Fast Hand Gesture Recognition for Real-Time Teleconferencing Applications

James MacLean†, Rainer Herpers‡, Caroline Pantofaru*,
Laura Wood*, Konstantinos Derpanis+, Doug Topalovic+, John Tsotsos+

†Dept. of Electrical & Computer Engineering,
University of Toronto,
Toronto, Canada M5S 3H5

‡Dept. of Applied Computer Science,
University of Applied Sciences, FH Bonn-Rhein-Sieg,
53757 Sankt Augustin, Germany

+ Dept. of Computer Science
York University,
Toronto, Canada M3J 1P3

* Dept. of Computer Science
University of Toronto
Toronto, Canada M5S 3G4

Abstract

Work on real-time hand-gesture recognition for SAVI (Stereo Active Vision Interface) is presented. Based on the detection of frontal faces, image regions near the face are searched for the existence of skin-tone blobs. Each blob is evaluated to determine if it is a hand held in a standard pose. A verification algorithm based on the responses of elongated oriented filters is used to decide whether a hand is present or not.

Once a hand is detected, gestures are given by varying the number of fingers visible. The hand is segmented using an algorithm which detects connected skin-tone blobs in the region of interest, and a medial axis transform (skeletonization) is applied. Analysis of the resulting skeleton allows detection of the number of fingers visible, thus determining the gesture. The skeletonization is sensitive to strong shadows which may alter the detected morphology of the hand. Experimental results are given indicating good performance of the algorithm.

1 Introduction

In recent years the field of computer vision has devoted considerable research effort to the detection and recognition of faces and hand gestures. Being able to recognize faces and hand gestures has tremendous potential in applications such as teleconferencing, telemedicine, and advanced interfaces for human-computer interaction. In particular, hand gestures are an appealing way to interact with such systems as they are already a natural part of how we communicate, and they don't require the user to hold or physically manipulate special hardware.

Most current teleconferencing systems just carry video and audio signals, with video being captured from static cameras with a wide angle of view. Any technique of interest for hand-gesture recognition is necessarily real-time, and preferably should be implemented without excessive hardware.

Kjeldsen & Kender [8] devised a technique for doing skin-tone segmentation in HSV space, based on the premise that skin tone in images occupies a connected volume in HSV space. They further developed a system which used a back-propagation neural network to recognize gestures from the segmented hand images [7, 9]. Freeman & Weissman [3] use hand movements to adjust "control bars" by detecting rising hand movements (as a trigger to start the system) and then relative hand movements to manipulate the controls. Freeman et. al. [2] also report on the development of a system where hand gestures are substituted for mouse interaction. The hand is imaged against an uncluttered background, and the hand's movements in 2-D are interpreted the same as mouse movements. The system is demonstrated using a flight-simulator game, where hand movements control the joystick. Utsumi & Ohta [14] used a 5-camera setup to view a hand against a uniform background, and segment it using Sobel edge detection. Their system was capable of recognising 7 different gestures (based on the number of fingers being held up). Hamdam et. al. [4] studied the dynamics of hand gestures, and used hidden Markov models (HMMs) to model the transitions between gestures. Recognition involves eigenimages derived from a training set, but the system is sensitive to the scale and orientation of the hand, and like previous approaches requires an uncluttered background. Finally, Jennings [6] demonstrates robust gesture detection results against cluttered backgrounds in a system which uses 4 cameras (3 grey-scale, 1 colour). However, the method is very compute-intensive.

In this paper we present a simple, real-time method

133

for gesture recognition within the context of SAVI. The method is in two steps. The first step detects the presence of a hand in a standard configuration (palm forwardand upright, all fingers outstretched). It utilizes skin-tone blob finding to identify regions of interest, application of direction-sensitive filters oriented vertically and at 45° to detect fingers and thumb (respectively). Once a hand is detected, the second step processes the skin tone blob on the assumption that is is a hand with 1 to 4 fingers raised. The region is binarized, morphologically pre-processed to smooth the hand's profile, and skeletonized to determine the hand's configuration. The identified gesture is then passed to the rest of the SAVI system.

