PERGAMON

Pattern Recognition 35 (2002) 2559–2569

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Gaze position detection by computing the three dimensional facial positions and motions

Kang Ryoung Park[a,*], Jeong Jun Lee[b], Jaihie Kim[b]

[a] *Digital Vision Group, Innovation Center, LG Electronics Institute of Technology, 16 Woomyeon-Dong, Seocho-Gu, Seoul 137-724, Republic of Korea*

[b] *Computer Vision Laboratory, Department of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Republic of Korea*

Received 25 May 2001; accepted 5 October 2001

Abstract

Gaze detection is to locate the position on a monitor screen where a user is looking. In our work, we implemented it by a computer vision system with a single camera above the monitor. We computed the initial three dimensional positions of facial features (both eyes, nostrils and lip comers) by a camera calibration and a parameter estimation algorithm, and also computed the moved 3D positions of those features by 3D rotation and translation estimations and affine transform. Finally, the gaze position on the monitor is computed from the normal vector of the plane determined by those moved 3D positions of features. Gaze detection error was 5.11 cm (RMS error) when the distance between the user and the monitor is 50–70 cm and a 19 in monitor is used. © 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Gaze detection; 3D facial position; 3D facial motion; Camera calibration; Parameter estimation; Rotation and translation estimation

1. Introduction

Gaze detection is to locate the position on a monitor screen where a user is looking, and it is applicable to many areas that users cannot use their hands for communicating with computers. These include view control in 3D simulation program, man–machine interface for handicapped person and intelligent interface for process control system [1], etc. In addition, it can be applicable to the field of business-to-consumer e-commerce. For example, in case a user can just gaze/click to select item during e-commerce navigation, then he will feel more comfortable than with traditional mouse. In our work, we implement it with a computer vision system setting a single camera above a monitor screen, and the user moves (rotates and translates) his face to gaze at different positions on the monitor. Here, the translation means the X, Y, Z movements of facial axis without any facial rotation. At this moment, the user is requested not to move the pupils of his eyes when he gazes at a different position on the monitor screen. We are currently working to relax this restriction. Previous studies were mostly focused on 3D facial motion estimation (rotations of yaw, pitch, roll and X, Y, Z translations) [2,3], and researches on the detection of a user's gaze direction have been vigorously investigated recently [4–11].[1] Ohmura and Ballard et al. [4,5] compute a gaze direction vector by the successive approximation method assuming that the 3D distances between facial features are not changed with facial rotation and translation. However, their method has the disadvantages that the depth between camera plane and feature points in the initial frame have to be measured manually and it takes much time (over 1 min) to compute the gaze

* Corresponding author. Tel.: +82-2-526-4162; fax: +82-2-526-4165.
E-mail address: parkgr@seraph.yonsei.ac.kr (K.R. Park).

[1] Eyemark Recorder Model EMR-NC, NAC Image Technology Cooperation.

direction vector. Gee et al. [6] and Heinzmann et al. [7] compute the gaze direction from affine projection algorithm and especially Heinzman et al. improve that algorithm using the temporal continuity of 3D rotation and the weights representing feature detection performances. However, they only compute the gaze direction vector whose origin is located between the eyes in the face coordinate and do not obtain the gaze position on a monitor. In addition, if 3D rotations and translations of face happen simultaneously, they cannot estimate the accurate 3D motion due to the increase of complexity of least-squares fitting algorithm, which requires much processing time. Rikert et al. [8] compute the gaze point on a monitor using morphable models which are trained by a neural network, but their method has the constraints that the distance between a face and the monitor must be kept same for all training and testing procedures and it can be cumbersome to user. The methods of Tomono, Muneo et al. [9,10] handle changes in both head movement and eye rotation using the stereo camera and expensive InfraRed (IR) light. In their method, a pair of glasses having marking points is required to detect facial features, which can give somewhat inconvenience to a user. So, in order to solve such problems of previous researches, this paper describes a new method for detecting the gaze position on a monitor by a single camera without wearing device and any prior knowledge (for example, the distance between a face and the monitor, etc.) by estimating the initial 3D positions and the 3D motion of feature points.

2. Locating facial region and feature points

In order to detect gaze position on a monitor, we firstly locate a user's face region and facial features (both eyes, nostrils and lip corners) in an input image. The facial region is detected by both difference image and facial color image. RGB pixels in the facial color image are transformed into YIQ pixels in order to eliminate illumination variation (Y) and the object which belongs to a predetermined I-value range is determined as the facial region. Then the positions of both eyes, nostrils, and lip corners can be detected by anthropometric constraints in a face and horizontal, vertical histograms as shown in Fig. 1.

In order to reduce processing time, we do not repeat above procedures, but only detect all the features in current search windows determined from previous feature positions in succeeding images [12], until the detected feature positions satisfy the geometric constraints in the face region. The reason for using both eyes, nostrils and lip corners as clues for gaze detection is that they can be detected more easily than any other feature points (for example, ears, cheeks and nose tip). The reason that the nostril detections have bigger errors than others is that the shadow regions near the nose can be frequently misrecognized as nostrils. The problem that black points of eyes are disappeared in case the user wears glasses due to reflection is somewhat solved using p-tile threshold method [13] and the user's clothes do not influence the feature detection accuracy. In the future, we .plan to improve the feature detection accuracy using some feature relocation algorithm [14]. From the detected feature positions, we compute 18 feature values ($FV_1 \ldots FV_{18}$ as shown in Fig. 2) for estimating 3D facial rotation and translation.

