Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

A Low Cost Human Computer Interface based on Eye Tracking

Jonathon B. Hiley, Andrew H. Redekopp, and Reza Fazel-Rezai, *Senior Member, IEEE*

Abstract—**This paper describes the implementation of a human computer interface based on eye tracking. Current commercially available systems exist, but have limited use due mainly to their large cost. The system described in this paper was designed to be a low cost and unobtrusive. The technique was video-oculography assisted by corneal reflections. An off-the shelf CCD webcam was used to capture images. The images were analyzed in software to extract key features of the eye. The users gaze point was then calculated based on the relative position of these features. The system is capable of calculating eye-gaze in real-time to provide a responsive interaction. A throughput of eight gaze points per second was achieved. The accuracy of the fixations based on the calculated eye-gazes were within 1cm of the on-screen gaze location. By developing a low-cost system, this technology is made accessible to a wider range of applications.**

I. Introduction

Conventional means for human computer interactions are manual. The ubiquitous keyboard and mouse provides a convenient interface for the majority of users in the many computer applications. However, many users lack the required manual dexterity to operate a keyboard or mouse effectively. A hands-free system for computer interaction would be a huge benefit to people who had been injured or handicapped and had lost precise control of their hands. It could be integrated into a home network, allowing paralyzed users to control home devices, or simply surf the internet. Also, there are certain applications, such as games, where manual control removes some of the seamlessness desired in the user's interface with the application.

An eye-gaze based system for human computer interaction brings the advantage of more directly converting a user's mental focus into an action on the computer [1]. There is no hand-eye coordination required to control a pointer, as opposed to using a mouse. A user merely needs to look at a screen object for an action to occur.

Video-oculography is a technique which uses an image of the eye to determine eye-gaze [2]. The image may be obtained from a camera mounted on the head, or from a remote camera facing the user. Key features in an image of the eye are located and categorized and the relative position of these features is used to extrapolate a gaze point. This technique relies heavily on image processing and has traditionally been limited due to lack of available computing power. The appeal of this type of system is that it may be totally unobtrusive to the user and can be implemented in a wide variety of ways. While commercial systems based on video-oculography exist, they are cost prohibitive for most applications.

This paper describes the implementation of a low-cost human computer interface based video-oculography. This was accomplished using readily available components without using any specialized equipment.

Manuscript received April 1, 2006. This work was supported in part by the Electrical and Computer Engineering Department of the University of Manitoba.

R. Fazel-Rezai is with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB R3T 5V6 Canada (corresponding author: phone: 1-204-474-9490; fax: 1-204-261-4639; email: fazel@ee.umanitoba.ca).

J. B. Hiley, is with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB R3T 5V6 Canada (email: umhileyj@cc.umanitoba.ca).

A. H. Redekopp, is with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB R3T 5V6 Canada (email: umrede02@cc.umanitoba.ca).

II. Methods

A. Hardware Implementation

To produce the necessary features for the video-oculography, 5 near infrared Light Emitting Diodes (LEDs) were used. The LEDs are individually controlled via the parallel port to create the desired illumination of the user's eye. Four wide angle LEDs were affixed to the corners of the monitor screen area. When illuminated, they generated bright corneal reflections or 'glints' in the image of the eye (see figure 1-a). A fifth higher power LED with narrow emission angle was suspended directly in front of the camera lens to produce a bright-eye effect, which is analogous to red-eye in photography (see figure 1-b). This effect is created by a coaxial light source, as the light is reflected off of the retina and back into the camera lens. Near-infrared light was used to illuminate the user's eye, as it is invisible, safe, and can be easily detected.

The images are captured using a CCD image sensor from a Logitech Quickcam Pro 4000. Since the webcam did not natively support the capture of infrared light, modifications were required. Removing the infrared blocking filter, which is embedded in most digital cameras, allowed the infrared light to reach the photosensitive array. However, due to the high sensitivity of the image sensor to both infrared and visible light, saturation of the sensor array occurs if visible light is not filtered. The magnitude of the saturation is such that it renders the captured images useless as large portions of the image are completely washed out and shown as white. To eliminate this negative effect, a filter is used to block the visible light spectrum. The filter was constructed out of an exposed and developed piece of film. This provides a visible light filter to effectively remove the problem of sensor saturation. A higher resolution image of the eye is obtained by using a standard 35mm camera lens to provide magnification. This is because the 35mm lens focuses the light onto a large area but only a small portion is on the image ¼" CCD sensor. With these modifications, a webcam can be used as an inexpensive camera, which is capable of capturing suitable images for video-oculography.