2 SAVI System Description

SAVI (Stereo Active Vision Interface) [5] is a system currently being developed in the authors' labs. Targetted at teleconferencing and telemedicine applications, it makes use of a Perception-Action-Cycle (PAC) paradigm to process sensory data and respond to its changing environment. SAVI employs TRISH-2 (Fig. 6, a descendant of TRISH-1 [11]), a robotically controlled binocular head. The head has a pair of 24-bit colour CCD cameras with 4 optical degrees of freedom (zoom, focus, iris, exposure/shutter for each camera). There are also 4 mechanical degrees of freedom (common tilt and rotation, as well as independent pan). The head motion controller is inherited from PLAY-BOT [13], a previous project in which the head was used. Both cameras can be controlled independently or in master/slave mode. In the latter case one camera (termed the 'attentive camera') attends, tracks or zooms in on an object of interest while the other camera may provide an overview of the visual scene and/or stereo information.

SAVI provides active control via its *supervisor* [5], a control module with a finite set of states and transitions which include a skin-tone search state, a tracking state, and face evaluation state. These allow SAVI to detect and track faces, and their implementation is described in [5]. A brief description of each is included here for completeness.

> **Skin Tone Search** During *Skin Tone Search*, the visual input of both cameras is evaluated to find the most salient skin blob. Both cameras are zoomed out, and the size, shape and other features of connected skin-tone blobs are evaluated. Regions *a priori* known as uninteresting are excluded via an inhibition mechanism. Both cameras are focussed on the most salient blob.

> **Tracking** During *tracking* the most salient skin-tone blob is kept fixated and actively followed with both cameras. When the blob stops moving or only a small amount of motion is present, SAVI evaluates the blob as a possible face.

> **Face Evaluation** During *face evaluation*, the blob is searched for facial features. If verified as a face, the system will continue to track the face and search for an associated hand gesture. If the blob is not a face the blob is inhibited and *skin tone search* is called to determine the next most salient skin tone region.

Also included in the supervisor are states for hand detection and gesture interpretation, which are the subject of this paper. Once a face blob has been verified, the supervisor enters a hand detection state in which skin tone blobs near the right side of the face are evaluated in order to determine if a hand is present. If a hand is found, the supervisor also tracks the hand, and repeatedly evaluates the hand blob to detect known gestures.

3 Hand Detection

SAVI uses hand gestures to give simple commands to the system. Each hand gesture involves showing a different number of fingers to the camera. For example, holding up one finger instructs SAVI's cameras to zoom in on the hand & face being attended. Holding up two fingers is the command to zoom out again. Holding up all five fingers is used initially to detect the hand.

The process of hand detection starts when the supervisor detects skin-tone blobs to the left of the currently attended face (all gestures are made with the right hand). It is expected that the hand will be in a standard pose, specifically the hand is upright, fingers outstretched, and the palm is facing the camera. For each blob the supervisor defines a *region of interest* (ROI), and segments the image into a single connected skin-tone region with black elsewhere. The red-channel image of the blob is then filtered using an elongated and vertical oriented G2 filter [1] and one oriented at 45°. The filters are tuned to respond to spatial objects about the right width for a finger. Scale assumptions are derived from the size of the detected face. It is expected that the three central fingers will produce strong responses from the vertical filter, and that the thumb will produce a strong response from the 45° filter.

