A Behavior-based Approach to Active Stereo Vision for Mobile Robots

Eric Huber and David Kortenkamp
Metrica Inc. Robotics and Automation Group
NASA Johnson Space Center — ER4
Houston, TX 77058
(281) 483-2740
Fax: (281) 483-3204
{huber,korten}@mickey.jsc.nasa.gov

May 14, 1997

Abstract

This paper describes an active stereo vision system that uses a behavior-based approach to control gaze. The system concentrates its computational resources on cubic volumes of space called proximity spaces. Each proximity space is controlled by a set of behaviors that combine to determine its location in the next processing frame. Multiple proximity spaces can be chained together using kinematic models to perform complex visual tasks. The vision system is mounted on a mobile robot and integrated into an intelligent control architecture. The intelligent control architecture serves to interpret the vision system output in task context and produce appropriate robot motion. The resulting robot system can perform tasks such as pursuing other agents, recognizing natural gestures and avoiding obstacles.

Keywords: active vision, stereo vision, mobile robots, intelligent control

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 paper uses a stereo-based active vision system to accomplish several different tasks, including pursuing other agents, obstacle avoidance, and human gesture recognition. 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 Related work

The active vision paradigm was espoused by Ballard [1] 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. Results from computation in one frame are carried over to the next frame of the image. A key concern in active vision research is *gaze control* [6]. Specifically, the vision system must decide where within the scene to focus its limited computational resources while controlling the motion of the cameras in order to appropriately change the visual scene available in successive frames.

Early active vision systems [14] were promising, but typically too brittle and slow for practical application. In 1992 Keith Nishihara developed the PRISM-3 high speed stereo vision system. The PRISM-3 is a third generation embodiment of Nishihara's Laplacian of Gaussian sign correlation theory; an extension of Marr and Poggio's classical zero-crossing theory. This machine contains several hardware accelerators, provided the means for robust, real-time, real-world, stereo vision for the first time. A few real-time temporal correlation-based vision machines have been built [8, 19], but they lack measurement in the third dimension which is not only necessary to locate objects in 3-D but also crucial for figure ground discrimination and therefore robust tracking.

Vision-based obstacle avoidance techniques have also received quite a bit of attention. This includes classic stereo vision algorithms such as those by Matthies [16] and less computationally intensive techniques such as those by Horswill [11]. However, the former is hampered by requirements for large computational resources and the latter is effective only in very constrained environments. Active vision techniques for obstacle avoidance have not been widely researched.

1.2 Overview

The next section of this paper introduces our active stereo vision system. The basic components of the system include dedicated hardware, a region of attention and control mechanisms for moving the region of attention. The Section 3 of the paper describes our mobile robot applications. First, we introduce our intelligent control architecture and show how we use it to integrate our vision system with robot motion. Then we describe in detail each application: pursuit of moving agents, human gesture recognition and obstacle avoidance, and give results of each. Finally, in Section 4 we discuss our future work and conclusions.