- When gazing at a monitor center
  P_1 (left eye: x_1, y_1), P_2 (right eye: x_2, y_2), P_3 (the center of two nostrils: x_3, y_3)
  P_4 (left lip corner: x_4, y_4), P_5 (right lip corner: x_5, y_5),
  P_6 (facial center: Mx_1, My_1)
- When gazing at some other point
  P'_1 (left eye: x'_1, y'_1), P'_2 (right eye: x'_2, y'_2), P'_3 (the center of two nostrils: x'_3, y'_3),
  P'_4 (left lip corner: x'_4, y'_4), P'_5 (right lip corner: x'_5, y'_5),
  P'_6 (facial: Mx_2, My_2)

FV_{1-5}: $x'_i - x_i - (Mx_2 - Mx_1)$ $(i = 1, 2, \ldots, 5)$,

FV_{6-10}: $y'_i - y_i - (My_2 - My_1)$ $(i = 1, 2, \ldots, 5)$,

FV_{11}: $S(\Delta P'_1 P'_2 P'_3) - S(\Delta P_1 P_2 P_3)$,

FV_{12}: $S(\Delta P'_3 P'_4 P'_5) - S(\Delta P_3 P_4 P_5)$,

FV_{13}: $\{(x'_1 + x'_4)/2 - x'_3\} - \{(x_1 + x_4)/2 - x_3\}$,

FV_{14}: $\{(x'_3 - (x'_2 + x'_5)/2\} - \{x_3 + (x_2 + x_5)/2\}$,

FV_{15}: $\{(y'_4 + y'_5)/2 - y'_3\} - \{(y_4 + y_5)/2 - y_3\}$,

FV_{16}: $\{y'_3 + (y'_1 + y'_2)/2\} - \{y_3 - (y_1 + y_2)/2\}$,

FV_{17}: $Mx_2 - Mx_1$,

FV_{18}: $My_2 - My_1$.

In order to compute 3D facial translation, we use the 2D movement of facial center (FV_{17}, FV_{18} as shown in Fig. 2). However, because we obtain a rough facial region by both difference image and facial color image, we use edge operation in order to detect an accurate facial region and facial center as shown in Fig. 3.

From an input image (Fig. 3(a)), we obtain edge image by two-directional sobel edge operator (Fig. 3(b)) and there is no edge component inside facial region if we use high edge threshold (Fig. 3(c)). Because we track the facial features in this moment, we can locate facial contour and face center by searching for outer contour from detected eye positions (Fig. 3(d),(e)).

3. Estimating the initial 3D positions of facial feature points

After feature detection, we take four steps in order to compute a gaze position on a monitor as shown in Fig. 4.

In the first step, when a user gazes at three known positions on a monitor, the 3D positions of initial feature points

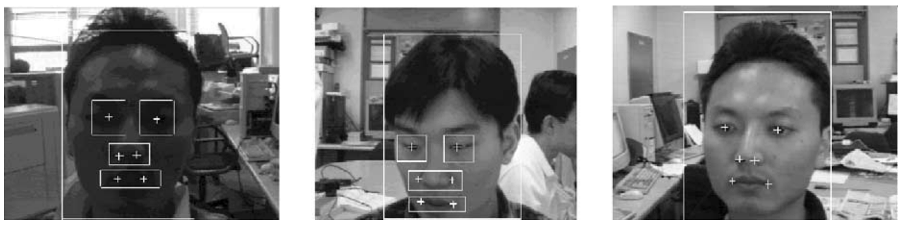


Fig. 1. The detected facial regions, facial feature points.

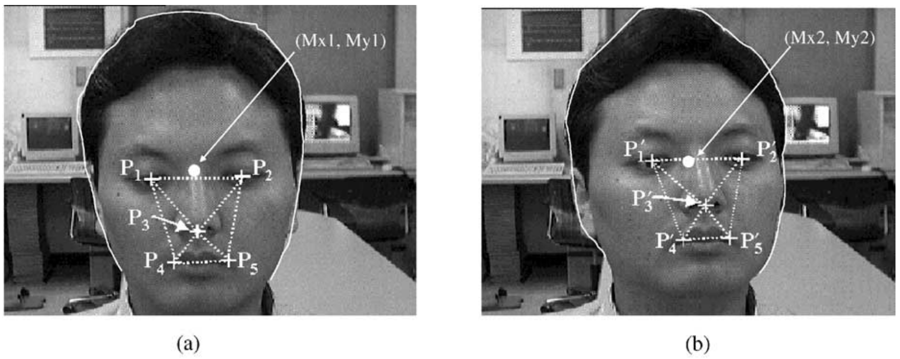


Fig. 2. The 18 feature values for 3D rotation and translation estimation: (a) Gazing at monitor center; (b) gazing at some other point on a monitor.

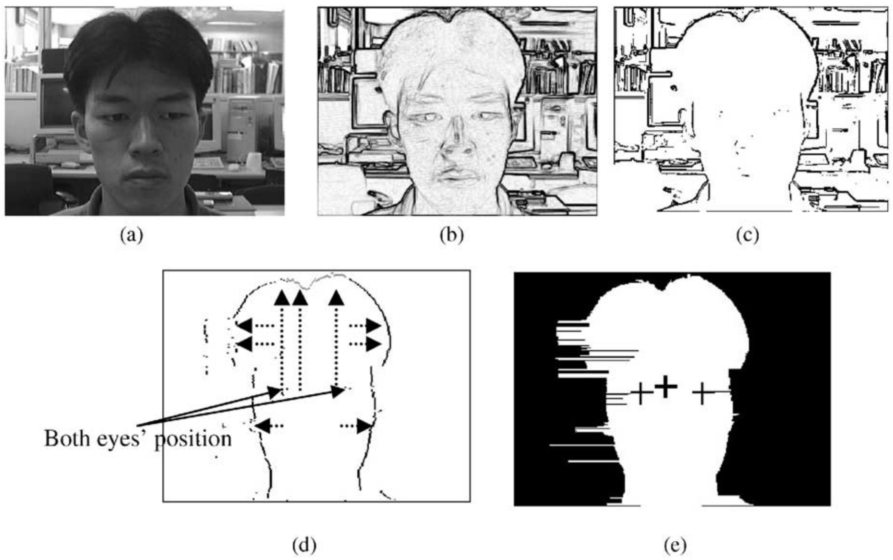


Fig. 3. Facial center detection by edge operation.

are computed automatically. In the second and third step, when the user moves (rotates and translates) his head in order to gaze at one position on a monitor, the moved 3D positions of those features can be computed from 3D motion estimation. In the last step, one facial plane is determined from the moved 3D positions of those features and the normal vector of the plane represents a gaze vector. Then, the gaze position on a monitor is the intersection point between the monitor plane and the gaze vector. Here, we assume that the eyeball in eye region and lip corner movements caused by mouth opening/closing do not happen find they will be considered in the future.

