Using Stereo Vision to Pursue Moving Agents with a Mobile Robot

Eric Huber and David Kortenkamp
Metrica Inc. Robotics and Automation Group
NASA Johnson Space Center — ER4
Houston, TX 77058
{huber,korten}@aio.jsc.nasa.gov

Abstract

To interact effectively with humans, mobile robots will need certain skills. One particularly important skill is the ability to pursue moving agents. To do this, the robot needs a robust visual tracking algorithm and an effective obstacle avoidance algorithm, plus a means of integrating these two behaviors in a seamless manner. In this paper we introduce the proximity space method as a means for performing real-time, behavior-based control of visual gaze. We then show how this method is integrated with robot motion using an intelligent control architecture that can automatically reconfigure the robot's behaviors in response to environmental changes. The resulting implementation pursues people and other robots around our laboratory for extended periods of time.

1 Introduction

An important, yet difficult task for mobile robots is to pursue a moving agent while still performing obstacle avoidance. To perform this task, the robot needs a robust tracking algorithm and an effective obstacle avoidance algorithm, plus a means of integrating these two behaviors in a seamless manner. The tracking algorithm has to maintain its hold on the moving agent even when the robot is swerving to avoid obstacles. The obstacle avoidance system must arbitrate between tracking the agent and avoiding obstacles, plus adjust the robot's speed as a function of the agent's speed and distance. In this paper, we describe a mobile robot system that can pursue moving agents (people or other robots) using a new approach for controlling stereo gaze while avoiding other obstacles using a sonar-based obstacle avoidance system. These two behaviors are integrated using a software architecture that can automatically reconfigure the robot's behaviors in response to environmental changes.

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 kind of interaction is only possible with a real-time, high-bandwidth sensory system coupled with an intelligent control architecture. If either is lacking, the robot will not be capable of performing many important tasks in complex and dynamic environments. In addition, coupling a real-time, high-bandwidth sensory system to an intelligent architecture raises many important research issues. Some issues that we have dealt with in this research are: 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 approach to acquisition and tracking described in this paper uses both stereo and motion measurements; as such it is different from systems that rely solely on motion detection (e.g., [1]). Our measurements are based on correlation of visual texture rather than simply frame differencing or measuring optical flow alone. Thus, our system tracks any object of sufficient texture and does not currently require a model of the object.

In this paper we will first present the stereo vision system that the robot uses to track an agent. Then we will show how we integrated this vision system onto a mobile robot with obstacle avoidance.

2 The stereo vision system

The stereo vision system we use for pursuing consists of two B&W CCD cameras mounted on a pan-tilt-verge head, which itself is mounted on the mobile robot (see Figure 1). In order to efficiently process the enormous amount of information available from the cameras, we use techniques that have recently been developed by the active vision research community [2, 3]. In particular, we address the issue of *gaze control*, i.e., where to focus attention and visual resources. The basis of our stereo vision system is the PRISM-3 system developed by Keith

Proceedings 1995 IEEE Conference on Robotics and Automation

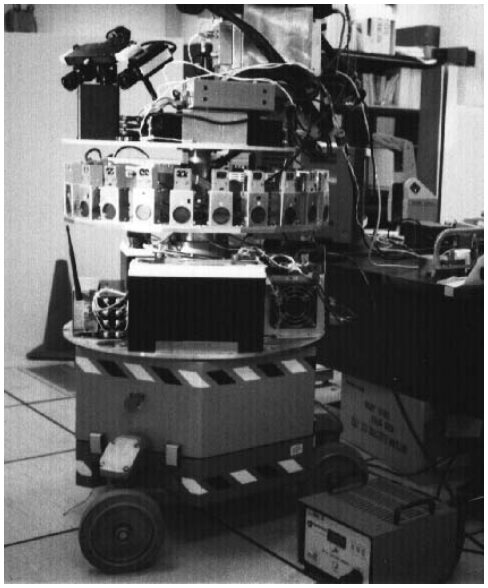

Figure 1: The stereo vision system is mounted on top of our robot; the sonar sensors are just below it.

Nishihara [4]. First, we give a short overview of the PRISM-3 vision system, then we describe how we extended the PRISM-3 system to allow it to acquire, track and re-acquire moving agents.

