The Effective Field of View Paradigm: Adding Representation to a Reactive System

Frank Z. Brill, Glenn S. Wasson, Gabriel J. Ferrer, and Worthy N. Martin

Abstract

Dynamic environments pose multiple problems for autonomous agents. Traditionally, agents were designed to develop an internal model of the world and use it to create plans to achieve their goals. Such a strategy fails when the actual environment differs from the stored representation. This realization lead to reactive agents, which stored no model of the world. Today, most researchers would agree that representation is not intrinsically bad, but the cost of keeping an arbitrary amount of representation up-to-date in a dynamic environment outweighs its benefits.

We present a simple task-dependent representation system, called the effective field of view, *which can be used to augment a reactive agent and improve its competence at a variety of tasks. We have embodied this system in two situated autonomous agents. With the first, we quantify the amount of representation necessary for its tasks, as well as the performance gains. With the second, we show how the effective field of view can be used in a hierarchical layered agent architecture. We show that the effective field of view can dramatically increase agent performance.*

1. Introduction

The primary characteristic of reactive systems is a direct connection between sensors and effectors, i.e., there can be no inference in the perception/action loop. Direct connection was initially interpreted as meaning reactive agents should store no representation [11]. However, there are many simple tasks that are either difficult or impossible without representation. For example, suppose you are driving your car and you come to an intersection. To make a left turn, you must determine if the cross-street is clear in both directions. While it may be possible to create a reactive system which can make this determination by rapidly redirecting its sensors, it is more practical to have a representation for “that direction is clear” and maintain the representation as the system looks in a new direction.

Now consider making a left turn in a different situation. You look left and see that the way is clear, then you look right and see a car coming. A truck pulls up next to you in the right lane, blocking your view of the oncoming car. You need to remember that the car was there so that you don’t pull out. Since the truck occludes the car, a purely reactive system (without x-ray vision) cannot use the knowledge of the car’s presence in its action selection. A reactive agent could wait for the truck to leave, just in case the truck was occluding any cars. However, this limits the agent to act only when it can gather *all* information necessary to select an action at the *current instant* in time. An agent with representation can act correctly in this situation (within some time frame) if it remembers the condition of the cross-street before the truck arrived. This is a simple example of a system with representation acting effectively with limited extent sensors and in the presence of occlusion.

We introduce a concept called the *effective field of view*, which allows the agent’s tightly coupled perceptual/action sub-systems to exchange information usefully. The effective field of view is a unified, task-oriented description of the agent’s environment built from current sensor data, the agent’s expectations and a carefully maintained collection of recent sensor values [9].

The remainder of this paper is organized as follows. Section 2 describes the effective field of view paradigm and discusses how its representations are acquired and maintained. Section 3 describes an autonomous agent we have implemented to perform a complex task using the effective field of view. Section 4 provides a quantitative analysis of the performance of the effective field of view agent vs. a pure reactive agent, as well as the performance afforded by varying amounts of representation. Section 5 discusses another autonomous agent which uses the effective field of view as part of a hierarchical layered agent architecture. The final two sections present related work and our conclusions.

2. Effective Field of View

Typically, classical planners (see [29] for an overview) make the impractical assumptions of complete and omniscient sensing. Reactive systems [10] alleviate the problems with these assumptions by always computing the currently necessary information from the current set of sensor values. Purely reactive systems need no assumptions about the ongoing correctness of their sensor data because they place no “temporal extent” on the information extracted from their sensors, i.e. they always recalculate the action selection criteria from the *most recent* sensor values. However, reactive systems do assume that all information relevant to action selection can be extracted from the current sensory view[1].

1. We use the term “view” to describe the area from which a sensor or set of sensors can extract information about the environment, at any one time. This includes sensors which are not traditionally thought of as having a view, e.g. sonar.

For complex tasks, however, certain information may be useful even if it cannot be inferred from the agent's current perceptual input. Consider the task of photographing a rhinoceros in the wild. The photographer must get out of his jeep and move around in order to take all the necessary photos. However, he will not always be able to keep the jeep in sight. Since it is difficult to predict the mood of a rhinoceros, the photographer wants to remain cognizant of the jeep's location at all times. In effect, he assigns a temporal extent to his last perception of the jeep and updates its location, in his mind, as he moves.

This leads us to the concept of *effective field of view.* What we typically think of as the field of view of a sensor, the area from which the sensor can extract information at any given time, we term the *absolute field of view* of that sensor. The effective field of view includes the absolute field of view, as well as limited amounts of task-dependent information which may or may not lie outside the current absolute field of view.

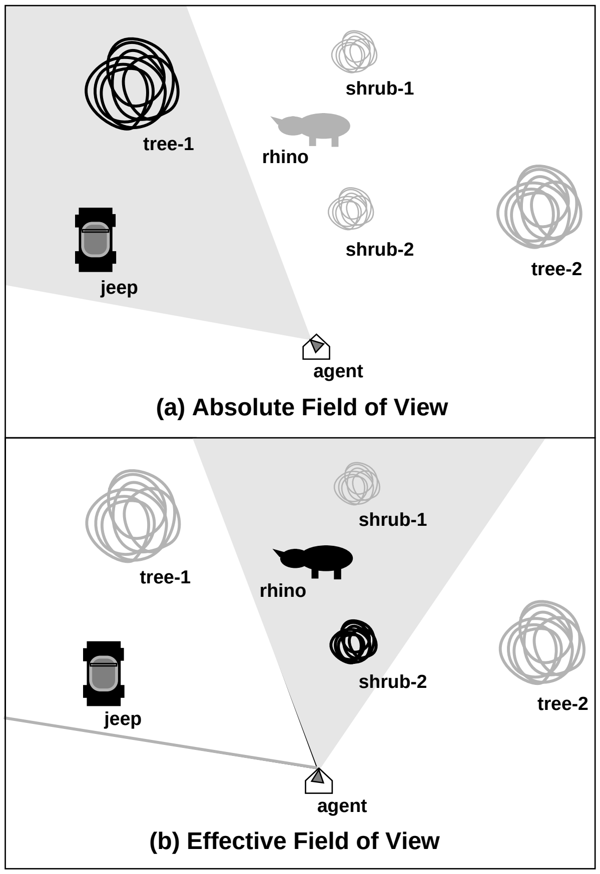

Figure 1: Effective and Absolute Field of View

Figure 1 shows the difference between the absolute and effective field of view. The agent is shown with a pentagon indicating body orientation and a triangle indicating the direction of the line of sight. The shaded area delineates its absolute field of view. The darkened shapes indicate the objects represented in the effective field of view (these came from the area of both the current and previous absolute fields of view). In figure 1a, both the jeep and the tree are in the agent's absolute field of view. We darken the jeep and tree-1 to indicate that the agent can take action based on current sensor readings about them.

