Camera Mouse Implementation Using 3D Head Pose Estimation by Monocular Video Camera and 2D to 3D Point and Line Correspondences

Masoomeh Nabati (Msc. Student)
Electrical Engineering Department
Shahed University
Tehran, Iran
masoomeh_nabaaty@yahoo.com

Alireza Behrad (Assist. Prof.)
Electrical Engineering Department
Shahed University
Tehran, Iran
behrad@shahed.ac.ir

Abstract— **In this paper, we present a novel approach to estimate the 3D head pose from a monocular camera images for the control of mouse pointer movements on the screen and clicking events. This work is motivated by the goal of providing a non-contact instrument to control the mouse pointer on the PC screen for helping handicapped people with severe disabilities using low-cost and widely available hardware. The required information is derived from video data captured using a web camera mounted on the computer monitor. The proposed algorithm is based on the 2D tracking of the markers on the face. Our approach proceeds in three stages. First, the positions of the markers are detected and tracked over video frames by LK algorithm. Then, the 3D rotation and translation between the web camera and 3D head pose are estimated using point and line correspondences. Finally, the 3D rotation and translation matrix is used for estimating the mouse pointer movements on the PC screen and clicking events. Experimental results showed the promise of the algorithm.**

Keywords- Camera mouse; 3D head pose estimation; mouse pointer control; visual tracking module;

I. INTRODUCTION

More than 500 million persons, 10 percent of the world's total population, suffer from some type of disability. Nowadays different support devices and care equipment have been developed to help the handicapped people. One of the main support devices for handicapped people with severe disabilities is an instrument for communication with computers or similar devices. However, the main problem with these devices is the control difficulties due to the limited physical abilities of the users. Thus, different type of interfaces have been developed to facilitate communication between the handicapped users and the devices like computers [1- 5]. These interfaces are mainly categorized into two groups including intrusive and non-intrusive methods. Intrusive methods mostly use contact sensors which measure human reflections or activities. Although intrusive methods can detect features or signals more accurately, they require expensive devices and are not flexible [6-10]. Non-intrusive methods mostly track human gestures by processing images or videos obtained via a camera. In contrast to intrusive methods, they are more comfortable for the users and involve less expensive communication devices.

A camera mouse system is a non-intrusive method that helps handicapped people to interact with computers. A camera mouse system is usually composed of one or multiple video cameras for capturing video frames and a processing unit like a PC which uses image processing algorithm to convert the motion events in video frames to mouse operations. The algorithm is usually formed from a visual tracking module and a mouse control module. The visual tracking module retrieves motion information from the video, and the mouse control module specifies the rules of control.

Different algorithms have been proposed for the implementation of camera mouse which most of them use head pose or movements, and facial features like eyes, nostrils and mouth. The use of head and facial feature for camera mouse implementation is due to two reasons; they have high communicative information and most camera mouse users have severe hand disability. Systems using eye movements have already been presented in [11-13] where mouse motion and click events are generated based on eye motion and fixing in a specific area for one or more seconds. Mouth movements and lip reading is another method for camera mouse implementation which has been presented in [14, 15]. In this method various mouth shapes is used for lip reading. The system proposed by Yunhee Shin et al [16] is able to control the mouse based on eye and mouth movements. Jilin Tu et al [17] presented a system that driven by visual face tracking based on a 3D model. This system needs to be initialized with face Action Units in the 3D space and then estimates rotation and translation matrix. Nostrils and nose tip systems presented in [18] and [19] respectively are the other systems for the control of the mouse pointer on the screen. The main problem with using facial feature for camera mouse is their small area in the input image; therefore we need to have enough zoom ratios or high resolution camera which is not possible in most of webcams. However the head area in the input image is large enough and head pose usually indicates the focus of attention.

