Using Vision to Guide an Hors d'Oeuvres Serving Robot

Bruce Maxwell
Swarthmore College
Dept. of Engineering
Swarthmore, PA 19081

Sven Anderson
University of North Dakota
Dept. of Computer Science
Grand Forks, ND 58202

Daniel Gomez-Ibanez
Wesleyan College

Elizabeth Gordon
Harvey Mudd College

Brett Reese
Matt Lafary
Tim Thompson
Mike Trosen
University of North Dakota

Aron Tomson
University of Michigan

Abstract

This paper describes the vision system used on the robot that took first place in both the technical and popular categories of the 1998 American Association of Artificial Intelligence Hors d'Oeuvres Anyone? competition. The robot used visual input to execute multiple tasks: 1) locate people to serve, 2) determine if someone was reaching for food from the robot's tray, 3) determine if the person being served was a VIP or robot competition human participant, and 4) determine where its refill station was located. The robot successfully executed all of these tasks during the competition. All of these tasks were accomplished using a low-cost, low-quality color camera.

1 Introduction

Each year the American Association of Artificial Intelligence holds a robot competition to demonstrate recent developments in robotics. In the fall of 1997 we decided to try to develop a robot system to compete in the Hors d'Oeuvres Anyone? part of the competition. For this competition the robot had to be able to navigate around and interact with people during an actual conference reception. Since part of the competition criteria was interaction with the crowd we determined that visual input for the robot was essential for the robot to be successful.

We had multiple requirements for the physical implementation of the robot vision system. First, it had to be light-weight and low-power given the limitations of the robot. Second, it had to connect to a laptop running Linux. Third, it had to be sufficiently low-cost since we had a limited budget. We decided to use a Connectix Color Quickcam 2 since it met all of the above requirements. This, in turn, forced us to use robust computer vision methods that did not rely on fast (30fps) motion capture, accurate color calibration, or low-noise images.

Given this input device, we wanted to use visual input to execute multiple tasks: 1) locate people to serve, 2) determine if someone was reaching for food from the robot's tray, 3) determine if the person being served was a VIP or robot competition human participant, and 4) determine where its refill station was located. This paper describes each of the algorithms we used for this process and how well they worked during the actual competition. Section 2 describes the robot's hardware, section 3 describes the computer vision algorithms, section 4 describes details of their performance and concludes the paper.

Overall the robot successfully executed all of these tasks. It was largely because of this success that the robot took first place in both the technical and popular categories in the 1998 American Association of Artificial Intelligence Hors d'Oeuvres Anyone? competition.

2 Robot hardware, navigation, and control

2.1 Hardware

Rusty the B.E.A.R. [Barely Educated Autonomous Robot] was a Pioneer 1 mobile robot modified to serve hors d'Oeuvres in a crowded room. Computing power was provided by a Toshiba Tecra 530CDT laptop running

Figure 1 Rusty the BEAR at the AAAI Hors D'Oeuvres Anyone? competition

Linux. The key physical modifications to Rusty included the addition of the following: a bracket to hold the laptop, a tray for serving hors d'Oeuvres, feelers to sense low-lying obstacles such as feet, a Connectix Color Quickcam 2 for visual input, a set of speakers for talking to people, and a microphone for speech recognition.

The camera was mounted at the rear of the serving tray on an aluminum I-beam with an adjustable mount. The Quickcam plugged directly into the laptop through the parallel port and drew power from the external keyboard connector. A picture of Rusty at the competition is shown in Figure 1. Note the Quickcam mounted in between the two speakers on top of the serving tray.

2.2 Navigation and control

The Pioneer 1 robot comes with a Saphira server application that provides a sophisticated and reasonably easy to use interface to the robot. The Saphira server runs on a Unix system and handles all communication with the robot's microcontroller via either a radio modem or, in our case, through the serial port of the laptop.

The Saphira server allows multiple methods of robot control. The most straightforward is direct motion control, where a C program sends move, sense, and act commands to the Saphira server for immediate execution by the robot. The robot can also be controlled through a set of competing reactive behaviors that can avoid obstacles, move to a point, or react to conditions such as the motors stalling or the feeler sensors being tripped. These behaviors can be switched on and off or mixed with direct motion commands to the robot.

We used a combination of direct motion commands and behaviors in order to implement Rusty's serving program. The behaviors would, for the most part, guide the immediate actions of the robot, while a higher level state machine would switch the behaviors on or off and determine short and long-term goals. Specialized routines with direct motion commands guided Rusty in particular situations, such as when Rusty was looking for someone to serve, or when it was looking for the food refill station.

