Active Visual Control by **S**tereo **A**ctive **V**ision **I**nterface SAVI

R. Herpers[1,2], K. Derpanis[2], D. Topalovic[2], C. Pantofaru[2],
W. J. MacLean[3], G. Verghese[4], A. Jepson[4], J. K. Tsotsos[2]

[1] Department of Computer Science, University of Applied Sciences, FH Bonn-Rhein-Sieg, Grantham Allee 20, 53757 St. Augustin, Germany
[2] York Centre for Vision Research, Department of Computer Science, York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada
[3] Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ontario, M5S 3G4, Canada
[4] Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Ontario, M5S 3G4, Canada

Abstract A real-time vision system called SAVI is presented which detects faces in cluttered environments and performs particular active control tasks based on changes in the visual field. It is designed as a Perception-Action-Cycle (PAC), processing sensory data of different kinds and qualities in real-time. Hence, the system is able to react instantaneously to changing conditions in the visual scene.
Firstly, connected skin colour regions are detected while the visual scene is actively observed by binocular vision system. The detected skin colour regions are merged if necessary and ranked by their order of saliency. Secondly, 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. This 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.

1. Introduction

Most current vision system 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.
The vision system presented here is thought of as a smart user interface for teleconferencing, telemedicine, and distance learning applications. For the example teleconferencing application discussed in this paper, hand gestures have been chosen to perform particular control tasks (see fig. 1). The hand gesture commands are used to control either the observing system itself or another active vision system connected to the principle one. 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.
In the next section the different logical units of the system and their interrelationship will be introduced. We will concentrate on a more detailed description of the hand gesture recognition system rather then on the face detection and tracking which have been published before [2].

2. SAVI System Description

Stereo Active Vision Interface (SAVI) [2] is a system currently being developed in the labs of the York Centre for Vision Research. Targetting teleconferencing and telemedicine applications, two robotically controlled binocular heads and their processing units have been connected via TCP/IP

sockets (fig. 1). 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. Two TRISH-2 camera heads are employed [4]; each 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 PLAYBOT [5], a previous project in which the head was used. 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. SAVI provides active control via its supervisor (see fig. 2), a control module with a finite set of *'states'* and *'transitions*' and it consists mainly of 5 logical modules:

- Skin tone search [2],
- Tracking [2],
- Face evaluation [2],
- Hand detection, and
- Gesture interpretation.

Figure 1. System setup with two TRISH-2 robotically controlled binocular camera heads.

2.1 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.

For efficient real-time skin colour blob detection, a radial scan line detection method has been developed. Starting at the centre of the attentive image region, it scans radially outward along approximately concentric circles with *step width* and *scan line distance* as free parameters [2]. Motivated by the human retina this approach enables a higher sampling density in central regions than in peripheral regions. The free parameters are computed dynamically dependent on the extension of the ROI and the particular use of the search method.

2.2 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.

Based on an adapted version of the radial scan line algorithm, an efficient and robust tracking method has been developed. For that, dynamic changes in position of the bounding box surrounding the attentive region in each frame of the video sequence are considered and motion vectors with respect to the centroid of the bounding box are computed [2].

2.2 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.

An *Eigenimage* based approach to detect and recognize eye regions has been developed to verify the most salient skin colour blob as a possible face [2]. In contrast to the classical eigenimage approach [6], here only spatially limited facial regions are considered. An *eye region representation* based on horizontal edge information, which includes also the eyebrow area, is established. The first three eigenimages are kept to model the *`eye space`*. For the detection of unknown eye regions the coefficient maps and the residual image are evaluated as well as model assumptions of the spatial relationship of the eye regions. The computed knowledge of the detected face, in particular its scale and position is essential for the subsequent hand gesture processing steps.