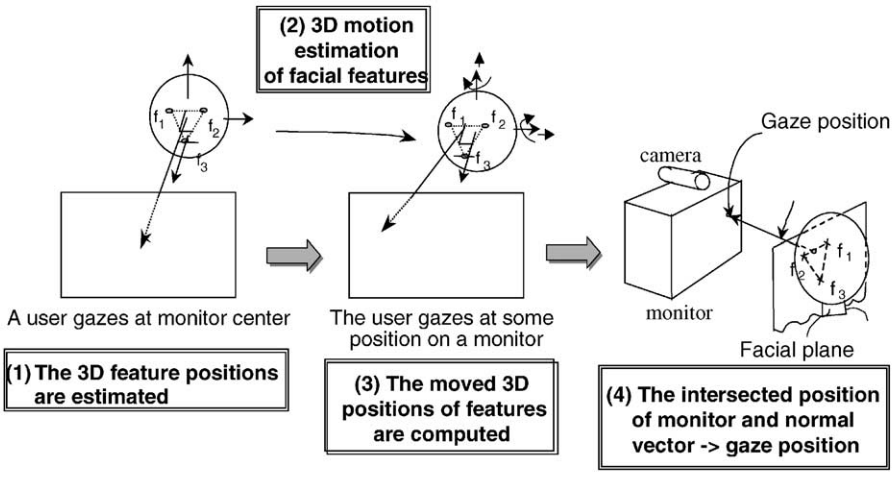

(2) 3D motion estimation of facial features
f1
f2
f3
camera
Gaze position
monitor
Facial plane
A user gazes at monitor center
The user gazes at some position on a monitor
(1) The 3D feature positions are estimated
(3) The moved 3D positions of features are computed
(4) The intersected position of monitor and normal vector -> gaze position

Fig. 4. Four steps in order to compute a gaze position on a monitor.

Now, we explain the first step. When a user gazes at one position on a monitor, the complicated transformation matrices of “a three-view” solutions (monitor, camera and face coordinate system) are required as shown in Fig. 5(b). That is why the facial features move (rotate and translate) in face coordinate and those positions are detected in camera coordinate when the user gazes at a position in monitor coordinate. In order to estimate the 3D positions of initial facial features in face and monitor coordinate, we use the information of 2D projected feature positions detected when a user gazes at three positions on a monitor, which positions are known in advance.

In detail, when a user gazes at two positions ($(0,0,0)$ and $(x_m,0,0)$) in monitor coordinate as shown in Fig. 5(a), the 3D position (x_0,y_0,z_0) of one feature point (for example, the center of both eyes) is moved into (x_i,y_i,z_i) like Eq. (1).

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \qquad \beta - a\tan\left(\frac{x_m}{K}\right). \tag{1}$$

The positions of (x_0,y_0,z_0) and (x_i,y_i,z_i) are defined in face coordinate system and we transform those into the points $((x_{0_w},y_{0_w},z_{0_w})$ and $(x_{i_w},y_{i_w},z_{i_w}))$ of monitor coordinate system according to Eq. (2).

$$\begin{aligned} x_{0_w} &= x_0, \quad x_{i_w} = x_i, \\ y_{0_w} &= y_0 + Y_t, \quad y_{i_w} = y_i + Y_t, \\ z_{0_w} &= -z_0 + K, \quad z_{i_w} = -z_i + K, \end{aligned} \tag{2}$$

where K denotes the Z-axis distance between face and monitor coordinate as shown in Fig. 5(a), Y_t denotes the Y-axis distance between face and monitor coordinate as shown in Fig. 5(b). In Fig. 5(b), if we assume camera tilt ($\alpha \neq 0$), swing angle ($\theta = 0$) and the translation ($T_x = 0,\ T_y,\ T_z \neq 0$), we obtain a transformation matrix between 2D positions in

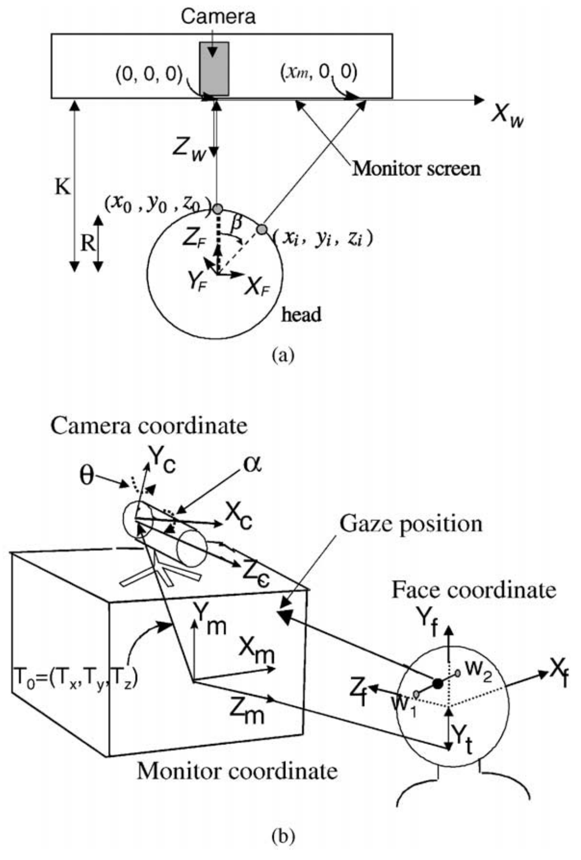


Fig. 5. Face, camera and monitor models for detecting gaze position.

camera coordinate and 3D positions in monitor coordinate system like Eq. (3) [15].

$$\boldsymbol{c}_i = \boldsymbol{P} \cdot \boldsymbol{R}_\alpha \cdot \boldsymbol{T}_0 \cdot \boldsymbol{w}_i, \tag{3}$$

$$\boldsymbol{T}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -T_y \\ 0 & 0 & 1 & -T_x \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{R}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \dfrac{-1}{f} & 1 \end{bmatrix},$$

where f is focal length, c_i denotes the homogeneous form of 2D positions in camera coordinate ($[x_i, y_i, z_i, 1]^{\mathrm{T}}$), w_i denotes the homogeneous form of 3D positions in monitor coordinate ($[x_{iw}, y_{iw}, z_{iw}, 1]^{\mathrm{T}}$). Here, we use a basic perspective camera model. From Eqs. (2) and (3), the x positions in 2D projected feature positions (x_0, y_0, f) and (x_i, y_i, f) are like Eqs. (4) and (5).