The serving program was a single executable that accomplished three things. First, it set up a block of shared memory to be used to communicate between the vision routines and the robot's high-level state machine. Second, it forked a child process so that the vision and robot control routines could operate independently. Finally, the child process began running the vision interface, while the parent process started up the state machine.

3 Vision system

The computer vision system to provided guidance and feedback for the robot while serving hors d'Oeuvres. While the robot could avoid obstacles using its sonar, it could not necessarily tell the difference between people and other obstacles using sonar alone. As noted above, the vision system consisted of a Connectix Color Quickcam 2, which is readily available for home computers for under $200.

The single Quickcam was used for multiple purposes. The primary task of the vision system was to identify a person to which Rusty could serve food. The primary solution to this problem used visual motion to detect the presence of people.

To verify the results of the motion detector and provide more accurate direction information, we also implemented a face--head and shoulders--finder based on image correlation. We selected a single training image of a person and cut out the person's head and shoulders to form a template. During operation, the vision system uses correlation to match the head and shoulders template to people in the new pictures. By using two modalities, the vision system is more robust than using a single method by itself.

As a second task for the camera, we wanted to know if someone was taking food off of Rusty's tray so that Rusty could stop and talk to the person. To enable this ability, we positioned the camera on the back side of the tray. Therefore, if someone took something from the tray, there should be some clue--like a hand with a lot of skin--we could easily detect with the camera.

As part of the competition, robots would score points if they could identify and interact with conference VIPs who were all wearing bright green strips on their name tags. As a third task, then we developed a badge detector that looked for bright green and bright pink badges--the bright pink badges were worn by human robot competition participants.

Finally, when the robot decided it need to be refilled it had to find the refill station. Since the dead reckoning of the robot could not reliably bring it back to the refill station, we created a large green and yellow sign--the colors of the University of North Dakota--that the robot could spot from far away. The final part of the vision system was a mechanism for the robot to uniquely recognize the sign and determine where to go.

3.1 Finding people to serve

To find people to serve, Rusty used two different modalities to find, locate, and verify the existence of a person. The primary method for finding a locating people was based on motion. We also implemented a template-based person detector which provided verification and more accurate localization in some cases.

3.1.1 Finding people using motion

The task of the people finder is to not only detect people, but to localize their position relative to the robot so that the robot has a goal--a person to serve--that it can attend to. The face detector of Rowley, Baluja, Kanade [4], solved a similar problem, that of finding upright, frontal faces in a scene. However, our human detector could depend on the presence of frontal faces, because the Quickcam's field of view would often crop out the face of a person close to the robot. Furthermore, people the robot should try to serve might not be facing the camera. To solve this problem, the main people-finding algorithm extracted motion information from a series of pictures. If there was an area of motion with the right shape and size, a human was hypothesized.

In order to take pictures quickly, we developed a customized frame grabbing program that took three to five pictures in a series as quickly as possible given the communication bandwidth of the Quickcam. In the competition, we used only three images in order to speed up the person detection process. The pictures were 120 by 160 pixels with 24 bit color. A complete description of the algorithm follows.

To process the images, the algorithm uses a 3x3 Sobel operator to calculate the gradient at each pixel of the incoming images [2]. The algorithm then thresholds the gradients to find edges. We do not try to thin the edges or connect them into boundaries. We then subtract consecutive edge-filtered images to obtain two edge-differenced images. If a pixel changes color from one edge-filtered picture to the next, it is on in the edge-differenced picture.

These resulting pictures should contain the edges of the moving objects in the image sequence. The pair of pictures error-check each other; if a person is especially slow-moving, he or she may not appear in one of the two edge-differenced pictures. But he or she usually shows up in the other one. Using the edge-difference images was inspired by Nair and Aggarwal's motion detection system [3].

Next, the algorithm counts the *on* pixels in a series of rectangles scanned over the image. The search rectangles correspond to the expected shape of people. The system uses three sizes of rectangle, corresponding to three distance ranges. Close people appear larger than far-away people. Each rectangle is placed at every possible x-coordinate along the horizon, and the number of *on* pixels in the region is counted.

If none of the search rectangles contain a more than a set threshold of *on* pixels, the algorithm hypothesize that there is no human in view. If one of the rectangles exceeds the threshold, it hypothesizes that there is at least one person in view.

