Alfred: The Robot Waiter Who Remembers You

Bruce A. Maxwell, Lisa A. Meeden, Nii Addo, Laura Brown, Paul Dickson,
Jane Ng, Seth Olshfski, Eli Silk, Jordan Wales

Swarthmore College
500 College Ave.
Swarthmore, PA 19081
maxwell@swarthmore.edu, meeden@cs.swarthmore.edu

Abstract

Alfred the robot won first place in the Hors d'Oeuvres Anyone? event and also received an award for the best integrated effort at the 1999 American Association of Artificial Intelligence robot competition. The three unique features that Alfred displayed were: the ability to nudge his way through a crowd to cover a large serving area, a strong personality--that of a proper British butler, and the ability to recognize people he had served before. This paper describes the integrated navigation, natural language processing, and vision system that enabled these capabilities.

1 Introduction and task definition

The American Association of Artificial Intelligence (AAAI) holds a national robot competition at their annual conference. This competition draws schools from around the country, and the event is judged by researchers and academics in the fields of artificial intelligence and robotics. This year's competitions included the *Hors D'oeuvres Anyone?* event which required robots to serve hors d'oeuvres to conference attendees during the main conference reception. The primary objective of this event was to have the robots unambiguously demonstrate interaction with the spectators. To evaluate the event, the judges looked at not only how the robots did during the final round at the reception, but also interacted with the robots during a preliminary round. This initial round gives the judges a chance to systematically test the full capabilities of each entrant in a more controlled setting.

In 1999, this event was held at the Orlando Convention Center. The area where the robots were required to serve was extremely large--approximately 45m x 45m, and the ceilings were 15-20m high. There was no sound dampening material on the walls, ceiling, or floor. The illumination for the event consisted of directional lights shining down on the floor, that alternated between "cool" and "warm" colors every 5m. All of these factors made the evening event extremely challenging for vision sensing and speech interaction.

Alfred the robot was designed and constructed during a 10 week period prior to the competition by the authors, all of whom were either faculty or students at Swarthmore College. Two students worked primarily on the speech interaction, three on the visual sensing, and two on navigation and integration. Complete integration of the parts took four weeks to accomplish Prior to the competition we had one "live" test run which gave us a benchmark and allowed us to focus our efforts on particular areas highlighted by the test. One of the major lessons of this experience was the need to begin integration even earlier in order to have a base platform from which to work.

The remainder of this paper outlines Alfred's technical details. Section 2 highlights the physical design, section 3 the navigation and decision-making algorithms, section 4 the speech interaction, and section 5 the vision system. Section 6 presents an overall discussion and future directions of research and development.

2 Physical design

The heart of Alfred's physical design was a Nomad Super Scout II Mobile Robot, manufactured by Nomadics, Inc. The base configuration comes with a 233 MHz Pentium II processor, built-in sound, and a Captivator video frame grabber. The computer runs Linux--Red Hat 6.0 distribution--and links to the robot's microcontroller through the serial port.

On top of the base robot we built a penguin-shaped structure out of plywood, screen, and black & white felt. A shelf inside the penguin held the amplified speakers, microphone power supply, and the interface box for the nudging device. Alfred appears in Figure 1.

The sensors attached to Alfred included a Costar CCD color camera with an 8mm lens, a Shure MX418S supercardioid (highly uni-directional) gooseneck microphone, and a rigid aluminum bumper with 5 contact switches for sensing feet. We installed the bumper on the front of the robot at a height of 4cm off the ground and interfaced the contact switches to the computer through the parallel port. Note that the bumper also kept the robot from pitching forward as there is no support wheel in front of the robot.

The other modification we made was to add a fan to the top of the Scout base to lower the internal temperature. This proved to be essential to running the robot under the load we gave it.

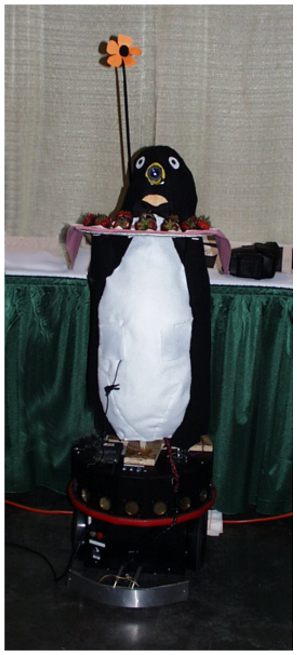

Figure 1 .Alfred the Robot (left), and Alfred serving hors d'oeuvres at the 1999 AAAI Conference Reception (right)

3 High-level control and navigation

It was our task to develop a software architecture that, based upon sensor data gathered from its environment, would respond in an intelligent manner. The goal was to create an autonomous system that would serve hors d'oeuvres while covering a large area of the room, and seeking out and interacting with people.

The robot had to be able to differentiate people from obstacles, offer food to people it encountered, cover a wide area, detect when more food was needed, and navigate to the refill station. Based on the situation in which the robot found itself, we wanted it to make appropriate commentary and engage people in some form of conversation. We extended the robot-human interaction to include nudging people out of the way when they blocked the robot's path.

When choosing the architecture for Alfred's behavior, there were several possible approaches. A common technique in mobile robotics is to develop a subsumption architecture where a set of task-achieving behavior modules are constructed and layered so that higher priority behaviors subsume control of lower priority behaviors by suppressing their outputs [2]. Traditionally, each behavior module in a subsumption architecture is a finite state machine (FSM). The subsumption style of architecture is quite robust and reacts quickly to changes in the environment. However, development must be staged, starting from the simplest behaviors and gradually adding more complex behaviors.

Due to the short time frame we had to prepare, we chose to construct a single, high-level FSM instead. This was also the technique used by the 1998 contest winner, Rusty the B.E.A.R. from the University of North Dakota [7]. A FSM lends itself well to the situation of controlled interaction for which we were developing the robot

By integrating the components of speech, vision, and navigation through the FSM, we were able to accomplish our goal of a basic serving behavior. Upon startup, the robot moves away from its refill station toward a guidance point set for it in advance. Upon reaching this point, the robot attempts to detect people in its field of view. When it finds them, it moves toward them and engages in conversation. If it has seen the person before and recognizes them, the robot acknowledges this. New people are given nicknames that the robot uses later when speaking to them again. It asks them if they would like an hors d'oeuvre, and demands proper decorum in their reply. If necessary, the robot avoids, ignores, or nudges people in order to cover a broader area. When the robot has served a predetermined number of people, it navigates back to the refill station by looking for a landmark, asking directions, and using dead reckoning. After refilling, it moves back onto the floor to continue serving at the next unvisited guidance point.