Let I_F represent the image filtered with the vertical (finger) filter, and I_T represent the image filtered with the 45° (thumb) filter. Each filtered image is thresholded to remove responses less than its average intensity value. Next, I_F is searched to find the 3 largest responses. The search algorithm finds a maximum, reports its location, and then "removes" it by deleting all pixels from around the maximum until a bounding valley is found (thus removing the "hill" corresponding to the maximum). When the search is repeated it will find the next largest peak, *etc.* In a similar fashion, the location of the maximum response in the right $1/3$ of I_T is determined. Designate the locations of the three peak responses from I_F as $\vec{f}_1$, $\vec{f}_2$ and $\vec{f}_3$, where

$\vec{f}_1 = (\begin{array}{cc} f_{1x} & f_{1y} \end{array})^T$, *etc.* These location vetors are ordered so that $f_{3x} \geq f_{2x} \geq f_{1x}$). Likewise designate the locations of the two maximum responses in the right $1/3$ of I_T as $\vec{t}_1$ and $\vec{t}_2$. (Two peaks are used by the thumb scoring function to ensure that a dominant peak is present.) The locations of the I_F maxima are then checked to see if they are consistent with the following criteria:

c_1 is assigned the maximum k_{1j} for which $|(f_{3x} - f_{2x}) - (f_{2x} - f_{1x})| < T_{1j}(w)$ holds, 0 otherwise. The threshold T_{1j} is a function of the width w of the ROI. This constraint is designed to favour evenness of finger[1] spacing.

c_2 is assigned the maximum k_{2j} for which $\max(f_{1y}, f_{2y}, f_{3y}) - \min(f_{1y}, f_{2y}, f_{3y}) < T_2(h)$ holds, 0 otherwise. The threshold T_{2j} is a function of the height, h, of the ROI. This constraint penalizes large vertical disparities in the maximum responses in fingers, and embodies the concept that fingers should all be at roughly the same height.

c_3 is assigned the maximum value of k_{3j} for which $f_{3x} - f_{1x} < T_3(w)$ holds, 0 otherwise. This constraint penalizes for fingers which are spread too widely.

c_4 is assigned the maximum value of k_{4j} for which $|(f_{1y} + f_{2y} + f_{3y})/3 - h/4| < T_4(h)$ holds, 0 otherwise. This constraint attempts to favour a particular vertical location for the maximum response for the fingers. It is scaled to the height of the ROI, and as such is an expectation of the proportions of a typical hand.

c_5 is assigned the maximum value of k_{5j} for which $|(f_{1x} + f_{2x} + f_{3x})/3 - w/2| < T_4(w)$ holds, 0 otherwise. Like c_4 this constraint attempts to favour a particular horizontal location for the maximum responses for the finger. It is scaled to the width of the ROI.

The use of the quantization parameter (usually $j \in \{1 \ldots 4\}$) allows each constraint to take on higher confidence values for data which better meets the constraint. In the preceding definitions of constraints, the constants k_{ij} are empirically determined and satisfy the following: $0 \leq \max_j k_{ij} \leq 1$ for each i and $\sum_{i=1}^{5} \max_j k_{ij} = 1$. The k_{ij} for the constraints which measure absolute location (c_4 & c_5) are smaller than those for constraints measuring relative parameters to give more weight to the latter. The evenness constraint is most important, since equally spaced vertical responses are considered less likely to occur in non-hands. Once the value of each constraint has

---

[1] It is not clear yet that we are dealing with fingers. For the sake of brevity while describing constraints we will use "fingers" where we mean "finger candidates."

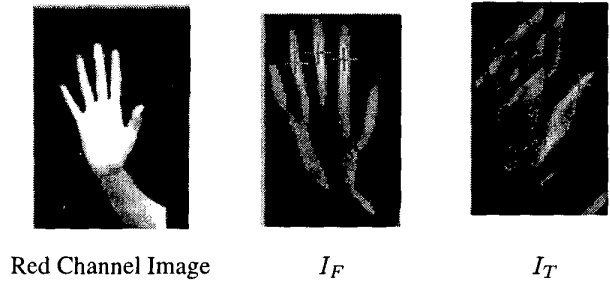Red Channel Image I_F I_T

Figure 1: In this figure the successful detection of a hand is shown. The green crosses show the recovered peak responses in the thresholded filter responses.

been computed, a scoring function is computed as follows: $C = \sum_{i=1}^{5} c_i$. This score can be thought of as a measure of confidence that the three middle fingers have in fact been found. If this score exceeds a pre-set threshold, then the algorithm proceeds to an evaluation of the thumb.

