Integrating a behavior-based approach to active stereo vision with an intelligent control architecture for mobile robots

David Kortenkamp, Eric Huber and Glenn Wasson

Metrica Inc.
Texas Robotics and Automation Center
1012 Hercules
Houston TX 77058
kortenkamp@jsc.nasa.gov

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 robot architecture remains an area of active research. In this chapter we describe our behavior-based active stereo vision system and how we integrated this system into a hybrid reactive/deliberative robot control architecture. The integrated system is used to perform tasks such as pursuing moving agents and attending to several agents at the same time.

1 Introduction

Our goal is to build a mobile robot that can interact with people to help them perform tasks. This interaction must be natural and driven by the robot's own autonomous goals and behaviors. We believe that this can only be accomplished by coupling a real-time, high-bandwidth sensory system with an intelligent control architecture. If either is lacking, the robot will not be capable of performing interesting tasks in complex and dynamic environments. Many important research issues can be explored with a coupled system of this kind, including: integrating independent motion of the robot's head, torso and wheels; deciding which aspects of robot motion are controlled by which software elements; and using active vision techniques to abstract critical information from the high-bandwidth sensory system in real-time.

The mobile robot system described in this chapter uses a stereo-based active vision system to accomplish several different tasks, including pursuing other agents and obstacle avoidance. The vision system and the mobile robot are under the control of an intelligent control architecture, which can interpret sensory data in task contexts. This integration of high-bandwidth sensing and intelligent control produces a highly reactive, goal-driven robot system.

1.1 Approach overview

Metrica Incorporated has, over the last several years, developed a real-time, active stereo vision software that provides fast and robust disparity information

[14, 15]. Our innovative method concentrates system resources in cubic volumes of space which we call *proximity spaces*. Within the bounds of this space an array of stereo and motion measurements are made in order to determine which regions of the space are occupied by surface material, and what the spatial and temporal disparities are within those regions.

Metrica Incorporated has also developed an intelligent robot control architecture called 3T [4]. The architecture combines a reactive control subsystem with a deliberative planning system, both mediated by a middle layer sequencer based on the Reactive Action Packages (RAP) system. This allows for long-range planning to take place while, at the same time, the system can react to immediate environmental events.

We believe that the proximity space method is particularly well suited for integration into the 3T architecture. Each proximity space acts as a virtual agent, designed to achieve localized perceptual goals. To support more complex robotic tasks, sets of proximity spaces can be organized to act in unison to reveal the most pertinent information about the structure of the environment. The 3T architecture is designed to organize, prioritize, and budget perceptual skills such as those of a proximity space. Sets of proximity spaces fit ideally into this architecture at the lowest level of the architecture in the form of skill sets.

1.2 Related work

Since our work integrates active vision and intelligent architectures, we will look at related work in both of these areas and in the area of integrating perception and intelligent architectures.

Active vision The active vision paradigm was espoused by Ballard [2] as a way to overcome the computational complexity of reconstructing a scene from a single image. In active vision, only a small portion of the visual field of view is analyzed at any given time and this analysis is performed many times per second on successive frames of the image. Early active vision systems [18] were promising, but typically too brittle and slow for practical application. In 1992 Keith Nishihara developed the PRISM-3 high speed stereo vision system [20], an embodiment of his Laplacian of Gaussian sign correlation theory, which itself is an extension of Marr and Poggio's classical zero-crossing theory [19]. Our work in this proposal builds directly on the work of Nishihara.

Hybrid architectures Hybrid architectures refer to the class of robot architectures that attempt to integrate reactivity and deliberation. A first step towards the integration of reaction and deliberation was the RAPs system of Jim Firby [9]. In his thesis [10], we see the first outline of an integrated, three-layer architecture. The middle layer of that architecture and the subject of the thesis was the Reactive Action Packages system (RAPs).

Independently and simultaneously, Pete Bonasso at MITRE, unaware of Firby's work, devised an architecture that began at the bottom layer with robot

behaviors programmed in the Rex [21] language as synchronous circuits. These Rex machines guaranteed consistent semantics between the agents internal states and that of the world. The conditional sequencer was a reaction plan [22] implemented in the GAPPs language [16], which would continuously activate and deactivate (set enabling "wires" to high states and low states) the Rex skills until the robot's task was complete. This architecture was used successfully in a number of experiments with underwater robots [3].

An alternative hybrid architecture to integrate deliberation and reaction was also being proposed by Erann Gat at the NASA Jet Propulsion Laboratory for control of a Mars rover. This architecture was called ATLANTIS [12] and contained RAPs and robot behaviors written in Gat's Alpha circuit language. Numerous other tiered architectures have been developed, Bonasso et al [4] gives a comprehensive overview of many of these and compares them to 3T.