3.1 Algorithms & theory

Navigation and most localization of the robot is accomplished using wheel encoders. Obstacle detection is accomplished by sonar and people are detected and recognized visually. As the robot navigates through a crowd, the FSM directs its general path toward a 'guidance point'. There are three types of guidance points – general, intermediate, and

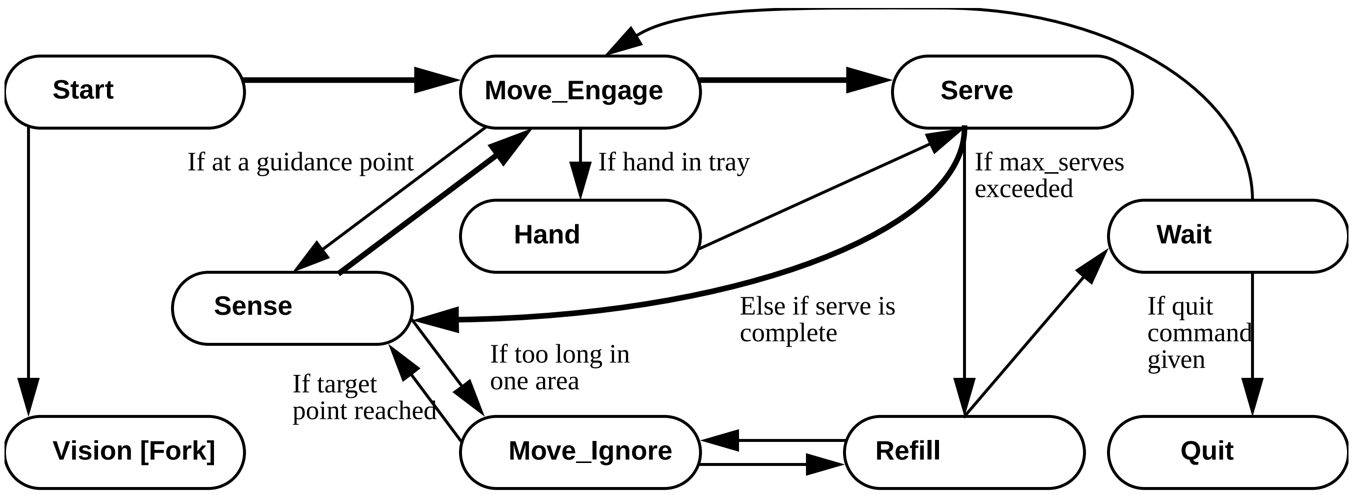

Figure 2 .The overall finite state machine for the robot. The primary action pattern is highlighted.

imperative. The general guidance points form a path around a room. They are set for the robot prior to initialization by a person and depend upon the serving environment. Intermediate guidance points are dynamically calculated during the processes of the FSM. They are used to direct the robot toward a certain point from which it can resume its course toward the general guidance points. When traveling to non-imperative guidance points, the robot will stop to engage people it detects along the way. Imperative guidance points are points to which the robot moves while ignoring people it meets along the way, and avoiding them or nudging if necessary. These guidance points allow the robot to move somewhere even when it is hemmed in by people that it would usually stop to serve. The robot avoids inanimate objects by testing if the thing directly in front of the robot displays visual human characteristics.

The FSM keeps track of the robot's position on a map stored in memory. When checking to ensure that the robot covers a certain area, the bounding box of the path covered on the map is calculated. The pixel size of the map is determined by a predefined 'resolution' value which indicates the size in inches of a pixel on each side. This value can be changed in the program code so that maps can be highly detailed if necessary.

A diagram of the overall FSM is given in Figure 2. The robot begins in state *Start*, from which the vision fork is initiated. Vision processes are run independently, with commands being passed to Vision from the FSM via a shared memory structure.

From state *Start*, the FSM goes into state *Move_Engage*, in which the robot moves away from the refill station, but will stop and serve any people encountered along the way (or hands that are detected in the serving tray). If the robot reaches its general guidance point, the FSM will go into state *Sense*, which searches for people to serve and then goes into state *Move_Engage* to move in the direction of a possible person with an intermediate guidance point.

Every three minutes, the FSM checks to ensure that the robot is moving outside a certain predetermined area. If that area has not been exceeded, the FSM goes into state *Move_Ignore* to navigate toward a dynamically-determined imperative guidance point that takes the robot outside its previous area. Upon reaching this point, the robot re-enters state *Sense*.

After a serve, if the FSM detects that the maximum number of serves have been exceeded, it goes into state *Refill* which uses state *Move_Ignore* to navigate back to the refill station without engaging people. At the station, the FSM goes into state *Wait*, from which the robot can shut down if such a command is issued. Otherwise, when the tray has been refilled, the FSM goes into state *Move_Engage* to move away from the refill station, and the cycle repeats as before.

3.2 Integrating vision, speech, and navigation

We took a practical approach to integrating the three main components of the robot: vision, speech, and navigation. Since we did not take a reactive or behavior-based approach to navigation, the FSM directly controlled the navigation component. The Move_Engage and Move_Ignore states, in particular, controlled the motion of the robot based on the current goal location and sensor readings. Each of the other states either required the robot to be still, or to be moving in a particular manner--such as rotating to look for the refill station.

The vision and speech components, on the other hand, were developed independently of the FSM and navigation, in part because a modular approach permits more efficient development and testing. Both vision and speech ran as separate processes--vision as a fork, and speech as a set of independently called programs.

To engage speech, the FSM would start the appropriate speech program and then read the results of that call to determine the speaker's response if the program engaged in recognition. This worked well because the speech generation and recognition could not be running simultaneously due to conflicts with the sound resources. The modularity of this approach allowed us to test and refine each speech interaction independently.

The vision process, by contrast, was a simple state machine of its own, running in parallel with the main FSM. The FSM controlled the vision process by setting a command field in a shared memory structure. This command field would trigger the vision process to grab image(s), execute the requested operator, and return the results in the shared memory field. Some operators--like testing for hands in front of the camera--were free running and would continue processing images until told to stop. Other operators--such as searching for a badge--executed only once. Because of the modularity of this approach, we could run the vision process independently of the main FSM, which facilitated development and testing.

