Robust Facial 2D Motion Model Estimation for 3D Head Pose Extraction and Automatic Camera Mouse Implementation

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 3D head pose estimation from 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 a PC system for handicapped people with severe disabilities using low-cost and widely available hardware. The required information is derived from video data captured using a monocular web camera mounted on the computer monitor. Our approach proceeds in six stages. First, the face area is extracted using Haar-like features and AdaBoost algorithm. Second, the locations of the point features are detected and tracked over video frames by LK algorithm. Third, the 2D transformation model between consecutive frames is estimated by matching features and robust RANSAC algorithm. Fourth, the estimated 2D transformation model is applied to four supposed points on the face area. Then, the 3D rotation matrix and translation vector between the web camera and 3D head pose are estimated using four points correspondences. Finally, the 3D rotation and translation matrix is applied for estimating the mouse pointer movements on the PC screen and clicking events. Experimental results showed the promise of the algorithm.**

Keywords-Camera mous; 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 has been developed to facilitate the 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: first, they have high communicative information and second: most camera mouse users have severe hand disability. Systems using eye movements have already been presented in [11-13], where mouse movements and click events are generated based on eye movements and fixing in a specific area for one or more seconds. Using 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 ratio for the 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 face detection and 3D head pose estimation from monocular camera images. In our previous work [20] we discussed about the algorithm for the calculation of 3D head pose using four artificial rectangular points on the face. To remove the need for marking artificial points on the face and increasing the accuracy of the camera mouse, in this work we utilized a new algorithm to obtain the coordinates of the four arbitrary virtual points on the face area in each frame. For this purpose we calculate the 2D transformation model between two consecutive frames using point features extraction and matching and RANSAC algorithm [21]. The calculated model is used to calculate the coordinates of four virtual points. Then, 3D head pose is estimated using the coordinates of four virtual points. Finally, the 3D head pose are used to implement mouse operations.

The extracted 3D rotation and translation matrix not only is applicable for camera mouse implementation but also for applications like virtual reality environment which need 3D head position.

The remainder of this paper is structured as follows: Section 2 describes the block scheme of the proposed algorithm. Section 3 describes the method for extraction of the face area and tracking features. Section 4 explains the algorithm for estimating of the 3D head pose from 2D to 3D point correspondences, followed by mouse controller which is discussed in Section 5. Section 6 presents the implementation results and some final conclusions are given in Section 7.

II. Block Scheme of the Proposed Algorithm

Fig. 1 shows the block scheme of proposed algorithm 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 correspondences. Our method requires four rectangular points correspondence on face screen for obtaining 3D head pose. However the precise detection and tracking of these points is mandatory for the robust camera mouse implementation. In our previous work [20] we used four artificial points for this purpose, however to increase the efficiency of algorithm we used four arbitrary virtual and rectangular points on the face in this work. The points are virtual and their coordinates are arbitrary on the face. The only restriction is that they should form a rectangle with two sets of parallel lines on the user face as shown in Fig. 2.

