Integrating Active Perception with an Autonomous Robot Architecture

Glenn Wasson
Computer Science Department
University of Virginia
Charlottesville, VA 22903
wasson@virginia.edu

David Kortenkamp and Eric Huber
Metrica, Inc.
Texas Robotics and Automation Center
1012 Hercules
Houston, TX 77058
{korten | huber}@mickey.jsc.nasa.gov

Abstract

Today's robotics applications require complex, real-time, high-bandwidth sensor systems. Although many such systems have been developed [12][14][17][10], integrating them into an autonomous agent architecture remains an area of active research. We will discuss how active perception systems can be integrated with agent architectures to perform complex tasks. We present an active stereo vision system integrated with a multi-tier agent architecture onboard a mobile robot. This robot "attends to" multiple humans in a complex and unstructured indoor environment.

Introduction

Many of today's autonomous agent architectures [5][11][15] contain components which interact with the world in a tight perception/action loop. The agent can act effectively if it can process its sensory input in a timely fashion. However, this task is complicated by the use of high-bandwidth sensors (when compared to sonar, or IR for example) such as vision. The amount of data available from such sensors led to the creation of active or foveal vision systems [3]. However, an active vision system cannot be attached to a robot architecture in a "plug and play" manner. Perception needs to know about action and action needs to know about perception. That is, an active perception system needs to be directed and controlled by the agent's current action or task. Action in turn, needs to know about the type and quality of information which is available from perception.

We present an active vision system based on proximity spaces [12][13] which has been integrated into an autonomous agent architecture, 3T [5]. The novelty of this system is the integration of the sensory systems with the agent architecture via a system of representation which can be effectively manipulated by the perception/action (PA) layer of the architecture, known as the skill layer.

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

Active Vision and Perceptual Memory

Active vision systems [3][2] must handle a large volume of data from the visual stream in a dynamic environment. As a means of coping with the processing demands of such environments, these systems process only a subset of the visual stream at any one time. This region, or fovea, represents the system's current focus of attention and the processing done on this region usually computes specific results designed to serve the system's current action(s).

However, the agent will rarely be able to determine all the information it needs to act from a single image region, or even the entire image. Autonomous agents need a means of remembering information recently determined from areas which are no longer the current focus of attention. They need perceptual memory systems which balance between the efficiency of a foveal, or attentive sensory system and the need to consistently update all stored information in a dynamic environment.

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

The layer(s) of the agent architecture above the perception/action layer (such as the RAP system [8]), can specify the current task by placing information into the PA layer's perceptual memory. The PA layer will act on this information to try and complete its task. The active vision system, in turn, will try and keep the perceptual memory up-to-date with the state of the world. Since the PA layer has a close

connection between perception and action, it is natural that active vision should be integrated into our architecture at the skill layer.

We propose a system of perceptual memory which stores task dependent information about recently perceived locations in the agent's environment. This memory provides the skills with a uniform interface to sensory input from the world and goal-directed control from the sequencing layer (the RAP system [8]).

Perceptual memory

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

Our system of perceptual memory is different from representation systems which operate at other levels of autonomous agent architectures [7][9]. It is composed of small, task dependent structures called markers [1][4][18]. The key element of markers is that they represent positions of objects in the agent's environment. Our perceptual memory system exists at the skill layer, allowing the skills to consult the markers to determine which action to take. The markers form the skill's interface to the sensory stream. Each marker is automatically kept up-to-date with the state of the world as long as it is in perceptual memory, without explicit action by the skills.

Markers consist of three component, called 'what', 'where' and 'identity'. A marker's 'what' component is a task dependent identifier which one or more skills use to determine commands to send to the agent's effectors. For example, a navigation skill may examine perceptual memory for a **destination** marker and any **obstacle** markers to determine the direction the agent should travel. Note that 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 the skill 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 skill uses the position stored in the **obstacle** marker to determine how to steer the agent, but in the second case, a sitting skill 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 identity component specifies how the object associated with the marker can be identified in the visual field. The 'where' and 'identity' components of a marker give the active vision system the necessary information to select an appropriate focus of attention and perform the required processing. In the case of our active vision system, the identity 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 try and 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 no longer believes the information stored there. At this point, the agent must either direct its vision system to re-acquire the associated object or drop the marker from perceptual memory.

One final aspect of perceptual memory is how markers become associated with objects in the environment. The agent's current task (typically specified by the RAP system) specifies certain roles which the agent must find objects to fulfill. For example, when the agent's task is "pour a bowl of cereal", it needs to identify objects to serve as *the* cereal box and *the* bowl. Marker's exist in 2 states, instantiated and uninstantiated. An uninstantiated marker is one which specifies a role in the agent's current task, but which has not yet been associated with a object in the environment. An instantiated marker has an association with an object and the perceptual memory is tracking that object (i.e. maintaining the 'where' component of the marker).