Integrating perception Agre and Chapman [1] presented a novel integration of perception and action using markers in their Pengi system. Their system and ours differ in that their agent operated from a 2-D overhead prospective with no occlusion and no early vision (they directly accessed their simulation's data structures). Our approach operates from a 3-D first person perspective and must deal with issues of occlusion, a limited field of view, and early vision.

Jim Firby et al [11] have proposed an architecture for vision and action that uses the RAPs system (the middle layer in our architecture). They have successfully incorporated gesture recognition and color histogram-based object recognition into their robotic tasks. A three tiered architecture at the University of Virginia [23] has incorporated perceptual markers into the bottom two tiers. Their use of markers is very similar to what we propose in our architecture. Ian Horswill has also looked at integrating vision and architecture [13] within the context of a purely behaviorist system.

2 A Behavior-based Approach to Active Stereo Vision

In order to efficiently process the enormous amount of information available from stereo cameras, we use techniques that have recently been developed by the active vision research community [2, 7]. In particular, we address the issue of *gaze control*, i.e., where to focus attention and visual resources. Figure 1 gives an overview of the software modules comprising our vision system. To summarize the figure, left and right images enter the system from the cameras. A Laplacian of Gaussian convolution is performed on the image data streams as they are simultaneously acquired. Only the sign of the LOG output is stored in memory. Then, a search is performed in which a patch from the left LOG image is compared with a portion of the right LOG image, producing correlation measurements. This search produces a series of correlations from which the strongest (the "peak") is chosen as the best. At the same time, the right LOG image from the frame before is compared with the current right LOG image to measure motion. This correlation data is used to assess information within a

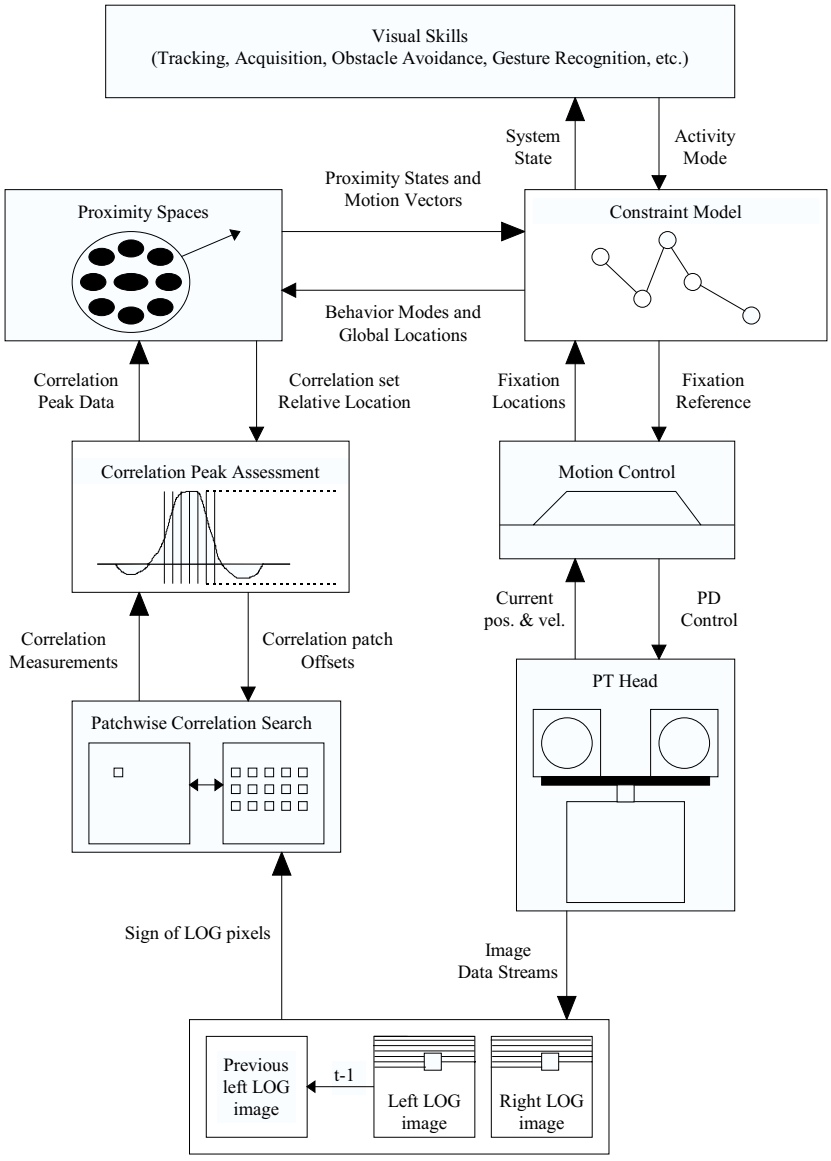

Fig. 1. The software modules comprising our vision system, with arrows denoting flow of information. Large arrowheads represent data flow, small arrowheads represent command flow.

