Integrating Active Perception with an Autonomous Robot Architecture

Glenn Wasson

Computer Science Department, University of Virginia, Charlottesville, VA 22903

David Kortenkamp

Metrica Inc., TRACLabs, 1012 Hercules, Houston, TX 77058

Eric Huber

Metrica Inc., TRACLabs, 1012 Hercules, Houston, TX 77058

Abstract

Today's robotics applications require complex, real-time, high-bandwidth sensor systems. Although many such systems have been developed, integrating them into an autonomous agent architecture remains an area of active research. We have integrated an active stereo vision system with an autonomous agent architecture using a system of perceptual memory. Perceptual memory is an important class of memory because it is designed for the "behavior-based" portion of the agent's architecture, and not the deliberative portion. This memory maintains current and recent task-dependent perceptual information, as well as expectations about the agent's immediate environment. Our system of perceptual memory is composed of visual primitives from our stereo system, called proximity spaces. Each proximity space represents a virtual fovea or locus of the agent's attention. As an application, we present a robot that uses our system of perceptual memory and proximity spaces to "attend to" multiple humans in a complex and unstructured indoor environment.

Key words: perceptual memory, proximity space, markers, agent architecture, stereo vision

1 Introduction

Many of today's autonomous agent architectures [5,12,24] contain components which interact with the world in a tight perception/action loop. The agent can act effectively if it can process its sensory input in a timely fashion. However,

this task is complicated by the use of high-bandwidth sensors such as vision (compared to sonar, or IR for example). Consideration of the amount of data available from such sensors led to the creation of active or foveal vision systems [3,4]. However, an active vision system cannot be attached to a robot architecture in a "plug and play" manner. Perception needs to know about action and action needs to know about perception. That is, a perception system needs to be directed and controlled by the agent's current action or task. Action, in turn, needs to know about the type and quality of information which is available from perception.

We present an attention-based vision system based on proximity spaces [14,15] which has been integrated into an autonomous agent architecture, called 3T [5]. The novelty of this system is the integration of the sensory systems with the agent architecture via a system of representation which can be effectively manipulated by the perception/action (PA) layer of the architecture, known as the skill layer.

The organization of this paper is as follows. We begin by defining a system of perceptual memory which can be used to integrate a general active perception system into a general agent architecture. We then discuss the needs of vision systems and agent architectures and how they can be addressed by this perceptual memory. Next we describe *our* vision system and architecture. Finally, we show how our system of perceptual memory integrates the two in the context of a task in which an autonomous agent follows multiple humans around in an unstructured environment and "attends to" them.

2 Perceptual Memory

Perceptual memory in an autonomous robot architecture stores task dependent information about recently perceived locations within the agent's environment. Perceptual memory is both local-space and short-term. That is, elements of perceptual memory typically represent task-relevant information about the environment in close proximity to the agent and must be modified frequently to reflect the current state of a dynamic environment. Both of these characteristics are crucial for perceptual memory to remain accurate. Elements of perceptual memory must constantly be checked for validity and modified according to the current world state, hence they are short-term. Information about the environment beyond a certain range from the agent's position cannot easily be verified and should no longer be stored, hence it is local-space.

Our system of perceptual memory is different from representation systems which operate at other levels of autonomous agent architectures [8,10]. It is

composed of small, task dependent structures called "markers" [2,6,28]. The key attribute of markers is that they represent objects in the agent's environment that are important to its current task. Our perceptual memory system is designed to be used by the perception/action layer of the agent architecture, allowing a collection of behaviors to consult the markers to determine which action(s) to take. The markers form the behavior's interface to the sensors. Each marker is kept up-to-date with the state of the world as long as it is in perceptual memory, without explicit action by the behaviors.

An important aspect of perceptual memory is how markers become associated with objects in the environment. The agent's current task specifies certain "roles" which the agent must find objects to fulfill. For example, when the agent's task is "pour a bowl of cereal", it needs to identify objects to serve as *the* cereal box and *the* bowl. Markers exist in 2 states, instantiated and uninstantiated. An uninstantiated marker is one which specifies a role in the agent's current task, but which has not yet been associated with an object in the environment. An instantiated marker has an association with an object.

Markers consist of four components, called 'what', 'where', 'identify,' and 'confidence'. A marker's 'what' component is a task dependent identifier, which specifies the marker's role in the task (i.e., how to act with respect to the associated object). For example, a navigation skill may examine perceptual memory for a **destination** marker and any **obstacle** markers to determine the direction the agent should travel [note: we use the notation "**destination** marker" to refer to a marker whose 'what' component is destination]. One of the key properties of markers which makes them effective for use by an architecture's perception/action layer is their task dependence. Which objects are represented by markers will depend on the task, as will the 'what' components of those markers. For example, a chair may be represented by an **obstacle** marker when the agent's task is to cross the room, but may be represented by a **seat** marker when the agent is trying to sit down. In the first case, a navigation behavior uses the position stored in the **obstacle** marker to determine how to steer the agent, but in the second case, a sitting behavior uses the marker to maneuver itself into the chair.

