ELSEVIER

Image and Vision Computing 19 (2001) 793–804

image AND vision COMPUTING

www.elsevier.com/locate/imavis

SAVI: an actively controlled teleconferencing system

R. Herpers[a,b,*], K. Derpanis[a], W.J. MacLean[c], G. Verghese[d], M. Jenkin[a], E. Milios[a], A. Jepson[d], J.K. Tsotsos[a]

[a]*York Centre for Vision Research, Department of Computer Science, York University, 4700 Keele Street, Toronto, Ontario, Canada, M3J 1P3*
[b]*Department of Applied Computer Science, University of Applied Sciences, Fachhochschule Bonn-Rhein-Sieg, Grantham Allee 20, 53757 Sankt Augustin, Germany*
[c]*Department of Electrical and Computer Engineering, University of Toronto, 6 King's College Road, Toronto, Ontario, Canada, M5S 3G4*
[d]*Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada, M5S 3G4*

Accepted 10 December 2000

Abstract

A Stereo Active Vision Interface (SAVI) is introduced which detects frontal faces in real world environments and performs particular active control tasks dependent on hand gestures given by the person the system attends to. The SAVI system is thought of as a smart user interface for teleconferencing, telemedicine, and distance learning applications.

To reduce the search space in the visual scene the processing is started with the detection of connected skin colour regions applying a new radial scanline algorithm. Subsequently, in the most salient skin colour region facial features are searched for while the skin colour blob is actively kept in the centre of the visual field of the camera system. After a successful evaluation of the facial features the associated person is able to give control commands to the system. For this contribution only visual control commands are investigated but there is no limitation for voice or any other commands. These control commands can either effect the observing system itself or any other active or robotic system wired to the principle observing system via TCP/IP sockets.

The system is designed as a perception-action-cycle (PAC), processing sensory data of different kinds and qualities. Both the vision module and the head motion control module work at frame rate on a PC platform. Hence, the system is able to react instantaneously to changing conditions in the visual scene.

Keywords: Active vision interface; Teleconferencing system; Face and hand gesture recognition; Computer Vision System

1. Introduction

Most current vision systems just record, compress and transmit video signals with video being captured from static cameras with a wide angle view. They mostly don't include any sophisticated features to react appropriately to changing conditions, e.g. the observed person has moved a bit or is pointing to a particular object or location of interest. Furthermore, based on their need to use wide angle optics no sophisticated but important features are available such as zoom in or track on an object of interest. For that reason static vision systems have to be replaced by more active vision systems, in which sensory data are processed in a closed loop and basic information about the observed scene can be analyzed. The derived information is then reused to guide and/or control the system.

SAVI is based on a robotically controlled binocular head called TRISH-2 (Fig. 1), a new prototype of the stereo vision head TRISH-1 [1]. TRISH-2 consists of two 24-bit colour CCD-cameras with 4 optical degrees of freedom (zoom, focus, iris and exposure/shutter for each camera). In addition, there are four mechanical degrees of freedom (common tilt and rotation as well as independent pan). The head motion controller is inherited from PLAYBOT [2], a previous project in which the head was used. In general, both cameras can be controlled independently or in the master/slave mode. In the first 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.

The vision system presented here is thought of as a smart user interface for teleconferencing, telemedicine, and distance learning applications. For the example of teleconferencing application discussed in this contribution, hand gestures have been chosen to perform particular

* Corresponding author. Corresponding address: Department of Applied Computer Science, University of Applied Sciences, Fachhochschule Bonn-Rhein-Sieg, Grantham Allee 20, 53757 Sankt Augustin, Germany. Tel.: +49-2241-865-217; fax: +49-2241-865-881.

E-mail address: rainer.herpers@fh-bonn-rhein-sieg-de (R. Herpers).

0262-8856/01/$ - see front matter
PII: S0262-8856(00)00107-4

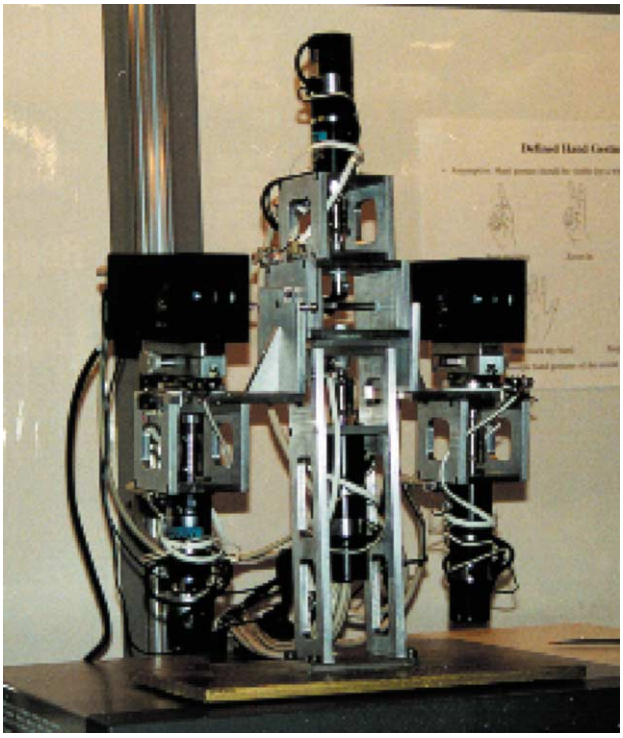

Fig. 1. TRISH-2, the binocular head of SAVI.

control tasks. The hand gesture commands are used to control either the observing system itself or another active vision system connected to the principle one (see Fig. 2). The control commands are thought of as an obvious way to interact with a mostly autonomous or self-guided system. Based on our system design the active vision system is performing closed control loops in real time on visual data only.

The presented work is mostly related to the work done in the area of intelligent environments [3,4], multimodal user interfaces, and advanced multi-media systems [5,6]. Most of these systems do not include any active control. They are based on a fixed camera setup with a wide angle view and do not provide any possibility of changing their internal settings with respect to the above mentioned degrees of freedom. Kjeldsen and Kender [7] 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 [8,9]. Freeman and Weissman [10] 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. [11] 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 and Ohta [12] 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 seven different gestures (based on the number of fingers being held up). Hamdam et al. [13] 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 [14] demonstrates robust gesture detection results against cluttered backgrounds in a system which uses four cameras (three grey-scale, one colour). However, the method is very compute-intensive.

In the next section the different logical units of the system and their interrelationship will be introduced briefly. Subsequently, several modules will be presented completely. Some details concerning hardware specifications and conclusions are given at the end of the contribution.

