Monocular Gaze Tracking System for Disabled People Assistance

Daniel Rodrigues Carvalho, Maurcio Pamplona Segundo
Universidade Federal da Bahia - UFBA
Salvador, Brazil
Intelligent Vision Research Lab
Email: odanrc@yahoo.com.br, mauricio@dcc.ufba.br

***Abstract*—With the difficulty of computer use by people with locomotion deficiency in sight, the objective of this paper is to create a low cost system to help them use computers by using eye and pupil detection and head orientation algorithms to emulate mouse cursor movement via user's gaze direction. Through techniques as non-rigid face tracking, pupil detection by sclera-pupil contrast, and gaze estimation based on screen distance, a software was created to simulate mouse utilization, allowing translation and clicks based on eye movement. The resulting system presents high accuracy, and improvable points are suggested in the end of the study.**

***Keywords*-Gaze Tracking, Blink Detection.**

I. Introduction

Physical disabilities are a limitation of the muscular or skeletal system, which may be partial or total. They are generally caused by genetic factors, viral or bacterial infections, or trauma. This limitation can make it difficult to carry out daily activities, such as technology usage, feeding, and practice of sports. In order to facilitate the use of computers by people with disabilities, the creation of a mouse emulator software controlled by user's gaze and eye blinking is proposed, making it possible to insert them in technological and informational means.

Similar works have already been developed, but most of them either require the user to be statically positioned (i.e. no head rotation is allowed), or are significantly expensive, due to the use of specialized hardware. Besides that, they usually can not accomplish tasks that require high precision due to the existence of calculation errors caused by lack of accuracy of the feature detectors, or low image resolution.

The proposed system does not fully allow human-computer interaction yet, as it can only emulate left button clicks, but it grants enough independence to accomplish small tasks, like web browsing and watching videos. This paper proposes a way to significantly increase focal point precision, at the cost of convergence time. The proposed method is not computationally expensive and can be implemented in any similar working system.

II. Related work

Comparable works have already been developed and commercialized, which can be classified into two different categories: monocular and stereo. This section will describe these categories, their advantages and difficulties.

A. Monocular Methods

Monocular methods rely on the usage of a single camera. Such methods can be based on fixed USB, mobile, or infrared cameras. Other approaches exist, such as the Electrooculography-based tracking and the use of special lenses inside the eyes, but they will not be discussed in this paper, as they are excessively expensive and intrusive.

1) Fixed USB Cameras: When using a single fixed USB camera, several problems arise, such as the lack of distance and position information, and the low resolution of images. Techniques based on this approach usually apply face detection and tracking algorithms to detect the points of interest on the eye and estimate the head pose [1][2][3]. Although imprecise, these methods are cheap and easy to reproduce, as most personal computers nowadays come with an embedded web camera.

2) Mobile Cameras: Mobile or head-mounted cameras are placed on the head of the subject pointing directly towards the eye, and do not require face tracking or head pose estimation steps. Such systems work by tracking the pupil and estimating the gaze. Techniques as the Stardust algorithm [4], that applies noise reduction by Gaussian Filtering, corneal reflection reduction by radial intensity interpolation, and pupil detection via the RANSAC algorithm, can be used for pupil tracking due to the high quality of the eye image. Besides that, as the eye features are easily detected, the gaze estimation is straightforward. Their disadvantages are being more uncomfortable and more expensive than fixed USB cameras.

3) Infrared Cameras: Systems based on infrared detection are less accessible because they need both an infrared emitter and an infrared camera. They work by analyzing positional relationship between the corneal reflection generated by the infrared light and the pupil. Some techniques have already been used to create commercial gaze trackers [5][6]. Fujitsu's eye tracker, for example, estimates the gaze by detecting corneal reflection candidates (white circles) and pupil candidates (black circles), and then applies consistency rules, such as the relationship between pupil, corneal reflection, emitter and camera positioning, and the relation with previous results.