$$x_0 = \frac{f \cdot x_{0_W}}{(y_{0_W} \cdot \sin\alpha - z_{0_W} \cdot \cos\alpha - T_y \cdot \sin\alpha + T_z \cdot \cos\alpha + f)}$$
$$= \frac{f \cdot x_0}{((y_0 + Y_t) \cdot \sin\alpha - (-z_0 + K) \cdot \cos\alpha - T_y \cdot \sin\alpha + T_z \cdot \cos\alpha + f)}, \tag{4}$$

$$x_i = \frac{f \cdot x_{i_W}}{(y_{i_W} \cdot \sin\alpha - z_{i_W} \cdot \cos\alpha - T_y \cdot \sin\alpha + T_z \cdot \cos\alpha + f)}$$
$$= \frac{f \cdot \sin\beta \cdot z_0}{((y_0 + Y_t) \cdot \sin\alpha - (-\cos\beta \cdot z_0 + K) \cdot \cos\alpha - T_y \cdot \sin\alpha + T_z \cdot \cos\alpha + f)}. \tag{5}$$

Now, when a user gazes at 2 positions ((0,0,0) and $(x_m, 0, 0)$) in monitor coordinate, we can obtain the 2D distance between x_0 and x_i ($d = x_i - x_0$) in camera coordinate like Eq. (6).

$$d_x = x_i - x_0 = \frac{f \cdot \sin[a\tan(x_m/K)] \cdot R}{((y_0 + Y_t) \cdot \sin\alpha - (-\cos[a\tan(x_m/K)] \cdot (R + K)) \cdot \cos\alpha - T_y \cdot \sin\alpha + T_z \cdot \cos\alpha + f)}, \tag{6}$$

where f denotes camera focal length, (T_y, T_z) denotes the translation vectors between the monitor and the camera coordinate, α denotes the camera tilt angle, R denotes the facial radius, K denotes the Z-axis distance between the monitor and the face coordinate, y_0 denotes the Y-axis position of feature point in face coordinate, and Y_t denotes the Y-axis distance between the monitor and face coordinate. In Eq. (6), since the parameters (T_y, T_z, α, f) are not changed after initial camera setup, we compute them by typical camera calibration method [15]. To do that, we use several black points dotted on the calibration panel as shown in Fig. 6 and Eq. (3). From that, we obtain calibration parameters ($f = 0.776$ cm, $T_y = 25$ cm, $T_z = 0.6$ cm, $\alpha = -11.76^\circ$, the horizontal and vertical distance between CCD elements are 0.0015 cm/pixel). For more accurate results, lens distortion must be calibrated also, but we will consider it in the future. The experimental results show the average calibration error of 0.08 cm (X-axis) and 0.16 cm (Y-axis) and the error increases in case of using calibration points near the 2D image boundary rather than image center due to lens distortion [13]. Here, the calibration error is calculated from the differences between the real 3D positions of calibration points (they are measured by 3D position tracker sensor) and the calculated 3D positions from the calibration parameters.

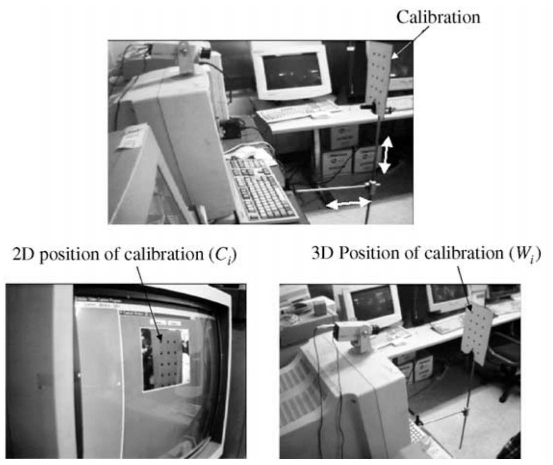

Calibration
2D position of calibration (C_i)
3D Position of calibration (W_i)

Fig. 6. Camera calibration.

Now, we estimate residual unknown parameters (K, K, y_0, Y_t) which are changed according to the user's sitting pose and facial size. For that, we use six pairs of (d, x_m) (two pairs for eye center, two pairs for nostril center and two pairs for mouth center) which are obtained while a user gazes at three positions on a monitor which we know in advance and parameter estimation algorithm. We compare the performance of three parameter estimation algorithms (Gauss–Newton method, steepest descent method and Davidon–Fletcher–Powell method [16]) and use one algorithm (Davidon–Fletcher–Powell method) that shows the best accuracy of parameter estimation. In order to make the convergence of Davidon–Fletcher–Powell method faster, we obtain average y_0, Y_t, R from several person's data by 3D position tracker sensor and use them for the initial values for Davidon–Fletcher–Powell method ($y_0 = 2.0$ cm, $Y_t = 4.6$ cm, $R = 7$ cm). In addition, because the distances between monitor and most users ($Z = K - R$) are almost from 50 to 70 cm, we use 57 cm as initial K of Davidon–Fletcher–Powell method.

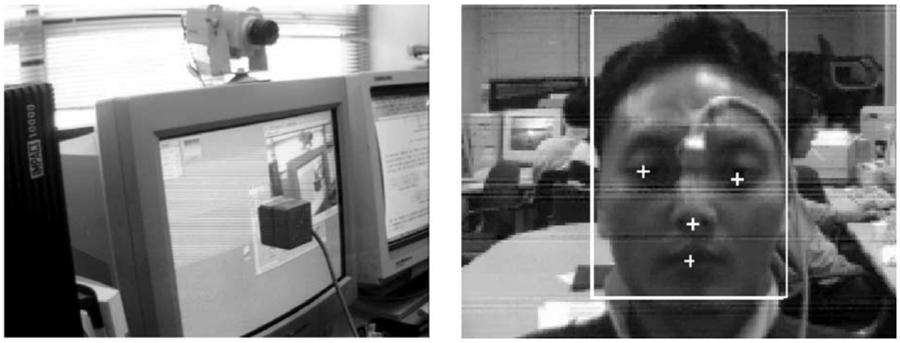


Fig. 7. The parameter estimation error is calculated from the real 3D feature positions and the estimated 3D feature positions.