B. Pupil Detection

Locating the centre of the pupil is a potentially difficult task in image processing. In a normal image of the eye, the pupil has similar intensity levels compared to its surrounding region, the iris. To assist in locating the pupil, the bright-eye effect is exploited to increase contrast between pupil and iris. Further enhancement is obtained by producing a difference image between one bright-eye image and one dark-eye image taken close together in time as suggested by [3]. As long as overall lighting power is approximately equal between bright and dark frames, the facial region will be similar intensity and will fall to near zero in the difference image (figure 1-c). Since the facial region almost completely falls away, the pupil is isolated in the difference image, making centre detection much easier. The difference image is then converted to binary based on a pre-determined threshold. The large intensity difference between the pupil and background makes the segmentation process fairly insensitive to the actual threshold used.

To locate the pupil centre, it is intuitive to use a circle detection method such as the Hough transform or any of its variants. These methods are robust against image noise and when combined with morphological filtering operations, work well for an image with an incomplete circle. To work correctly, an edge detection algorithm must first be run on the image. Although these methods have been shown to work well in the past, computation times may become too great when implemented in a high level programming language.

To reduce computation time, but still provide accurate pupil detection, an algorithm was developed based on a simple centre of mass calculation. Starting with a cropped binary image of the pupil, a centre of mass is located in the image (figure 2-a). Due to obstructions caused by eyelashes, the pupil is not a complete circle, and the centre of mass will be relatively inaccurate. To improve the circular quality of the pupil, a routine was developed which first estimates the radius. Based on this radius, it generates a circular mask and overlays it on the pupil (figure 2-b). This process of finding a new centre, radius, and overlaying a mask is repeated until the centre of mass stops changing within a certain tolerance. The final pupil image will be an almost complete circle (figure 2-d), whose centre of mass will reflect the true circle centre with a reasonable accuracy.

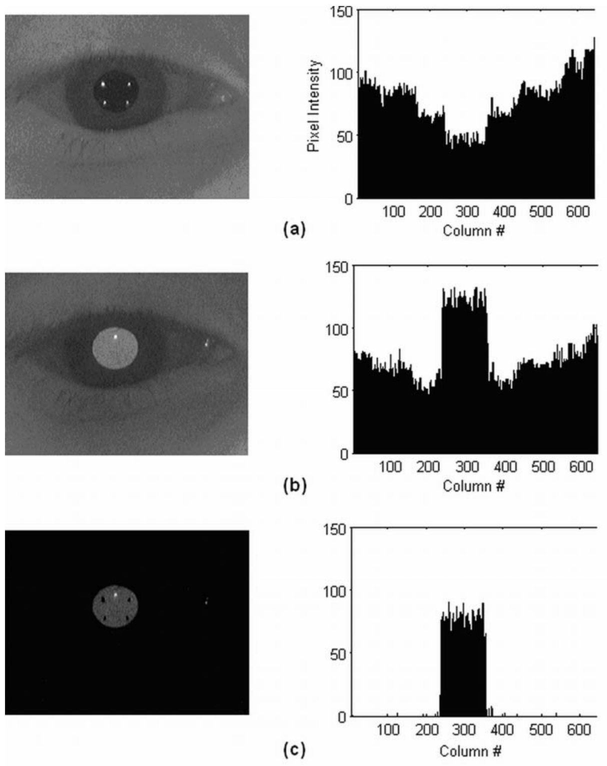

Fig. 1. Pixel intensity by column under controlled lighting conditions.

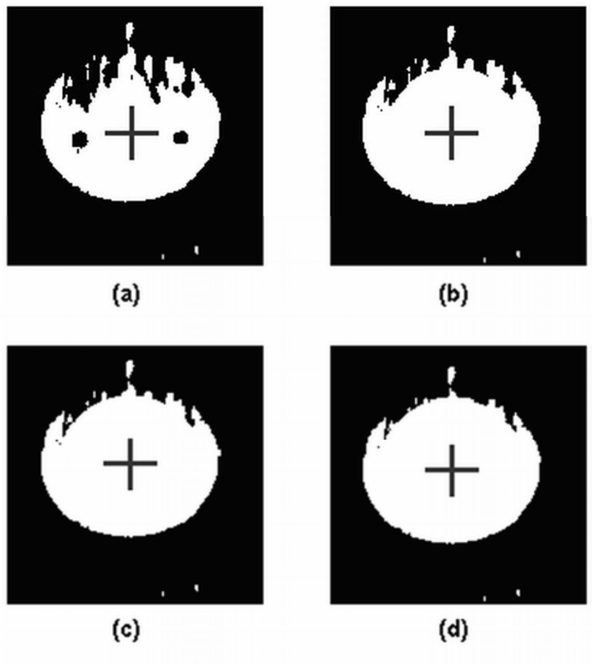

Fig. 2. Iterative routine of improving circular quality of the pupil and calculating centre of mass.