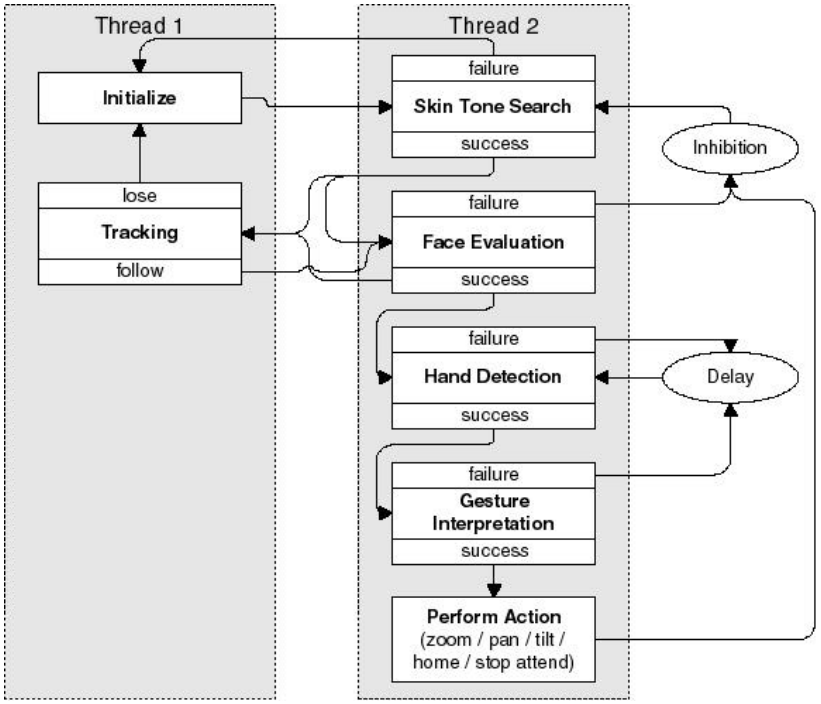

Figure 2. Control flow of SAVI's supervisor.

2.4 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 [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 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 $\mathbf{I_F}$ represent the image filtered with the vertical (finger) filter, and $\mathbf{I_T}$ represent the image filtered with the **45°** (thumb) filter. Each filtered image is thresholded to remove responses less than its average intensity value. Next, $\mathbf{I_F}$ 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 $\mathbf{I_T}$ is determined. Designate the

locations of the three peak responses from $\mathbf{I_F}$ as x_1, x_2, and x_3, where $x_1 = (\mathbf{f_{1x}}, \mathbf{f_{1y}})^T$, etc. These location vectors are ordered so that $x_{3x} \geq x_{2x} \geq x_{1x}$. Likewise designate the locations of the two maximum responses in the right **1/3** of $\mathbf{I_T}$ as t_1 and t_2. (Two peaks are used by the thumb scoring function to ensure that a dominant peak is present.) The locations of the $\mathbf{I_F}$ maxima are then checked applying additional model knowledge assumptions.

red-channel image I_F I_T

Figure 3. Successful detection of the trigger gesture. The crosses on the fingers mark the recovered peak responses in the thresholded filter responses.

Figure 3 shows the successful detection of a hand. The filtered finger image $\mathbf{I_F}$ shows three strong maxima which meet the scoring criteria for fingers. The filtered thumb image $\mathbf{I_T}$ 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 which 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 real-time. It has less assumptions, but needs a high degree of robustness. For subsequent gesture recognition we use a faster methodology, which relies on more assumptions.

2.5 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: - the ROI contains a hand, - the trigger gesture has been detected successfully, - the hand gesture is roughly vertical, - a hand gesture has 1 to 5 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:

- Pre-processing, application of morphological operators including hole filling.
- Skeletonize the image.
- Clean the skeleton.
- Identify the gesture applying model knowledge.

2.5.1 Gesture Interpretation

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 [3] 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 (see Section 2.5.3).

2.5.2 Skeletonize the image

The skeletonization is performed using a technique devised by Zhang & Suen [7]. 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. 4). 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.

Original ROI Binary image after pre-processing Result of the skeletonization

Figure 4. Successful detection of a 3-finger hand gesture.

2.5.3 Clean the skeleton

The spurious skeleton segments are observed to be short in comparison to segments representing the fingers and wrist (fig. 4 right image). 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 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.

2.5.4 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 $r = (p_x, p_y)^T$ 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 **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 shows a recovered 3-finger gesture (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 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.

3. Results

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 have to be raised 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 trigger gestures and 200 *'finger counting gestures'*. During the evaluation of the results we distinguished between hand gestures 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 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 (fig. 4). 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% since the segmentation failed often 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%. 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. The obtained results have been improved by integrating a 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 ensures the processing in real-time.
SAVI is still under construction but the general underlying idea of the system design 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 kinds of active behaviours.

Acknowledgements
The authors would like to thank the following for their assistance in this work: R. McCready, A. Levin and L. Wood. Development of SAVI has been funded by IRIS, NSERC and IBM Canada. R. Herpers acknowledges support of the Deutsche Forschungsgemeinschaft (DFG), Grant: He-2967/1-1.

References

[1] Freeman W.T. and E.H. Adelson. The design and use of steerable filters for image analysis. *IEEE Trans. PAMI,* 13:891-906, 1991.
[2] Herpers R. et al. Detection and tracking of faces in real environments. *Proc. IEEE Int. Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, 96-104, 1999.
[3] Klette R. and P. Zamperoni. *Handbook of Image Processing Operators*, J. Wiley and Sons, 1996.
[4] Milios E., M. Jenkin, J.K. Tsotsos. Design and performance of trish, a binocular robot head with torsional eye movements, *IJPRAI* 7(1): 51-68, 1993.
[5] Tsotsos J.K. et al. Playbot a visually-guided robot for physically disabled children, *IVC* 19: 275-292, 1998.
[6] Turk M., A. Pentland. Eigenfaces for Recognition. *J. Cogn. Neurosci.* Vol. 3: 71-86, 1991.
[7] Zhang T.Y., C.Y Suen. A fast parallel algorithm for thinning digital patterns. *Comm. ACM* 27(3):236-239, 1984.