In figure 1b, the agent has changed its view, so that the rhino and a couple of shrubs are in the new absolute field of view. Shrub-1 is occluded and so is shown undarkened. As we indicated above, it is in the agent's interest to know where the jeep is while looking at the rhino. We contend that for an agent's perception/action (PA) component(s) to function practically, a full world model (including numerous details about the jeep) is not needed. Rather, a limited amount of representation can be added to the data from the absolute field of view to form an effective field of view appropriate to the current task. Thus, we have darkened the jeep to indicate that the effective field of view in figure 1b includes the jeep, but not tree-1.

The effective field of view augments the absolute field of view by giving temporal extent to limited amounts of information, extracted from the absolute field of view in the recent past. Two fundamental questions are: what information is maintained and for how long? The answer to both of these questions is task dependent, i.e. what information is useful? For the photographer agent in figure 1b, the location of the jeep is useful, while the location of the tree from the absolute field of view of figure 1a is not.

How long information must be represented will depend on the current task and the agent's knowledge of the environment. For example, the photographer has more faith that the jeep will not move from where it was last seen (where it was parked) than that the rhinoceros will not move from where it was last seen. The jeep's representation can safely be given a longer temporal extent than the rhino's representation.

The uniform interface to information in the effective field of view (described below) allows the agent's action selection machinery to consult the effective field of view, just as it would an ordinary sensor with a larger perceivable area.

Since the effective field of view involves the representation of information outside the absolute field of view, we must address the problems of acquisition and maintenance for our system of representation.

The "low-level" problems (such as those encountered in real-time active vision) of acquisition and maintenance of a stored representation are computationally severe for even moderately dynamic environments. We argue that any such representation must be *task dependent* for efficient and effective operation. That is, representation is so inherently difficult to maintain that the agent's current task must determine what should be represented, in order for the representation be as limited as possible. We do not claim to have the optimal representation for every task; rather we claim that under this paradigm an agent's representation can be greatly reduced (relative to a 3-D reconstruction and therefore easier to maintain) by considering the task at hand.

2.1. Markers

Representation in the effective field of view is composed of small structures called *markers*. Markers represent objects which are important to the agent's current task. They are generally thought to contain a 'what' and a 'where' component. The `what` component describes the object's role in the current task. For the task of escaping a charging rhino, the jeep could be associated with a **hiding-place** or **escape-mode** marker, while for the photographing task, the jeep could be associated with an **elevated-platform** marker. The `where` component describes the location of the marked object in some local frame of reference. Markers do not contain global, objective positions because they are local-space representations describing an object's relationship to the current task. Our agent's markers are stored as polar coordinates, *(r, θ)*, with the agent at the origin and the agent's current body facing as the 0° azimuth. Notational convention: throughout the rest of this paper we will use the courier font, e.g. `action`, to indicate components of markers, and bold-face, e.g. **destination**, to indicate values of those components. Also, a phrase such as **hiding-place** marker indicates a marker whose `what` component is **hiding-place**.

Markers also contain an `identity` component which contains information about the perceptual properties of their associated objects. In section 5, we will see that markers can also contain an `action` component.

2.2. Marker Maintenance

Markers describe the location of task dependent objects in relation to the agent. This section describes the issues associated with keeping those positions up-to-date, while details of how our agent performs these functions appear in the next section.

The position of marked objects (the `where` component of markers) must be kept up-to-date as the agent moves throughout its environment and as the environment changes. This is accomplished by compensation for ego-motion and correspondence for objects in the absolute field of view.

The `identity` component of a marker contains information about the perceptual properties of the object with which it is associated. The agent examines the absolute field of view for properties of objects it expects to appear there and computes new positions for the objects it finds. The agent decides if a given object should be in view, based on the object's location (its `where` component), the azimuth of the optic axis of the agent's current absolute field of view, and knowledge of the size of the absolute field of view. Ego-motion transformations are applied to the coordinates of all objects which the agent does not expect to find in the absolute field of view [9]. In figure 1b, as the agent moves toward the rhino, the rhino's position is determined from the absolute field of view, while the jeep's location is calculated from ego-motion transforms.

We associate a confidence with each marker which is expressed in a timer. When the agent detects an object associated with a marker in its absolute field of view, the marker's timer is reset. Once this object moves outside the absolute field of view, the timer begins to count down. When the timer expires, the agent is alerted to re-direct its absolute field of view in an attempt to relocate the object. The timer can be thought of as the agent's current confidence that the position in the marker's `where` component is correct. The values of the timers will depend on the task. So, when fleeing the rhino, the marker associated with the rhino should have a short timer. Monitoring a group of turtles would call for longer timers.

Effectively maintaining a set of markers involves valuing perception over memory, because information derived from perceptual input is more important in the tight sensor/effector loop needed for autonomous action than information stored in memory. For marker representations, one form of this preference for immediate sensor information is the dropping of any marker for which the agent expects to find the associated object in the absolute field of view, but does not. However, this policy must be amended to account for occlusion. The agent must check if an object appears, in the sensory stream, along the same azimuth as the marker, but closer than the object associated with the marker. This computation need only be performed when the agent believes objects should be in the absolute field of view, but their specific perceptual cues cannot be detected.

3. The Agent

We have constructed an autonomous agent which inhabits a virtual world. Figure 2 shows a sample view of the world, which consists of rocks, trees, berries and a predator. The agent's task is to stay alive, by eating berries and avoiding the predator. The agent is equipped with a "neck", which allows its absolute field of view to be panned independent of its body heading. The agent is capable of estimating its own motion based on the passage of time and how it has previously decided to move. In this section, we will describe the system architecture and the agent's design and implementation.

3.1. System Architecture

The system consists of a rendering process and an agent control process. The rendering process creates time-stamped color images from the agent's point-of-view, such as figure 2. These images are quantized, as in figure 3, and transmitted to the control process. The rendered images are the *only* data about the external environment available to the agent. The control process consists of task agencies which perform low-level visual routines on the images and use the results to determine motion commands to send to the rendering process.

