

# A Distributed Control Architecture for Guiding a Vision-based Mobile Robot

Pantelis Elinas, James J. Little  
Computer Science Dept.

University of British Columbia  
Vancouver, BC, Canada V6T 1Z4  
(elinas,little)@cs.ubc.ca

## Abstract

*In this paper, we propose a human-robot interface based on speech and vision. This interface allows a person to interact with the robot as it explores. During exploration the robot learns about its surrounding environment building a 2D occupancy grid map and a database of more complex visual landmarks and their locations. Such knowledge enables the robot to act intelligently. It is a difficult task for a robot to autonomously decide what are the most interesting regions to visit. Using the proposed interface, a human operator guides the robot during this exploration phase. Because of the robot's insufficient on-board computational power, we use a distributed architecture to realize our system.*

## 1 Introduction

We wish to implement robots to help people, especially the disabled and elderly, in performing daily tasks. It is naive to assume that someone is familiar with the use of computers or has the time and patience to learn how to use one. We want to build robots that are easy and intuitive to use. So our focus is on developing an interface based on vision and speech, the way people engage in inter-personal communication.

The robot must perform robustly and intelligently in dynamic environments such as those designed and occupied by humans. It is often challenging to devise algorithms to achieve complicated tasks such as speech recognition and people tracking, but it is even harder to make them run efficiently and without exhausting the robot's computational resources. Since a robot is usually equipped with a single on-board computer, it does not have the computational power to run many complicated tasks. A robot control architecture is a system that allows a programmer to define the tasks, if and how they exchange information and which computer they run on.

There are many different robot control architectures, [6, 15]. In recent years, behavior-based architectures, [1, 3], have gained wide popularity for several reasons. One is the highly parallel nature of the architecture that allows the workload to be distributed among many CPU's within the same workstation or spread over multiple workstations in local area networks. It also allows the modularization of the system as each task can be developed, debugged and im-



Figure 1: Eric, the robot.

proved mostly independently of all others. These modules are called behaviors and each is designed to solve a specific problem. Often, more than one behavior must be active at a time to solve complex problems that a single behavior can not solve by itself.

Although our long term goal is to build a general interface for robots that can achieve multiple tasks, in this paper we present a first system that allows a person to guide the robot during exploration.

This paper is structured as follows. Section 2 describes the hardware we use. Section 3 discusses the hierarchical organization of the behaviors and gives a brief description of each with emphasis to the ones most directly involved with human-robot interaction. Section 4 explains how a person can get the robot's attention and direct it to a specific area to explore. In section 5 we give an example of a user directing the robot. We conclude in section 6.

## 2 The Robot

Our work is centered around a Real World Interfaces (RWI) B-14 mobile robot named Eric (see Figure 1). Eric is equipped with an Intel PIII-based computer running the Linux operating system. He senses the surrounding environment with his numerous on-board sensors. These include built-in sonar, infrared and tactile (bump) sensors.

Eric can see in 3D using a Triclops stereo vision system from PointGrey Research and in color with a SONY EVI-G20 camera. Triclops is a real-time trinocular stereo vision system that we have used extensively in our lab for tasks

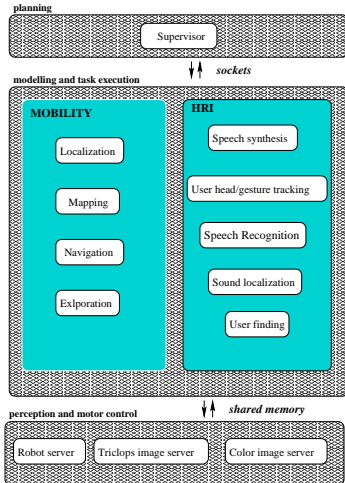


Figure 2: Diagram of the robot control architecture.

such as robot navigation, mapping and localization, [9, 10, 12].

Eric's on-board computer communicates with the rest of the workstations in our lab using a 10Mbps wireless Compaq modem.

The color camera is necessary for detecting human skin tone in images, as described in section 4.3, since the Triclops system uses gray scale cameras. The color and Triclops reference camera are calibrated so that their images match. Currently we do the calibration manually as we are in the process of upgrading to a color Digiclops stereo system that will eliminate the need for a separate color camera.

Lastly, we have added three microphones attached to the top of the the robot's body. Two of the microphones are used for sound localization and the third is used for speech recognition. The data from each microphone are wirelessly transmitted to two workstations. The maximum range of the transmitters is 50 meters. One workstation is dedicated to sound localization where the received signal is input to a SoundBlaster Live sound card. The second workstation is dedicated to speech recognition.

