3D Face Tracking and Gaze Estimation Using a Monocular Camera

Tom Heyman[#1], Vincent Spruyt[*2], Alessandro Ledda[#3]

[#]*Faculty of Applied Engineering: Electronics-ICT, Artesis University College of Antwerp*
Paardenmarkt 92, B-2000 Antwerp, Belgium
[1]tom.heyman@ieee.org
[3]alessandro.ledda@artesis.be

[*]*Ghent University, TELIN, IPI, IBBT*
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
[2]v.spruyt@ieee.org

***Abstract*— Estimating a user's gaze direction, one of the main novel user interaction technologies, will eventually be used for numerous applications where current methods are becoming less effective. In this paper, a new method is presented for estimating the gaze direction using Canonical Correlation Analysis (CCA), which finds a linear relationship between two datasets defining the face pose and the corresponding facial appearance changes. Afterwards, iris tracking is performed by blob detection using a 4-connected component labeling algorithm. Finally, a gaze vector is calculated based on gathered eye properties. Results obtained from datasets and real-time input confirm the robustness of this method.**

I. Introduction

User-friendly applications are generally built with consideration to a user's intentions and interests. Ideally, such applications are able to adapt to different users. With the aid of computer vision a person's focus of attention can be detected without the need for physical contact, while employing this gathered data as an interaction method or to determine the user's regions of interest for further processing. Similar technologies are already available today for various purposes, for example heat mapping, ATM security and human-computer interfaces, yet all of these applications still introduce restrictions or require complex hardware setups.

A. Gaze estimation technologies

Several possible approaches using head-mounted [1] or multiple near-infrared [2] cameras have been put forward regarding gaze estimation. Eye tracking needs to be highly accurate if used for gaze estimation. Therefore, head-mounted cameras are the optimal solution when dealing with low-resolution images or large distances between the camera and user. However, such an approach is extremely intrusive and therefore not suitable for user friendly applications.

Using near-infrared illuminators offers great advantages above purely vision based eye trackers. Not only do they reduce the effect of different ambient light conditions, they also bring out the difference between the iris and pupil as well as the reflections from light sources on the iris, which can be used to track the current eye gaze direction as explained in [3]. However, infrared sensors are active devices which are not part of most people's day-to-day working environment and can introduce damage of the eyes [4]. Furthermore, infrared signals are easily disturbed by the presence of other infrared sources such as remote controls or sunlight.

In case of detecting a 3D gaze point, stereo-vision can be employed to accurately determine the gaze direction and gaze depth using tracking data from both eyes. Being able to detect this gaze allows for an accurate detection of the 3D point of focus as long as the user's gaze is directed towards a position between himself and the camera system.

B. Tracking face position

Different face tracking methods exist, for example, involving active contours [5], Active Shape Models (ASM) [6] or Active Appearance Models (AAM) [7], [8], [9]. While ASM are based solely on the shape of a person's face, AAM also includes texture or other appearance based features within this shape. They both use Principal Component Analysis (PCA) [10] for dimensionality reduction while training.

Methods using a single low resolution camera while allowing free head movement however, cannot solely depend on the position of both irises to estimate a user's gaze direction as the incoming frames are of such low quality they are nearly impossible to track. This accuracy problem is resolved by combining 3D face pose tracking and iris tracking, proposed by [11]. We have approached the 3D face pose and animation tracking by finding a linear relationship between face parameters and facial appearance changes. This is done using a Canonical Correlation Analysis (CCA) [12], similar to techniques described in [13], [14]. CCA based tracking can be compared to AAM as both methods find the relationship between a residual and pose parameters. However, our method creates an adjustable 3D face model which gives us the possibility to track the 3D pose parameters. We also compare the current face with a reference image to adjust the pose parameters, while AAM performs an analysis by synthesis: creating a face image by means of the learned AMM and compares this to the face in the actual frame in order to retrieve a best possible duplicate.

C. Our approach

By retrieving the face position using CCA, we are able to estimate the gaze direction with respect to the face position instead of the camera position, providing a better prediction. Also, a wider region of interest including the visible sclera of the eyes can then be used to estimate the gaze direction vector. As soon as the position and frontal direction of the face are known, the position of both irises can be extracted from the eye's region of interest by implementing a blob detection using a fast and efficient two-pass 4-connected component labeling algorithm [15]. Finally, the gaze angle with respect to the camera can be calculated from both face and eye rotations.