2. SAVI system description

Stereo Active Vision Interface (SAVI) [15] is a system currently being developed in the labs of the York University Centre for Vision Research. Targeting teleconferencing and telemedicine applications, two robotically controlled binocular heads and their processing units have been connected via TCP/IP sockets (Fig. 2). The chosen system setup should demonstrate that our developed active vision system is able to cope with real time control tasks based on visual input only and regardless of the distance between the receiving and the affecting unit. SAVI provides active control via its supervisor (see Fig. 3), a control module with a finite set of '*states*' and '*transitions*' and it consists mainly of five logical modules:

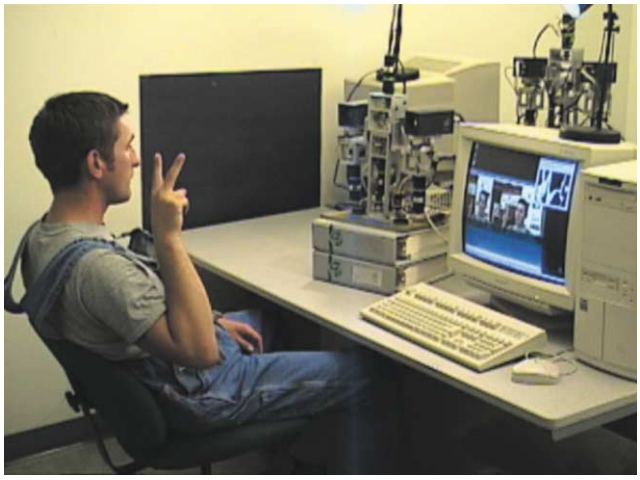

Fig. 2. System setup with two TRISH-2 robotically controlled binocular camera heads. In front of the acting person the receiving system is placed while the affecting system is located in the back (top right corner of the image). The receiving system is monitored on the PC-screen.

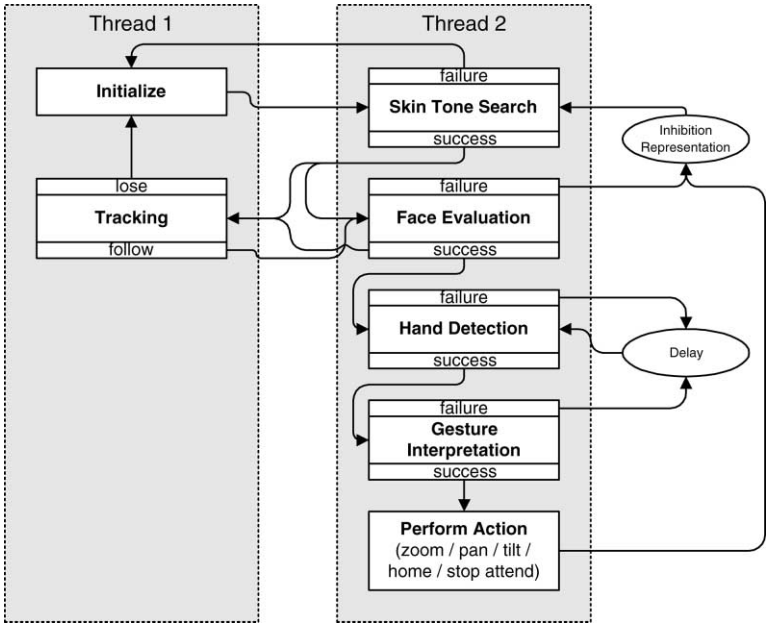

Fig. 3. Control flow of SAVI's supervisor.

Skin tone search — During *skin tone search*, the visual input of both cameras is evaluated to find the most salient skin tone blob. Both cameras are zoomed out, and few features based on raw data of the connected skin tone blobs are evaluated. Regions a priori identified as uninteresting are excluded from further processing steps.

Tracking — During *tracking,* the most salient skin tone blob is kept fixated and actively followed. When the blob stops moving or only a small amount of motion is present, SAVI evaluates the blob either as a possible face or detects the initial hand gesture.

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.

Hand detection — During *hand detection*, a skin tone blob to the right of the attended face is evaluated for an initial hand gesture. The attentive camera is zoomed in and the detected face–hand area remains fixated upon. If successful, the skin tone blob is passed to the *gesture interpretation* module. Otherwise, the process is delayed for a while before a further trail is started.

Gesture interpretation — During *gesture interpretation*, the right hand is fixated upon and tracked with the leading camera, while the other camera is zoomed out and keeps the corresponding face in the visual field. When the hand stops moving, SAVI evaluates the hand blob's shape for identifiable gestures. If successful, a corresponding action is performed. Otherwise, the *hand detection* module is called to find the initial hand gesture again.

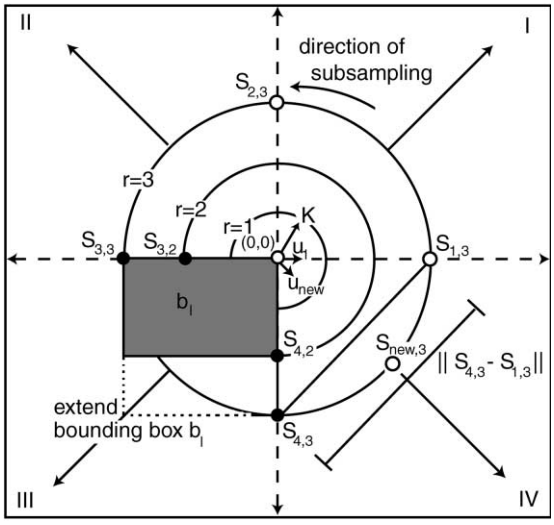

Fig. 4. Method of the radial scanline algorithm. In quadrant I and II the notation is presented. u_1 refers to the unit vector of scanline s_1, $S_{1,3}$ refers to the scanline position of scanline s_1 at circle number $r = 3$ and K denotes the step width. In quadrant III the extension of an existing bounding box is explained by inclusion of scanline point $S_{4,3}$ to the skin colour blob b_1 and in quadrant IV the insertion of a new scanline s_{new} starting at position $S_{new,3}$ between $S_{4,3}$ and $S_{1,3}$ is shown; the associated unit vector is $\mathbf{u}_{new}$.

During the operation of the supervisor, there are several possible failures that may occur (e.g. losing track of the current blob, or a head controller or vision server failure), which require particular error management. All of these control tasks are intergated within the supervisor as well.

