding of occlusion, so as to know when to look at recently unoccluded regions, or the agent must stop and scan on every small step.

Another approach, which is the one I adopted in the construction of this agent, is to scan "all the time," and move "all the time." In this constant-scan approach, the agent always moves towards the closest berry it has marked, and simultaneously with moving, the agent scans the environment to find other nearby berries. The use of a marker on the current destination allows the agent to move towards the current goal while looking in a different direction. The constant-scan admits the possibility of going in the "wrong" direction for a little while, toward what is perhaps the second nearest berry, until the scan comes around to the region in which the actual nearest berry can be seen. This approach can again be used with a single marker placed on the current goal, which is moved if another berry is seen to be nearer. The scanning behavior is implemented by using the set of pseudo-markers described earlier (see Figure 5-5). The duty-cycle of these markers is set so that the agent looks around enough (as determined empirically), yet the pseudo-marker duty-cycles are longer than the duty-cycle of the current goal marker, so that the agent is usually looking at the current goal, in order to accurately maintain the goal marker.

If only one marker is used in the constant-scan approach, consider what happens immediately after the agent eats a berry. The marker was on the eaten berry, and is now available to be placed somewhere else. It is immediately placed on the nearest berry in the current field of view, which may or may not be the actual nearest berry. The agent immediately moves towards this marker, and if it is not on the actual the nearest berry, the agent moves in the wrong direction until the scan comes around to the area with the actual near-

est berry. One way around this is to stop and wait for a complete scan before moving, but another is to have a second marker, placed on the second nearest berry. Once the first nearest berry is eaten, the second nearest berry now becomes the first nearest, and the agent, having marked this berry, can turn immediately towards it. It is as if the agent still "sees" this second nearest berry, since it is still inside the agent's effective field of view.

This discussion is by way of explaining the data in Figure 5-20, which shows a comparison of the performance of seven different agents. The vertical axis is mean distance between berries as in Figures 5-15 and 5-16 (lower is better). The leftmost cluster of data points is from an agent with one marker, but this agent does not scan at all, i.e., the head is fixed pointing forward relative to the body. The performance of this agent is nearly identical to that of a completely reactive agent. This is to be expected, since its behavior is also nearly the same, in that it always moves towards the nearest berry in the current field of view; it just also happens to "mark" the berry.

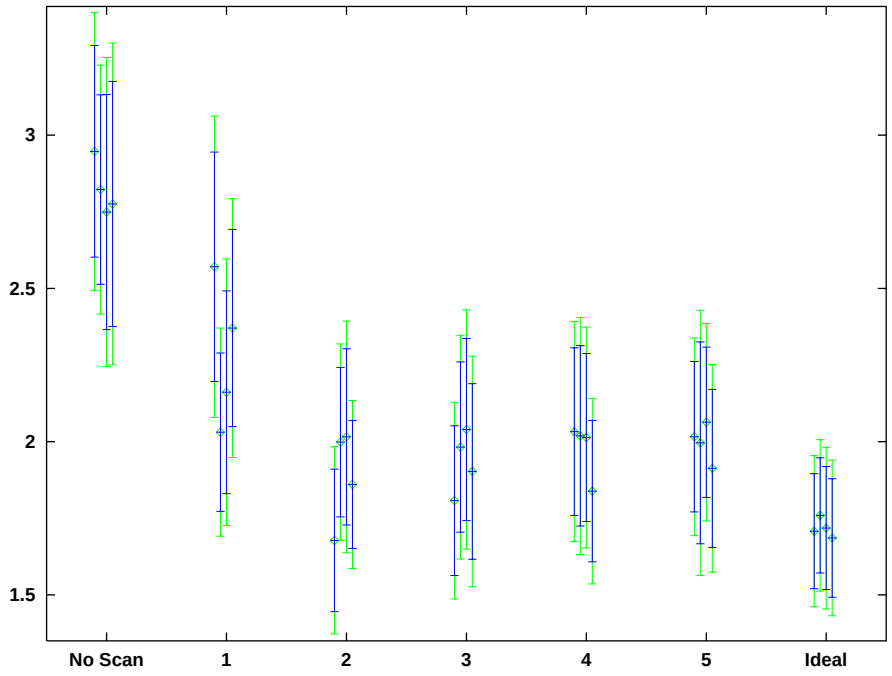


Figure 5-20: Increasing the performance via marker use

The next five agents have non-fixed necks, and use 1, 2, 3, 4, or 5 markers in the constant-scan behavior. Adding the scanning behavior increases the performance of the agent using a single marker, since it can now compare currently seen items with items outside the field of view. Adding a second marker also increases performance, as is discussed above. Adding the 3rd. 4th, and 5th markers does not appear to affect performance. The rightmost agent shown is an "ideal" agent: an omniscient agent that always moved directly

to the nearest berry (through obstacles if necessary), and provides a lower bound on the potential inter-berry distance.

This ideal agent performance is actually only optimal for an agent that has the strategy of going to the nearest berry, but what if the agent used a strategy that looked ahead more, by plotting a shortest path through the berries it knows about? Will this improve performance more, and be a bigger win for the use of representation? I implemented just such a strategy, to see if it would improve the performance of the agent, and found that it did not. The problem is that the world is dynamic enough (a new berry appears at a random location each time the agent eats a berry) and the information the agent has about its environment is sufficiently incomplete (the agent can only sense the few nearest berries) that plotting a course that looks ahead more is not helpful. The world is likely to change enough or the agent is likely to acquire new information that invalidates the latter steps of the longer plan. I found no significant difference in performance using this several-step shortest-path method.

As I "drive" the rabbit manually, however, I am able to achieve better performance by observing that the berries are often found in groups, and by preferring to go to groups of berries, rather than single berries, I can do significantly better. I did not implement this strategy in an automated controller, since it involves a "chunking" strategy [39] that is beyond the scope of this research, however, chunking as described here is not inconsistent with the other strategies used in this research, and can (and should) be used in conjunction with marker-based memories.

Given the example analyzed above, what can we say about the larger question of how many markers are useful for typical problems? We have shown that a few (in this case, two) markers can significantly improve performance, but that additional markers may not help. We can make further general statements about markers using the idea of "useful predicates." In the agent constructed here, the nearest berry location is a useful predicate, in that there is a current plan which uses the information regarding the berry location in order to determine action. Any other marked berries are only potentially useful, since we may construct a plan later using these predicates as preconditions. If no such plan is created, then it is of no benefit to acquire and maintain these extra representations. Furthermore, if acquiring the representation (i.e., doing the perception) is cheap and easy, rather than acquiring potentially useful predicates now and saving them for later, we can simply wait until the information is needed, and acquire it just before it is used.

As an example, contrast the performance of a rabbit agent using a single marker with rabbit agents using two or more. First of all, the rabbit agent with "extra" markers should place them where they are more likely to become useful, i.e., the nearby berries,

rather than berries that are far away. The marker then becomes useful when the agent decides to use it, and get that second berry. If the agent does not get the berry marked by the extra marker, possibly because other berries are later found to be nearer, then the extra marker was in fact useless. Even if the extra-marker agent does get that second berry, this may not represent an improvement over the single-marker agent, since the single marker agent, after eating the primary berry it has marked, may immediately sense the location of the second berry and put a marker on it then. The extra marker only yields a performance improvement if the agent uses it *and* it is more efficient to use the stored information rather than re-acquire the information later. In the experiments described above, the performance improvement of the two-marker agent over the one-marker agent was such a case, since the scanning behavior needed for re-acquiring the information takes time. The failure of three or more markers to improve the performance was either because the additional marker was never used (i.e., the marker was deleted before the agent actually got that berry), or because the agent could just as efficiently re-acquire that berry later. Moreover, since the saved information may become less accurate over time in a dynamic world, the information re-acquired "just in time" may be more accurate as well. The point to be made here is that representation, even if correct, is not necessarily *useful* in terms of a performance measure on some task, since the information is only useful if it helps the agent do something better than if it did not have the representation. By judicious selection of those things to be represented, only a small amount of representation can enable the agent to approach the performance of an agent which has complete information about the environment.

The observation that it is often better to acquire or re-acquire information as it is needed rather than use an internal representation is the crux of the "using the world as its own model" argument made by supporters of the reactive paradigm. Advocates of the use of representation must address this issue, since careful analysis of a problem may reveal that far less representation than initially considered necessary is actually needed. Using an efficiency argument, I have shown that in the case of the rabbit agent, the use of representation does in fact improve the agent's performance, even with respect to this "re-acquiring" criteria. However, there is an even stronger argument for the use of representation, which addresses capability, not just efficiency. If the agent's plan has conjunctive preconditions, and these preconditions cannot be sensed simultaneously, then the agent *must* use representation to achieve its goal. We will see examples of this later in rabbit world in the case of the predator avoidance behavior.

5.3.2. Obstacle avoidance

When evaluating an obstacle avoidance scheme, it must be done in the context of a navigation goal, otherwise the easiest way to avoid obstacles would be to not move at all (assuming other objects aren't moving). A good obstacle avoidance scheme must both minimize collisions and yet still allow the agent to reach its goal, so the evaluation measure must take both factors into account. In order to measure the performance of the obstacle avoidance schemes, I convert the berry gathering and collision avoidance criteria to a common currency—energy. Eating berries raises the energy level of the agent, and collisions lower it. We can observe the performance of the agent by plotting the energy level of the agent over time, as is done in Figure 5-19. Of course, the relative rates of energy increase/decrease is a parameter to the system, and we must account for this in the analysis.

In the experiments described below, I compared the performance of three obstacle avoidance schemes: no avoidance, a reactive avoidance, and a marker-based avoidance. In the no-avoidance scheme, the agent does not do anything special for obstacle avoidance, it merely moves directly towards the current target berry. However, we should note that a certain amount of obstacle avoidance is achieved due to the fact that the agent only moves towards berries for which it has a line-of-sight. This does not mean, however, that the path to the berry is free of obstacles, as illustrated in Figure 5-21. Since the agent has a non-zero width, the path towards the target object may have an obstacle in it, even though there is a direct line of sight to the target.