II. Face tracking

Firstly, the 3D pose and animation of the user's face is tracked by creating a frontal view and training different poses of this view using CCA [12]. This provides us with face poses relevant for future tracking of the eyes, as well as the position of the eyelids and eyebrows.

A. 3D face model

In this paper, the CANDIDE-3 face model [16], a parameterized face mask developed for model-based representations of human faces, is used. This model can be controlled by animation units and initialized by shape units and is represented as follows:

$$g = \bar{g} + S\tau_s + A\tau_a \tag{1}$$

with $\bar{g}$ being the standard CANDIDE-3 shape model, S and A the columns containing our shape units and animation units respectively, τ_s being 14 shape parameters able to reshape the wireframe to the most common head shapes and τ_a being the animation parameters to control facial movement. We have limited τ_a to five parameters relevant to our goal, being the inner and middle eyebrow movement points and the combined movement of the eyelids.

B. Initialization phase

During an initialization phase, the first frame is used as reference to manually adjust the face model wireframe to the user's face using 14 adjustable shape parameters. A 3D model is also created and shown rotating around its vertical axis to visualize the depth changes by wrapping this snapshot around the face model wireframe, as seen in Figure 1.

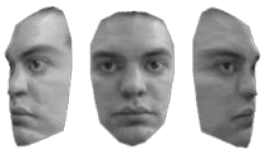

Fig. 1. Visualizing the textured 3D face model

C. Image patch

Once the face model has been initialized, an image patch will be created. Image patches will be created during both training and tracking phases, and represent a frontal view of the face, as well as two semi-profile side views. We create this 2D image patch by projecting three 3D face models on a 2D environment, which have been rotated -60°, 0° and 60° respectively. A simple color normalization method is used to increase robustness to lighting changes:

$$RGB = min(\frac{127.5\, RBG_t}{RGB_{avg}}, 255) \tag{2}$$

Where RGB_{avg} is a vector containing the averages of all pixels underneath the face model wireframe for each RGB channel at initialization, and RGB_t is a vector containing all pixels for each RGB channel of the current frame. The result is then clipped when exceeding 255. Afterwards, we convert this frame into a grayscale image. Slight differences between an original and a normalized frame are noticeable at the facial edges, which can be useful during facial rotation tracking as it creates slightly better results after performing a canonical correlation analysis.

The 2D reference image patches are finally created using the texture from the initialization phase, illustrated in Figure 2. These patches are important during training as this phase depends on the differences between image patches created from incoming frames and this reference image patch.

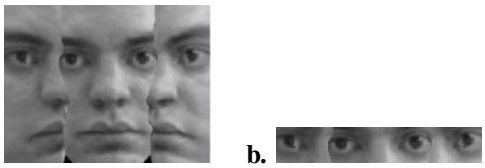

Fig. 2. 2D Reference image patch for pose tracker (a), and animation tracker (b).

D. Training phase

All possible face model rotations and translations are trained separately, as well as several rotational combinations and the five animation parameters mentioned in section 2.1. A single tracker was found to be far too inaccurate, therefore, two independent trackers have been implemented, training on the pose and animation states respectively. The different state vectors assigned to these trackers, with θ containing rotations and t translations, are:

$$b_{Pose} = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z]^T \tag{3}$$

$$b_{Animation} = [\tau_{a1}, \tau_{a2}, \tau_{a3}, \tau_{a4}, \tau_{a5}]^T \tag{4}$$

During the training phase, reference face textures from the initialization phase are wrapped around face models adjusted by b_{Pose} or $b_{Animation}$. While the first tracker is based on the full image patch which is resized to 88x75, the second tracker uses only a region of interest around the eyes, resized to

140x38. This setup improves robustness for tracking both the pose and the animation state differences. For practical reasons, we will from now on assume a single tracker using state vector:

$$b_{Training} = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z, \tau_{a1}, \tau_{a2}, \tau_{a3}, \tau_{a4}, \tau_{a5}]^T \quad (5)$$

Two datasets Q_1 and Q_2 are created during training. These datasets hold the normalized facial appearance variations and the variation in the state vector $\Delta b_{Training}$ respectively. By facial appearance variations x' we mean the difference between a created 2D image patch and the reference image patch made during initialization:

$$x' = (x_t - x_t^{(ref)}) \quad (6)$$

Currently, 535 different shape vectors are trained, out of which 329 are poses (rotations and translations) and 206 are different animation states. These values are sampled densely around the reference position b_0, and more sparsely when moving away from this position as illustrated in Figure 3.