C. Glint Detection

Glint detection is normally an easier process than pupil detection. The glints have a high contrast to their background, which is normally the pupil or iris. By examining the dark eye images alone, high intensity regions are identified, and their centre is returned as the glint location.

Although finding glints is easy, the key challenges are to remove false positives caused by unwanted reflection off the cornea or tear duct and return the position of only the four desired glints. Another problem is that a single glint may be fragmented into two high intensity regions by eyelashes.

To avoid these problems, some heuristics were incorporated into the glint detection algorithm. The image was first cropped around the calculated pupil centre to try and remove reflections from the tear duct. When a high intensity region was identified, all high intensity pixels within a small area around that region were considered to be part of a single glint to avoid fragmentation. As glints were detected, only the four closest to the pupil centre were collected to further reduce the number of false positives.

This routine was shown to work robustly and quickly without the need for difference imaging or any morphological filtering.

D. Projection

Calculation of the on-screen gaze position is calculated using the relative position of the pupil centre and four glints in images of the eye. The purpose is to project a point within a trapezoid in image space to a point within a rectangle in monitor space. The trapezoid is formed by the four glints in the image of the eye. If the user is looking on the screen, their pupil centre will be somewhere within this trapezoid. The four glints in the image correspond to the four LEDs mounted on the corners of the monitor screen, which is a rectangle. The pupil centre corresponds to the user's gaze point on the screen.

To perform this projection, a method as suggested by [4] was used. This involves calculating cross ratios in the image space based on positions of key points. Since cross ratios are constant when collinear points are projected to a different coordinate space, these values can be used to solve for the X and Y components of the gaze point in monitor space.

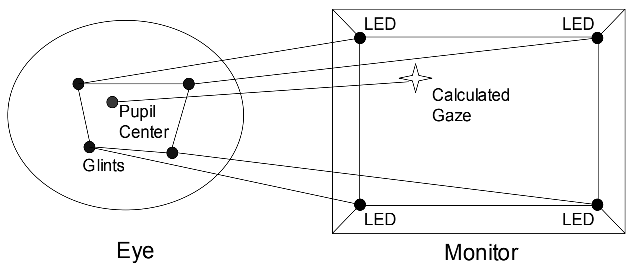

Fig. 3. Projection between an image of the eye and monitor

E. Fixation Analysis

To make the calculated gaze points useful in an application, it is necessary to do some higher level categorization of the results. Also, due to occasional incorrect feature detection or large head movements between bright and dark frames, some calculated gaze points had an acutely large amount of error from the user's actual gaze point.

To deal with these issues, it becomes desirable to group individual gaze calculations into sustained fixations to both remove error points and give intelligent information on where the user is concentrating for certain amounts of time.

When a new gaze point is calculated, it is grouped either into a current fixation, or as part of a possible new fixation. If the new gaze point is within a certain distance of the current fixation, it becomes a part of the current fixation and all other possible fixation points are cleared. If it is not close enough to the current fixation, it becomes a possible new fixation and is grouped with other points which may also be a fixation and are sufficiently close to this one. When one of the possible fixation points gathers a certain minimum number of points, this now considered the current fixation and all of the other possible fixation groupings are cleared.

This method returns information on whether the point was accepted as a new fixation, rejected as an error point, as well as information on the current fixation such as the population and centre. Having this information allows any application to perform action based on having a user fixate on a location for a certain amount of time. For example, a button could be triggered to press when a fixation centered over that button gains a certain minimum population.

III. Results

To test the accuracy and throughput of the system, 24 trials were performed by two different users. All tests were performed on a Pentium© IV, 2.8 GHz PC with a 19 inch CRT monitor of resolution 1280x1024. To setup the tests, 12 designated gaze points were identified at regular spacing on the screen. The user then looked at each point for about 5-8 seconds. Results from a typical trial are shown in figure 4.

One of the concerns when developing the system was that different ambient lighting conditions would affect feature detection. For example, incandescent light gives off much more IR than fluorescent which may produce false glints or lead to difficulties in thresholding. For this reason, tests were performed under three common lighting conditions, incandescent, fluorescent, and darkness. Note that the monitor provides a certain amount of illumination on the subject's faces in all conditions.

