Sensing and Representing Dynamic Enviornments: Judicious Use of Representation Measurable Improves Performance

Frank Z. Brill
Glenn S. Wasson
Worthy N. Martin

Computer Science Report No. CS-96-11
January, 1996

Sensing and Representing Dynamic Environments: Judicious Use of Representation Measurably Improves Performance

Frank Z. Brill, Glenn S. Wasson, and Worthy N. Martin
Computer Science Department, University of Virginia, Thornton Hall, Charlottesville, VA 22901

Abstract

A model for integrating perception and action systems operating in dynamic environments is presented, in which the information gathered by a sensor is maintained over time to expand the sensor's effective field of view. *The information in the effective fields of view of the sensors is used to determine appropriate actions to achieve the agent's goals. An implementation of the approach in dynamic virtual and physical environments achieves improved performance over purely reactive strategies. The virtual environment is instrumented to quantify the performance improvement. We show improved performance for agents using very limited memory structures, and further, that use of additional memory may not improve performance, especially in dynamic environments.*

1. Introduction

In dynamic environments, use of representation can be a risky proposition. In the context of mobile robotics, a representation is an internal model of the state of the world, which the agent examines and/or manipulates to determine an appropriate action. The difficulty with maintaining an internal model of a dynamic world is that a model which was an accurate representation of the world at a given point in time may no longer be accurate at a later time, since a dynamic environment, by definition, changes over time regardless of the action (or inaction) of the agent.

Consider an agent which obtains an accurate representation of a environment at time t_1, spends some time cogitating over this representation to decide upon an action, and then takes the action at time t_2. In a dynamic environment, the world may change between time t_1 and t_2. Even if we (unrealistically) assume that the agent's representation is a complete and accurate reflection of the environment at t_1, and that the agent can determine a "correct" action, this only means that the action the agent takes is appropriate at t_1. But since the action is taken at t_2, it may not longer be appropriate.

Taking inappropriate actions in this way can lead to disastrous results. Consider the case in which a robot is to pick up a piece of trash and put it in the garbage can. The robot has a completely accurate model of the environment, including the location of the trash and garbage can at time t_1. While the robot is planning a path to the trash and then on to the garbage can based on its t_1 representation, a baby comes along and picks up the piece of trash, puts it in its mouth, and sits down in the trash's previous location. At time t_2, the robot executes its plan and throws out the baby.

If the robot accounts for this possibility by sensing more often, the robot may spend all of its time sensing, and none acting, especially if the robot's perception system uses a reconstructive approach to compute the baby's shape, pose, etc. The traditional completely-reconstructive vision approach is unworkable in dynamic environments. The computational complexity of determining the state of the entire environment is overwhelming, and the environment will change before the computation is complete. Designers of perception/action systems in dynamic environments must limit the computation done by the perceptual system so that the computations can be accomplished before the information they compute becomes stale. By limiting the perceptual systems to extracting only the information needed to accomplish a given task, the problem may become tractable.

In the example above, we assumed that the t_1 representation was complete and accurate—however, agents in realistic environments must obtain their representations via their sensors, which are not capable of providing such perfect information, due to noise, occlusion, and a limited field of view. In order to achieve competence in realistic environments, an agent must directly address the problems of acquiring a representation via limited sensing, and maintaining that representation given a dynamic world. Some researchers have suggested eliminating the use of representation altogether [5] which ensures that the agent operates on the basis of its immediate sensor inputs, rather than on a (potentially stale) representation. However, this approach severely limits the capabilities of the agent (results presented in this paper will quantify this claim).

In this paper, we describe an approach in which the agent acquires and maintains *exactly* those representations which facilitate a given task, thereby reducing the maintenance overhead and potential for error to only that which is necessary for the given task. In an implementation of this approach, we will show that a very limited set of memory structures can improve the competence of an agent, and show furthermore that additional representation beyond the minimum needed may not help. The centerpiece of our approach is the concept of an effective field of view, which is described further in section 3. The importance of an effective field of view is that it provides a framework through which the designer of an autonomous agent can strike the appropriate balance between the extremes of purely reactive control and complete-knowledge control for given tasks in dynamic environments.