In this paper, we present a novel approach for camera mouse implementation using the 3D head pose estimation. Most important novelty of the proposed approach is the robust estimating of the 3D rotation and translation matrix by using only one camera. In addition both click and mouse movement event are implemented using 3D head pose. The proposed

algorithm is real time and experimental results showed higher accuracy of the proposed method. To make the algorithm real time we used four artificial markers on the face which is shown in Fig. 1. These markers are detected and tracked to estimate 3D head pose over video frames. The 3D head pose are used to implement mouse operations. Since we extract 3D rotation and translation matrix, the algorithm may be used in other 3D applications like virtual reality applications.

The remainder of this paper is structured as follows: Section 2 presents block scheme of the proposed algorithm. Section 3 describes 3D head pose estimation from 2D to 3D point and line correspondences. The algorithm for mouse controller is described in Section 4. Section 5 presents the implementation results and some final conclusions are given in Section 6.

Figure 1. Four markers on the face for camera mouse implementation

II. BLOCK SCHEME OF THE PROPOSED ALGORITHM

Fig. 2 shows the block scheme of proposed method for camera mouse implementation. The method estimates 3D head pose including its 3D translation and rotation for camera mouse implementation. To estimate 3D head pose, we used 2D to 3D point and line correspondences. To make the algorithm real time we used four artificial markers on the user face which form a rectangle with two sets of parallel lines as shown in Fig. 1.

First stage of proposed approach is the detection and tracking of the markers. We used small black markers on the face which can be detected using a color segmentation algorithm precisely. To detect markers we first detect face area in the image using AdaBoost based face detector algorithm [20]. Then the location of the markers is estimated in face area and corrected using color segmentation. To track the marker we used recursive and multiresolution implementation of LK algorithm [21]. In the second stage of algorithm, the 2D coordinates of markers are used to estimate the 3D rotation and translation matrix by 2D to 3D point and line correspondences method. Finally, we calculate rotation angles around X, Y and Z axes using the calculated rotation matrix. The rotation angles around X and Y axes are then translated into mouse pointer movements on the screen. To implement clicking events we used the distance change between user and camera which is obtained using Z component of translation vector obtained in the previous stage.