The 'where' component contains the marker's position in some ego-centric local frame (we use polar coordinates). The 'identify' component specifies how the object associated with the marker can be identified in the visual field. The 'where' and 'identify' components of a marker give an active vision system the necessary information to select an appropriate focus of attention and perform the required processing. In the case of our vision system, the 'identify' component consists of a set of visual behaviors to enable. These behaviors control the processing and placement of the focus of attention to maintain the position of the objects associated with the markers. In order to strike a balance between the efficiency of stored representation and the need

to keep information from becoming stale, markers also have an associated 'confidence' in the information they contain. In our system, this confidence is based on timers which begin counting down whenever the object associated with the marker is not within the visual field. When a marker's timer reaches 0, the agent should no longer believe the information stored there. At this point, the agent can either direct its vision system to re-acquire the associated object or drop the marker from perceptual memory.

2.1 Active Vision and Perceptual Memory

In a dynamic environment, vision systems [3,4] must handle a large volume of data from the visual stream. As a means of coping with the processing demands of such environments, active vision systems process only a subset of the visual stream at any one time. This subset, or fovea, represents the system's current focus of attention and the processing done on this region usually computes task-specific results designed to serve the system's current goal(s).

Our agent has multiple goals with physically separated foci of attention. Therefore, the agent will rarely be able to determine all the information it needs to act from a single image region, or even the entire image. Autonomous agents need a means of remembering information recently determined from areas which are no longer the current focus of attention. They need perceptual memory systems which balance between the efficiency of a foveal, or attentive sensory system and the need to consistently update all stored information relating to the dynamic environment. Our perceptual memory is such a system because the markers represent the foci of attention that are important to the agent's current task. The marker's confidence measure regularly provides a signal to the system to check the validity of the stored information.

A feature which is common to both perception/action layers (of agent architectures) and attentive vision systems is their task dependence. As discussed above, both the placement and processing of the fovea of an attentive vision system are task dependent. Clearly, the information that an agent's perception/action layer needs to select an action depends on the currently active behaviors or skills, which in turn are specified by the higher layers of the architecture. It is this similarity that allows an attentive vision system to be integrated into an agent's perception/action layer via a system of task-dependent perceptual memory.

The layer(s) of the agent architecture above the perception/action (PA) layer (such as the RAP system [9]), can specify the current task by placing information into the PA layer's perceptual memory. The PA layer will act on this

information to try to complete its task. The vision system, in turn, will try to keep the perceptual memory up-to-date with the state of the world. Since the PA layer has a close connection between perception and action, it is natural that active vision should be integrated into our architecture at the skill layer [5]. Figure 1 shows a block diagram of our agent architecture.

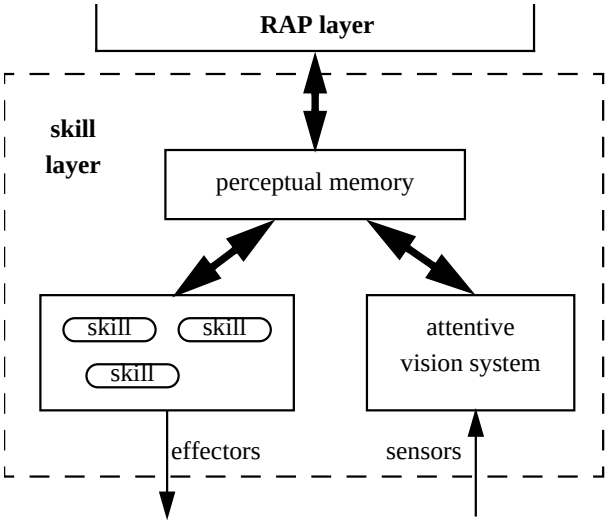

Fig. 1. The agent architecture showing integration via perceptual memory

3 An Active Vision System

Since we will be using the perceptual memory described above as a means of integrating vision into our architecture (described in the next section of this paper), we need to understand the particulars of our vision system to see how perceptual memory will be used.