The thumb scoring function examines the relative values of the peak filter responses, $p_1 = I_T(\vec{t}_1)$ and $p_2 = I_T(\vec{t}_2)$. The scoring function is binary, and has value 1 if $p_1, p_2 > 0$ and $p_1 - p_2 > T_t$, or if $p_1 > 0$ and no second maximum is found. The threshold T_t is empirically determined. If these two conditions are not met, then the scoring function has value 0.

Figure 1 shows the successful detection of a hand. The filtered finger image I_F shows three strong maxima which meet the scoring criteria for fingers. The filtered thumb image I_T has a dominant peak identifying the location of the thumb, and the recognition gesture is accepted. Figure 2 shows a rejected hand candidate. The hand is seen to be in a position other than required—the thumb is not visible. The three-middle fingers are detected as expected, but no dominant peak in the I_T image is found, and the thumb-scoring function fails.

This algorithm has been implemented, and obtained results show it to be real-time and robust. It is adversely affected by rings which provide sufficient contrast with the skin-tone, as a connected skin-tone blob is hard to achieve in this case. It can also be fooled by a left hand with the little finger oriented at 45°, thus appearing like a thumb. It is important that the ROI fits tightly around the hand as the constraint thresholds are based on the dimensions of the ROI.

Note that once the skin-tone blob has been detected (and a bounding box determined) the subsequent processing is independent of the dimensions of the hand (assuming, of course, that sufficient resolution exists to distinguish the individual fingers). The model knowledge is described in

135

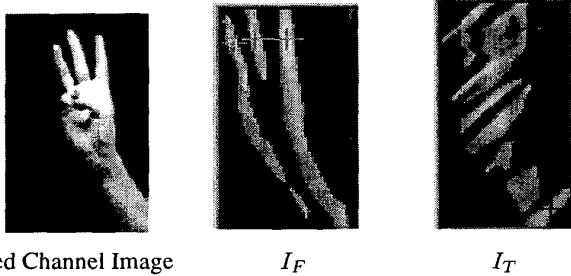Red Channel Image I_F I_T

Figure 2: In this figure the rejection of a hand is shown. The three fingers pass the finger detection criteria, but the evaluation fails when no thumb is recovered (no single strong maximum exists).

terms of proportions, not absolute sizes, allowing the system to detect hands and gestures over a wide range of distances.

4 Gesture Recognition

Once a skin tone blob has been detected as a hand, it is tracked by SAVI, and the region-of-interest (ROI) associated with the blob is identified for use by the *Gesture Recognition* state of the supervisor. One can think of the *Hand Detection* as a sort of trigger gesture; it is computed once, and need not be done in real-time. It has less assumptions, but needs to be very robust. For subsequent gesture recognition we use a faster methodology, which relies on more assumptions. The gesture recognition state makes the following assumptions:

- the ROI contains a hand,

- the hand gesture is roughly vertical,

- the hand gesture has 1 to 4 fingers raised.

The image in the ROI is binarized so that skin-tone pixels are white and non-skin-tone pixels are black. The image may be downsampled to increase speed if desired. The subsequent image processing involves three steps:

1. Morphological pre-processing.

2. Skeletonize the image.

3. Clean the skeleton.

4. Identify the gesture using basic model knowledge.

4.1 Morphological Pre-Processing

Before skeletonization is performed, a pre-processing step is performed to smooth the contours of the hand. This step consists of applying an opening filter followed by a closing filter [12, 10]. Opening filters consist of an erosion operation followed by a dilation operation, and closing filters are the reverse of this. The net effect of this processing is to smooth the contours of the skin-tone region. SAVI implements a simplified erosion operator, in which all white pixels with less than 4 white neighbours in a 3×3 neighbourhood are set to black. Dilation considers black pixels, and sets them to white if 4 or more of its neighbours are white [10]. Figure 3 shows the effect of this pre-processing step. Its application reduces the number of spurious skeleton branches caused by 'bumps' on the contour of the skin-tone region. This in turn simplifies the task of identifying individual fingers. Any remaining spurious branches are removed in a later processing step (see Section 4.3). Also, a "hole filling" algorithm is applied to remove holes from the segmented region, thus preventing unnecessary cycles in the skeleton.