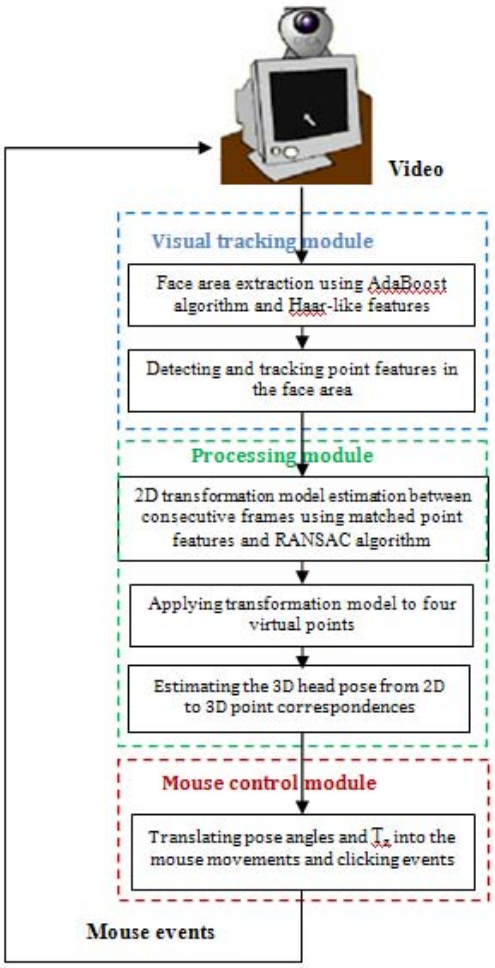

Figure 1. The block scheme of proposed approach for camera mouse implementation

Figure 2. Four virtual points on the face for camera mouse implementation

As it is shown in Fig. 1, the proposed algorithm includes six stages structured in three modules. Algorithm starts with face detection stage using Haar-like features and AdaBoost algorithm [22] at first frame. Then we extract point features and track them in the next frames. To extract and match point features different algorithms like SIFT features [23] or recursive and multiresolution implementation of LK algorithm [24] may be used. The SIFT algorithm is slow and very time consuming, therefore we used multiresolution implementation of LK algorithm points to extract point

features called good features to track (GFTT) and matching them. The point features and their matches are used for the estimation of four virtual points coordinates during consecutive video frames. Since the matching points for some feature points may be erroneous we used robust RANSAC algorithm to estimate 2D projective transformation between consecutive frames. Then the calculated 2D transformation is employed to calculate the location of virtual points at current frame. The 2D coordinates of virtual points are used to estimate the 3D rotation and translation matrix by 2D to 3D point correspondences method. Then we calculate rotation angles around x, y and z axes using the calculated rotation matrix. Finally in the third module, rotation angles around x and y axes are 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.

III. VISUAL TRACKING MODULE

First module of the proposed algorithm is visual tracking module which receives video frames from webcam. The module has two responsibilities including extraction face area and detection and tracking of point features.

A. *Extraction of the Face Area*

For extraction of the face area in the first frame, we used of the Haar-like features and AdaBoost algorithm presented by Viola and Jones [22]. To make the algorithm real-time we apply the face detection algorithm at first frame and track the face area in the next frames using the proposed tracking algorithm. The tracking algorithm has the disadvantage of accumulative error. To handle this problem we frequently detect the face area each 10 frames and correct the face area.

B. *Detecting and Tacking Point Features*

For detection of the point features on the face area we used the Good Feature to Track (GFTT) algorithm. The extracted point features are then matched using recursive and multi-resolution implementation of LK algorithm. We also tested other algorithms like SIFT features, where its accuracy and speed was not proper for reality of time purpose.

IV. PROCESSING MODULE

The second module of the proposed algorithm is the processing module. As mentioned before, the 2D coordinates of the four virtual points in each frame is calculated in this module.

A. *2D Ttransformation Model Estimation*

To calculate the 2D coordinates of the four virtual points in each frame, we first calculate 2D transformation model between consecutive frames. Different transformation model may be used for this purpose. Since the four virtual points are considered to be located on a plane, we used 2D projective transformation model in our algorithm which completely covers 3D rotation and translation for points on a plane. Since some matching points may be erroneous, we used RANSAC algorithm [21] to remove outliers and calculate transformation model as follows:

- Select M set of point features and their matches which each set contains four point features and their matches.
- Calculate parameters of projective model for each set.
- Calculate the total number of consistent matches for each projective model.
- Select model with more consistent number of matches.