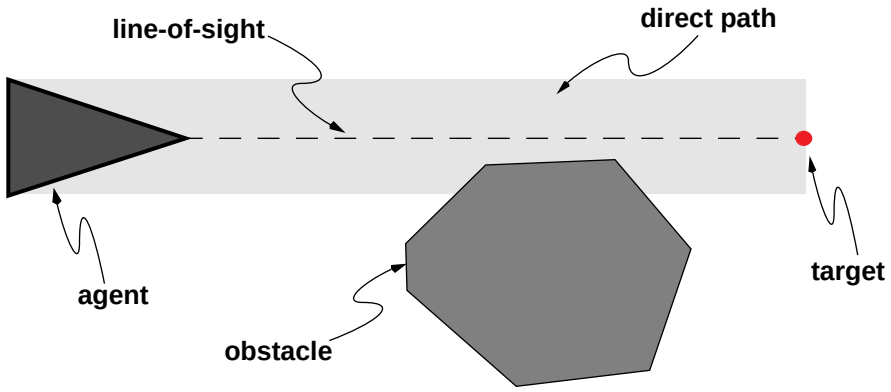


Figure 5-21: Direct path contains an obstacle

Note also that this does not mean that the no-avoidance agent must currently be looking at the target berry. All the agents discussed in this section use three "berry" markers and have a "neck." Rather, *if* the agent looks in the direction of the target, then it must

find the target in the visual input, otherwise the agent deletes the marker. The agents discussed in this section therefore, delete markers on occluded objects.

The reactive-avoidance agent uses a reactive behavior which is triggered whenever the agent is “close enough” to something. When looking in the direction of the target, the agent examines the ground line to determine if there are nearby obstacles. If so, the normal behavior of moving toward the target is subsumed, and the agent is directed to move towards the “open space,” i.e., the direction in which the ground line is sufficiently far away. The reactive-avoidance agent does not retain any memory for the location of obstacles, however, and therefore cannot know to avoid obstacles which are not in the current absolute field of view.

The marker-avoidance agent implements the obstacle avoidance routine described in sections 3.3.1.3 and 5.2.2.1, which establishes markers at the edges of the obstacle, and at an intermediate destination. The agent moves towards the intermediate destination before proceeding to the target berry.

All three of these agents use markers for berry locations and a “neck,” since these markers were established to be useful in the previous set of experiments. They all used three markers, which is actually one more than was established to be useful, but the extra marker doesn’t decrease performance either. Agents that don’t use berry markers perform more poorly on this obstacle avoidance task, since it incorporates the berry-gathering task.

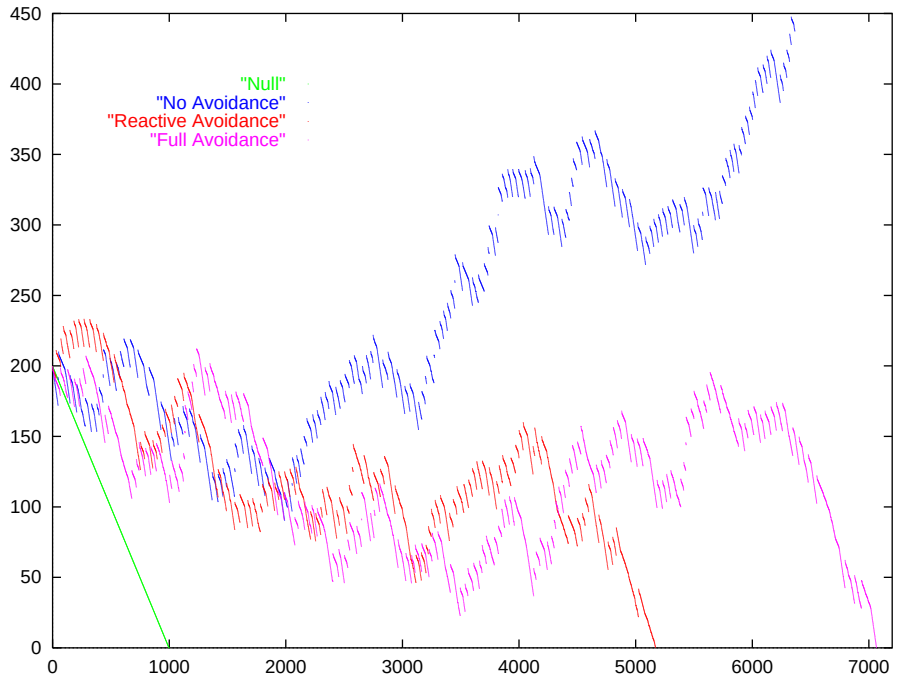


Figure 5-22: Agent performances with no penalty for collisions

As a basis for comparison, the agents are first evaluated in a environment that has no penalty for collisions. The energy time series for the agents in this environment are depicted in Figure 5-22. The agents are given an initial energy of 200 units, and allowed to run until they either expired, or it becomes clear that they are likely to survive indefinitely

I made several runs of each type; representative runs of each of the three types (plus the null agent, which stays in one place) are shown in Figure 5-22. The agents using obstacle avoidance strategies eventually "died" (their energy levels went to zero), whereas the energy level of the no-avoidance agent continued to increase (its energy level had reached 1200 when I stopped the run). These observations are true of all runs made, including those not shown in the figure. By changing other parameters, e.g., the energy value of berries, it is possible to construct environments in which the no-avoidance agents die, and reactive- and marker- avoidance agents survive indefinitely. However, at interesting parameter values, at which some agents survive and some do not, the no-avoidance agents survive, and the others do not.

This indicates that the obstacle avoidance strategies carry a cost, since they require the agent to slow down and go around obstacles, rather than simply bounce off them at full speed as does the no-avoidance agent described above. In order for the obstacle avoidance strategies to be beneficial, the penalty for collisions must be sufficiently high that the expense of using an obstacle avoidance strategy is justified.

In the following experiment, all parameter values are the same as those in the previous experiment, except that there is a penalty for collisions which is linear in the forward speed of the agent. The exact value of the penalty is not important, only that it is sufficiently high to justify the cost associated with avoiding obstacles. In order to determine the penalty, I simply raise the penalty at regular intervals until there is a qualitative difference in the relative performances of the agents. Once this point is found (at a penalty of 200 energy units per units of agent speed), the penalty is fixed for all the runs.

Since this is a strictly harder problem, we expect the performance of all agents to decrease to some extent as compared to the problem in which there is no penalty for collisions. In accordance with this expectation, all the agents in this experiment eventually died. However, if we simply look at the survival times of the agents, a clear pattern emerges. Figure 5-23 shows the survival times for five runs of each of the three agents.

None of the five runs of the no-avoidance agent is longer than 4000 time steps. The same is true of the reactive-avoidance agent. On the other hand, all runs of the marker-avoidance agent survived at least 8000 time steps. One marker-avoidance run lasted over 18000 time steps. Clearly the marker-based avoidance scheme is effective, i.e., the markers have expanded the agent's effective field of view.

The reactive avoidance scheme appears to be completely ineffective. We cannot conclude purely on the basis of this evidence that all reactive strategies are ineffective on this problem, only that one person (with no particular interest in constructing a good reactive obstacle avoidance scheme) failed to construct one. However, if the absolute field of view does not contain enough information to determine an effective action, then no reactive strategy can be effective. Consider the situation back in Figure 5-21. As the agent moves forward on the path to the berry, the rock may pass outside the field of view, but back end of the agent may still collide with the rock. If the agent can bump into things that are not in the absolute field of view but have been seen before, i.e., things which an agent with marker could have in its effective field of view, then the agent using markers must out-perform the reactive agent.

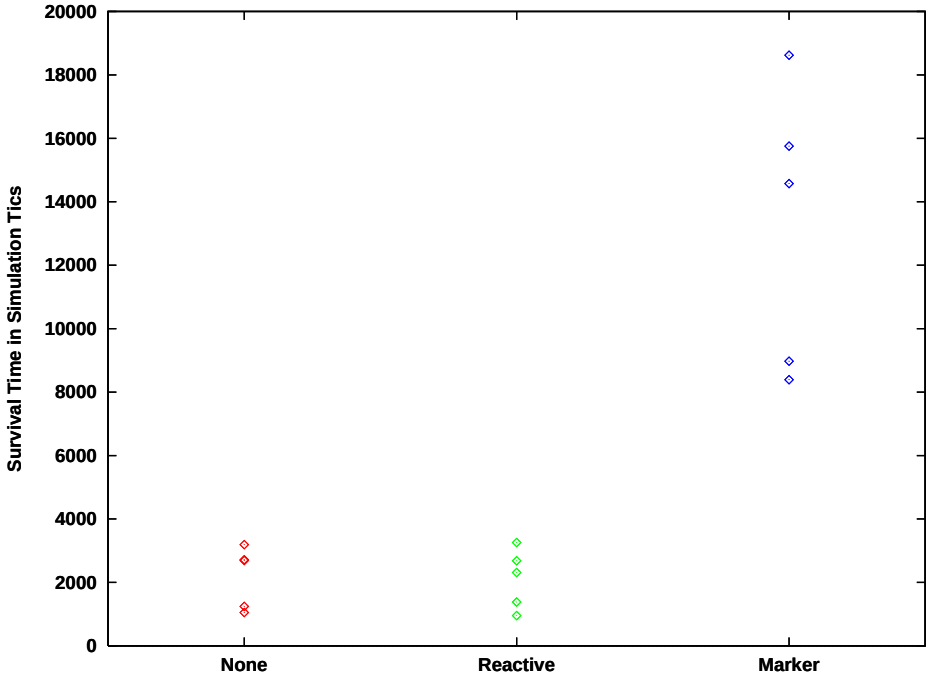


Figure 5-23: Survival times of obstacle avoidance approaches

Rather than looking at just the bottom line as is done in Figure 5-23, we can also look at the time series of all the runs, which are shown in Figure 5-24. This figure dramatically depicts the superior performance of the marker-avoidance agent (in blue). As a point of reference, it took about 40 minutes to do a run of 20000 tics, which works out to an average of about 8.3 tics per second. These numbers are approximate, however, since the actual speed of the simulation depends on the job mix on the machines, network load, etc. The simulation usually ran at slightly more than 10 hertz, with occasional pauses of 1 to 20 seconds caused by external load, bringing the average down to the 8.3 hertz mentioned.