To prevent a far away person from causing a close human detection, the two larger search rectangles are subdivided into two and three horizontal bands, respectively. There must be movement in all the component bands in order for there to be a person detected at the two closer distances. That way, movement close to the horizon will trigger a far human detection, but since there is no movement in the upper part of the image, the algorithm does not detect a close human. This arrangement also prevents noise from bright lights from setting off the close human detector. Since the Quickcam has a limited sensing range, a bright light may appear as motion in the top of the image. If there is nothing below it, however, the human detector is able to avoid returning a false positive.

The direction to the person is calculated by selecting the center of the search rectangle which contained the highest number of *on* pixels. This x coordinate is converted to degrees to the right of center.

Figure 2 shows how the human finder would work on three example images of a person moving very slowly away from the camera.

The distance to the person is estimated by the size of the rectangle with the most *on* pixels. Since there are three possible size rectangle, there are three possible distance estimates. Based on some ground truth measurement, we estimated that a person was 2 meters away or less when detected in the largest box, 4 meters away in the medium sized box, and at least 6 meters away in the smallest box. This number in meters was passed to the robot's main routine. The source code for this algorithm is available at http://www.cs.und.edu/REU/projects/vision/ibanez.

Using an edge filter provided a couple of advantages over simple time-differencing. It filtered out noise from bright light sources, which sometimes created a halo around people. It also filtered out most shadows, which were usually diffuse enough not to have edges.

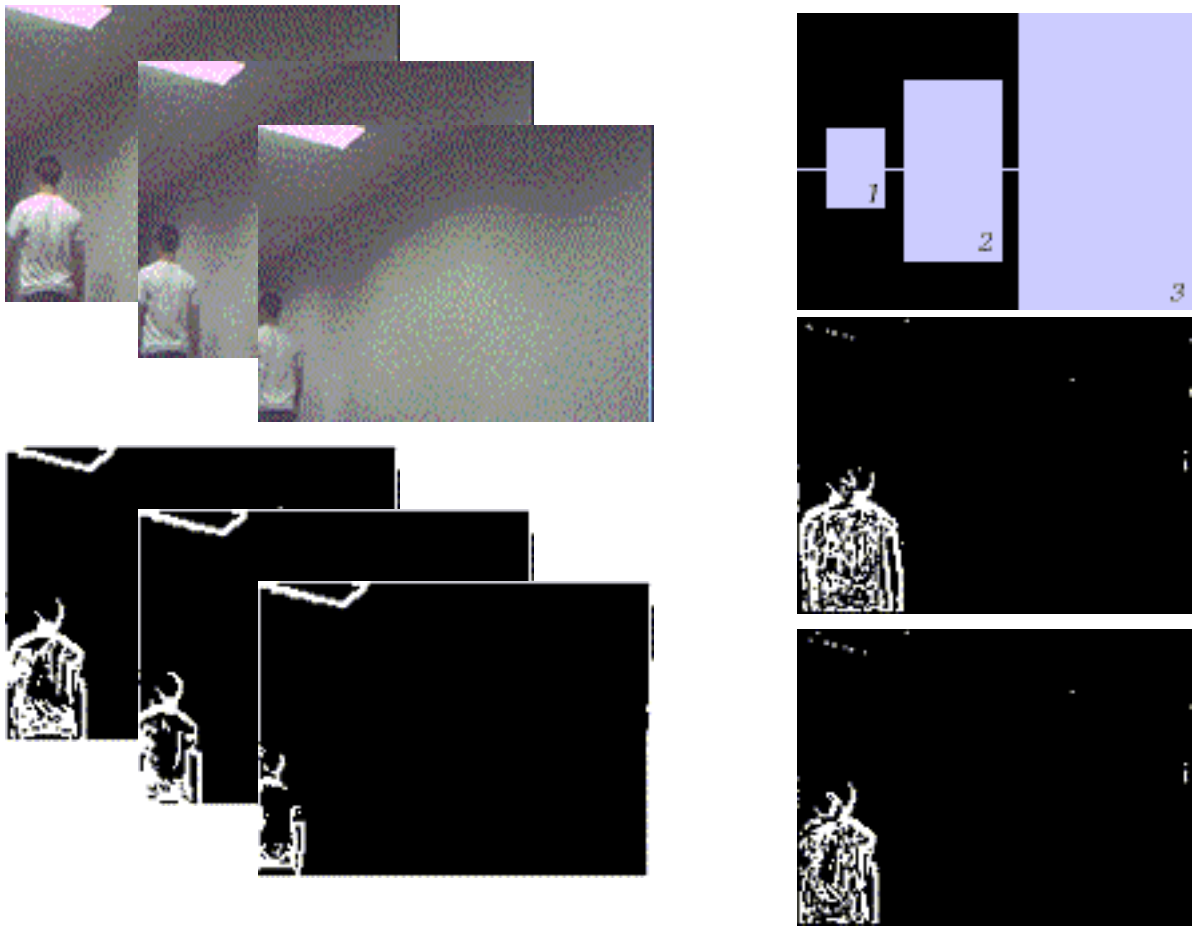

Figure 2 Original images (top left), edge images (bottom left), edge difference images (bottom right), box filters for counting pixels that moved (top right).