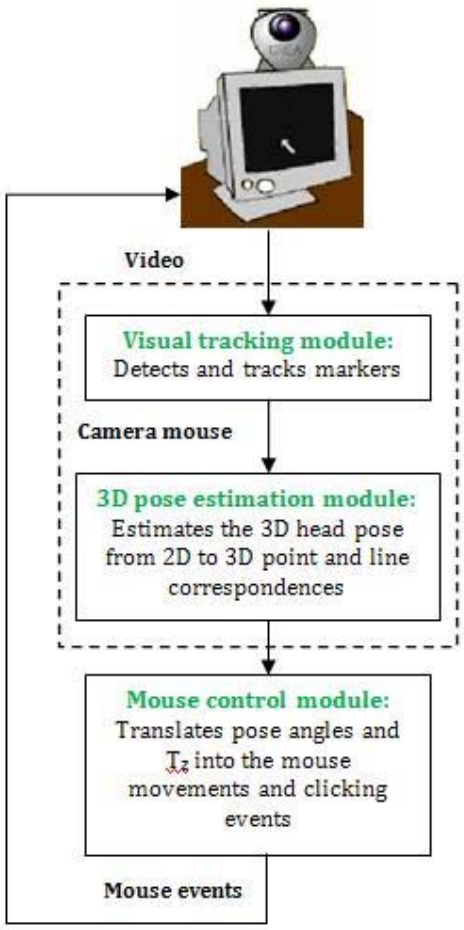

Figure 2. The block scheme of proposed approach for camera mouse implementation

III. ESTIMATING THE HEAD POSE FROM 2D TO 3D POINT AND LINE CORRESPONDENCES

A. *Coordinates System*

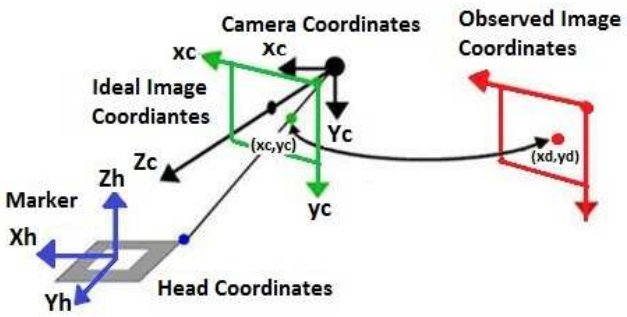

Figure 3. Different coordinate systems for head pose estimation

Fig. 3 shows different coordinate systems used for head pose estimation. The definitions for different coordinate systems are as follows:

1) Head coordinate system

The system is used for specifying four marker points on the user head. We assumed that all marker points on the face are located on the same plane, therefore the origin of the object coordinate frame, X_h and Y_h axes lie on the plane containing marker points, while the Z_h axis is perpendicular to the plane. Since all markers are in the X_h Y_h plane, their Z values are zero in head coordinate system.

2) Camera coordinate system

This system is used to convert 3D coordinates to image coordinates. To convert coordinates in head coordinate system to camera coordinate system the following equations are used [22]:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = T_{CH} \begin{bmatrix} X_H \\ Y_H \\ Z_H \\ 1 \end{bmatrix} \tag{1}$$

$$T_{CH} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where R_{11} to R_{33} are the parameters of rotation matrix and T_x, T_y and T_z represent translation between two coordinate systems.

3) Ideal image coordinate system

This coordinate system shows the ideal 2D coordinates of image pixels neglecting lens distortion. To convert 3D camera coordinates to ideal image coordinate system pinhole camera model and perspective projection are used as follows [22]:

$$\begin{bmatrix} hx_I \\ hy_I \\ h \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_c & 0 \\ 0 & f_y & y_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = C \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3}$$

where (x_I, y_I) are ideal image coordinate (without lens distortion) and f_x, f_y, x_c and y_c are camera parameters.

4) Observed Image Coordinate System

This coordinate system shows the real 2D coordinates of image pixels in computer. Computer program detect markers in this coordinate system. To obtain observed image coordinates we should consider the effect of lens distortion as follows [22]:

$$d^2 = (x_I - x_{do})^2 + (y_I - y_{do})^2 \tag{4}$$

$$p = 1 - kd^2 \tag{5}$$

$$x_o = p(x_I - x_{do}) + x_{do} \tag{6}$$

$$y_o = p(y_I - y_{do}) + y_{do} \tag{7}$$

where k is distortion factor, (x_{do}, y_{do}) are coordinates of distortion center and (x_o, y_o) are observed image coordinates.

B. Head Pose Estimation

Given a calibrated camera and correspondences between 3D marker points in head coordinate system and 2D detected marker points in the input image, the goal of head pose determination algorithm is to estimate the rotation and translation matrix between head coordinate system and camera coordinate system i.e. T_{CH}.

Calculation of T_{CH} may be considered as an optimization problem, where the following error function should be minimized [22].

$$e = \sum_{j=1}^{N} \left[(\hat{x}_j - x_j)^2 + (\hat{y}_j - y_j)^2 \right] \tag{8}$$

where (x_j, y_j) are the coordinates of detected marker points, N is the number of detected marker points which is 4 for our algorithm, and $(\hat{x}_j, \hat{y}_j)$ are calculated using the following equation[22]:

$$\begin{bmatrix} \hat{x}_j \\ \hat{y}_j \end{bmatrix} = F \left[C \times T_{CM} \begin{bmatrix} X_{Mj} \\ Y_{Mj} \\ Z_{Mj} \\ 1 \end{bmatrix} \right] \qquad j = 1:N \tag{9}$$

where *F* presents the lens distortion function and $[X_{Mj}, Y_{Mj}, Z_{Mj}]$ are the 3D coordinates of markers in head coordinate system. By solving the optimization problem of equation (8) it is possible to obtain T_{CH} matrix which represents 3D head pose.

