OMG! - A new robust, wearable and affordable Open source Mobile Gaze tracker

Kristian Lukander **Sharman Jagadeesan** **Huageng Chi** **Kiti Müller**

Brain Work Research Centre, The Finnish Institute of Occupational Health
Topeliuksenkatu 41 a A, FIN-00250 Helsinki, Finland
corresponding author: kristian.lukander@ttl.fi

ABSTRACT
We present a novel, robust, affordable and wearable, mobile gaze tracker. The tracker takes a model-based approach to tracking gaze and maps the calculated gaze on to a scene video. The system is built from standard off-the-shelf components, and is the first to our knowledge using a 3D printed frame. The system will be published as open source, and the total cost of the components for building the system is 350€. The model-based tracking provides a solution robust to changing lighting conditions and frame slippage on the head of the user.

Author Keywords
gaze tracking; eye tracking; mobile; wearable

ACM Classification Keywords
H.5.1.[User interfaces]:Gaze interfaces; H.5.2[Input devices and strategies]; I.4.9.[Image Processing Applications]

General Terms
Human Factors; Measurement

INTRODUCTION

While gaze tracking studies have a long history within controlled laboratory environments and tracking gaze on fixed computer screens delivering stimuli, current trends of HCI research demand for a more mobile approach applicable to field studies, tracking gaze with mobile devices and in actual operational environments. For reviews on eye tracking development, see e.g. Hayehoe and Ballard's work [9]. Hansen [8] presents a survey of models for tracking gaze and eye features and Duchowski [3] delivers a broad survey of eye tracking applications. Evans et al. [5], focus on outdoor gaze tracking and some of the complexities inherent in taking gaze tracking out of the lab.

Constructing a wearable gaze tracker poses three fundamental problems: a) the sensitivity to changing lighting making the detection of tracked eye features challenging, b) the error introduced by frame slippage in relation to the head/eye of the user, and c) the typical complexity of user calibration.

Earlier open source gaze tracker systems include the openEyes system [12] that introduced the popular Starburst algorithm for detecting eye features; the ITU Gaze Tracker system [18] that aims at providing a low-cost alternative to commercial gaze trackers; the Haytham [15], developed more toward direct gaze interaction in real environments; and the wearable system by Ryan et al. [17] aiming at operating under visual light conditions.

We have developed a novel wearable gaze tracking system, taking a model-based approach to localizing the eye and the gaze vector in 3D, making the system considerably robust against slippage of the headgear and requiring only simple user calibration. The system is built from affordable off-the-shelf components, a custom printed circuit board (PCB) with standard electrical components, and as a novel development, our system is the first to our knowledge that uses a 3D-printed frame for the headgear. The tracking and calibration software, and the frame and PCB designs will be released as open source under a GNU General Public License. Apart from a standard laptop computer for running the software, the total cost for the system is about 350€.

Here we explain the design rationale for the system, introduce the operational principles, the geometry and the hardware, describe the modular software design, and present the first tracking results.

OBJECTIVES AND DESIGN RATIONALE

Typical barriers standing in the way of more wide-spread use of gaze tracking include the intrusive nature of tracking devices, complex issues with robustness and calibration under varying operating conditions, price and availability, and limited use in natural environments. Proprietary solutions also somewhat limit access to the measurement principles and technical solutions.

Hansen [8] lists a number of features for designing future gaze trackers, such as *development of head mounted trackers* for accurate and flexible measurements in natural surroundings, *flexible setups* for adaptable tracker designs, *limiting calibration* to simplify experimental setups and add robustness, and *decreasing the cost* of the systems. We would suggest extending the list with *modular designs and*

MobileHCI 2013, Aug 27–30, 2013, Munich, Germany.

algorithms for supporting quick iteration, and modifiability, the *real-time, online tracking and visualization* of gaze for controlling signal quality, and *minimizing obtrusiveness while maximizing the wearability of the trackers*.