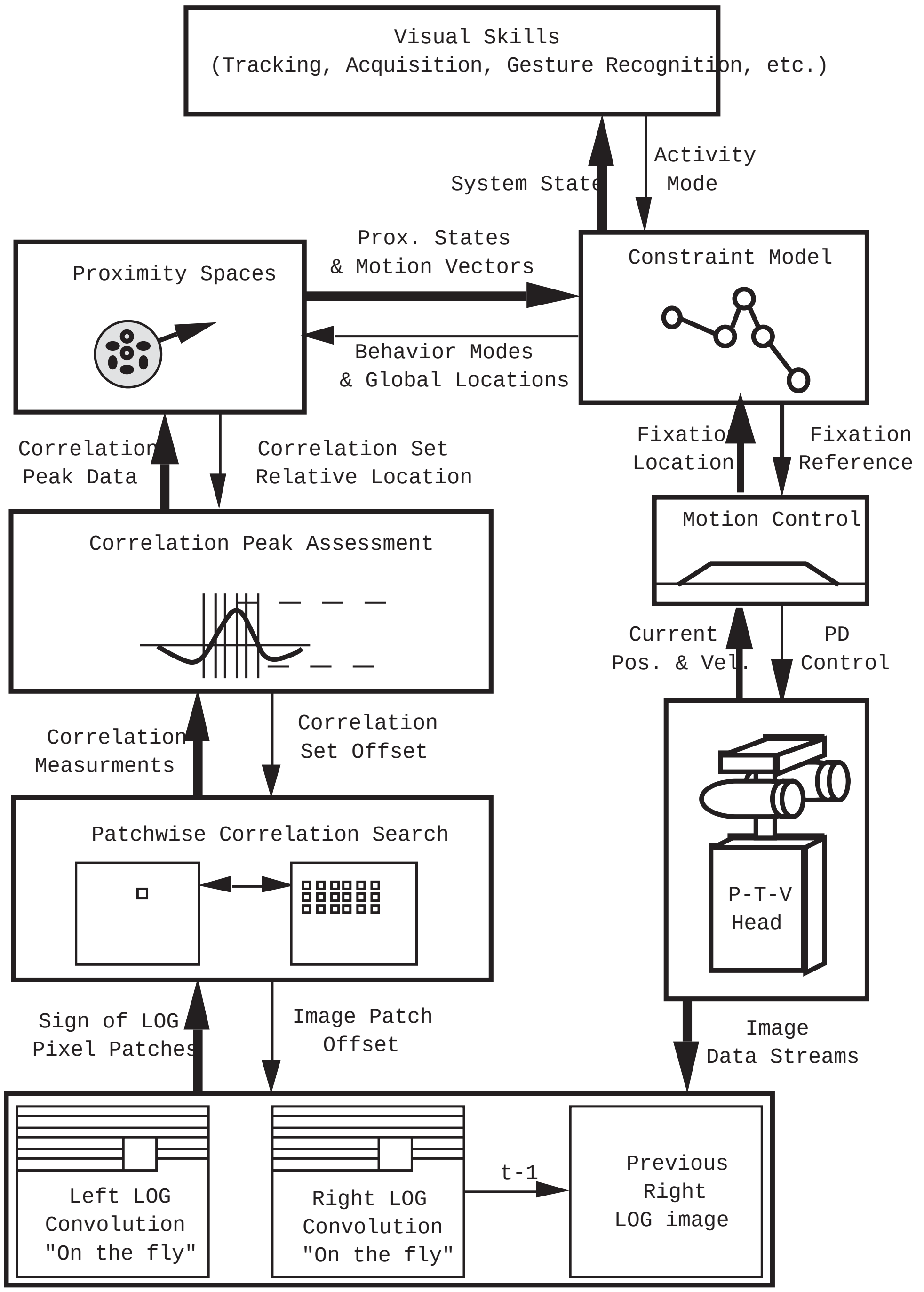
Visual Skills
(Tracking, Acquisition, Gesture Recognition, etc.)
System State
Activity Mode
Proximity Spaces
Prox. States & Motion Vectors
Behavior Modes & Global Locations
Constraint Model
Correlation Peak Data
Correlation Set Relative Location
Fixation Location
Fixation Reference
Correlation Peak Assessment
Motion Control
Current Pos. & Vel.
PD Control
Correlation Measurments
Correlation Set Offset
Patchwise Correlation Search
P-T-V Head
Sign of LOG Pixel Patches
Image Patch Offset
Image Data Streams
Left LOG Convolution "On the fly"
Right LOG Convolution "On the fly"
t-1
Previous Right LOG image

Figure 1: The software modules comprising our visions system. Thick arrows represent data flow, thin arrows represent command flow.

2 An Active Stereo Vision System

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 [1, 6]. In particular,

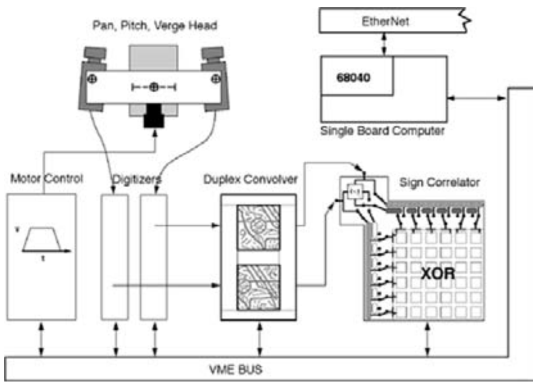

Figure 2: A schematic of our hardware system.

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 Guassian 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 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 proximity space location with respect to the task being performed. Each of these modules will now be discussed in more detail.

2.1 Vision hardware

The hardware set-up we use in our work consists of two B&W CCD cameras mounted on a pan-tilt-verge head, which itself is mounted on a mobile robot. The two cameras are connected to a pair of Datacube framegrabbers that are contained in a VME card cage. Also in the card cage are special purpose boards that comprise the PRISM-3 stereo vision system, developed by Keith Nishihara [17]. Finally, the card cage contains a 68040 board running OS9 to perform general purpose computations. Figure 2 shows the hardware layout of our system.

2.2 Convolution and correlation

The PRISM-3-based stereo vision system is the embodiment of Keith Nishihara's sign correlation theory [17]. This theory is an extension of work by Marr and Poggio [15] that resulted in a stereo matching algorithm consistent with psychophysical data. The PRISM system employs dedicated hardware including a Laplacian-of-Gaussian convolver and a parallel sign correlator to compute spatial and/or temporal disparities between correlation patches in each image. Stereo measurements are made by correlating sets of patches along the epipolar lines of left and right image pairs. Motion measurements are performed (in effect) by correlating a reference patch from a previous image frame with a tessellation of patches in the current frame. Programmable gate array technology makes it possible to perform a set of correlations with 32 by 32 pixel windows at 36 different disparities in 100 microseconds. Other stereo vision systems (e.g., [12]) have also benefited from hardware accelerated convolution and correlation.