B. Stereo Methods

The use of a single camera does not generate enough information to calculate the depth of the points and the gaze direction accurately. This problem can be overridden if two or more cameras are used by applying triangulation [7][8], weights to the possible points [9], or similar approaches to the recorded images. These methods provide more information regarding the face, and usually generate smaller errors. Since these approaches require the use of more than one camera, they are more expensive and require special calibration for camera placement.

C. Proposed Method

Although stereo methods and methods based on infrared or mobile cameras are more precise, in this system techniques based on a fixed USB camera are implemented due to its lower cost and easier accessibility. The algorithm to improve precision proposed on this paper, however, can be used with these other methods.

III. Technique Description

In order to generate high accuracy, the system is divided in three main parts: gaze tracking, where facial features and orientation are detected in pursuance of focal point estimation, blink detection, and fine tuning, where the gaze estimation error is reduced. Fig. 1 presents an overview of the algorithm, and more details are given in the following subsections.

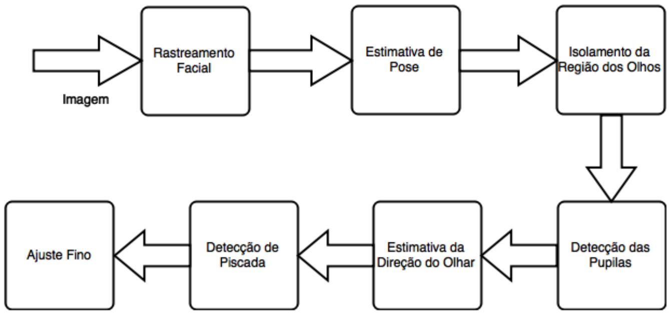


Fig. 1. Technique Overview

A. Gaze Tracking

The FaceTracker library is used for pose estimation. This library detects the face and then tracks its position and orientation through Convex Quadratic Fitting applied to the space-time domain [10]. Then, a library implementing Uricar *et al.*'s algorithm [11] named Flandmarks was used to obtain a better estimation of the eye region and correct eye canthi position. It uses a structured output classifier based on Deformable Part Models to generate a graph containing eyes canthi, tip of the nose, face center and mouth corners. From the image of the face, features and the displacement vector are calculated, and then the restrictions related to the graph's format are applied through dynamic programming, in order to detect the interest points.

The facial features detected by the trackers described above are shown in Fig. 2: the filled circles are the features acquired by the Flandmark library, the red points are landmarks detected

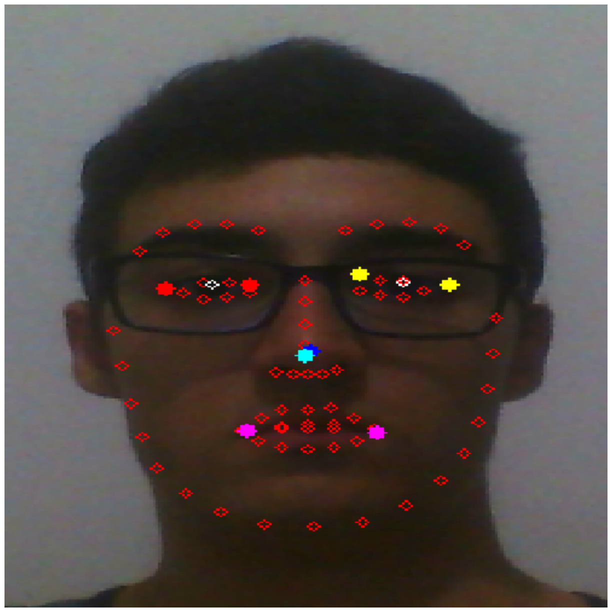
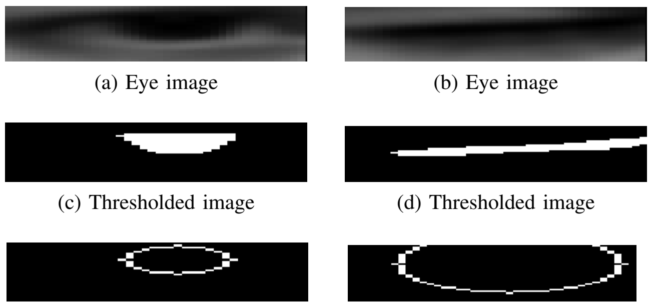