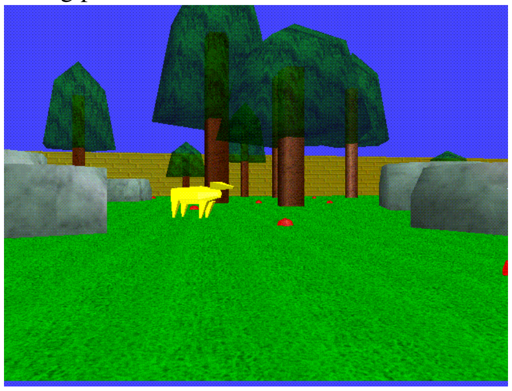

Figure 2. The Agent's Environment

Since the two processes are asynchronous, a non-deterministic amount of time passes between when the control process issues commands and the rendering process generates new views of the environment. The agent possesses a neck angle sensor to detect when a given neck motion command has finished. This allows the agent to know from which direction a given image was taken.

3.2. Agent Design

The agent is designed in a task-oriented manner. The control process contains three "task agencies", or control entities which attempt to actively and/or opportunistically direct the agent towards accomplishing its goals. The agencies used by our agent are FORAGE, AVOID-OBSTACLES and EVADE-PREDATOR. Each task agency examines images from the rendering process for features relevant to its task and outputs a "strength" indicating the relative urgency of acting on the task it oversees.

FORAGE directs the agent toward berries (small red blobs), by marking them in the image. The closest one is then used as the agent's destination. AVOID-OBSTACLES protects the agent by keeping it from hitting obstacles. AVOID-OBSTACLES marks the extent of objects which constitute obstacles between the agent's current location and the agent's destination. EVADE-PREDATOR tries to keep the agent away from the predator. EVADE-PREDATOR marks the predator (yellow areas above "jagged" places in the groundline) and hiding places when trying to escape.

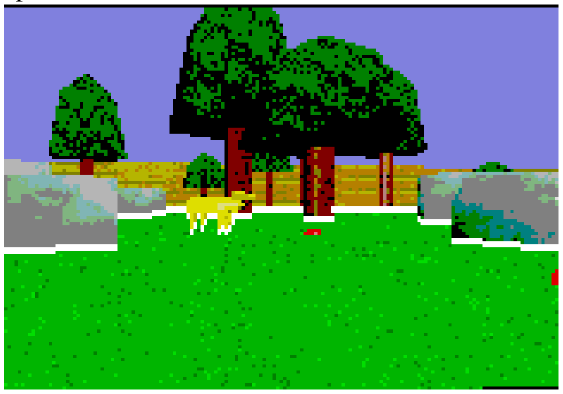

Figure 3. The Ground Line

The overall behavior of the agent emerges from the interaction of the task agencies. The agencies communicate via the markers they maintain. Consider the interaction producing navigation toward a berry. Navigation requires a destination (here, the location of a berry), indicated by a marker and possibly associated obstacle markers (produced by AVOID-OBSTACLES).

FORAGE establishes possible destinations, i.e. berries, by associating a marker with an appropriate image feature. Then, at regular intervals, FORAGE updates the estimated positions of those destinations.

The destination's estimated position is updated through a "hypothesis and confirm (if possible)" process. The hypothesis is created from the previous estimate and proprioceptive data, e.g. the most recent agent movement command and the current image time stamp. If the hypothesized position should be visible in the current absolute field of view, the agent looks for a correspondent in the current image. If a correspondent can be found, the object's image position (based on its azimuth and elevation in the image with the assumption that all objects lie on the ground), is stored instead of its expected location. If the hypothesized position should not be visible, or no correspondent is found, then that position becomes the current estimate.

The agencies use the positions stored in the markers to determine commands to send to the effectors. For example, the agent determines the ground/non-ground boundary [17], referred to here as the groundline, and indicated graphically by a thick white line as in figure 3. AVOID-OBSTACLES processes the groundline to determine possible obstacles. In particular, when a destination marker is created (by FORAGE or EVADE-PREDATOR), AVOID-OBSTACLES is triggered to look for obstacles. If the destination is in the absolute field of view, the agent's direct path to that destination appears as a cone in the image with the destination at its apex and the cone's axis along the destination's azimuth. If any portion of the groundline falls within this cone, the object represented by that portion of the groundline is an obstacle with respect to the current destination. In figure 4a,

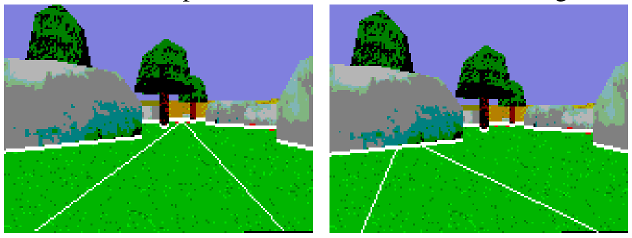
(a) (b)

Figure 4. Detecting an Obstacle

the agent's destination is relatively far away and its entire path cone is unobstructed. Since there is no obstacle for this path, the agent can proceed directly toward its destination along this path. As FORAGE tracks the destination marker, AVOID-OBSTACLES continues to determine if the path is clear.

In figure 4b, however, the cone intersects the groundline, meaning that there is an obstacle[17]. In this case, the agent will establish an intermediate goal based on the location of left and right obstacle extent markers created by AVOID-OBSTACLES.

Each agency can command the agent's effectors, so a means of resolving conflicting signals is necessary. AVOID-OBSTACLES rarely conflicts with the other agencies (unless the agent is very close to an obstacle). FORAGE and EVADE-PREDATOR conflict for control of the agent's destination. FORAGE tries to steer the agent toward berries, while EVADE-PREDATOR steers the agent away from the predator. The agency with the higher output strength (as measured by level of hunger and proximity of predator) will control the agent's destination. However, cooperation can be achieved via the agent's least-commitment strategy. Since it is not necessary to run directly away from the predator, EVADE-PREDATOR will allow FORAGE to select a destination (berry) which is "generally" away from the predator.

3.3. Agent Survival Behaviors

Two behaviors important to agent survival emerge from the interaction of the task agencies. They are berry gathering and predator avoidance. We will discuss each behavior in turn, paying attention to the use of markers in each behavior.