Our vision system is an attentive (foveated) stereo vision system which confines its processing to virtual, three-dimensional regions of space called "proximity spaces" [14]. A proximity space maps the region of 3-space it "occupies" to corresponding 2-D regions of the left and right camera images. Within these image regions, a set of correlation measurements is made to determine disparity and motion vectors. These vectors serve as input to a set of visual behaviors which control the motion of the proximity space through 3-space. The volume of the proximity space (for this work spherical volumes were used) is represented by sub-volumes, which we call "stacks" [14]. Each stack is an array of correlation measurements made between patches of the Laplacian-of-Gaussian (LOG) [18] filtered left and right images (see figure 2). A breakdown of the stacks is shown in the lower right corner of figure 3. The light grey sphere represents the proximity space, while the dark grey boxes represent the stacks.

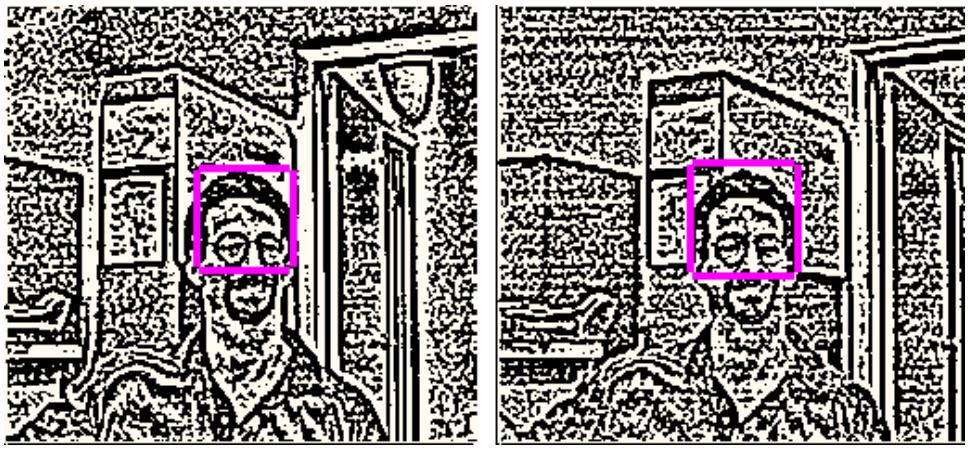

Fig. 2. Left and right LOG images with reference and search regions

The placement of a stack within the proximity space controls which regions of the left and right images are correlated against each other. A stack's offset from the centroid of the proximity space sets the center of the region of the left image (called the reference region). The size of this region is determined by the height and width of the stack. The center and size of the region of the right image used (called the search region) are also specified by the location and size of the stack. However, the depth of the stack also affects the size of the search region. The search region is divided into sub-regions whose sizes are equal to the size of the reference region. These sub-regions are placed at equally spaced offsets within the entire search region. The number of sub-regions is determined by the depth of the stack. A correlation is performed between the reference region and each sub-region of the search region. The offset in image coordinates between the reference region and the sub-region of the search region with which it is correlated represents a disparity, i.e. depth. The correlation value represents the likelihood that something is at that depth within the proximity space. The collection of values (correlations between the reference region and the various sub-regions of the search region) that form a stack can be thought of as a graph of confidence verses depth and that graph can be analyzed for peaks of appropriate strength, uniqueness and distinctiveness. Appropriate peaks indicate the presence of some surface within the volume of space occupied by the stack.

A threshold on the maximum peak for each stack provides the system with a measure of "occupancy" for the proximity space, i.e. if the maximum peak for some stack is above the threshold, then there is some object within that part of the proximity space at the depth indicated by the disparity of the two regions whose correlation produced the maximum peak. In addition to the 9 stacks pictured in the lower right of figure 3, the proximity space also contains a motion correlation stack whose correlation values are computed in a similar fashion to the disparity stacks, but with the current and previous left images being used instead of the left/right stereo pair. The peak values of