B. *Applying Transformation Model to Virtual Points*

The projective model obtained in the section IV.A, is applied to the coordinates of four virtual points in previous frame to obtain the coordinates of virtual points in current frame as follow:

$$x_i' = \frac{ax_i + by_i + c}{dx_i + ey_i + 1} \quad (1)$$

$$y_i' = \frac{fx_i + gy_i + h}{dx_i + ey_i + 1} \quad (2)$$

where (x_i, y_i) are coordinates of virtual points in previous frame, (x_i', y_i') are coordinates of virtual points in current frame and *(a ,b, c, d, e, f, g)* are projective model parameters. Fig. 3 shows the results of applying transformation model to virtual points.

Figure 3. The location of four virtual points after applying transformation model

C. *Estimating 3D Head Pose from 2D to 3D Point Correspondences*

1) Coordinates System*:*

Fig. 4 shows different coordinate systems used for 3D head pose estimation.

The definitions for different coordinate systems are as follows.

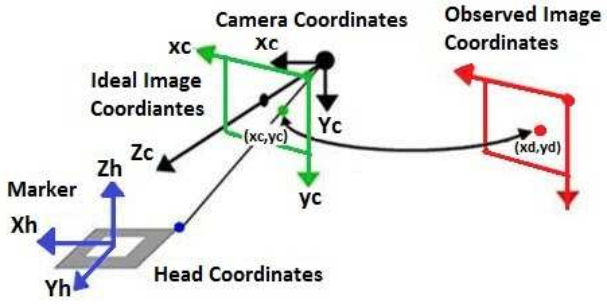

Figure 4. Different coordinate systems for 3D head pose estimation

a) Head coordinate system:

The system is used for specifying 3D coordinates of four virtual points on the user head. We assumed that all virtual points on the face are located on the same plane, therefore the origin of the head coordinate frame, X_h and Y_h axes lie on the plane containing virtual points, while the Z_h axis is perpendicular to the plane. Since all virtual points are in the X_h Y_h plane, their Z values are zero in head coordinate system.

b) 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[25]:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = T_{ch} \begin{bmatrix} X_h \\ Y_h \\ Z_h \\ 1 \end{bmatrix} \tag{3}$$

$$T_{ch} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

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.

c) 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 is used as follows[25]:

$$\begin{bmatrix} hx_I \\ hy_I \\ h \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ 0 & P_{22} & P_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = P \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{5}$$

where (x_I,y_I) are ideal image coordinate (without lens distortion) and P_{11} to P_{23} are camera parameters.

d) 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[26]:

$$d^2 = (x_I - x_{d0})^2 + (y_I - y_{d0})^2 \tag{6}$$

$$p = 1 - kd^2 \tag{7}$$

$$x_o = p(x_I - x_{d0}) + x_{d0} \tag{8}$$

$$y_o = p(y_I - y_{d0}) + y_{d0} \tag{9}$$

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

2) 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}. We assumed that the distance between four virtual points are known and used it for the estimation of the T_{ch} [25]. T_{ch} is estimated in two steps which are 1- rotation matrix estimation and 2-translation vector estimation.

a) Estimating rotation matrix

When two parallel lines between virtual points in 3D space are projected on the image plane, the equations of line in the ideal image coordinates are as follows [25]:

$$a_1x + b_1y + c_1 = 0, \quad a_2x + b_2y + c_2 = 0 \tag{10}$$

Given the perspective projection matrix P in Eq. 5 that is obtained by the camera calibration, it can be shown that the equation of the planes passing these two lines in camera coordinate system can be represented as follows respectively[25]:

$$a_1P_{11}X_c + (a_1P_{12} + b_1P_{22})Y_c + (a_1P_{13} + b_1P_{23} + c_1)Z_c = 0 \tag{11}$$
$$a_2P_{11}X_c + (a_2P_{12} + b_2P_{22})Y_c + (a_2P_{13} + b_2P_{23} + c_2)Z_c = 0 \tag{12}$$