Berries give the agent energy. As the agent's energy ebbs lower, its hunger rises and likewise its desire for food. FORAGE maintains markers on the four closest berries in its effective field of view (reasons for the number 4 are given in section 4). FORAGE selects the closest berry as the agent's destination. Since the closest berry is in the effective field of view, it may be occluded or not currently visible. If the berry is occluded, AVOID-OBSTACLES will detect that the agent is heading for an obstacle, and the agent will steer around it, as discussed in section 3.2.

Predator avoidance is central to agent survival. When a predator is detected and is sufficiently close to be dangerous, the agent tries to escape. Escape proceeds through four stages: looking for a place to run, running, looking for a place to hide and hiding. The current stage is encoded in the currently allocated markers, so the agent can back up a stage or completely abandon the plan at any time, with no additional mechanism.

An escape begins with detection of the predator by the EVADE-PREDATOR agency. The agent has a series of pseudo-markers placed around itself at regular intervals. The timers on these markers are short, causing the agent to glance around more often than is necessary for gathering berries. We term these markers pseudo-markers because they are not attached to any object, and are not maintained as other markers. They provide cues to the agent that particular directions have not been scanned in a while.

A detected predator is marked by a marker with a short timer, so that the agent closely monitors the predator's location. If the predator gets too close to the agent, the agent begins its escape.

The first step is to determine a place to run. This is done by evaluating each pseudo-marker in order to determine the best direction to run. As the agent runs, it must keep looking back at the predator to verify the predator's location and check for serendipitous escape, i.e. predator occlusion.

When the agent reaches its run destination, it must look for places to hide. This proceeds by scanning the groundline for objects and computing a hiding-destination which puts some object between the agent and the predator.

Finally, the agent navigates to the selected hiding place. Once the agent has checked to be sure that the predator is occluded and sufficiently far away, the predator marker will be dropped and the agent will cease trying to escape.

The most important aspect of the predator escaping behavior is how the current step of the escape plan is encoded in the markers, and not kept in some internal program counter. When a predator is marked, is close enough, and is not occluded, the agent knows the predator is dangerous and it must look for a place to run. When the predator is dangerous, and a search marker is allocated, the agent knows it must evaluate the marker's direction to find a place to run. If the predator is dangerous and the agent has arrived at the location designated by a **run-destination** marker, the agent must look for a place to hide. When the agent reaches a **hide-destination**, it must check to see if the predator is still dangerous. If so, it must run more. Note the importance of keeping an eye on the predator. The agent will believe that the predator is dangerous if it was dangerous on the last sighting.

4. Evaluation

In order to assess the advantages of marker based representation, a number of performance measures are needed. The first of these is inter-berry distance. We can say one agent is more efficient than another if it travels a shorter distance to collect the same number of berries. This means that the average distance between berries is lower. Figure 5 shows the average inter-berry distances, on several runs, for

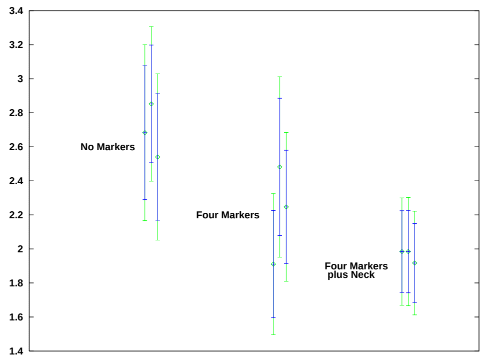

Figure 5. Average Inter-berry Distance

an agent with no markers, an agent with four markers and an agent with four markers plus a neck (allowing the agent's view to change without changing the orientation of its body). The error bars indicate 95% and 99% confidence intervals.

The difference in performance between the agents with markers and the markerless agent can be explained by the fact that the markerless (reactive) agent does not remember the location of close berries, which pass out of the absolute field of view. The performance improvement afforded by the neck is due to the fact that the agent glances around as it collects berries. This expands its effective field of view, allowing it to discover more, potentially closer, berries, as well as berries which may be occluded until the agent is past an obstacle (when the no-neck agent cannot look back).

In order to assess the amount of representation needed for this task, we evaluated inter-berry distance for seven different agent types (see figure 6). The first is a fixed neck reactive agent (no scan). The others have functioning necks and 1, 2, 3, 4 or 5 markers. These agents look around as they gather berries to gain more information about the environment. The "ideal" agent is omniscient in that it always moves to the nearest berry (through obstacles if necessary) and provides a lower-bound on the inter-berry distance.

Adding the ability to look around increases the performance of the marker based agents over the fixed-neck agent. Two markers provide increased performance over only 1 marker. However, additional markers do not appear to measurably effect performance. This is because markers represent useful predicates in the agent's berry gathering "plan". The two marker agent performs better than the one

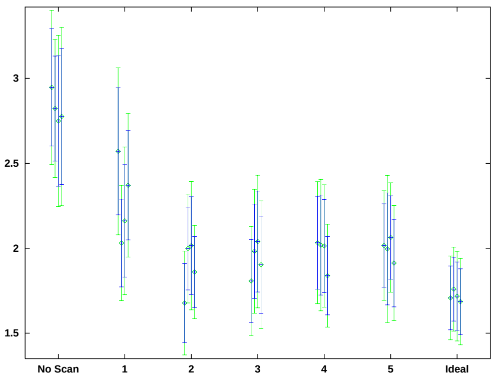

Figure 6. Increasing Performance via Marker Use

marker agent because it already has a destination in mind, once the current plan (to get the closest berry) completes. The agent need not spend the time acquiring a new goal berry. The problem with markers 3, 4, or 5 is that they are useful to a plan which will not be executed until 2, 3 or 4 other berries are eaten. In a dynamic environment, planning too far ahead is not useful as the environment will change or new information will become available and invalidate previous plans. So, the plans for which these markers are useful are seldom executed.