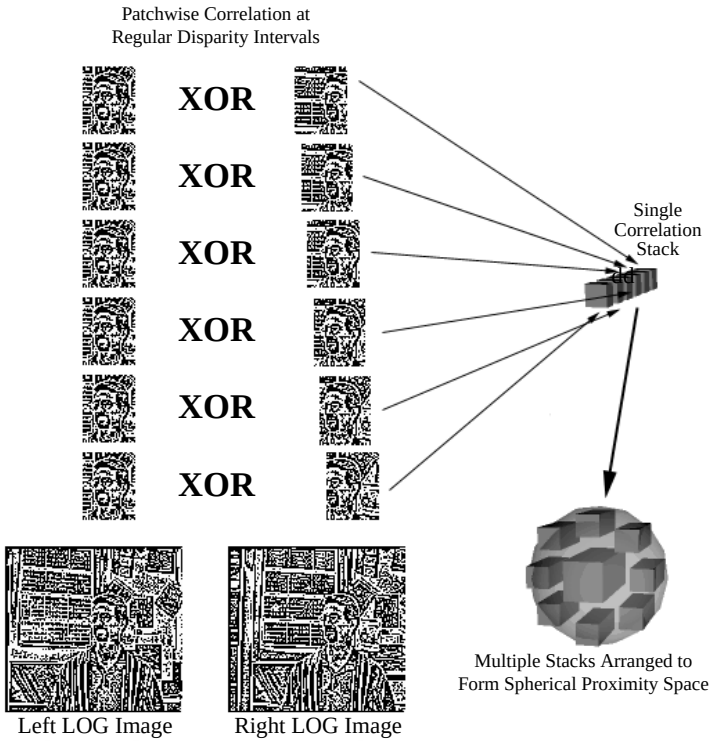

Fig. 3. Proximity space decomposition

the disparity and motion stacks are used by a set of visual behaviors.

3.1 Proximity Space Behaviors

One of our main objectives is to develop a method for gaze control that allows us to acquire and track natural salient features in a dynamic environment. Using the proximity space to focus our attention, we developed a method for moving the proximity space within the field of view. This method is inspired by recent research into behavior-based approaches [7], which combine simple algorithms (called behaviors) in a low-cost fashion.

In our system, each behavior assesses information within the proximity space in order to influence the future position of the proximity space. The information being assessed by each behavior is simply the "texture-hits" (presence of an object at some depth within a stack) and "texture-misses" (no object within a stack) within the proximity space. Based on its unique assessment, each behavior generates a vector, the direction and magnitude of which will influence the position of the proximity space. With each image frame, the behavior-based system produces a new set of assessments resulting in a new set of vectors. When a number of behaviors are active concurrently, their vectors are added together to produce a single resultant vector, which controls

the position of the proximity space. In previous research [15], a set of visual behaviors was developed:

- Follow: This behavior takes an average of several correlation-based motion measurements within a proximity space in order to produce a 2-D vector in the direction of motion.
- Cling: This behavior is attracted to surfaces and produces a vector that tends to make the proximity space "cling" to them. This vector points in the direction of the greatest number of "texture-hits".
- Avoid: This behavior is repulsed by surfaces and produces a vector that tends to make the proximity space stay away from them. This behavior is particularly useful as a front end for obstacle avoidance.
- Lead: This behavior pushes the proximity space towards the intended path of the mobile platform. It also biases the proximity space to maintain a standoff distance from the mobile platform.
- Pull: This very simple but useful behavior produces a pull vector toward the stereo head. This vector tends to move the proximity space toward local depth minima.
- Resize: This behavior influences the size of the proximity space inversely proportionally to its distance from the robot.
- Search: This behavior causes a proximity space to begin systematically searching a given volume of space for texture. It is used to initially locate the object to be tracked and also to re-acquire the object if tracking fails.

Based on the task we want to perform, we activate different sets of behaviors with different parameters. For example, tracking involves the cling, follow and search behaviors. The active set of behaviors determines the overall behavior of the proximity space (or proximity spaces).

4 An Autonomous Agent Architecture

The architecture into which we are integrating our active vision system using perceptual memory is a prototypical hybrid software architecture that has been used to program dozens of different robots [5]. This control architecture, called 3T, separates the general robot intelligence problem into three interacting layers, or tiers:

- A set of robot-specific situated skills representing the architecture's connection with the world. The term "situated skills" [25] is intended to denote a capability that, if placed in the proper context, will achieve or maintain a particular state in the world. For example, grasping, object tracking, and local navigation. The skills are maintained by a "skill manager".

- A sequencing capability which can differentially activate the situated skills in order to direct changes in the state of the world and accomplish specific tasks. For example, exiting a room might be orchestrated through the use of reactive skills for door tracking, local navigation, grasping, and pulling. In each of these phases of operation, the skills of the reactive level are connected to function as what might be called a "Brooksian" robot [7] – a collection of networked state machines. We are using the Reactive Action Packages (RAP) system [10] for this portion of the architecture.
- A deliberative planning capability that reasons in depth about goals, resources, and timing constraints. The deliberative tier was not used in this work.

The architecture works as follows: the deliberative layer (which was not used in the work described in this paper) takes a high-level goal and synthesizes it into a partially ordered list of operators. Each of these operators corresponds to one or more RAPs in the sequencing layer. The RAP interpreter (sequencing layer) decomposes the selected RAP into other RAPs and finally activates a specific set of skills in the reactive layer. Also activated are a set of event monitors which notifies the sequencing layer of the occurrence of certain world conditions. The activated skills will move the state of the world in a direction that should cause the desired events. The sequencing layer will terminate the actions, or replace them with new actions when the monitoring events are triggered or when a timeout occurs.