The normal vectors of these planes are defined as $\mathbf{n_1}$ and $\mathbf{n_2}$ as follows:

$$n_1 = \begin{bmatrix} a_1P_{11} \\ a_1P_{12} + b_1P_{22} \\ a_1P_{13} + b_1P_{23} + c_1 \end{bmatrix} \tag{13}$$

$$n_2 = \begin{bmatrix} a_2P_{11} \\ a_2P_{12} + b_2P_{22} \\ a_2P_{13} + b_2P_{23} + c_2 \end{bmatrix} \tag{14}$$

It can be shown that the normal vector to both $\mathbf{n_1}$ and $\mathbf{n_2}$ vectors, defined as the outer product of $\mathbf{n_1} \times \mathbf{n_2}$, is the representation of one of the head coordinate axis in camera coordinate frame. Since we have two sets of parallel lines obtained from virtual points, we can calculate the representation head coordinate axis in camera coordinate frame which give rise to the calculation of rotation matrix as follows:

$$\mathbf{u_1} = \mathbf{n_1} \times \mathbf{n_2}, \ \mathbf{u_2} = \mathbf{n_3} \times \mathbf{n_4} \tag{15}$$

$$\mathbf{R_1} = \mathbf{u_1}, \quad \mathbf{R_2} = \mathbf{u_2}, \quad \mathbf{R_3} = \mathbf{u_1} \times \mathbf{u_2} \tag{16}$$

$$\mathbf{R_{3\times3}} = \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \\ \mathbf{R_3} \end{bmatrix} \tag{17}$$

$$\mathbf{R_i} = [\mathbf{R_{i1}} \quad \mathbf{R_{i2}} \quad \mathbf{R_{i3}}] \tag{18}$$

where $\mathbf{n_3}$ and $\mathbf{n_4}$ are normal vectors of planes corresponding to the two other parallel lines of virtual points.

b) Estimating translation vector

To calculate the translation vector we present the problem as optimization problem as follow[26]:

$$err = \frac{1}{N}\sum_{j=1}^{N}\left\{\left(x_j - \tilde{x}_j\right)^2 + \left(y_j - \tilde{y}_j\right)^2\right\} \tag{19}$$

where (x_j,y_j) is the estimated coordinates of virtual points, N is the number of virtual points, and $(\hat{x}_j, \hat{y}_j)$ are calculated using the following equation:

$$\begin{bmatrix}\hat{x}_j\\ \hat{y}_j\end{bmatrix} = F\left[C \times T_{ch}\begin{bmatrix}X_{hj}\\ Y_{hj}\\ Z_{hj}\\ 1\end{bmatrix}\right], \qquad j = 1{:}\,N \tag{20}$$

where F presents the lens distortion function and $[X_{hj}, Y_{hj}, Z_{hj}]$ are the 3D coordinates of virtual points in head coordinate system. By solving the optimization problem of Eq. 20 and having rotation matrix parameters from previous stage, it is possible to obtain T_{ch} matrix completely which represents 3D head pose.

V. Mouse Pointer Control

When the 3D head pose including rotation matrix and translation vector 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:

(21)

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 the following equations:

$$\beta = 360 \times \sin^{-1}\left(\frac{R_{13}}{2\pi}\right) \tag{22}$$

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

Equations (24, 25) are utilized for the translation of the rotation angle into mouse movement on the screen.

$$\text{mouse}_x = \left(\frac{\beta}{\beta_{max}}\right) \times \left(\frac{w}{2}\right) + \left(\frac{w}{2}\right) \tag{24}$$

$$\text{mouse}_y = \left(\frac{\alpha}{\alpha_{max}}\right) \times \left(\frac{h}{2}\right) + \left(\frac{h}{2}\right) \tag{25}$$

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 and y axes respectively. In this paper $\alpha_{max} = 10^o$ and $\beta_{max} = 30^o$ are supposed.

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 0.5s or 5 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. 5 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 5. Pseudo code for mouse click events implementation

VI. Experimental Results

We implemented the proposed algorithm using a Microsoft Visual C++ program and Intel's OpenCV library for computer vision [27]. To solve the optimization problem of Eq. 19 and showing the result as 3D graphics, we utilized the functions provided by ARToolKit [28]. 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.