Fig. 2. Facial Features from both detectors employed in this work

by FaceTracker, and the white circles are the estimated pupils locations.

For the sake of efficiency and effectiveness, an algorithm for pupil tracking and blink detection has been created using an hybrid approach, assuming that pupils are dark and semi-circular. Algorithm 1 presents an overview of the process. First, there is some pre-processing: eye regions are isolated, Gaussian Blur is applied to reduce image noise, and the image is normalized to increase contrast. Then, the darkest point of the image is found, and a threshold is applied to the image, in order to isolate the dark areas (*i.e.* areas where the pupil is likely to be located at) as in Fig. 3.



(a) Eye image
(b) Eye image
(c) Thresholded image
(d) Thresholded image
(e) Minimum enclosing circle
(f) Minimum enclosing circle

Fig. 3. Pupil and blink detection

In sequence, the algorithm for contour detection described by Suzuki *et al.* [12] is applied, and the minimum enclosing circles are calculated. The circle with biggest radius is selected and its area is compared to the area of its corresponding contour. If both areas have similar value, a pupil is likely to be located inside that circle (figure Fig. 3e). Otherwise, the eye is presumably closed (figure Fig. 3f).

Algorithm 1 Algorithm for Pupil and Blink Detection

```
procedure PUPILBLINKDETECTION
    Gaussian Blur
    Normalization
    Find Darkest Point of Image
    Apply Threshold
    Find Contours
    Calculate Minimum Enclosing Circles
    $circle \leftarrow$ Find Circle With Biggest Radius
    $area \leftarrow$ Calculate Filled Area of Circle With Biggest Radius
    if $area$ * $kArea >circle$ area then
        $pupil \leftarrow circle$ center
        $blink \leftarrow FALSE$
    else
        $blink \leftarrow TRUE$
```

The distance from screen can not be precisely estimated due to the use of a monocular camera. After removing the head rotation distortion, the distance d_1 is calculated based on the difference between the measured eye size s_1 and the average human eye size s_2 at a known distance d_2, according to the following equation:

$$d_1 = s_2 * d_2 / s_1$$

To determine the focal point, the 3D coordinates of the center of the eye globes are estimated by applying head rotation to the detected 2D eye centers. Then, two rays passing through these centers and the pupils are traced, and the points where these rays intercept the screen plane are averaged to determine the focal point. The head rotation matrix M_r is initially calculated through the pose estimation conducted previously, according to Equations 1, 2, 3 and 4.

$$rot_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos pose_x & -\sin pose_x \\ 0 & \sin pose_x & \cos pose_x \end{pmatrix} \tag{1}$$

$$rot_y = \begin{pmatrix} \cos pose_y & 0 & \sin pose_y \\ 0 & 1 & 0 \\ -\sin pose_y & 0 & \cos pose_y \end{pmatrix} \tag{2}$$

$$rot_z = \begin{pmatrix} \cos pose_z & -\sin pose_z & 0 \\ \sin pose_z & \cos pose_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$M_r = rot_z * rot_y * rot_x \tag{4}$$

Then the 3D positions of the facial landmarks obtained previously (pupils, canthi and eye centers) are calculated based on the head rotation, assuming that the human eye is a perfect sphere with diameter approximately equal to the distance between eye canthi. To this end, for each facial landmark the relative position of the points is calculated, then they are rotated based on the rotation matrix and their absolute position is recalculated 5. The base point is the center of the 3D Cartesian space on which the rotation will be applied, *i.e.*, the facial landmark closest to the camera, as it is most likely to have been detected correctly.