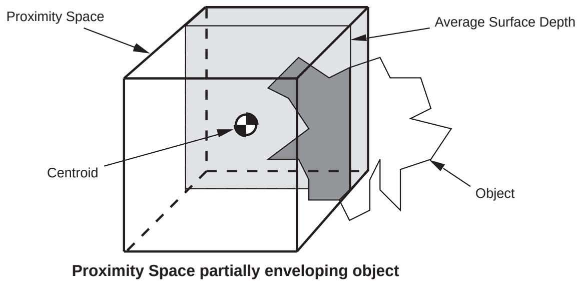

Figure 3: A proximity space, represented here as a cube although its shape can change from task to task.

2.3 The proximity space

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 limited three-dimensional region of space called the *proximity space*. For maximum efficiency this method capitalizes on measurements within the horoptor that can be made with the least expense (time) and minimizes the number of measurements necessary per frame.

The horoptor is a region of near zero disparity in the stereo field of view which lends itself well to accurate, high speed correlation [18]. This region is roughly centered about the point of intersection (*fixation point*) of the optical axes of the right and left cameras. For gaze control, the horopter is of particular significance since the surface texture of objects occupying this region is readily detected (correlated).

Our method employs a volume of space, typically located in and about the horoptor, which is the proximity space (see Figure 3). Within the bounds of this space an array of stereo and motion measurements are made in order to determine which regions of the space (measurement cells) are occupied by surface material, and what the spatial and temporal disparities are within those regions. We will refer to a positive identification of surface material within a measurement cell as a "texture-hit" and a negative as a "texture-miss". Surface material is identified by detecting visual texture, i.e., variations in light intensity across a region of the image. Confining our measurements in this manner ensures that they remain focused and limited in number. As a result, we can perform all of our computation within a 33ms (30Hz frame rate) cycle.

2.4 Controlling proximity spaces

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 [5], 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.

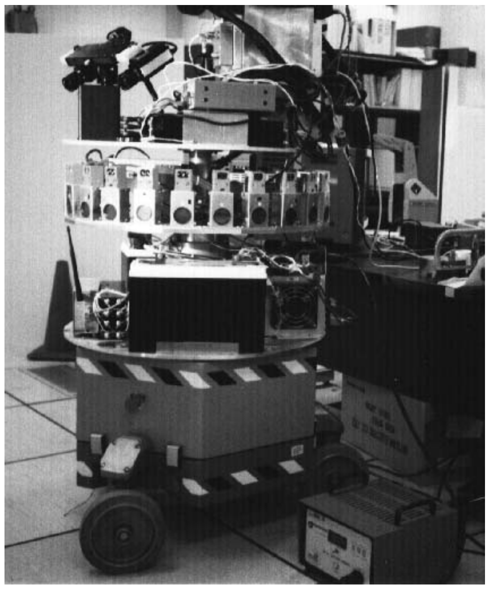

Figure 4: The stereo vision system is mounted on top of our robot; the sonar sensors are just below it.

- **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.
- **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 the degree of occupancy, also changing the density of measurements within the space. The net effect of this behavior is that it tends to increase the search space if an object is "getting away" or even momentarily lost.
- **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 reacquire the object if tracking fails.
- **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.
- **Free path search**: This behavior is the opposite of object search in tracking. It searches for the nearest gap (empty volume) to its current location.

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). In some cases, we have an *a priori* model of how we expect the object to move; in these cases a constraint model can be used to limit proximity space motion. This will be described more completely in Section 3.2

3 Mobile Robot Applications

Our mobile platform is a Cybermotion K2A base with a ring of 24 sonar sensors (see Figure 4). All processing done by the stereo system is performed on-board the robot. The integration of the stereo vision system and our mobile platform took place within the context of an intelligent architecture we are developing for control of robots [2]. The architecture is composed of three components: a set of reactive skills, a conditional sequencer and a planner. The skills are C routines that achieve or maintain some state in the world. We have a core set of skills for controlling the mobile platform, including turning and moving. We have another set of skills that performs obstacle avoidance using the Vector Field Histogramming (VFH) technique [4]. The conditional sequencer is the Reactive Action Packages (RAP) system [9]. It activates and deactivates sets of skill depending on the task and the state of the world and robot. The planner was not used in the system described in this paper.

Our architecture allows us to easily create sophisticated mobile robot applications from a core set of motion control and vision skills. On-board the robot are the following motion control skills:

- MAKE-MAP: This skill takes sonar information and generates as output a histogram map of obstacle positions using an algorithm described in [3].
- FIND-FREE-DIRECTION: This skill takes the histogram map and the desired goal and finds a free direction of travel using the VFH obstacle avoidance algorithm [4].
- DRIVE-AND-STEER: This skill takes the free direction and the desired velocity of the robot and generates drive and steer velocities based on obstacle density and the acceleration limits of the robot.
- TURN-ROBOT: This skill takes a goal angle (either relative or absolute) and turns the robot to that angle.

By enabling this set of skills with appropriate parameters, the RAP system can make the robot move about while also avoiding obstacles. In the next several sections we begin adding visual skills to the robot. These visual skills, when enabled by the RAP system, provide input to the core motion control skills to drive the robot under visual control.

3.1 Pursuit

The first application of our vision system to mobile robots involved having our robot pursue a moving agent (usually a person, but sometimes another robot). We use the basic set of proximity space behaviors (described in Section 2.4) to implement visual skills such as acquiring the agent to be tracked and tracking the agent. The results of the visual skills are passed to the motion control skills to perform the Pursuit task.