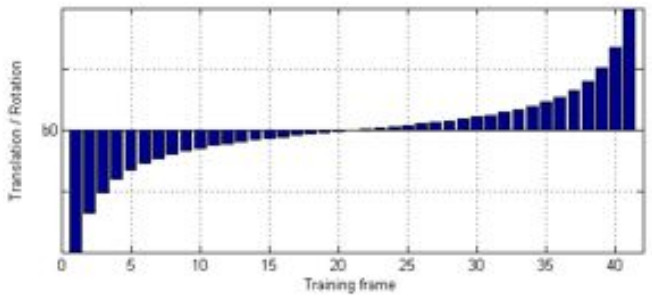

Fig. 3. Empirically chosen values for a parameter of $b_{Training}$

These values were obtained empirically and produced good results with only 8° variation of possible rotations and a fraction of possible translations trained. Changing these training shape vectors will result in a less accurate tracking result.

E. Canonical Correlation Analysis

After creating Q_1 and Q_2, a linear relationship between these sets can be found to create a motion model matrix G which will be used during tracking. By using CCA we try to maximize the correlation between these datasets, which are sometimes referred to as independent and dependent data respectively. To situate the mathematical problem, we map the centered data A_1 and A_2 to the direction vectors w_1 and w_2 and obtain two new vectors, called the canonical variates or scores:

$$z_1 = A_1 w_1 \quad (7)$$

$$z_2 = A_2 w_2 \quad (8)$$

The cross-correlation for estimating the degree to which two series are correlated is calculated, with μ_1 and μ_2 being the expected values, by:

$$\rho = \frac{(z_2 - \mu_2)(z_1 - \mu_1)}{\sqrt{(z_2 - \mu_2)^2}\sqrt{(z_1 - \mu_1)^2}} \quad (9)$$

However, because the data is centered, this formula can be simplified to:

$$\rho = \frac{z_2^T z_1}{\sqrt{z_2^T z_2}\sqrt{z_1^T z_1}} \quad (10)$$

This equation should now be maximized to retrieve the best canonical variates. As ρ should not be affected by rescaling z_1 or z_2, we introduce the constraints:

$$z_1^T z_1 = w_1^T A_1^T A_1 w_1 = w_1^T \Sigma_{11} w_1 = 1 \quad (11)$$

$$z_2^T z_2 = w_2^T A_2^T A_2 w_2 = w_2^T \Sigma_{22} w_2 = 1 \quad (12)$$

With Σ_{11} and Σ_{22} being covariance matrices. After writing this maximization problem in Lagrangian form as explained and solved in [12], two equations which govern CCA are found:

$$(\Sigma_{12}^T \Sigma_{22}^{-1} \Sigma_{12} - \rho^2 \Sigma_{22}) w_1 = 0 \quad (13)$$

$$(\Sigma_{12}^T \Sigma_{11}^{-1} \Sigma_{12} - \rho^2 \Sigma_{22}) w_2 = 0 \quad (14)$$

Given the eigenvectors for one of these equations, the eigenvectors for the other can be resolved. This generalized eigenvalue problem is solved using singular value decompositions [17] on known data matrices A_1 and A_2 as done in [12] and [14], which results in two canonical correlation basis vectors:

$$w_1 = V_1 D_1^{-1} U \quad (15)$$

$$w_2 = V D_2^{-1} V_2 \quad (16)$$

While searching for a maximized correlation we assume $\rho \approx 1$, and using equations 10, 11 and 12 we find