In the next section of this paper we describe how our perceptual memory and attentive vision system were integrated into the bottom two layers of the existing 3T architecture.

5 An Integrated Architecture

We have integrated a proximity space vision system with the skill layer of the 3T architecture and embodied it in a robot agent (see figure 4). The agent is an RWI B14 with 3 cameras (two grey scale and one central color camera) mounted on a pan/tilt platform. The two outer cameras produce the stereo images used by the proximity space system, while the central camera provides input to a color vision system. These cameras (and the agent's shaft encoders) are the only sensors used by the agent. The agent's task is to monitor the location of multiple (2) humans in its environment and attend to them. The robot could be bringing tools or supplies to workers on a factory floor or serving guests in a restaurant.

Fig. 4. Our robot

5.1 An Agent Application

Our agent's task is to assist multiple people in an unstructured indoor environment. The agent must locate and detect the presence of humans based on initial estimates of their locations supplied by the RAP system. The field of view of the cameras is limited enough that it will seldom include both the humans that the agent is to attend. The agent must monitor and track the position of each human it is assisting. The agent will attend to each person in turn, staying a small fixed distance from them. When the agent decides to move on to a different human, it uses its stored knowledge of the person's last known position in an attempt to locate that individual. Once the human has been located, the agent can attend to it. Figure 5 shows the agent (the short, dark column in the foreground) with two humans. The agent is attending to the kneeling person. Figure 6 shows a screen capture of the vision system output. It contains the left and right LOG images, as well as the output of the color vision system shown in the window in the lower left corner. The white areas in this lower left window denote image regions containing human skin tones.

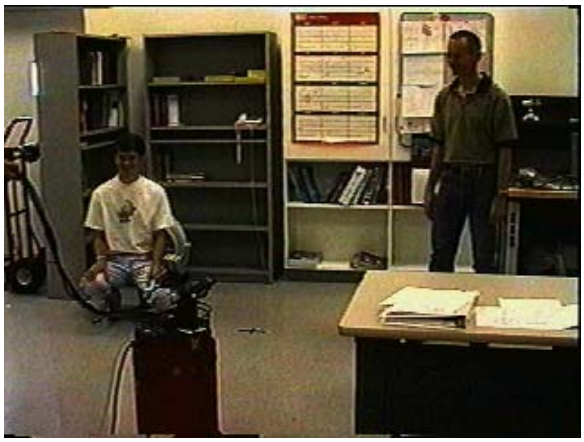

Fig. 5. The agent's task environment

Fig. 6. The agent's view

5.2 Proximity Space Implementation

There are various tradeoffs to be made in the implementation of a proximity space system. For example, the size and shape of the proximity space will determine the computational demands of the tracking task. For this work, we use spherical proximity spaces with a radius of roughly 0.1 meter. The gains for the proximity space motion induced by the cling and follow behaviors is also important. Higher gains for cling will allow the proximity space to "hold on" to less and less of an object, but will make the space more sensitive to noise. Higher gains for follow will allow the space to track larger motions, but cause more "overshoots".

The vision system implemented in this work runs on a 200MHz MMX Pentium-based machine at approximately 10 frames per second. This allows us to track humans at a slow walk, although maximum trackable target speed is dependent on distance from the agent's cameras.

5.3 Perceptual Memory Implementation

Our agent's skill layer interacts with both the proximity space and color vision systems through perceptual memory. In this section, we show how the agent uses its vision systems to instantiate and track markers for use by the skills. In this task each human is represented in perceptual memory by a marker. The RAP system places two **person** markers in the skill layer's perceptual memory. There are two skills which are active in the skill manager: one controls the robot's wheels and the other controls the agent's pan/tilt "neck". These two skills use the 'where' component of the marker representing the human to which the agent is currently attending, to move the agent within a fixed distance of the human. Each marker's confidence measure begins to decrease when its position is outside the agent's current field of view. When the confidence reaches zero, the agent attempts to attend to the human associated with the marker. This means it will direct its cameras to the last known location of the human and attempt to reacquire it (and move toward it).

Each marker in perceptual memory has an associated proximity space that it uses to track its associated human. Maintenance of a marker's 'where' component is as follows. First, its 'where' coordinate is transformed using encoder readings to compute the robot's ego-motion since the last update. If this new position projects within the agent's current field of view, the associated proximity space is activated (and its correlations performed). If the proximity space reports sufficient occupancy, the 3-D position of the centroid of the proximity space relative to the agent is stored as the current 'where' for the marker. Otherwise, the ego-motion-determined position becomes the new stored position.