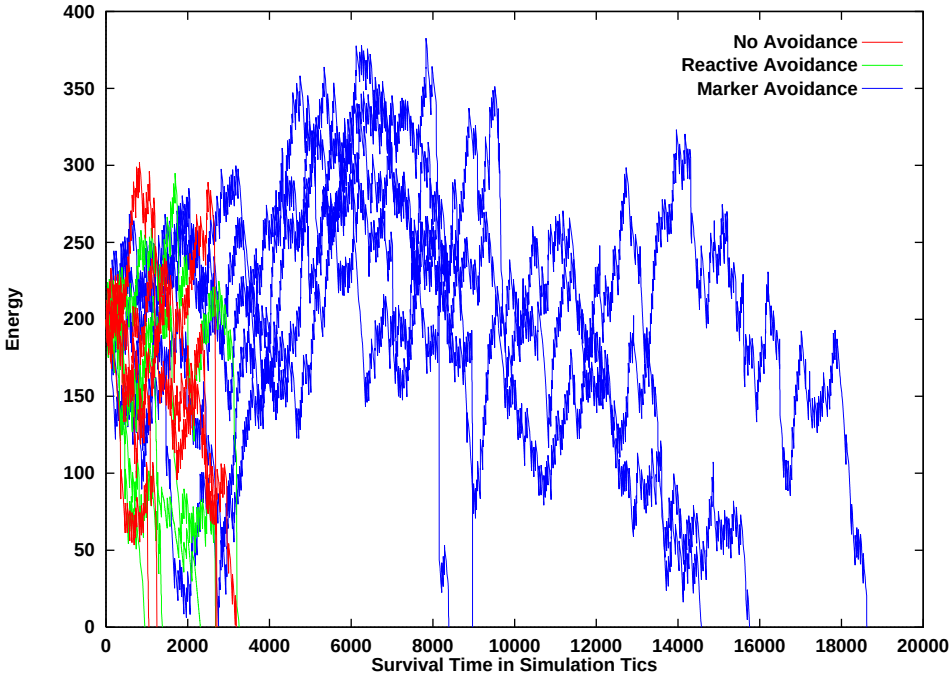


Figure 5-24: Time series of obstacle avoidance performance

The data points in Figure 5-24 are joined by lines, in order to make it easy to see the expire times of the marker-avoidance agent, at around 8300, 9000, 14700, 15800, and 18700. Looking closely at this figure, it appears that a run of the marker-avoidance agent did expire at around 2000 tics, and again around 2700 tics. This agent did not expire, however, although its energy level did get very low (to just 1 energy unit at about 2700 tics). Even if we allow that this particular agent just "got lucky," the performances of the marker-avoidance agents are clearly superior to the alternatives. The run in which the marker-avoidance agent nearly expired at 2000 and 2700 tics is interesting in that several "bad things" happened on this particular run. Figure 5-25 shows the time series of this run

During the early part of the run, the agent performed well. The energy slowly decreases as the agent moves between berries, and then sharply increases when the agent eats a berry. This pattern of alternating slow decrease followed by sharp increase persisted through the first thousand tics of the run. During the second thousand tics of the run, the agent runs into some difficulty. Around 1200 tics, there is a long slow decrease, indicating that the agent took a long time to find the next berry. Then, around 1500 tics, there are some large downward jumps in energy—these correspond to collisions. The marker-avoidance strategy (or more accurately, my implementation of the marker-avoidance strategy) is not perfect, and there are still a few collisions. These collisions, combined with long stretches between berries, result in the agent's energy plummeting from around 245

units at 1000 tics, to 6 units at 2000 tics. The agent then finds a cluster of several berries, and the energy briefly recovers to 100 at about 2500 tics. But once this cluster is consumed, the agent again has trouble finding berries, and the energy decreases to just one energy unit at 2755 tics. Just as the agent is about to expire, however, it finds a a large cluster of berries, enabling the agent to recover to 220 energy units at around 3500 tics.

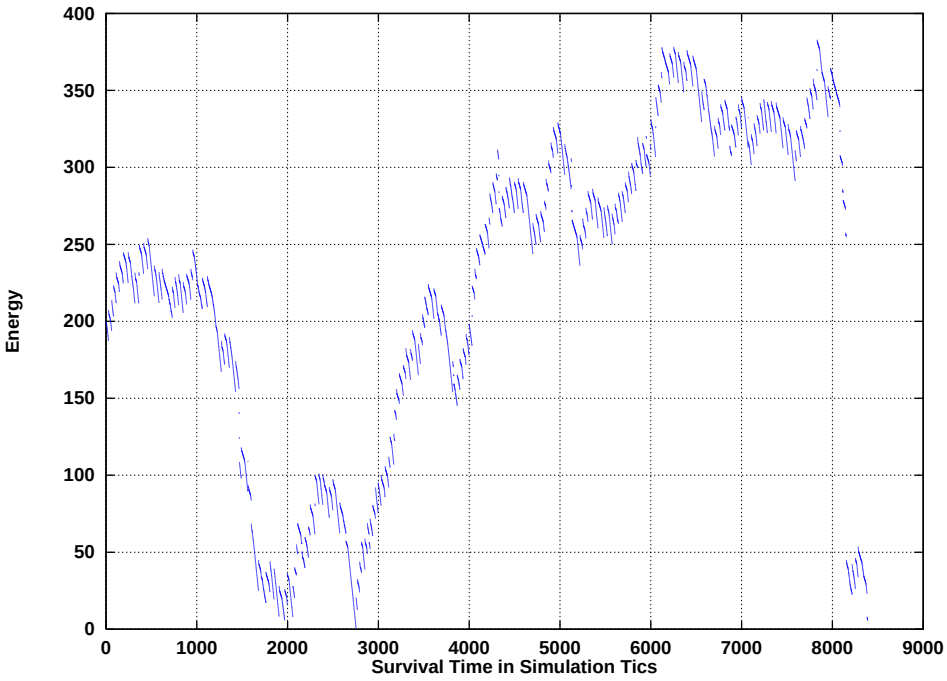


Figure 5-25: An "interesting" run of the marker-avoidance agent

From 3000 to about 8000 tics, the agent alternates between periods of steady increase and decrease, with the occasional collision. This alternating pattern is observed in all the runs, with the periods of increase corresponding to the agent consuming all the berries in a local region, and the periods of decrease corresponding to the agent moving out of the current region (which is now sparse) to a region with more berries. On this particular run, between 3000 and 8000 tics, the periods of increase are greater than the periods of decrease, resulting in a net increase over the long term. This agent achieved an energy level of 380 units at around 7800 tics, which is the highest ever achieved by any of the agents shown depicted Figure 5-24.

Unfortunately for this agent, at 8155 tics a disastrous event cost the agent 210 energy units, and it is unable to recover. The agent located a berry, and moved towards it, but in order to reach the berry it had to pass between two obstacles, as depicted in Figure 5-26. The agent overestimated the size of the gap between the obstacles, and attempted to move through the gap. Unfortunately, the gap is too narrow to safely pass through, and the agent became "wedged" between the obstacles. In this situation, the simulation semi-ran-

domly “bounces” the agent between the two obstacles (penalizing the agent for each bounce), until the agent happens to bounce through to the other side (or dies). In the case of this agent, the agent eventually bounced through, but it cost a penalty of 210 units. The marker-avoidance agent that expired at around 9000 tics had a similarly disastrous experience.

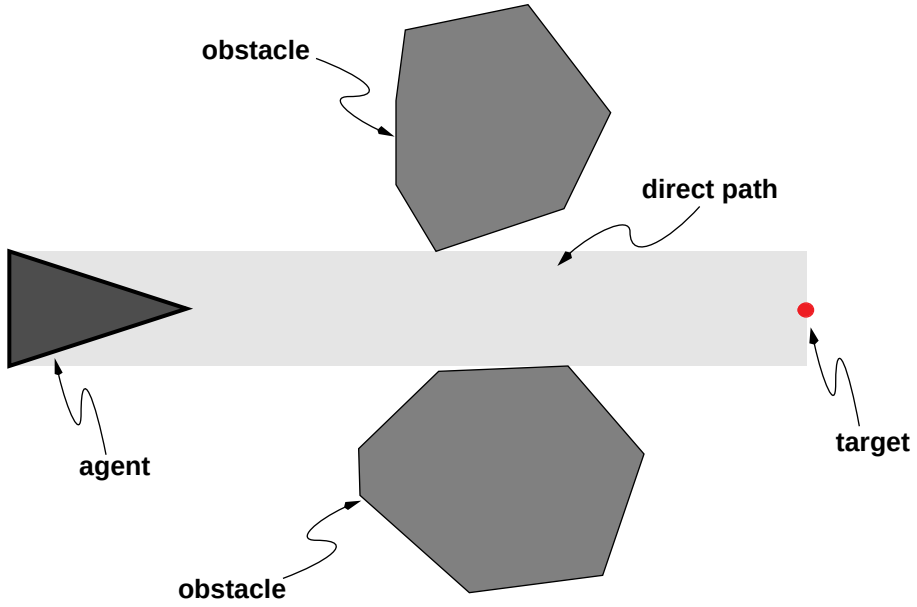


Figure 5-26: A disaster in the making

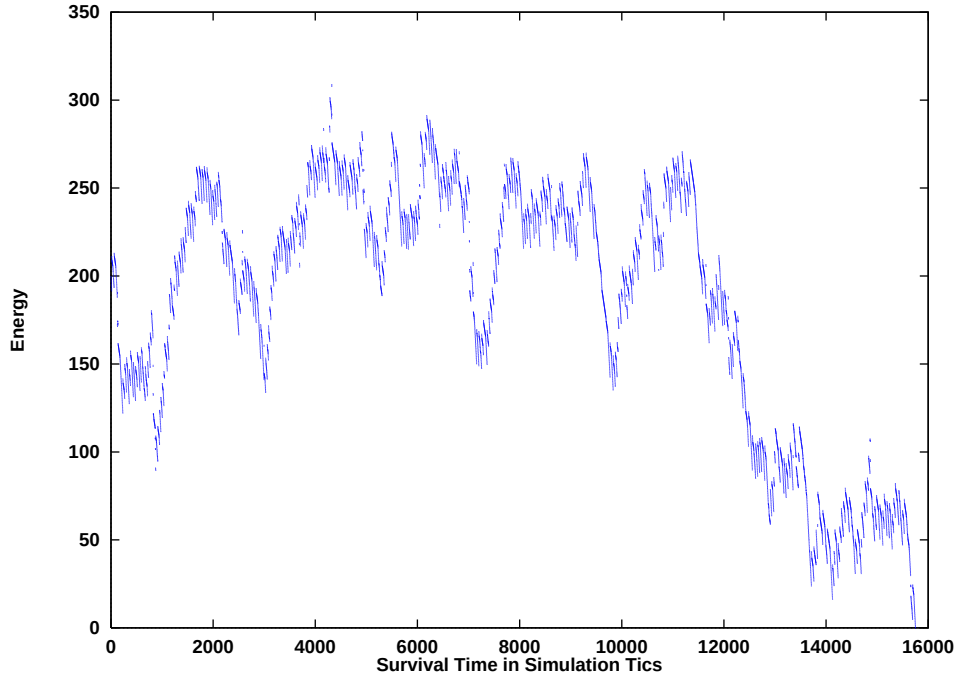


Figure 5-27: A representative run of the marker-avoidance agent

Figure 5-27 shows a more representative[1] run of the marker-avoidance agent. This run displays the typical alternation between steady increase while in clusters of berries, and steady decrease in the sparse periods between clusters. In the run in Figure 5-27, this cycle very roughly appears to have a period of 2000 tics. Near the end of run, a sparse period goes on longer than usual (from 11000 to 14000 tics), and the agent eventually expires. For comparison, Figure 5-28 shows representative runs of the no-avoidance and reactive-avoidance agents (note the change in scale). In both runs shown in Figure 5-28, the agents have trouble with collisions (around 1500, 2500, and 2700), that eventually contribute to an early demise.