Overall, the major principle of integration we followed was modularity. We also gave the robot the ability to accomplish a sequence of complex tasks by using the central FSM for high-level control. The weakness of this approach was the speed and reactivity of the robot to its surroundings since the sensing processes started and stopped based on the FSM state. These general observations are echoed by Bryson's comparative analysis of robot architectures, which claims that they should have:

- a modular structure,
- a means to control action and perception sequences for complex tasks, and
- a means for reacting quickly to its environment [3].

In response to the weaknesses of the pure FSM approach, we have since evolved to a completely modular architecture with sensing and acting processes that run continuously in parallel with a central controller.

3.3 Experiments & results

To test the integrated functioning of all elements within the FSM, we put the robot into several situations involving varying numbers of people with whom the robot would have to interact. Early in the testing we found that vision was the slowest process, so we coordinated speech to overlap while vision was running, camouflaging the delay. Note that a part of this delay was due to the fact that in most of the test situations we were logging images to disk to keep track of the accuracy of the vision processes.

We first tested the robot under indirect natural lighting in a large room containing 7 people spread throughout. The robot was able to navigate to its guidance points and engage people along the way. These tests did not include a situation where the robot was completely surrounded by a group of people. Also, there wasn't any background noise to confuse the robot's speech recognition. The robot achieved 70% accuracy in person recognition, 90% accuracy in person detection, and 75-85% accuracy in speech recognition which varied from person to person.

The robot was also tested in an incandescently-lit room with about 50 people. Again, the room was very quiet, and the people gave the robot space so it could move with ease. This test too was successful, however, as we did not log images--logging images slowed down the robot--for this exercise, this is a purely qualitative assessment.

The conditions of the competition were quite different. The lighting was more bluish, which dulled many of the colors picked up by the camera, and varied from place to place across the competition area. In the first round of the competition, where only the judges and a few onlookers were involved, the robot worked almost flawlessly from the point of view of the audience--again, we did not log images because of speed considerations. Although the robot was presented with a larger number of people, they were all attentive to it and there was relatively little background noise. The robot was given plenty of room to move, which allowed it to navigate freely.

In the second round of the competition, during the AAAI conference reception, it was much more difficult for the robot to be successful. The robot was often hemmed in by jostling people, causing it to rotate in circles searching for a way out. The nudging algorithm turned out to be too nice, not sustaining its nudging long enough to get the robot out of the circle if the people were not accommodating. As noted below, the worst of the robot's problems, however was background noise, which greatly inhibited the speech recognition and conversation aspects of the interaction.

In most of the trial runs, navigation encountered significant errors in dead reckoning, caused by wheel skids while nudging and occasionally bumping objects. Obstacle avoidance was still good; there was only one collision with an inanimate object. The errors in dead reckoning, however, were offset by the robot's ability to find the refill station landmark, go to it, and then reset its world coordinate system before heading out again. This way it was able to correct its dead reckoning errors every 10-15 minutes.

The greatest problem revealed in the methodology we employed was that of speed of interaction. Many people walked quickly past the robot and took an hors d'oeuvre without stopping. This triggered the speech interaction, but by the time the robot spoke, people were already long gone. Thus, the robot would wait for an answer from a person who was no longer present or willing to speak to it.

4 Speech interaction

Alfred's speech system was developed primarily to act as the interface between human and machine. It was

```
<<root>> = <affirmative_1> | <negative> | <vernacular> | <properYes> |
<properNo> | <affirmative_2>.
<affirmative_1> = yes: 1 | yeah: 1 | “yes i really would like a tasty
snack”: 1.
<affirmative_2> = “thank you very much”: 6 | “why thank you”: 6.
<negative> = no: 2.
<vernacular> = okay: 3 | sure: 3 | “why not”: 3 | “i guess so”: 3 | “of
course”: 3 | cope: 3.
<properYes> = “yes please”: 4 | “but of course”: 4 | certainly: 4 | “of
course”: 4 | “yes” “please”: 4.
<properNo> = “no thank you”: 5 | “I’m fine thank you”: 5 | “I’ll pass”: 5 |
“I’ll pass thanks”: 5.
```

Figure 3 .An example of a BNF file. This file was the grammar file for the serving interaction.

through speech that all the various human interactions were carried out. We decided to augment this interaction by making it more lifelike. As such the entire speech system served to build Alfred’s “British butler” personality. These goals were all achieved using IBM’s beta version of the ViaVoice software development kit (SDK) for Linux [6], and standard audio playback software that comes with Redhat release version 6.0.

4.1 Speech recognition system

ViaVoice for Linux is available for public download from IBM’s website [6]. We had the choice of using IBM’s speech kit or the Center for Spoken Language Understanding (CSLU) speech toolkit developed at the Oregon Graduate Institute of Science & Technology (OGI). ViaVoice was chosen because of its simplicity to implement and high-level interface that focused more on the abstract features of speech recognition. Unlike CSLU, ViaVoice did not require the programmer to specify any low-level preprocessing techniques of the audio file before recognition was performed; the ViaVoice engine performed all this preprocessing. Another factor that contributed to our choice of ViaVoice was the ease of development of the grammar file. In speech recognition an utterance is a stream of speech that represents a command. A grammar file is a set of words and phrases governed by rules that define all the utterances that are to be recognized at run-time. In ViaVoice there was no need to make additional inputs of pronunciations, since there was a built in dictionary of pronunciations. On the other hand, CSLU required this additional input. The major drawback of ViaVoice was that it relied greatly on the quality of the spoken utterance, and therefore the environment needed to be reasonably quiet to achieve high recognition rates. This was in part due to the fact that all of the preprocessing was performed by the engine and therefore we were unable to modify the filters to suit our environment. Furthermore the ViaVoice input came directly from the microphone and not an audio file. We obtained help in understanding the ViaVoice SDK from accompanied documentation and the developers at IBM.