As experimental results, we can obtain the estimated parameters (for example, the estimation results of one test person are $y_0 = 3.01$ cm, $Y_t = 7.4$ cm, $R = 9.06$ cm, $K = 58.2$ cm, iteration number is 620 (0.9 s in Pentium-Pro 200 MHz) and the 3D depth of feature position ($Z = K - R$). Then, we can compute the 3D feature positions in monitor coordinate by both parameters (T_y, T_z, α, f, Z) and Eq. (7) which is obtained from Eq. (3).

$$\boldsymbol{w}_i = [\boldsymbol{P} \cdot \boldsymbol{R}_\alpha \cdot \boldsymbol{T}_0]^{-1}\boldsymbol{c}_i. \tag{7}$$

From that, experimental results show that the average RMS error between the real 3D feature positions (measured by 3D position tracker sensor) and the estimated 3D feature positions like Fig. 7 is 2.79 cm (1.501 cm in X-axis, 1.601 cm in Y-axis and 1.723 cm in Z-axis) for 20 person data which were used for testing the feature detection performance.

However, this error is accumulated on the 3D motion estimation error (which is shown in the next section) and becomes the increase factor of the final gaze position error on a monitor. So, in order to reduce the 3D depth and the 3D feature position estimation errors, we compute the normal vector of a facial plane composed of those estimated 3D feature positions and the gaze position on a monitor (which is the intersection position determined from a monitor and the normal vector). From that, if the compute gaze position does not correspond to the monitor center (because the 3D feature positions obtained by parameters (T_y, T_z, α, f, Z) and Eq. (7) are obtained in case a user gazes at a monitor center as shown in Fig. 4), the estimated 3D positions are recomputed repeatedly until computed gaze position corresponds to the monitor center. From that, the average RMS error between the real 3D feature positions and the estimated ones is much reduced (1.15 cm (0.64 cm in X-axis, 0.5 cm Y-axis, 0.81 cm in Z-axis)).

4. Estimating the 3D motion of feature points

This section explains the second step of Fig. 4. Many 3D motion estimation algorithms have been investigated, for example, extended Kalman filter (EKF) [2,17], neural network [3] and affine projection method [6,7], etc. The 3D estimation method using affine projection algorithm [6,7] has the constraint that the depth change of the object does not exceed 0.1 of the distance between camera and object. In addition, if 3D rotations and translations of feature points happen simultaneously, it cannot estimate the accurate 3D motion due to the increase of complexity of least-square fitting algorithm, which requires much processing time. The EKF has showed good performances in 3D motion estimation, but has the constraint that it may miss the tracking objects easily in case the objects change their direction abruptly. In addition, its tracking performance is susceptible to the initial values. So, we use a neural network (backpropagation) for 3D rotation estimation as shown in Fig. 8.

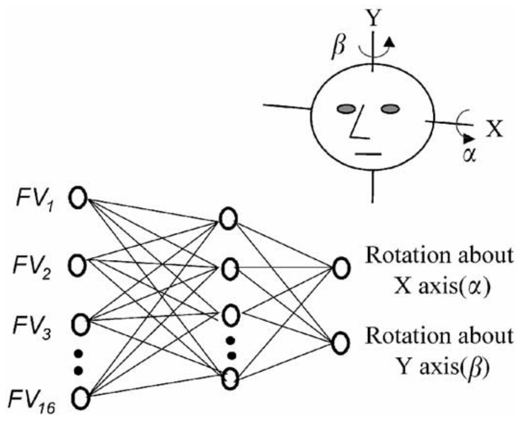

Y
β
X
α
FV_1
FV_2
FV_3
FV_{16}
Rotation about X axis(α)
Rotation about Y axis(β)

Fig. 8. Neural network for 3D rotation estimation.

Here, we use 16 feature values as mentioned in section II for neuron input and 2 continuous output representing for 3D rotation estimation (rotations about X, Y-axis). The estimation accuracy of neural network is compared with 3D position tracker sensor as shown in Fig. 9, when a user gazes at 42 gaze positions on a monitor. As experimental results, the vision and 3D position tracker sensor estimates are similar. The RMS errors between vision estimates and 3D position tracker sensor are about 2.98°.

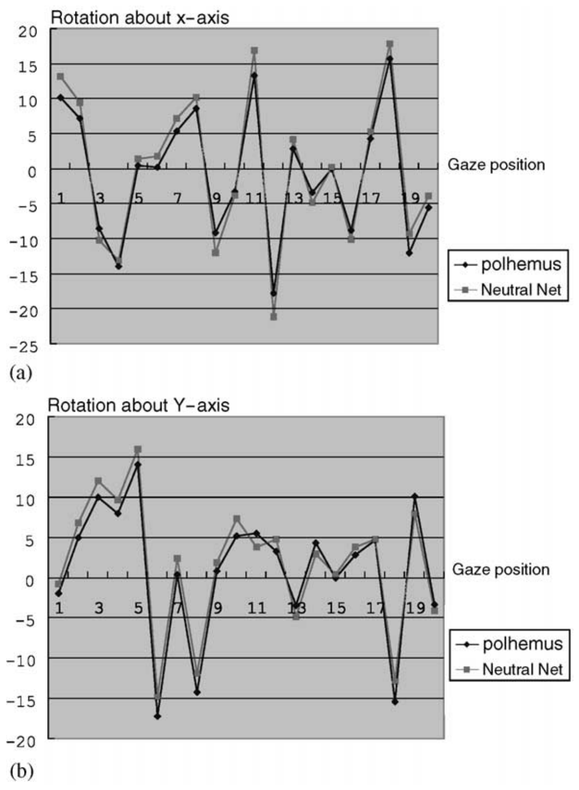


Fig. 9. The accuracy of 3D rotation estimation by neural network. (a) The estimation accuracy of rotation about X-axis(α); (b) the estimation accuracy of rotation about Y-axis(β).

For the test images, the same 20 person data used in Sections 2, and 3 are used. Now we explain our method of computing 3D facial translation. We experimented how accurate the 2D movement (FV_{17}, FV_{18}) of facial center detected in Section 2 can represent the 3D translation of face. Experimental results show that when the 3D depth between a face and monitor is measured accurately (measured by 3D position tracker sensor), we can obtain the accurate 3D translation of face from the 2D movement of facial center using Eq. (8). RMS error is 0.29 cm in X-axis and 0.45 cm in Y-axis. The RMS error is calculated between the computed 3D translation from the 2D movement of facial center and the real one (measured by 3D position tracker sensor). In the second experiment, when the 3D depth between a face and a monitor computed by the method in Section 3 is used, the RMS error of 3D translation estimation is increased, about 1.8 cm. If the 3D depth is changed when a user move his face, the 3D rotation and translation estimation are affected by that. So, we use the fact that the width of the 2D detected face contour (mentioned in Section 2) is changed according to the 3D depth and the 22 feature values as shown in Fig. 2 for 3D rotation and translation estimation are normalized by the width change of the 2D face contour.