$$\begin{aligned} \|A_1 w_1 - A_2 w_2\| &= \sqrt{(A_1 w_1 - A_2 w_2)(A_1 w_1 - A_2 w_2)^T} \\ &= \sqrt{2 - 2 A_1 w_1 A_2^T w_2^T} \quad (17) \\ &\Downarrow \\ \|A_1 w_1 - A_2 w_2\|^2 &= 2(1 - \rho) \quad (18) \end{aligned}$$

the canonical variates are equal to each other:

$$A_1 w_1 \approx A_2 w_2 \Rightarrow z_1 \approx z_2 \quad (19)$$

As A_1 represents image patch vectors $(x_t - x_t^{(ref)})$ and A_2 the state vectors Δb_t, this can be written as:

$$(x_t - x_t^{(ref)}) w_1 \approx \Delta b_t w_2 \quad (20)$$

Our motion model G can then be retrieved:

$$\begin{aligned} \Delta b_t &= (x_t - x_t^{(ref)}) w_1 w_2^{-1} \\ &= (x_t - x_t^{(ref)}) V_1 D_1^{-1} U (V D_2^{-1} V_2)^{-1} \\ &= (x_t - x_t^{(ref)}) G \quad (21) \\ &\Downarrow \\ G &= V_1 D_1^{-1} U V^T D_2 V_2^T \quad (22) \end{aligned}$$

This model can now be used while tracking to obtain the state variation Δb_t, which represents the motion prediction of the model.

F. Tracking phase

While tracking a user's face, initialization parameters b_0 are used as starting position. An image patch vector x_t containing the normalized pixel values is then created using the texture underneath the model's position. The state vector b_t for the current frame is calculated by:

$$b = b_{t-1} + G(x_t - x_t^{ref}) \tag{23}$$

$$b_t = [10 \cdot b_{0..2}, 4 \cdot b_{3..10}] \tag{24}$$

With b_{t-1} being the last known state vector, x_t the current image patch vector and x_t^{ref} the reference image patch vector created at initialization. Mind that this state vector estimate is multiplied by 10 for rotations and 4 for translations and animation states. These empirically chosen adjustments speed up the tracking with minimum loss of accuracy. The whole process is iterated four times for every incoming frame, ensuring a good 3D representation of the model's position. Finally, at each new frame the reference image patch is slightly adjusted to overcome lighting changes, which is a similar approach as in [14];

$$x_{t+1}^{ref} = \alpha x_t^{ref} + (1 - \alpha)x_t \tag{25}$$

With x_{t+1}^{ref} and x_t^{ref} being respectively the new and old reference image patches, x_t the image patch from the current frame and α= 0.99, proven by [13] to provide the best results. Keep in mind we have trained and tracked the pose and the animation states separately.

III. Eye tracking

As explained above, an image patch is created which recreates a frontal view from any out-of-plane facial rotations, along with two side-views, by wrapping an incoming frame's region of interest on a 3D face model.

The eye region is now extracted from this image patch and converted to a binary image using a predefined threshold. This threshold is the average grayscale value from the irises' region of the face during our face model initialization. All values above this threshold are ignored while darker colors are treated as possible blobs, considering the iris and pupil usually having the darkest gray levels in the eye region. Keep in mind this region is restricted to the sclera of each eye, minimizing false positives while becoming more dependent on a robust face tracker.

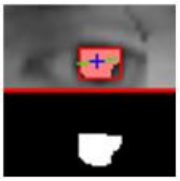

Fig. 4. Grayscale and binary regions of interest during eye tracking

Finally, a fast and simple two-pass Connected Component Labeling (CCL) algorithm [15] for finding 4-connected components is used, meaning every pixel with neighborhood pixels having coordinates $(x \pm 1, y)$ or $(x, y \pm 1)$ will receive the same label, giving every tracked blob a unique label. Afterwards, we calculate the centroid of the largest blob and use this position as estimated center of the iris, seen in Figure 4.

On a side note, implementations of a Hough transform method [18] or normal canny edge detection [19][20] for finding circles in both the grayscale image patch as the binary image showing only the darker colors, have proven not to be robust enough for usage in this implementation. The highly variable contour shapes of the iris or detected blob cause these algorithms to fail.