4.2 Speech interaction method

All of the speech interactions were pre-defined and based on scripts that we wrote. Each of these scripts was associated with a stand-alone speech program, and it contributed to the development of Alfred’s personality. The stand-alone programs had a specified finite state grammar (FSG) file, which contained the words and the phrases to be recognized by the ViaVoice recognition engine. These FSG files were the compiled output of Backus-Naur Form (BNF) files. These BNF files are simple, but structured text files, written in a speech recognition control language (SRCL and pronounced “circle”). The Speech Recognition API Committee and Enterprise Computer Telephony Forum jointly developed SRCL. The general form of a SRCL grammar file consists of production rules in the form of (1)

$$< \text{rule}> = \text{words or “phrase”} \tag{1}$$

The left side of the production rule is synonymous to a variable name and the right side specifies the individual words or phrases (given in quotes) that are to be recognized. An example taken from one of Alfred’s BNF files is given in Figure 3. This example annotates each recognized word and phrase with an integer, so that our speech programs could more easily parse them. More information on SRCL grammars can be found in the ViaVoice SDK documentation.

An FSG file was written for each of Alfred’s primary interactions including the “serving interaction,” the “search for the refill-station interaction” and the “at the refill-station interaction”. Each FSG file had many variations of responses to questions like “Would you like an hors d'oeuvre?” and “Where is the refill station?” We also developed a generic FSG file to interpret yes/no type responses. The associated program for this FSG file, served to confirm output from the vision system. No explicitly defined speech algorithms were used in developing these programs. However, each speech interaction tree was based on production systems, with if-then-else and case statements specifying what response was made based on a recognized utterance.

Work by Clifford Nass proposes that people tend to respond psychologically to computer personalities in the same way that they respond to human personalities [8], we decided to make recorded human responses for Alfred as opposed to using a text-to-speech synthesizer, thereby achieving a more convincing "human personality". To add to Alfred's anthropomorphic nature, we made a minimum of five audio files for each response and one was selected randomly at runtime of the speech program. Consequently no two runs of the same speech program were alike, since different audio files were played back at runtime.

4.3 Experiments & results

Tests of the speech interaction were quite good in the laboratory, achieving approximately an 85% recognition rate. This number takes into consideration the fact that all the speech programs were designed to make three attempts at recognition per question asked, given that the first attempt failed. However, this was not the case at the AAAI reception where the final round of the hors d'oeuvres competition took place. Recognition rates dropped significantly to about 35% due to the very loud background noise in the conference hall, in spite of the unidirectional microphone used. Another factor that may have contributed to this drop was Alfred's onboard sound system. The built-in audio system, developed by ESS Technologies, was perceptibly low in quality compared to the 64-bit Creative Labs sound card used in the laboratory.

Our decision to use recorded human responses proved successful, and Alfred was referred to by his given name and not treated like a machine. In fact, some guests proceeded to talk casually to him as if he were a real person. Consequently, they talked to him in full sentences instead of the short phrases or single words which Alfred was designed to understand.

5 Visual sensing

5.1 Detecting conference VIPs

With a single color camera, Alfred used blob-detection to identify conference VIP's by detecting colored ribbons on their badges. For example, note the ribbon hanging below the badge of the person in the center of Figure 1. The color blob detection process searched over an image and compared single pixels with the target color, so calibration for specific lighting conditions was necessary.

5.1.1 Relevant work Blob detection is a standard task in vision and robotics. In a project similar to ours, a NASA mobile robot that strives to recognize faces, Wong *et. al.* [13] used just color information to detect blobs. The blob detection simplified the search for people by requiring people in the testing environment to wear a sweatshirt of a specific color. The robot used a chromaticity comparison technique to detect the color of the sweatshirt.

Chromaticity is dependent on color and not intensity. For our ribbon detection, instead of using chromaticity we used RGB color bounds. The reason for this was that the specific range of target "colors" for detection were a non-linear mix of intensity and brightness, since some color bands had greater variation than others. Furthermore, the RGB color space worked well for this task, and we avoided the extra computation by not using a different color space.

5.1.2 Algorithms & theory The blob detection function takes as input the pixels of an image, and the RGB color bounds of the blob it is searching for. First a loop is run through the pixels, counting the number of pixels which fall within the color bounds in each column and summing the results into bins. A window of specified width is then scanned across the bins, finding where the most target pixels are at within a localized region. If the result is above a given threshold, then the function returns a 1 and the leftmost column location of the blob, otherwise, 0 is returned. This function is called when there is a person directly in front of the robot. The image is, therefore, already entirely that of the person's torso. This method is significantly faster than scanning a box across the image, because each pixel is only processed once.

The RGB color bounds were determined by using a linear search algorithm. The program needed seven parameters for simple blob detection, the low range and the high range of each color band of the target color, as well as the cutoff threshold. The linear search algorithm searches for the specified number of iterations over all of the parameters one at a time for the best solution, as specified by an evaluation function. The evaluation function takes as arguments the number of parameters and the value of the parameters and returns a value that should increase as the solution improves. A training set of twenty images containing both positive and negative images is taken under the lighting conditions of the test site and run through the evaluation function. Since the RGB values of the target color may vary under different lighting situations, a calibration using the linear search function should be run before detection is needed in a new location.

5.1.3 Experiments & results One of the biggest hurdles of computer vision with color is its dependence on illumination. As expected, the blob detection processes had to be calibrated at the operation site. The pink ribbon detection was extremely accurate after appropriate calibration, with no false positives, and it found all visible ribbons in our

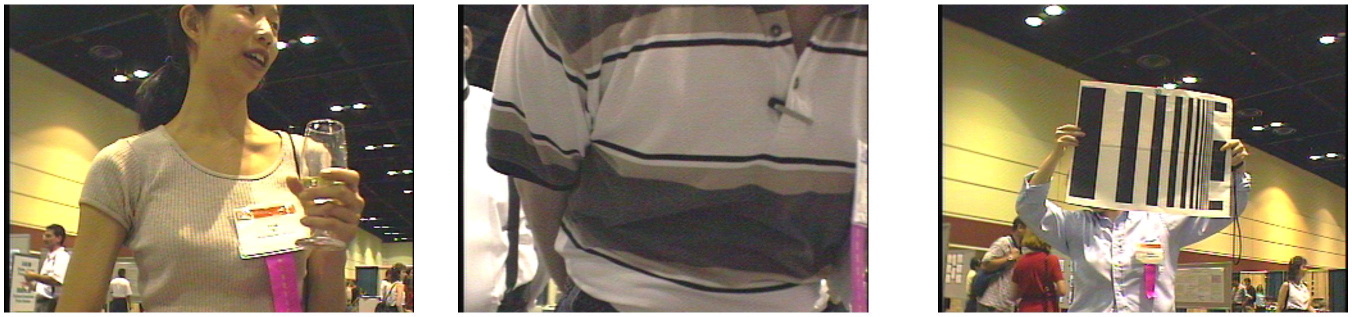

Figure 4 **Images from the final round of the competition. The left and center images were successful badge detections, while the right-most image was a successful landmark detection.**

logged images. During the competition we only looked for pink ribbons since the other important ribbon color, white, could not be consistently detected. Figure 4 shows examples images of successful badge detections. Note that in Figure 4(b) the badge is barely visible in the lower right of the image, but the system was still able to detect it because of the large vertical extent.