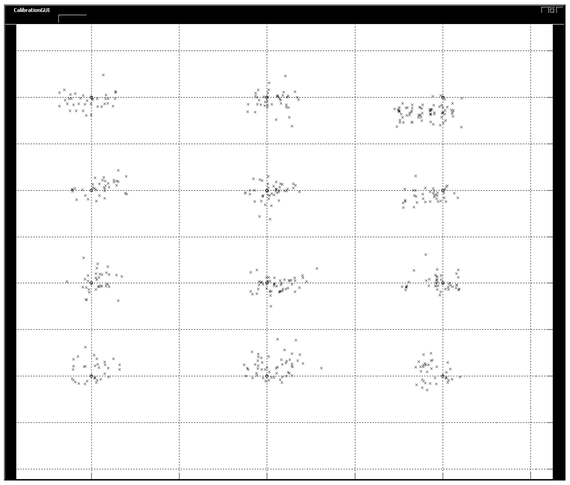
Fig. 4. Results from a typical trial. Each dot is a calculated gaze point.

The system performance was shown to be about 8 calculated gaze points per second. Figure 5 for a breakdown of computation time per calculated gaze point.

The total time from acquiring the image to displaying results in the GUI is 125ms within a MATLAB environment.

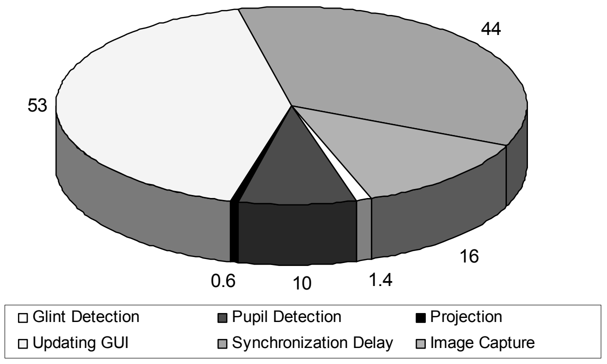

Fig. 5. Breakdown of computation time for one gaze point. Times are shown in ms.

The throughput was limited to 8 gaze points per second due to the maximum frame rate available. Although the webcam is capable of providing 30 images per second, this cannot be realized because the lighting must stay synchronized with image capture. Due to image buffering on the camera as well as on the PC, a delay was introduced to ensure the lighting condition was known in each image.

All feature detection and projection routines required only 12ms on average, which translates to a potential throughput of over 80 gaze points per second. This shows that if the delay was not required the throughput could be greatly improved.

Lighting Condition	**Error** (pixels)		**Deviation** (pixels)	
	X	Y	X	Y
Incandescent	48.98	29.58	47.04	29.77
Fluorescent	44.98	29.83	44.75	29.01
None	49.60	32.24	44.61	29.53
Average	47.85	30.54	45.46	29.44

Table 1. Average accuracy under various ambient lighting conditions.

IV. Conclusion

The throughput and accuracy obtained are sufficient to provide many forms of human computer interface. As an example, a 19 inch monitor may be split up into 35 regions with the system being able to determine reliably which region the user is gazing at. One can imagine these regions representing buttons or other interactive controls. Furthermore, a throughput of 8 gaze points per second allows for timely detection of sustained user fixations.

The system described in this paper demonstrates the feasibility of a low-cost and sufficiently accurate method of eye tracking for a human computer interface. This was accomplished at a cost of around $130US, not including a standard desktop PC.

Expanding on this system, there are several changes which would greatly improve functionality and performance. To allow for tracking of fast eye movements, the throughput would need to be increased. If a camera that has hardware triggering capabilities was used, images could be captured at an increased frame rate, while maintaining synchronization with the lighting. Also, the software overhead could be reduced by implementing the algorithms in a lower level language such as C or C++. Finally the functionality of the system could be extended beyond a demonstration level by creating useful applications such as a web browser, or word processor.

References

[1] A. Duchowski, *Eye Tracking Methodology: Theory and Practice*. London: Springer, 2003.

[2] S. T. Moore, T. Haslwanter, I. S. Curthoys, and S. T. Smith, “A geometric basis for measurement of three-dimensional eye position using image processing,” *Vision Research*, vol. 36, pp. 445-459, 1996.

[3] Y. Ebisawa, “Improved Video-Based Eye-Gaze Detection Method,” *IEEE Trans. Instrumentation and Measurement*, vol. 47, no. 4, pp. 948–955, August 1998.

[4] D. H. Yoo, J. H. Kim, B. R. Lee, and M. J. Chung, “Non-contact Eye Gaze Tracking System by Mapping Corneal Reflections,” in *Proc. IEEE FGR ’02*, 2002.