### 3 Robot Control Architecture

Eric is driven by a behavior-based control architecture. The entire system consists of a number of different programs working in parallel. Some are always active producing output available across the system while others only run when triggered. The behaviors are hierarchically organized in low, middle and high levels (Figure 2). We have used the same architecture with a small variation on the behavior set to build an award winning robotic waiter, [4].

Behaviors that need direct access to the robot's hardware operate at the lowest level. These modules are responsible for controlling the robot's motors and collecting sensor data. Data are made available to other modules through a shared memory mechanism on the on-board computer. This forces all low-level behaviors to run on the local computer.

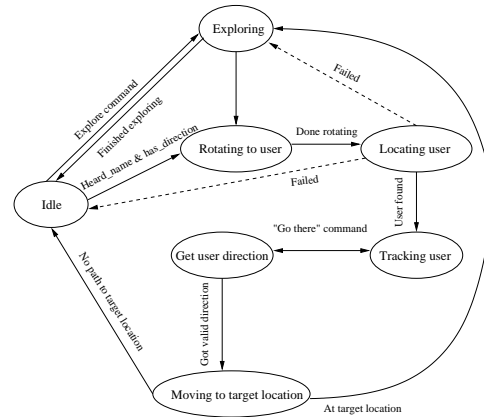


Figure 3: The finite state diagram for the supervising behavior.

The middle-level behaviors fall into two groups: mobility and human-robot interaction. The first group, mobility, consists of those behaviors that control Eric's safe navigation in a dynamic environment. There are behaviors for 2D occupancy grid mapping using stereo information, navigation, localization and exploration. The second group, human-robot interaction, consists of all the behaviors needed for Eric to communicate with people. These include speech recognition and synthesis, user head/gesture tracking, sound localization and user finding. Inputs to the middle level behaviors are the outputs of other behaviors from all levels of the hierarchy. Depending on the input requirements, i.e., if they need access to the sensor data, of each middle-level behavior, they run either on-board the robot or on a remote workstation.

At the highest level of this hierarchy exists the supervising behavior (also referred to as the supervisor.) It performs high-level planning. It satisfies Eric's high-level goals by coordinating the passing of information among the middle-level behaviors and also by selectively activating/deactivating the middle-level behaviors as needed. It communicates with all others using UNIX sockets.

Research of others has focused on the coordination of behaviors and the extension of a robot's abilities by dynamically and autonomously adding new behaviors. Our system is an exercise to developing the behavior set needed for natural human-robot interaction and so we have not added the ability for the supervisor to learn with experience. The supervisor's structure and abilities are pre-programmed and hence fixed. The supervising behavior's finite state diagram is shown in Figure 3.

### 4 Human-Robot Interaction

For Eric to function in a dynamic environment, he must first gather information about the environment's static features. We place Eric in this space and let him explore using his exploration behavior. Eric decides which areas to visit first using an algorithm that evaluates all possible locations based on the available resources i.e., battery power remaining, while trying to maximize information gain. These are

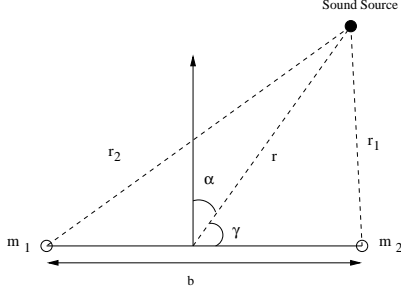


Figure 4: The geometry for the sound localization problem in 2D using two microphones. The two microphones are  $m_1$  and  $m_2$  separated by distance  $b$ .  $\alpha$  is the angle of incidence and  $r_1, r_2$  are the distance of each microphone to the sound source.

constraints that are difficult to satisfy and so we provide an interface for a person to interrupt the robot and guide it to the best location to explore.

When Eric hears his name, he responds by moving out of exploration mode and turning toward the caller. Eric turns to look at the person and applies his vision algorithm for locating people in images. Once Eric successfully locates the person, she can optionally point to a location where she wants Eric to explore. Eric navigates to that location and resumes exploration.

Eric locates and tracks a person that he is communicating with. There are three middle-level behaviors responsible for these tasks. We continue to briefly describe each of these behaviors. Then we give an example of the system at work.

#### 4.1 Speech Recognition

We use IBM's ViaVoice speech recognition engine for interpreting speech heard by Eric. ViaVoice is programmable to operate with either a speech-and-control or a dictation grammar. We use the former. We have specified a simple grammar that allows about 25 sentences to be spoken.

