Detection and Tracking of Eyes for Gaze-camera Control

Shinjiro Kawato and Nobuji Tetsutani
ATR Media Information Science Laboratories
{skawato | tetsutani}@atr.co.jp

Abstract

We propose new algorithms to extract and track the positions of eyes in a real-time video stream. For extraction of eye positions, we detect blinks based on the differences between successive images. However eyelid regions are fairly small. We propose a method to distinguish them from head movement. For eye position tracking, we use an updating template based on a "Between-the-Eyes" pattern instead of the eyes themselves. Eyes are searched based on the current position of "Between-the-Eyes" and their geometrical relations to the position in the previous frame. The "Between-the-Eyes" pattern is easier to locate accurately than eye patterns. We implemented the system on a PC with a Pentium III 866MHz CPU. The system runs at 30 frames per second and robustly detects and tracks the eyes.

1 Introduction

Head-free and head-off gaze detection systems can provide a good interaction interface for computers. A head-off gaze-camera captures an eye image and analyzes it to estimate the gaze direction. To increase the accuracy of this estimation, the eye image must be taken at high resolution. Consequently, the view-field is fairly small, typically about 4×3 cm so that an eye is fully in the image, and the depth of the focus is very shallow. With only a gaze-camera, users cannot move their heads. To make the system head-free, some other means is required to detect and track the eye in order to control pan, tilt, and focus of the gaze-camera.

This paper proposes algorithms for eye detection and tracking that allow such a system to control a gaze-camera but not gaze detection itself.

Matsumoto has reported excellent eye tracking performance in his gaze direction measurement system[5]. However each user has to register his/her face and feature points beforehand. We also use video cameras to detect and track eyes, but we want to treat this subject more generally and only take template patterns on the fly.

The situation we assume is that a user is sitting in front of a gaze-camera at a distance of 50—100 cm and is looking at a display monitor. In this situation, the range of face movement is not large, so we can use a fixed camera to take a face image with enough resolution to detect eyelid movements or blinks. Eyelid location means eye location.

A human blinks involuntarily and periodically. It doesn't take us much time to wait for natural blinks. In the case of using gaze detection as an interface, we can even expect a user's voluntary blink. The fact that both eyes blink at the same time provides useful information for distinguishing the blinking from other motions in the scene.

In a recent survey paper on face detection[3], only one report[2] was mentioned in which the authors detected faces from blinking. Their blink detection method was based on the differences between successive images. According to their description, significant differences in luminance appear only in the small boundary region around the outside of the head and eyelid movement regions. This means the head is nearly still, at least between the two successive images when blinks occur.

In another paper[1], eyes were also located by blink detection. First, they extracted a face region based on a combination of background subtraction and skin-color information. Then they analyzed luminance differences between successive images in the face region to extract blinking. However, there was no description of head movement compensations. Therefore, the head is assumed to be nearly still.

We propose another blinking detection algorithm based on the differences between successive images, which distinguishes eyelid movements from head movement so that it detects blinks even while the head is moving. This function increases the flexibility of applications.

Once we detect locations of the eyes, we have to track them in successive images. Considering the changes in face orientation, we cannot apply a simple template matching technique. Even when we update the templates frame-by-frame with patterns of current located positions, the tracking points will gradually migrate out from the eyes, because the tracked points are not always the centers of the eye pattern.

In the report [1], each eye is tracked with a combination of a fixed template and an updating template. The fixed eye template is taken when a blink is detected, and the updating template is taken at located eye position in the latest image. When they search for an eye with these template patterns, they put 3/4 weight on the fixed template and 1/4 on the updating one. The reason for keeping the initial template is not

described explicitly, but it seems to be to prevent the tracking point migration problem mentioned above. In principle, the template matching technique is weak in pattern rotations and scale changes. If the initial template is kept through the entire tracking process, it likely to fail in tracking when a user declines his/her neck or rotates the face in the image plane.

We propose a new eye tracking algorithm with an updating template. We do not take templates of the eyes themselves. We take a template pattern of "Between-the-Eyes"[4] and update it frame-by-frame. "Between-the-Eyes" has good features for accurately tracking, and is observable for a wide range of face orientations. We track "Between-the-Eyes" and search for eyes in a very simple manner around positions relative to "Between-the-Eyes" where the eyes are located in the previous image. The template of "Between-the-Eyes" is updated based on the current located eye positions.