When there was more than one person in view, it did not cause problems for the algorithm, because the region selector would select the one region with the most motion. This did not result in human detections between two people unless the two people were close enough to be in the same rectangular search region, in which case there was not much difference in their direction anyway.

The distance estimate was sometimes wrong by a meter or two, but that also did not cause problems. If the robot ended up short of the person it would check its camera again and continue another two meters. If it over-estimated the distance, the robot would sense the person in its sonar, stop, and check to see whether the obstacle was a person.

3.1.2 Finding people using templates

The main algorithm for this aspect of the person finder is a sum-squared-differences correlation algorithm. The program uses a previously created template stored in the local robot directory. That image is matched against a new image at every position where the entire small image fits inside the new image. At each position, the algorithm computes the difference between each pair of corresponding pixels. Those differences are squared and then summed to provide an estimate of the match; the position with the smallest result is the best match.

We tried several versions of the algorithm using both color and greyscale images; the greyscale images had better results and were faster. We also tried using transparent backgrounds where the white pixels in the template image were ignored, but this did not significantly improve the results. We did include a shift term in the SSD algorithm to equalize intensities between the template and the image: the difference is shifted by an amount equal to the difference between the average intensities of the template and the new image.

The effectiveness of the algorithm is highly dependent on the selection of an appropriate template. While testing,

individual faces were found to be reasonably effective. Testing on a data set collected at the North Dakota Museum of Art, however, showed that a template that included a person's head and shoulders worked better than a face alone. Using a data set of twenty pictures of people, various templates were found to yield accuracies of 50% to 80%. Unfortunately, the range of minimum values returned by the algorithm in the presence of people turned out to be quite similar to the range of minimums returned when using pictures with no people in them. To try to better estimate when there were people, we implemented a double threshold system. The function returns one of three values. One value indicates that there is a person in the image, another that there is no person in the image, and the third that the algorithm is not sure.

Given a typical data set of twenty images, and a template yielding 16 correct or close results, the algorithm would be sure there was a person in approximately four of those 16 cases. This is a highly reduced percentage, but for pictures where it is known there is a person the heading returned is generally very close to the center of the person. When combined with the motion based human detector, this algorithm helped to improve the accuracy of the robot's heading.

3.1.3 Combining the results to locate people

As noted above, the primary person locator was the algorithm based on motion. When the motion-based detector found a person then the person finder would return TRUE to the main program. Likewise, when it did not find a person, it would return FALSE. If the template-based detector did not return a SURE reading, then the algorithm returned the angle provided by the motion-based detector. If, however, the motion-based detector found a person and the template-based detector returned a SURE reading, then the algorithm returned the direction provided by the template-based detector. Overall, the pair of algorithms provided more accurate localization than the motion-based detector by itself.

3.2 Detecting a hand in the cookie jar

As part of the serving task, the robot needed to know if somebody was reaching into the tray to take food. To accomplish this, the camera was positioned behind the tray. We assumed that if a person reached into the tray, their hand would be a large part of the image. Therefore, the hand finder was designed to look for a large number of skin-colored pixels in the image.

The hand finder needed to recognize hands of all skin colors while excluding other objects. It also needed to run while the robot was moving so it had to be fast. Finally, it needed to be incorporated into the main robot code so it was written as a function that could be called from the main program. In addition, it needed to work accurately on poor quality pictures given the capabilities of the Color Quickcam.

We based the color finding mainly on research done by Margaret Fleck and David Forsyth [1]. They wrote a naked people finder which used a skin filter. As it turns out, human skin colors lie in a range from red to yellow (or brown). The Fleck-Forsyth skin filter finds all skin colors by looking in a certain range of hues and saturations and some pre-processing. They used a log transformation to try to subtract the effects of illumination, as well as some smoothing of the image.