The main problem with the optimization problem is the selection of initial values which affects the accuracy of the optimization algorithm. To cope with this problem we estimate rotation matrix first and use the estimated rotation matrix as the initial value for the optimization problem.

C. Estimating Rotation Matrix

As mentioned before to obtain accurate results using optimization problem, it is better to estimate rotation matrix and use the estimated rotation matrix as the initial value for the optimization problem. The marker points which we have considered on user face have rectangular geometry; therefore the following geometrical constraint can be used to obtain 3D coordinates of marker points [23].

$$\overrightarrow{AB} = \overrightarrow{DC} \Leftrightarrow \begin{bmatrix} X_B - X_A \\ Y_B - Y_A \\ Z_B - Z_A \end{bmatrix} = \begin{bmatrix} X_C - X_D \\ Y_C - Y_D \\ Z_C - Z_D \end{bmatrix} \tag{10}$$

where A, B, C and D are four marker points on the user face. When the 3D coordinates of marker points are obtained the rotation matrix vectors are calculated using the following equation [23]:

$$r_{1*} = \frac{\overrightarrow{AB}}{\left\|\overrightarrow{AB}\right\|} \qquad r_{2*} = \frac{\overrightarrow{AC}}{\left\|\overrightarrow{AC}\right\|} \qquad r_{3*} = r_{1*} \wedge r_{2*} \tag{11}$$

IV. Mouse Pointer Control

When the 3D head pose including rotation and translation matrix is calculated, it is used for the control mouse pointer and generating mouse events. The rotation matrix is an orthogonal matrix and can be represented using three rotation angles around x, y and z axes as follow:

(12)

where α, β and γ are rotation angles around x, y and z axes respectively. Considering flexible rotation of human face around x and y axes, we used α and β for generating mouse movement in x and y directions respectively. Since the R matrix are obtained using some measurements and optimization algorithm it may not be an orthogonal matrix because of measurement and calculation error, therefore we orthogonalize it using singular value decomposition and calculate the required rotation angles in degree using equations (13, 14):

$$\beta = 360 \times \sin^{-1} \frac{(R_{13})}{2\pi} \tag{13}$$

$$\alpha = 360 \times \sin^{-1} \frac{\left(-\frac{R_{23}}{\cos\left(\sin^{-1}(R_{13})\right)} \right)}{2\pi} \tag{14}$$

Equations (15, 16) show the translation of the rotation angle into mouse movement on the screen.

$$mouse\_x = \left(\frac{\beta}{\beta_{\max}} \right) \times \left(\frac{w}{2} \right) + \left(\frac{w}{2} \right) \tag{15}$$

$$mouse\_y = \left(\frac{\alpha}{\alpha_{\max}} \right) \times \left(\frac{h}{2} \right) + \left(\frac{h}{2} \right) \tag{16}$$

where mouse_x and mouse_y are the coordinates of mouse pointer on the screen, w and h are screen width and height in pixel and α_{max} and β_{max} are maximum head rotation around x ad y axes respectively.

To generate click events we used the distance between camera and user head i.e. T_z. We assumed that the natural distance between camera and user head is between two user defined parameters called min_distance and max distance. When user head distance is less than min_distance for more than 500 ms or 7 frame and then head returns to natural distance, left click event is generated. The same procedure is used to generate right click event if the distance is larger than max_distance. Fig. 4 shows the pseudo code to generate mouse click events.

```
While (frame)  do
    if  min_distance <  Tz < max_distance
        if  L_Number ≥ 7
            generate left click event
        end if
        if  R_Number ≥ 7
            generate right click event
        end if
        R_Number=L_Number=0
    End if
    If  Tz < min_distance
        L_ Number++
        R_Number=0
    End if
    If  Tz > max_distance
        R_ Number++
        L_Number=0
    End if
End  while
```

Figure 4. Pseudo code for mouse click implementation

V. Experimental Results

We implemented the proposed algorithm using a Microsoft Visual C++ program and Intel's OpenCV library for computer vision [24]. To solve the optimization problem of equation (8) and showing the result as 3D graphics, we utilized the functions provided by ARToolKit [25]. In order to set suitable initial values for rotation and translation matrices, the results of previous frame is used as initial values for current frame. For the first frame we used the method described in section 3.3 for the initialization of rotation matrix.