2. Related work

Agre and Chapman introduced *deictic*, or "pointing" representations to the reactive planning literature [1, 2]. They implemented an agent that used deictic representations to play a video game called Pengi. Rather than identify and label all the objects in the game, as would be the case in the classical planning paradigm, *markers* were placed on only the nearest (or otherwise most task-relevant) items. The markers served as input to reactive-style planning circuitry. Agre and Chapman addressed the problem at the level of intermediate vision, and did not attempt to address problems such as occlusion and the underconstrained nature of early vision. Also, markers were placed only on currently visible objects; no state was retained concerning off screen items [7].

Maja Mataric developed the use of maps in a reactive planning context [11]. Sonar sensors and a low-resolution digital compass were used to identify landmarks and construct a 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 in navigating large-scale space, but are fundamentally different from the local-space representations of our research. 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. Topological maps are useful for navigating the large-scale space, while local representations are used for coping with the immediate surroundings. In this way, the two representations are complimentary.

Andrew McCallum conducted experiments with deictic representations in a virtual environment [12]. He constructed a "go-cart" simulator, for virtual driving to investigate the role of deictic visual behaviors. His work is primarily concerned with the use of the fovea as a marker, and the learning of the visual behaviors. Spatial memory is limited to simple left-right distinctions, whereas we are concerned with the maintenance of multiple markers indicating locations with higher spatial resolution.

3. Effective field of view

Sensor data is *useful* if it can be used to determine an appropriate action. The usefulness of sensor data must therefore be determined in the context of a goal and a set of possible actions. 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 environment and the agent's goals and abilities. We 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 to a specific task.

All sensors have a limited region of space over which they operate. Specifically, a sensor can be thought of as extracting *predicates* concerning the environment from some limited region of space. For example, a visual sensor can only be extract information in the region of space in which the camera is pointing; not the area behind the camera, and furthermore not in the area which is in front of the camera but is so far away that the resolution is insufficient to extract information. We will refer to the region in which a given sensor can extract predicates as its "field of view," and apply this 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 detect properties and thus extract predicates. 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 directly extract predicates.

Furthermore, predicates extracted from real sensors have "certainties" associated with them. Outside of the sensor's field of view, these certainties are at a minimum (i.e., complete uncertainty). We define the *absolute field of view* of a sensor at a given instant of time to be the region of space in which the sensor can extract predicates with some nonzero degree of certainty at that time.

Limiting consideration to only the instantaneous properties in the absolute field of view is overly restrictive, since a predicate determined to be true at a given time will rarely become false immediately after that time. Once again, certainty is an appropriate concept, with certainty decreasing with increased time since extraction.

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 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.[1]

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 information that can be used to accomplish a given task, 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. Upon closer examination, the concept of usefulness turns out to be remarkably complex—see [4] for further discussion.

1. Systems usually considered to be purely reactive may implement such storage by a delay circuit between transducer and decision logic.

By explicitly managing the certainty associated with representations of the environment, and addressing the fact that certainty may decrease over time in a dynamic environment, an agent can ensure that it does not take actions that are inappropriate because they are based on information that is stale by the time the actions are taken. Moreover, when a given task is analyzed, much of the information intrinsic to the environment is simply not relevant to the task. An agent can "deal with" the uncertainty of such information by simply not representing or even extracting this information, and still act effectively. An agent can perform a task by extracting only that information that is relevant to the given task, maintaining that information as it evolves over time, and taking actions based on this limited, yet usefully accurate information. Such information, when extracted via a sensor, forms the sensor's effective field of view.

3.1. The "effective field flashlight"

We can clarify the concept of an effective field of view via the use of the analogy of a flashlight, in which the flashlight represents a sensor, and the cone of light projected by the flashlight 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 we need to complicate this 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 are useful in the context of a given task. Since usefulness is the more important concept, we should instead have brightness encode usefulness, in which case the flashlight leaves a trail of glowing isolated points. 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 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. Points exterior to the box are bright because we are certain of the "non-boxness" of those points. 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 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.