bounded volume of space called a proximity space. Each proximity space is controlled by a set of behaviors and, possibly, by a constraint model. The location of the proximity space within the field-of-view is used as a reference to control the pan-tilt-verge head. Visual skills interpret the proximity space location with respect to the task being performed. Each of these will be described in more detail in the following sections.

Fig. 2. A stereo pair of LOG images with the proximity space.

2.1 Key components

The key components of our active stereo vision system build upon each other. At the lowest level is convolution.

Convolution The left and right grayscale images are convolved by a Laplacian of Gaussian (LOG) operator as they are acquired. In order to minimize computation, a 7x7 pixel LOG kernel is approximated by decomposition into a 3x3 Laplacian kernel followed by two 3x3 Gaussian kernels. Only the sign of the LOG is retained after the third stage of convolution. Sign of LOG convolution normalizes the image and brings out salient features.

Patchwise Correlation The foundation of our approach is the correlation of two dimensional regions (patches) of binary LOG images. These patches are matched pixel for pixel using an XOR operator and the total number of matches are summed. The correlation sum for a pair of patches is normalized to produce a net value between zero and one. White noise images will typically correlate half of the time thus producing correlation values fluctuating around 0.5.

Proximity spaces are composed of large numbers of patchwise correlations between two images. In order to search for surface material (visually textured

regions of an object) within a given volume correlation is performed between patches in right and left images. Right and left camera images, which are taken from different perspectives, provide the foundation for 3D stereo measurements. In order to determine the motion of surface material, correlation is performed between patches in the current and previous images from the same camera.

Fundamentally, the number of pixels that correlate between two images gives some clue as to the likelihood that the same surface region is imaged in the each of the two patches. The larger a patch is, the greater the likelihood that it will uniquely correlate with its "true match" in the other image. Larger patches require more computation to correlate and smaller patches are more likely to miss match. In order to find an appropriate match, a number of equally spaced patchwise correlations are performed.

Correlation Stacks Object surfaces can be searched for, in three dimensions, by performing a number of patchwise correlations between right and left image pairs. If a patch is chosen in the right image (reference image) it can be searched for in the left image (search image). For stereo measurements, the reference image patch is correlated against patches of equal size in the search image. Search patches are sampled a regular intervals horizontally in the image. The difference between a patch's horizontal location in the reference image and the horizontal location of the patch it correlates best with in the search image, is referred to as the search disparity. The disparity of a search relates directly to the distance between the imaged object and the camera pair. Thus a set of stereo correlations made at a number of disparity intervals, constitutes a search for surface material in depth. In this way correlations in 2-dimensional images are used to provide 3-dimensional measurements in the real world. We call this set of correlation measurements, a correlation stack (see Figure 3). How ordered sets of these stacks are used to build a proximity space will be described in the Section 2.1.

Evidence of surface material (an object) within a stack is determined by analysis of its correlation values. This set of correlation values reveals much about the environment within a stack. Analysis of this data usually begins by finding the greatest correlation value. The greatest correlation value is further tested to determine if it reveals a adequate "peak" in the correlation set. The peak is interpolated to refine its value and disparity to sub-pixel accuracy. This peak value is then filtered to ensure that it is "strong". Correlation values near the peak value are also analyzed to ensure that the peak is "distinct." The highest peak is then compared to the next highest peak in the set in order to ensure that it is "unique." This filtering process dramatically improves the reliability of correlation matches by reducing the effects of noise and repetitive structure in the environment. Filtering, as well as all other aspects of the correlation process compose the inner workings of the proximity space providing a useful abstraction to the intelligent control architecture.