We programmed the speech recognition behavior using the JAVA Speech API (JSAPI) under the Microsoft Windows operating system. This behavior outputs a unique integer for each sentence in the grammar, when that sentence is heard. The input is the voice signal received over a wireless microphone. For efficiency we have dedicated a workstation just for speech recognition.

#### 4.2 Sound Localization

Eric gets an approximate idea of where a person talking to him is, by localizing on the person's voice. Sound localization is the process of determining the location of a sound source using two or more microphones separated in space.

We use two microphones at a distance of 20cm. The microphones are mounted on top of the robot. The signal captured by each of them is input to a sound card and both are processed to compute the direction to the sound source. With two microphones we can only determine the direction on the plane parallel to the floor. The sound localization algorithm we use is described in [11]. The geometry for the problem in 2D is shown in Figure 4. Since our robot operates in a 3D

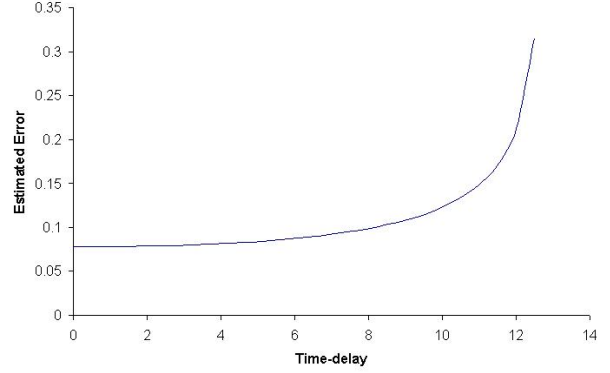


Figure 5: The sound localization error function given by Equation 2.

world, using the simplified 2D case for sound localization might at first appear inadequate; however, our experiments have shown that sound localization in this simple form performs adequately enough for the task at hand. The algorithm computes the time delay,  $n$ , between the signals captured by the two microphones. The direction to the sound source as a function of this delay is given by:

$$\sin(\alpha) = \frac{cn}{fb} \quad (1)$$

where  $c$  is the speed of sound,  $n$  is the time delay in samples,  $f$  is the sampling frequency and  $b$  is the distance between the microphones. The range of discernible angles ranges from  $\alpha = 0$  for ( $n = 0$ ) to  $\alpha = 90$  for ( $n = 14$ ). The situation is symmetric for a user to the left or right of the robot. Our setup allows for a total of 27 angles in the range  $[-90, 90]$  degrees.

The localization error can be computed by differentiating Equation 1. It is given by:

$$\delta\alpha = \frac{d}{dn} \left( \sin^{-1} \frac{cn}{bf} \right) = \left( \frac{\frac{cn}{bf}}{\sqrt{1 - \left( \frac{cn}{bf} \right)^2}} \right) \quad (2)$$

Figure 5 shows a plot of Equation 2. One notices that the localization error increases rapidly for angles larger than  $70(n = 10)$  degrees.

The output of this behavior is the direction to the sound source. It localizes on all sounds with power above a certain environmental noise threshold determined during a calibration phase at start. The supervisor monitors the output of the sound localization behavior and it only interrupts the robot (stops exploration) when there is a valid sound direction and the speech recognition behavior returns the name of the robot.

#### 4.3 People Finding

When Eric hears his name and has an estimate of the direction his name was spoken from, he turns to look at the

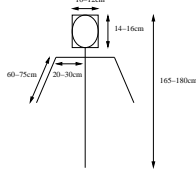


Figure 6: A simple model of a human including the dimensions of the head, shoulders, arms and overall height.

user. He must then locate the user precisely in the images captured with his two cameras.

We start with the estimate of the user’s head location guided by the sound localization results. If sound localization had no error then the user would always be found in the center of the images. Since this is not the case, we use our estimate of the localization error to define a region around the center of the image where it is most likely to find the user. This region grows as the localization error grows and it reaches maximum at angles larger than 60 degrees.

We then proceed to identify candidate face regions in the image. We segment the color image searching for skin tone, [5, 13, 14]. We transform the color image from the Red-Green-Blue (RGB) to the Hue-Saturation-Value (HSV) color space. During an offline calibration stage we determine the hue of human skin by selecting skin colored pixels for several different people. Figure 7 (a) and (b) show the original image and the skin pixels detected. We then apply a 3x3 median filter to remove noise pixels and fill-in the regions of the face that are not skin color such as the eyes and the mouth. Finally, we perform connected component analysis, [7], on the binary image resulting from the skin color segmentation.