IV. Eye gaze estimation

As the head position and direction are known, as well as the irises' position with respect to the face position, the gaze direction can be estimated by regarding the eyeballs as spheres similar to [21]. By doing this, several parameters are needed to result in a gaze direction vector:

- Position of the iris with respect to the center of the eyeball
- Radius of the eyeball

The eyeball radius can be approximated by the radius of a 2D projection of the eyes during our initialization phase, which is retrieved from the face model adjustments made by the user. Afterwards, the eyeball center is then easily found. The position of the iris is retrieved by introducing a blob detection algorithm on the current image patch as explained in Section 3. Finally, the calculated angle θ between the iris center and the eyeball center, for both horizontal as well as vertical movement, defines the orientation of the gaze direction vector as seen in Figure 5.

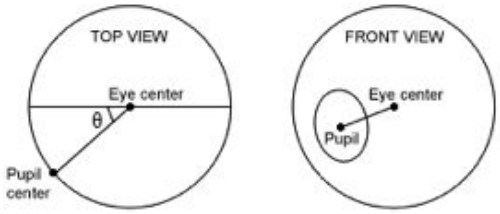

Fig. 5. Visual representation of θ on an eye

V. Results

Two separate datasets by Asteriadis et al. [22] and Weidenbacher et al. [23] have been used for testing purposes, which enables comparison of our method with future gaze tracking algorithms.

A. Implementation details

Tests were performed on an Intel Core2 Duo P9500 at 2.53GHz using the Logitech 1.3 MP Webcam C300 which was placed in front of the user. The project was written in C++, including OpenGL libraries for enabling a 3D environment and OpenCV 2.0 libraries for image processing. An image resolution of 320x240 has been used throughout these tests, while enlarging the output frames to 640x480. The current

non-optimized implementation runs at 256 ms per frame, averaged to 4 fps. However, because a tracking iteration of four has been included, we can assume that tracking can be done at a much higher frame rate if we discard the iteration. This action would have a small negative impact on the tracking accuracy.

The duration of training our method depends on the amount of poses trained. Currently, 328 positions and 205 animation states are trained, creating 553 different image patches and taking 35.5 seconds to complete while including the computation time for retrieving a motion model using canonical correlation analysis.

B. HPEG Dataset

The Natural Head Pose and Eye Gaze (HPEG) dataset presented by Asteriadis et al. [22] was recorded using a monocular camera, while three LED's placed in front of the face are used for extraction of the face position ground truth. Eye gaze directionality, with respect to the head pose, has been presented by three eye gaze classes being: looking forward, looking to the left and to the right. Eye gaze ground truth is given by appointing video time segments to the appropriate eye gaze class.

Although the gaze estimation can not accurately be compared with the dataset, it still provides a good perspective of our method's robustness and, although this is not our major interest, we were able to compare our 3D face tracking with the ground truth head positions.

Our method has been applied to three randomly chosen video sequences. Because this dataset is restricted to yaw movements, the graphs found in Figure 6, 7 and 8 represent only the yaw rotation of the head and eyes during each frame of their sequence, while a frame of each eye position has been included.

Figure 6 shows a correct tracking during the whole video sequence.

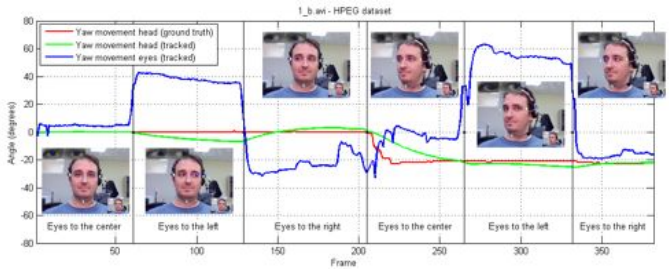

Fig. 6. Graphical representation of face/eye yaw movement using video sequence 1_b.avi

As seen in Figure 7, during the test our face tracking algorithm has accuracy problems while tracking the user's head movement in video 7_b.avi, due to the slow position adjustment of the face mask in this sequence.

While testing video 9_b.avi we notice the eye gaze tracking becomes inaccurate for eyes positioned looking straight forward, from frames 210 to 260 as seen in Figure 8.