5.2 Recognizing landmarks

When Alfred ran out of food on its tray, it used vision along with confirmation from the handler to recognize a distinctive black and white landmark placed above its initial starting point to guide it back to the refill station

5.2.1 Relevant work The landmark detection method we used was designed by D. Scharstein and A. Briggs [10] at Middlebury College. They developed a robust algorithm that recognizes self-similar intensity patterns that works under a wide range of viewing and lighting conditions in near-real time.

5.2.2 Algorithms & theory Self-similar intensity patterns are based on self-similar functions. The graph of these functions are identical to themselves scaled by a constant *p* in the horizontal direction. A property of self-similar functions is that they are also self-similar at a scale of *pk*, meaning that the self-similar property is invariant to viewing distance.

This method operates reliably on single scanlines without any preprocessing and runs in near-real time. Since the method uses single scanlines, it successfully recognizes the landmark even when part of the pattern is being occluded.

We used a pre-compiled program obtained from Middlebury College which takes as input any PGM image, and if a self-similar landmark is found, returns the pixel locations of the two Xs marking the vertical height of the rightmost strip of the landmark. After some experiments, as described below, we were able to convert the pixel locations to a bearing and approximate distance to the refill station. We used knowledge of the camera's field of view to calculate bearing and an empirically-calculated equation to find the distance based on the vertical height of the detected landmark. The distance equation was derived by taking pictures of the landmark at known distances and fitting a function to the data, knowing that the vertical height is inversely proportional to distance.

5.2.3 Experiments & results The landmark recognition worked remarkably well. In analyzing the capabilities of the self-similar pattern recognition program, we determined that if we used the 8.5" x 11" pattern provided, we could get reliable results--better than 90% correct detection and localization--from up to 10 feet away using 320x240 images. To use this method for refill station recognition, we needed to customize it so it recognized the landmark at least 40 feet away. Since the detection of the landmark is limited by the number of landmark pixels per scanline, we doubled the size of the landmark and captured 640x480 grayscale images for this purpose, increasing the detectable range to about 50 feet.

During the competition, the landmark detection worked well enough that, although sometimes the landmark was partially blocked by a head or a hand in the conference, it still returned a reliable bearing, as judged by the direction the robot headed after each successful recognition. The approximate distance returned, however, was not as reliable since a few occluded pixels meant several feet of miscalculation. To compensate for this, Alfred would ask whomever was nearby if it was at the refill station. If the reply was negative, the robot would repeat looking for the landmark. Figure 4 shows an example image from a successful landmark detection during the final competition.

5.3 Locating people

As Alfred's primary task was to serve people, he had to have a robust, fast, and accurate person detection algorithm. In addition, to make the interaction more interesting we developed a short-term recognition algorithm based on people's clothes. The person detection combined two independent methods: one used movement detection, the other used skin-region detection combined with eye template match-

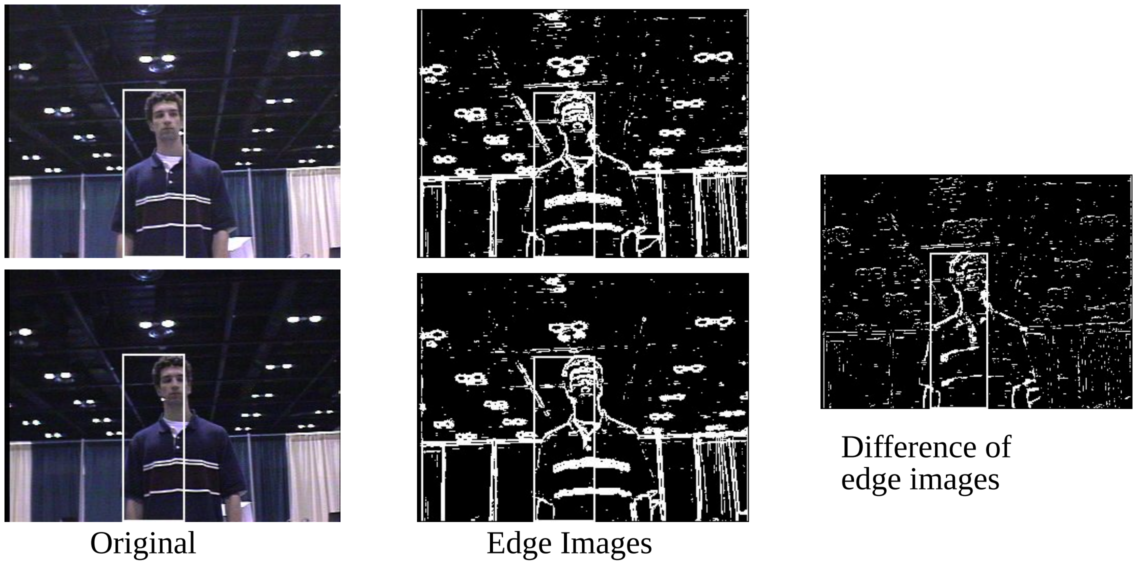

Figure 5 .Person Locator -- Successive captured images (left), calculated edge images (middle), and one difference image (right).

ing. The combination of these two methods provided more robust and accurate results than either method by itself.

5.3.1 Relevant work The human locator based on movement detection was modeled after the vision system used in Rusty the B.E.A.R., the 1998 hors d'oeuvres serving robot from the University of North Dakota [7]. We considered a neural network-based detector [9], but the movement detector was chosen for its speed and because it does not require an extensive search through an image. In addition, the movement detector is simpler since there is no explicit training needed for this type of system.

The person detection based on skin detection combined work on prefiltering images [4], and fuzzy pattern matching [14]. The fuzzy pattern detection was used as a fast method of filtering for skin color. These filtered images where then used for template matching similar to that described by Chan and Lewis [4].