To measure the distance from camera to face, we apply a conventional binocular stereo method. The stereo method is well known, so we don't explain it in detail in this paper.

In section 2, a method to distinguish eyelid movements from head movement is described. In section 3, we describe the details of the blink detection algorithm, and in section 4, the details of the eye tracking algorithm. In section 5, the implementation and some experimental results are described. Section 6 concludes the paper.

2 Head Movement Cancellation

A technique to analyze the differences between successive images is often used to detect moving objects. Detecting the eyelid movements of blinking is not difficult when the head and the background are still. Detection of eyelid movements while the head is moving is not simple, because many candidates can appear not only around the head boundaries but also around the eyebrows, nose, mouth, and other parts of the face. If we can distinguish those caused by head movement from others caused by other movements, detection of eyelid movements becomes easier. Although those based on changes in mouth shape or movement of eyebrows may still remain, they should be distinguished by using other conditions.

Head motion can be categorized into five cases: movements (1) parallel to the image plane, and (2) parallel to the optical axis of the lens, and rotations of (3) turning right or left, (4) facing up or down, and (5) neck declining. We assume the interval from frame to frame is adequately short so that a movement within the interval is small.

First, we consider the case of the movement parallel to the image plane.

Figure 1 shows a case where a colored circle plate with a small white mark in frame f_{t-1} moves by $\boldsymbol{d}$, and f_t is the

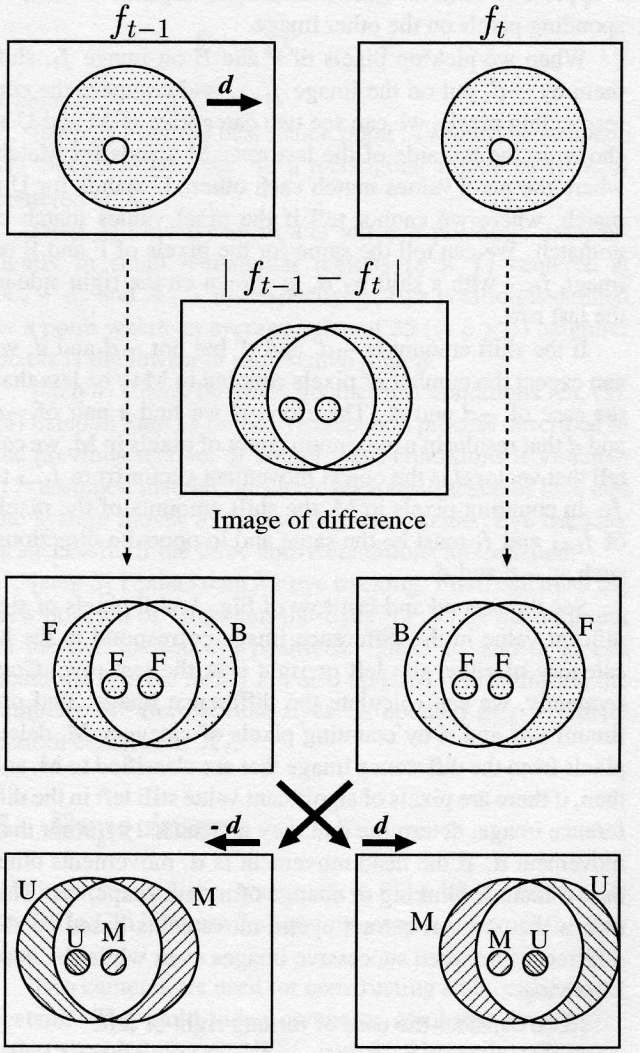

Figure 1: *Estimation of movement vector.*

resulting next-frame image. The background does not necessarily have to be uniform but it has to be still. The second row shows the difference image in which each pixel has an absolute value of the difference in corresponding pixels on f_{t-1} and f_t.

The pixels on f_{t-1} and f_t where a significant value appeared in the difference image can be categorized into F and B, as shown in the third row. Here, F stands for Foreground or points on the circle plate, and B stands for Background. Of course we cannot tell which part is F and which part is B, because we don't know the moving object. While part B is hidden in one image and appears in the other image, part

F appears in both images, indicating that part F has corresponding pixels on the other image.

When we pick up pixels of F and B on image f_t, shift them by $-\boldsymbol{d}$, put on the image f_{t-1}, and compare the corresponding pixels, we can see two categories of M and U as shown in the left side of the last row. M stands for Match, where the pixel values match each other. U stands for Unmatch, where we cannot tell if the pixel values match or unmatch. We can tell the same for the pixels of F and B on image f_{t-1} with a shift by $\boldsymbol{d}$, as shown on the right side of the last row.