$$pos_{3D} = (pos_{2D} - basePoint) * M_r + basePoint \tag{5}$$

Subsequently both the horizontal (α) and the vertical (β) angles between the center of the eye (C) and the pupil (P) are calculated using the Equations 6 and 7.

$$\alpha = \arctan \frac{P_x - C_x}{C_z} \tag{6}$$

$$\beta = \arctan \frac{P_y - C_y}{C_z} \tag{7}$$

Finally, the on-screen X and Y positions are estimated based on Equations 8 and 9.

$$X = C_x - \tan \alpha * (headPos_z + C_z) \tag{8}$$

$$Y = C_y - \tan \beta * (headPos_z + C_z) \tag{9}$$

It is worth noting that, for calculation correctness, all coordinates should be on the same measurement system. To simplify and make computations easier, both facial points (in video resolution coordinates) and the distance from screen (in centimeters) were mapped to screen coordinates by rule of three.

B. Blink Detection

As mentioned before, the pupil detection algorithm can be used for blinking detection as well, but it generates many false positives due to lack of illumination, lack of sclera-pupil contrast, or when the eyebrows are within the extracted eye region. To avoid these errors, an eye Haar cascade classifier[1] is applied for eye detection, and if it does not find a match, then the eye is considered to be closed. Both these approaches are then combined by an AND operation to provide the estimated state. As the human eye constantly blinks for eye lubrification, a threshold buffer is applied to the estimated closure state, so a blink command is only confirmed if it is longer than a regular blink.

C. Fine Tuning

Despite being able to estimate the gaze, the error accumulation prevents the system from working correctly, as the desired focus point can be far from the estimated location. To fix this problem, the use of a MxN magnification grid is proposed. An overview of the algorithm is presented in Algorithm 2.

[1]A tree based frontal eye detector with handling for eyeglasses created by Shameem Hameed (http://umich.edu/~shameem).

Algorithm 2 Algorithm for Gaze Estimation Fine Tuning

```
procedure FINETUNING
    Magnified Region Rendering
    $focalPoint \leftarrow$ GAZETRACKING
    $blinkLeft \leftarrow$ Left Eye Blink detection
    $blinkRight \leftarrow$ Right Eye Blink detection
    if $blinkLeft \vee blinkRight$ then
        if Significant Region Size then
            ZOOM($focalPoint$)
        else
            CLICK($focalPoint$)
```

The grid displays and divides the desktop area in smaller regions, the zoom areas. Each zoom area, when magnified, becomes the current view screen, which is then divided again until there is enough precision (*i.e.* its width or height are below a desired threshold), moment at which a click action is emitted.

For maximum convergence velocity, a high dimensional grid should be used. In the proposed system, due to the lack of precision of the gaze tracking routine, a 3x3 grid is used, and the zoom area being focused is iteratively magnified to achieve high precision. Fig. 4 shows a complete click sequence.

Sometimes, when the user is located far from the camera, the vertical eye resolution is too low and is not enough to differentiate many vertical positions, requiring the user to move the head to the desired direction. In order to allow people with low to zero head movement to use the system, vertical gaze estimation data is not used, which causes the mouse cursor to move sideways only. To switch between rows, a threshold is applied to both vertical edges of the monitor and if the gaze position is too close to the left or right screen border, the line being analyzed is changed, respectively, to the previous or next one.

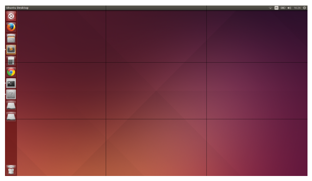
D. Filtering

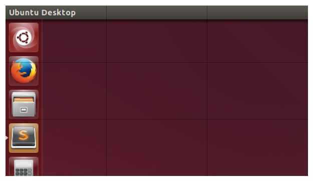
Due to noise and low image quality, the location of facial landmarks and pupil centers tend to oscillate. Therefore, to improve the estimation of these points of interest, the data is filtered through a mean buffer: a FIFO of limited size that contains the detected values and estimates the probable desired position through the average of these values.