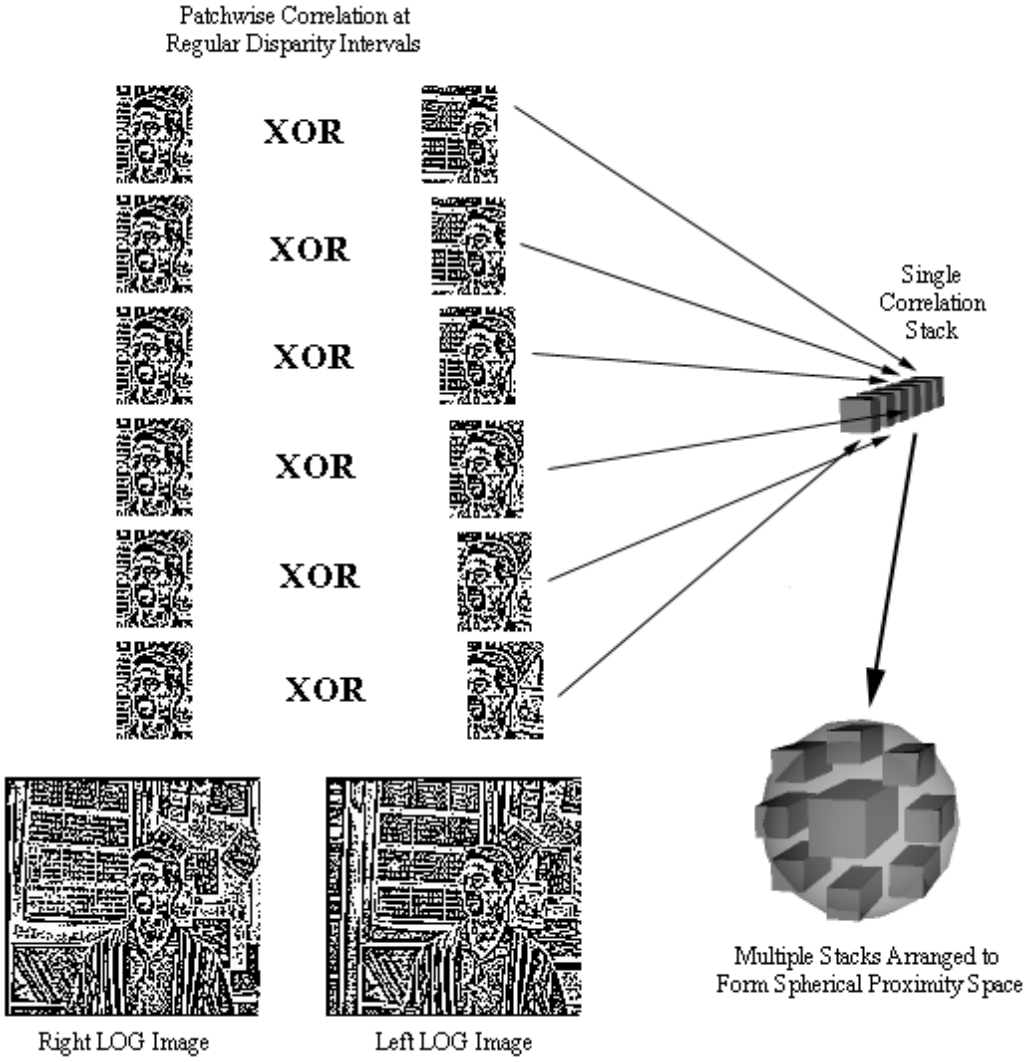

Fig. 3. Composition of a 3D proximity space from a series of 2D correlation measurements. Patches from a pair of right and left sign of LOG images (bottom left) are correlated at regular disparity intervals (upper left); the resulting set of correlation values constitute a "correlation stack" (upper right) of 3D measurements. An appropriate arrangement of locally clustered stacks forms a proximity space (lower right). The proximity space can be used as a 3D perceptual agent.

Proximity spaces As a means to focus attention and ensure a high degree of reactivity to the environment we developed a method in which all of the visual processing is confined to a virtual, three-dimensional region of space called the *Proximity Space* (PS). The proximity space is composed of an arrangement of correlation stacks. The stacks are sized and shaped to give the proximity space a desirable 3D form. For the generic tracking and obstacle avoidance behaviors desired in this project, the proximity spaces were typically spherical or oblong (sausage-shaped). This shape was approximated by placing a ring of eight narrow stacks around a single stack of slightly greater length. Within the bounds of the proximity space, each stack continuously provides feedback indicating whether it is occupied by surface material. Surface material within a stack is identified by peaks which meet the strength, distinctiveness, and uniqueness criteria described above. If occupied, a stack provides further information describing its degree of occupancy; the disparity of its surface material; and possibly a 2D motion vector (via patchwise motion correlation). Confining our measurements to limited volumes of space ensures that they remain focused and limited in number.