3. Skin tone search

For efficient real-time skin colour blob detection, a radial scanline detection method has been developed. Starting at the center of a particular rectangular image region (region of interest, ROI), it scans radially outward along approximately concentric circles with a particular step width K (Figs. 4 and 5). A scanline s is defined by its unit vector $\mathbf{u}$ and the circle number r, which is multiplied by the constant factor K mentioned previously. The unit vector $\mathbf{u}$ is defined by its origin $O = (0,0)$ and a point $P_{\text{unit}} = (x,y)$ on the unit circle, with $\|\mathrm{u}\| = 1$. For simplicity the unit vector is defined only by its point on the unit circle $\mathbf{u} := (x,y)$ assuming that the origin O is fixed. The inital set of scanlines is given by an ordered set (counter clockwise) of unit vectors $\mathbf{U} := \{\mathbf{u}_1,\ldots,\mathbf{u}_h\}$ with $h := 2^a$, $a \geq 2$. The head position $S_{u_i,r}(x,y)$ or for simplicity $S_{i,r}(x,y)$ of a particular scanline s_i is calculated by:

$$S_{i,r}(x,y) = u_i(x,y) \times r \times K$$

Fig. 4 shows the main principles of the scanline algorithm. Fundamental to the radial scanline algorithm is the construction and continuous update of a number of bounding box representations, b_j, which cover the underlying connected skin tone regions. For the computations of the bounding box representations, some additional functions are necessary:

- A boolean function:

$$\text{skin}(x) := \begin{cases} 1; & \text{if colour}(x) = \text{skin tone} \\ 0; & \text{otherwise} \end{cases}$$

- A neighbourhood function $N(s_i) := \{S_{i,r-1},\ S_{i-1,r}\}$ for $i > 1$. For $i = 1$ the neighbourhood N is reduced to $N(s_1) := \{S_{1,r-1}\}$. Only at the beginning of the scan, the bounding box representations b_j are initialized so that in this case no neighbourhood exists.
- A blob association function $bl(x) := b_j$ with b_j blob number of the blob list B, $B := \{\text{null}, b_1, \ldots b_m\}$,, $m \in N$. A skin tone blob is defined as $b := (TL, BR, \Omega)$, where $TL := (x,y)$ refers to the top left corner, $BR := (x,y)$ refers to the bottom right corner of the associated bounding box, and Ω is a set of scanline positions associated with that blob, $\Omega_j := \{S_{i,j};\ S_{i,j}$ are associated to $b_j\}$. For the blob association of a scanline position $S_{t,r}$ with $t \in$ Indexset of U several cases have to be considered:

$$bl(S_{t,r}) = \begin{cases} b_{\text{new}}; & \text{if skin}(S_{t,r}) = 1 \wedge sk(N(S_{t,r})) = 0, \\ bl(N(S_{t,r})); & \text{if skin}(S_{t,r}) = 1 \wedge sk(N(S_{t,r})) = 1 \vee \\ & (sk(N(S_{t,r})) = 2 \wedge bl(S_{t,r-1}) = bl(S_{t-1,r})), \\ M(b_i, b_j); & \text{if skin}(S_{t,r}) = 1 \wedge sk(N(S_{t,r})) = 2 \wedge bl(S_{t,r-1}) \neq bl(S_{t-1,r}), \\ \text{null}; & \text{otherwise.} \end{cases}$$

where b_{new} denotes a new blob representation to be included in B and $sk(X) := \sum_{i=1}^{||X||} \text{skin}(x_i)$ where X is a set of positions $X := \{x_1,\ldots,x_n\}$. To ensure that the neighbourhood relationship between the first $S_{1,r}$ and the last $S_{h,r}$ scanline heads with respect to a particular circle r is also considered, these two scanlines are checked if they are both skin colour after each complete scanning iteration. If so and they do not point to the same blob b_l, the merge function $M(b_1,b_h)$ is called. Therefore, to be precise, the just mentioned blob association function has to be expanded by the following special case: $M(b_1, b_h)$; if $\text{skin}(S_{1,r}) = \text{skin}(S_{h,r}) = 1 \wedge bl(S_{1,r}) \neq bl(S_{h,r})$.

In quadrant III of Fig. 4 the extension of an existing bounding box according to this scheme is explained by a sample scanline point $S_{4,3}$ to be included in the skin colour blob b_l.

- A merge function $M(b_i, b_j) = M((TL_i, BR_i, \Omega_i), (TL_j, BR_j, \Omega_j)) := b_{\text{new}}$ if b_i and $_{\text{b}_j}$ are adjacent, that means they share at least one direct neighbour in terms of the neighbourhood definition of N.

$$M(b_i, b_j) = \begin{cases} TL_{\text{new}}(x) = \begin{cases} TL_i(x); & \text{if } TL_i(x) \leq TL_j(x) \\ TL_j(x); & \text{otherwise} \end{cases} \\ TL_{\text{new}}(y) = \begin{cases} TL_i(y); & \text{if } TL_i(y) \geq TL_j(y) \\ TL_j(y); & \text{otherwise} \end{cases} \\ BR_{\text{new}}(x) = \begin{cases} BR_i(x); & \text{if } BR_i(x) \geq BR_j(x) \\ BR_j(x); & \text{otherwise} \end{cases} \\ BR_{\text{new}}(y) = \begin{cases} BR_i(y); & \text{if } BR_i(y) \leq BR_j(y) \\ BR_j(y); & \text{otherwise} \end{cases} \\ \Omega_{\text{new}} = \Omega_i \cup \Omega_j \end{cases}$$

b_j has to be removed from B while b_i will be replaced by the new merged blob b_{new}.

If the distance between the heads of two successive scanlines exceeds a predefined threshold $\|S_{i,r_f} - S_{i+1,r_f}\| > D$ then a new scanline is spawned and inserted between them with intermediate orientation (Fig. 4, quadrant IV). The new scanline s_{new} is defined only for circle numbers $r \geq r_f$. The first head position is calculated using the

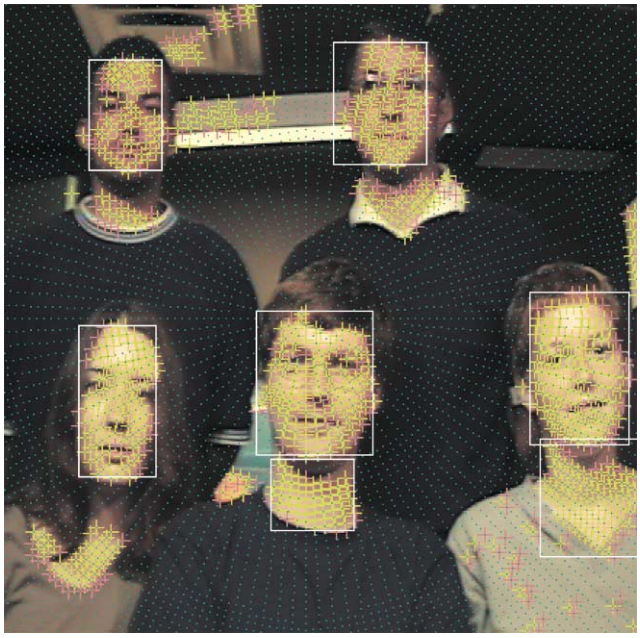

Fig. 5. Computation result of the radial scanline algorithm. Detected skin colour pixels are marked by yellow and pink plus signs dependent on two different thresholds. Connected skin colour regions marked with a white bounding box have already passed consistency checks and may be processed further. All 'yellow' and 'pink' pixels not included in bounding boxes are assigned to be noise or of lower interest.