If the shift amount is $-\boldsymbol{d}'$ and $\boldsymbol{d}'$ but not $-\boldsymbol{d}$ and $\boldsymbol{d}$, we can expect the number of pixels counted in M to be less than the case of $-\boldsymbol{d}$ and $\boldsymbol{d}$. Therefore, if we find a pair of $-\boldsymbol{d}$ and $\boldsymbol{d}$ that results in a maximum count of pixels in M, we can tell that vector $\boldsymbol{d}$ is the object movement vector from f_{t-1} to f_t. In counting pixels in M, the shift amounts of the pixels of f_{t-1} and f_t must be the same and in opposite directions, such as $-\boldsymbol{d}$ and $\boldsymbol{d}$.

See the second and last row of Fig. 1. All pixels of significant value in the difference image correspond to the M category of either the left or right side the last row. Consequently, we can calculate the difference image, find optimum $-\boldsymbol{d}$ and $\boldsymbol{d}$ by counting pixels of category M, delete pixels from the difference image that are classified to M, and then, if there are pixels of significant value still left in the difference image, determine that they are caused by other than movement $\boldsymbol{d}$. If the head movement is $\boldsymbol{d}$, movements other than $\boldsymbol{d}$ include blinking or change of mouth shape, etc. This means that we can extract eyelid movements based on the differences between successive images even while the head is moving.

Next, consider the case of turning right or left.

Figure 2 shows a situation when a face looking at the camera is about to turn left. From right edge of the face, some self-occluded part appears and, at the left edge, some part disappears by self-occlusion. These parts may have significant value in the difference image. Such parts appear only at the boundary region of moving objects, where eyes are not likely to exist. Therefore we can remove the boundary pixels from the difference image before searching for $\boldsymbol{d}$ as mentioned above. The movement of the central part of the face, on the other hand, is assumed to be parallel to the image plane provided that the amount of rotation is small.

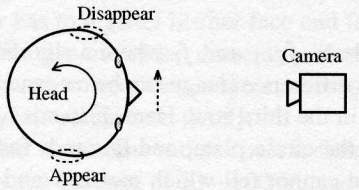

Figure 2: *Case of head turning.*

In the cases of facing up or down and neck declining movements, the rotation center is in the neck, and we can assume that the face movement is parallel to the image plane.

A movement parallel to the optical axis of the lens results in hair lines in the difference image, which can be removed by a smoothing filter.

3 Detection of the Eyes

The detection process we implemented in our prototype system is described below in detail. To distinguish true eyes from false candidates, we assume such conditions as blinks occurring in both eyes at the same time, the distance between the eyes being within a certain range, the face rotation angle in the image plane being within a certain range, the gray pattern at the "Between-the- Eyes" being almost mirror-symmetric. All parameters in the description are given real values in the section of experiment.

[Step 1] Make a binarized difference image $\boldsymbol{D}$. Put pixel value 1 when the absolute value of the difference of corresponding pixel values in the previous image and current image exceeds a threshold value N, else put 0. Because video signals are always fluctuating, even when the scene is static, we should apply such a threshold value. N can be interpreted as noise level.

[Step 2] If the number of 1's in image $\boldsymbol{D}$ exceeds a threshold value G, skip all of the processes below and wait for the next frame. In this case, the head movement seems to be larger than we expect, or something is changing in the background.

[Step 3] If the number of 1's in image $\boldsymbol{D}$ is less than a threshold value E, skip to Step 7. In this case, the head seems to be still. Therefore, the head movement cancellation process should be skipped.

[Step 4] Scan every raster line of image $\boldsymbol{D}$ from the left end and put 0 to k pixels from the first pixel of 1. Do the same scanning from the right end.

[Step 5] Find optimum shift vector $\boldsymbol{d}$ as described in the previous section. Search range of $\boldsymbol{d} = (d_x, d_y)$ is $d_x = -m, .., m$ and $d_y = -n, ..., n$.

[Step 6] Based on the $\boldsymbol{d}$ obtained in Step 5, find pixels in image $\boldsymbol{D}$ that are classified to M as mentioned in the previous section and change their value 1 to 0.