If the agent can track the objects represented by the markers, the question arises: how is the association between the two made initially? There are several steps in the process that begins when the markers are placed in the skill layer's perceptual memory by the RAP system. These markers are uninstantiated and thus contain only estimated positions for the humans. The color vision system acts as a peripheral vision system that analyzes its entire image to compute coarse positions to be foveated by the proximity space system for further analysis. The color vision system consults perceptual memory to find any uninstantiated markers whose 'where' component falls within the current field of view.

For each such marker, the color vision system examines the image for skin tones (red hues) associated with humans. Image regions with an appropriate response are matched against a simple constraint model of the positions of the human head and arms. Since the color vision system is monocular, when a human is detected, its azimuth and elevation can be determined, but its

depth cannot. We say a marker with an azimuth and elevation, but no depth, is "hypothetically" instantiated. The proximity space system places a proximity space, at the minimum vergable depth, along the vector indicated by the marker's azimuth and elevation. This proximity space then "slides" along the vector performing an analysis for "occupancy" along the way. When the proximity space occupancy is greater than a certain value, the proximity space is considered to have "come to rest" on the object spotted by the color vision system. The associated marker is now said to be instantiated. The normal proximity space tracking behaviors are now enabled and the object tracked in 3 dimensions.

The agent attempts to keep the human in view and tries to stay within a fixed distance of the human. Since there are multiple humans to attend to, the agent must decide how to allocate its resources. This is done based on the confidence value associated with each marker. When a marker's confidence reaches 0, the agent will attend to the object associated with that marker, otherwise, it will continue attending to the same object.

When the agent must redirect its gaze to reacquire a marked object that is outside its field of view, it uses the 'where' component of the marker as a starting point for its reacquisition. If the target is not immediately detectable, the proximity space moves to random locations within a sphere around the object's last known position. At each point a texture analysis, similar to the initial instantiation, is performed. When the proximity space lands on an object, it begins tracking it. In general, the humans do not move far from their last known locations while the agent is elsewhere. If they move too far, the system will be unable to locate them. If this happens, the agent can declare the marker uninstantiated and start the instantiation process again.

The agent's perceptual memory assists in this task in three ways. First, it provides information about objects outside the agent's current field of view. Since the field of view of the agent's cameras is limited, it can seldom (if ever) keep all its targets in view at the same time. Perceptual memory allows the agent to remember the ego-centric locations of a small collection of task-relevant objects.

Second, perceptual memory forms an interface to the sensors for the skills. In our system, the proximity space behaviors require information about the individual stacks, but the navigation and neck skills in the skill manager only require the position of the proximity space as a whole. The perceptual memory provides exactly this information without requiring the skills to know about the details of proximity spaces. In our system, the markers serve as a basis for sensor fusion between the color and stereo vision systems. In general, the 'where' component of markers could represent combined information from all the agent's sensors.

Finally, the perceptual memory provides a communication mechanism for information from the RAP system about the agent's environment beyond the agent's current location. The RAP system initially provides the perceptual memory system with estimated positions for the two humans. The proximity space system subsequently refines those estimates for use by the skills. However, the RAP system could be directing the skill layer to look for certain objects as the agent moves through its environment. For example, if the RAP system believes that the agent is at a particular location on some map, it can create markers for various landmarks that should be visible to the vision system. These markers will be instantiated by the perceptual memory system and then can be used by the skills to direct the motion of the robot. It is important to note that using perceptual memory as the communication channel with the RAP system does not allow arbitrary information to be exchanged with the skill layer. Rather, only information about the portion of the environment that is important to the agent's current task is passed to the skill layer in the form of markers. These markers will be removed when they are no longer relevant to the agent's task. Using perceptual memory as the RAP layer's interface to the skill layer dissuades the agent designer from passing arbitrarily complex data structures to the skill layer (with the expectation that they can be effectively maintained). Since markers represent only a limited amount of information and have a well-defined structure, they cause the designer to consider only what is important for the agent to complete its task(s). This, in turn, leads to (necessary) skill layer efficiency by processing only relevant portions of the environment.

Note that we could have implemented any number of search strategies to allow the robot to find and attend to people without initial estimates of their positions. In fact, the RAP system could send a marker to the skill layer with no 'where' component information, allowing the agent to instantiate that marker on a human at any location. However, the marker's ability to use estimates from the RAP system allows us to take advantage of knowledge when it is available.

5.4 Integration Issues

Using markers to integrate the stereo and color vision systems with the agent architecture brings up some technical issues. We use a simple model of the objects (humans) we are tracking, i.e. skin tone hues in appropriate spatial configurations and then areas of high texture. Errors in the acquisition and tracking process can lead to incorrect marker information and thus incorrect action on the part of the agent.