A more complex evaluation criteria can be developed by creating an "energy budget" for the agent. The agent's energy is increased for good behavior (eating berries) and decreased for bad behavior (colliding with obstacles). We consider an agent successful if it manages to maintain a positive energy budget (0 energy constitutes death). A resting energy consumption was added so that the uninteresting sitting-still behavior fails. We select parameters (berry energy, collision penalty, etc.) such that all agents eventually run out of energy. Figure 7 shows the lifetimes of two

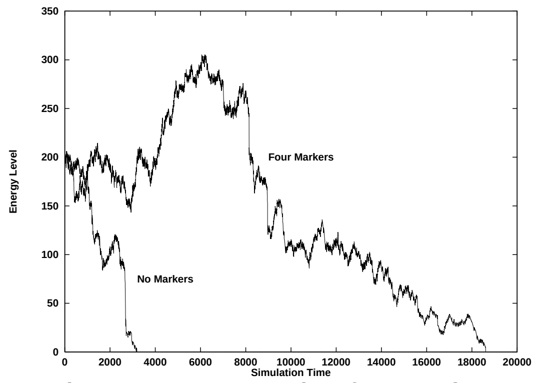

Figure 7. Energy vs. Time for Reactive and Marker-based Agents

agents that are identical (perception processing, speed, energy consumption, etc.), except for their use of markers.

Each line represents the average of five runs with the different environmental configurations.

The four marker agent knows about more berries because it has a larger effective field of view than the reactive agent. Also, the obstacle avoidance behavior which the marker-based agent can use allows it to cleanly avoid obstacles, while the reactive agent often collides with objects just outside its absolute field of view, even though they have been seen recently.

A final important evaluation involved the most dynamic part of the environment, the predator. Measuring the agent's success at escaping was difficult because of the problem of creating a reactive agent which could also escape the predator. Memoryless agents are prohibited from escaping by our previous definition, since hiding from the predator requires an estimate of the predators location, even when occluded. We therefore had to evaluate our agent against a reactive agent with no predator avoidance strategy. This agent will only "hide" if predator occlusion happens by chance.

The metric we use to evaluate predator avoidance is whether or not the agent manages to hide from the predator, given that an "encounter" occurs. An encounter is defined as a line of sight existing from the predator to the agent. The encounter ends when the line of sight path becomes obstructed (an "escape") or the agent is caught (an "eat"). Figure 8 shows the ratio of escapes to eats for the reactive

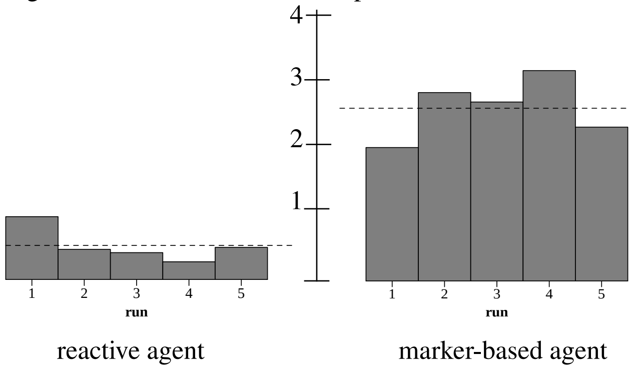

Figure 8. Escape Ratio for Two Agents

agent (chance escapes) and the marker-based agent (marker escapes) for several runs of each agent. The mean for each agent is shown by the dashed line. While better than the reactive agent, the marker-based agent does get eaten. It can fail to detect the predator, fail to find a place to run in time, or fail to hide. However, the marker-based agent did escape roughly 70% of its encounters, as opposed to 30% for the reactive agent.

5. A Layered Agent Architecture

We have also implemented a perception/action system as part of a layered agent architecture on a physical robot, Bruce. Bruce is an MC68HC11 based robot possessing a single color camera on a pan/tilt platform, front and side tactile sensors, and shaft encoders on his wheels. Both video and data transmitters are used as most processing is performed "off-board".

Bruce's architecture consists of a planner layer (DL) at the "top" level, a manager layer called the Task Executor (TE) in the middle, and a perception/action (PA) layer using the effective field of view at the "lowest" level. In this section, we will describe how markers can be used as effective communication primitives between the PA layer and the TE. We will demonstrate this in terms of an implemented application where our robot agent searches its environment for its opponent in a game of tag.

The rules of the tag game are simple. Bruce must search his environment for his opponent, a remote controlled toy truck. Once the truck's location is discovered, Bruce must chase the truck and activate his tactile sensors by touching the truck (a "tag") before the truck reaches a pre-defined "home-base".

5.1. Markers for Layer Interaction

In the virtual world discussed in section 3, we described the maintenance of markers by tracking their associated objects. Obviously, there was some method by which markers initially become associated with objects. We have made this notion more formal by classifying markers as *instantiated* or *uninstantiated*.

An instantiated marker is associated with some object in the agent's environment and is maintained as previously discussed. An uninstantiated marker has not yet been associated with an object and is treated somewhat differently. Uninstantiated markers represent the agent's expectations about the environment, i.e. what objects will be where. This information could come from a map, previous experience, or an outside source (a human). In our system, the TE has a map of the agent's environment and can estimate the agent's current position on that map. Based on this location, the TE can generate expectations about relative locations of various objects with respect to the agent. These expectations can be used to drive the PA layer's actions.

A round of tag proceeds as follows. The planner uses its map of the environment to determine a search path such that every point in the environment will eventually be in Bruce's absolute field of view [30]. An example plan is shown in figure 9. The solid lines represent paths to travel, while the dashed lines represent points where the agent must turn its camera in the direction indicated by the arrow. The agent is shown as a shaded pentagon with the point indicating its direction of motion. The lightly shaded cone indicates the agent's absolute field of view. This plan is passed to the TE for execution. Since we are primarily interested in the communication between the TE and the PA layer, the inner workings of the TE are not detailed here.

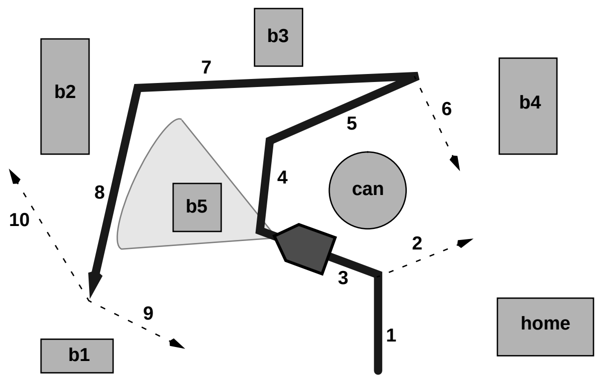

Figure 9. The Search Plan

The TE guides the PA layer by creating uninstantiated markers representing objects in the plan. Each of these markers expresses a goal of the TE to the PA layer. By giving these markers appropriate `action` components (typically **goto** or **look-at** for this application), the PA layer can perform the search task. .