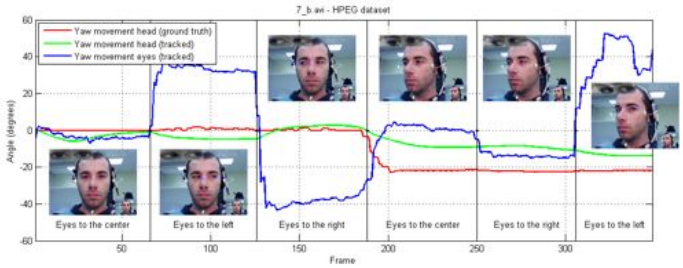

Fig. 7. Graphical representation of face/eye yaw movement using video sequence 7_b.avi

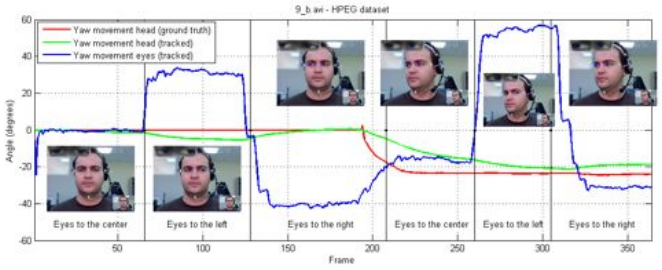

Fig. 8. Graphical representation of face/eye yaw movement using video sequence 9_b.avi

C. uulmHPG Dataset

This data consists out of several image sequences of different subjects including faces in various combinations of head pose and eye gaze, and can give us a more precise accuracy measurement. We have documented the mean error in degrees for six unique tracking events of three image sequences using two slightly different initialization parameters for each set.

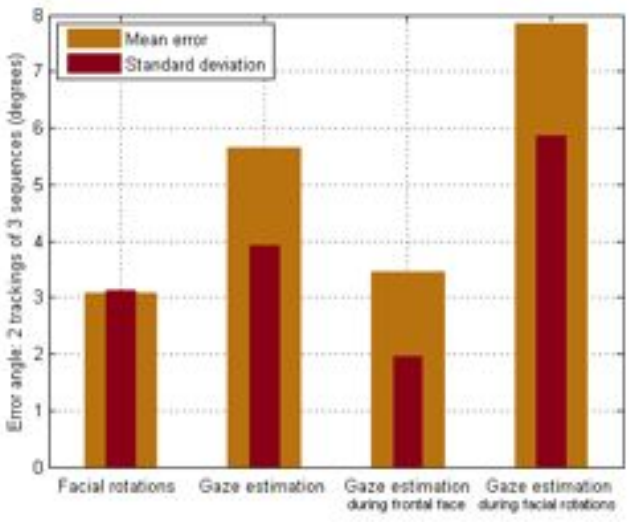

Fig. 9. Mean error and stdev measurements of 3 image sequences

Head rotations have been restricted to [-30°,30°] and gaze vector rotations to [-40°,40°] with respect to the camera.

Figure 9 shows the average error angle of our algorithm during three image sequences (Head 3,12 and 16), which lies

around 3.08° for facial rotation tracking, and 5.64° for our gaze estimation vector. The latter however, proves to be more robust during a frontal face view and becomes less accurate during facial rotations.

D. Real-time webcam results

Because former results have been obtained from ideal dataset frame sequences, a real-time webcam input has been tested while the user made natural head and gaze movements. Several frames during this tracking are shown in Figure 10 and 11.

Fig. 10. Gaze tracking during constant, uniform lighting conditions

Fig. 11. Face and gaze tracking during non-constant, non-uniform lighting conditions

VI. Conclusion

Our paper proposed a novel solution for tracking the eye gaze vector, making it possible to determine a gaze point. This was done by modifying an existing CCA based face tracking algorithm and include a simple yet effective blob tracking method to retrieve robust results concerning eye gaze estimation. Finally, video sequences including ground truth and real-time input data have been used to provide an accuracy measurement of the system, which confirmed the robustness considering low-resolution video input.

VII. Future work

Several methods can be improved concerning the eye gaze estimation. Currently, a sphere is used to represent the eye during the eye gaze vector estimation, although a spheroid would give a more accurate representation. Based on these mathematical shapes the gaze direction could be calculated, although other techniques, for example, a calibration system as in [24], could prove to be more accurate.