4.2 Skeletonize the Image

The skeletonization is performed using a technique devised by Zhang & Suen [15]. The 2-step algorithm iteratively marks border points for deletion, and terminates when a complete iteration of the algorithm leaves the image unchanged. The first step marks east and south boundary points, as well as north-west corner points for deletion, and the second step marks north and west boundary points, as well as south-east boundary points for deletion. The algorithm guarantees that skeleton segments will not be disconnected and skeleton end-points are not deleted (thus unduly shortening the skeleton). Also, undesired erosion into the region at points other than boundary points will not happen. The algorithm is quick, but certainly doesn't produce the same quality of skeleton as techniques which involve solving the heat equation. Spurious skeleton segments are observed, but it turns out that these are easily removed.

4.3 Clean the Skeleton

The spurious skeleton segments are observed to be short in comparison to segments representing the fingers and wrist. In order to describe the algorithm to remove them, first consider the following definitions:

End Point An end point is one with only one 8-neighbour, and represents the terminal pixel of a skeleton segment.

Feature Point This is a point on the skeleton with more than two 8-neighbours. This represents a point on the skeleton where two or more segments meet. Note that skeleton points which are neither end points
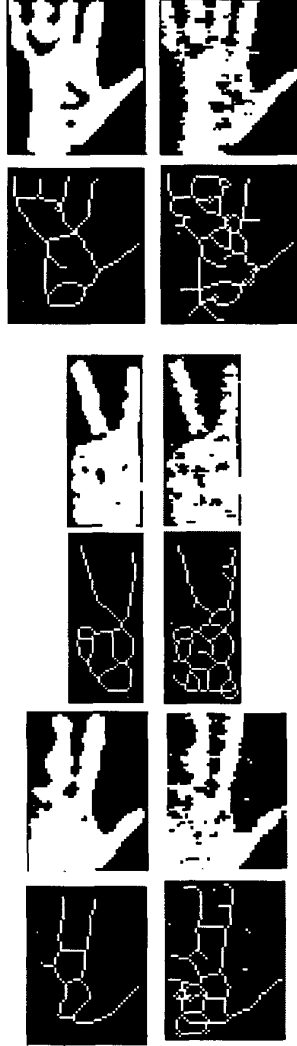
136

nor feature points will have two 8-neighbours on the skeleton.

Maximum Path Length The maximum distance between any two endpoints, measured in pixels traversed along skeleton segments. Note that for a properly skeletonized hand we expect this to be from the endpoint representing the wrist and the most outstretched finger. This distance gives a sense of the overall scale of the skeleton.

We assume that we are starting with a connected skeleton. For each end point on the skeleton we evaluate the path length from that end point to a feature point. The end-point/feature-point pair define a skeleton segment. If the length of this segment is less than 10% of the maximum path length, the segment is removed. Note that when determining the length of a segment, if we start at an endpoint and arrive at another endpoint without encountering a feature point, then either the skeleton is a simple line, or the assumption of a connected skeleton has been violated. In either case an error is flagged and the gesture recognition fails. However, since we assume we start with a connected skin-tone region, this situation should never be encountered in practice.

4.4 Identify the gesture using basic model knowledge

This part of the gesture recognition involves examining endpoints to see if they correspond to potential fingers or a thumb. The current algorithm is simple, and assigns segments to be fingers or thumbs based on the position of their endpoints.