3.1.1 Acquiring the agent

Before an object can be tracked it must be acquired. To do this, the vision system needs to search a given volume of space in an efficient manner. This is achieved using the **search** behavior to move 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 25% 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.

3.1.2 Tracking the agent

Tracking of a moving agent involves combining several simple behaviors to move the proximity space to follow the agent. Our initial attempt at tracking used two behaviors called **follow** and **pull**. The first measures motion disparity and produces a 2D-vector that attempts to move the proximity space in the direction of motion. The second behavior measures stereo disparity (depth) and produces a 1D-vector that attempts to move the proximity space closer to or further from the stereo head. However, we found that using only these two behaviors, the system would track the agent for a few minutes, but accumulated correlation errors eventually caused the proximity space to fall off of the agent. Also, rotations of the agent quickly cause the system to lose track of the agent because the rotation causes the motion disparity vector to point off the rotating agent.

To compensate for these deficiencies, we added a new behavior (the **cling** behavior) that produces a 2D-vector pointing from the centroid of the proximity space to the centroid of the texture-hits within that space. If the proximity space approaches the occluding contour (boundary) of an object, the texture attractor behavior will tend to direct it away from the nearby depth discontinuity and back onto the body of the agent. This behavior will sometimes be in direct conflict with the influences of the motion disparity behavior. When the cling behavior is well positioned on the body of the agent (i.e., it detects an even distribution of texture-hits), it produces a vector of low magnitude. An interesting side effect of the cling behavior is that it also tends to direct the proximity space away from foreign bodies of unlike disparity. Thus, partially occluding (intervening) object boundaries have the same effect as occluding contours on the tracked body itself. For example, as the agent moves behind a desk or chair the texture attractor behavior causes the proximity space to be repulsed by the intervening obstacles and towards the head and chest (when tracking people).

An additional problem when tracking terrestrial agents is that, at times, accumulated correlation errors cause the proximity space to slowly migrate all the way down the body and "slip" onto the floor. This is due to the lack of depth discontinuity at the point of contact between the agent and the floor. To remedy this, we added a simple behavior **migrate** that produces a 1D-vector that biases the proximity space upward. The result is that the proximity space tends to settle on the head or upper torso when tracking a human. An added benefit of this is that the human face provides a high degree of unique texture and thus lends itself well to Nishihara's algorithm [17]. Finally, this behavior keeps measurements away from the less stable areas of the body such as the arms and legs.

The nature of binocular geometry imposes more inherent difficulties to the tracking task; varying distances to the moving agent cause changes in several key parameters such as image and disparity scale. To compensate for these scale variations we developed a proximity-space-sizing behavior (**resize**). Using area moment computations on texture-hits and texture-misses, this behavior acts to resize the proximity space until it is properly scaled to the agent's image. This behavior is different from the others in that it influences the scale of the proximity space and not its position. This method of auto-normalization also allows us to track objects of significantly disparate scales such as softballs and humans, without having to adjust any "magic" numbers.

Each of the behaviors described above runs 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.

3.1.3 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 (in pixel disparity units) 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, running on our real-time hardware, produces a smooth and robust flow of attention.

3.1.4 Reacquiring 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 10%) it indicates that the system has probably lost track of the surface(s) it was tracking. If the system does lose track, it can reacquire 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.

3.1.5 Integration with motion control

To integrate the above visual abilities into our intelligent control architecture, we created several visual skills that perform distinct actions. Essentially, these visual skills provide an abstracted way to command the vision system. The set of skills (both visual and robot control) resident on the robot for the Pursuit task are:

- TRACK-AGENT: This skill communicates with the vision system as it tracks the agent. It outputs the (x,y,z) position of the object relative to the robot as well as the status of the tracking process (i.e., tracking or lost track).
- ACQUIRE-AGENT: This skill causes the vision system to begin a systematic search for texture. Inputs to this skill give a starting direction and distance for the search.
- TRACK-GOAL-GEN: This skill takes input from the track-agent skill and generates goal positions and velocities for the robot based on the location, distance and speed of the agent being pursued. This is passed to the FIND-FREE-DIRECTION skill.

In order to execute the pursuing task, we wrote several RAPs that activate the appropriate skills depending upon the situation. First, the ACQUIRE-AGENT skill is activated and told to search a particular volume of space for an agent (the search process is described in Section 3.1.1). When the ACQUIRE-AGENT skill acquires the agent, the TRACK-AGENT skill is activated and begins reporting the agent's (x, y, z) position with respect to the robot. At the same time, the ROBOT, MAP-MAKING, FIND-FREE-DIRECTION, and DRIVE-AND-STEER skills are activated (these motion control skills are described in Section 3). The TRACK-GOAL-GEN skill takes the position of the agent relative to the robot and converts it to world coordinates, making it a goal for the robot to attain. The TRACK-GOAL-GEN skill also calculates the distance and speed of the object and chooses a desired speed for the robot. This desired speed may be zero if the robot is close enough to the agent or even negative if the agent is too close (i.e., the robot will be instructed to back up). The FIND-FREE-DIRECTION skill takes the goal and speed and, using the sonar map of obstacles, determines a free direction of travel. The DRIVE-AND-STEER skill then takes the free direction and desired speed and computes the robot's drive and steer velocities. This cycle continues, with the TRACK-AGENT skill producing a new agent location four times a second for the robot (while we could produce new locations at frame rate, the physical robot cannot react that quickly, the cameras do track at frame rate). If the TRACK-AGENT skill loses the agent the TRACK-GOAL-GEN skill sets the robot speed to zero and the ACQUIRE-AGENT skill is activated to being searching for the agent.