Although several facial animation shapes are currently being tracked, i.e. the eyelids and eyebrows, they have no direct effect on the eye gaze direction. A more complex system can be built, using the same trackers yet trained for more animation units to find different facial expressions and incorporate those expressions into a possible eye gaze direction.

References

[1] D. Young, H. Tunley, and R. Samuels, *Specialised hough transform and active contour methods for real-time eye tracking*. University of Sussex, School of Cognitive and Computing Science, 1995.

[2] Z. Zhu and Q. Ji, "Robust real-time eye detection and tracking under variable lighting conditions and various face orientations," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 124–154, 2005.

[3] ——, "Novel eye gaze tracking techniques under natural head movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 12, pp. 2246–2260, 2007.

[4] T. van den Berg and H. Spekreijse, "Near infrared light absorption in the human eye media," *Vision research*, vol. 37, no. 2, pp. 249–253, 1997.

[5] G. Panin and A. Knoll, "Real-Time 3D Face Tracking with Mutual Information and Active Contours," *Advances in Visual Computing*, pp. 1–12.

[6] S. Milborrow and F. Nicolls, "Locating facial features with an extended active shape model," *Computer Vision–ECCV 2008*, pp. 504–513.

[7] J. Zhu, S. Hoi, E. Yau, and M. Lyu, "Automatic 3d face modeling using 2d active appearance models," in *Proc. 13th Pacific Conf. Computer Graphics and Applications*. Citeseer, 2005, pp. 133–135.

[8] F. Dornaika and J. Orozco, "Real time 3D face and facial feature tracking," *Journal of Real-Time Image Processing*, vol. 2, no. 1, pp. 35–44, 2007.

[9] S. Ayala-Raggi, L. Altamirano-Robles, and J. Cruz-Enriquez, "Interpreting Face Images by Fitting a Fast Illumination-Based 3D Active Appearance Model," *Computer Vision/Computer Graphics CollaborationTechniques*, pp. 368–379.

[10] I. Jolliffe, *Principal component analysis*. Springer verlag, 2002.

[11] J. Heinzmann and A. Zelinsky, "3-D facial pose and gaze point estimation using a robust real-time tracking paradigm," in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. IEEE, 2002, pp. 142–147.

[12] D. Weenink, "Canonical correlation analysis," in *Proc. Institute of Phonetic Sciences University of Amsterdam*, vol. 25, 2003, pp. 81–99.

[13] J. Alonso and Y. Zepeda, "A linear estimation method for 3D pose and facial animation tracking." 2008.

[14] ——, "A linear estimation of the face's tridimensional pose and facial expressions," PhD Thesis, l'cole Nationale Suprieure des Tlcommunications, 2008.

[15] H. Works, C. Variants, and I. Experimentation, "Connected Components Labeling."

[16] J. Ahlberg, "Candide-3: an updated parameterised face," 2001.

[17] S. Oh, "A Note on Singular Value Decomposition," 2005.

[18] D. Ballard, "Generalizing the Hough transform to detect arbitrary shapes* 1," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[19] J. Wang, E. Sung, and R. Venkateswarlu, "Eye gaze estimation from a single image of one eye," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 2. Citeseer, 2003.

[20] V. Vezhnevets and A. Degtiareva, "Robust and accurate eye contour extraction," in *Proc. Graphicon*. Citeseer, 2003, pp. 81–84.

[21] Y. Matsumoto, T. Ogasawara, and A. Zelinsky, "Behavior recognition based on head pose and gaze direction measurement," in *IEEE International Conference on Intelligent Robots and Systems*, 2000.

[22] S. Asteriadis, D. Soufleros, K. Karpouzis, and S. Kollias, "A natural head pose and eye gaze dataset," in *Proceedings of the International Workshop on Affective-Aware Virtual Agents and Social Robots*. ACM, 2009, pp. 1–4.

[23] U. Weidenbacher, G. Layher, P. Strauss, and H. Neumann, "A comprehensive head pose and gaze database," in *3rd IET International Conference on Intelligent Environments*.

[24] T. Ishikawa, S. Baker, I. Matthews, and T. Kanade, "Passive driver gaze tracking with active appearance models," in *Proceedings of the 11th World Congress on Intelligent Transportation Systems*. Citeseer, 2004.