5.2. Searching

To understand how uninstantiated markers are used by the PA layer, we must first understand how markers become instantiated, i.e. associated with an object in the world. The `where` component of an uninstantiated marker represents the TE's estimate of the position of the object that the TE wants the PA layer to associate with the marker. The PA layer can maintain this position with proprioceptive data just as it does with instantiated markers. When an uninstantiated marker's `where` component indicates that its stored position should be within the absolute field of view, the PA layer tries to find a correspondent for the marker using its `identity` routine(s). If a correspondent is found, the marker becomes associated with this object and is now considered instantiated.

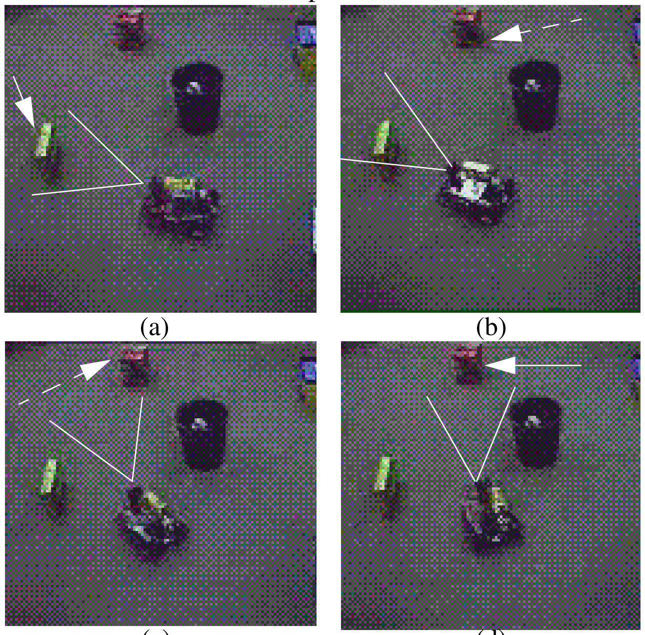
(a) (b) (c) (d)

Figure 10. Marker Instantiation

Uninstantiated markers allow the PA layer to act on information which it has not acquired in its effective field of view. For example, figure 10a shows Bruce in the same location as the agent indicated in figure 9 (near the end of step 3). The white arrows indicate objects associated with markers currently being maintained by the PA layer. Solid arrows correspond to instantiated markers, while dashed arrows correspond to uninstantiated markers. The lines indicate Bruce's absolute field of view.

Near the end of step 3, the PA layer has an instantiated **destination** marker whose `identity` is **b5** and whose `action` is **goto**. At this point, Bruce's current task is **goto destination**, where the exact destination is designated by this marker. The actual **goto** action is carried out by a set of parallel "PA processes", which are activated by the `action` component of the marker. They are similar to the task agencies used by the virtual agent. When Bruce completes step 3, the **destination** marker can be dropped. Now the plan indicates that he needs to go toward object b3. The TE will create a new **destination** marker whose `identity` is **b3**. However, b3 is outside of the current absolute field of view and has never been associated with a marker by the PA layer. Therefore, this **destination** marker is uninstantiated. Figure 10b shows the state of the PA layer's markers at this point.

Based on the TE's estimate of b3's position, Bruce begins to move toward b3 (figure 10c). At some point, b3 comes within Bruce's absolute field of view and the **destination** marker gets associated with it (see section 5.3). This is shown in figure 10d. At this point, b3's location would be maintained in the marker even if b3 went outside of the absolute field of view. The important aspects of this interaction between the TE and the PA layer are that the effective field of view allows the TE to not only provide goals to the PA layer, such as **goto** b3, but to describe important objects for which no sensory information is currently available. Markers describe the objects that are important to the agent's current task and the `action` components of markers describe that task. This is a semantically appealing way of expressing goals, as opposed to parameterizing a skill network [12]. Also, markers allow the PA layer to begin the **goto** b3 task based on a higher layer's expectation and not on sensor values. A traditional reactive system can select actions based only on current sensory input, and so would have no means of selecting appropriate actions to move toward b3.

Another important consideration for our agent's PA layer is the number of markers that it can effectively maintain, i.e. the number it can maintain at a "reasonable" rate. The virtual agent is always maintaining between 5 and 9 markers (depending on whether it is avoiding an obstacle, running from the predator, or just eating berries) These markers are fixed because the agent's task/environment dictate that there will always be 4 closest berries, a predator, etc. Also, since the virtual agent has no executor or planning layer, the PA system must maintain all the information which the agent may ever need.

For the tag game, it is ineffective for the PA system to try to maintain markers for all the objects that it will encounter on its search route. For example, when the agent is executing step 1 in figure 9, it does not need to know about the position of object b2. In fact, the limited accuracy of Bruce's encoders and limited range of his camera make it unlikely that he could successfully maintain the position of b2 through the first 6 steps of the plan until he needed it. For even more complex tasks than tag, in larger environments, it becomes clear that the PA layer can become overburdened with representation. However, the effective field of view is meant to be a representation of local space, while the higher layers of the architecture maintain "global" information. The communication between the TE and the PA layer allows the TE to have a hand in reconfiguring the effective field of view's representation over time.

5.3. Vision

Bruce's vision system is similar to the vision system of the virtual agent. Objects are segmented from the background by finding the groundline [17]. Sufficiently large vertical discontinuities in this line represent the edges of objects. The area between pairs of discontinuities is analyzed by the `identity` routines of the appropriate markers present in the PA layer. Bruce uses visual properties such as color histogram matching [28] to identify objects. Bruce's collection of vision algorithms runs at approximately 15 frames per second on a Datacube MV200.

The constraints of real hardware further reinforce the utility of having an effective field of view. The viewable depth of any physical camera is intrinsically limited, so by using markers the agent can continue to keep track of objects which are not occluded and are still within the camera angular field of view, but too far away to be clearly identified.

6. Related Work

Ullman [32] uses the term "marking" to refer to remembering locations and properties of previously analyzed portions of the visual field. Attneave and Farrar [4] suggest that locations outside the visual field can be marked. Pylyshyn describes a similar concept in his FINST model [23].

Agre and Chapman [1][2] introduced *deictic* or "pointing" representation to reactive planning. Their Pengi system played a video game by placing markers on the most task relevant objects in the game, rather than exhaustively labeling all objects, as in the classical planning paradigm. These markers then served as input to their reactive planning circuitry. 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 and no memory of off-screen objects was retained. More recently, Horswill [18] has made use of deictic representations in a reactive robot.