following equation:

$$S_{\text{new},r_f} = \frac{S_{i,r_f} + S_{i+1,r_f}}{2}$$

while the corresponding unit vector $\mathbf{u}_{\text{new}}$ is given by:

$$\mathbf{u}_{\text{new}} = \frac{S_{\text{new},r_f}}{\left\| S_{\text{new},r_f} \right\|}$$

Fig. 6. Tracking of a face. History (yellow plus signs) and prediction (green square) of the bounding box's centroid of a moving skin colour target.

The new unit vector $\mathbf{u}_{\text{new}}$ must be inserted into U, hence, the elements of U must be reindexed appropriately to maintain the ordered set. In quadrant IV of Fig. 4, the insertion of a new scanline s_{new} starting at position $S_{\text{new},3}$ is shown.

The processing of the scan line algorithm presented ensures that the entire area of interest is considered, by starting with a high density in the center and reducing slightly the scan-density in the periphery (Fig. 5). Subsequently, the most salient skin colour blob satisfying particular constraints, e.g. with respect to the size and position, is processed further.

4. Tracking

Based on a modified version of the radial scan algorithm, an efficient tracking method has been developed. The dynamic change in position(s) of the bounding box(es) surrounding the skin tone blob in each frame of the video sequence is considered and motion vectors with respect to the centroid(s) of the bounding box(es) are calculated. A history of past centroid positions is maintained for each bounding box. Error management ensures that possible misleading positions are excluded. Only a fixed fraction of centroid positions fitting a special homogeneity criterion are kept. A streak of bright plus signs (approximately 30 tail positions of the centroid) in Fig. 6 indicates the past positions of the bounding box centroids, which are stored and/or updated in the location history. During tracking, the size of the bounding box of the attended skin colour object is dynamically adjusted by continuously applying a simplified version of the radial scan algorithm. Furthermore, another error management process ensures that image collisions that occur, while the supervisor is attending multiple moving objects, can also be handled.

To predict the position of the blob's centroid with respect to a particular future time step, a finite set of past motion steps is evaluated. The green square in Fig. 6 indicates the predicted centroid position in the next time step. The computation of a time dependent prediction is particularly important in maintaining the tracking of a skin tone blob because tracking cannot be performed continuously without any break. Breaks may occur if processing power is used for other tasks or if multiple targets are tracked simultaneously. It is clear that confidence will drop rapidly with increasing time delays, but based on a history of past locations for a number of frames of the sequence, such a prediction may be useful for a small time range.

5. Face evaluation

After the skin colour blob detection, a verification of the most salient skin colour blob is performed. Typical facial features must be detected in the bounding box area to ensure that a face is present. The most characteristic and distinctive facial features are the eye regions. Therefore, an

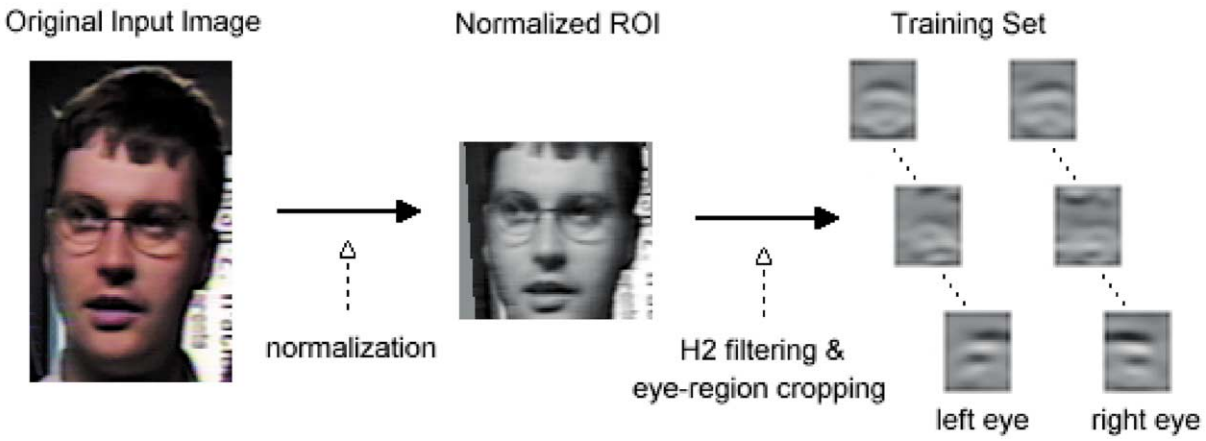

Fig. 7. Processing to obtain a training set. Computation of the eye region representation (training).

eigenimage-based approach to detect and recognize eye regions has been developed. In contrast to the classical eigenimage approach published by Turk and Pentland [16], here only spatially limited facial regions are considered and some other fundamental changes in the calculation have been made. Eye regions, which include the eyebrow, show very prominent horizontally oriented edge and line structures. Accordingly, the eye representation developed here is based on the edge information computed by a horizontally oriented H2 edge detection filter [17]. In the following, we refer to the filtered image as the '*input image*' or simply as '*the image*' because all further processing is done with respect to the H2 filtered image.

$$\mathrm{H2}(x,y,\sigma) = \frac{x}{\sigma}\left(0.3458\left(\frac{x}{\sigma}\right)^2 - 1.559\right)\mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}}$$