Fig. 5 shows the project interface consisting of a 1.3M pixel webcam mounted on the laptop PC's monitor. The laptop CPU is an Intel Core 2 Duo CPU with the speed of 2GHz and Windows XP operating system. According to equations (15, 16) the resolution (width and height) of the computer screen can be set arbitrarily in the proposed system, however we used the resolution of 320*240 in our experimental tests.

Fig. 6 shows the calculated head coordinate system displayed using ARToolKit functions.

Figure 5. Camera mouse implementation using a laptop PC

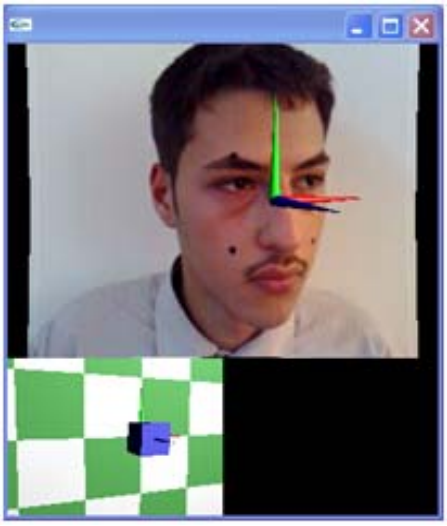

Figure 6. The calculated head coordinate system displayed using ARToolKit functions

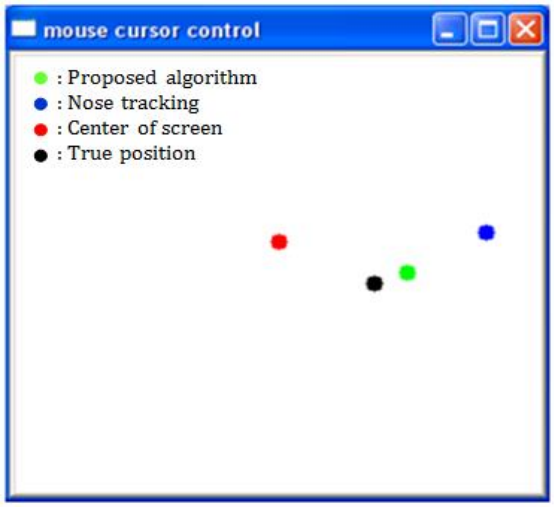

Figure 7. Position of mouse pointer on the screen for a typical frame of a test video

We evaluated the performance of our system in different situations and lighting conditions. We also tested it on different users. To compare the results of proposed method with that of another method, we also implemented the camera mouse using nose tracking [18]. To evaluate the performance of camera mouse algorithm we calculate the position of mouse pointer using the algorithm and compare it with true position which is estimated using human user. Fig. 7 shows the true position and the position of the mouse pointer obtained using the proposed method and nose tracking algorithm [18] for a typical frame of a test video. The results for all frames of the test video are shown in Fig. 8. As it is obvious from Fig. 7 and Fig. 8, the proposed algorithm gives rise to better results. Table 1 summarizes the results of camera mouse algorithm over 3000 frames of different test videos. The average time to process a frame is about 67ms for the proposed algorithm which means the proposed system is capable of processing 14 frames/sec. The table shows the mean absolute error (MAE) and standard deviation (SD) and average deviation (AD) of MAE for mouse positions in x and y directions. These parameters are defined in equations (17-19) for x direction:

$$MAE = \frac{\sum ex_i}{N} \tag{17}$$

$$SD = \sqrt{\frac{\sum (ex_i - MAE)^2}{N}} \tag{18}$$

$$AD = \frac{|ex_i - MAE|}{N} \tag{19}$$

where ex_i is the absolute error in x direction for i^{th} frame and N is the total number of frames.