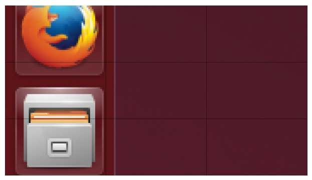
The buffer size must neither be too small nor too big to avoid not reducing the noise interference or causing a delay during video tracking. The filtering process allowed a considerable improvement, reducing the interference of errors in the location of landmarks.

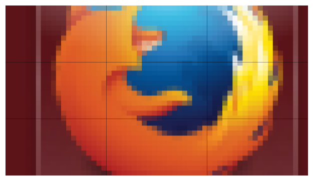
IV. Experimental Results

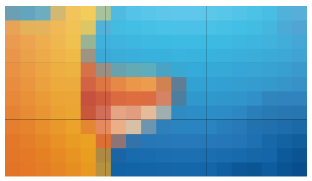
In order to validate the technique, an online system has been developed to estimate click accuracy. It consists in showing twenty targets 50x50 pixels wide on the screen in random locations, and the user was required to click as close to the center of the target as possible. Fig. 5 shows the online system during execution.

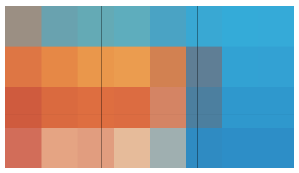

(a) Initial screen


(b) First magnification


(c) Second magnification


(d) Third magnification


(e) Fourth magnification


(f) Fifth magnification

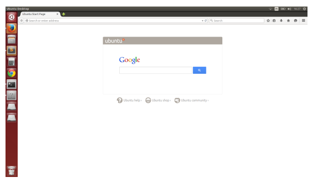

(g) Sixth magnification, which results in a click

Fig. 4. Sequence of magnifications that result in a click

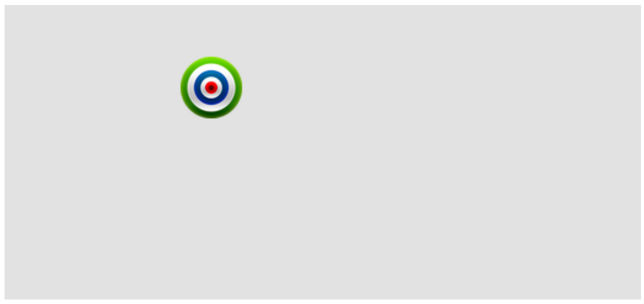
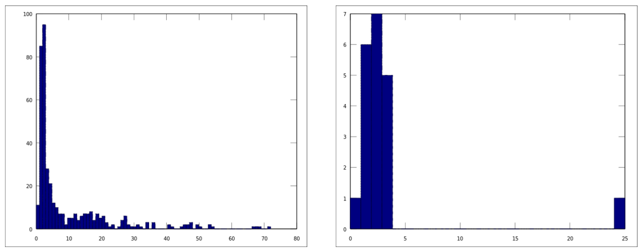


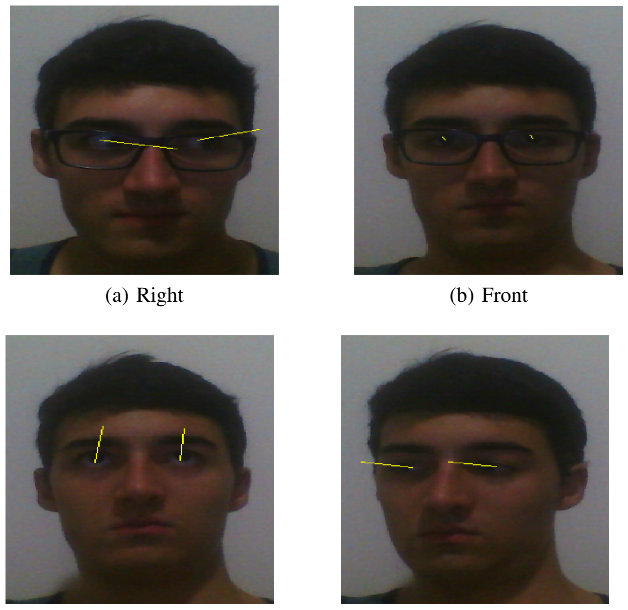
Fig. 5. Online system: The user clicks at the center of the target to provide precision estimation.