5.3.2 Algorithms and theory The person locator based on motion used two different modes: general detection, and close-person detection. Both modes require the robot to be stopped, and thus are activated by the FSM only in the appropriate states. For general person detection, Alfred searched the entire image for concentrated movement and returned an approximate distance and heading for the movement. Three 320x240 color images were captured in quick succession. For each image, a 3x3 Sobel operator [11] was applied to the pixel values in order to identify edge pixels. Consecutive images were then subtracted to form two difference images that represented edges present in the second of the two images but not in the first [5]. See Figure 5 for an example of the captured and processed images involved in the movement detection system.

Then, three passes were made through each of the difference images to determine whether there existed localized movement and to identify at what distance and heading this movement occurred. In order to improve speed, Alfred calculated table values for each column of the difference images so that repeated calculations were eliminated. Each pass corresponded to an approximate distance away from the robot by running appropriate size boxes over the bottom of the image that looked for different concentrations of movement pixels. The large box corresponded to a distance approximately 4 feet away; the medium size box to a distance approximately 8 feet away; and the small box approximately 13 feet away. Note that a person far away would generally trigger a response for each box filter. To help select the appropriate distance, the large and medium size boxes were broken into horizontal regions such that each horizontal region had to satisfy a certain threshold. Thus, the large box filter would only detect a person if they filled each horizontal region (i.e., the whole box). Finally, if a human was detected in one of the boxes, a heading was calculated based on the column number in the center of the search box. Specifically, we determined experimentally that the lens had a field of view approximately equal to 42.74 degrees. Therefore, the following equation determined the angle from the center of the field of view to the person.

$$\text{heading} = (\text{columns} / 2 - \text{column number}) * (\text{FOV} / \text{columns}) \quad (2)$$

The resulting heading is in the same units as the FOV. For Alfred's physical setup, we used 320 as the number of columns, and 42.74 as the FOV. Figure 6 shows a person detected at each of the three distances and the search boxes

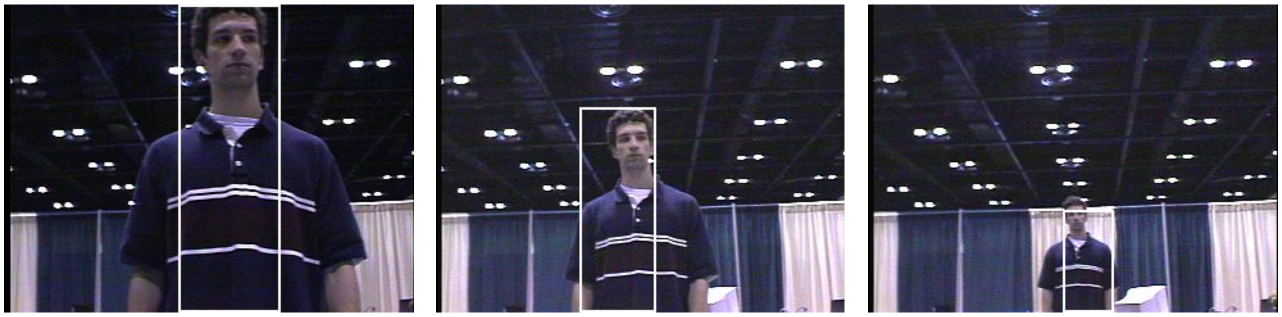

Figure 6 .Distance Finder – Close size search box approximately 1.3m away (left), medium size search box ~2.5m away (middle), and far size search box ~4m away (right).

that were used in the detection.

The second mode--close person detection--was activated after detecting an obstacle which might be a human to be served. The close-person detector captured two successive 240x320 images, performed edge calculations, and created a single difference image all in the same manner as in the first phase. In this phase, however, Alfred searched only the center half of the image to see if enough movement pixels were present to distinguish its forward obstacle as a person and not as a static object. Alfred returned a true value only if the forward obstacle displayed significant motion.

The person detection algorithm based on color worked in two phases: filtering and template matching. The filtering pass used a trained fuzzy histogram specifying the set *Skin* to filter the pixel values into likely and unlikely face locations. The template pass consisted of one or more eye templates scanned over the regions of the image selected in the prefiltering stage.

To create the fuzzy histogram, we took a series of pictures of people in the area where Alfred was to serve, and then edited them to replace all non-skin color areas of the picture with black. The program then went through each picture, using all non-black pixels to generate the fuzzy-histogram. For all non-black pixels the image color the training program normalized them by using equation (3),

$$c_i = \frac{C_i}{\sum_N C_n} \qquad (3)$$

where $C_i \in \{R, G, B\}$ are the three color components found in the original 320x240 color images, and $c_i \in \{r, g, b\}$ are the three normalized colors. The program then used the *r* and *g* values (*b* values were too noisy) to index into a 32x32 histogram and increment the appropriate cell. This same procedure was followed for the entire test set of images. The program then located the final histogram's largest value, and divided each of the cells in the histogram by that value, scaling the histogram values to the range [0, 1]. We can consider the resulting histogram to be a fuzzy membership set for pixels belonging to the fuzzy set *Skin* [14].

Once the *Skin* set is trained, we can use it to filter a new image for skin tones. The computer accomplished this by normalizing the color of each pixel using equation (3) and then indexing into the appropriate cell of the *Skin* histogram to transform the image into skin tone membership values. It then reduced the new image by a factor of four in each axis to speed up skin block detection. Using an appropriately sized block, it located all potential face regions by comparing the average skin membership value against a threshold.

If the average was greater than the threshold, the program considered it a possible face and began the second phase of detection, template matching. The template was created by cropping a test image down to a block the size of a pair of eyes and then shrinking them by a factor of four in each axis so that they would match the shrunk image. By running the template across the top half of the hypothesized head, the program calculated the sum of the square of the differences of the pixel values in the image and template. If this value was less then a preset threshold the area was considered to be a person, and the program returned the horizontal location of the person in the image.

To increase the accuracy of the algorithm, we used two different head sized blocks, and two corresponding eye templates. Using two different sized templates helped us ensure that people at different distances from the camera could be found reliably. Note that in this implementation the algorithm stopped once it found a likely candidate, rather than searching for all possible candidates, in order to reduce computation time. Since Alfred only needed one target to head towards, this decision worked well.