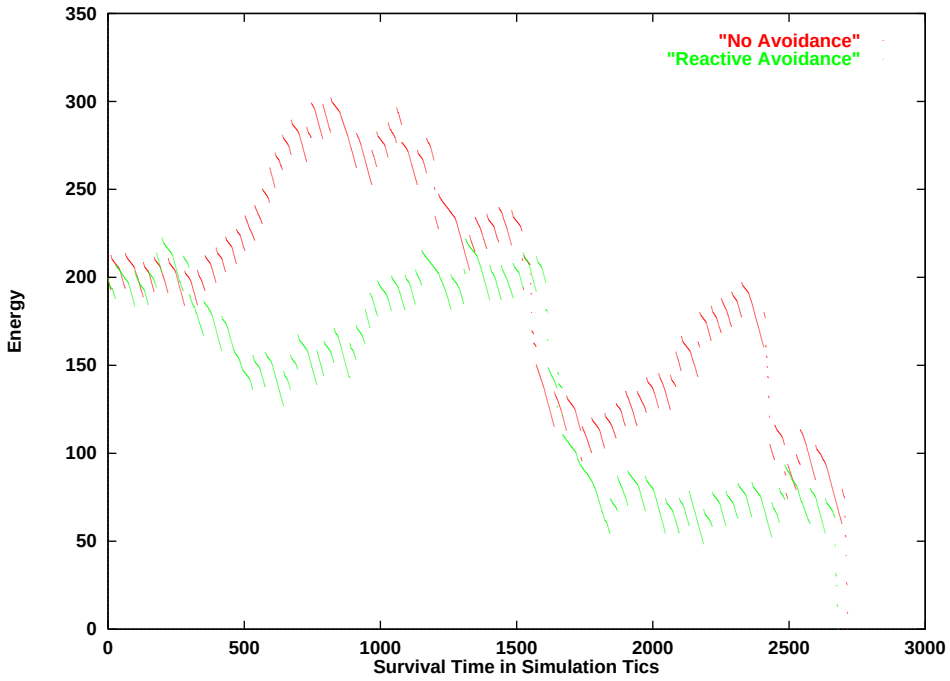


Figure 5-28: Representative runs of no- and reactive- avoidance agents

It appears that the reactive scheme is not effective, since the agent is required to look at the obstacle in order to avoid it (i.e., keep the obstacle in the absolute field of view). Given a camera with a limited, cone-shaped field of view, it is sometimes impossible to look at the nearby obstacles that are slightly to one side, and also look forward in the direction of travel toward the target. The collisions that the reactive-avoidance agent makes occur when the agent is looking towards the target berry, and cannot see the obstacle nearby and slightly to one side. The absolute field of view did not contain enough information about the situation to determine an appropriate action.

1. In Figures 5-27 and 5-28, the “representative” runs are the second longest surviving of the five runs for each agent type.

The marker-based obstacle avoidance scheme is an effective strategy. The use of a marker at the edge of the obstacle is an example of expanding the agent's effective field of view. Even when the obstacle passed outside the agent's absolute field of view, the agent could still know to avoid it, since the marker kept the obstacle in the agent's *effective* field of view.

5.3.3. Predator avoidance

In evaluating performance of the marker-based predator avoidance scheme described in section 5.2.2.2, it is difficult to determine a "reactive" alternative to use as a basis for comparison. Since the goal is to run *away* from the predator, and we have a limited absolute field of view, it is not possible to look at the predator and look where we are going (to avoid collisions) simultaneously. Furthermore, given the behavior of the predator, the most effective strategy for the agent is to position some obstacle between itself and the predator, and then move sufficiently far away from the predator. In order to do this effectively, the agent should maintain an estimate of the location of the predator even when the predator is occluded. Reactive agents are incapable of maintaining such an estimate. Only through the use of memory can the effective field of view of an agent with these sensor limitations be expanded to include the predator and obstacle locations in the presence of occlusion. We must therefore compare the performance of the marker-based agent with a reactive agent without a predator avoidance strategy, because there is no effective reactive predator avoidance strategy.

Next, we must consider the performance metric to use. If we simply look at "mean time to failure," there will be a great deal of noise in the measurements, since the mean time to failure is largely dependent on whether the predator happens to encounter the agent at all. Instead, we want to look at whether the agent escapes, given that an encounter has occurred. In order to measure this, the simulation must be instrumented to determine when an encounter occurs. Once an encounter begins, it either ends with an escape, or with the predator catching and eating the agent.

In order to detect "encounters," the simulation is instrumented to determine whether an unobstructed straight-line path existed between the predator and the agent, i.e., there are no obstacles between the predator and the agent. If such an unobstructed path exists, this is taken to be the beginning of an encounter. The encounter ends when either the straight-line path becomes obstructed (an *escape*), or the agent is caught (an *eat*). The predator is omniscient, so it "knows" where the agent is regardless of whether the path is obstructed.

The performance of a reactive agent is shown in Table 5-1. Since the reactive agent really has no strategy for escape, the escapes occur by chance. However, this agent does "escape" by chance on occasion, since an occlusion might occur by accident as the agent continues its pursuit of berries. The simulation is allowed to run until there are 100 total encounters. If the encounter ended in an escape, the escape is recorded and the simulation allowed to continue. If the agent is caught, the "eat" of the agent by the predator is recorded, and both the predator and agent are transported to a random location. The simulation then continued as before. Table 5-1 shows the results of 5 different runs of 100 encounters each. The reactive agent escaped by chance on an average of a little more than 30% of the encounters. The average escape to eat ratio is 0.49, or about 1:2.

	Escapes	Eats	Ratio
Run 1	49	51	0.96
Run 2	32	68	0.47
Run 3	29	71	0.41
Run 4	21	79	0.27
Run 5	33	67	0.49
Average	32.8	67.2	0.49

Table 5-1: Chance escape performance

Table 5-2 shows the performance of the marker-based predator-avoidance strategy. The agent escaped on slightly over 70% of the encounters. The average escape to eat ratio is 2.7, or nearly 3:1.

	Escapes	Eats	Ratio
Run 1	68	32	2.13
Run 2	75	25	3.00
Run 3	74	26	2.85
Run 4	77	23	3.35
Run 5	71	29	2.45
Average	73.0	27.0	2.70

Table 5-2: Marker-based escape performance

Clearly, the marker-based strategy is not perfect. In order to successfully escape, the agent must detect the predator, find an open space to run, then find a hiding place, and successfully get behind the obstacle that constitutes the hiding place. Any or all of these steps can and do fail. The predator travels faster than the agent does when the agent is collecting berries; only when the predator is detected does the agent accelerate to slightly faster than the predator. If the agent does not detect the predator, the predator will easily overtake the agent and catch it. The agent can fail to detect the predator because it never looks in the direction of the predator as it approaches, or because the predator detection visual process fails to find the predator in the image even when it does look. The predator must be quite close for the predator detection visual process to work.

If the predator comes upon the agent when the agent is in a corner, the agent may not find an open space in which to run. Even if an open space is found, the agent may need to turn around almost 180 degrees in order to run towards the open space, which usually takes long enough that the predator catches the agent before the turn operation is complete. If a hiding place is found, then the agent may bump into an obstacle on its way towards the hiding place. This can effectively slow the agent down to the point that the predator catches it.

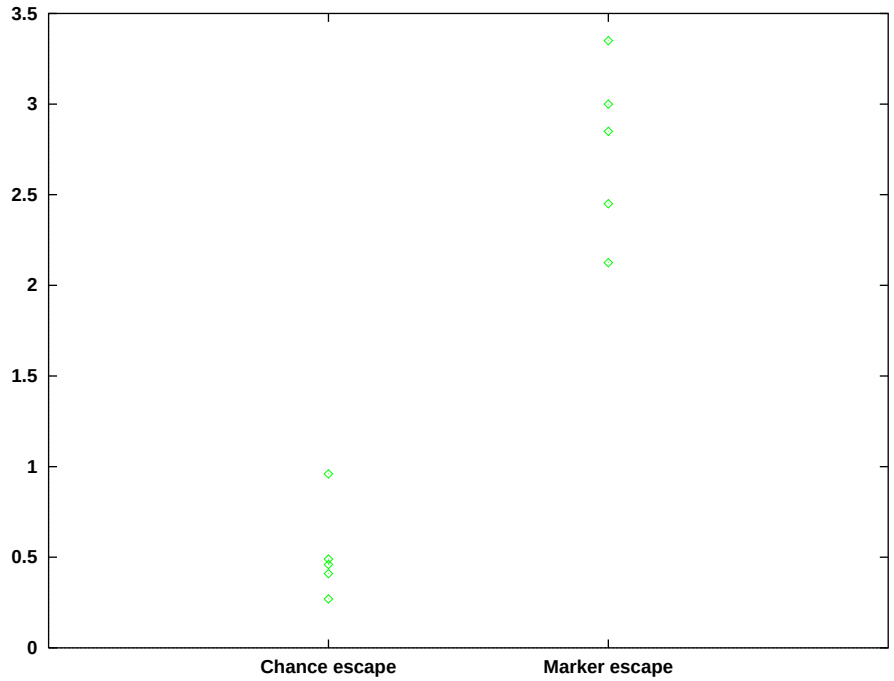


Figure 5-29: Escape ratios of the two agents

Figure 5-29 compares the performances of the reactive escape-by-chance and the marker-based escape strategies. Although the variance of the marker-based agent's performance is high, the marker-based strategy is clearly effective in comparison with chance.

As the agents are performing the escape task, I observed the performance from an overhead viewpoint, as in Figures 5-7 through 5-14. From this vantage point, the escape task appears trivially easy, since there is effectively an unlimited field of view, and there is no occlusion. If the agent had access to this viewpoint, there would be no possibility of the predator "sneaking up" on the agent, and the agent would have no difficultly in locating and moving to a hiding place. It is only when the agent's knowledge is limited due to the first person viewpoint that the task is interesting. I controlled the agent myself using the first person viewpoint, and found the predator avoidance task to be quite difficult, and usually never saw the predator coming before being caught.

The predator avoidance task given the first-person viewpoint therefore exemplifies the importance of considering the effective field of view when the absolute field of view is inadequate to perform the task. Given perfect information, the task is easy; unfortunately, perfect information is rarely available. The use of marker-based memory structures increase the effective field of view so that the agent has enough information about the current situation to determine an effective action.