[Step 7] Apply a filter to image $\boldsymbol{D}$ for smoothing and removing isolated pixels. Among eight neighbors and the center pixel, if more than three pixels have value 1, put 1 to the center pixel, otherwise put 0.

[Step 8] Calculate a bounding box for each connected component in image $\boldsymbol{D}$. Then each center of the bounding box is a candidate of eye position.

[Step 9] If the number of eye candidates is less than two, skip all of the steps below. Eyes are not detected.

[Step 10] For each candidate, calculate S or a sum of differences between the previous image and the current image of $(a \times b)$ pixels centered at the candidate point. Then leave the top two values of S as the final candidates for the eyes.

[Step 11] Test four conditions below for the final candidates. If any of the conditions is not satisfied, the candidates are not eyes.

(1) Both S's calculated in Step 10 are over a threshold F.

(2) The distance between the candidates is over L_{min}.

(3) The distance between the candidates is less than L_{max}.

(4) The angle between the line connecting the candidates and the raster line is less than A degrees.

[Step 12] Finally, test the mirror-symmetry of the gray pattern at the midpoint of the candidates. First, derotate the current image at the midpoint of the candidates so that the line connecting them becomes parallel to the raster line. Then extract $(c \times d)$ pixels for the test. Add d pixel values for each column to make a projection profile. Represent the profile by the percentages for right and left halves respectively. If the difference of each three-column sum between symmetric positions in right and left halves does not exceed p, then those candidates are eyes.

4 Tracking of the Eyes

For tracking the eyes, we apply a template matching technique. To cope with changes in face orientation, we should use updating templates. However patterns of eyes change drastically and quickly when they blink, which makes it possible to locate eye positions from the differences between successive images. Even updating templates cannot follow these changes. When we use template patterns of the eyes themselves and update them frame-by-frame, the tracking points migrate out from the eye positions.

We have a template of "Between-the-Eyes"[4] instead of the eyes themselves and track it. Its pattern is fairly stable for any change in facial expression. It has a relatively bright part at the nose-bridge and relatively dark parts at the eyes like wedges on both sides. This is a very good feature for accurate location by template matching. After the detection of "Between-the-Eyes," the eyes are re-searched in very small areas, because their relative positions to "Between-the-Eyes" are known in the previous frame. Even if we use a template of "Between-the-Eyes," we have to update it frame-by-frame. We update it based on the current eye positions.

Now, we describe the steps of the eye tracking process. Before that, when eyes are located by blink detection, a pattern of $s \times t$ pixels centered at the midpoint of the eyes is saved as a template of "Between-the-Eyes." The relative positions of eyes to it, $\boldsymbol{e}_r$、 $\boldsymbol{e}_l$ $(\boldsymbol{e}_r = -\boldsymbol{e}_l)$ are also saved.

[step 1] Predict the position of "Between-the-Eyes." When its position in the previous frame and in the two previous frame are $\boldsymbol{X}_{t-1}$ and $\boldsymbol{X}_{t-2}$, then the predicted point $\hat{\boldsymbol{X}}_t$ in the current frame is

$$\hat{\boldsymbol{X}}_t = 2\boldsymbol{X}_{t-1} - \boldsymbol{X}_{t-2}, \tag{1}$$

where $\boldsymbol{X}_{-1} = \boldsymbol{X}_0$.

[step 2] Find the best match point $\tilde{\boldsymbol{X}}_t$ with the template of "Between-the-Eyes" in a rectangular region of $(p \times q)$ centered at $\hat{\boldsymbol{X}}_t$.

[step 3] Find the right and left eye positions independently in small rectangular regions $(e \times f)$ centered at $\tilde{\boldsymbol{X}}_t + \boldsymbol{e}_r$ and $\tilde{\boldsymbol{X}}_t + \boldsymbol{e}_l$, respectively. Eye position is defined as a point where an average value of 25 $(= 5 \times 5)$ neighbor pixels is the lowest in each search region.

[step 4] Check the three geometrical conditions (2), (3), (4) listed in Step 11 in the eye detection process described in the previous section. If one of these conditions is violated, it is assumed that the eyes have been mistracked or lost, and the system moves to the eye detection mode. Eye tracking is successful if the three above conditions are satisfied.

[step 5] Update data for eye tracking. First, calculate the new position of "Between-the-Eyes" or $\boldsymbol{X}_t$ as the midpoint of the two located eye positions. Then, $\boldsymbol{e}_r$ and $\boldsymbol{e}_l$ as eye positions relative to $\boldsymbol{X}_t$ are also updated. Furthermore, the template of "Between-the-Eyes" is updated as $s \times t$ pixel pattern centered at $\boldsymbol{X}_t$.