Previous students at UND had implemented a simplified version of the Fleck-Forsyth filter which converted a picture from RGB to Rg, I, and By where Rg, I and By were defined as in (1) to (4)

$$L(x) = 105 * \log(x + 1 + n) \quad (1)$$

$$I = L(G) \quad (2)$$

$$Rg = L(R) - L(G) \quad (3)$$

$$By = L(B) - (L(G) + L(R))/ 2 \quad (4)$$

The log is a base ten logarithm, and n is a random noise value. For a full explanation of this log transformation, see the details of the Fleck-Forsyth skin filter. When looking at the results of this filter using images from the Quickcam, it was apparent that the values falling into the hue and saturation ranges used by Fleck and Forsyth were usually not skin pixels. However, the skin pixels were consistently transformed into a bright blue color when viewing the transformed images. Therefore, the hand finder does the log transformation and then marks pixels that fall into a certain range of blue.

Unfortunately, testing of this method showed that the particular shade of blue is dependent upon the particular lighting conditions. (Even though the log transformation is supposed to get rid of the effects of illumination.) So, it is necessary to re-calibrate the hand finder for each new location. Up until the competition we were able to recalibrate it by hand. However, at the competition we tried using an automatic calibration routine used for the badge-finding algorithm described below. The automatic calibration worked better, and ended up being faster even though it was close to a brute force search.

The main hand finder function takes a picture and then calls several processing functions to determine whether there is a hand present. The first process is simply to adjust for the black level of the camera. The algorithm determines the minimum value among all three color planes and then subtracts that value from each pixel. The next process is the log transformation described above. The next step is to find skin pixels and mark them. The last

process is a despeckling algorithm, designed to eliminate misidentified skin pixels. In the despeckling algorithm a pixel is kept as skin if it is mostly surrounded by skin pixels in its 8-connected neighborhood. The exact number of surrounding pixels required can be adjusted, but is normally not lower than 6 or possibly 5. Finally, the number of marked pixels in the image is counted, and if it is greater than a threshold the image is considered to contain a hand.

Because the hand finder needed to run in real-time, we optimized it as much as possible. First, the picture needed to be captured as quickly as possible. The hand finder runs under Linux, using a driver for the Quickcam written by Patrick Reynolds which is available on the World Wide Web. The hand finder calls the executable, cqcam, using popen to avoid file IO. The camera program has an auto-adjust feature, which takes several pictures and adjusts the colors. In the interests of speed, this feature was disabled. Instead, we used command line arguments to cqcam to set the camera's black level (approximately 70) and the brightness of the room. We had to determine the brightness hard code it for each new location. We also used the smallest picture the Quickcam could capture. Further optimizations consisted of the combination or elimination of loops wherever possible, and also the utilization of a piece-wise linear approximation to the function 105 * log(x). The three linear regions were $x \in [0,30]$, $x \in [30,80]$, and $x \in [80,255]$, and values were interpolated between the points on the log function at the ends of each region.

The handfinder worked quite well at the North Dakota Museum of Art tests, but did not fare as well in the competition. Unfortunately, the walls, ceiling, and floor of the reception hall were all approximately skin-colored. Therefore, we had to significantly tighten the parameters of the handfinder in order to avoid getting a large number of false positives. This in turn caused it to rarely find a hand in the image during the actual competition.

3.3 Badge detector

The requirement for the badge detector was to determine if a badge was present in the field of view when the robot was stopped in order to serve a person. Since the robot was assumed to be serving a person, we only needed to detect whether the badge existed in the field of view, not where it was or how far away it might be.

The badges worn by the robot competition human participants and the conference VIPs were unique colors--bright green and bright pink--so we implemented a simple color-based detector that looked for specific colors in the image. We implemented a simple line-search procedure in order to train the badge detector from a set of training images.

A line search procedure works as follows. Given a set of initial parameters and a fitness function, the line search starts by varying one parameter while holding the remainder constant. After finding the best value for the varied parameter, it fixes that parameter and selects another parameter to vary. This process is continued through one iteration of all the parameters. The line search then continues a second iteration through all of the parameters and so on. The termination condition occurs when none of the values change through one entire iteration of all of the parameters.

The parameters we trained were an upper and lower bound on each color band and a count threshold indicating how many pixels had to be the desired color in order to recognize a badge. We created a training set of images at the competition site by taking 20 images of people with the camera looking in various directions. In half of the images a person was wearing a VIP badge, and in half no one was wearing a badge. We took a similar training set for the robotic competition participant badges.