3.1.6 Results

We tested our system by having our mobile robot pursue us around our laboratory. Our lab is approximately 20 meters square with uniform fluorescent lighting and normal office furniture. Once the robot begins pursuing an agent, it continues until it loses the agent permanently. Even if the agent stops moving, the robot maintains a fixed distance from the agent (two meters in our case) and waits for the agent to start moving again. If the vision system loses the agent, the robot stops and the vision system attempts to reacquire the agent. Often it finds the agent before the robot even comes to a complete stop and pursuit continues, almost uninterrupted. The cases where the robot loses the object entirely mostly fall into the following three categories: 1) The object becomes fully occluded; 2) The agent (or the ego-motion of the mobile base as it turns to avoid an obstacle) exceeds the tracking speed of the stereo head (about 60 degrees per second); and 3) The object moves too close (within one meter) or too far away (further than 10 meters) from the camera. In these cases, the robot has to start its initial search again. The agent can aid this search by moving in front of the robot.

This system was tested over a period of a month using a variety of people as agents and several times using another robot as an agent. The maximum speed at which the robot could travel was set at 0.4 meters/second. We had the robot pursue people for up to fifteen minutes, with the limiting factor usually being boredom on the part of the person being tracked. The person being pursued had to be careful not to become fully occluded, not to move laterally too quickly and not to get too far away from the robot, especially when the robot slowly maneuvered through obstacles that didn't slow a person down.

3.2 Gesture recognition

The second application of our vision system to mobile robots was to perform human gesture recognition. Gesture recognition is especially valuable in mobile robot applications for several reasons. First, it provides a redundant form of communication between the user and the robot. For example, the user may say "Halt" at the same time that they are giving a halting gesture. The robot need only recognize one of the two commands, which is crucial in situations where speech may be garbled or drowned out (e.g., in space, underwater, on the battlefield). Second, gestures are an easy way to give geometrical information to the robot. Rather than give coordinates to where the robot should move, the user can simply point to a spot on the floor. Or, rather than try to describe which of many objects the user wants the robot to grasp, they can simply point. Finally, gestures allow for more effective communication when combined with speech recognition by grounding words such as "there" and "it" with recognizable objects.

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.

3.2.1 Chaining multiple proximity spaces using a human model

We have chosen a model that resembles the human skeleton, with similar limitations on joint motion. The model, shown in Figure 5, consists of a head connected to two shoulder joints. There are four proximity spaces that are tracking various parts of the body. The first proximity space (PS1) is tracking the head. Its behaviors allow it to move freely, but are also biased to migrate it upward. The second proximity space (PS2) tracks the shoulder. PS1 and PS2 are connected by two links, L1 and L2. These links are stiff springs

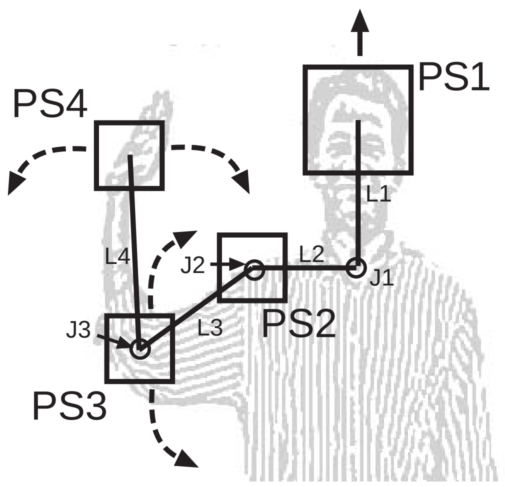

Figure 5: Coarse 3-D model of a human used for gesture recognition.

that allow very little movement along their axes. The lengths of L1 and L2 are set at the start of tracking and do not change during tracking. We have a procedure for automatically determining the lengths of the links from the height of the person being tracked, which is described later in the paper. The connection between L1 and L2 (J1) acts as a ball joint. The motion of L2 relative to L1 (and thus, PS2 relative to PS1) is constrained by this joint to be 0 deg in the up-down direction (i.e., shrugging of the shoulders will not affect the location of the proximity space). The joint J1 also constrains the movements of L2 relative to L1 by ± 38 deg in the in-out direction (i.e., towards and away from the camera). This means that the person must be facing the cameras to within 38 deg for the gesture recognition system to work correctly.