2.1 The PRISM vision system

The PRISM-3 stereo vision system is the embodiment of Keith Nishihara's sign correlation theory [4]. This theory is an extension of work by Marr and Poggio [5] 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. The speed and relevance of data provided by this system makes it well suited for the study of real-time active vision [6]. Other stereo vision systems (e.g., [7]) have also benefited from hardware accelerated correlation.

2.2 A behavior-based approach to active vision

As delivered, the PRISM system provided software for tracking objects in a very simplistic way. We have extended this basic software set with a behavior-based system that provides more robust tracking and also performs other tasks such as searching. This subsection describes our behavior-based, active vision system.

2.2.1 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 horopter 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 [8]. 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 nearly cubic volume, typically located in and about the horoptor, which is the proximity space (see Figure 2 d). 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.

2.2.2 Attentive behavior competition

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

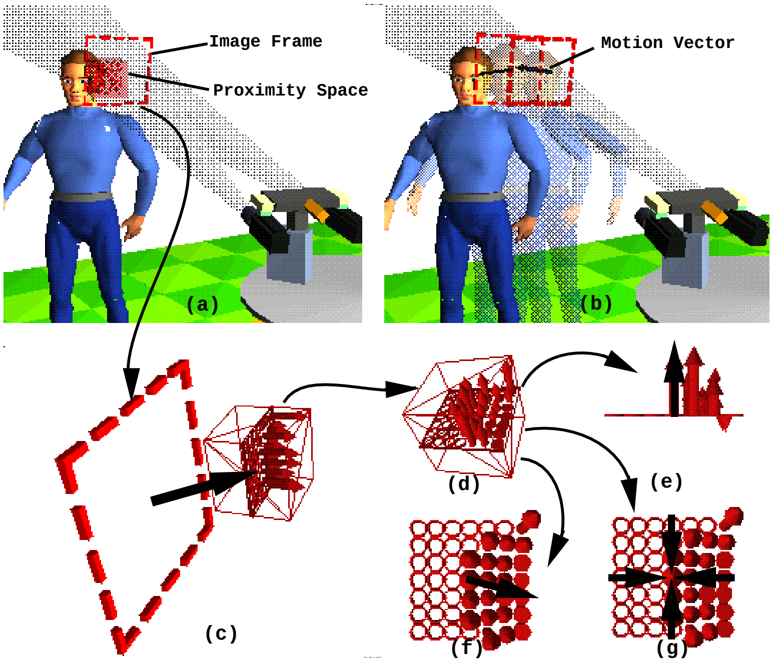

Figure 2: a. Stereo measurements within proximity space; b. Motion measurements; c. Pan-tilt-verge control reference vector; d. Texture-hits/misses within the proximity space; e-g. Examples of behavior-based control vectors.

behavior-based approaches [9], 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 effect 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 several sets of gaze control behaviors that produce several modes of robust tracking and exploratory behavior (see Figure 2 e-g). These will be described in the following subsections.

2.3 An example of using proximity spaces

To test our method we chose a task that imposes many real-time requirements and is important to mobile robots. A task well suited to our goals is that of acquiring, tracking and re-acquiring a moving agent (either a person or another robot).

2.3.1 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 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.

2.3.2 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 provided with PRISM-3 system. The first measures motion disparity and produces a 2D-vector that attempts to move the proximity space in the direction of motion (see Figure 2 b). 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 (see Figure 2 a). 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 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 texture attraction 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 texture attractor 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 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 [4]. 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 (see Figure 2 g). 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.

2.3.3 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(see Figure 2 c). 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.

2.3.4 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 10%) it indicates that the

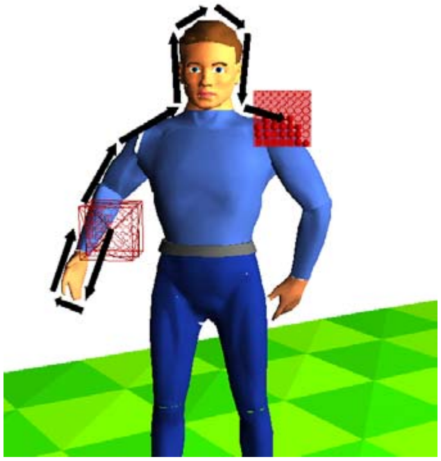

Figure 3: Tracing objects to estimate shape and size using the proximity-space method.