An Active Vision System

Since we will be using perceptual memory as a means of integrating vision into our architecture, we need to understand the particulars of our vision system to see how perceptual memory will be used.

Our active vision system is a foveal stereo vision system which confines its processing to virtual, three-dimensional regions of space called proximity spaces [12]. A proximity space maps the region of 3-space it "occupies" to corresponding 2-D regions of the left and right camera images. Within these image regions, a set of correlation measurements is made to determine disparity and motion vectors. These vectors serve as input to a set of visual behaviors

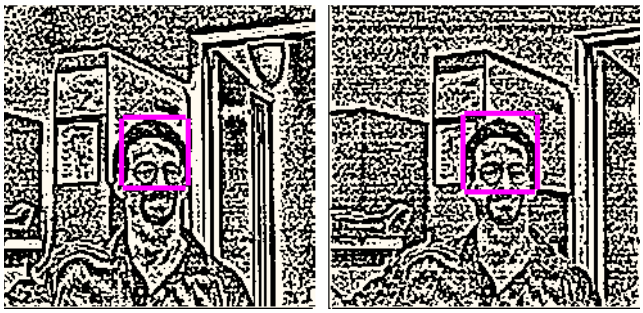

Figure 1. Left and Right LOG Images w/ Proximity Space

which control the motion of the proximity space through 3-space.

The volume of the proximity space (for this work spherical volumes were used) is represented by sub-volumes, which we call stacks. The stacks are shown in the lower right corner of figure 2. The light grey sphere represents the proximity space, while the dark grey boxes represent the stacks.

Each stack is an array of correlation measurements made between patches of the LOG filtered left and right images (see figure 1). A breakdown of the stacks is shown in figure 2. The placement of a stack within the proximity space controls which regions of the left and right images are correlated against each other. A stack's offset from the centroid of the proximity space sets the center of the region of the left image (called the reference region). The size of this region is determined by the height and width of the stack. The center and size of the region of the right image used (called the search region) are also specified by the location and size of the stack. However, the depth of the stack also effects the size of the search region. The search region is divided into sub-regions whose size are equal to the size of the reference region. These sub-regions are placed at equally spaced offsets within the entire search region. The number of sub-regions is determined by the depth of the stack.

A correlation is performed between the reference region and each sub-region of the search region. The offset in image coordinates between the reference region and the sub-region of the search region with which it is correlated represents a disparity, i.e. depth. The correlation value represents the likelihood that something is at that depth within the proximity space. The collection of correlations (between the reference region and the various sub-regions of the search region) that form a stack, make a vector which can be analyzed for peaks of appropriate strength, uniqueness and distinctiveness. Appropriate peaks indicate the presence of some object within the volume of space occupied by the stack.

A threshold on the maximum peak for each stack provides the system with a measure of occupancy for the proximity space. If the max. peak for some stack is above the threshold, there is some object within that part of the proximity

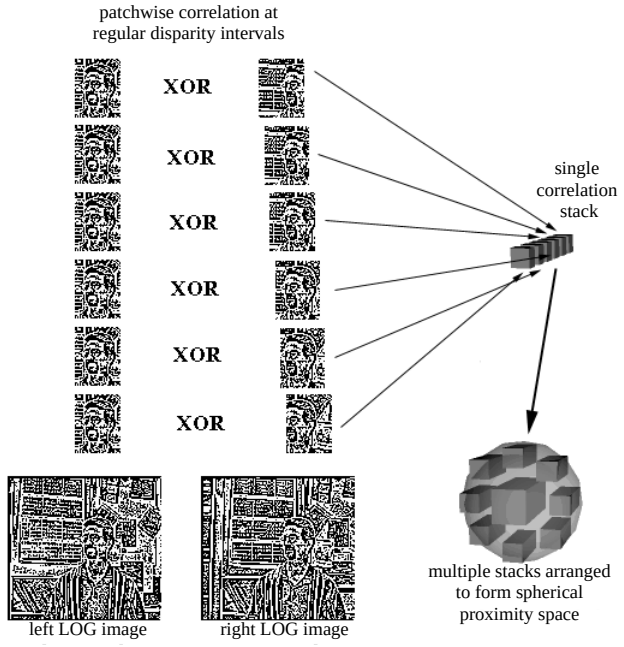

Figure 2. Proximity Space Decomposition

space at the depth indicated by the disparity of the two regions whose correlation produced the max. peak. In addition to the 9 stacks pictured in figure 2, the proximity space also contains a motion correlation stack whose correlation values are computed in a similar fashion to the disparity stacks, but with the current and previous left images being used instead of the left/right stereo pair. The peak values of the disparity and motion stacks are used by a set of visual behaviors.