The experiment was initially conducted on 20 different users using usual clicking methods (e.g. mouses and touchpads). The medium error of the results, which is the average user precision, was found to be 3 pixels. The same experiment was then applied to a single user using the proposed system, producing a median error of 2 pixels. These results, presented in Fig. 6, indicate that the system is promising, and that it may be used as a substitute for conventional clicking methods to help disabled people, but more work needs to be done to make it more robust.



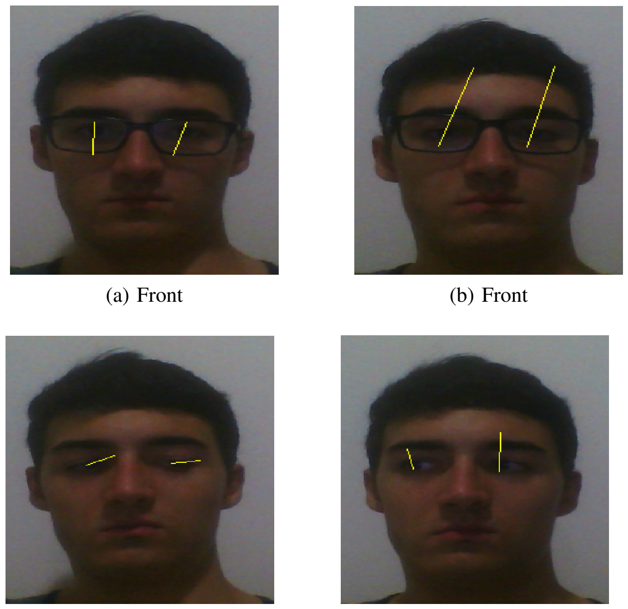
(a) Histogram of user precision using conventional clicking methods. (b) Histogram of precision achieved by the system

Fig. 6. Precision Distribution of Experiments. The X axis is the distance from the center of the target, and the Y axis represents the amount of users



(a) Right (b) Front (c) Up (d) Left

Fig. 7. Correct Gaze Estimation



(a) Front (b) Front (c) Down-Right (d) Left

Fig. 8. Incorrect Gaze Estimation

A. *Performance*

The system has been tested on a 2.3GHz Intel Core i7-3615QM processor and a NVIDIA GeForce GT 630 M GPU running Linux. The monitor resolution was 1600x900, and the camera resolution was 640x480.

As the detection part of the FaceTracker is more computationally expensive than the tracking, there is a performance drop when no users are detected. During normal usage, the system performed well, achieving an average rate of 25 frames per second.

B. *Quality*

The errors generated in each step of the system accumulate, creating an incorrect final gaze estimation and, thereafter, a long convergence time is required. The pose estimation generated by FaceTracker provided robust results, with an average error of less than five degrees. However, sometimes the tracker looses the correct position of the face, so face detection is forced every 10 seconds to bypass this problem, at the cost of execution speed loss.