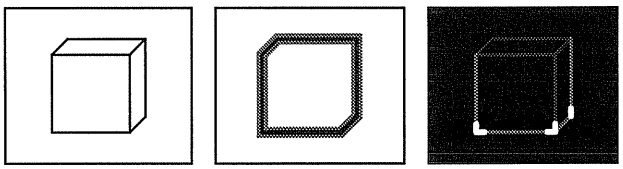

Figure 1: Images of a box (a) in which brightness encodes certainty (b) or usefulness to obstacle avoidance (c)

Having brightness encode usefulness instead of certainty affects the region *inside* the absolute field of view as well as 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 task relevant things about which the agent can draw useful inferences.

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 to do that reliably with a camera.

As we have discussed, the usefulness of a percept depends on the end-to-end operation of the 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 agent design 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.

4. Markers and the effective field of view

A marker is a predicate with temporal extent concerning the location of an object in space. Markers are important because they efficiently expand an agent's effective field of view via their temporal extent.

Recall the "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; the autonomous agent must extract and maintain these predicates. They only light up if the agent extracts them, and they only stay lit up if the agent maintains them. The effective field of view only contains those predicates that are lit up by the agent's 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. An agent 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.

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 more cameras (or sonars, laser range finders, etc.). More, bigger, better sensors come at a cost, and they 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.* For a given 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—by moving that sensor around in the environment [3] and remembering some of what was seen. 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.

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 (e.g., 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.

5. Marker maintenance

Contrary to what been argued by strong proponents of the reactive approach [6], 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. 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.

5.1. Correspondence for visible markers

Objects that are relevant to the task 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 mark an object, (i.e., form a task specific representation of that object), the agent must retain some of the perceptual information associated with the marker from frame to frame, i.e., primary visual cues such as the color and size of an object. The obvious place to keep this information is in the marker. Markers must 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 perceptual input, and will therefore retain any additional information necessary to do so.

Given markers with this additional information, the general marker maintenance procedure at each time frame has two steps. First, all markers "find themselves" in the input. Second, the agent may instantiate new markers for any objects in the new input that aren't already marked.

As we have discussed, memory of the location of objects in a dynamic environment runs the risk of being stale and therefore incorrect, and acting on incorrect information is potentially hazardous. Information derived from current per-

ceptual input is more reliable than that stored in memory, and therefore overrides the information in memory. As a first cut 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 don't. For visual input, the two main cases in which we don't expect to find the marked object are when the object is outside the absolute field of view, 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 absolute field of view, but no visual cue for the object is found.

6. Marker hierarchies

Complex tasks can be decomposed into a set of sub-tasks, which may in turn be decomposed further, until the lowest-level tasks can be carried out directly by the agent's effectors. This decomposition imposes a hierarchy among tasks and their sub-tasks. Tasks may have markers associated with them, e.g., the goal destination in a navigation task is marked. The task hierarchy imposes a similar hierarchy on any markers associated with particular tasks. For example, if there is a marker on a goal destination, it may not be possible to proceed directly to the goal, due the presence of an obstacle. The agent can formulate a sub-task of circumnavigating the obstacle, and then instantiate new markers (e.g., on the obstacle, and on an intermediate destination) that are used to accomplish the sub-task. In general, complex domains necessitate establishing goals subsidiary to the primary goals.

The navigation system, rather than blindly moving towards the destination, should launch perceptual machinery to find any obstacles to its given destination. If an obstacle is found, it is marked with an *obstacle* marker. We say that obstacle markers are *dependent* on the associated primary destination, so that if the destination is ever deleted or changed, the obstacle marker is deleted as well.

If an obstacle marker is instantiated, the navigation system looks for a path around the obstacle. When an appropriate location is found, the navigation agency instantiates an *intermediate-destination* marker which is dependent on the obstacle marker. The navigation system guides the agent towards the intermediate destination just as it would any other destination. Reaching the intermediate destination 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 [9], 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.

7. Validating Implementations

We have implemented a marker based control system that illustrates many of the concepts developed above. Using virtual reality rapid prototyping software developed at UVa [8], we created an environment with trees, rocks, berries, and walls. An agent survives in this environment by eating berries and avoiding obstacles. The environment also contains a predator which actively pursues the agent. The agent has the goals of finding food, avoiding obstacles, and avoiding the predator. This section describes some details of this agent.