Once all the connected components have been identified we compute a score for each component being a user’s face. The component with the higher score is selected as the user. This score is a function of three variables: a face template match cost, an expected head size functional and an expected distance of the person to the robot penalty term. Finally, the entire score is multiplied with the value of a linear function with slope that depends on our confidence to the sound localization results. Equation 3, below express this more formally:

$$s(c_i) = F_i \times (w_1 \times S_s + w_2 \times S_t + w_3 \times D_z(i)) \quad (3)$$

where  $c_i$  is the  $i$ -th component,  $w_j$  is the weight of the  $j$ -th attribute,  $F_i$  is a linear function determined by the localization error and the location of the component in the image,  $S_s$  is the size attribute,  $S_t$  is the matching cost of the component gray scale image and a template of the user’s face and finally  $D_z(i)$  is a penalty term for components that are too close or too far from the camera. The three weights are determined beforehand experimentally to yield the best results.

$S_t$  is the standard normalized cross-correlation cost computed using the image and templates of the user’s face. The user provides a full frontal view of his/her face and images are captured for use as templates. We center and scale the

template to the component’s location and size respectively and then we compute:

$$S_t = \frac{\sum_{k=1}^m \sum_{l=1}^n g[k, l] f[k, l]}{\sqrt{\sum_{k=1}^m \sum_{l=1}^n f^2[k, l]}} \quad (4)$$

where  $g$  denotes the  $m \times n$  face template and  $f$  denotes the  $m \times n$  face region in the gray scale image.

The second term,  $S_s$ , is a function of the size of the component and the expected size at the component’s distance from the robot. It is given by:

$$S_s = \begin{cases} \frac{Size_{expected}}{Size_{actual}} & Size_{actual} > Size_{expected} \\ \frac{Size_{actual}}{Size_{expected}} & \text{otherwise} \end{cases} \quad (5)$$

where  $Size_{actual}$  is the area of the component in pixels and  $Size_{expected}$  is determined using a model of a person’s head at the component’s distance. The model is shown in Figure 6.

The last term, we refer to as the distance penalty and we define it as:

$$D_z(i) = \begin{cases} 0 & \text{if } z < 3.0m \text{ and } z > 5.0m \\ \alpha \times z & \text{otherwise} \end{cases} \quad (6)$$

where  $\alpha$  is set to  $-0.5$ . So, we penalize a component the further away from the camera that it is.

Figure 7 shows an example of finding the user. Notice that the user need not be standing. Our approach can distinguish between the face and the two hand components even though they have similar sizes.

#### 4.4 User Head/Gesture Tracking

Once we identify the user’s head location in the image, we keep this location current by tracking the head over time. We also need to be able to tell where the user is pointing for Eric to move there. In the following two sections, we describe how we solve these two problems.

##### 4.4.1 Head Tracking

We employ a head tracker very similar to the one presented by S. Birchfield in [2]. Birchfield’s is a contour-based tracker with a simple prediction scheme assuming constant velocity. The contour tracked is the ellipse a person’s head traces in the edge image, [8].

We make one addition to the algorithm to increase robustness by taking advantage of the stereo information available at no significant computational cost. We begin by selecting the skin colored pixels in a small region around each hypothesized head location. Then, we compute the mean depth of these pixels and we use this value to depth segment the image using our stereo data. This allows us to robustly remove the background pixels. The tracking algorithm can run in real-time (30fps) but we limit it to running at 5fps for computational reasons especially since our experiments reveal that this is sufficiently fast.