The third proximity space (PS3) is tracking the end of the upper arm (essentially the elbow) and is connected to PS2 by the link L3. Again, L3 is modeled as a very stiff spring with a length fixed at the start of tracking. L3 is connected to L2 with a ball joint (J2). L3 can move relative to L2 up at most 75 deg and down at most 70 deg. It can move into or out-of perpendicular to the camera by at most ± 45 deg. This means that a pointing gesture that is towards or away from the robot by more than 45 deg will not be recognized. Finally, the fourth proximity space (PS4) tracks the end of the lower arm (essentially the hand) and is connected to PS3 with link L4 at joint J3. The range of motion of L4 relative to L3 at J3 is up at most 110 deg, down at most -10 deg and into or out-of perpendicular with the camera by at most ± 45 deg. All of the links in the model are continuously scaled based on the person's distance from the cameras.

One limitation of our gesture recognition system is that it can only track one arm at a time. There are two reasons for this limitation. First, we do not have enough processing power to calculate the locations of seven proximity spaces at frame rate. Second, when both arms are fully extended the hands fall out of the field of view of the cameras. If the person backs up to fit both hands into the field of view of the cameras then the pixel regions of arms are too small for tracking. The arm to be tracked can be specified at the start of the tracking process and can be switched by the user. While Figure 5 only shows the model of the right arm for simplicity, the model for the left arm is simply a mirror image of the right.

3.2.2 Acquisition and reacquistion using the model

Gesture recognition is initiated by giving the system a camera-to-person starting distance and a starting arm. Four proximity spaces are spawned and lay dormant waiting for some texture to which to attach themselves. As a person steps into the camera at approximately the starting distance the head proximity space will attach itself to the person and begin migrating towards the top, stopping when it reaches the boundary of the person. The other three proximity spaces are pulled by their links up along with the head. While the

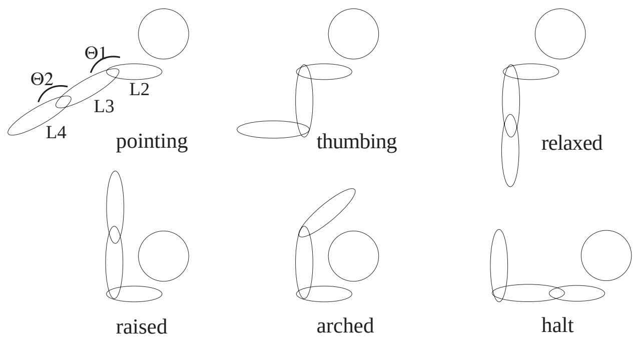

Figure 6: Recognizable gestures.

person's arm is at their side, these proximity spaces are continually sweeping arcs along the dashed arrows shown in Figure 5 looking for texture to which to attach themselves. When the arm is extended the three proximity spaces "lock onto" the arm and begin tracking it. If they lose track (e.g., the arm moves too fast or is occluded) they begin searching again along the dashed arcs shown in Figure 5. If the head proximity space loses track it begins an active search starting at the last known location of the head and spiraling outward. Many times this reacquisition process works so quickly that the user never realizes that tracking was lost.

3.2.3 Defining gestures

The two types of information used in gesture recognition are relative PS location and PS contentment. Aggregate contentment of the system reveals how well correlated the model is with its human host. A moving weighted average of contentment is used to derive a gesture confidence measure. If a specific gesture is identified, and it is supported by a high degree of contentment, then the confidence in that state builds. Confidence assessment serves a number of purposes: First, as a moving average, it filters out momentary (due to system noise) false positive (and negative) gesture matches; Second, as the human attempts to communicate, he may momentarily traverse gesture configurations which were not intended to be interpreted; these also get filtered out. Finally, this form of gesture filtering tends to "debounce" gesture transitions making time domain gesture sequences, such as waving, clearly detectable.

Figure 6 shows the gestures that are currently recognized by the system. These gestures are very easily determined by looking at the relative angles between the links L2, L3 and L4 at the joints J2 and J3 (see Figure 5). Let's call the angle between L2 and L3 (i.e., the upper arm angle) Θ1 and the angle between L3 and L4 (i.e., the lower arm angle) Θ2. 0 deg is pointing in the direction of the previous link. Angle values increase clockwise for the right arm (pictured) or counterclockwise for the left arm. Then, if $\Theta1 < -50$ deg and $\Theta2 < 45$ deg the gesture is *relaxed.* If $\Theta1 < -50$ deg and $\Theta2 > 45$ deg the gesture is *thumbing.* If -50 deg $< \Theta1 < 45$ deg and $\Theta2 < 45$ deg the gesture is *pointing.* If -50 deg $< \Theta1 < 45$ deg and $\Theta2 > 45$ deg the gesture is *halt.* If $\Theta1 > 45$ deg and $\Theta2 < 45$ deg the gesture is *raised.* If $\Theta1 > 45$ deg and $\Theta2 > 45$ deg the gesture is *arched.* Thus, the person is always producing some kind of gesture based on the joint angles.

3.2.4 Integrating with robot motion

Once again, we integrated the gesture recognition ability with robot motion using our intelligent control architecture. We abstracted the visual processing described above into an additional set of visual skills that were added to the visual skills described in Section 3.1.5. These additional visual skills were:

- RECOGNIZE-GESTURE: This skill encapsulates the processes described in Section 3.2.3 and produces one of the five gestures or no gesture as output. It also generates as output the (x,y,z) locations of the four proximity spaces when the gesture was recognized.