5 Experiment

We implemented the system on a PC with a Pentium III 866MHz CPU. Figure 3 shows the configuration of the system.

Two cameras are used for constructing a binocular stereo system. The multi-video composer combines four NTSC video signals into one NTSC signal. Although each video image becomes one fourth of the original size, the system can capture four synchronized images simultaneously with an ordinary video capture board. No other special hardware for image processing is implemented.

To achieve real-time processing, the capture size is 320 × 240 and the upper half of 320 × 120 is used, an area which corresponds to the two camera images.

The parameters we used in the experiment are shown in Table 1 and explained in the descriptions of eye the detection

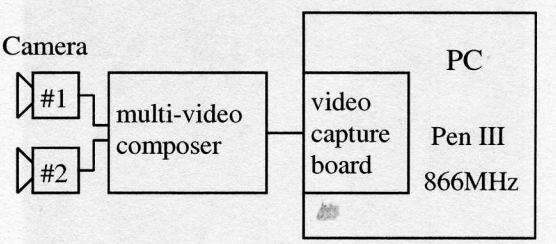

Figure 3: *Configuration of the system.*

and tracking steps in the previous sections.

Table 1: *Parameters used in experiment*

N	40	L_{min}	24 pixels
G	3000	L_{max}	56 pixels
E	80	A	35 deg.
k	7	(c, d)	(51, 17)
(m, n)	(5, 5)	p	3 %
(a, b)	(11, 7)	(s, t)	(31, 17)
F	1500	(e, f)	(7, 7)

Figure 4 shows a monitored image of the processing. The lower half shows images from the left and right cameras in color corresponding to the left and right halves. A tracking result is shown by overlay graphics. Although we don't need color information for eye detection and tracking, we use color cameras and a color display because displaying in color helps visual perception. The upper left is a monochrome image of the left camera image (below it) with the red component only. The eye detection and tracking process is applied to this (160 × 120) monochrome image.

A binarized difference image at the moment of eye detection is displayed in the center of the upper-right image. Located eye positions are marked with small double circles. A mirror symmetry test (step 12 of eye detection) is done on the small image (51 × 17) on the upper edge. As can be seen, the image is derotated so that the eyes are aligned horizontally. This image remains as it is at the moment of eye detection until the system loses the eyes. Other parts are updated frame-by-frame.

A small pattern (31 × 17) at the center corner is the current template of "Between-the-Eyes" extracted from the left image. A rectangular mark on the lower right image indicates that the search for "Between-the-Eye" was done using the template of "Between-the-Eyes" that was extracted from the left image. From these corresponding points of

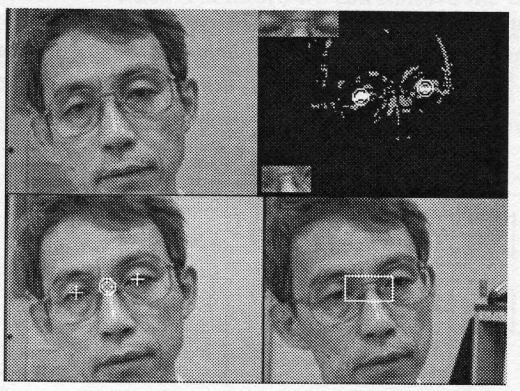

Figure 4: *Monitored image of processing.*

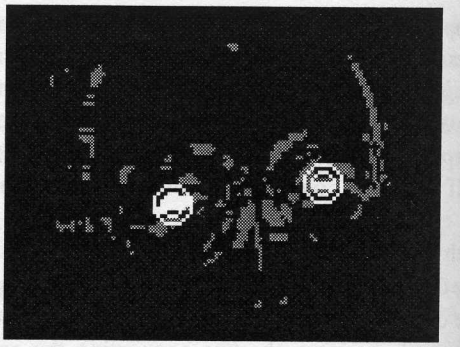

Figure 5: *Enlarged difference image from Fig. 4.*

"Between-the-Eye" in the left and right images, we can calculate the distance from camera to face by the binocular stereo principle.