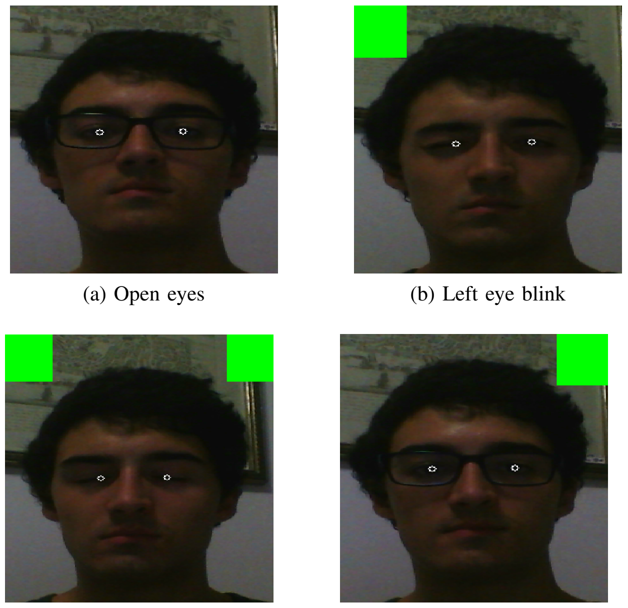
The detection of eye regions and respective points of interest by Flandmark, after being filtered to overcome data inconsistency, was accurate, generating significant errors only in faces rotated on the Z axis (depth) or with rotation higher than 35 degrees on the Y axis (vertical), due to self-occlusion in the region of the face.

The algorithm for pupil tracking provides high accuracy in poorly lit environments, but may give erroneous results when there are occlusive elements in the image, such as hair and glasses. Furthermore, small contrast between the pupil and sclera, or excessive sagging of the upper eyelids prevent detection. Gaze estimation results are shown in Fig. 7 and Fig. 8.

The eye blink detection fails when there is high self-shading of the eye region, and the occurrence of false positives disrupts the system as they can activate commands in undesired moments. To improve the command detection, possible solutions are being studied, such as the use of Haar classifiers trained for closed eyes or the usage of speech recognition algorithms. Some blink detection results are shown in Fig. 9.

The filtering step enhances the estimation results, as can be seen in Fig. 10, which presents the results of a test where a user had to look at a fixed point on the screen. The continuous line represents the estimated point, and the dashed line represents the filtered point. It's valid to note the precision difference between the axis: while on the horizontal axis the discrepancy between the estimated point and the real point is inferior to 10%, the vertical axis' error exceeds 20%.

The implementation of the gaze correction through grids and magnifications and the elaboration of mouse control have been successfully carried out. Consequently, even though the



(a) Open eyes (b) Left eye blink

(c) Left and right eyes blink (d) Incorrect blink detection

Fig. 9. Blink detection. A green square is drawn on the side in which the blink is detected.

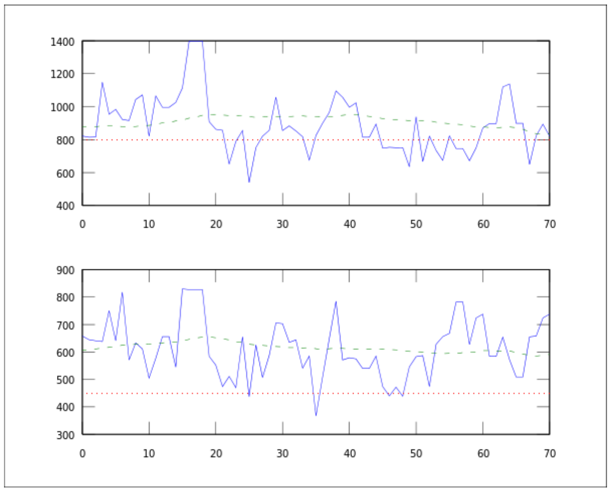


Fig. 10. Comparison between the estimated focal points with and without the filtering step. The dashed line represents the estimated point after filter application and the dotted line represents the real position of the point (ground-truth).

estimation of the gaze direction presented errors generated by the previous steps, there is feasibility of using the system, and the desired points can be obtained with accuracy near 100%, as shown in figure Fig. 6(b).