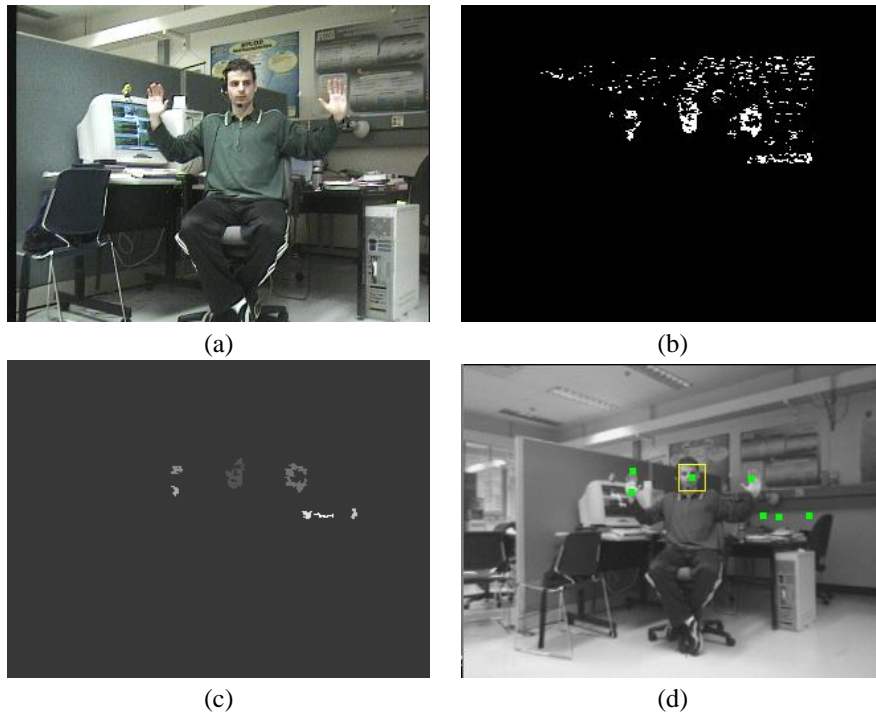


Figure 7: Example of finding the user in images. (a) The color image, (b) the skin regions, (c) the connected components and (d) the Triclops image annotated with the location of the user's head shown with a square.

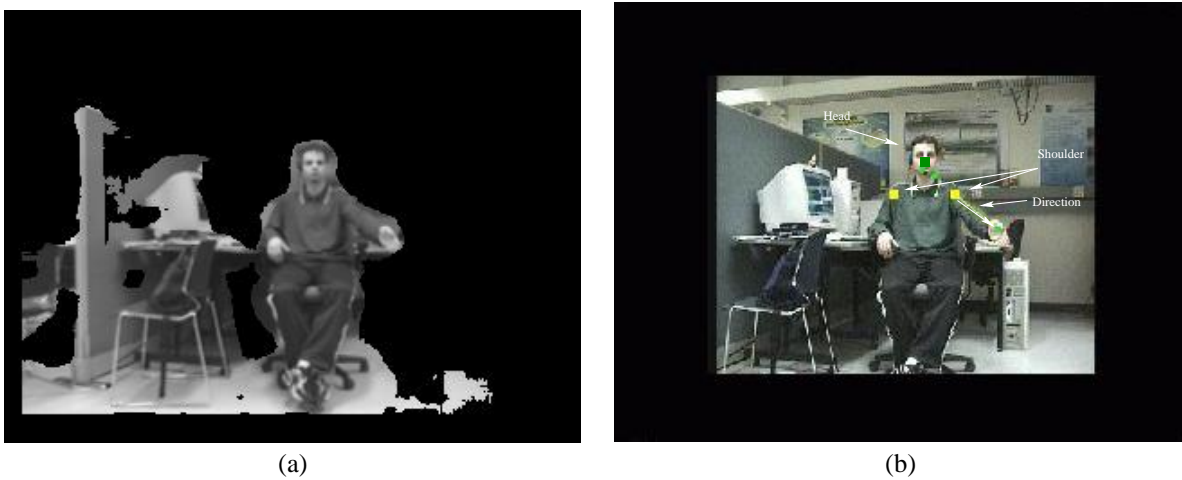


Figure 8: Example of gesture tracking showing two of the steps in the algorithm, (a) the selected foreground pixels in the gray scale image and (b) the final result showing the direction selected, and the location of the user's head and shoulders.

#### 4.4.2 Gesture Recognition

The user points to a location on the floor and instructs Eric to move there. Eric computes this location and reports the results waiting for the user's confirmation.

The process is guided by the already known location of the user's head taking advantage of color and stereo information along with our model of the body of a human (see Figure 6). We proceed as follows:

1. Get the gray, color and stereo images denoted  $I_g$ ,  $I_c$  and  $I_s$  respectively. These are the input data along with the last known location of the user's head,  $l_{head}$ .
2. Select the skin pixels in the neighborhood of  $l_{head}$ , and generate the set of skin pixels  $N$ .
3. Compute the average depth,  $d_N$ , of the pixels in  $N$ .
4. Segment the color and gray images using  $d_N$  as a guide, Figure 8 (a).
5. Select the skin pixels in the segmented image forming the set of foreground pixels  $M$ .
6. Compute the connected components in  $M$ .
7. Select the component closest to the last known head location as the face.
8. Classify the remaining components as being to the left or right of the face component. Construct the sets  $L$ ,  $R$  respectively.
9. Remove the components from  $L$ ,  $R$  that are too far or too close to the face component using the model in Figure 6 as a guide.
10. From the components remaining select the dominant direction using a heuristic based on the angle that each arm makes with respect to the person's body. Define a vector from the shoulder to the hand. This is the direction,  $f$ , the user is pointing, Figure 8 (b).
11. Compute the point on the floor using direction  $f$  and return.