To combine these two independent algorithms we used the following rule: if only one of the two person-detection methods found a person, the robot would follow that result, else if both of the two methods found a person, then the robot would use the face-detection method as it tended to give a more accurate heading. As the two methods are complementary--the face detection will work when a person is standing still, while the motion detection will work if the person's face is not detectable--this combination provided

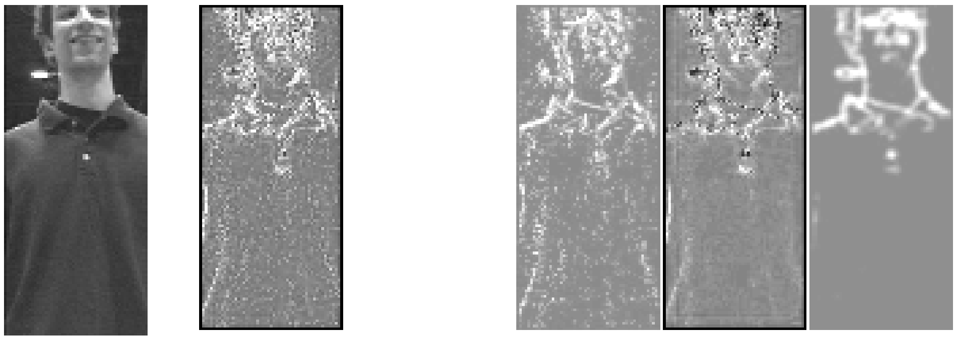

Figure 7 .Person Recognition – Original image (far left), calculated texture image (left), texture band corresponding to edge orientation (middle), texture band corresponding to edge magnitude (right), and texture band corresponding to edge density (far right).

better performance than either method by itself.

5.3.3 Experiments and results During the final round of the competition, Alfred logged images for a 15-20 minutes portion of its time serving in the conference hall. This involved approximately eight interactions. The movement-based person locator logged a total of 15 images, correctly detecting a person at a proper distance and heading 12 times; correctly detecting a person at an improper distance and heading 1 time; incorrectly detecting a person 1 time when no person existed in the image; and not detecting one person when it should have. The result was a success rate of 80%. The close person identifier logged 31 total images, correctly identifying the forward obstacle as a person 22 times and incorrectly identifying it as a person 9 times. Thus, the success rate for the close-human detector was approximately 71%. (Note, the close-person detector only logged an image upon a detecting a person.)

As regards the histogram-based detection, in the Swarthmore Engineering building where the algorithm was initially tested, it performed successfully over 90% of the time. Upon arriving at the robotics competition, however, the system experienced several difficulties. First, the creme-colored walls were similar enough to skin tone to appear in the probability histogram. This problem was compounded by the lights in the convention center which cast shadows on the walls that could fool the eye template at certain distances. Also, the convention center lighting used light bulbs with two different spectrums that alternated in their coverage of the room. The wide variance between the spectrums of the different types of light would throw off the person detection unless the histogram was trained with a large data set. We took over 50 training images in a variety of locations around the convention center in order to provide a robust training set for the fuzzy histogram. When we logged images in the final round of judging, the robot detected four different people using the histogram-based skin detection algorithm. Of these four, three were correct detections, while the fourth was a wall, for a success rate of 75%. The small number of detections was due to the fact that when Alfred was logging images fewer people were at the reception and paying attention to the robots.

5.4 Recognizing people

Color and texture histogram matching was chosen for person recognition, using the standard histogram matching criteria described in [12]. Alfred's recognition system focused on the color and texture of clothing, as the physical placement of the camera allowed Alfred's field of view to see only the torso portion of a person once the person was conversing with the robot. We decided to use both color and texture to increase the size of the search space since we had to deal with arbitrary colors and textures, unlike the work described in [13] where people of interest wore specific, differentiable colors.

5.4.1 Algorithms & theory Alfred attempted recognition whenever he entered the *Serve* behavior of the FSM and subsequently detected a close person. Alfred captured a single 240x320 image and cropped the image to include only the middle third so that extraneous colors corresponding to the environment surrounding the person of interest were not included in the image used for processing. A texture image was created from the RGB image based on three different properties of calculated edges in the color image. The red band of the texture image corresponded to edge orientation, in which orientations from 0 to 180 degrees were assigned values from 0 to 255 accordingly. Similarly, the green band of the texture image corresponded to the amount of contrast, which is characterized by edge magnitudes. Last, the blue band corresponded to coarseness, which is defined by edge density, or the number of surrounding edge pixels in a 5x5 area. Together, they create an image with RGB values that can be manipulated using histogram matching in the same manner as the original color image. Example texture images are shown in Figure 7.

The three-dimensional histograms are compared by adding to a running total for each comparison. Each axis, red, green, and blue are divided into 8 buckets, so that there are 512 buckets in each histogram. Every pixel in the RGB image is put into a bucket corresponding to the amount of red, green and blue it contains. A histogram comparison consists of comparing each of the buckets to the corresponding bucket in the other histogram and adding the lower of the two values to the total. The higher the total value, the closer the match.

The comparison took place by dividing both the original color image and the calculated texture image into three equal horizontal regions to distinguish different areas of the torso. In total, each person is defined by six histograms which are stored in a dynamically-created database to which Alfred adds throughout the time he is serving. Whenever a person is served, Alfred goes through the same process of capturing their image, creating the six histograms, and then sequentially comparing the histograms to all those currently in the database. Alfred returns the best match and a level of certainty as to whether he believes that he has served that person before. Three levels were used: 0 meant no best match was found, 1 meant an unsure match was found, and 2 meant a confident match was found.

5.4.2 Experiments & results A test run for the recognition system conducted before the preliminary round of the competition yielded the following results on 7 subjects, with a total of 20 test pictures taken; Alfred determined the correct best match 13 times; an incorrect best match 2 times; correctly found no good match 1 time; and incorrectly found no best match 4 times. Thus, the success rate for the recognition system was 70%. It should be noted that in this test the subjects were all aware of when the Alfred was capturing test images. This allowed Alfred to take accurate image representations of the subjects, which was not always easy to accomplish in the dynamic environment of the competition's final round.

During the preliminary judging round, the robot correctly identified the one judge who interacted with Alfred twice. Likewise, each of the other judges received a unique name from the robot. Alfred also correctly identified the wall twice when people moved out of the camera's field of view as Alfred took their picture.

In the final round, most people interacting with the robot seemed to avoid the camera. Thus, Alfred took many pictures of the wall and classified them as one of two labels. It did correctly identify two of the robot team members who stood directly in front of the robot. However, it also incorrectly mistook one member of the audience for another because both were wearing similarly colored and textured shirts. Since the robot asks whether it got the identification correct, this actually gave it a chance to be apologetic about the mistake.