TABLE I shows that the error for the proposed algorithm is twice less than camera mouse implementation using nose tracking.

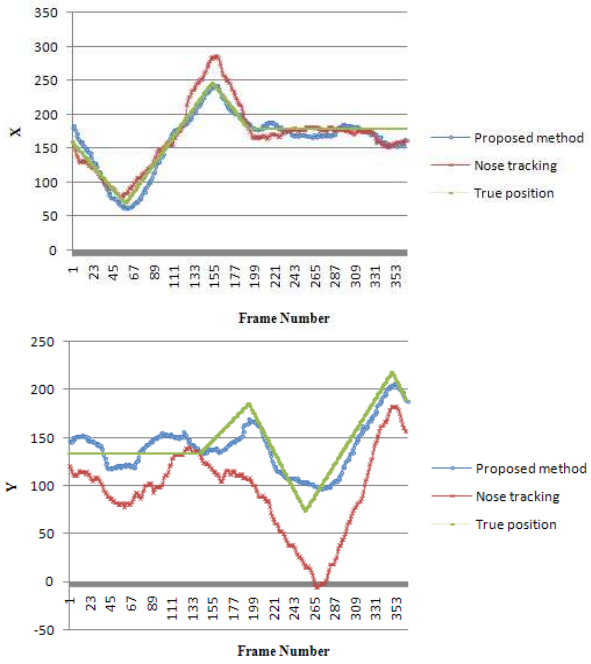

Figure 8. Mouse position in x and y directions for different frames of a typical test video

TABLE I. CAMERA MOUSE ERROR FOR 3000 FRAMES OF DIFFERENT VIDEOS

Mouse direction		Proposed method	Nose tracking
X	MAE	7.5	19.85
	AD	5.7	13.33
	SD	8.3	16.92
Y	MAE	10.17	32.9
	AD	8.38	20.02
	SD	11.78	25.14

VI. CONCLUSION

In this paper we proposed a new method for camera mouse implementation using monocular video camera and 3D head pose estimation by 2D to 3D point and line correspondences. The algorithm is real time and can process 14 frames/sec. Experimental results showed that the error for the proposed algorithm is twice less than that of another algorithm. To obtain the 3D pose of head we need four feature points on the face constituting a rectangle. For this purpose and make the algorithm real time we used artificial markers as feature points. For future development we are going to use natural features on the face.

REFERENCES

[1] D. Stefanov, Z. Bien, and W. Bang, "The Smart House for Older Persons and Persons With Physical Disabilities: Structure, Technology Arrangements, and Perspectives," IEEE Transaction on Neural Systems and Rehabilitation Engineering, vol. 12, no. 2, pp. 228-250, 2004.

[2] T. Carlson and Y. Demiris, "Using Visual Attention to Evaluate Collaborative Control Architectures for Human Robot Interaction," Imperial College London, 2008.

[3] J. Alon, V. Athitsos, and S. Sclaroff, "Simultaneous localization and recognition of dynamic hand gestures," in Proc. 2005 IEEE Motion Workshop, 2005.

[4] C. S. Lin, C. W. Ho, C. N. Chan, C. R. Chau, Y. C. Wu, and M. S. Yeh, "An eye-tracking and head-control system using movement increment-coordinate method," in Proc. 2007 Optics & Laser Technology Conf., pp. 1218–1225.

[5] M. C. SU, K. C. Wang, and G. D. Chen, "An eye tracking system and its application in aids for people with severe disabilities," Department of Computer Science and Information Engineering, National Central University, Chung Li, Taiwan, vol. 18, no. 6, pp. 319-327, December 2006.