If Eric fails to detect any hands, most often because of failure during the skin detection, he reports this to the user.

#### 5 Example of HRI

Eric can only understand a small number of spoken sentences. He also has a limited vocabulary i.e., he can only speak a limited number of sentences. We have kept the number of the possible sentences that Eric can recognize and speak to be small in order to speed-up speech processing and also keep his interaction with people a straightforward process. The dialog mostly consists of the user giving commands to Eric and Eric responding by confirming the heard command and/or by reporting the results of a computation. In general two-way communication may proceed as follows:

```
Person: (loudly) Eric!  
Eric : (stops motion and turns towards  
       the person)  
Person: Find me!  
Eric : I can see you. You are 4 meters  
       away and 1.6 meters high. Is  
       this correct?  
Person: Yes!  
Eric : Confirmed!  
Person: (Points with finger) Go there!  
Eric : You are pointing 2 meters ahead  
       and 4 meters to my left. Do  
       you want me to go there?  
Person: Yes!  
Eric : I am on my way! (Eric moves to  
       that location and enters  
       exploration mode)
```

This is only one instance of the possible dialogs a user can have with Eric. At any time during the interaction, the user can simply instruct the robot to return to exploration. Additionally, the user may request status information about Eric's mechanical status. Eric might also fail in one of the tasks of finding and tracking the user. In that case, Eric reports this and waits for new instructions.

#### 6 Conclusions

In this paper we presented a distributed system for natural human-robot interaction. It is designed to perform in real-time by distributing the workload among several workstations connected over a local area network. People can communicate with the robot using speech and hand gestures to direct it to specific locations to explore.

We are currently working on improving the interaction process by allowing more sentences to be understood and spoken by the robot. We are also adding to the number of gestures that can be understood and finally more software for modeling the user's emotional state by analyzing his facial expressions.

#### Acknowledgments

Special thanks to Jesse Hoey for reviewing this paper and Don Murray for his help with programming the robot.

#### References

- [1] R. C. Arkin. *Behavior-Based Robotics*. Intelligent Robots and Autonomous Agents. The MIT Press, 1998.
- [2] S. Birchfield. An elliptical head tracker. *31st Asilomar Conference on Signals, systems and Computers*, pages 1710–1714, November 1997.
- [3] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–33, 1986.
- [4] P. Elinas, J. Hoey, D. Lahey, D. J. Montgomery, D. Murray, S. Se, and J. J. Little. Waiting with jesse, a vision-based mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'02)*, Washington D.C., May 2002.

- [5] B. Funt, K. Barnard, and L. Martin. Is machine colour constancy good enough. *5th European Conference on Computer Vision*, pages 445–459, 1998.
- [6] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, (26):251–321, 1985.
- [7] R. Jain, R. Kasturi, and B. Schunck. *Machine Vision*. MIT Press and McGraw-Hill, Inc., 1995.
- [8] D. Marr and E. C. Hildreth. Theory of edge detection. *Proc. R. Soc. Lond. B*, 207:187–217, 1980.
- [9] D. Murray and C. Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proc. ICRA*, Albuquerque NM, 1997.
- [10] D. Murray and J. Little. Using real-time stereo vision for mobile robot navigation. In *Proceedings of the IEEE Workshop on Perception for Mobile Agents*, Santa Barbara, CA, June 1998.
- [11] G. L. Reid and E. Milios. Active stereo sound localization. Technical Report CS-1999-09, York University, 4700 Keele Street North York, Ontario M3J 1P3 Canada, December 1999.
- [12] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2051–2058, Seoul, Korea, May 2001.
- [13] K. Sobottka and I. Pittas. Segmentation and tracking of faces in color images. *Proc. of the Second International Conference on Automatic Face and Gesture Recognition*, pages 236–241, 1996.
- [14] J.-C. Terrillin and S. Akamatsu. Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images. *Vision Interface*, pages 1821–1828, 1999.
- [15] J. K. Tsotsos. Intelligent control for perceptually attentive agents: The S\* proposal. *Robotics and Autonomous Systems*, 5(21):5–21, January 1997.