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

tion of the proximity space. In [13] we developed a set of visual behaviors which are summarized below.

- Follow: This behavior takes an average of several correlation-based motion measurements within a proximity space in order to produce a 2-D vector in the direction of motion.
- Cling: This behavior is attracted to surfaces and produces a vector that tends to make the proximity space "cling" to them. This vector points in the direction of the greatest number of "texture-hits".
- Avoid: This behavior is repulsed by surfaces and produces a vector that tends to make the proximity space stay away from them. This behavior is particularly useful as a front end for obstacle avoidance.
- Lead: This behavior pushes the proximity space towards the intended path of the mobile platform. It also biases the proximity space to maintain a standoff distance from the mobile platform.
- Pull: This very simple but useful behavior produces a pull vector toward the stereo head. This vector tends to move the proximity space toward local depth minima.
- Resize: This behavior influences the size of the proximity space inversely proportionally to its distance from the robot.
- Search: This behavior 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. For example, tracking involves the cling, follow and search behaviors. The active set of behaviors determines the overall behavior of the proximity space (or proximity spaces).

An Autonomous Agent 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).

1) A set of robot specific situated skills that represent the architecture's connection with the world. The term situated skills [16] 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".

2) A sequencing capability which can differentially activate the situated skills in order to direct changes in the state of the world and accomplish specific tasks. For example, exiting a room might be orchestrated through the use of reactive skills for door tracking, local navigation, grasping, and pulling. In each of these phases of operation, the skills of the reactive level are connected to function as what might be called a "Brooksian" robot [6] -- a collection of networked state machines. We are using the Reactive Action Packages (RAPs) system [9] for this portion of the architecture.

3) A deliberative planning capability which reasons in depth about goals, resources and timing constraints. We are using a state-based non-linear hierarchical planner known as AP [7]. AP is a multi-agent planner which can reason about metric time for scheduling, monitor the execution of its plans, and replan accordingly.

The architecture works as follows, the deliberative layer takes a high-level goal and synthesizes it into a partially ordered list of operators. Each of these operators corresponds to one or more RAPs in the sequencing layer. The RAP interpreter (sequencing layer) decomposes the selected RAP into other RAPs and finally activates a specific set of skills in the reactive layer. Also activated are a set of event monitors which notifies the sequencing layer of the occurrence of certain world conditions. The activated skills will move the state of the world in a direction that should cause the desired events. The sequencing layer will terminate the actions, or replace them with new actions when the monitoring events are triggered, when a timeout occurs, or when a new message is received from the deliberative layer indicating a change of plan.

An Integrated Architecture

We have integrated a proximity space vision system with

Figure 3. Our Robot

the skill layer of the 3T architecture and embodied it in a robot agent (see figure 3). The agent is an RWI B12 with 3 cameras (2 grey scale and one central color camera) mounted on a pan/tilt platform. These cameras (and the agent's shaft encoders) are the only sensors used by the agent. The agent's task is to monitor the location of multiple (2) humans in its environment and attend to them. The robot could be bringing tools or supplies to workers on a factory floor or serving guests in a restaurant.

An Agent Application

Our agent's task is to assist multiple people in an unstructured indoor environment. The agent must locate and detect the presence of the humans based on initial estimates of their locations supplied by the RAP system. Since the humans are spread throughout the environment, the agent will seldom be able to keep all of them in its field of view at any one time. The agent must monitor and track the position of each human it is assisting. The agent will attend to each person in turn, staying a small fixed distance from them. When the agent decides to move on to a different human, it uses its stored knowledge of the person's last known position in an attempt to locate that individual. Once the human has been located, the agent can attend to it. Figure 4a shows the agent

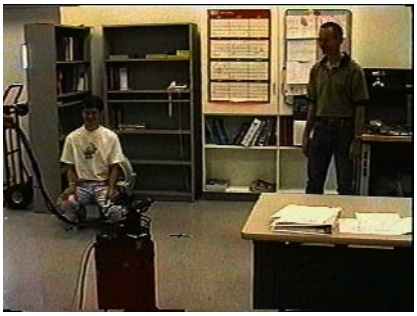

Figure 4a. The Agent's Task Environment

Figure 4b. The Agent's View

(in the foreground) with two humans. The agent is attending to the standing person. Figure 4b shows the vision system. The left and right LOG images are shown. The window in the lower left corner shows the output of the color vision system. The white areas contain human skin tones.