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.

2.4 Other complex gaze behaviors

Besides tracking, proximity-space-based behaviors can be used to control gaze for exploration, obstacle avoidance, etc. As discussed in subsection 2.3.2, the texture attractor behavior was used to make tracking more robust because it was repulsed by object boundaries. Use of this behavior without motion tracking is still capable of tracking, albeit unstably. An interesting characteristic of this crippled tracking mode is that the proximity space bounces around inside the image of the body, freely traveling until it hits an occluding contour and ricochets off to remain "in bounds". We have used this behavior in conjunction with a behavior that seeks boundaries to arrive at a resultant gaze control behavior which tends to reach equilibrium at occluding contours. The addition of an edge following behavior produced a resultant object-tracing behavior (see Figure 3). Such control schemes can readily be applied to estimate the shape and size of an object. A variation on the texture attractor, called the texture repulsor has been used for obstacle avoidance.

3 Integration onto a mobile robot

We have mounted our stereo vision system described in the previous section on a mobile robot. The robot is a Cybermotion K2A base with a ring of 24 sonar sensors (see Figure 1). All processing done by the stereo system is performed on-board the robot. The integration of the stereo vision system and our mobile robot took place within the context of an intelligent architecture we are developing for control of robots. The architecture is composed of a set of skills and methods for turning on and turning off subsets of the skill set to achieve different tasks.

Skills are defined as a closed loop of software that seeks to achieve or maintain a state (either internal or external). In our implementation, skills are small chunks of C code that take inputs, either from the environment or from other skills, and generate outputs, which are passed to the robot or to other skills. The robot's pursuing behavior is created by using the following skill set:

1. VISION: This skill communicates with the vision system as it acquires and 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).

2. TRACK-AGENT: Takes input from the vision skill and generates goal positions and velocities for the robot based on the location, distance and speed of the agent being pursued.

3. MAKE-MAP: Takes sonar information and generates as output a histogram map of obstacle positions using an algorithm described in [10].

4. FIND-FREE-DIRECTION: Takes the histogram map and the desired goal and finds a free direction of travel using the VFH obstacle avoidance algorithm [11].

5. DRIVE-AND-STEER: 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.

6. ROBOT: Gathers the robot's sensory and encoder data and passes it to other skills. Takes drive and steer commands and passes them to the robot.

When all of these skills are activated, information flows through the skill network as shown in Figure 4. The outputs of one skill become the inputs to the next skill. Some of the skills work with information obtained directly from the robot or stereo system and other skills only work on information that has been generated by

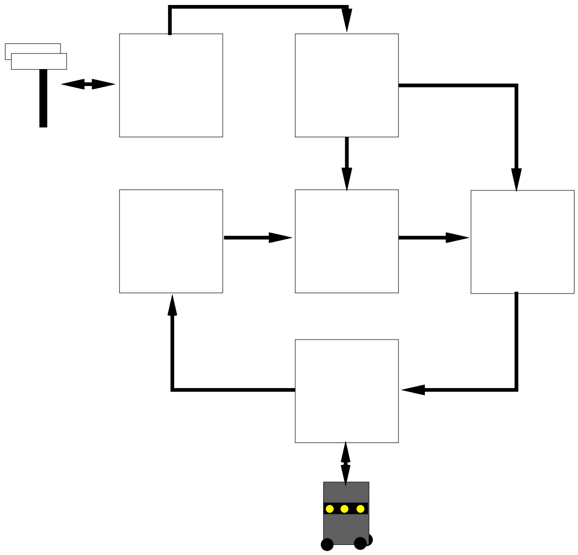

Figure 4: The network of skills that are used to perform the pursue agent task.

another skill. A system called the *skill manager* [12] maintains the flow of information through the network and automatically reconfigures the network when skills are activated or deactivated. The activation or deactivation of skills is done using the Reactive Action Packages (RAP) system [13].