For each endpoint, it is first checked to see if it is a thumb. Let $\vec{p} = (p_x, p_y)^T$ be the location of an endpoint, and let w and h be the width and height, respectively, of the ROI. An endpoint is considered to be a thumb if $p_x \geq 2w/3$ and $h/4 \leq p_y \leq 3h/4$. Any endpoint which is not considered a thumb is considered a finger if $p_y \leq h/3$. If more than one endpoint is identified as a potential thumb, or if the number of finger candidates is 0 or greater than 4, then the gesture identification fails. Note that this requires that the ROI bounding the hand skin-tone blob be accurately determined prior to invoking the algorithm. Finally, it should be mentioned that even if the thumb is not found, a valid gesture can be identified.

Figure 4 show recovered 3- and 5-finger gestures (included in this count is the thumb—note that the algorithm allows for special detection of the thumb).

The gesture identification will work for cluttered backgrounds so long as the hand blob is properly segmented from the clutter. Strong shadows across the boundary of the hand may cause the topology of the skeleton to be altered in an unrecoverable manner. Another cause of prob-
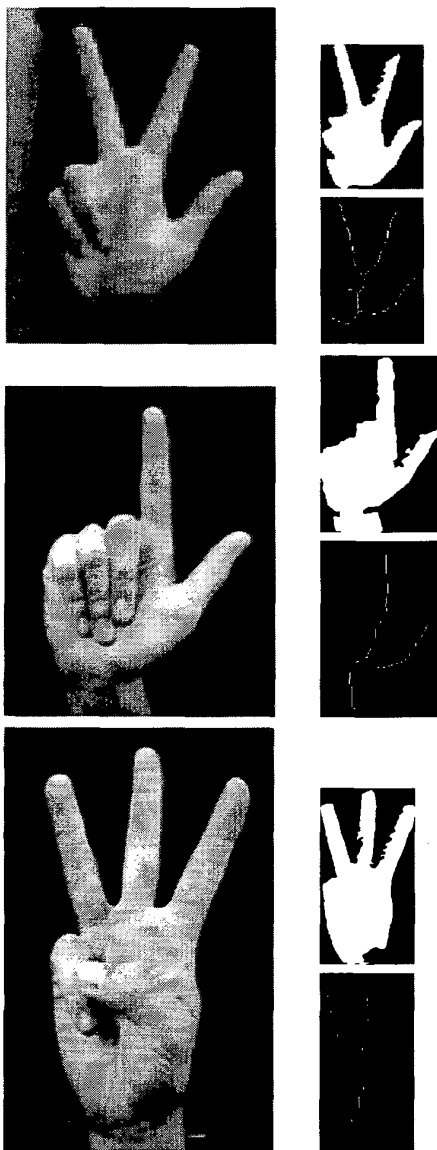

Figure 3: In this figure three pairs of images are shown. In each pair, the image on the right is the recovered skin-tone segment before application of the opening/closing filters, and the image on the left is the result. Skeletons resulting from each are shown below. It is seen that performing this pre-processing step greatly reduces the number of spurious branches on the skeleton, thus making finger detection simpler. The top set of images would fail hand detection based on the finger-proportion criteria (c_4). (Images shown without hole-filling performed.)
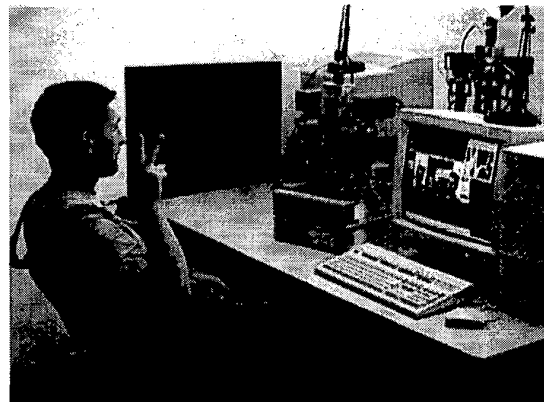
Figure 5: SAVI is shown attending to a hand gesture. In the background a second TRISH-2 head is shown. In this configuration, SAVI uses hand gestures to control pan, tilt and zoom features on the second head.

lems are rings which deviate considerably from skin-tone, causing the finger to be disconnected from the main skin tone blob.