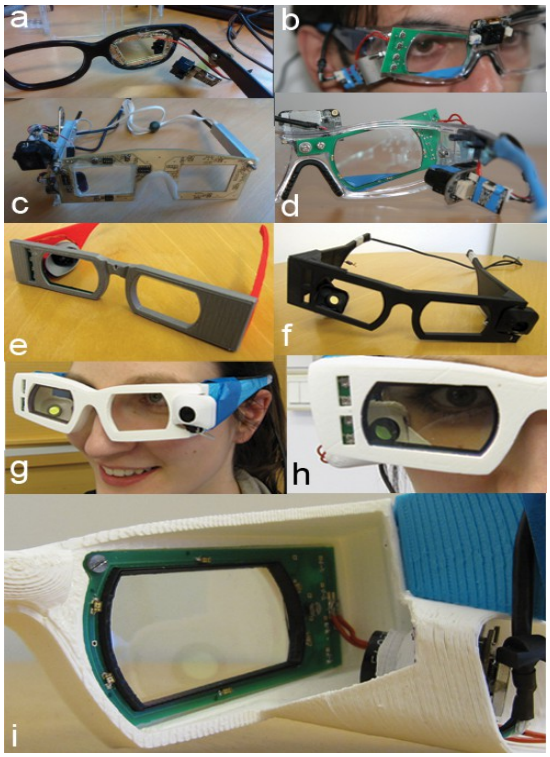

Figure 1. Various iterations of the headgear. (a) geometry tests with visible light; (b,d) system mounted on modified protective goggles; (c) 1st functional system; (e,f) early 3D printed designs; (g,h,i) current 3D printed frame.

SYSTEM OVERVIEW

Building a tracker is an open-ended problem, and design choices affect one another. Here we report our efforts to the extent they illuminate the design choices we have made.

We considered using a camera in front and below of the eye (boom-attached, or frame-attached with wide optics), using fiber-bundle optics, and using a miniature PAL camera. To get a wide enough image to cover the eye in various viewing angles, a camera should be positioned several cm away from the eye, making it hard to protect the image from variable lighting conditions, and creating a visible object in the field-of-view of the user. Mounting the camera closer would require a wide FOV (fisheye) lens, creating large distortions in the image. Fiber bundle optics turned out expensive, and providing limited resolution. Very small PAL cameras available did not turn out to provide adequate image quality for tracking eye features. Ultimately, we decided on using a mirrored USB camera with a small footprint, wide enough optics, good image quality and sensitivity in low-light conditions, positioned next to the cheek of the user, viewing the eye via an infrared (IR) mirror. The camera was modified by removing the IR filter from within the optical arrangement, and adding a visible light filter to limit the camera to IR wavelengths. We modified the USB camera driver to allow simultaneous full frame streaming from two USB cameras.

We investigated using visible and polarized light, but decided to use a controlled lighting solution with IR LEDs. We decided to use a combination of a bandpass filter in front of the camera for excluding visible light, and a hot mirror for allowing the subject to see through the lens with visual light wavelengths, while mirroring the IR range for the camera view. Due to the employed model-based tracking method (see below), the eye is illuminated with six infrared LEDs, one above and below, and two on either side of the lens. The PCB for positioning and powering the LEDs draws its +5V power from the USB lead of the eye camera.

We iterated through various constructions for the headgear (see figure 1), including mounting the system components on modified protective glasses, and using a custom PCB for structural support. Table 1 summarizes the selected components and their approximate prices. The system runs on a standard laptop (Dell E6320, Intel Core i5) running Linux Mint v.13. Toward the end of the project, consumer grade 3D printers became readily available, allowing unbeatable flexibility in the design, and implementation of the head frame. The frame was modeled in Rhinoceros 4.0 [16], and printed with a Makerbot Replicator [14] using PLA (rigid front part) and ABS (flexible rims) plastic.

The software is written in C++, utilizing a number of open source components: images are processed with the OpenCV library[2], the position of the corneal sphere is estimated using functions from GNU Scientific Library [6], the vector and matrix calculations make use of the Eigen library [4], and the optional TLD/Predator algorithm uses a modified version of the OpenTLD library [11]. The implementation of the Starburst algorithm is a modified version based on the openEyes project [12].

Table 1. approximate component pricing

component	part	price
eye/scene camera	Microsoft Lifecam HD-6000	2 x 60€
hot mirror	Thor Labs FM201 0°	60€
visible light filter	Thor Labs FL05850-10	40€
infrared LEDs	Vishay VSMY 1850 850nm	6 x 0,70€
3D frame	Variable, dependent on printer	50€
printed circuit board	Variable, dependent on service	70€
	Estimated total cost	**350€**

GAZE TRACKING PRINCIPLE

The gaze tracking principle is based on the 3D model-based method originally presented by Hennessey et al. [10], and developed further for constructing a wearable tracker. The system relies heavily on a strong prior model (hardware)

calibration to allow for a lighter user calibration.