3.1 Executing the pursuing task

In order to execute the pursuing task, we wrote several RAPs that activate the appropriate skills depending upon the situation. First, the VISION skill is activated and told to search a particular volume of space for a agent (the search process is described in subsection 2.3.1). When the VISION skill acquires the agent it begins reporting the agent's (x, y, z) position with respect to the robot and it also triggers the activation of the TRACK-AGENT skill. At the same time, the ROBOT, MAP-MAKING, FIND-FREE-DIRECTION, and DRIVE-AND-STEER skills are activated. The TRACK-AGENT 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-AGENT 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 VISION skill producing a new agent location four times a second. If the VISION skill loses the agent the TRACK-AGENT skill sets the robot speed to zero and the VISION skill begins an automatic search for the agent (described in subsection 2.3.1). When the agent is re-acquired, the cycle continues.

3.2 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 re-acquire 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 are often the result of three distinct circumstances: 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. To see examples of our robot pursuing agents see our video in the video proceedings of this conference.

4 Conclusions and future work

We have demonstrated a combined stereo vision and mobility system that operates interactively with humans in real-time in a real-world environment without artificial cues. In particular, we have introduced a novel, behavior-based approach to active vision and applied it to the task of tracking an agent. We have also shown that the intelligent control architecture we are developing can incorporate high-bandwidth perception with other sensing modalities, while working towards specific robot goals. In addition to these technical contributions, we discovered that the coupling of these systems produced a robot with an attentiveness that was described by those being pursued as surprisingly animate. We think this noticeable attentiveness is the result of several factors: 1) The stereo system had a behavior that caused its attention to be drawn to the head; 2) Even the slightest movements of the agent caused reciprocal movements of the stereo head; and 3) The robot's persistence in pursuing even as the agent stopped to talk or attempted to lose it.

The fact that our system does not use models to acquire or track objects means that it is capable of tracking *any* textured object. However, by not using models, certain conditions such as full occlusions may cause the system not to re-acquire the same object that it was originally tracking.

It is our intention to expand the collection of perceptual behaviors and combine them in a network to improve human/robot interaction. We envision a system that employs several proximity spaces each with its own set of behaviors revealing more globally significant data about the salient features of an object(s). For example: if three proximity spaces were free to migrate across a body, one "Northbound" one "North-eastbound" and one "North-westbound" their relative steady-state positions could provide key information about a human's pose. By this scheme the vision system could recognize a deliberate pointing stance and follow the implied pointing vector to the object of interest, possibly a new "master" or an object to be retrieved. Truly effective human/robot interaction will require many skills, of which we have only implemented pursuing at this time.

References

[1] M. R. Blackburn and H. G. Nguyen, "Autonomous vision control of a mobile robot," in *Proceedings of the 1994 ARPA Image Understanding Workshop*, 1994.

[2] D. H. Ballard, "Animate vision," *Artificial Intelligence*, vol. 49, no. 1, , 1991.

[3] D. J. Coombs and C. M. Brown, "Cooperative gaze holding in binocular vision," in *Proceedings of the Fifth IEEE International Symposium on Intelligent Control*, 1991.

[4] H. Nishihara, "Practical real-time imaging stereo matcher," *Optical Engineering*, vol. 23, no. 5, , 1984.

[5] D. Marr and T. Poggio, "A computational theory of human stereo vision," in *Proceedings of the Royal Society of London*, 1979.

[6] E. Huber, "Confidence assessment for stereo depth mapping using a PRISM-3," in *Proceedings of the SPIE Conference on Sensor Fusion and Aerospace Applications*, 1994.

[7] H. Inoue, T. Tachikawa, and M. 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.

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

[9] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, , 1986.

[10] J. Borenstein and Y. Koren, "Histogramic inmotion mapping for mobile robot obstacle avoidance," *IEEE Journal of Robotics and Automation*, vol. 7, no. 4, , 1991.

[11] J. Borenstein and Y. Koren, "The Vector Field Histogram for fast obstacle-avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, no. 3, , 1991.

[12] S. T. Yu, M. G. Slack, and D. P. Miller, "A streamlined software environment for situated skills," in *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space (CIRFFSS '94)*, 1994.

[13] R. J. Firby, "Task networks for controlling continuous processes," in *Proceedings of the Second International Conference on AI Planning Systems*, 1994.