V. Conclusion and Future Work

In this paper the concepts of gaze tracking have been applied, and a new technique to increase the accuracy of focal point estimation has been introduced, which reduces errors caused by the lack of precision of landmarks detectors at the cost of execution time.

The convergence time can be reduced by increasing the overall effectiveness of the trackers with methods such as Supervised Descent Method [13] and Daugman's Integro-Differential Operator [14]. Improvements in accuracy can also be achieved by the use of Kalman Filters for head pose and landmarks location [15][16][17].

In the future, other operation modes can be added, allowing the user to effectively use the computer without human-assistance. Such modes must simulate possible mouse and key-board commands, as double clicking, right clicking, window switching and typing. The latter can be better achieved by the use of a word prediction algorithm, as suggested by Ward *et al.* [18].

References

[1] J.-G. Ko, J.-H. Yu, and K.-I. Jeong, "Facial feature detection and head orientation based gaze tracking," in *Proceedings of the 2005 WSEAS international conference on Dynamical systems and control.* World Scientific and Engineering Academy and Society (WSEAS), 2005, pp. 415–420.

[2] R. Valenti, N. Sebe, and T. Gevers, "Combining head pose and eye location information for gaze estimation," *Image Processing, IEEE Transactions on*, vol. 21, no. 2, pp. 802–815, 2012.

[3] G. Boulay, "Eye pose tracking & gaze estimation," *Institut National des Sciences Appliquées de Lyon, France. Département Télécommunications–Services & Usages*, 2008.

[4] D. Li, J. Babcock, and D. J. Parkhurst, "openeyes: a low-cost head-mounted eye-tracking solution," in *Proceedings of the 2006 symposium on Eye tracking research & applications.* ACM, 2006, pp. 95–100.

[5] Tobii, "Tobii."

[6] Fujitsu, "Fujitsu's eye tracker."

[7] J. Chen, Y. Tong, W. Gray, and Q. Ji, "A robust 3d eye gaze tracking system using noise reduction," in *Proceedings of the 2008 symposium on Eye tracking research & applications.* ACM, 2008, pp. 189–196.

[8] S. Kawato and N. Tetsutani, "Detection and tracking of eyes for gaze-camera control," *Image and Vision Computing*, vol. 22, no. 12, pp. 1031–1038, 2004.

[9] N. M. Arar, H. Gao, and J.-P. Thiran, "Robust gaze estimation based on adaptive fusion of multiple cameras," in *Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition*, no. EPFL-CONF-206164, 2015.

[10] S. Lucey, Y. Wang, J. Saragih, and J. F. Cohn, "Non-rigid face tracking with enforced convexity and local appearance consistency constraint," *Image and vision computing*, vol. 28, no. 5, pp. 781–789, 2010.

[11] M. Uřičář, V. Franc, and V. Hlaváč, "Detector of facial landmarks learned by the structured output svm," *VISAPP*, vol. 12, pp. 547–556, 2012.

[12] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[13] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on.* IEEE, 2013, pp. 532–539.

[14] J. Daugman, "How iris recognition works," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, no. 1, pp. 21–30, 2004.

[15] O. V. Komogortsev and J. I. Khan, "Kalman filtering in the design of eye-gaze-guided computer interfaces," in *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments.* Springer, 2007, pp. 679–689.

[16] W. Abd-Almageed, M. S. Fadali, and G. Bebis, "A non-intrusive kalman filter-based tracker for pursuit eye movement," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 2. IEEE, 2002, pp. 1443–1447.

[17] T. Grindinger, "Eye movement analysis & prediction with the kalman filter," Ph.D. dissertation, Clemson University, 2006.

[18] D. J. Ward, A. F. Blackwell, and D. J. MacKay, "Dashera data entry interface using continuous gestures and language models," in *Proceedings of the 13th annual ACM symposium on User interface software and technology.* ACM, 2000, pp. 129–137.