The cameras are calibrated according to standard OpenCV calibration [2]. The 3D corneal fitting relies on accurate location of the light sources, and a custom calibration method was developed for defining the LED positions in relation to a camera using the reflections of the light sources through a first-surface mirror placed in the camera view. The reflections are used for collecting multiple aim vectors, whose intersection defines the positions of the light sources. For mapping the gaze vector to the scene video, we calibrate the relation between the eye and the scene camera using a calibration frame with calibration targets for both cameras set at a known distance and orientation.

All calibration procedures above are rig-specific and do not change between users. For each user, the system can adjust three parameters for accurate gaze estimation: the radius of the corneal curvature, the distance between the pupil and the cornea surface, and the kappa angle, i.e. the individual angle between the optical and the foveal axis of the eye (for population averages, see [7]). The corneal radius can be obtained e.g. with a keratometer. The kappa angle can be measured with a one-point calibration for each user.

Figure 3 presents the operation pipeline of the gaze tracker. An image is grabbed from both cameras. The Starburst algorithm is run to find the initial estimate for the pupil location, the area-of-interest (AOI), and the threshold level for the next phase. The resulting image is thresholded, clusterized, and small holes are removed. The best pupil candidates are selected from the clusters based on sanity checks and ellipticity, and the pupil edges for the selected candidate are further refined by double ellipse fitting.

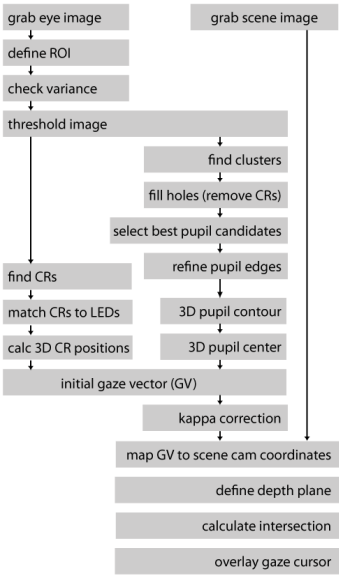

Figure 3. The pipeline for determining gaze direction and cursor

The thresholded image is then processed for identifying corneal reflecions (CRs). The identified CRs are matched to the LEDs and the 3D position of the corneal sphere is calculated based on the reflections as according to Hennessey [10]. The pupil contour is then defined in 3D, taking into account the refraction index of the eye, and the 3D position of the pupil center is then calculated. The initial gaze vector (corresponding to the optical axis of the eye) is then defined as the vector through the center of the corneal sphere and the center of the pupil, and is further modified with a simple kappa correction. The resulting gaze vector is mapped to the scene camera using calibration information. The intersection of the gaze vector and a given depth plane is calculated and the gaze cursor overlayed on scene video.

Figure 4. Sample frames from output video. On left: the captured eye image with the registered features - the pupil, the corneal reflections and the 3D gaze vector. On right: the gaze cursor overlayed on scene video.

SYSTEM PERFORMANCE

System performance was validated by measuring the accuracy of the calculated gaze coordinate from a setup where users looked at fixation crosses in a marker visible in the scene video. The marker based approach allows us to calculate viewing distance and to map the gaze vector in the marker coordinates to measure actual pixel distance in the scene camera image.

Table 2. Validation data from different viewing distances. The average disparity describes the offset of the gaze cursor

Viewing distance	Average disparity	stdev
100cm	2.47º	0.25º
150cm	1.67º	0.18º
200cm	1.25º	0.28º

The results are presented in table 2, and show that while the gaze cursor stays very steady (the standard deviation is small), there is a systematic offset (disparity of around 2º that will be studied in future work. It would seem that the

error gets smaller with increasing viewing distance suggesting that cause might be the parallax between the scene camera and the gaze vector.

The presented system performs relatively well in changing lighting conditions apart from direct sunlight. The model-based tracking algorithm allows the subject to even remove the frame during a measurement without losing calibration accuracy when the frame is put back on, as long as the eye camera has a good view of the eye.

The software runs in real-time, and tracks the gaze with the 30Hz sampling rate of the camera, making it easy to control the quality of the measurement during the test setups.