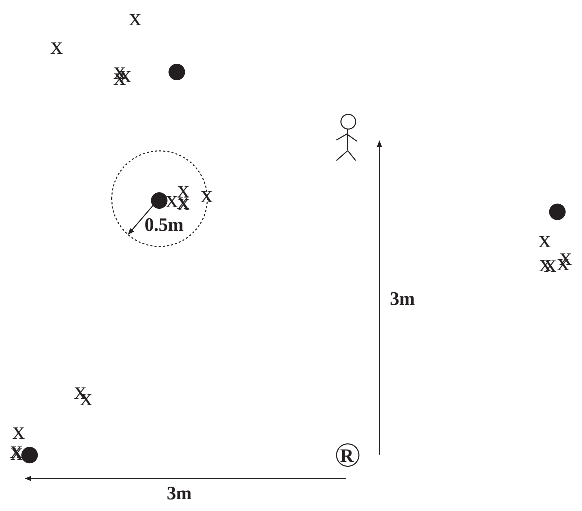

Figure 7: A sample of experimental results. The person is standing directly in front of the robot and pointing at different points on the floor (black circles). The 'X' is the point that the robot calculated as the intersection between the pointing gesture and the floor.

- MOVE-TO-POINT: This skill takes the (x,y,z) location of the centroid of the shoulder proximity space (PS2 in Figure 5) and the hand proximity space (PS4 in Figure 5) and computes a three-dimensional vector. It then determines the intersection point of this vector with the floor. Assuming the vector does intersect with the floor, the skill generates a goal for the robot, which is used by the motion control and sonar-based obstacle avoidance skills to move the robot to that point.
- ACQUIRE-ALONG-VECTOR: This skill takes the (x,y,z) location of the centroid of the shoulder proximity space and the hand proximity and computes a three dimensional vector. The skill then causes the vision system to search along through a tube of space surrounding that vector until a patch of significant texture is encountered. Either the vision system starts tracking that object or, if no object is found within a set distance, it returns to its starting location.
- TRACK-AGENT: This skill was updated from the pursuit application. It uses the head proximity space (PS1) to produce the x,y,z location of the agent with respect to the robot.

The RAP system chooses the set of visual and robot skills that are active depending on the task that is to be accomplished. For example, if the task is to have the human point to a spot on the floor and have the robot move there, then the RAP system would activate the RECOGNIZE-GESTURE skill along with the MOVE-TO-POINT skill. When the pointing gesture is recognized it will pass the location of the shoulder and hand to the MOVE-TO-POINT skill, which will compute a goal location for the robot. Then the RAP system activates the MAKE-MAP, FIND-FREE-DIRECTION, and DRIVE-AND-STEER skills to cause the actual robot motion. The robot will then move to the point, avoiding obstacles along the way.

3.2.5 Results

Experiments show that our system can recognize gestures from distances as close as 1.25 meters from the camera (at which point the person's arm extends out of the field of view of the cameras) to as far as 5 meters away from the robot. The system can track a fully extended arm as it moves at speeds up to approximately 36 deg per second (i.e., a person can move their arm in an arc from straight up to straight down in about five seconds without the system losing track).

We conducted a number of experiments to determine the accuracy of the pointing gesture. The experimental set-up was to have a person point to a marked point on the floor. The vision system would recognize the pointing gesture and the MOVE-TO-POINT skill would determine the intersection of the pointing vector with the floor. We would then compare this point with the actual location of the target. We choose eight different target points on the floor in various directions and at various distances. We pointed five times at each target point. Two different people did the point, both of them familiar with the system. No feedback was given to the user between trials. Figure 7 shows a sample of those points and the system's performance.

For the five points that were between 2.5 and 4.5 meters away from the person, the mean error distance from the target to the vector intersection was 0.41 meters, with a standard deviation of 0.17. As the distance from the person to the target grew the error also grew rapidly, up to a mean error of over 3 meters at 5.5 meters away.

These results need to be taken with a grain of salt. There are several factors that can introduce errors into the system and that cannot be accounted for, including: how accurately a person can actually point at a spot; the initial accuracy of the robot both in position and orientation; and the tilt of the robot due to an uneven floor.

3.2.6 Related work

Gesture recognition has become a very important research area in recent years and there are several mature implementations. The ALIVE system [7] allows people to interact with virtual agents via gestures. The ALIVE system differs from ours in that the cameras are fixed (i.e., not on a mobile platform as ours are) and that it requires a known background. Similar restrictions hold for a system by Gavrila and Davis [10]. The Perseus system [13] uses a variety of techniques (e.g., motion, color, edge detection) to segment a person and their body parts. Based on this segmentation the Perseus system can detect pointing vectors. This system is very similar to ours in that it is mounted on a mobile robot and integrated with robot tasks. The Perseus system differs from ours in that it requires a static background, doesn't detect gestures in three dimensions and relies on off-board computation, which can cause delays in recognition of gestures. Wilson and Bobick [20] use a hidden Markov model to learn repeatable patterns of human gestures. Their system differs from ours in that it requires people to maintain strict constraints on their orientation with respect to the cameras.