The simulation system is a separate process, running on separate hardware from the agent control software. The only information that passes from the simulation system to the agent is a sequence of images of the environment taken from the viewpoint of the agent. The image is 160x120 pixels, and the color is quantized to 64 colors, resulting in a series of images such as those shown in Figure 2 (except in color).

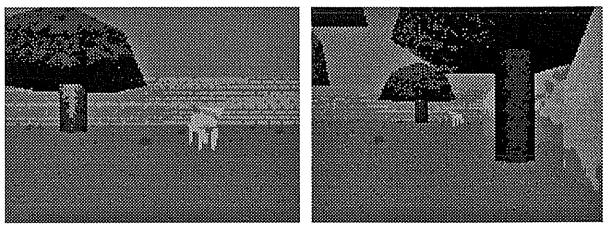

Figure 2: The virtual agent's environment

The agent's control process decides upon an action based on the images, and sends velocity commands to the simulation system, which updates the display based on the new location of the viewpoint. The agent and simulation are completely asynchronous, and run in real time at 10-15 frames per second. The world is dynamic, and the predator pursues the agent regardless of the inaction of the agent. Also, each time the agent consumes a berry, a new berry is generated at a random location, providing an additional source of uncertainty.

Object locations are determined based on azimuth and elevation in the visual field. The system attempts to created and keep marker-based representations for the four closest berries and any nearby predator 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 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. We have to carefully follow the "percep-

tion overrides memory" doctrine with this agent. Due to noise in the image caused by severe color quantization[1] 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 should 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 [10], 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/non-ground boundary (which we refer to as 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. Markers on occluded objects 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 6 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.

7.1. Performance Measurement

The simulation system can easily be instrumented to measure the performance of the agent. We 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' performance on the berry gathering task is a field containing several obstacles. The three agents compared are a completely reactive agent, an agent with four[2] 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.

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 on that run, assuming that the inter-berry distance is roughly normally distributed. Three different runs with different initial berry and agent locations are made for each agent.

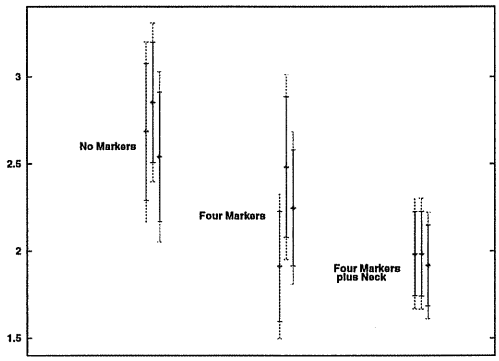

Figure 3: Average inter-berry distance

The poorer performance of the memoryless agent is a result of the agent not taking full advantage of areas in which there is a cluster of several berries. As the agent approaches a cluster, many of the other nearby berries go out of the absolute field of view. This is illustrated in Figure 4, 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 approx-

1. This is not a deficiency of the virtual reality prototyping software, but rather because of the separation of the simulation and the agent control system. We did not try to improve it because such problems are representative of real low-level vision processes.

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

imate locations of these nearby berries, and can turn towards them appropriately. 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.

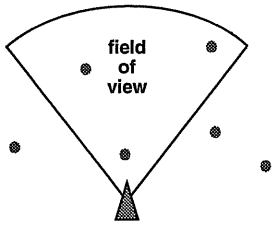

Figure 4: Nearby berries passing out of field of view

Figure 5 shows a comparison of the performance of seven different agents. The left-most 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.