Fig. 6 shows the camera mouse 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 (24,25) the resolution (width and height) of the computer screen can be set arbitrarily in the proposed system, therefore we used the resolution of 600*600 in our experimental tests.

Figure 6. Camera mouse implementation using a laptop PC

Fig. 7 shows the calculated head coordinate system displayed using ARToolKit functions.

Figure 7. The calculated head coordinate system displayed using ARToolKit functions

We evaluated the performance of our system on five users and different lighting conditions. We also tested our system with SIFT and LK trackers. 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. 8 shows the true position and the position of the mouse pointer obtained using the proposed method and nose tracking algorithm for a typical frame of a test video. The results with SIFT and LK tracker for all frames of a typical user are shown in Fig. 9 and Fig. 10. As it is obvious from Fig. 9 and Fig. 10, the proposed algorithm with LK tracker has better results. Table I summarizes the results of camera mouse algorithm for five users with different test videos. The average time to process a frame for the proposed algorithm with LK tracker is about 0.1s while the processing time for proposed algorithm with SIFT is 0.58s which means the proposed system is capable of processing 10 frames/sec with LK tracker and 1.7 frames/sec with SIFT tracker. The table shows the mean absolute error (MAE) and average deviation (AD) of MAE for mouse positions in x and y directions. These parameters are defined as follows for x direction:

$$\mathrm{MAE} = \frac{\sum \mathrm{ex_i}}{\mathrm{N}} \tag{26}$$

$$\mathrm{AD} = \frac{\sum |\mathrm{ex_i} - \mathrm{MAE}|}{\mathrm{N}} \tag{27}$$

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

TABLE I shows that the error for the proposed algorithm with LK tracker is approximately twice less than camera mouse implementation using proposed algorithm with SIFT tracker and nose tracking. Also TABLE II confirms this result for α and β angles.

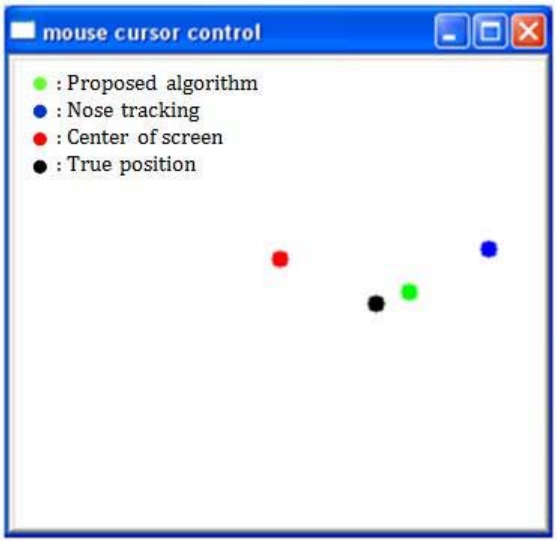

Figure 8. Position of mouse pointer on the screen for a typical frame of a test video

VII. 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 correspondences. The algorithm is real time and can process 10 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 virtual points on the face constituting a rectangle. For this purpose we estimated transformation model using of RANSAC algorithm and LK feature tracker between two consecutive frames.

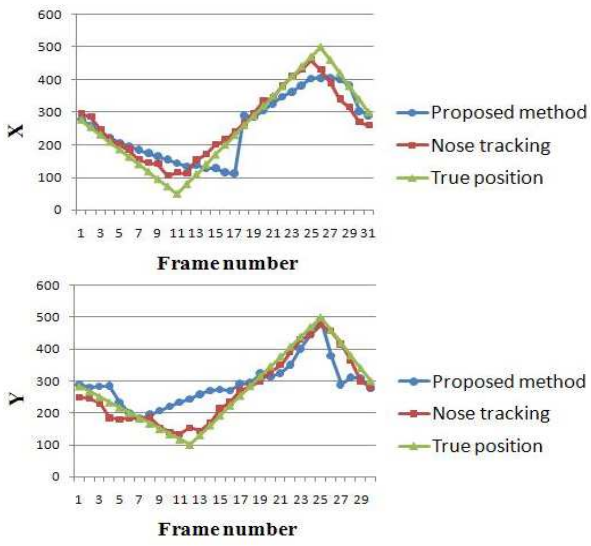

Figure 9. Mouse position in x and y directions for different frames of a typical test video with SIFT tracker