6 Future directions

6.1 Navigation and integration

Although the Finite State Machine worked well, in the future a less-rigid model would be better. A subsumption architecture would enable the robot to exhibit a much more dynamic set of behaviors that could respond more fluidly to events (such as being trapped in a group of people). Although this approach will probably require more development time, we believe it will be worth the effort.

6.2 Speech and personality

There are several modifications to the speech and personality system that we want to implement prior to next year's competition. First, we intend to implement some method of noise cancellation perhaps by using an adaptive noise cancellation (ANC) filter [1]. Adaptive filters allow only the desired signal to be processed and are constantly self-updating to account for environmental changes. Two algorithms that can be used to implement adaptive filters are least means squares (LMS), which is robust and easy to implement, and the recursive least squares (RLS), which is faster but its convergence is not reliable. Two microphones working simultaneously are used in ANC; one is unidirectional while the other is omni-directional. The noise input is from the omni-directional microphone and this signal is passed to the adaptive filter. The unidirectional microphone would then be used to record the subject's utterances and an adaptive noise cancellation is performed on it with the adaptive filter. The error signal or noise is thus removed.

A second modification that we intend to implement next year is to enable the speech processing system to adapt to a situation. If there is a lot of background noise, for example, Alfred might listen less and just make one-sided conversation.

Finally we also intend to implement some auditory speaker-recognition features to supplement the visual person recognition output from the vision system. A crude implementation of this would be to have each guest say a particular phrase the first time we meet them, and extract unique features from their speech waveform that would be stored as their voice template. When vision reports later that a person has been recognized we would confirm this by asking the person to repeat the same phrase again, to carry out the template recognition.

6.3 Vision system

The high-level vision processing was relatively accurate, but was often not fast enough to be effective in the rapidly-changing environment experienced in a crowded exhibition hall. A person detected at one time may move to a completely different location by the time that Alfred processes the image information and navigates to the calculated destination. Similarly, the timing involved in capturing images to be used in the recognition system was vital in order to be accurately assessing only those colors and textures associated with the person, and not those associated with the background of the exhibition hall. Therefore, a more useful system would have the ability to track people in real time so that only relevant information is processed and updated dynamically along with the changing behavior of the humans to be served. Overall, the system was very reliable and performed well in smaller, more controlled environments. In order to make the system more robust, a method for continually updating the information and for properly segmenting the image to include only relevant information must be added.

With respect to the blob detection, the color based bin counter method is a fast and reliable method of blob detection in a homogeneous illumination environment. The RGB bound is suitable for bright colors saturated in one of the color bands, but if detection of "mixed" colors is needed, an approach using histogram matching would be more appropriate. The use of self-similar landmarks turned out to be an excellent choice, and future work may want to incorporate the use of bar codes to provide more specific navigational information [10].

Finally, the fuzzy histogram-based method of face detection turned out to be a good choice. Future work in this area will be to combine this prefilter with active tracking techniques and better structural matching techniques than a simple template.

6.4 New capabilities

Based on our competition experience and our experience with Alfred in a variety of situations, there are at least three new capabilities that we believe a truly successful waiter robot needs to possess, and which will be the main focus of our work for the 2000 competition.

The first of these is the ability to track a person that it trying to serve from at least 4-5 meters away. This ability is necessary in order to avoid the situation where the robot heads in a direction, only to find that there is no one there when it arrives. It would also enable the robot to demonstrate dynamic adaptability to its environment.

The second new ability is that the robot needs to be able to adapt to the sensor noise levels in its environment, particularly with regard to speech. As noted above, a robot waiter needs to know both when it can be understood and when it can understand others. Only then can it derive an appropriate interaction for a given situation.

Finally, a robot waiter needs to display more biomimetic behavior--mimicing human reactions physically--than Alfred could. Small characteristics such as eyes that track the person being served, the ability to raise a tray up and down, or the ability to turn its head in response to stimuli would make the robot's interaction more natural and endow it with more perceived intelligence. Some of these capabilities appeared in the early stages at the 1999 competition, but bringing them together into a single, robust robot structure is the challenge of the coming year.

References

[1] S. T. Alexander, *Adaptive Signal Processing: Theory and Applications*, New York: Springer-Verlag, 1986.

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

[3] J. Bryson, "Cross-Paradigm Analysis of Autonomous Agent Architecture", *J. of Experimental and Theoretical Artificial Intelligence*, vol. 12, no. 2, pp 165-190, 2000.

[4] S. C. Y. Chan and P. H. Lewis, "A Pre-filter Enabling Fast Frontal Face Detection", in *Visual Information and Information Systems*, D. Huijsmans and A. Smeulders (ed.), Springer-Verlag, Berlin, pp. 777--784, June 1999.

[5] E. R. Davies. Machine Vision: Theory, Algorithms, and Practicalities. 2nd Ed. Academic Press: London, 1997.

[6] IBM ViaVoice SDK for Linux, http://www.software.ibm.com/is/voicetype/dev_linux.html

[7] B. Maxwell, S. Anderson, D. Gomez-Ibanez, E. Gordon, B. Reese, M. Lafary, T. Thompson, M. Trosen, and A. Tomson, "Using Vision to Guide an Hors d'Oeuvres Serving Robot", *IEEE Workshop on Perception for Mobile Agents*, June 1999.

[8] B. Reeves, C. Nass, *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*, Cambridge Univ. Press, June 1999.

[9] H. A. Rowley, S. Baluja, and T. Kanade, "Neural Network-Based Face Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, January 1998.

[10] D. Scharstein and A. Briggs, "Fast Recognition of Self-Similar Landmarks", IEEE Workshop on Perception for Mobile Agents, June 1999.

[11] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision. 2nd Ed.*, PWS Publishing. Pacific Grove, CA, 1999.

[12] M. J. Swain and D. H. Ballard, "Color Indexing", *Int'l Journal of Computer Vision*, vol. 7, no. 1, pp. 11-32, 1991.

[13] C. Wong, D. Kortenkamp, and M. Speich, "A Mobile Robot That Recognizes People," in *IEEE International Conference on Tools with Artificial intelligence*, April, 1995.

[14] H. Wu, Q. Chen, and M. Yachida, "Face Detection From Color Images Using a Fuzzy Pattern Matching Method", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, June 1999.