5.4. A physical robot agent

In order to further demonstrate the viability of the marker-based approach to expanding the effective field of view in physical environments, we have implemented this approach on a physical robot [10]. The robot's task is to detect and navigate to a goal location, circumventing obstacles as necessary. The environment is our laboratory, a cluttered room in daily use, with some cones on the floor—the goal location is a striped cone. Figure 5-30 shows two images acquired from the camera on the robot.

If there is a clear path from the agent to the goal (as in the image on the right in Figure 5-30) the agent can proceed directly to it. If the path is blocked (as in the image on the left in Figure 5-30) the agent must go around the obstacle. The agent searches for the goal and places a marker on it. If necessary, it may also mark any obstacles in the direct path to the goal location. In a tight loop, the agent computes the current position of the markers, and based on their location, it decides what action to take.

Our robot, Bruce, is a Rug Warrior based robot [35] (MC68HC11A1 microcontroller) whose sensors consist of a single, front-mounted, black and white camera, one axle-mounted shaft encoder for each wheel, and a "bump skirt" to detect collisions (see Figure 5-31). The camera is mounted on a pan/tilt "neck," though in the trials described here, the

robot only looked directly to the front. Host communication takes place via thin cable or wireless transmission of both video and control signals.

Figure 5-30: The physical agent's environment

.

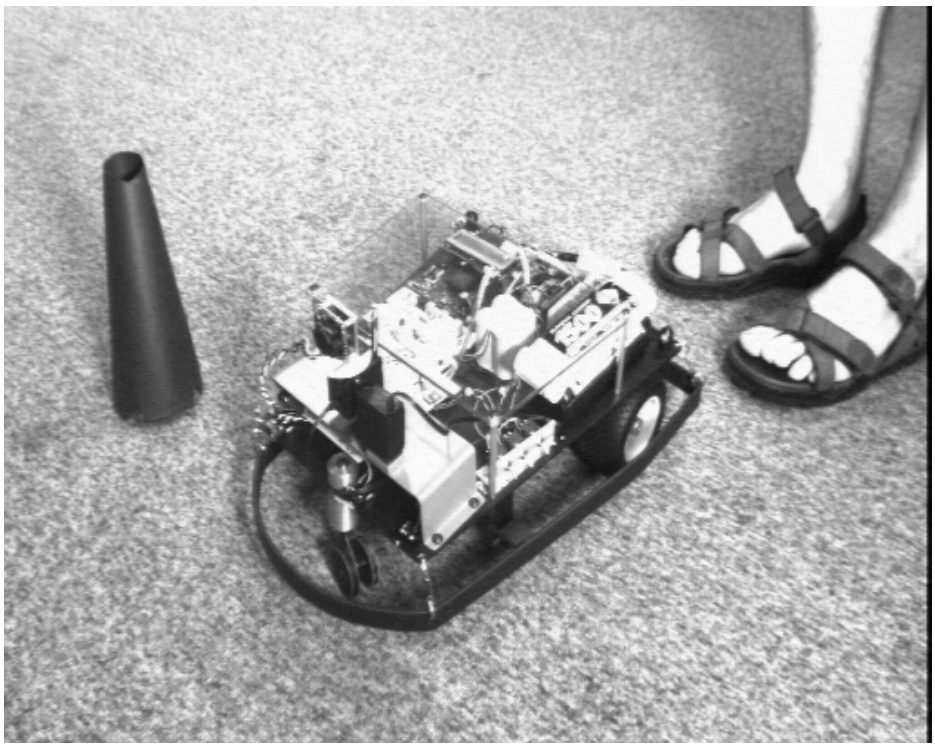


Figure 5-31: The physical robot

The vision system consists of a gray-scale camera connected to a Datacube MV200 image processor and a Sparc 2 Workstation. The MV200 captures a 512x512 image, sub-samples it to 128x128, smooths this image with a 3x3 Gaussian kernel, and then convolves it with an edge magnitude operator. The resultant image is thresholded and

sent to the Sparcstation for connected component analysis, which eliminates very small connected components from the image, since they are assumed to be noise (or dirt on the carpet). A modified flood fill (similar to that used by Horswill [33]) of this image from the bottom produces the *ground line*, which in this case separates carpet and non-carpet

5.4.1. Goal detection

Points at which the ground line "jumps" (i.e., makes large changes in height in the image when traversing the ground line from left to right) indicate the edges of objects: downward jumps being left edges, and upward jumps being right edges (see Figure 5-32). Left/right pairs are matched to obtain potential goal-object locations in the image, slightly above the ground line and between the matching pairs of jumps. The real-world location of the object is computed using height of the ground line in the image to compute depth.

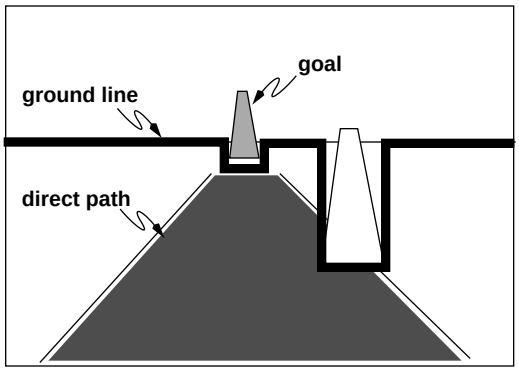


Figure 5-32: Detecting obstacles to the goal

The agent's goal is a horizontally striped cone. To find it, the vision system subsamples the 512x512 square image obtained above to 256x256 and convolves the result to compute the vertical derivative image. The absolute value of the derivative image is subsampled again to 64x64, convolved with a box mask to smear it, and then thresholded to create the *goal detection image*. The potential goal-object locations found by processing the ground line as described above are then checked for their response at the corresponding locations in goal detection image. If the response is high (i.e., there are a lot of horizontal lines at that location), the location is marked as a goal.

5.4.2. Obstacle detection

Finding obstacles is accomplished by determining whether the direct path to the goal, as projected into the image, intersects the ground line (see Figure 5-32). The intersection indicates an obstacle along the path to the goal. The direct path is determined from the given width of the robot and the assumed horizontal orientation of the ground plane. If

an obstacle is found, it is marked as an obstacle to the goal. This is a case of a dependent marker, since this obstacle is dependent on the detected goal and its direct path. If the goal were to be deleted for any reason, the dependent obstacle marker would be deleted as well.

If an obstacle is found, the agent must navigate around it. This is done by placing an intermediate-destination marker at a fixed position relative to the obstacle. This marker is dependent on the obstacle marker instantiated above. Its location is determined based on the location of the obstacle and the width of the robot. It is placed outside the direct path to the goal, and does not correspond directly to any image feature.

5.4.3. Marker maintenance

As discussed in section 3.1.1, the positions of marked objects are stored in an ego-centric polar coordinate system. The system knows the width of its field of view and which markers it ought to be able to see. Nearest-neighbor matching via a stable-marriage algorithm is used to determine correspondence between "remembered" (marked) objects and the currently visible objects. Marked objects which lie outside the agent's field of view are updated to account for ego-motion. The ego-motion is computed based on readings from shaft encoders on the robot's wheels.

5.4.4. Action selection

What action to take an any given time is determined by the state of the markers (i.e. their position). If there is an allocated goal marker, and no obstacle markers, the path to the goal has been determined to be clear, so the agent proceeds directly. If an intermediate-destination marker is associated with the goal, the direct path to that goal is obstructed, and the agent must go to the intermediate destination instead. Once the intermediate-destination marker is reached, both it and the obstacle marker can be deallocated, leaving the goal as the only marker. The goal can then be pursued directly.

One of the important behaviors validated through our implementation of a marker-based system to control an actual robot is the maintenance of a primary goal. In particular, our robot system has, without operator intervention, been able to detect a goal object, identify an obstacle along the path to that goal, and allocate an intermediate-destination marker. Then, when proceeding towards the intermediate-destination, the robot can have the goal and obstacle both pass outside of the field of view of the robot's camera. The use of the marker representation enables the robot to maintain the primary goal (which is outside the sensory input), while still reacting to the *current* sensory input coming from the camera, and to finally attain the goal location.

Chapter VI: Future Work

This chapter discusses possibilities for extending or augmenting the work done for this dissertation. The concept of the effective field of view, and the use of marker based-representations can be augmented by using other representations. A means of constructing plans compatible with agents with limited information is also needed. Another avenue for investigation is to search for possible biological implementations of extensions to the effective field of view. I will close this chapter with some speculation as to how this work in spatial representations has consequences for artificial intelligence in general.

6.1. Relationship to other spatial representations

The marker-based representations developed in this dissertation are not meant to replace other forms of representation of the environment, but rather to augment those other forms. For large-scale navigation, it is useful to have a more traditional topological map representation. In a topological representation, one navigates from node-to-node in order to reach a goal. Path planning in the large-scale space may be done independent of any marker-based representations, but once a path is chosen in the large-scale, navigation *within* a node or edge in the topological map should be done via marker-based representations. In fact, at the level of actually interacting with objects in the environment, one must use an egocentric representation, since the egocentric representation contains the information needed by the effectors for competent action. Marker-based representations are where the “rubber meets the road” so to speak. Large-scale maps and local-space representations combine to provide more information (and enable more competent behavior) than either type of representation alone.

Another type of representation concerns the spatial relationships of external objects. Rather than have the agent represent all objects in the local space egocentrically, the agent can represent the locations of some objects egocentrically, and other objects can be located relative to the egocentrically represented objects [6]. For example, consider an example in which there are two important objects in the environment, a lamp and a table, and the lamp is on the table. The agent may represent the location of the table, and then simply remember that the lamp is on the table. This results in more efficient algorithms for updating the locations of the objects as the agent moves. Rather than represent and transform the coordinates of both objects independently, only the table coordinates need be transformed. The coordinates of the lamp (which are “table-centric”) remain unchanged.

An additional advantage of representing the spatial relationships of external objects is that the external spatial representations are more robust to error than using

purely egocentric representations. Imagine that some small percentage error is incurred during each ego-motion inverting transformation. If the lamp and table locations are represented independently and egocentrically, it may be the case that as the agent moves, accumulated error of the transformations needed to compensate for ego-motion causes the represented lamp location to "drift" under the table. This cannot happen if we explicitly represent the fact that the lamp is on top of the table. Yantis' [68] experiments with visual tracking of multiple objects indicate that representation of the spatial relationships of the tracked objects are critical to tracking performance in humans; it is much easier to track multiple objects if the spatial relationships of those objects are do not change. More evidence for the importance of representing and considering the spatial relationships of external objects is provided by Barnard and Thompson [7], who constructed a robust system for solving correspondence problem by choosing corresponding points such that the spatial relationships were best maintained across the two images.