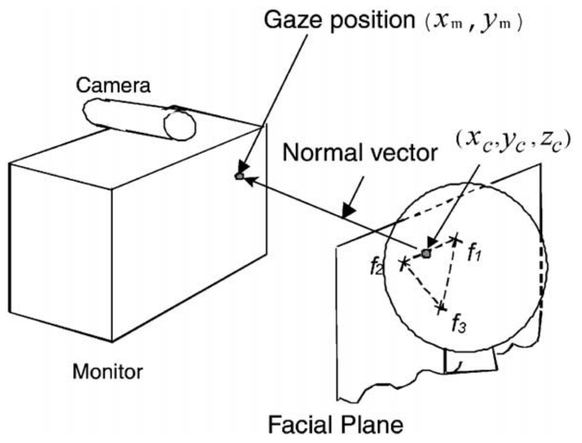


Fig. 10. Gaze position detection from facial plane.

5. Estimating the changed 3D positions of facial features

This section explains the third step of Fig. 4. The 3D feature positions computed in monitor coordinate using Eq. (8) in Section 3 are converted into the 3D feature positions in face coordinate by Eq. (2) and using these converted 3D feature positions (x_i, y_i, z_i), 3D motion matrices (rotation $[R]$ and translation $[T]$) estimated by Neural Network and affine transform described by Eq. (9), we can obtain the moved 3D feature positions (x_i', y_i', z_i') in face coordinate when a user gazes at one position on a monitor [13].

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ 0 \end{bmatrix}. \tag{9}$$

6. Computing the gaze position on the monitor

Then, those feature positions are transformed into the 3D positions in the monitor coordinate using Eq. (2). From the three facial feature points, one facial plane is determined (Eq. (10)). And the normal vector of the plane which starts at the center of both eyes (x_c, y_c, z_c) represents a gaze vector. The gaze position (x_m, y_m) on a monitor is the intersection position between a monitor plane and the gaze vector as shown in Fig. 10.

$$Ax + By + Cz = D, \tag{10}$$

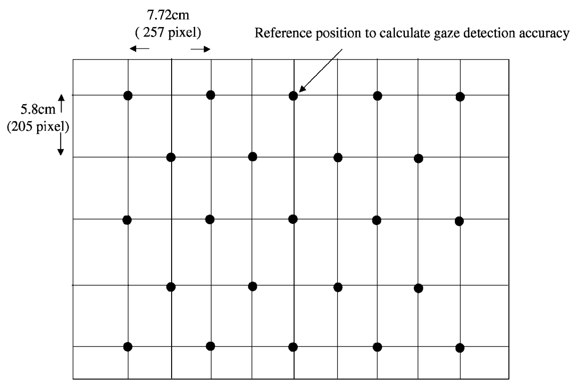


Fig. 11. The reference positions to calculate gaze detection accuracy on a monitor.

Table 1
Gaze detection error

	Only facial rotation occurs	Facial rotation and translation occur
Gaze detection error (RMS error)	4.21	5.11

$$x_m = -\frac{A}{C} * z_c + x_c, \quad y_m = -\frac{B}{C} * z_c + y_c, \tag{11}$$

where, $x_c = \frac{x_1' + x_2'}{2}$, $y_c = \frac{y_1' + y_2'}{2}$, $z_c = \frac{z_1' + z_2'}{2}$.

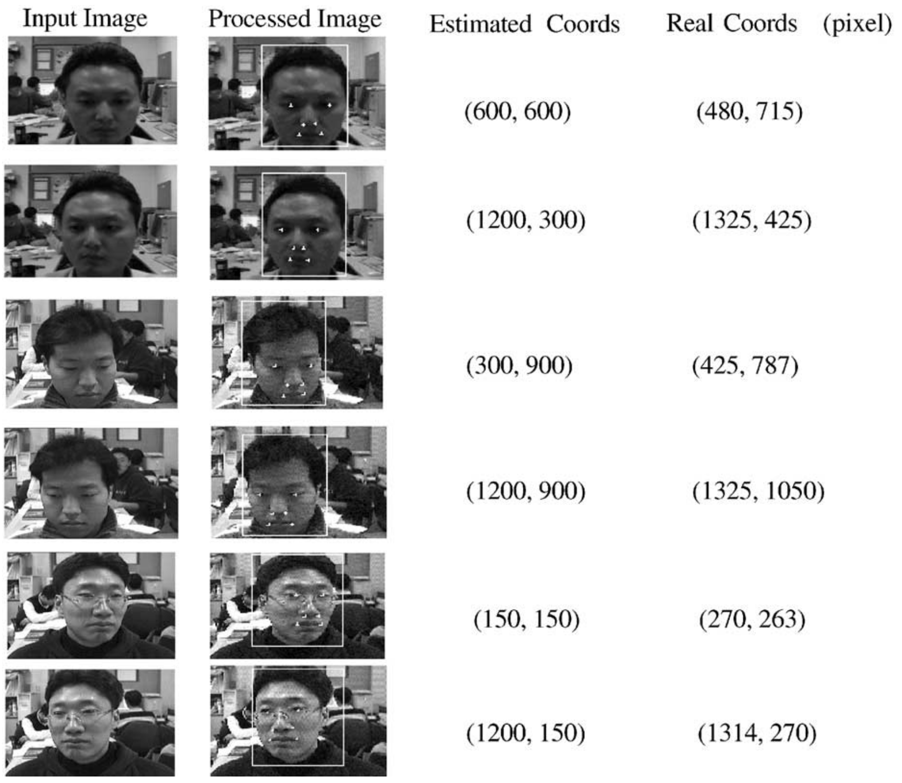
We experimented on the accuracy of the proposed gaze detection algorithm. The gaze detection error is the RMS error between the real gaze positions and the computed ones. Here, we experimented for two different cases: first, for the case with only the facial rotation, and second for the case with both the facial rotation and the facial translation. For each case, the test data are acquired with 10 users gazing at 23 positions on a 19 in monitor (1280×1024 pixel resolutions). The 23 positions on a monitor are shown in Fig. 11.