5 Experimental Results

In this section results from running the hand detection and gesture recognition algorithms are presented. In the next section, a brief description of the hardware is given.

To evaluate the performance of the hand gesture processing modules a testing sequence has been developed. The task was to show a two word hand gesture language; first the five finger trigger gesture followed by a free finger counting gesture. For simplicity subjects were asked to count ten times upwards and downwards. Ten subjects performed the sequence and showed in total 200 finger counting gestures. During the evaluation of the results we distinguished between hand gestures shown including and excluding the thumb, because the system is designed to determine if the thumb is raised or not. We obtained an overall correct finger-counting rate of 95.4%. The rates for each number of fingers raised differs significantly. If two fingers are raised the obtained detection rate was 100% regardless if the thumb was raised or not. Nearly the same results could be obtained for the three finger gesture. For the five finger gesture we obtained a detection rate of 90%, in most cases the pinkie finger was not detected correctly. Showing just one finger was also not so successful with 94.5% because the segmentation failed because of shadows. For the four finger gesture the detection rate including the thumb was significantly higher with 96.9% in contrast to the gesture excluding the thumb of 92.2%.

Figure 4: Segmentation with hole-filling, and the resulting skeletons, are shown. Notice the non-uniform backgrounds.
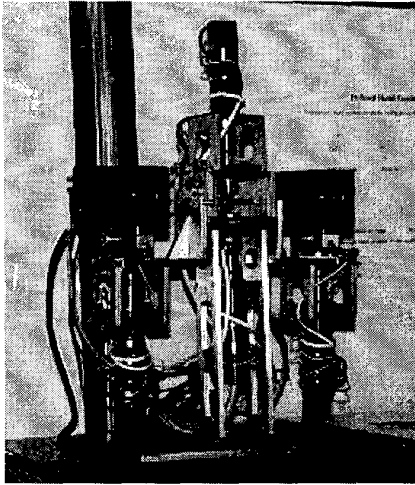
Figure 6: TRISH-2, the binocular head employed by SAVI.

Finally, SAVI has been setup with two stereo heads in different locations, with images from one set of cameras being displayed at the other location. In this setup, commands seen by one set of cameras were used to position the other head. This setup was used as a live demonstration in a non-controlled environment, and performed well in these circumstances. In Figure 5 SAVI is shown in the described configuration attending to a hand gesture.

6 Hardware Specifications

The SAVI hardware system configuration consists of a Tyan Tiger motherboard with dual Pentium II 400MHz MMX CPUs, 512KB L2 cache per CPU, 128 MB RAM, two Imaging Technologies IC4-COMP S-Video frame grabbers, one Precision Motion Control DCX-PC 100 motion controller with 4 DCX-MC110 modules. Two Sony EVI-310 cameras, each having a resolution of 512 × 480 pixels, together with IF-51 serial controllers are connected to the frame grabbers and the motherboard's serial ports. Four servo actuators are connected to the DCX-110 modules for controlling pan and tilt of the head, and pan of each camera. There are 3 main threads in the program. One thread runs the supervisor, the second and third threads control the display of the left and right cameras, respectively. The head motion controller is designed as a separate process that communicates with the vision system via TCP/IP sockets. In this manner the head can be controlled from anywhere on the internet.

7 Conclusion & Discussion

In this paper we have described a real-time technique for detecting the presence of a hand, and determining hand gestures. The real-time constraint requires a relatively sim-

ple approach, but initial results indicate that the system performs well.

Much of the success of the algorithm depends on good segmentation of the skin-tone blob. To date this has worked well against cluttered backgrounds, but real-world environments do sometimes fool it by providing regions whose colour is skin-tone like and that cause the segmentation to fail. The system behaves well when multiple (non-attended) people are in the cameras' field of view, except when another person moves in front of the attended face. This may cause the system to follow the new target instead of remaining with the original face. More work is necessary to remove this unwanted behaviour. It may be possible to improve the hand segmentation by combining motion cues with the skin-tone segmentation, and propagating the segmentation over time throughout an image sequence.