This application is similar to the one implemented by Franklin et. al. in [10]. While both [10] and this work integrate active vision into an agent architecture, this work concerns itself with the redirecting of the agent's attention to previously seen portions of the environment.

Perceptual Memory Implementation

Our agent's skill layer interacts with both the proximity space and color vision systems through perceptual memory. In this section, we show how the agent uses its vision systems to instantiate and track markers for use by the skills.

In this task each human is represented in perceptual memory by a marker. The RAP system places two **person** markers in the skill layer's perceptual memory. There are 2 skills which are active in the skill manager, one which controls the robot's wheels and one which controls its pan/tilt "neck". These two skills use the 'where' component of the marker to which the agent is currently attending, to move the agent within a fixed distance of the human. Each marker's confidence measure begins to decrease when its position is outside the agent's 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 it (and move toward it).

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 based on the ego-motion of the robot since the last update. If this new position should fall 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 in the marker. Otherwise the transformed position becomes the new stored position.

If the agent can track the objects represented by the markers, the question of how the association between the two is first made arises. When the markers are placed in the skill layer's perceptual memory by the RAP system, they are uninstantiated and thus contain only hypothesized positions for the humans. The color vision system consults perceptual memory to find any uninstantiated markers whose hypothesized position falls within the current field of view.

For each such marker, the central color camera is used to examine the image for skin tones (red hues) associated with humans. Image regions with an appropriate response are matched to a simple constraint model of the positions of the human head and arms. Since the color vision system is monocular, when a human is detected, its azimuth and elevation can be determined, but its depth cannot. We say a marker

with an azimuth and elevation, but no depth, is "hypothetically" instantiated. The proximity space system places a proximity space, at the minimum vergable depth, along the vector indicated by the marker's azimuth and elevation. This proximity space then "slides" along the vector performing an analysis for surface texture along the way. When the proximity space occupancy is greater than a certain value, the proximity space has come to rest on the object spotted by the color vision system. The normal proximity space tracking behaviors are now enabled and the proximity space can track the object in 3 dimensions. The associated marker is now said to be instantiated.

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 in a random spherical pattern 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 may wish to declare the marker uninstantiated and start the instantiation process again.

The agent's perceptual memory assists in this task in 3 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.

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 active 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] Agre, P.E.; and Chapman, D. 1987. Pengi: An Implementation of a Theory of Activity. *AAAI-87*: 268-272.

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

[3] Ballard, D. H. 1991, Animate Vision. *Artificial Intelligence* 48 (1): 57-86.

[4] Brill, F.Z., Wasson G.S., Ferrer, G.J. and Martin W.M. 1997. The Effective Field of View Paradigm: Adding Representation to a Reactive System. *Engineering Applications of Artificial Intelligence issue on Machine Vision for Intelligent Vehicles and Autonomous Robots*.

to appear.

[5] Bonasso, R. P., Kortenkamp, D., Miller, D. P., and Slack, M. 1997. Experiences with an Architecture for Intelligent, Reactive Agents. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2).

[6] Brooks, R.A. 1986. A Robust Layered Control System for a Mobile Robot, *IEEE Journal of Robotics and Automation*, RA-2(1):14-23.

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

[8] Firby, R.J. 1987. An Investigation into Reactive Planning in Complex Domains. *AAAI-87: 202-206*.

[9] Firby, R.J. 1989. *Adaptive Execution in Complex Worlds*. PhD Thesis. Yale University.

[10] Franklin, D., Kahn, R.E., Swain, M.J. and Firby, R.J. 1996. Happy Patrons Make Better Tippers: Creating a Robot Waiter Using Perseus and the Animate Agent Architecture. *International Conference on Face and Gesture Recognition*.

[11] Gat, E. 1992. Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. *AAAI-92*: 809-815.

[12] Huber, E. 1994. Object Tracking with Stereo Vision. *AIAA/NASA Conference on Intelligent Robots in Factory, Service and Space (CIRFFSS '94)*.

[13] Huber, E., and Kortenkamp, D. 1995. Using Stereo Vision to Pursue Moving Agents with a Mobile Robot. *IEEE International Conference on Robotics and Automation*.

[14] Olson, T.J. and Coombs, D.J. 1991. Real-Time Vergence Control for Binocular Robots. *International Journal of Computer Vision* 7(1): 67-89.

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

[16] Slack, M.G. 1992. Sequencing Formally Defined Reactions for Robotic Activity: Integrating raps and gapps. *SPIE Conference on Sensor Fusion.*

[17] Uhlin, T., Nordlund, P., Maki, A., and Eklundh, J.O. 1995. Toward and Active Visual Observer. *ICCV*: 679-686.

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