Another important aspect of the PS method is that it enforces continuity for attentive control as successive image pairs are processed. The flow (movement) of visual texture and its distribution within the space is used by active sets of behaviors which competitively influence the motion of the space through time. In dynamic terms, the PS acts as an inertial mass and the behaviors as forces acting to accelerate that mass. The utility of this characteristic will become apparent in the following sections.

2.2 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 [6], 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 and texture-misses 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. We have developed a set of gaze control behaviors:

- **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.
- **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.
- **Migrate**: The migration behavior influences the proximity space to traverse a surface in a fixed direction until it reaches the surface boundary where it eventually runs out of material and "stalls."
- **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 cause 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. The active set of behaviors determines the overall behavior of the proximity space (or proximity spaces).

2.3 Eye/head coordination

The previous subsection discussed a method for moving the proximity space electronically within the field-of-view. An important point, which remains to be addressed, is how to move the head in pan, tilt and verge to keep the agent within the center of the field-of-view of both cameras. In effect, as the proximity space moves to track the agent, the fixation point of the cameras is moved to follow the proximity space. Specifically, the pan-tilt-verge control reference is determined by the relative position of the centroid of the proximity space with respect to the fixation point. This is analogous to eye-head coordination in animals in that the electronics provide for rapid but small scale adjustments similar to the eyes, and the mechanics provide for slower but larger scale adjustments similar to the way an animal's head follows its eyes. This control scheme produces a smooth and robust flow of attention.

2.4 Implementation of our active vision system

We have implemented our proximity space method on several different types of hardware, including Pentium MMX, C80 processors and the Teleos Prism-3 vision system. All work described in this chapter was done using the Pentium MMX system, except where noted. All of the coding was done in Microsoft Visual C++ Version 5.0. Two M-Vision 1000 video digitizers from MuTech were

used to capture simultaneous stereo images being produced by two Pulnix 9701 digital CCD cameras. The two cameras were mounted on a Directed Perception pan/tilt head.

2.5 Applications of our active vision system

Proximity spaces are simply a mechanism to measure certain attributes of the image. They can be used for a wide variety of purposes. We have implemented three different applications using proximity spaces: tracking, obstacle avoidance and gesture recognition.

Tracking Tracking a moving agent is the simplest application of proximity spaces, but it is a very important one. Tracking requires acquiring the agent, tracking it and then re-acquiring it when it is inevitably lost. We describe each of these in turn.

Acquiring the agent Before an object can be tracked it must be acquired. To do this, the robot needs to search a given volume of space in an efficient manner. This is achieved by mechanically moving the fixation point of the stereo camera pair through large sweeping trajectories while quickly moving the proximity space in search of substantial surface material (i.e., texture-hits in at least 50% of the measurement cells) within the field of view. Once registered, the system quickly establishes fixation on the surface and starts to track it. A key point of this method is that the system does not require a model of an object in order to acquire it, rather its attention is attracted to the first object that its gaze comes across. The object also need not be moving to be acquired; acquisition relies purely on the existence of surface texture.

Tracking the agent After acquisition has occurred, tracking the agent is done using the **Follow** and the **Cling** behaviors. These two behaviors are run concurrently and their resultant vectors are added to determine the next position of the proximity space. An important point to note about this scheme for tracking is that it does not require that the body be moving in order to track it. This is because the motion disparity behavior is only one of several that combine to maintain focus of attention on the agent. As long as the agent remains distinct (depth-wise) from its surroundings, the robot will track it whether the agent is walking briskly, sitting down, standing back up, even leaping around. The system is, however, brittle to full occlusions because they tend to "pinch" the proximity space between depth discontinuities until it finally loses track of the body altogether. In that case, re-acquisition needs to take place.

Re-acquiring the agent The system described above, while robust, does periodically lose track of the agent. This condition is detected by monitoring the ratio of texture-hits to texture-misses within the proximity space. When this value falls below a certain threshold (about 30%) it indicates that the system has probably

lost track of the surface(s) it was tracking. If the system does lose track, it can re-acquire in a manner identical to initial acquisition except that it may use the additional information of the body's last known velocity to bias the search.

Obstacle avoidance Obstacle avoidance requires a behavior which is opposite of **cling**; we call this behavior **avoid** and its goal is to generate a vector away from the net location of texture. As long as the volume within the proximity space is not intersecting with surface texture (an object's surface) this behavior will not produce an effect. If an object does intersect with the proximity space, **avoid** will act to alleviate the situation by influencing the proximity space to move away from the object until it is again empty.