During the instantiation process, it can take several seconds for the proximity

space to land on the human detected by the color vision system, depending on that human's distance from the agent and the granularity of the search along the skin tone vector. If the intended person moves sufficiently during this process, the proximity space will either find no texture along the vector, or worse yet, will land on some other object along the vector. We handle the first case by having the instantiation process stop at some maximum depth and declare failure (leaving the marker uninstantiated). The second case results from a failure of our (human) model. A more sophisticated system is needed to verify that the proximity space continues to track the entity with which it is meant to be associated. For humans, this could be done, for example, by periodically checking for skin tones within the proximity space or by storing and checking the person's clothing color(s).

Since proximity spaces maintain no model of the objects they track, errors can occur in the tracking process as well. A potential problem situation that arises with multiple markers (and hence proximity spaces) is when one person walks in front of another. We wish to avoid having two markers associated with one human, while the other goes untracked. If the two people are at sufficiently different depths, the proximity spaces themselves can resolve the difference between people because the space that is further from the agent will not be able to correlate the image of the closer person at the expected disparity (though the space may become lost if the further person is occluded for too long).

However, if they people are close enough in depth, the proximity spaces may both track one person. This issue, like the instantiation problems discussed above, should be resolved by the agent architecture. That is, the agent architecture's maintenance process must include checks on entities associated with its markers. Placing this burden on the vision system is unrealistic because it ignores the agent architecture's ability to use task specific knowledge to resolve such problems. A more complex, and/or a periodically checked human model would be an excellent addition to the skill layer's maintenance loop. It is also worthwhile noting that the proximity space to human assignment problem is an instance of the well-known correspondence problem [17], with its equally well-known difficulties (even for humans [1]).

6 Related Work

Much work has been done on active vision systems [3,4] and on active stereo vision in particular [11,14,21,26]. Proximity spaces are themselves based on the work of Nishihara [20], which is based on the work of Marr and Poggio [19]. However, such systems are seldom integrated into an agent architecture. Proximity spaces have been used to grasp weightless objects [14] and interpret

gestures [16], but this work marks the first time they were incorporated into a system of perceptual memory.

There has also been much work on agent architectures [5,7,10,12,24], but none of these contains a "skill layer" memory system for vision integration or communication with the other layers of the agent architecture.

However, there has been some notable work in perception/architecture integration. A common approach is that taken by the Animate Agent Architecture [11] in which the RAP system [10] activates behaviors composed of action routines and the visual routines that serve them. This architecture has no uniform perceptual memory that is accessible to all the behaviors, and information must be transferred among behaviors and between behaviors and the RAP system by separate means. In this architecture, the communication channel to the RAP system is unconstrained. As discussed earlier, the marker communication used in our perceptual memory system provides important limits.

Horswill's Ludwig and Bertrand systems [13] answer queries about the environment by computing all necessary percepts for the query when the query is asked. In this way, a system is built to have the feel of a traditional AI (prolog) database query system, but it need not store a database of facts that will go stale. Such a system can be easily integrated into a purely behavioral system by having the behaviors ask relevant questions of the vision system. This work differs from ours partly in the way it implements markers.

In Bertrand [13], markers are used to store information from one stage of a perceptual computation to the next. The markers are essentially registers used by the visual routine processor (VRP) to store data in retinotopic (image) coordinates. This has two effects. First, Bertrand's markers cannot represent information outside the agent's current field of view. Second, since these registers are used and reused by the VRP, their data does not necessarily persist beyond a particular query. In our work, markers are associated with task dependent entities in the agent's environment. They hold semantic information about some object and its role in the agent's current task. As such, they remain in perceptual memory (being maintained) for the duration of the task. They can also represent information that is not currently perceivable by the agent.

The PURE architecture of Riekki and Kuniyoshi [23] uses markers to represent important entities for the tasks this architecture supports. Stereo vision is also used to determine whether objects are present at the fixation point. In this architecture, the markers are associated with features in the image plane (and therefore cannot be associated with features outside the field of view). They use markers as a means of sharing data between behaviors in a subsumption

style architecture. Their architecture does not have any higher-level, deliberative control, so they do not use markers to communicate with architectural components outside the behavioral system.

Finally, perceptual memory systems with a similar flavor to the one used in this work have been proposed. The concept of markers is most familiar to agent researchers from the work of Agre and Chapman [2]. Their Pengi game was played from a 2-D overhead perspective with no early vision and no occlusion. We have extended the marker concept to operate in 3-D, first-person perspective domains involving both early vision and occlusion. Agre and Chapman's marker concept comes from the work of Ullman [27] and has been studied in the perceptual psychology field by work such as Pylyshyn's FINST model [22] and Yantis's parallel tracking studies [29].