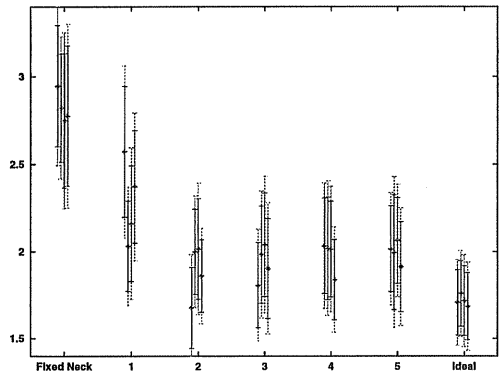

Figure 5: Increasing performance via marker use

The next five agents have functioning necks, and use 1, 2, 3, 4, or 5 markers in a behavior in which they "look around" constantly. Adding the look-around 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. This is because when the agent consumes the berry marked by the first marker, the location of the *next* goal berry is retained in the second marker, and can be pursued immediately. The agent with only one marker must take time to search for the next target once a berry is consumed.

Adding the 3rd, 4th, and 5th markers does not appear to affect performance. Since the agent moves to the marked berries one at a time, the additional markers will not be used until much further in the future. In a sufficiently dynamic and uncertain environment, planning far into the future does not help, since the environment may change or new information will become available which will invalidate the plan.

The right-most agent shown is an "ideal" agent: an omniscient agent that always moves directly to the nearest berry (through obstacles if necessary), and provides a lower bound on the potential inter-berry distance.

More complex evaluation criteria of agent performance can be created by modelling the agent as having an "energy budget" and increasing the agent's energy for successful behavior, while decreasing the energy for undesirable behavior. For example, we might consider a shorter path to be more efficient under an energy-expended metric. In this model, 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. Furthermore, we can penalize the agent for collisions by decreasing its energy when it collides with an obstacle, by an amount proportional to the agent's speed. In this way, we can evaluate the agent's simultaneous berry-gathering and obstacle avoidance performance.

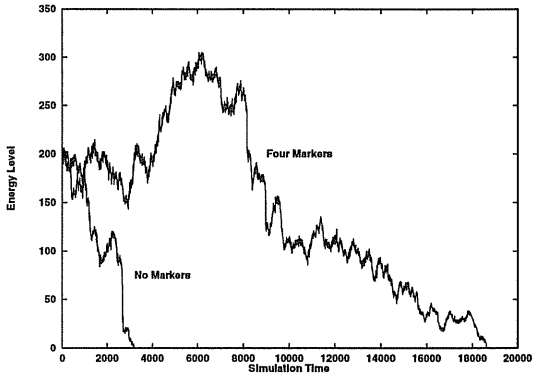

Figure 6: Energy time-series of reactive and marker-based agents

This experiment was carried out for agents using the reactive and marker-based strategies. The parameters of the environment (e.g., berry energy value, collision penalty, etc.) were adjusted to make the environment be sufficiently difficult that all agents tested eventually "died." The agents were otherwise identical (perception processing, speed, energy consumption, etc.), even in their berry-gathering strategy, i.e., to go to the nearest berry they "know about." Again, the marker-based agent knows about more berries, due to the expanded field of view afforded to it via the markers—the reactive agent only knows about the berries in the current absolute field of view. Their obstacle avoidance strategies were similar as well (don't run into any obstacle you know about) however, the marker-based agent marked the edges of the obstacles and used the obstacle avoidance strategy described in section 6. The reactive agent turned to avoid imminent collisions with obstacles it could see, but often collided with objects which were just outside the absolute field

of view, even though the obstacle had been seen recently. The results of this experiment are shown in Figure 6. Each line represents the average performance over five runs with difference initial conditions. On average, the marker-based agents survived more than five times longer than the reactive agents.

8. A physical robot

In addition to the virtual system, we have implemented our approach on a physical robot. The robot, named Bruce, is an MC68HC11 based robot with a single grey-scale camera as its sensor (see Figure 7).

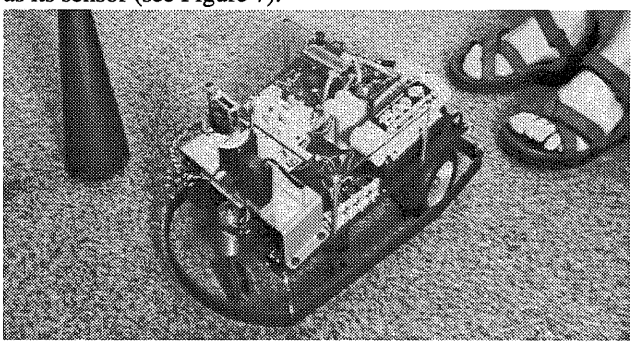

Figure 7: The physical robot

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 8 shows two images acquired from the camera on the robot.