Arkin's AuRA system [3] contains the ability to instantiate schemas when various sensory events occur, such as the detection of a certain object. This capability is similar to our instantiation of markers (and possible ensuing actions). However, in his system, perceptual schema must be explicitly scheduled by the task designer. In our system, a marker indicates that the appropriate perceptual routine should be executed when the underlying system believes the object is in sensor range. Also, there is no mention of updating the world model once a perceptual schema has detected its features. If the agent's plan was interrupted after a perceptual schema had run, and the agent had to do some other actions, either the original perceptual schema would have to be run again, or the agent would have to store the potential field that the original schema computed. We would argue that the object which caused the potential field should be stored, rather than a field calculated on possibly old information.

Andrew McCallum [20] conducted experiments with deictic representations in a virtual environment. 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.

Other researchers, such as Terzopoulos and Rabie [31], have attempted to create realistic simulated environments inhabited by realistic agents possessing active perception systems. It should be noted that while Terzopolous and Rabie were attempting to model fish biology, it is our aim to study the properties of the effective field of view, and not to model any particular animal.

We believe that benefits of the effective field of view paradigm and the use of markers as communication can be realized within the contexts of a variety of different approaches to agent architectures.

Maja Mataric developed the use of maps in a reactive planning context [19]. 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 that are the focus of this 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 useful for coping with the immediate surroundings. In this way, the two representations are complimentary. In using markers as a means of communication between the perception/action layer and the task executor layer of our three-layer architecture, we take advantage of this complimentary relationship between these different types of representation [33][34][35].

Lynn Stein expanded on Mataric's work in her imagination architecture [27]. Mataric's subsumption-based agent Toto was enhanced with an "imagination" system called MetaToto. While Toto was limited to mapping only places it had actually sensed, MetaToto can provide Toto with a rough "sketch map" (complete with appropriate inputs to Toto's sonar sensors) of locations it has not visited, and Toto will "imagine" travelling through it, adding what it senses to its normal map just as if it were exploring actual space. Toto can then be instructed to visit locations that it added to its map in an "imagined" journey. Our work is compatible with this concept because we are investigating the processing of sensory information into memory structures. While the imagination architecture considers the robot's main architecture to be a "black box", into which the imagined sensory signals are fed, we are developing an internal representation system for the "black box".

Many researchers have investigated the integration of a reactive system with a classical planner. The 3T architecture [5][6] and the ATLANTIS architecture [14] both use Firby's RAP system [12][13] to interface a planner and a reactive system. The reactive layers (analogous to our PA layer) are composed of sets of "skills" which communicate with each other and the upper layers using channels [14] or shared memory [5][12]. Different RAPs enable and disable sets of skills as needed. Since our work is concerned with the level of the architecture "below" the RAP system, it would be fairly straightforward to incorporate a PA system similar to ours into this kind of architecture. In [5][6][14], communication between different skills and between skills and RAPs is unstructured and arbitrary. We have defined a more structured communication mechanism between the TE and PA layers via the passing of markers associated with task-dependent roles. We believe that this interface allows the effective exchange of important information between behaviors or between behaviors and the upper levels of an architecture. This gives the upper layers an efficient and effective paradigm for instructing the PA layer.

Another approach to integrating deliberation and reaction is the Task Control Architecture (TCA) [25]. In contrast to the RAP-based architectures, a top-down approach is taken. The TCA paradigm of *structured control* is to start with planned behavior that is expected to work correctly in nominal situations, and then add reactive behaviors to deal with exceptional situations. Reactive behaviors are typically used for tasks such as monitoring and exception handling. Adding markers to these constructs would be useful for certain tasks. For instance, a monitoring behavior that caused an agent to follow an object upon detection could make use of the effective field of view to handle situations where the object becomes occluded.

The supervenience model of Spector and Hendler [26] consists of a set of communicating levels in which lower levels pass facts about the world to higher levels while higher levels pass goals down to lower levels. Each level in this architecture contains a knowledge base in the form of a blackboard system, with each level having its own uniform knowledge structures. In the implementation of supervenience described in [26] (called APE), the same knowledge structures are used at all levels. We have presented the marker as a knowledge structure for use by a PA layer. We believe that the types of knowledge structures needed to perform the tasks required of higher levels are different than those that are usable by a PA layer in a dynamic environment. These "higher level" structures may contain more information than can be effectively maintained by a PA layer. It is important to design the knowledge structures of the PA layer carefully so that the advantages of the knowledge are obtained while real-time performance remains robust.

Some other examples of agent architectures integrating deliberation and a reactive system include [15] [16] [21] [22] and [24].

7. Conclusions

We have described a concept called the *effective field of view*, which effectively and efficiently integrates task-dependent representation into perception/action systems. A sensor's *effective field of view* is different from what we define as its *absolute field of view*, or the area which the sensor can currently perceive. The effective field of view of a sensor includes both its current absolute field of view and some limited information about objects recently in view. We have characterized this limited information in structures called *markers*.

We have developed an agent with a simple neck-based perception system. The current position of the neck defines the agent's absolute field of view, while representations collected and maintained from recent positions of the neck characterize the agent's effective field of view.

The agent inhabits a dynamic world containing food, obstacles and a predator. We have shown the effectiveness of the effective field of view, for the survival task, by the metrics of inter-berry distance, survival time, and predator escapes. Further, we have shown that adding additional markers to the effective field of view may not increase performance. This supports our claim that representation for objects outside the absolute field of view is important, but that these representations must be minimalist in order that their contribution to agent performance outweighs the cost of their maintenance. We have also implemented a physical agent which uses an effective field of view to play a game of tag with a human controlled opponent.

The contribution of this work goes beyond the simple fact that representation can improve performance. We have described a system of representation and addressed the problems of acquisition and maintenance for a real agent. Further, our work has attempted to quantify the amount of representation needed and the improvement gained by using our form of representation, the effective field of view. This is the only work known to the authors which has attempted to quantify the performance of a system with representation over a pure reactive system. The virtual agent developed here is also the only known agent to have a reactive system augmented with a representation scheme (and no planner) which deals with the realistic problems of limited field of view and occlusion in 3D environments. Our physical agent demonstrates that such a PA system can be effectively used as part of a hierarchical layered architecture. We have shown that markers and the effective field of view form an effective means of communicating semantically meaningful goals between the PA layer and the executor layer.