Note that the use of external spatial relationships is merely another (perhaps in some cases more efficient and robust) means of retaining the information within the effective field of view than direct egocentric coordinates. Note also that when the agent actually goes to interact with an object (to pick up the lamp, for example) the object's location must be represented in egocentric coordinates.

6.2. On-line planning

When constructing a plan for action in the real world, it hardly makes sense to plan in minute detail very far in advance, since the information needed for such a detailed plan is usually either not available, or subject to change. However, it may be possible to construct a high-level plan in advance based on relatively stable information, and then only fill in the details as necessary. For example, consider the case for planning a trip from my home in Charlottesville to a hotel in Miami. It isn't necessary to plan my route from the airport in Miami to the hotel until I get to the airport. I only need to make sure I get to Miami in time, and have enough money to rent a car. I can make a high-level plan as illustrated in Figure 6-1a.

Once I've made my high-level plan, I can fill in the details for getting from home to Dulles airport, as in Figure 6-1b. Moreover, I can start *executing* my plan to get to Dulles before I even make the plan to get from the Miami airport to my hotel. Starting to execute the plan before it is complete is what I mean by "on-line planning." Starting with a high-level plan and adding detail is similar to the approach used by Sacerdoti's ABSTRIPS [53], except that ABSTRIPS constructed the entire plan before executing. On-line planning is compatible with agents having incomplete information. Not only *can* I

start executing my plan to get to Dulles before making a plan to get to the hotel from the Miami airport, I *must*, since I don't have enough information yet to make the latter plan. I can get a map of Miami when I get to the airport, and start planning the last stage of the trip then. Future research is needed determine the viability of this approach in general. Combined with the use of techniques employed by partial information planners, a powerful perception/action system may be realized.

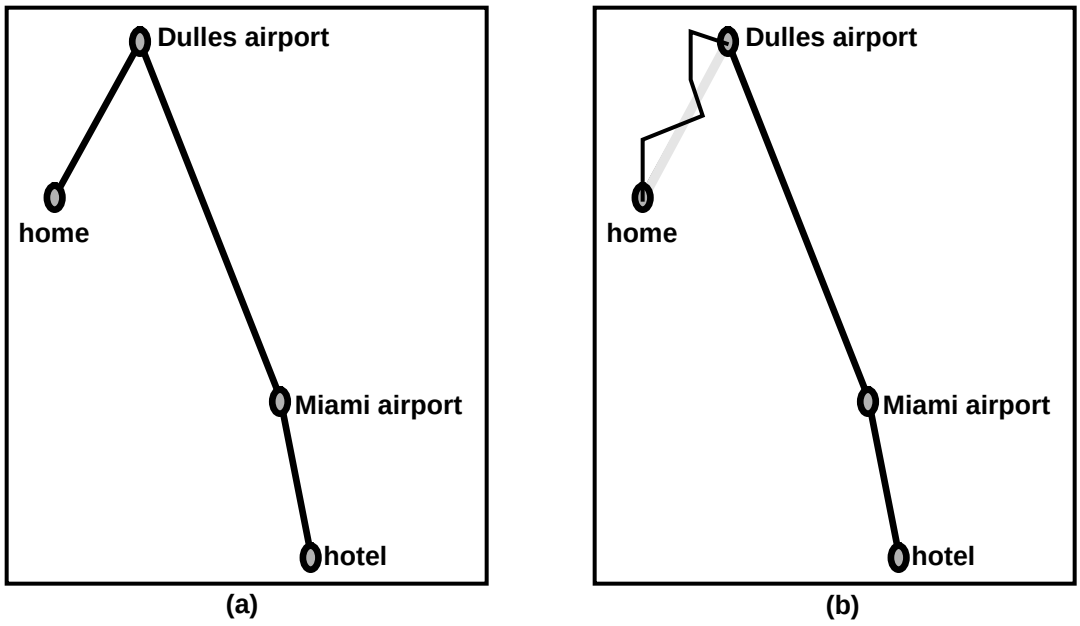


Figure 6-1: An on-line plan to get to my hotel

At each level of the abstraction hierarchy, some (small) number of "control points" are chosen; the example above uses two control points corresponding to the airports. As the planner descends each level of the hierarchy, it only add control points between the start state the first control point. Execution can begin as soon as the earliest part of the plan consists of executable actions.

The significance of on-line planning to marker-based memory systems is that the memory needed (an more importantly, the maintenance of that memory so that is reflects the state of the world) is limited. If we assume that the number of control points selected at each level in the hierarchy is a small constant, and further assume that the control points are selected such that they are "evenly distributed" along the length of the plan, then the total number of control points in existence at any given time is logarithmic in the plan length (control points are deleted after the sub-goals they represent are achieved). This may result in a small enough amount of memory to allow the agent to use its sensors to maintain this memory with a high degree of certainty, using marker-like structures. The

markers representing the control points are all dependent markers, each dependent on the goal it subserves which is one level up in the hierarchy. In the example above, the airport control points are dependent on the goal of getting to the hotel in Miami. At the next lower level in the hierarchy, the control points on the path from Charlottesville to Dulles Airport are dependent on the goal of getting to Dulles, (which is in turn dependent on the goal of getting to the hotel in Miami). If the goal of getting to Miami is abandoned, then all of the dependent control points are deleted as well.

6.3. Biological implementation

The marker-based representations used by the artificial agent constructed for this research were implemented via memory locations containing the egocentric coordinates of the object, and those coordinates were updated as the agent or object moved. Such an implementation is not biologically plausible, so one wonders how these egocentric locations can be stored in biological systems.

For objects inside the absolute field of view of the eyes, there is a retinotopic representation available, in which there are cells corresponding roughly to each point in the visual field. One would expect a similar solution, i.e., a coding by location rather than a coding by value, for the extended field of view, albeit at a lower resolution than that used in the retina (certainly lower resolution than the fovea). Attneave and Farrar's [5] subjects described their experience as seeing objects out of "eyes in the back of the head," which suggests that the implementation of the representations used for objects outside the field of view is similar to that used for objects inside the field of view. Along these lines, Attneave and Farrar postulated the existence of a cycloramic, 360 degree field of view to account for their data.

Some of Attneave and Farrar's subjects also reported a kinesthetic experience, i.e., they "felt" where the objects were, and I suggest that this is the most illuminating description of the experience. If we are to encode locations in a biologically plausible manner, we need a roughly spherical organization for a group of "direction cells." We have a ready-made group of such cells in the form of skin receptors, or at least in the set of neurons a few layers removed from the skin receptors. I suggest that some portion of the mechanism used to determine where an object is touching the skin is co-opted to determine where an object is located when it is not touching the skin. This (admittedly speculative) theory could be verified via cell recordings which indicate similar neural activity when an object is near an animal as when it is actually touching the animal.

6.4. Navigation: the universal problem?

One wonders whether the problem of getting around in the physical world is a narrow sub-problem of artificial intelligence, or whether developments in robot navigation have implications for general intelligence. If we look to human evolution for an example of the creation of an intelligence, we find evidence for the application of navigation to general intelligence.

Before humans evolved to the point of being capable of solving abstract problems, creatures (including humans) needed to navigate their environments. Mechanisms for navigating the environment evolved into mechanisms for general problem solving. Humans generally appeal to physical models to solve abstract problems, and use a notion of forward progress to guide the search for a solution. We speak of "steps" in a proof, even though no one is really going anywhere. An artificial evolution might follow the same lines as natural evolution by constructing the capability to deal with the physical environment, and then evolving the techniques used into general problem solving methods. The pure reactive approach, by rejecting representation, is insufficient for general intelligence. By reintroducing and redefining the role of representation to the model of an agent, this dissertation opens the possibility that the mechanisms developed here are applicable to general intelligence.

Chapter VII: Conclusion

There have been two primary paradigms used to determine actions for mobile robots: classical and reactive planning. In order to build mobile robots which perform useful tasks in the physical world, we must capitalize on the strengths and compensate for the weaknesses of both approaches. Until now, it has been difficult to reconcile the paradigms' treatment of perception, representation, and action.

The classical approach, by divorcing perception from action, produces systems incapable of effective interaction with the real world. The nature of the real world and the nature of sensing dictate that an task-oriented approach be followed in the design of perception-action systems. Perception and action are so fundamentally intertwined that they must be studied in conjunction with one another.

Reactive planning, while being direct perception-based, abandons the machinery needed to analyze the task components and the situation, severely limiting the range of tasks that reactive agents can perform. The actions an agent performs in the world are often sufficiently complex that the agent must consider alternatives, consider the interactions of tasks, and remember relevant events and objects that may not be currently sensed.

7.1. The effective field of view

The concept of an *effective field of view* forms a foundation for building systems that combine the best features of the classical and reactive approaches. The effective field of view of a sensor is defined to be the information extracted from the sensor that is useful. The agent decides upon an action at each instant, based on the information in the effective fields of view of its sensors. The primary functions of the perception, representation, and action systems can be thought of as expanding these effective fields of view.

The perception system expands the effective field of view of a sensor by extracting additional information, i.e. *useful* predicates, regarding the environment. Which predicates are useful depends on the task to be performed. At any given instant, the sensor can only extract *sensor predicates* within its *absolute field of view.* Additional information, i.e. *inferred predicates*, can be deduced from the sensor predicates. The agent can also be endowed with some amount of useful information *a priori*. The *a priori* information is a set of facts about the world that can be hard-wired into the agent; e.g., an inference rule encodes a fact about the real world. In order to deal with the realities of the physical world and perception of it, all of the predicates must have *certainties* associated with them, and some means of propagation of these certainties must be employed.

The memory system may retain some representation of useful predicates over time. The use of representation can expand the effective field of view of a sensor beyond its absolute field of view by endowing predicates with a temporal extent. These predicates and their associated certainties must be updated to reflect change over time. When the certainty of a predicate falls sufficiently low, it is no longer useful. If the predicate involves an important aspect of the task, then it must be re-sensed. Preferably, the information is re-sensed before the predicate ceases to be useful. The rate at which the predicate must be re-sensed to retain its usefulness is its *duty cycle*.

The total information available to the action system is the union of the information in the effective fields of view of the sensors, plus any *a priori* information, plus the information embedded in any plans constructed by the agent. The action system maps this information to an action that is executed by the effectors. The action system can construct plans that use information to achieve the agent's goals; the existence of a plan determines which predicates are *theoretically useful*. The theoretically useful predicates are those which would be useful if the perception system were to extract them. The action system can expand the effective field of view by selecting actions that control the sensors, e.g., to change the camera orientation. These actions can expand the sensors' effective field of view to encompass the environment containing the theoretically useful predicates, thereby converting them into *practically useful* predicates.