3.3 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 2.4). 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. We believe this will be a powerful new technique for robust mobile robot obstacle avoidance. We plan to integrate this vision-based technique with our existing sonar-based obstacle avoidance techniques. The vision system will look ahead of the robot, finding free corridors for robot movement, while the sonar system will handle immediate obstacles around the robot.

3.3.1 Results

We have implemented the obstacle avoidance behaviors while our vision system was on the bench and done some general testing. Other demands on our mobile robot have not allowed us to integrate the vision system and the robot to perform tests of the obstacle avoidance method. We expect to have this done well before the final version of this paper is due and can include results then. If these results are not finished in time for the final paper, we will move obstacle avoidance to the future work section of this paper.

4 Conclusions

Effective vision processing is a requirement before mobile robots can become widely useful in many tasks. In particular, when interacting with humans, the ability to track and to recognize gestures is crucial. In addition, vision-based obstacle avoidance can provide a compliment to sonar and IR-based methods, especially in generating long-range paths. We have described a vision system than can perform real time tracking, gesture recognition and obstacle avoidance. This vision system takes advantage of recent work in active vision and advances in behavior-based control. We then mounted this vision system on a mobile robot and performed a variety of tasks. Many of our results are subjective, i.e., they are based on the perceptions of those who interact with the robot. However, in the course of these interactions (by both the robot developers and non-roboticists), our combination of real time vision and intelligent motion control resulted in a surprisingly animate robot that acted robustly and intelligently.

4.1 Future work

Future work on our system will proceed along two distinct tracks. First, we are developing additional applications for our existing stereo vision system. In particular, we want to extend gesture tracking to allow it to recognize moving gestures (e.g., waving). We also want to extend gesture recognition to allow it to track non-humans, for example, to track a robotic manipulator and report link positions. An extension of this last task would be to watch each link for imminent collision with obstacles. We also want to extend our vision system to allow for sophisticated eye-hand coordination, for example, pursuing and catching a moving object. This will require mounting a manipulator on our mobile platform and we are in the process of doing that.

The second track of future work involves reimplementing our system on different hardware. The VME-based system we currently use is large and power-hungry. We are examining several different technologies (gate arrays, DSP chips, VSLI chip design, etc.) in which we could implement our core visual algorithms. We are also looking at digital cameras as a means of by-passing framegrabbers and their large power requirements. Our goal is a compact, low-power, inexpensive stereo vision system for mobile robot applications.

References

[1] Dana H. Ballard. Animate vision. *Artificial Intelligence*, 49(1), 1991.

[2] 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.

[3] Johann Borenstein and Yoram Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Journal of Robotics and Automation*, 7(4), 1991.

[4] Johann Borenstein and Yoram Koren. The Vector Field Histogram for fast obstacle-avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3), 1991.

[5] Rodney A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1), 1986.

[6] 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.

[7] Trevor J. Darrell, Pattie Maes, B. Blumberg, and Alex Pentland. A novel environment for situated vision and behavior. In *Workshop on Visual Behaviors: Computer Vision and Pattern Recognition*, 1994.

[8] Ernst D.Dickmanns. Expectation-based dynamic scene understanding. In Andrew Blake and Alan Yuille, editors, *Active Vision*. MIT Press, Cambridge, MA, 1992.

[9] R. James Firby. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1987.

[10] Dariu M. Gavrila and Larry Davis. 3-D model-based tracking of human upper body mvement: A multi-view approach. In *IEEE Symposium on Computer Vision*, 1995.

[11] Ian Horswill. Polly: A vision-based artificial agent. In *National Conference On Artificial Intelligence (AAAI-93)*, 1993.

[12] Hirochika Inoue, Tetsuya Tachikawa, and Masayuki Inaba. Robot vision system with a correlation chip for real-time tracking, optical flow and depth map generation. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, 1992.

[13] R. E. Kahn and M. J. Swain. Understanding people pointing: The perseus system. *International Symposium on Computer Vision*, November 1995.

[14] E. Kroktkov. *Active Computer Vision by Cooperative Focus and Stereo*. Springer-Verlag, Berlin, 1989.

[15] David Marr and T. Poggio. A computational theory of human stereo vision. In *Proceedings of the Royal Society of London*, 1979.

[16] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. Mars microrover navigation: Performance evaluation and enhancement. In *Proceedings IEEE/RSJ International Conference on Robots and Systems (IROS)*, 1995.

[17] H.K. Nishihara. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5), 1984.

[18] T. J. Olson. Stereopsis for verging systems. In *Proceedings IEEE Computer Vision and Pattern Recognition Conference*, 1993.

[19] K. Sumi, M. Hashimoto, and H. Okuda. Three-level broad-edge matching-based real-time robot vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1995.

[20] Andrew Wilson and Aaron Bobick. Configuration states for the representation and recognition of gesture. In *International Workshop on Automatic Face and Gesture Recognition*, 1995.