Fig. 7 shows a diagram of the processing steps required to obtain a training set for the eye representation. First, the red channel of the 24-bit colour image is manually scaled, aligned, and normalized with respect to the iris centers. Then, the H2 edge detection is performed. To avoid special border considerations during the edge filtering, the entire facial region is convolved first and subsequently, the eye regions are cropped, taking a spatially fixed area surrounding the iris centers. The cropped and H_2-filtered eye regions $F_{\mathrm{H}_2}(I_{\mathrm{eye}})$ of both eyes from a gallery of 60 randomly chosen faces served as training set Γ. The data matrix $\mathbf{A}$ of training set Γ is defined as follows:

$$A = \begin{bmatrix} P_1^{\mathrm{T}} \\ P_2^{\mathrm{T}} \\ \vdots \\ P_n^{\mathrm{T}} \end{bmatrix}$$

where the $P_i = F_{\mathrm{H}_2}(I_{\mathrm{eye},i})$ and each image P is considered an image vector of size $m \times 1$. Using singular value decomposition (SVD), A is decomposed into $A = \mathbf{U}\mathbf{\Sigma}\mathbf{V}_{\mathrm{T}}$ where $\mathbf{U}$ is an $n \times n$, $\mathbf{V}$ is an $m \times m$ and $\mathbf{\Sigma}$ is an $n \times m$ matrix. $\mathbf{\Sigma}$ is a diagonal matrix that contains the singular values S_i of A. It has at most $p = \min(m,n)$ non-zero diagonal elements. $\mathbf{U}$ and $\mathbf{V}$ are both orthogonal matrices. All m columns of $\mathbf{V}$ are eigenvectors of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ corresponding to S. Each eigenvector $\mathbf{V}_i$ is understood as an image vector, which is referred to as an '*eigenimage*', or in our case as an '*eigeneye*' of the training set. The SAVI system uses the first three eigeneyes as a representation or model for eye regions (Fig. 10c). The underlying assumption is that these three eigeneyes carry enough information to reliably detect an eye region when a skin tone region is given.

For detecting an eye position, a skin tone blob under consideration is scaled appropriately using the size parameters of the skin colour blob as initial estimates (Fig. 10a). In the next step, the first eigenimage $\mathbf{V}_1$ (Fig. 10c top row) is applied to the H2 filtered image $F_{\mathrm{H}_2}(I_t)$ (Fig. 10b) and the coefficient map C_1 is computed.

$$c_j = \sum_{i=1}^{m} f_{\mathrm{H}_2}(x_i)V_j(y_i)$$

where c_j refers to the jth-coefficient ($j = 1,\ldots,3$) at a particular image position x and $f_{\mathrm{H}_2}(x_i)$ denotes the value of the filtered image at position x_i.

All image positions with $c_1 \leq 0$ are excluded from further considerations because these are known to be uninteresting in advance. Subsequently, the second and third coefficients are computed for the remaining image positions. The residual R is computed using the following equations for each image position x:

$$r = \frac{p - \sum_{i=1}^{3} c_i^2}{p} \tag{1}$$

where p denotes the power at image position x_i:

$$p = \sum_{i=1}^{m} \left(f_{\mathrm{H}_2}(x_i)\right)^2$$

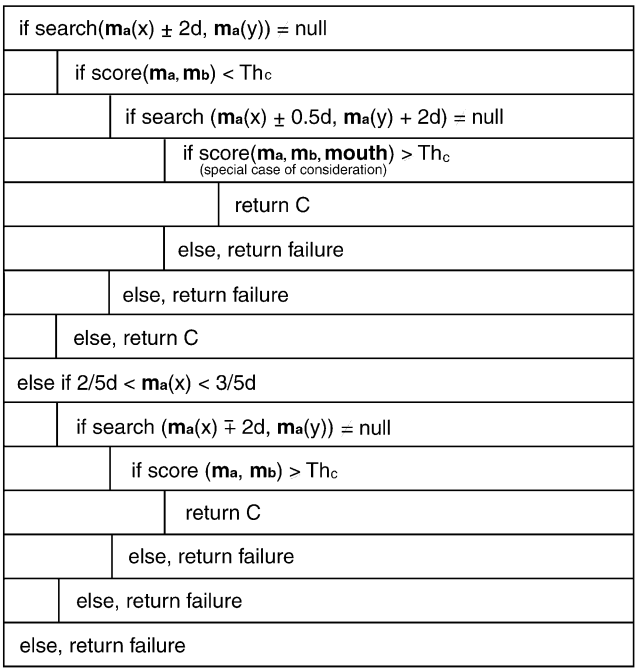

Fig. 8. Application of model knowledge.

Fig. 10e shows a sample residual image, in which all excluded areas have been assigned to a white colour value. Each of the remaining image positions carry a particular grey value. By making these exclusions, time consuming computations for the second and third eigenimage coefficients, as well as for the residual image, are avoided. The values are only computed, on average, for less than half of the image positions in the considered region.

The absolute minimum and a second minimum, which is spatially located at the expected second eye position, are searched for during the evaluation of the residual image (Fig. 8). The presence of both eyes in an approximately horizontal position, also satisfying additional model knowledge assumptions, is taken as a characteristic and reliable feature for a human face (Fig. 9).

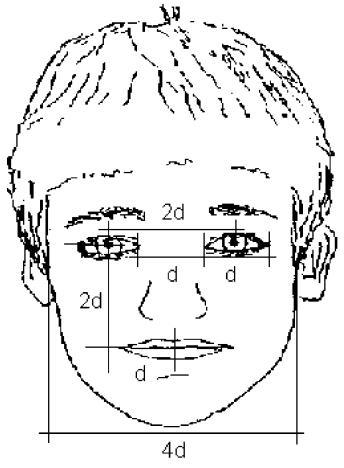

Fig. 9. Ideal facial proportions which are relative to the reference facial distance d.

While the position of the absolute minimum m_a can be computed in parallel with the calculation of the residual map, the position of a second local minimum m_b has to be determined relative to the absolute one in a separate process where $m_a \neq m_b$. Furthermore, model knowledge about the facial relations, FK, must be applied in order to find the expected position for a reliable minimum. For that task a gradient descent-like algorithm called search(x,FK) is applied. The search algorithm is defined iteratively as (for simplicity, in the following the model knowledge assumptions FK have been waived from the function):

$$\text{search}(m_{b,t+1}) = \min(r(N_i(m_{b,t})))i := \{0, \ldots, 3\} \text{ until } m_{b,t+1} = m_{b,t}, \text{ withsearch}(m_{b,0}) = (m_a(x) \pm 2\hat{d}, m_a(y)),$$

(‘ ± ’ depending on whether m_a is associated to the left or the right eye), where $N_i(x) = ((2i + 1) \times (2i + 1)$ pixel neighbourhood surrounding $x) - \sum_{k=1}^{1} N_{k-1}(x)$, for $1 \leq i \leq 3$, with $N_0(x) = x$, and $r(x)$ refers to the residual of x (Eq. (1)). The search for $m_{b,t+1}$ is cancelled at the smallest calculated neighbourhood if $r(m_{b,t}) > \min(r(N_i(m_{b,t})))$, for $1 \leq i \leq 3$.

$\hat{d}$ denotes the estimate of the ‘reference facial distance’ d, which represents an ideal proportion and appears several times in the measurement of distances between facial features [18,19]. Fig. 9 shows several examples of distances of facial features relative to the reference facial distance d. In the current implementation of SAVI, $\hat{d}$ is calculated by the width of the face, which is $4d$ in the ideal case.