As the current version is monocular, it does not measure the length of the gaze vector, and the gaze position in the scene camera image has to be defined as the intersection between the gaze vector and a predefined depth plane. However, possible solutions exist: 1) A binocular version would allow using vergence information for defining gaze depth; 2) Optical markers in scene video allow localizing the gaze vector and its intersection within an environment; 3) The gaze depth can also be adjusted in the analysis stage.

As localizing the corneal sphere is based on measuring the reflections of a number of LEDs on the corneal surface, the system is not compatible with eye glasses.

The use of off-the-shelf cameras sets inherent limits to the frame design, although this can be remedied in the future with the use of smaller cameras. The current frame version also limits the user's view like heavy-rimmed eye glasses.

CONCLUSIONS

We have presented a novel, robust and affordable gaze tracking system including the hardware and a modular software solution. The low cost of the device is expected to open new versatile possibilities for applying gaze tracking. The software and extra material will be made available at http://www.brainworklab.fi/gazetracker.

ACKNOWLEDGEMENTS

This project was supported by the Finnish Academy, grant number 132639.

REFERENCES

1. Babcock J.S., Pelz J.B. (2004) Building a lightweight eyetracking headgear. In Proc. 2004 Symp. Eye Tracking Res. & Appl., Rochester Inst. of Technol., San Antonio, TX: ACM, Mar. 2004, pp. 109–114.
2. Bradski G. (2000) Programmer's tool chest: The OpenCV library. Dr. Dobbs Journal, November 2000. Available online at http://opencv.willowgarage.com/
3. Duchowski A.T. (2007) Eye Tracking Methodology: Theory and Practice. Secaurus, NJ/New York: Springer-Verlag, 2007.
4. Eigen C++ Template Library. http://eigen.tuxfamily.org
5. Evans, K. M., Jacobs, R. A., Tarduno, J. A., Pelz, J. B. (2012). Collecting and analyzing mobile eye-tracking data in outdoor environments. Journal of Eye Movement Research, 5(2):6,1-19.
6. Galassi M. et al. GNU Scientific Library Reference Manual (3rd Ed.), ISBN 0954612078. Library available online at http://www.gnu.org/software/gsl/
7. Gross, H., Blechinger, F., Achtner, B. 2008. Handbook of Optical Systems, Volume 4, Survey of Optical Instruments. ISBN: 978-3-527-40380-6
8. Hansen, D.W.; Qiang Ji; , "In the Eye of the Beholder: A Survey of Models for Eyes and Gaze," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.32, no.3, pp.478-500, March 2010.
9. Hayhoe, M, Ballard, D. (1995) Eye movements in natural behavior. Trends in Cognitive Sciences, 9(4), pp. 188-194.
10. Hennessey, C., Noureddin, B., Lawrence, P. 2006. A single camera eye-gaze tracking system with free head motion. In Proc. 2006 symposium on Eye tracking research & applications (ETRA '06). ACM, New York, NY, USA, 87-94.
11. Kalal, Z., Matas, J., Mikolajczyk, K. 2010. P-N learning: Bootstrapping binary classifiers by structural constraints. Proc. CVPR 2010. pp.49-56
12. Li D., Babcock J., Parkhurst D.J. (2006) openEyes: A low-cost head mounted eye-tracking solution. in Proc. 2006 Symp. Eye Tracking Res. & Appl.. San Diego, CA: ACM, Mar. 2006, pp. 95–100.
13. Li D., Winfield D., Parkhurst D. J. (2005). Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. Proceedings of the IEEE Vision for Human-Computer Interaction Workshop at CVPR, 1-8.
14. Makerbot Inc., USA. http://www.makerbot.com
15. Mardanbegi D. (2013) Haytham open source eye tracker software. Available online (Feb 14, 2013) at http://www.itu.dk/research/eye/?page_id=19
16. Rhinoceros 4.0 software, http://www.rhino3d.com/
17. Ryan W.J., Duchowski A.T., Birchfield S.T. (2008) Limbus/pupil switching for wearable eye tracking under variable lighting conditions. in Proc. 2008 Symp. Eye Tracking Res. & Appl., Savannah, GA: ACM, Mar. 2008, pp. 61–64.
18. San Agustin, J., Skovsgaard, H., Mollenbach, E., Barret, M., Tall, M., Hansen, D. W., and Hansen, J. P. 2010. Evaluation of a low-cost open-source gaze tracker. In Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications (Austin, Texas, March 22 - 24, 2010). ETRA '10. ACM, New York, NY, 77-80.