[6] C. Mauri, T. Granollers, J. Lorés, M. García, "Computer vision interaction for people with severe movement restrictions," An Interdisciplinary Journal on Humans in ICT Environments, vol. 2, no. 1, pp. 38-54, April 2006.

[7] G. M. Eom, K. S. Kim, C. S. Kim, J. Lee, S. C. Chung, B. Lee, H. Higa, N. Furuse, R. Futami, and T. Watanabe, "Gyro Mouse for the Disabled: 'Click' and 'Position' Control of the Mouse Cursor," International Journal of Control, Automation, and Systems, vol. 5, no. 2, pp. 147-154, April 2007.

[8] C. Veigl, "An Open-Source System for Biosignal- and Camera-Mouse Applications," studying Medical Computer Science, Technical University Vienna, 2006.

[9] C. Topal, A. Doğan, and Ö. N. Gerek, "A Wearable Head-Mounted Sensor-Based Apparatus for Eye Tracking Applications," in Proc. VECIMS 2008 IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems.

[10] M. A. Qamar, and A. Jehanzeb, "Retina Based Mouse Control (RBMC)," World Academy of Science, Engineering and Technology, 2007.

[11] B. Scassellati, "Eye Finding via Face Detection for a Foveated, Active Vision System," MIT Artificial Intelligence Lab Cambridge, MA, 02139, USA, 1998.

[12] K. E. Yi, and K. S. Kuk, "Eye tracking using neural network and mean-shift," LNCS, vol. 3982, pp. 1200–1209, 2006.

[13] J. J. Magee, M. R. Scott, B. N. Waber, and M. Betke, "EyeKeys: A real-time vision interface based on gaze detection from a Low-grade Video Camera," Computer Science Department, Boston University, 2004.

[14] H. E. Cetingul, Y. Yemez, E. Erzin, and A. M. Tekalp, "Discriminative analysis of lip motion features for speaker identification and speech-reading," IEEE Trans Image Process, vol. 15, no. 10, pp. 2879-91, 2006.

[15] S. Stillittano, and A. Caplier, "Inner lip segmentation by combining active contours and parametric models," in Proc. of International Conference on Computer Vision Theory and Applications., pp. 297-304. 2008.

[16] Y. Shin, J. S. Ju, E. Y. Kim, "Welfare interface implementation using multiple facial features tracking for the disabled people," Pattern Recognition Letters, vol. 29, pp. 1784–1796, 2008.

[17] J. Tu, H. Tao, T. Huang, "Face as mouse through visual face tracking," Computer Vision and Image Understanding, vol. 108, pp. 35-40, 2007.

[18] C. M. Yee, J. Varona, and F. J. Perales, "Face-Based Perceptual Interface for Computer-Human interaction," Departament de Matemàtiques i Informàtica, 2006.

[19] J. Na, W. Choi, D. Lee, "Design and Implementation of a Multimodal Input Device Using a Web Camera," ETRI Journal, vol. 30, no. 4, pp. 621-623, August 2008.

[20] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proc. 2001 IEEE Conference on Computer Vision and Pattern Recognition.,

[21] J. Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm," Intel Corporation., Microprocessor Research Labs, 1994.

[22] H. Kato, K. Tachibana, M. Billinghurst, and M. Grafe, "A registration method based on texture tracking using ARToolKit", In The Second IEEE Int. Augmented Reality Toolkit Workshop, 7th October 2003.

[23] J. Y. Didier, F. E. Ababsa, and M. Mallem, "Hybrid camera pose estimation combining square fiducials localization technique and orthogonal iteration algorithm," International Journal of Image and Graphics (IJIG), vol. 8, pp. 169-188, 2008.

[24] Open Source Computer Vision Library [Online]. Available: http://sourceforge.net/projects/opencvlibrary

[25] ARToolKit library, [Online]. Available: http://www.hitl.washington.edu/artoolkit