The evaluation of the detected minima positions is performed by computing a confidence score score(m_a, m_b) of the absolute minimum m_a and the second minimum m_b.

$$\text{score}(m_a, m_b) := \sum_{i=0}^{N} w_i c_i$$

where the c_is are particular constraint confidence scores, $0 \leq c_i \leq 1$, for $i \in \{1,\ldots,N\}$, and w_i are weighting factors associated with each constraint c_i, $0 \leq w_i \leq 1$ for $i \in \{1,\ldots,N\}$ and $\sum_{i=0}^{N} w_i = 1$. Typical number of constraints is $N = 5$. There are a number of different quantization steps during the calculation of constraint confidence scores c_i, which are referred to by an additional running index j, $j \in \{1,\ldots,N\}$. Each quantization step j is associated with an empirically established, constant confidence value $k_{i,j}$, where $0 \leq k_{i,j} \leq 1$, and a threshold $th_{i,j}$, $th_{i,j} \in \Re$. Most thresholds $th_{i,j}(d)$ are functions of d, e.g. for the confidence rating c_2. In this case, the confidence value depends on the difference between the currently measured distance between the pupil centers and the ideal distance $2d$ and is set relative to d.

- Evaluation of the depth of both minima:

$$c_1(m_a, m_b) := c_{1,h}(r(m_a)) \times c_{1,h}(r(m_b))$$

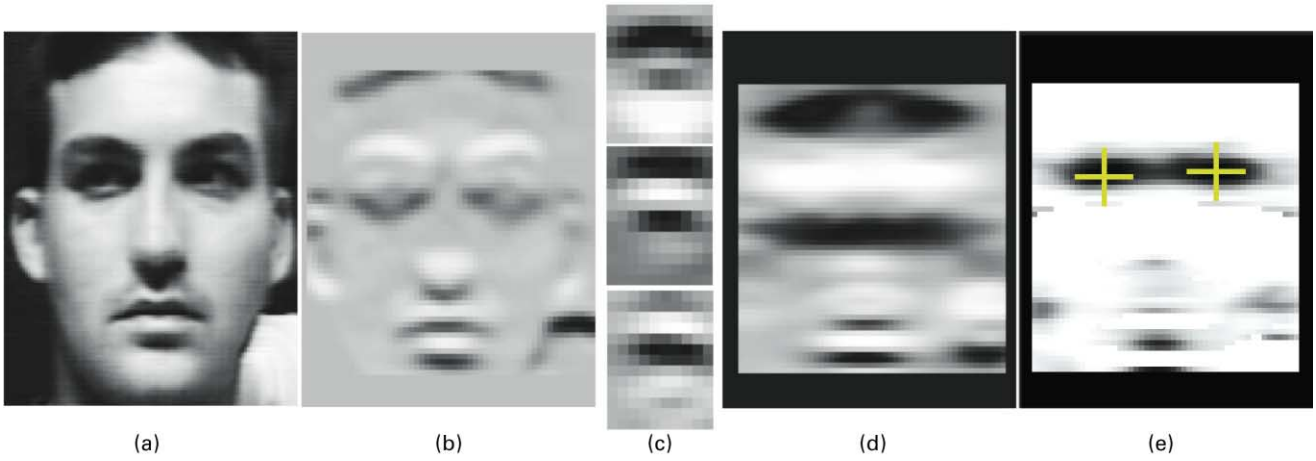

Fig. 10. Evaluation of a possible facial region by detecting both eye regions and checking model knowledge assumptions. (a) Red channel of an appropriately scaled bounding box region of a test image; (b) filtering result using a horizontal oriented H2 filter; (c) first three eigenimages of the eye region representation (from top to bottom); (d) coefficient map of the application of first (top) eigenimage to the filtered image (all bright areas show positive coefficient values); (e) residual map after applying all three eigenimages (dark values show low values or low errors). Only positions which have positive values in the first coefficient map (d) are considered during the evaluation step of the residual image. The white areas in (e) indicate the excluded image positions while the plus signs show the positions where the eyes were detected.

$$\text{where } c_{1,h}(r(x_l)) := \begin{cases} k_{1,j,l}; & \text{if } r(x_l) < th_{1,j,l} \\ 0; & \text{otherwise} \end{cases}$$

and $x_l \in \{m_a, m_b\}$ and $l := \{1, 2\}$.

- Distance between pupil centers:

$$c_2(m_a, m_b) := \begin{cases} k_{2,j}; & \text{if } \|2d - \|m_a - m_b\|\| < th_{2,j}(d) \\ 0; & \text{otherwise} \end{cases}$$

- Horizontal displacement of the pupil centers:

$$c_3(m_a, m_b) := \begin{cases} k_{3,j}; & \text{if } \| \frac{\text{face width}}{2} - \frac{m_a(x)+m_b(x)}{2} \| < th_{3,j}(d) \\ 0; & \text{otherwise} \end{cases}$$

- Vertical displacement of the pupil centers:

$$c_4(m_a, m_b) := \begin{cases} k_{4,j}; & \text{if } \| \frac{\text{face height}}{3} - \frac{m_a(y)+m_b(y)}{2} \| < th_{4,j}(d) \\ 0; & \text{otherwise} \end{cases}$$

- Tilt of the eyes:

$$c_5(m_a, m_b) := \begin{cases} k_{5,j}; & \text{if } \| \frac{\arctan(\|m_a(y)+m_b(y)\|)}{\|m_a(x)+m_b(x)\|} \| < th_{5,j}(d) \\ 0; & \text{otherwise} \end{cases}$$

During a possible postprocessing step, several additional facial features of a more precise value can be examined in more detail (see for instance Refs. [20,21]). To enable such a processing, the facial region has to be zoomed in and tracked appropriately. The tracking of the face will be continued until either it disappears from the visual field or a particular command indicates a new action.

6. Hand detection

The process of hand detection starts when the supervisor detects skin-tone blobs to the left of the currently attended face (all gestures are assumed to be 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) dependent on the location of the face, and segments the image part into a single connected skin-tone region. The red-channel image of the segmented skin tone blob is then filtered using an elongated and vertical oriented G2 filter [17] and one oriented at 45°. The filters are tuned to respond to spatial objects about the expected width for a finger. Scale assumptions are derived from detected facial features. 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 three 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

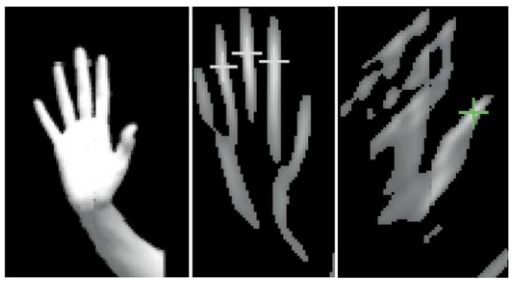

Fig. 11. Successful detection of the trigger gesture. The crosses on the fingers mark the recovered peak responses in the thresholded filter responses.

third of I_T is determined. Designate the locations of the three peak responses from I_F as ξ_1, ξ_2, and ξ_3, where $\xi_1 = (f_{1x}, f_{1y})^T$, etc. These location vectors are ordered so that $\xi_{3x} \geq \xi_{2x} \geq \xi_{1x}$. Likewise designate the locations of the two maximum responses in the right third of I_T as τ_1 and τ_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 applying additional model knowledge assumptions.