Experimental results show that the gaze detection error is a little larger in the case with both the facial rotation and translation than in the case with only the facial rotation (as shown in Table 1). The gaze detection error is a composition of the facial feature position extraction error, the facial edge detection error and the error in the estimation of the 3D facial positions and motions. This result is similar to that of Rikert's method. However, Rikert's method has the drawback that the distance between the user and the monitor screen must be always the same (50 cm) since the method does not consider the 3D facial position and motion but only use the two dimensional image data. Thus, when the user rotates and translates his head, the gaze detection error increases in Rikert's method, which can be a severe problem in real situations. This is not the case with our algorithm. In addition, Rikert's method is restricted to the application with simple background and also takes much time to detect the gaze position (1 min with alphastation 333 MHz), whereas our algorithm can be applied to the case with many complicated objects in the background and takes less than 3 s to compute the gaze position with a Pentium-Pro 200 MHz PC.

In the second experiments, we experimented the RMS errors of gaze position according to the distance (55, 60, 65 cm, here the distances are measured by 3D position tracker sensor) between the monitor and the users. The RMS errors are like; 4.85 cm in the distance of 55 cm, 5.09 cm in the distance of 60 cm, 5.13 cm in the distance of 65 cm. It is proved that our method does not have the constraints that the distance between the user and the monitor screen must be kept same.

In the third experiment, in order to calculate the gaze detection accuracies on a monitor with more reference-gaze positions, the points of 5-pixels-radius are spaced vertically and horizontally at 150 pixel intervals (4.04 cm in X-axis and 3.79 cm in Y-axis) on a 19 in monitor (total 56 gaze positions on a monitor) with a 1280×1024 pixel resolutions. The experimental conditions are almost same as Rikert's method and total 1120 examples (20 persons $\times$ 56 gaze positions with the corresponding monitor) are used for calculating gaze position error. Because we need no prior training procedures, all the data are used for testing. Fig. 12 shows some examples of experimental results. The RMS error of gaze position between the real and calculated position is 5.07 cm (X-axis: 3.64 cm, Y-axis: 3.48 cm). This RMS error is correspond to 172 pixels (X-axis: 121 pixels, Y-axis: 123 pixels) in 1280×1024 pixel monitor. That results shows that we can use about 20 clickable menus (X size is 242 pixels, Y size is 246 pixels) in 1280×1024 pixel monitor not feeling uncomfortable caused by gaze detection error. However, in general windows display or Web browser, the menu size is much small (X size is below 60 pixels, Y size is below 60 pixels in 1280×1024 pixel monitor) and it is difficult to control such small menus only by gaze detection. So we can solve this problem with head dragging. In details, the exact gaze position on a monitor can be acquired by the additional head dragging (similar to mouse dragging) after gaze detection.

Fig. 13 shows a gaze detection example in a conventional chemical vapor deposition window (1280×1024 pixels resolution). The CVD is a process utilized in many areas including fabrication and surface coating of semiconductor devices and it has so many control buttons in window [1]. In Fig. 13, when a user gazes at right upper position, then the cursor appears in the gazed position (a). Because of gaze detection error, the cursor seems to be placed in near position of which button the user wish to control. Then the cursor can be moved to the exact position by head drag-



Input Image	Processed Image	Estimated Coords	Real Coords (pixel)
		(600, 600)	(480, 715)
		(1200, 300)	(1325, 425)
		(300, 900)	(425, 787)
		(1200, 900)	(1325, 1050)
		(150, 150)	(270, 263)
		(1200, 150)	(1314, 270)

Fig. 12. Gaze detection error in 1280 × 1024 pixel monitor.

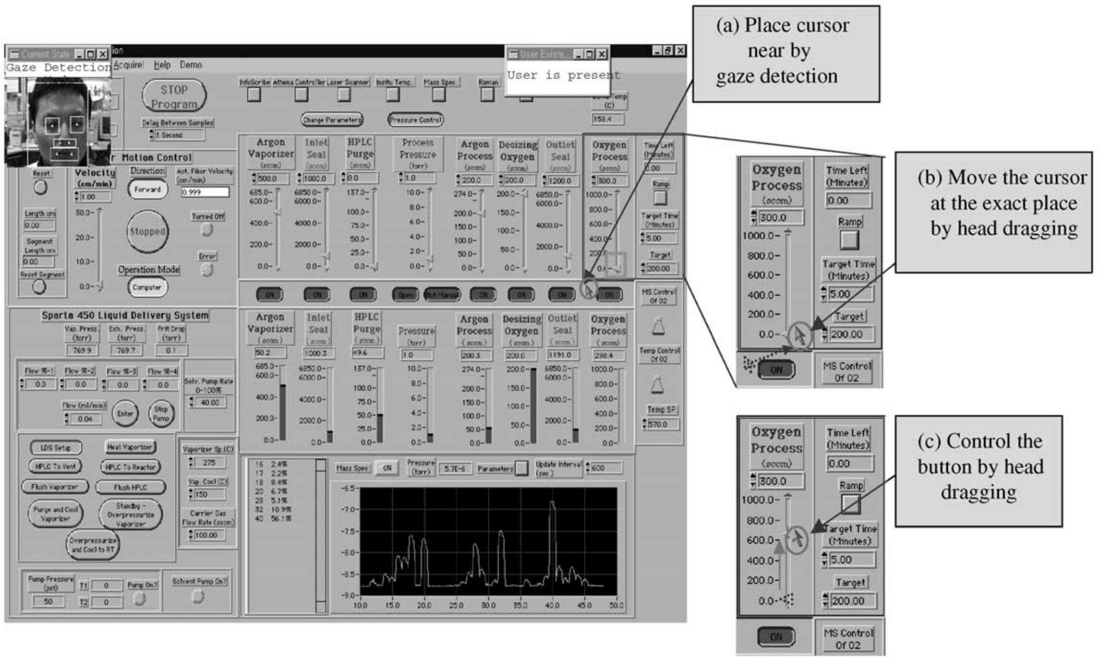

(a) Place cursor near by gaze detection
(b) Move the cursor at the exact place by head dragging
(c) Control the button by head dragging
Gaze Detection
User is present
Oxygen Process
Time Left (Minutes)
Ramp
Target Time (Minutes)
Target
MS Control Of 02
ON

Fig. 13. Gaze detection example in CVD windows (1280 × 1024 pixel resolution).

ging (b). After that, the button can be controlled by head dragging (c).