The fundamental idea then is to create a proximity space that is approximately as large as the robot (in all three dimensions) and place it a set distance in front of the robot. As the mobile platform moves it "pushes" this proximity space along in front of it. As it is being pushed the proximity space will **avoid** objects and try to fit into gaps between objects using **free path search** (see Section 2.2). As the proximity space does this it will guide the robot safely through obstacles. In essence, what is happening is that we are "projecting" a virtual robot represented by the proximity space, in front of the robot and having it find a free path before the robot gets there.

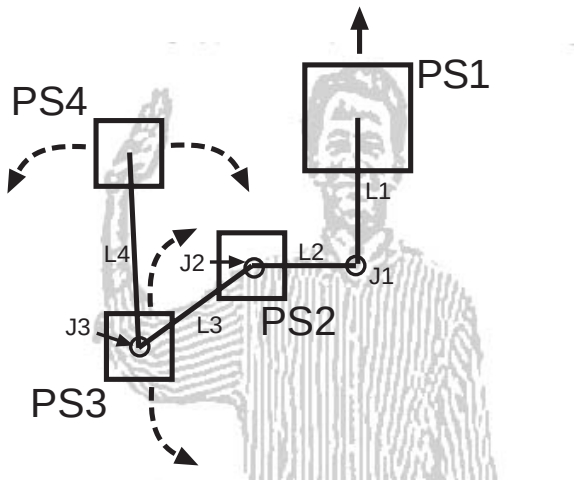

Fig. 4. Coarse 3-D model of a human used for gesture recognition. Four proximity spaces were used in this application.

Gesture recognition Gesture tracking is a natural extension of the pursuit work described in the previous section. However, instead of a single proximity space, several proximity spaces are active concurrently. They attach themselves to various body parts in the image of the gesturing person. Each of these proximity spaces has its own set of tracking behaviors independently controlling its location in space. However, these behaviors are constrained by a coarse three-dimensional, kinematic model of a human that limits their range of motion. With perfect tracking there would be no need for a model as the proximity spaces would track the body parts in an unconstrained manner. However, real-world noise may sometimes cause the proximity space to wander off of their body parts and begin tracking something in the background or another part on the body. While the behaviors acting on the proximity spaces continue to generate motion vectors independent of the model, the final movement of the proximity spaces is overridden by the model if the generated vectors are not consistent with the model. The model is shown in Figure 4. This system, implemented on the Teleos PRISM 3 hardware on a mobile robot, could recognize six gestures in real time. The gesture recognition system is described in details in [17].

3 Integrating Perception and an Intelligent Control Architecture

The perception system that we outlined in the previous section, while powerful, is limited to operating on local sensory information. Proximity spaces are purely reactive "agents" that don't have knowledge about the current task or their relationship to other proximity spaces. Thus, they have no way of knowing if their behaviors are "good" in the global sense, that is, helping the robot in performing its task. In order to take full advantage of our perceptual system, it needs to be placed under the control of an intelligent architecture (described in the next section) that uses proximity spaces, in conjunction with other perceptual processes, to achieve globally desirable tasks.

3.1 Overview of existing 3T architecture

Metrica Incorporated has, over the last several years, developed an autonomous robot control architecture that separates the general robot intelligence problem into three interacting layers or tiers (and is thus known as 3T, see Figure 5):

- A set of robot specific situated skills that represent the architecture's connection with the world. The term situated skills 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

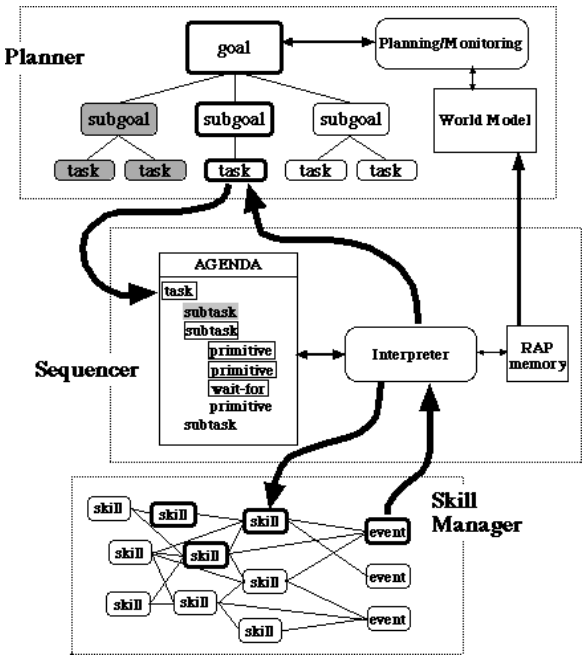

Fig. 5. The 3T intelligent reactive control architecture.

reactive skills for door tracking, local navigation, grasping, and pulling. We are using the Reactive Action Packages (RAPs) system [10] for this portion of the architecture.