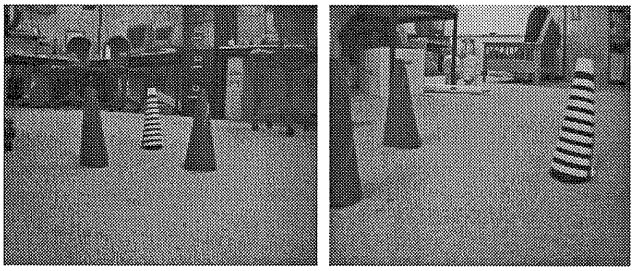

Figure 8: The physical robot's environment

Using a marker system such as that used by the virtual agent, the robot is capable of reaching the goal location, circumnavigating obstacles en route if necessary. The robot searches for the goal and places a marker on it. If necessary, the robot will also mark any obstacles in the direct path to the goal location. If there is a clear path from the robot to the goal (as in the image on the right in Figure 8), the robot can proceed directly to the goal. If the path is blocked (as in the image on the left in Figure 8, where the robot is too wide to fit between the cones), the robot must go around the obstacle.

If the robot must go around an obstacle, it will point its camera at the goal to monitor whether the obstacle is still interposed between itself and the goal (i.e., whether the object is still an obstacle to the goal). As the robot moves around the obstacle, the robot may reach a point where it can no longer turn its camera enough to view the goal. At this point, the robot must rely on its memory for the locations of the objects (i.e., the robot's expanded effective field of view as retained by the markers) to decide when it has cleared the obstacle and can pursue the goal directly.

The robot operates in a dynamic environment. In a tight loop, the robot computes the current position of the markers, and based on their location, decides what action to take. Obstacles can be placed in the robot's path or removed while the robot is in operation, and the robot adjusts to the new situation. If an obstacle is suddenly placed between the robot and its goal, the robot immediately marks the new obstacle and begins moving around it. If the obstacle is removed while the agent is looking towards the goal, the agent immediately notices that the path to the goal clear, and all obstacle markers are deleted.

9. Conclusions

The conventional wisdom is that memory intensive strategies perform best in static, perfect-information environments, while reactive strategies perform best in dynamic, uncertain environments. We have shown that a very small amount of memory, properly managed, can vastly improve the performance of an agent in dynamic and uncertain environments. We have shown further that additional memory beyond the minimum needed to accomplish a given task may not help, especially in dynamic environments.

10. 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. Aloimonos, J., 1990. "Purposive and qualitative active vision," *IEEE 10th International Conference on Pattern Recognition*, pp. 346-360.
4. Brill, F. Z., 1996. *Representation of Local Space in Perception/ Action Systems*. Ph.D. dissertation, University of Virginia.
5. Brooks, R., 1991. "Intelligence without representation", *Artificial Intelligence*, **47**, pp. 139-159.
6. Brooks, R. A., 1990. "Elephants don't play chess," *Robotics and Autonomous Systems*, **6**, pp. 3-15.
7. Chapman, D., 1991. *Vision, Instruction, and Action*, The MIT Press, Cambridge, Massachusetts.
8. Conway, M., R. Pausch, R. Gossweiler, and T. Burnette, 1994. "Alice: a rapid prototyping system for building virtual environments," *Proc. ACM CHI'94*, pp. 295.
9. Georgeff, M.P., 1987. "Planning," *Ann. Rev, Comput. Sci.*, **2**, 359-400.
10. Horswill, I., 1993. "Polly: a vision-based artificial agent," *Proc. AAAI-93*, pp. 824-829.
11. Mataric, M. J., 1992. "Integration of representation into goal-driven behavior-based robots," *IEEE Trans. on Robotics and Automation*, **8**(3), pp. 304-312.
12. McCallum, R. A., 1993. "Overcoming incomplete perception with utile distinction memory," *Proc. 10th Intl. Mach. Learning Conf.*