7. Conclusions

This paper describes a new method for detecting the gaze position on a monitor by a single camera. In order to detect the gaze position, we locate facial features automatically in 2D camera images and estimate the initial 3D positions of those features by camera calibration and parameter estimation algorithm. Then, the 3D motions of feature points in succeeding images are estimated and the gaze positions are computed from those estimation results. Our study resulted that the initial 3D feature position estimation error between the computed feature positions and the real ones is below 1.15 cm and the 3D motion estimation errors of feature points are about 2.98° and 1.8 cm in rotation and translation, respectively. From that, we can detect the gaze position on a monitor and the gaze position detection error is about 5.11 cm (RMS error) when the distance between the user and the monitor is 50–70 cm and a 19 in monitor is used. In a further study, we are going to improve the gaze detection performance by using some feature relocation algorithms, improving the 3D motion estimation and considering the eye movements.

8. Summary

Gaze detection is to locate the position on a monitor screen where a user is looking. In our work, we implemented it by a computer vision system with a single camera above the monitor. Then a user moves (rotates and/or translates) his face to gaze at a certain position on the monitor. For our case, the user is requested not to move the pupils of his eyes as he gazes at different positions on the monitor screen, though we are working on relaxing this restriction. To detect the gaze position, we locate the facial region and the facial features (both eyes, nostrils and lip corners) automatically in 2D camera images. From the movement of the feature points detected in starting images, we can compute the initial 3D positions of those features by a camera calibration and a parameter estimation algorithm. Then, when the user moves (rotates and/or translates) his face to gaze at a certain position on the monitor, the moved 3D positions of those features can be computed from 3D rotation and translation estimations and affine transform. Finally, the gaze position on the monitor is computed from the normal vector of the plane determined by those moved 3D positions of features. Especially, in order to obtain the exact 3D positions of the initial feature points, we unify three coordinate systems (face, monitor and camera coordinate system) based on perspective transformation. Our study resulted that the 3D position estimation error of initial feature points, which is the RMS error between the estimated initial 3D feature positions and the real positions (measured by 3D position tracker sensor), is about 1.15 cm (0.64 cm in X-axis, 0.5 cm in Y-axis, 0.81 cm in Z-axis) and the 3D motion estimation errors of feature points are about 2.98° and 1.8 cm in rotation and translation, respectively. From that, we can obtain the gaze position on the monitor (19 in). The gaze position accuracy between the computed positions and the real ones is about 5.11 cm of RMS error.

References

[1] J. Kim, K.R. Park, S.R. LeClair, Process control via gaze detection technology, The Second International Conference on Intelligent Processing and Manufacturing of Materials, Honolulu, Hawaii, July 10–15, 1999, pp. 1263–1269.

[2] A. Azarbayejani, Visually controlled graphics, IEEE Trans. PAMI 15 (6) (1993) 602–605.

[3] T. Fukuhara, T. Murakami, 3D-motion estimation of human head for model-based image coding, IEE Proc. 140 (1) (1993) 26–35.

[4] K. Ohmura, A. Tomono, Y. Kobayashi, Pointing operation using detection of face direction from a single view, IEICE Trans. Inf. Systems J 72-D-II (9) (1989) 1441–1447.

[5] P. Ballard, G. Stockman, Controlling a computer via facial aspect, IEEE Trans. SMC 25 (4) (1995) 669–677.

[6] A. Gee, R. Cipolla, Fast visual tracking by temporal consensus, Image Vision Comput. 14 (1996) 105–114.

[7] J. Heinzmann, A. Zelinsky, 3-D facial pose and gaze point estimation using a robust real-time tracking paradigm, Proceedings of the International Conference on Automatic Face and Gesture Recognition, 1998, pp. 142–147.

[8] T. Rikert, M. Jones, Gaze estimation using morphable models, Proceedings of the International Conference on Automatic Face and Gesture Recognition, 1998, pp. 436–441.

[9] A. Tomono, F. Kishino, Gaze point detection algorithm based on measuring 3D positions of face and pupil, IEICE Trans. Inf. Systems J 75-D-II (5) (1992) 861–872.

[10] A. Tomono, I. Muneo, O. Kazunori, Eye Tracking Method Using an Image Pickup Apparatus, European Patent Specification-94101635.4, 1994.

[11] Seika-Tenkai-Tokushuu-Go, ATR J. 1996.

[12] S.W. Nam, K.R. Park, S.C. Han, J. Kim, A facial-feature tracking algorithm based on the constant acceleration model in multimodal interface environment, The 10th Workshop on Image Processing and Understanding, Jaeju Island, Korea, 1998, pp. 209–214.

[13] R. Jain, Machine Vision, McGraw-Hill, New York, 1995.

[14] J. Heinzmann, A. Zelinsky, Robust real-time face tracking and gesture recognition, Proceedings of the International Joint Conference on Artificial Intelligence, Vol. 2, 1997, pp. 1525–1530.

[15] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1995.

[16] S.C. Chapra, R.P. Canale, Numerical Methods for Engineers, McGraw-Hill, New York, 1989.

[17] A. Kiruluta, Predictive head movement tracking using Kalman filter, IEEE Trans. SMC 27 (2) (1997).

About the Author—KANG RYOUNG PARK received the B.S. and M.S. degrees in Electronic Engineering from Yonsei University, Seoul, Korea, in 1994 and 1996, respectively. He received the Ph.D. degree in Computer Vision, Image Processing at the Yonsei University, in 2000.

About the Author—JEONG JUN LEE received the B.S. and M.S. degrees in Electronic Engineering from Yonsei University, Seoul, Korea, in 1995 and 1998, respectively. He is currently in the course of Ph.D. of Electrical and Electronic Engineering. His research interests include the computer vision and object recognition.

About the Author—JAIHIE KIM got the B.S. degree in Electronic Engineering at the Yonsei University, Korea, in 1979 and the M.S. degree in Data Structures and the Ph.D. degree in Artificial Intelligence at the Case Western Reserve University, Ohio, USA, in 1982 and 1984, respectively. Since 1984, he has been a professor at the School of Electrical and Mechanical Engineering at Yonsei University. He has been researching in the areas of computer vision, image processing/understanding and pattern recognition. Dr. Kim was the president of the AI, Neural Network and Fuzzy System Society of the Institute of Electronics Engineers of Korea from January–December in 1993. He has been the associate editor of Neurocomputing (Netherlands) from September 1995 and the director of the Institute of Electronics Engineers of Korea from January 1998. He also served as the vice president of the Center for Signal Processing Research in Yonsei University from February 1998–February 2000.