– A deliberative planning capability which reasons in depth about goals, resources and timing constraints. The planning tier was not used in the work described in this chapter.

The architecture works as follows. The deliberative tier 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 tier. The RAP interpreter (sequencing tier) decomposes the selected RAP into other RAPs until primitive RAPs are reached, which activate a specific set of skills in the reactive tier. Also activated are a set of event monitors which notifies the sequencing tier 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 tier will terminate or replace the actions when the monitoring events are triggered or when a timeout occurs.

3.2 Integrating active perception and 3T

Integrating perception into this architecture raises a number of interesting research issues, including:

- How can knowledge be moved between the continuous realm to the discrete (or symbolic) realm during visual processing? What advantages does this abstraction give an architecture?
- How can scarce perceptual resources (processing power, head movements, etc.) be allocated to maximum benefit during a task?
- What memory structures are useful for perceptual information? How can these memory structures be kept up-to-date?

Our approach to these issues is to augment the existing skill and sequencing tiers of our architecture with a perceptual memory. This memory is designed to interact directly with high-bandwidth perception and to allow the robot's skills to access perceptual data even if the object is no longer being perceived.

Perceptual memory in the skill manager 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 that operate at other levels of autonomous agent architectures [8, 9]. It is composed of small, task dependent structures called "markers" [1, 5, 23]. The key element 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.

Components of markers Markers consist of three components, called 'what', 'where' and 'identify'. A marker's 'what' component is a task dependent identifier, which one or more behaviors use to determine if they are interested in this marker. 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 (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 (i.e. track). 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.

Integrating perception and RAPs The RAP system is responsible for coordinating sets of skills to achieve tasks. For example, the RAP for tracking an agent would have to enable the visual tracking skill, the robot motion skill and any sonar-based obstacle avoidance skills. It would then monitor this set of skills to verify that the task was being performed. If the task is failing (e.g., the target is lost) then the RAP system can enable a different set of skills, perhaps those associated with searching for an agent. The RAP system is also responsible for creating and deleting markers at the skill level. As such, the RAP system will need to create a deictic memory of what each marker represents in the current task context. While a marker at the skill level will have an 'identity,' this will be visual data that can be used to recognize the marker. The RAP system will need to abstract this information to create a symbolic representation, for example, marker 32 might be "the person that I am tracking." This is one level of abstraction above the proximity space level, as proximity spaces (and associated markers) only know that they are tracking something, not that it is a person or what the reason for tracking it is. This information is contained at the RAPs level. This abstraction can be taken up one more level and the planner could

have information about who that person is, their usual schedule, route, etc., which could be used by the RAP system to perform its tasks better. The RAP system will also be used to allocate scarce perceptual resources and to constrain the perceptual system using *a priori* information.

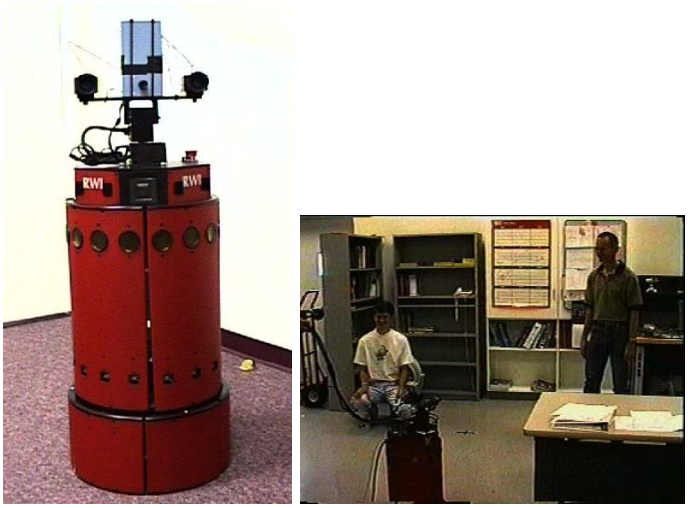

Fig. 6. Left: Our mobile robot with a stereo pair of camera and a single color camera. Right: The robot attending to two different people.