The fitness function for the line search was simply the percent of images correctly classified as badge/no badge. After three iterations through the parameters--about 1-2 minutes of processing time--the line search returned a set of parameters that correctly classified all of the images in the training set.

3.4 Refill station locator

The final task for the vision system was to find the refill sign so that it could locate the refill station from a distance. The refill sign was a 3’ x 4’ green and yellow poster with the text “UND Refill Station” across it.

Since green and yellow are not particularly rare colors--unlike the fluorescent pink and green of the badges--we needed a more robust detector than simple color. Since we could not guarantee the orientation of the sign, however, something like a hard template was not acceptable either due to problems caused by oblique projections of the sign during template matching.

To solve this problem we used histogram matching as originally defined by Swain and Ballard for object recognition tasks [5]. Histogram matching provides an identification method that is invariant to rotation and shape, and can be made invariant to size by using variable-sized windows. Histogram matching is based on the histogram intersection calculation, which is defined as in (5) where a and b are similarly sized histograms of part of all of an image.

$$\sum_{\text{bins j}} min(a(j), b(j)) \qquad (5)$$

Histogram intersection returns a high value when there is a high degree of overlap between the histograms, and a low value when there is little or no overlap.

Our template histogram was created by taking a picture of the refill sign and extracting a section of it of a specific size. To detect the refill sign, we used a window of similar size and panned it over the input image. At each location, we calculated the histogram of the pixels in the window and executed a histogram intersection with the template. When the intersection calculation exceeded a manually set threshold--set as a percentage of the total possible score--then the sign was detected.

4 Testing, performance, and conclusions

All of the vision routines except the badge detection--where we did not know the colors of the badges until two days before the competition--were tested prior to the competition. We tested the robot twice at receptions at the North Dakota Museum of Art [NDMA]. As a result of those tests, we were able to significantly refine and improve the people detection aspect of the vision system.

At both of the North Dakota Museum of Art trials, all images were logged to disk, off-loaded to the web, and passed to the Human Detector.

In brief, the results were: 27 true positives (person found in the right place), 19 true negatives (nobody in view), 6 false negatives (person in view not found), and 7 false positives (human not present but detected). This gave a total of 46 correct answers and 13 incorrect answers for an accuracy of 77%.

In this trial, the last trial that was analyzed completely, the false negatives were acceptable and desirable, because they were always caused by a person mostly out of the field of view, or too far away. When the robot turned to examine a different field of view, either the same person was found the second time, or a closer person was found. Because of this, the percentage of desirable responses was actually closer to 88 percent. The false positives were always caused by movement of the camera, which was subsequently improved by a refinement of the robot's movement program.

As noted above, during the competition the robot did successfully respond to all of the desired visual inputs except the hand finder. While the hand finder worked well at the NDMA tests prior to the competition, the colors of the competition environment precluded its ability to successfully avoid false positive identifications of skin color in the image.

Once during a preliminary round, and twice during the competition the robot was able to find its way back to the refill station using the sign as a guide. Twice during the competition the robot identified a pink badge and inquired whether the person was a human robot competition participant. Largely because of this multitude of skills, the robot demonstrated interaction with its environment and was able to win the competition.

Given the hardware limitations that Rusty labored under and the low quality of the camera, the success of the system is a testament to the robustness of the methods employed. Furthermore, all of the routines ran in real-time on a laptop computer along with the robot control software.

What this project demonstrated is that vision can be used to reliably make decisions in a complex environment, so long as the decisions can be formulated in ways that allow the use of robust methods. The other lesson learned is that using training images and optimization methods is a significant improvement over manually setting parameters.

5 Acknowledgments

Thanks to Digi-Key, NSF REU program, and UND Computer Science Department. This project would not have been possible without help from these people.

References

[1] M. Fleck, D. Forsyth, and C. Bregler, “Finding Naked People”, in Proceedings of *European Conference on Computer Vision*, 1996.

[2] R. Jain, R. Kasturi, and B. Schunk, *Machine Vision*, McGraw-Hill, New York, 1995.

[3] D. Nair and J. Aggarwal, “Moving obstacle detection from a navigating robot”, *IEEE Transactions on Robotics and Automation*, 14:3, 404-416, 1998.

[4] H. Rowley, S. Baluja, and T. Kanade, “Neural Network Based Face Detection,” in Proceedings of *Computer Vision and Pattern Recognition*, 203-208, 1996.

[5] M. Swain and D. Ballard, “Color Indexing”, *Int'l Journal of Computer Vision*, 7:1, 11-32, 1991.