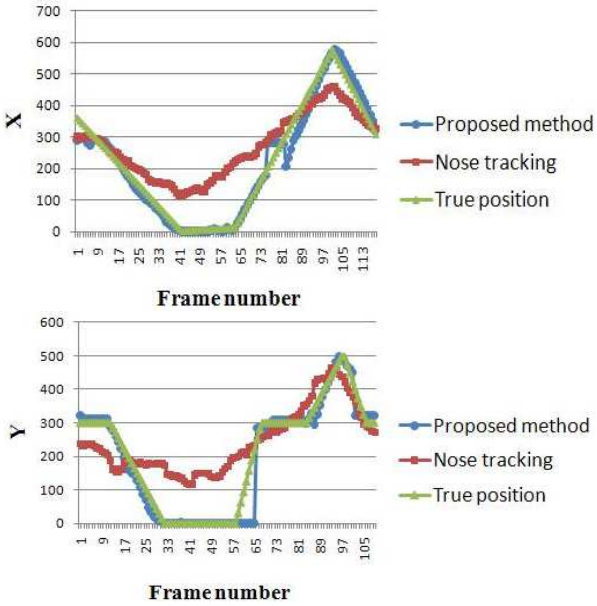

Figure 10. Mouse position in x and y directions for different frames of a typical test video with LK tracker

TABLE I. CAMERA MOUSE ERROR FOR FIVE USERS IN X AND Y DIRECTIONS

Mouse Direction	Method / Error	Proposed method using SIFT tracker	Proposed method using LK tracker	Nose tracking [18]
X(pixel)	MAE	54.91	24.21	73.73
	MAE(%)	9	4.03	12.28
	AD	46.09	17.94	44.62
Y(pixel)	MAE	55.40	22.60	57.40
	MAE(%)	9.23	3.76	9.56
	AD	43.12	19.90	37.35

TABLE II. CAMERA MOUSE ERROR FOR FIVE USERS IN α AND β ANGLES

Pose angle	Method / Error	Proposed method using SIFT tracker	Proposed method using LK tracker
α (Deg)	MAE	1.74	0.75
	AD	1.38	0.65
β (Deg)	MAE	5.65	2.44
	AD	4.63	1.79

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 Transl. 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. 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. Optics & Laser Technology Conf., pp. 1218–1225, 2007.

[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, and 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. IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, 2008.

[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. 2008 International Conference on Computer Vision Theory and Applications., pp. 297-304.

[16] Y. Shin, J. S. Ju, and 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, and 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, and 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] M. Nabati and A. Behrad, " Camera mouse implementation using 3D head pose estimation by monocular video camera and 2D to 3D point and line correspondences," submitted to 6th Iranian Conference on Machine Vision and Image Processing (MVIP 2010), 2010.

[21] O. Chum, "Two-view geometry estimation by random sample and consensus," PhD thesis, 2005.

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

[23] D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110 2004.

[24] J. Y. Bouguet, “Pyramidal implementation of the lucas kanade feature tracker description of the algorithm”, Intel Corporation Microprocessor Research Labs, Online: http://robots.stanford.edu/cs223b04/algo_tracking.pdf.

[25] H. Kato and M. Billinghurst, “Marker tracking and HMD calibration for a video-based augmented reality conferencing system”, In Proc. of the 2nd Int. Workshop on Augmented Reality (IWAR 99), October 2000.

[26] 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.

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

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