4 An Integrated Task

We designed a scenario in which our stereo system could be integrated with our architecture. The scenario was to attend to multiple people in an unstructured indoor environment. The robot must locate and detect the presence of the 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 of the humans that the robot is to attend. The robot must monitor and track the position of each human it is assisting. The robot will attend to each person in turn, staying a small fixed distance from them. When the robot decides to move on to a different human, it uses its stored knowledge of last known position in an attempt to locate that individual. Once the human has been located, the robot can attend to the new target.

The Scenario was implemented on the Pentium MMX system mounted on a Real World Interface B-14 robot (see Figure 6(left)). The cameras were mounted on the robot, but tethered to the off-board Pentium for image processing. The

B-14 robot has an internal dual processor Pentium computer running Linux. This computer was used to control the robot motion as well as to control the pan-tilt head.

Figure 6 (right) shows the robot (the short, dark column in the foreground) with two humans. The robot is attending to the standing person. For this scenario, we added a cheap color camera to the stereo pair to provide additional perceptual information and to examine how multiple sensors could be integrated using perceptual memory.

4.1 Scenario implementation

In order to perform the integration scenario, we implemented a simple version of a perceptual memory (see Section 3.2) in the skill tier of our architecture. This allows the robot's skill tier to interact with both a proximity space and a color vision system. A 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 to a fixed distance from 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 object (human) associated with the marker. This means it will direct its cameras to the last known location of the human and attempt to reacquire him or her (and move toward him or her).

Each marker in perceptual memory has an associated proximity space which it uses to track its associated object. Maintenance of a marker's 'where' component is as follows. First its 'where' coordinates are 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 of how the association between the two is first made. There are several steps in the process beginning 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 analyzing 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 estimated position 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 can not. 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 vergeable 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 which 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 along the vector, 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.

4.2 Benefits

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 its

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 which 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.

5 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. The perceptual memory provides just the information which is needed for the skills to accomplish this task (and many others we believe).

References

1. Philip E. Agre and David Chapman. Pengi – an implementation of a theory of activity. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1987.
2. Dana H. Ballard. Animate vision. *Artificial Intelligence*, 49(1), 1991.
3. R. Peter Bonasso. Using parallel program specifications for reactive control of underwater vehicles. *Journal of Applied Intelligence*, 2(2), 1992.
4. R. Peter Bonasso, R. J. Firby, E. Gat, David Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.
5. F.Z. Brill, G.S. Wasson, G.J. Ferrer, and W.M. Martin. The effective field of view paradigm: Adding representation to a reactive system. *Engineering Applications of Artificial Intelligence*, January 1998.
6. Rodney A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1), 1986.
7. David J. Coombs and C. M. Brown. Cooperative gaze holding in binocular vision. In *Proceedings of the Fifth IEEE International Symposium on Intelligent Control*, 1991.

8. Chris Elsaesser and Richard MacMillan. Representation and algorithms for multiagent adversarial planning. Technical Report MTR-91W000207, The MITRE Corporation, 1991.
9. R. James Firby. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1987.
10. R. James Firby. *Adaptive Execution in Complex Dynamic Worlds.* PhD thesis, Yale University, 1989.
11. R. James Firby, Roger E. Kahn, Peter N. Prokopowicz, and Michael J. Swain. An architecture for vision and action. In *International Joint Conference on Artificial Intelligence (to appear)*, Montreal, Canada, August 1995. IJCAI.
12. Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1992.
13. Ian Horswill. Find this title. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.
14. Eric Huber. Object tracking with stereo vision. In *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space (CIRFFSS '94)*, 1994.
15. Eric Huber and David Kortenkamp. Using stereo vision to pursue moving agents with a mobile robot. In *1995 IEEE International Conference on Robotics and Automation*, 1995.
16. Leslie Pack Kaelbling. Goals as parallel program specifications. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1988.
17. David Kortenkamp, Eric Huber, and R. Peter Bonasso. Recognizing and interpreting gestures on a mobile robot. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1996.
18. E. Kroktkov. *Active Computer Vision by Cooperative Focus and Stereo.* Springer-Verlag, Berlin, 1989.
19. David Marr and T. Poggio. A computational theory of human stereo vision. In *Proceedings of the Royal Society of London*, 1979.
20. H.K. Nishihara. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5), 1984.
21. Stan J. Rosenschein and Leslie Pack Kaelbling. The synthesis of digital machines with provable epistemic properties. In *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–98, Monterey, CA, 1988.
22. Marcel Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 1987.
23. Glenn Wasson, Gabe Ferrer, and W. N. Martin. Systems for perception, action and effective representation. In *FLAIRS-97*, 1997.

This article was processed using the LaTeX macro package with LLNCS style