7.2. Task-oriented design

The construction of an agent with its effective field of view is most easily thought of in terms of a single task. But any agent in a reasonably complex environment will have multiple tasks to accomplish, and there may be resource conflicts among the tasks which must be resolved. By organizing the components that the agent uses to accomplish a given task into *task-agencies*, these resource conflicts can be recognized and resolved in either an adversarial or a cooperative manner. Complex tasks can be organized into a set of sub-tasks with associated sub-task-agencies. Some subset of the task- and sub-task- agencies are *active* at any given moment, depending on the current situation and goals.

7.3. Marker-based memory systems

The important case in which predicates are obtained through expanded effective fields of view was demonstrated through the use of marker-based representations. Markers represent predicates regarding the location of task-relevant objects in local space. They can be stored and maintained when the objects they mark are outside of the absolute field of view, which gives the marker a temporal extent and expands the effective field of view.

Since markers are stored in an egocentric coordinate system, their maintenance procedure must compensate for ego-motion. If the marked object is in motion and the agent has a model of that motion, the marker maintenance procedure can account for this as well. The certainty of the information stored in the marker may deteriorate over time, so the marker has a duty cycle which dictates when to re-sense the information. The information acquired via perception overrides that in memory, since the certainty of the new information is greater than that of the information in memory.

Different types of markers can be distinguished based on their relationship to a task and the task's agency. At any given time, some task-agencies may be active or inactive, depending on the current situation and goals. The instantiation of a marker may cause a task-agency or sub-task-agency that was previously inactive to become active. Such a marker, from the perspective of the activated task-agency, is an *activating* marker. Once the task-agency becomes active, it may as a part of its activity perform perceptual routines which result in the production of some other marker. Such a marker, from the perspective of the active task-agency which instantiated it, is an *active-only* marker.

Some markers are placed on objects which may constitute a destination considered to be a primary goal; such markers are referred to as *primary-goal* markers. Other markers may be placed on objects that subserve the primary goal, such as an intermediate destination en-route to a primary destination. These markers are called *dependent* markers, since they only are used to help achieve the primary goal. If the primary goal is abandoned, so is are the subservient intermediate goals. The corresponding markers are treated in kind; if the primary goal marker is deleted, so are all the related dependent markers.

It is sometimes useful to place markers on objects for which direct perceptual evidence is weak or even non-existent. If the perceptual system detects some evidence for the existence of an object, but that evidence is weak, the agent may place a *tentative* marker on the supposed location of the object. The tentative marker can be used to direct the collection of further data to sufficiently raise the certainty that the object actually exists in the environment so that the marker is upgraded to a regular marker.

There may be no perceptual evidence at all for the existence of an object, but some higher level process (in the action system) may hypothesize the existence of an object, and instantiate a *hypothesized-object* marker, which again, can be used to direct the collection of the evidence needed to instantiate a regular marker. The main difference between tentative markers and hypothesized-object markers is whether the "evidence" originates in the perceptual system or the action system.

Quantitatively, tentative and hypothesized-object markers are simply markers with low certainty values. Once the certainty is sufficiently high, however, a *qualitative* differ-

ence between these markers and regular markers is realized. Regular markers are those which the agent treats as containing true predicates about the world, and on which actions in direct pursuit of goals can be based. Tentative and hypothesized-object markers are only used as a basis for gathering more information in order to increase the marker's certainty.

7.4. Demonstration of applicability and viability

This dissertation developed the concepts of an effective field of view, task-oriented design, and marker-based memory systems, and then applied these concepts to a number of common tasks. A broad range of tasks was discussed, and several common sub-tasks were abstracted from these various specific tasks. To further pursue these concepts, an in-depth example, Rabbit World™, was developed and implemented.

The task-oriented design methodology was applied to the construction of an agent in Rabbit World, a dynamic, three-dimensional environment in which the perceptual input consists exclusively of a real-time stream of images of the environment from the agent's viewpoint. The agent's task is to collect food while avoiding obstacles and a predator. The perception system was designed an built to deal with real-world problems such as noisy input, occlusion, and a limited absolute field of view. The agent dealt with these problems by using markers to retain information over time and expand the effective field of view.

Tentative markers were used to screen out noise by requiring a percept to be stable over a number of images before it was upgraded to a regular marker. Markers retained information that passed out of the absolute field of view of the sensor in order to give the agent a more complete picture of the local space (i.e., expand the effective field of view) to include objects behind the sensor and occluded objects. Markers were also used to lead the agent through multi-step plans for avoiding obstacles and evading a predator. Markers represented intermediate steps in the plan, such as dependent intermediate-destination markers used to circumnavigate obstacles.

An important observation is that although a few markers can expand the effective field of view, the use of more markers than necessary does not expand the effective field of view purely by virtue of retaining more information in memory. It is often more efficient and more accurate to sense or re-sense the information as needed, rather than maintain a representation in memory.

A relatively complex plan was implemented to evade predators—a plan was executed entirely on the basis of the information embodied by the several markers, e.g., markers for the predator, for the place to look for room to run, for the open space to run, for the obstacle to hide behind, for the location behind the obstacle where the agent will be hidden, and for the intermediate destination on the path to the hiding place. *Implicit sequenc-*

ing moved the plan from one step to the next, based on the situation implied by the marker configuration. The action taken at each instant based on the markers is likely to be appropriate to the current situation in the world, since the perceptual system works to ensure that the state of the markers is firmly grounded in the state of the environment.

The expanded effective field of view resulting from marker use measurably improved the performance of the agent over a purely reactive version, i.e. a memoryless version in which the effective field of view was equal to the absolute field of view. Performance improvements were achieved in all aspects of the agent's performance: food collection, obstacle avoidance, and predator avoidance. Finally, to further demonstrate the viability of the approach in real robots, a marker-based system was implemented which enabled a physical robot to avoid obstacles and reach a destination, even if both the obstacle and destination pass outside of the absolute field of view [10].

The thesis of this dissertation, stated briefly, is that *representation can expand the effective field of view.* As developed in this dissertation, the effective field of view is more than just a perceptual concept; the concept affects and is affected by all of an agent's modules: memory and action as well as perception. To expand the effective field of view is to enable an agent to perform more tasks, more effectively. One of the contributions of this dissertation is the establishment and development of the concept of an effective field of view and its associated concepts (potential and theoretical usefulness, duty cycles, etc.). These concepts enabled the succinct expression of such a consequential thesis statement. More importantly, expansion of the effective field of view is a criterion against which research in perception, representation, and action must be evaluated. Certainly all artificial perception research must be accompanied by an argument as to how the research expands the effective field of view of some agent with respect to some task, in order for the research to be deemed relevant.

My research in local-space representation is accompanied by such an argument. *Use of local-space representations is shown to expand the effective field of view* of an agent, thereby increasing the capabilities of the agent beyond those of agents built using the reactive approach. The contribution of this research in local-space representations is not only that the effective field of view can be expanded in this way, but more importantly, this research demonstrates exactly *how* an agent establishes, maintains, and uses these representations to expand the effective field of view. The techniques developed are applicable any agent using sensors to operate in a dynamic three-dimensional environment, i.e., any mobile robot. The use of these techniques enable an agent to perform tasks unattainable by agents using a pure reactive or a pure classical approach to interaction with the world.

References

[1] Agre, P., and D. Chapman, 1987. “Pengi: an implementation of a theory of activity,” *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-87*), pp. 268-272.

[2] Agre, P., and D. Chapman, 1990. “What are plans for?” *Robotics and Autonomous Systems*, **6**, pp. 17-34.

[3] Allen, J., J. Hendler, and A. Tate, ed., 1990. *Readings in Planning*, Morgan-Kauffman, San Mateo, CA.

[4] Aloimonos, J., 1990. “Purposive and qualitative active vision,” *IEEE 10th International Conference on Pattern Recognition*, pp. 346-360.

[5] Attneave, F., and P. Farrar, 1977. “The visual world behind the head,” *American Journal of Psychology*, **90**(4), pp. 549-563.

[6] Ballard, D., 1989. “Reference frames for animate vision,” *Proc. IJCAI-89*, pp. 1635-1641.

[7] Barnard, S., and W. Thompson, 1980. “Disparity analysis of images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-2**(4), pp. 333-340.

[8] Brill, F., 1994. “Perception and action in a dynamic three-dimensional world.” *Proc. IEEE Workshop on Visual Behaviors*. IEEE Computer Society Press, pp. 60-67.

[9] Brill, F., W. Martin, and T. Olson, 1995. “Markers elucidated and applied in local 3-space,” *Proceedings of the 1995 IEEE Symposium on Computer Vision*, pp. 49-54.

[10] Brill, F., and G. Wasson, 1995. “Virtual markers in the real world,” submitted to the *1996 IEEE International Conference on Robotics and Automation.*

[11] Brooks, R., 1986. “A robust layered control system for a mobile robot”, *IEEE Journal of Robotics and Automation*, **RA-2**(1), pp. 14-23.

[12] Brooks, R., 1990. “Elephants don't play chess,” *Robotics and Autonomous Systems*, **6**, pp. 3-15.

[13] Brooks, R., 1991. “Intelligence without representation”, *Artificial Intelligence*, **47**, pp. 139-159.

[14] Bylander, T., 1991. “Complexity results for planning,” *International Joint Conference on Artificial Intelligence (IJCAI-91)*, pp. 274-279.

[15] Chapman, D., 1987. “Planning for conjunctive goals”, *Artificial Intelligence*, **32**, pp. 333-377.

[16] Chapman, D., 1991. *Vision, Instruction, and Action*, The MIT Press, Cambridge, Massachusetts.

[17] Chapman, D., 1992. “Intermediate vision: architecture, implementation, and use”, *Cognitive Science*, **16**(4), pp. 491-537.

[18] Connell, J.H., 1989. “A colony architecture for an artificial creature”, Tech. report 1151, MIT Artificial Intelligence Lab.

[19] Conway, M., R. Pausch, R. Gossweiler, and T. Burnette, 1994. “Alice: a rapid prototyping system for building virtual environments”, *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, pp. 295.

[20] Erol, K., D. Nau, and V. Subrahmanian, 1992. "When is planning decidable?" *Proceedings of the First International Conference on AI Planning Systems*, pp. 222-227.

[21] Erol, K., D. Nau, and V. Subrahmanian, 1994. "Complexity, decidability and undecidability results for domain-independent planning," *Artificial Intelligence, Special Issue on Planning.*

[22] Fikes, R.E., and N.J. Nilsson, 1971. "STRIPS: a new approach to the application of theorem proving to problem solving", *Artificial Intelligence*, **2**, pp. 189-208.