Figure 5 is an enlarged binarized difference image of the upper-right part of Fig. 4. The diameters of the double circles indicating the detected eye positions are seven and eleven pixels. The gap between them is one pixel. Gray pixels are those deleted by the head motion cancellation or step 6 of the eye detection process. White pixels are those left through step 6, although white circles are overlayed there. Pixels that are deleted in step 4 are not left here. Figure 5 shows the effectiveness of head motion cancellation. The luminance changes caused by blinks are in very small regions. Without head motion cancellation, it is very difficult to distinguish them from the luminance changes caused by head motion.

The system runs at a rate of 30 frames/second, including the calculation of face distance done by binocular stereo. It robustly detects and tracks eyes. When it runs at 30 frames/second, eyes are often detected at the moment when they are partly closed. On the other hand, when it runs at 15 frames/second, eyes are detected at the moment when they are completely closed in almost all cases. Therefore, there seems to be an optimum frame rate for detecting blinks that is between 15 and 30 frames/second.

In the tracking process of the eyes, we determine their locations with a very simple criterion of "darkest position in the predicted search area." Therefore, it seems that the system loses eye locations when they blink and pupils are hidden. However, the tracking points stay between the upper

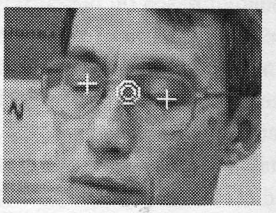

Figure 6: *Tracking points when eyes are closed.*

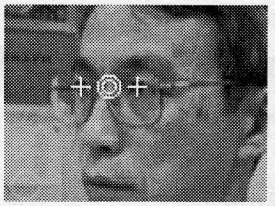

Figure 7: *Limit of face rotation for tracking eyes.*

and lower eyelids, as shown in Fig. 6, and when eyes are opened the points move on the pupils again.

Figure 7 shows the limit to face turning where our algorithm can still track the eyes. As far as pupils are visible, the system can track the eyes. But when the background is very dark and the eye search area includes a part of the background, the system mistracks the eyes in the background.

Similar mistracking occurs when one of the eyes gets very near to the hair in some head poses or when long hair drops down near the eyes. In these cases, the tracking points likely jump to the hair part. In some head poses, the end of an eyebrow and an eye corner become very near for some faces, causing the tracking point to move to the eyebrow part.

When we intentionally move both eyebrows up and down quickly, the system sometimes mistakes the eyebrows for the eyes.

6 Conclusions

We proposed new algorithms for detection and tracking of the eyes. For our application, we could take face images at a fairly large scale. Therefore, we could detect the eyes by blink detection based on the difference between successive images. To allow head movement during blink detection, we implemented a head movement cancellation algorithm, which works quite effectively. To prevent mistaking false candidates for the eyes, we applied simple geometric conditions and a pattern symmetry test at the midpoint of the candidates. For tracking the eyes, we used a template of "Between-the-Eyes" instead of eye patterns. After tracking "Between-the-Eyes," the eyes were again searched for in the areas relative to "Between-the-Eyes" as the small darkest regions. Then the template of "Between-the-Eyes" was updated based on the current eye positions.

Our algorithm uses only the small regions of the two eyes and that between them. Consequently, it is not affected by a beard or mustache and by most variations in hair style. Even when the lower half of the face is hidden, it works.

We implemented the algorithm on a PC with a Pentium III 866MHz CPU. It ran at the rate of 30 frames/second. It worked for several faces quite robustly.

As mentioned in the last part of the previous section, our prototype system still has some points that need to be improved. However, the system's CPU still has enough unused power to add additional process to increase the system's robustness. We will carry out work on implementing such additional process in the near future.

This research was supported in part by the Telecommunications Advancement Organization of Japan.

References

[1] L.-P. Bala, K. Talmi, and J. Liu. Automatic detection and tracking of faces and facial features in video sequencces. *1997 Picture Coding Symposium*, Sept. 1997, Berlin, Germany.

[2] J. L. Crowley and F. Berard. Multi-modal tracking of faces for video communications. *Proc. CVPR 97*, pages 640–645, 1997.

[3] E. Hjelmas and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, 2001.

[4] S. Kawato and J. Ohya. Real-time detection of nodding and head-shaking by directly detecting and tracking "between-eyes". *Proc. IEEE 4th Int. Conf. on Automatic Face and Gesture Recognition*, pages 40–45, 2000.

[5] Y. Matsumoto and A. Zelinsky. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement. *Proc. IEEE 4th Int. Conf. on Automatic Face and Gesture Recognition*, pages 499–504, 2000.