Brill [6] has created a virtual agent that inhabits a world in which it must survive by eating food and avoiding predators. This agent uses markers to represent portions of its local environment outside its field of view. This agent uses only simple behaviors and so needs only a monolithic architecture. The system of perceptual memory presented here was designed to facilitate communication in a multi-tiered architecture.

7 Conclusions

Autonomous robots need powerful sensors. Active vision systems such as our proximity space system can provide information about important aspects of the environment at high speed. However, care must be taken when integrating an attentive vision system into an agent architecture because of the high volume of data which must be processed and the limited area in which the processing takes place at any instant in time.

We have presented a system of perceptual memory, based on markers, which allows us to retain sensor input from the proximity spaces over time. The perceptual memory system attempts to compromise between the efficiency of foveated processing and the need for a high level of maintenance on the information contained in the markers via a confidence measure.

Our agent performs its task effectively in a complex and unstructured environment. The proximity space system deals with the difficulties of the environment while keeping the perceptual memory accurate.d The perceptual memory provides just the information which is needed for the skills to accomplish this task (and many others, we believe).

References

[1] J.K. Aggarwal, L.S. Davis, and W.N. Martin. Correspondence processes in dynamic scene analysis. *IEEE Special Issue on Image Processing*, 69(5):562–572, 1981.

[2] P.E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proc. AAAI-87*, pages 268–272, 1987.

[3] J. Aloimonos. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

[4] D.H. Ballard. Animate vision. *Artificial Intelligence*, 48(1):57–86, 1991.

[5] R.P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

[6] F.Z. Brill, G.S. Wasson, G.J. Ferrer, and W.N. Martin. The effective field of view paradigm: Adding representation to a reactive system. *Engineering Applications of Artificial Intelligence Special Issue on Machine Vision for Intelligent Vehicles and Autonomous Robots*, 11:189–201, 1998.

[7] R.A. Brooks. A robust layered control system for a mobile robot. *Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[8] C. Elsaesser and R. MacMillan. Representation and algorithms for multiagent adversarial planning. Technical Report MTR-91W000207, MITRE Corporation, 1991.

[9] R.J. Firby. An investigation into reactive planning in complex domains. In *Proc. AAAI-87*, pages 202–206, 1987.

[10] R.J. Firby. *Adaptive Execution in Complex Worlds*. PhD thesis, Department of Computer Science, Yale University, 1989.

[11] R.J. Firby, R.E. Kahn, P.N. Prokopowicz, and M.J. Swain. An architecture for vision and action. In *International Joint Conference on Artificial Intelligence*, pages 72–79, 1995.

[12] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proc. AAAI-92*, pages 809–815, 1992.

[13] I. Horswill. Visual architecture and cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2):277–292, 1997.

[14] E. Huber. Object tracking with stereo vision. In *Proc. AIAA/NASA Conference on Intelligent Robots in Factory, Service and Space - CIRFFSS '94*, pages 763–767, 1994.

[15] E. Huber and D. Kortenkamp. Using stereo vision to pursue moving agents with a mobile robot. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2340–2346, 1995.

[16] D. Kortenkamp, E. Huber, and R.P. Bonasso. Recognizing and interpreting gestures on a mobile robot. In *Proc. AAAI-96*, pages 915–921, 1996.

[17] D. Marr. *Vision*. W.H. Freeman and Company, 1982.

[18] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B207:187–217, 1980.

[19] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London*, B204:301–328, 1979.

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

[21] T.J. Olson and D.J. Coombs. Real-time vergence control for binocular robots. *International Journal of Computer Vision*, 7(1):67–89, 1991.

[22] Z.W. Pylyshyn and R.W. Storm. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial Vision*, 3(3):179–197, 1988.

[23] J. Riekki and Y. Kuniyoshi. Architecture for vision-based purposive behaviors. In *Proc. IROS-95*, pages 82–89, 1995.

[24] R. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43, 1994.

[25] M.G. Slack. Sequencing formally defined reactions for robotic activity: Integrating raps and gapps. In *Proc. SPIE Conference on Sensor Fusion*, 1992.

[26] T. Uhlin, P. Nordlund, A. Maki, and J.O. Eklundh. Toward and active visual observer. In *Proc. ICCV-95*, pages 679–686, 1995.

[27] S. Ullman. Vision routines. *Spatial Vision*, 18:97–159, 1984.

[28] G.S. Wasson, G.J. Ferrer, and W.N. Martin. Systems for perception, action and effective representation. In *Proc. FLAIRS-97 Track on Real-Time Planning and Reacting*, pages 352–356, 1997.

[29] S. Yantis. Multielement visual tracking: Attention and perceptual organization. *Cognitive Psychology*, 24:295–340, 1992.