[23] Firby, J., 1987. "An investigation into reactive planning in complex domains," *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-87*), pp. 202-206.

[24] Gat, E., 1991. "ALFA: a language for programming reactive robotic control systems", *IEEE International Conference on Robotics and Automation*, pp. 1116-1121.

[25] Georgeff, M. P., 1987. "Planning," *Ann. Rev, Comput. Sci.*, **2**, 359-400.

[26] Golden, K., O. Etzioni, and D. Weld, 1994. "Omnipotence without omniscience: efficient sensor management for planning," *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 1048-1054.

[27] Green, C., 1969. "Application of theorem proving to problem solving," *First International Joint Conference on Artificial Intelligence* (IJCAI-69), pp. 219-239.

[28] Grimson, W., A, Ratan, P., O'Donnell, and G. Klanderman, 1994. "An active visual attention system to play 'Where's Waldo?®'," *Proc. IEEE Workshop on Visual Behaviors*. IEEE Computer Society Press, pp. 85-90.

[29] Hayes, P., 1985. "A second naive physics manifesto," in *Formal Theories of the Commonsense World,* J. Hobbs and R. Moore, ed., Norwood, NJ, pp.1-36.

[30] Hayes-Roth, B., 1993. "Opportunistic control of action in intelligent agents," *IEEE Transaction on Systems, Man, and Cybernetics*, **23**(6), pp. 1575-1587.

[31] Hayhoe, M.M., D.H. Ballard, and J.B. Pelz, 1994. "Visual representations in natural tasks", *Proc. IEEE Workshop on Visual Behaviors.* IEEE Computer Society Press, pp. 1-9.

[32] Horn, B.K.P., and G. Schunck, 1981. "Determining optical flow", *Artificial Intelligence*, **17**, pp. 185-203.

[33] Horswill, I., 1993. "Polly: a vision-based artificial agent," *Proc. AAAI-93*, pp. 824-829.

[34] Ikeuchi, K., and B.K.P. Horn, 1981. "Numerical shape from shading and occluding boundaries", *Artificial Intelligence*, **17**, pp. 141-184.

[35] Jones, J.L., 1994. *The Mobile Robot Assembly Guide with Interactive C Manual,* A.K. Peters, Ltd. 289 Linden Street, Wellesley, MA 02181, ISBN 1-56881-303-x.

[36] Kaelbling, L.P., 1988. "Rex: a symbolic language for the design and implementation of embedded systems," *Proceedings of the AIAA Conference on Computers in Aerospace*.

[37] Kaelbling, L.P. and S.J. Rosenschein, 1990. "Action and planning in embedded agents", *Robotics and Autonomous Systems,* **6**, pp. 35-48.

[38] Kosaka, A., and A. Kak, 1992. "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *Computer Vision, Graphics, and Image Processing*, **56**(3).

[39] Laird E., P. Rosenbloom, A. Newell, 1986. *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies*, Kluwer, Dordrecht.

[40] Loomis, J.M., J.A. Da Silva, N. Fujita, S.S. Fukusima, 1992. "Visual space perception and visually directed action", *Journal of Experimental Psychology*, **18**(4), pp. 906-921.

[41] Maes, P., 1990. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT Press, Cambridge, MA.

[42] Marr, D., 1982. *Vision*, W. H. Freeman and Company, New York.

[43] Martin, W., ed., 1994. *Proceedings of the IEEE Workshop on Visual Behaviors*, Seattle WA. IEEE Computer Society Press, Los Alamitos, CA.

[44] Mataric, M. J., 1992. "Integration of representation into goal-driven behavior-based robots," *IEEE Trans. on Robotics and Automation*, **8**(3), pp. 304-312.

[45] McCallum, R. A., 1993. "Overcoming incomplete perception with utile distinction memory," *Proc. 10th Intl. Mach. Learning Conf.*

[46] McCarthy, J.M. and P.J. Hayes, 1981. "Some philosophical problems from the standpoint of artificial intelligence," in *Readings in Artificial Intelligence* (Tioga, Palo Alto, CA), 431-450.

[47] Minsky, M., 1986. *Society of Mind*. Simon and Schuster, N.Y.

[48] Mittelstaedt, H., and M.L. Mittelstaedt, 1982. "Homing by path integration", *Avian Navigation*.

[49] Muller, M., and R. Wehner, 1988. "Path integration in desert ants", *Proceedings of the National Academy of Sciences*, **85**, pp. 5287-5290.

[50] Nishihara, H. K., 1984. "Practical real-time imaging stereo matcher", *Optical Engineering*, **23**(5), pp. 536-545.

[51] Pylyshyn, Z., and R.W. Storm, 1988. "Tracking multiple independent targets: evidence for a parallel tracking mechanism", *Spatial Vision*, **3**, pp. 179-197.

[52] Pylyshyn, Z., 1989. "The role of location indexes in spatial perception: A sketch of the FINST spatial-index model", *Cognition*, **32**, pp. 65-97.

[53] Sacerdoti, E.D., 1974. "Planning in a hierarchy of abstraction spaces", *Artificial Intelligence*, **5**, pp. 115-135.

[54] Sacerdoti, E.D., 1975. "The nonlinear nature of plans", *International Joint Conference on Artificial Intelligence (IJCAI-75)*, pp. 206-214.

[55] Shafer, G., and J. Pearl, ed., 1990. *Readings in Uncertain Reasoning*, Morgan Kauffman, San Mateo, CA.

[56] Shoham, Y., 1993. "Agent-oriented programming," *Artificial Intelligence*, **60**, pp. 51-92.

[57] Simon, H., 1970. *Sciences of the Artificial*, MIT Press, Cambridge, Massachusetts.

[58] Sussman, G., 1974. "The virtuous nature of bugs," *Proceedings of the AISB Summer Conference*.

[59] Tate, A., 1977. “Generating project networks”, *International Joint Conference on Artificial Intelligence (IJCAI-77)*, pp. 888-893.

[60] Tate, A., J. Hendler, and M. Drummond, 1990. “A review of AI planning techniques,” in Knowledge Engineering, H. Adeli, ed., McGraw-Hill, New York.

[61] Tsotsos, J., 1994. “Behaviorist Intelligence and the Scaling Problem”, *to appear in Artificial Intelligence Journal*.

[62] Ullman, S., 1984. “Visual routines,” *Cognition*, **18**, pp. 97-159.

[63] Vere, S., 1983. “Planning in time: windows and durations for activities and goals,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-5**(3), pp. 246-267.

[64] Weld, D.S., 1994. “An Introduction to Least-Committment Planning.” *AI Magazine*, Winter 1994, AAAI Press.

[65] Wilkes, D., and J. Tsotsos, 1994. “Integration of camera motion behaviours for active object recognition”, *Proc. IEEE Workshop on Visual Behaviors*. IEEE Computer Society Press, pp. 10-19.

[66] Wilkins, D., 1984. “Domain-independent planning: representation and plan generation”, *Artificial Intelligence*, **22**, pp. 269-301.

[67] Yanco, H., and L. Stein, 1993. “An adaptive communication protocol for cooperating mobile robots”, From Animals to Animats: International Conference on Simulation of Adaptive Behavior, pp. 478-485.

[68] Yantis, S., 1992. “Multielement visual tracking: attention and perceptual organization”, *Cognitive Psychology*, **24**, pp. 295-340.

Index

Reference Index

Jones, J.L., 1994. The Mobile Robot Assembly Guide with Interactive C Manual, A.K.) 106
Kaelbling, L.P. and S.J. Rosenschein, 1990. “Action and planning in embedded) 8
Kaelbling, L.P., 1988. “Rex: A symbolic Language for the design and implementation) 8
Kosaka, A., and A. Kak, 1992. “Fast vision-guided mobile robot navigation using) 10
Laird E., P. Rosenbloom, A. Newell, 1986. Universal Subgoaling and Chunking: The) 93
Loomis, J.M., J.A. Da Silva, N. Fujita, S.S. Fukusima, 1992. “Visual Space Percep) 10
Maes, P., 1990. Designing autonomous agents: theory and practice from biology to) 4
Marr, D., 1982. Vision, W. H. Freeman and Company, New York.) 8, 29, 38
Martin, W., 1994. Proceedings of the IEEE Workshop on Visual Behaviors, Seattle) 4, 9
Mataric, M. J., 1992. “Integration of representation into goal-driven behavior-based) 11
McCallum, R. A., 1993. “Overcoming incomplete perception with utile distinction) 12
McCarthy, J.M. and P.J. Hayes, 1981. “Some philosophical problems from the stand) 5, 15
Minsky, M., 1986. Society of Mind. Simon and Schuster, N.Y.) 28
Mittelstaedt, H., and M.L. Mittelstaedt, 1982. “Homing by path integration”, Avian) 10
Muller, M., and R. Wehner, 1988. “Path integration in desert ants”, Proceedings of the) 10
Nishihara, H. K., 1984. “Practical real-time imaging stereo matcher”, Optical Engi) 8
Pylyshyn, Z., 1989. “The role of location indexes in spatial perception: A sketch of) 9
Pylyshyn, Z., and R.W. Storm, 1988. “Tracking multiple independent targets: evi) 9
Sacerdoti, E.D., 1974. “Planning in a hierarchy of abstraction spaces”, Artificial Intel) 6, 28, 111
Sacerdoti, E.D., 1975. “The nonlinear nature of plans”, International Joint Confer) 6
Shafer, G., and J. Pearl, ed., 1990. Readings in Uncertain Reasoning, Morgan Kauff) 18
Shoham, Y., 1993. “Agent-oriented programming,” Artificial Intelligence, 60, pp. 51-) 51
Sussman, G., 1974. “The virtuous nature of bugs,” Proceedings of the AISB Summer) 6
Tate, A., 1977. “Generating project networks”, International Joint Conference on) 6
Tate, A., J. Hendler, and M. Drummond, 1990. “A review of AI planning techniques,”) 4
Tsotsos, J., 1994. “Behaviorist Intelligence and the Scaling Problem”, to appear in) 23
Ullman, S., 1984. “Visual routines,” Cognition, 18, pp. 97-159.) 31, 38
Vere, S., 1983. “Planning in time: windows and durations for activities and goals,”) 6
Weld, D.S., 1994. “An Introduction to Least-Committment Planning.” AI Magazine,) 4, 7, 22
Wilkes, D., and J. Tsotsos, 1994. “Integration of camera motion behaviours for active) 19
Wilkins, D., 1984. “Domain-independent planning: representation and plan genera) 6
Yanco, H., and L. Stein, 1993. “An adaptive communication protocol for cooperating) 27
Yantis, S., 1992. “Multielement visual tracking: attention and perceptual organiza) 9, 111