Fig. 11 shows the successful detection of a hand. The filtered finger image I_F shows three strong maxima that 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. This algorithm has been implemented, and the obtained results show it to be real-time and robust. It may be affected by reflections or highlights caused by rings that provide problems during segmentation, as a connected skin-tone blob is hard to achieve in this case. Generally one can think of the *Hand detection* as a sort of trigger gesture; it is computed once, and need not be done in frame rate. It has less assumptions, but needs a high degree of robustness. For subsequent gesture recognition and interpretation we use a faster methodology, which however can rely on more assumptions.

7. Gesture interpretation

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 interpretation* state of the supervisor. The gesture interpretation state makes the following assumptions: (a) the ROI contains a hand, (b) the trigger gesture has been detected successfully, (c) the hand gesture is roughly vertical, and (d) a hand gesture has one to five fingers raised. The image part is segmented and binarized. The image may be down sampled to increase speed if desired. The developed image processing involves four further steps:

- *Preprocessing*
- *Skeletonize the image*
- *Clean the skeleton*
- *Identify the gesture*

7.1. Pre-processing

After binarization and possible down sampling, a further pre-processing step is performed to smooth the segmented region. It consists of applying a series of morphological operators [22] followed by a hole filling algorithm. This is necessary to optimize the subsequent skeletonization algorithm. Its application reduces the number of spurious skeleton branches caused by holes and non-smooth contours. This in turn simplifies the task of identifying individual fingers in the subsequent processing steps. Remaining spurious branches are removed in a later processing step.

7.2. Skeletonize the image

The skeletonization is performed using a technique devised by Zhang and Suen [23]. 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) (Fig. 12). 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 that involve solving the heat equation. Spurious skeleton segments are observed, but it turns out that these are easily removed.

7.3. Clean the skeleton

The spurious skeleton segments are observed to be short in comparison with 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* nor *feature points* will have two 8-neighbours on the skeleton.

Maximum path length: The maximum distance between any two *end points*, measured in pixels traversed along skeleton segments. Note that for a properly skeletonized hand we expect this to be from the *end point* representing the wrist and the most out-stretched finger. This distance gives a sense of the overall scale of the skeleton.

We assume that we are starting with a connected

Fig. 12. Successful detection of a 3-finger hand gesture.

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 defines 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 *end point* and arrive at another *end point* 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 interpretation fails. However, since we assume we start with a connected skin-tone region, this situation should never be encountered in practice.

7.4. Identify the gesture using model knowledge

This part of the gesture recognition involves examining *end points* to see if they correspond to potential fingers or a thumb. The current algorithm assigns segments to be fingers or thumbs based on the position of their *end points*. For each *end point*, it is first checked to see if it is a thumb. Let $\rho = (p_x,p_y)^T$ be the location of an *end point*, and let w and h be the width and height, respectively, of the ROI. An *end point* is considered to be a thumb if $p_x \geq 2w/3$ and $h/4 \leq p_y \leq 3h/4$. Any *end point* which is not considered a thumb is considered a finger if $p_y \leq h/3$. If more than one *end point* is identified as a potential thumb, or if the number of finger candidates is zero or greater than four, 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.

Fig. 12 shows a recovered 3-finger gesture (included in this count is the thumb — note that the algorithm allows for special detection of the thumb). Based on the introduced processing steps a three word hand gesture language has been invented (Fig. 13), which may be used to give control commands to another robotically controlled binocular head (see Fig. 2). The gesture identification will work for cluttered backgrounds as 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.

8. Hardware specifications

The hardware system configuration consists of a Tyan Tiger motherboard with dual Pentium II 400 MHz MMX CPUs, 512 KB 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 three main threads in our parallel program. One thread runs the supervisor, the second and third threads control the display of the left and right cameras, respectively. The head motor controller is designed as a separate process that communicates with the vision system via TCP/IP sockets. In this manner the head is controllable from anywhere on the Internet.

9. Conclusions and discussion

One fundamental aspect of SAVI's design is the integration of model knowledge at different stages of processing. This integration is absolutely necessary to reduce the search space and to pay attention to only those aspects of the visible scene that are relevant. To find reliable and robust model knowledge for real world applications, which also satisfies performance requirements, is not an easy task. In addition, the drawback of the selection of particular model knowledge is that it may cause failures in situations that are not covered by the model knowledge assumptions. But there is no easy resolution to this problem because without a minimum set of

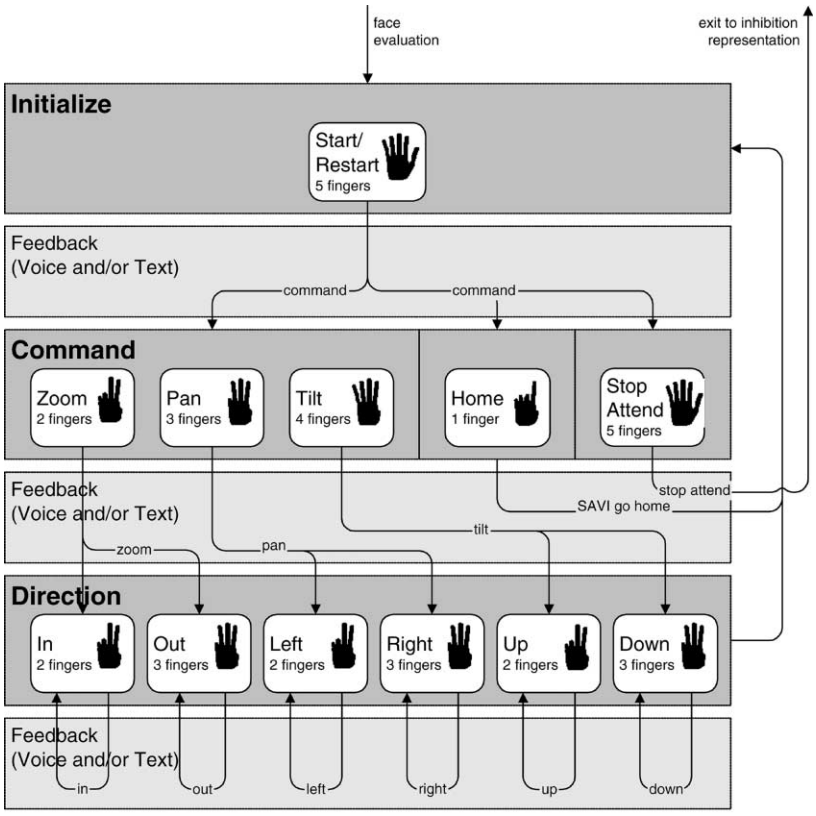

Fig. 13. Three word language developed to control a second binocular stereo head.

constraints and restrictions a real-time system on currently modest hardware may be impossible.