The effective field of view of an agent's perception system is a useful conceptualization for autonomous systems. It provides an agent's perception/action system with access to task dependent information outside the current sensory "view", extending the agent's capabilities beyond those of pure reactive agents.

8. References

1. Agre, P.E.; and Chapman, D. 1987. Pengi: An Implementation of a Theory of Activity. *AAAI-87*: 268-272.
2. Agre, P.E. and Chapman, D. 1990. What Are Plans For?. *Robotics and Autonomous Systems* 6: 17-34.
3. Arkin, R.C. 1987. AuRA: An Architecture for Vision-Based Robot Navigation, *DARPA Image Understanding Workshop*, Los Angeles, 417-431.
4. Attneave, F.; and Farrar, P. 1977. The Visual World Behind the Head, *American Journal of Psychology* 90(4): 549-563.
5. Bonasso, R.P.; Firby, R.J.; Gat, E.; Kortenkamp, D.; Miller, D.P.; and Slack, M.G. 1997. Experiences with an Architecture for Intelligent, Reactive Agents. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2):237-256.
6. Bonasso, R.P.; and Kortenkamp, D. 1995. Characterizing an Architecture for Intelligent, Reactive Agents. *AAAI Spring Symposium*. http://tommy.jsc.nasa.gov/er/er6/mrl/symposium.html.
7. Brill, F.Z. 1994. Perception and Action in a Dynamic Three-Dimensional World, *Proc. IEEE Workshop on Visual Behaviors:* 60-67.
8. Brill, F.Z., Martin, W.N. and Olson, T.J. 1995. Markers Elucidated and Applied in Local 3-Space. *IEEE Symposium on Computer Vision,* November 1995: 49-54.
9. Brill, F.Z. 1996. Representation of Local Space in Perception/Action Systems: Behaving Appropriately in Difficult Situations. Ph.D. Dissertation, Department of Computer Science, University of Virginia.
10. Brooks, R.A. 1986. A Robust Layered Control System for a Mobile Robot, *IEEE Journal of Robotics and Automation*, RA-2(1):14-23.
11. Brooks, R.A. 1991. Intelligence without Representation. *Artifical Intelligence 47*: 139-159.
12. Firby, R.J.; and Slack, M.G. 1995. Task Execution: Interfacing to Reactive Skill Networks. *AAAI Spring Symposium*. http://tommy.jsc.nasa.gov/er/er6/mrl/symposium.html.
13. Firby, R.J. 1987. An Investigation into Reactive Planning in Complex Domains. *AAAI-87: 202-206.*
14. Gat, E. 1992. Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. *AAAI-92*: 809-815.
15. Hayes-Roth, B. 1995. An Architecture for Intelligent Adaptive Systems. *Artificial Intelligence*, 72: 329-365.
16. Hexmoor, H. H. 1995. Smarts are in the Architecture! *AAAI Spring Symposium*. http://tommy.jsc.nasa.gov/er/er6/mrl/symposium.html.
17. Horswill, I. 1993. Polly: A Vision-Based Artificial Agent. *AAAI-93*: 824-829.
18. Horswill, I. 1997. Real-time Control of Attention and Behavior in a Logical Framework. In *Proceedings of the First International Conference on Autonomous Agents*: 130-137.
19. Mataric, M.J. 1992. Integration of Representation Into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation* 8(3): 304-312.
20. McCallum, R.A. 1993. Overcoming Incomplete Perception with Utile Distinction Memory. *Proc. 10th International Machine Learning Conference*: 431-450.
21. Musliner, D. J.; Durfee, E.; and Shin, K. 1993. CIRCA: A Cooperative, Intelligent, Real-time Control Architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1561-1573.
22. Noreils, F.; and Chatila, R. 1995. Plan Execution Monitoring and Control Architecture for Mobile Robots. *IEEE Transactions on Robotics and Automation* 11(2): 255-266.
23. Pylyshyn, Z.W.; and Storm R.W. 1988. Tracking Multiple Independent Targets: Evidence for a Parallel Tracking Mechanism. *Spatial Vision* 3(3):179-197.

24. Sahota, M. 1994. Reactive Deliberation: An Architecture for Real-time Intelligent Control in Dynamic Environments. *AAAI-94*: 1303-1308.

25. Simmons, R. 1994. Structured Control for Autonomous Robots. *IEEE Transactions on Robotics and Automation*, 10 (1): 34-43.

26. Spector, L.; and Hendler, J. 1992. Planning and Reacting Across Supervenient Levels of Representation. *International Journal of Intelligent and Cooperative Information Systems* 1 (3 & 4):411-449.

27. Stein, L.A. 1995. Imagination and Situated Cognition. Android Epistemology. K.M. Ford, C. Glymour, and P.J. Haynes, eds. AAAI Press/MIT Press. 167-182.

28. Swain, M.J.; and Ballard, D.H. 1991. Color Indexing. *International Journal of Computer Vision* 7(1): 11-32.

29. Tate, A.; Hendler, J.; and Drummond, M. 1990. A Review of AI Planning Techniques, *Readings in Planning*, Allen, Hendler and Tate ed., Morgan Kaufman: 26-49.

30. Taylor, C.J.; and Kriegman, D.J. 1994. Vision-Based Motion Planning and Exploration Algorithms for Mobile Robots. *Workshop on the Algorithmic Foundation of Robotics*.

31. Terzopoulos, D. and Rabie, T.F. 1995. Animat Vision: Active Vision in Artificial Animals. *ICCV-95*: 801-808.

32. Ullman, S. 1984. Visual Routines. *Cognition* 18:97-159.

33. Wasson, G.S. and Martin, W.N. 1996. Integration and Action in Perception/Action Systems with Access to Non-Local Space Information. *AAAI-96 workshop, "Theories of Action, Planning and Robot Control: Bridging the Gap"*: 130-134.

34. Wasson, G.S., Ferrer, G.J. and Martin, W.N. 1997. Hide-and-Seek: Effective Use of Memory in Perception/Action Systems. *First International Conference on Autonomous Agents*. 492-493.

35. Wasson, G.S., Ferrer, G.J. and Martin, W.N. 1997. Systems for Perception, Action and Effective Representation. *FLAIRS-97 Track on Real-Time Planning and Reacting*. 352-356.