The obtained results have been improved by integrating a sophisticated pre-processing step and applying as much as possible of the available model knowledge. The application of model knowledge is essential for the system because it allows processing in real-time.

Future work to improve the gesture recognition involve using a more sophisticated skeletonization algorithm to reduce the number of spurious segments, and a more sophisticated model for the hand. For example, one might use length and orientation of skeleton segments to disambiguate between thumb and fingers, not just location in the ROI. Another technique would be to compare the topology of the graph resulting from the skeleton to a model hand graph.

Acknowledgements

References

[1] W.T. Freeman and E.H. Adelson. The design and use of steerable filters for image analysis. *IEEE Trans. PAMI*, 13:891–906, 1991.

[2] W.T. Freeman, K. Tanaka, J. Ohta, and K. Kyuma. Computer vision for computer games. In *Proc. IEEE Int. Conf. on Face & Gesture Recognition*, pages 100–105, 1996.

[3] W.T. Freeman and C. Weissman. Television control by hand gestures. In *Proc. Int. Workship on Autom. Face & Gesture Recognition*, pages 179–183, 1995.

[4] R. Hamdam, F. Heitz, and L. Thoraval. Gesture localization and recognition using probabilistic visual learning. In *Proceedings of the IEEE Comp. Soc. Conference on Computer Vision & Pattern Recognition*, volume 2, pages 98–103, 1999.

[5] R. Herpers, G. Verghese, K. Derpanis, R. McReady, J. MacLean, A. Levin, D. Topalovic, L. Wood, A. Jepson, and J. Tsotsos. Detection and tracking of faces in real environments. In *Proceedings of the International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, pages 96–104, September 1999.

[6] Cullen Jennings. Robust finger tracking with multiple cameras. In *Proc. IEEE Int. Workshop on Recognition, Analysis & Tracking of Faces & Gestures in Real-Time Systems*, pages 152–160, 1999.

[7] Rick Kjeldsen and John Kender. Visual hand gesture recognition for window system control. In *Proc. Int. Workshop on autom. Face and Gesture Recognition*, pages 184–188, 1995.

[8] Rick Kjeldsen and John Kender. Finding skin in colour images. In *Proc. IEEE Int. Conf. on autom. Face and Gesture Recognition*, pages 312–317, 1996.

[9] Rick Kjeldsen and John Kender. Toward the use of gesture in traditional user interfaces. In *Proc. IEEE Int. Conf. on Autom. Face and Gesture Recognition*, pages 151–156. IEEE, 1996.

[10] Reinhard Klette and Piero Zamperoni. *Handbook of Image Processing Operators*. John Wiley and Sons, 1996.

[11] E. Milios, M. Jenkin, and J. K. Tsotsos. Design and performance of trish, a binocular robot head with torsional eye movements. *Int. J. of Pattern Recognition and Artificial Intelligence*, 7(1):51–68, 1993.

[12] A. Pitas and A. N. Venetsanopoulos. *Nonlinear Digital Filters: Principles and Applications*. Kluwer Academic Publishers, 1990.

[13] J. K. Tsotsos, G. Verghese, S. Dickinson, M. Jenkin, A. Jepson, E. Milios, S. Stevenson, M. Black, D. Metaxis, S. Culhane, Y. Ye, and R. Mann. Playbot a visually-guided robot for physically disabled children. *Image and Vision Computing*, 19:275–292, 1998.

[14] Akira Utsumi and Jun Ohya. Multiple-hand-gesture tracking using multiple cameras. In *Proceedings of the IEEE Comp. Soc. Conference on Computer Vision & Pattern Recognition*, volume 1, pages 473–478, 1999.

[15] T.Y. Zhang and C.Y. Suen. A fast parallel algorithm for thinning digital patterns. *Comm. ACM*, 27(3):236–239, 1984.