To evaluate the performance of both 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 arbitrary '*finger counting gesture*'. For simplicity subjects were asked to count ten times upwards and downwards. Ten subjects performed the sequence and everybody showed in total 100 '*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.

Fig. 14. System setup with two robotically controlled binocular camera heads and an operator who has raised her hand. The system provides robust and reliable results even with cluttered backgrounds.

We obtained an overall correct '*finger counting rate*' of 95.4%. The individual rates for the several numbers of fingers raised differ 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 wasn't detected correctly. Showing just one finger was also not so successful with 94.5% because the segmentation failed due to 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%. 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 and during tracking (Fig. 14).

SAVI is still under construction and changes may be necessary to improve its performance. But the general underlying idea of SAVI will remain the same. Static vision systems, which do not enable any kind of reaction to changing conditions, will be replaced by more sophisticated

solutions incorporating a particular set of active behaviors needed to cope with more specific and/or extensive applications. The set of actions provided by these new systems is then mainly dependent on the task or application and the environment for which they are designed.

Acknowledgements

We thank the many students and staff members at York University and the University of Toronto who have contributed: Lee Chang, Chakra Chennubhotla, Karine Darcourt, Laura Hopkins, Jonathan Kaufman, Andrea Levin, Rob McCready, Caroline Pantofaru, Douglas Topalovic, Cathrine Stinson, Joyce Wong, Laura Wood, Nancy Yuen. Development of SAVI has been funded by IRIS (Institute for Robotics and Intelligent Systems, a Government of Canada Network of Centres of Excellence), NSERC (the Natural Science and Engineering Council of Canada) and IBM Canada, Centre of Advanced Studies. R. Herpers acknowledges the support of the Deutsche Forschungsgemeinschaft (DFG), Grant: He-2967/1-1 and the German Academic Exchange Service (DAAD) Grant: 222-stpd-III-23/00-cdn.

References

[1] E. Milios, M. Jenkin, J.K. Tsotsos, Design and performance of trish, a binocular robot head with torsional eye movements, IJPRAI 7 (1) (1993) 51–68.

[2] J.K. Tsotsos, G. Verghese, S. Dickinson, M. Jenkin, A. Jepson, E. Milios, S. Stevenson, M. Black, D. Metaxas, S. Culhane, Y. Ye, R. Mann, Playbot a visually-guided robot for physically disabled children, IVC 19 (1998) 275–292.

[3] C.R. Wren, A. Azarbayejani, T. Darrell, A. Pentland, Pfinder: real-time tracking of the human body, IEEE Trans. PAMI 19 (7) (1997) 780–785.

[4] M. Lucente, G.-J. Zwart, A.D. George, Visualization space: a testbed for deviceless multimodal user interfaces, Intelligent Environments Symposium '98, AAAI Spring Symposium, 1998.

[5] J. Cai, A. Goshtasby, C. Yu, Detecting human faces in color images, in: International Workshop on Multi-Media Database Management Systems, 1998.

[6] Y. Raja, S.J. Mckenna, S. Gong, Tracking and segmenting people in varying lighting conditions using colour, in: Third International Conference on Automatic Face and Gesture Recognition, Nara, Japan, IEEE Computer Society Press, 1998, pp. 228–233.

[7] R. Kjeldsen, J. Kender, Finding skin in colour images, in: Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, 1996, pp. 312–317.

[8] R. Kjeldsen, J. Kender, Visual hand gesture recognition for window system control, in: Proceedings of the International Workshop on Automatic Face and Gesture Recognition, 1995, pp. 184–188.

[9] R. Kjeldsen, J. Kender, Toward the use of gesture in traditional user interfaces, in: Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, IEEE, 1996, pp. 151–156.

[10] W.T. Freeman, C. Weissman, Television control by hand gestures, in: Proceedings of the International Workship on Automatic Face and Gesture Recognition, 1995, pp. 179–183.

[11] W.T. Freeman, K. Tanaka, J. Ohta, K. Kyuma, Computer vision for computer games, in: Proceedings of the IEEE International Conference on Face and Gesture Recognition, 1996, pp. 100–105.

[12] A. Utsumi, Jun Ohya, Multiple-hand-gesture tracking using multiple cameras, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 1999, pp. 473–478.

[13] R. Hamdam, F. Heitz, L. Thoraval, Gesture localization and recognition using probabilistic visual learning, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 1999, pp. 98–103.

[14] C. Jennings, Robust finger tracking with multiple cameras, Proceedings of the IEEE International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems, 1999, pp. 152–160.

[15] R. Herpers, G. Verghese, K. Derpanis, R. McCready, J. Maclean, A. Levin, D. Topalovic, L. Wood, A. Jepson, Detection and tracking of faces in real environments, in: Proceedings of the IEEE International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems, 1999, pp. 96–104.

[16] M. Turk, A. Pentland, Eigenfaces for recognition, J. Cogn. Neurosci. 3 (1991) 71–86.

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

[18] L. da Vinci, Treatise on Painting, Vol. 1, Princteon University Press, Princeton, New Jersey, 1550.

[19] R. Herpers, GAZE: a common attentive processing strategy for the detection and investigation of salient image regions, Tech-Rep. No. 9714, University of Kiel, Germany, 1997.

[20] R. Herpers, H. Kattner, H. Rodax, G. Sommer, GAZE: an attentional processing strategy to detect and analyze the prominent facial regions, in: M. Bichsel (Ed.), International Workshop on Automatic Face- and Gesture-Recognition, Zurich, Switzerland, 1995, pp. 214–220.

[21] R. Herpers, M. Michaelis, K.H. Lichtenauer, G. Sommer, Edge and keypoint detection in facial regions, in: Second International Conference on Automatic Face and Gesture Recognition, Killington, Vermont, IEEE Computer Society Press, 1996, pp. 212–217.

[22] R. Klette, P. Zamperoni, Handbook of Image Processing Operators, J. Wiley and Sons, 1996.

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